PROCESSING DATA STREAMS

Andrew McGregor

A DISSERTATION

in

Computer and Information Science

Presented to the Faculties of the University of Pennsylvania in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

2007

Sampath Kannan
Supervisor of Dissertation

Rajeev Alur
Graduate Group Chairperson

Acknowledgements

I am fortunate to have learnt a lot from some great people during my time in graduate school. First and foremost, I would like to thank Sampath Kannan for being a fantastic advisor. Talking with Sampath was always a pleasure and I can only hope that I have picked up some of his great taste in choosing problems. I am also very grateful to Sasha Barg, Sudipto Guha, and Bruce Shepherd for additional mentoring over the last five years. Working with Sudipto was a lot of fun, especially when our opinions diverged and caffeine-fueled hours would be spent in a back-and-forth of conjectures and counter-examples until a perfectly-formed theorem would emerge! During four internships at DIMACS and Bell Labs, Sasha convinced me of the benefits of thinking geometrically about coding theory and Bruce taught me not to fear case-analysis in combinatorial optimization. All have been a tremendous source of advice and encouragement.

A big thank you to the wonderful people of the St. Andrew's Society of the State of New York for the scholarship that partially funded my first year at Penn.

I would like to thank Sudipto Guha, Piotr Indyk, Michael Kearns, and Sanjeev Khanna for agreeing to be on my thesis committee. Thanks for your time and suggestions, and a special thanks to Piotr for waking up at 4 a.m. to make my defense after a day of travel upsets. Also, no thesis would ever be completed in Penn C.I.S. without the help of the indomitable Mike Felker, the department's paper-work tsar. Thank you for making the administrative side of things run so smoothly.

I am a firm believer that the best way to grow as a researcher is to collaborate with smart, enthusiastic people. I am fortunate to have had the opportunity to work with co-authors that fit that bill: Deepak, Stan, Sasha, Tuğkan, Amit, Matthew, Graham, Joan, Peter, Boulos, Piotr, Sampath, Sanjeev, Keshav, Eduardo, Muthu, Jeff, Bruce, Sid, Suresh, Jian, and Zhengyuan. Equally important to me has been the theory students at Penn: Stan, Milan, Yael, Boulos, Niel, Kuku, Sid, Mirko, and the ill-fated theory fish, Turing. Keep watering the plants and take care of Gollum!

Research has its moments of both frustration and utter elation. It is also addictive to the point of requiring "interventions" once in a while. For these I'd like to thank the "normal" people in my life, the people who distinguish between offices and coffee shops (thanks to Intermezzo and Capriccio) and who remain unconvinced that "Let $\epsilon < 0$" is the world's funniest joke. To old friends from the UK who dragged me to far-flung destinations or came out to visit (including my awesome sister), thanks for ensuring that America became a second home, rather than just a new home. (For specific instances of having a roof over my head, I would like to thank Ewan and Rachel, Simon and Rashmin, the Barg family, the Fraser family, Nick, Helen, David and Laura, Graham and Valentine, and the inhabitants of the inimitable 19 Frank's Lane. Thanks for letting me cover your kitchen tables with papers and I promise to replenish your coffee jars at the earliest opportunity!) To newer friends in the States with whom I explored the bizarre world that is graduate school: Stan, Vlad, Monica, Ilinca, Ryan, Tania, Anne, Kilian, Ameesh, Sasha, Niel, Nikhil (thanks for printing this thing!), Nick, Hanna, Sloop, and especially Christie, I owe you all a lot. Thank you.

This thesis is dedicated to my mum and dad, Hilary and Robert. Thank you for the belief you have in me and for not objecting *too* much when I decided study in the States. I never doubted that I would be finishing this thesis and I have you both to thank for this.

ABSTRACT

PROCESSING DATA STREAMS

Andrew McGregor

Sampath Kannan

Data streams are ubiquitous. Examples include the network traffic flowing past a router, data generated by an SQL query, readings taken in a sensor network, and files read from an external memory device. The data-stream model abstracts the main algorithmic constraints when processing such data: sequential access to data, limited time to process each data item, and limited working memory. The challenge of designing efficient algorithms in this model has enjoyed significant attention over the last ten years.

In this thesis we investigate two new directions of data-streams research: *stochastic streams* where data-streams are considered through a statistical or learning-theoretic lens and *graph streams* where highly-structured data needs to be processed. Much of the work in this thesis is motivated by the following general questions.

- *Stream Ordering*: Almost all prior work has considered streams that are assumed to be adversarially ordered. What happens if we relax this assumption?
- *Multiple Passes*: Previous work has focused on the single-pass model. What trade-offs arise if algorithms are permitted multiple passes over the stream.
- *Space-Efficient Sampling*: Rather than approximating functions of the empirical data in the stream, what can be learned about the source of the stream?
- *Sketchability*: A fundamental problem in the data-stream model is to compare two streams. What notions of difference can be estimated in small space?

In the process of considering these questions, we present algorithms and lower-bounds for a range of problems including estimating quantiles, various forms of entropy, information divergences, graph-diameter and girth; learning histograms and piecewise-linear probability density functions; and constructing graph-matchings.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

This thesis we will be concerned with the challenges of processing massive amounts of data. As the amount of information to be processed grows, we have had to reassess how we access data. The principal assumption of the standard *random access model* is that the data can be accessed in an arbitrary order and that each data access has unit cost. In many applications these assumptions do not hold. Rather, the data arrives as a stream of data elements and the order of this stream is predetermined. We now discuss three such scenarios.

As the internet becomes integral to our daily lives, it is increasingly important to be able ensure the smooth running of the network. To this end, it is necessary to carefully monitor network traffic. This includes intrusion detection, identification of “zombie” machines on the network, as well as computing statistics about the traffic that can be used to tweak network hardware and software configurations. Typically IP packets are monitored as they flow through the routers of the network. Clearly, in such a scenario, it is impossible to read the packets in any order other than the order in which they arrive. Furthermore, as each router may be responsible for forwarding up to a billion packets per hour, each packet must be processed quickly and only a vanishingly small fraction of the information presented can be stored.

The need to be able to process streams also arises in the context of databases. A typical problem is *query planning*, where a complicated query is made to a large database. This complicated query may be decomposed in numerous ways into a sequence of simpler queries but different decompositions may differ in efficiency. Unfortunately, the efficiency of a query plan is dependent on the data being queried and determining the most efficient query plan can be more time consuming than simply following a less efficient query plan! However, statistical summaries of the data contained in the database can aid this process significantly. The goal of research in this area is to compute these summaries based on updates to the database and the output streams of, for example, SQL queries.

In other applications we may have large unstructured data files such as search-engine log-files or biological data. As these files are too large to reside in the main memory of a computer, they need to be stored further down the memory-hierarchy, typically utilizing external devices such as massive hard-drives, optical media, and, even today, magnetic media. The problem that arises is that access to data on such devices can be very slow if the data is accessed in an arbitrary order. Specifically, while such devices have reasonable transfer rates, i.e., data can be transferred sequentially at speed, the seek times of these devices is often prohibitively large. Hence, being able to process the data in these files as a stream of sequential accesses can lead to large increases in efficiency.

The need to process data streams arises in many other application areas such as processing in sensor networks, analyzing scientific data arising in meteorology and other fields, monitoring financial transactions, etc. The interested reader is directed to a survey by Muthukrisnan [Mut06]. Further details about database applications may be found in a survey by Babcock et al. [BBD$^+$02].

While this thesis will concentrate on the theoretical challenges faced in the above applications, it would be remiss not to mention that there has been considerable effort made in the implementation of systems to process data streams. These are collectively known as *data stream management systems*. Examples include Brandeis/Brown/MIT's Aurora system [ACÇ+03], University of Wisconsin's Niagara system [CDTW00], Stanford's STREAM system [ABB+03], Berkeley's Telegraph system [CCD+03, KCC+03] and AT&T's Gigascope system [CJSS03].

In the next two sections we will formally define the data-stream model and then review some of the research that has been done in the area.

1.1.1 The Data-Stream Model

The data-stream model evolved primarily over the course of three papers [HRR99, AMS99, FKSV02a] although there are numerous older papers that foreshadowed that work [Mor78, MP80, FM85]. In this section we describe the core characteristics of the model.

A data stream consists of a long sequence of data items, $A = \langle a_1, a_2, \ldots, a_m \rangle$ where each data item a_i comes from some large universe. Depending on the application the universe could be numerical values in some range, points in some geometric space, edges in a graph, etc. The computational challenge is to compute some function of the data stream subject to the following restrictions:

Sequential Access: Data items may only be accessed in the order $a_1, a_2, \ldots, a_m$. However, the algorithm may be allowed to inspect the stream numerous times, each time accessing the data items in this sequential manner. Each such inspection is called a *pass* over the stream and there is a limit to the number of passes that the algorithm may make.

Fast Per-Item Processing: There is a limit on the time that may be spent processing each data item.

Small Workspace: The algorithm may only use a limited amount of space. This will always be sub-linear in the length of the stream and the universe size but will typically be considerably smaller.

In different settings the limits on these three resources vary. Specifically, in network monitoring applications, any algorithm will be limited to a single pass over the data, whereas, in external memory applications, multiple passes may be feasible. Ideally, algorithms would only use space that is poly-logarithmic in the length of the stream and universe size but for some problems such algorithms provably do not exist and a less stringent space restriction is applied.

One important facet of the model is how the data items are ordered. Sometimes the ordering of the data items will be relevant to the function being computed. This is the case with time-series data where, for example, the t-th element of the stream may represent a sensor reading taken at time t and the problem could be to fit a histogram to this data. Alternatively, it may be desirable that the function being computed is more sensitive to data that has recently appeared in the stream; an extreme being the sliding-window model in which the function is only evaluated on the most recent w items in the stream. However, in many situations the function we are trying to compute is invariant under permutation of the data items. For example, consider computing the median of a set of integers. Clearly, for any permutation σ,

$$\text{median}(\langle a_1, a_2, \ldots, a_m \rangle) = \text{median}(\langle a_{\sigma(1)}, a_{\sigma(2)}, \ldots, a_{\sigma(m)} \rangle) \ .$$

For such functions it is natural to attempt to design algorithms that correctly compute the relevant function even when the ordering is chosen adversarially. However, this is not always possible. One of the main contributions of this thesis is a study of the complexity of processing streams that are ordered randomly, i.e., the ordering of the m data items is chosen uniformly at random from the $m!$ possible ordering. The idea of random ordering was first proposed by Munro and Paterson [MP80] but received very limited subsequent treatment [DLOM02] prior to the appearance of the work presented here [GMV06, GM06, GM07b, GM07c].

Lastly, we mention that there have been models proposed that extend the data-stream model by allowing the algorithm to write to the stream during each pass

[ADRR04, DFR06]. These *annotations* can then be utilized by the algorithm during successive passes. The authors of [ADRR04] go further and suggest a model in which *sorting passes* are permitted in which the data stream is sorted according to a key encoded by the annotations. We do not consider these augmentations of the model.

1.1.2 Overview of Research in Data Streams

In recent years, the design of algorithms for the data-stream model has become an active area of research in numerous communities including theoretical computer science, databases and networking. In what follows, we give a brief overview of some of the work that has been done concerning algorithmic issues in processing data streams. In subsequent sections and chapters, we shall go into further detail about some of the work that is directly relevant to the results presented in this thesis. While we will restrict the survey to focus primarily on the theoretical work that has been done, we do not pretend that what follows is an exhaustive list. In particular, there is a lot of work on sampling based algorithms that can be used in the data-stream model. Again, the reader is directed to surveys [Mut06] and [BBD$^+$02] for further background.

Quantiles and Frequent Items: Many papers have considered estimating relatively "simple" statistics when the data items are numerical values. The problem of estimating the median of these values, or more generally, the quantiles has enjoyed significant attention particularly in the database community [MRL98, MRL99, GK01, GKMS02, SBAS04, GM06, GM07b]. Estimating biased quantiles, e.g., the 99-th or 99.9-th percentile, has also been considered [GZ03, CKMS06, GM06]. We will discuss this work in more detail in Chapter 3 in the context of studying random-order streams. Appropriately enough, sorting and selection were the subject of one of the first streaming papers [MP80]. Another natural problem is to find frequent items or *heavy hitters*. This is related to finding quantiles since the set of items with relative frequency at least ϵ is a subset of the set of ϵ-quantiles. There are both classical algorithms for the problem of finding the frequency of the most frequent element(s), e.g., [MG82, FS82] as well as more recent results [DLOM02, KSP03, MAA05]. The Count-Min Sketch [CM05a] and Count Sketch [CCFC02] constructions can be used for a variety of problems including quantiles and point-queries.

Norms and Distances: Another area that has been extensively studied is the approximation of ℓ_p norms, or *frequency moments*, [AMS99, Woo04, IW05, BGKS06]. In particular, work has been done on estimating F_0, the number of distinct items in a stream [BYJK$^+$02, IW03]. This problem was originally considered by Flajolet and Martin [FM85] in another of the "classic" streaming papers. Estimation of F_1, the length of the stream, using sub-logarithmic space was considered by Morris [Mor78]. F_∞ is the frequency of the most frequent item and is discussed above.

There has also been work done on estimating the ℓ_p distance between two streams [Ind00, FS01, FKSV02a, CG07a]. Given the importance of estimating distances between streams, other distance measures have been considered, including the Hamming distance [CDIM03] and a thorough treatment of the estimation of information-theoretic distances that arise in machine learning and statistics contexts is given in [GIM07]. Intrinsically related to these information distances is the entropy of a distribution. The estimation of entropy has been considered in a sequence of papers [CDM06, GMV06, LSO$^+$06, BG06, CCM07]. We will discuss this work in more detail in Chapter 7.

Succinct Representation: The problems considered so far have been of estimating a function of the data stream. We now turn our attention to problems in which the algorithm seeks to construct a small-space representation of the data. Examples include the construction of approximate histograms and wavelet decompositions

[GKMS01, GGI$^+$02b, GGI$^+$02a, GIMS02, CGL$^+$05, GKS06, GH06]. A slightly different problem is to learn a probability density function from independent samples given that the probability density function is k-piecewise constant. This was considered in [CK06, GM07c]. Various clustering problems have also been considered, including k-center [CCFM04] and k-median [COP03, GMMO00]. Problems related to finding succinct representation of matrices have been tackled. These are mainly based on sampling rows and columns, an idea first explored in [FKV04] where the goal was to compute the best low-rank approximation of a matrix. A multiple-pass algorithm was given by [DRVW06]. Other papers use similar ideas to compute a singular-value decomposition of a matrix [DFK$^+$04], approximate matrix multiplication [DKM04a], succinct representations [DKM04b] and approximate linear programming [DKM04c].

Geometry and Graphs: Geometric problems have been considered where the stream consists of points in (possibly high-dimensional) Euclidean space. Problems include finding minimum enclosing balls [AHPV04, ZC06], estimating the diameter of a set of points [FKZ04, AHPV04, CS06], computing convex hulls [HS03, CC05, CS06, GM07b], constructing core-sets [FS05, AY07, Cha06], line simplification [AdBHZ07], and finding areas of high discrepancy [AMP$^+$06].

There have also been attempts to solve graph problems in the data-stream model. Here, the elements of the stream are the edges of the graph. Typically these edges may arrive in an arbitrary order but sometimes it is assumed that all the edges adjacent to the same node arrive together. Problems studied have included counting triangles [BYKS02, JG05, BFL$^+$06], distance estimation [FKM$^+$05b, FKM$^+$05a, EZ06], constructing large matchings [FKM$^+$05b, McG05], estimating connectivity [FKM$^+$05b, Zel06], and computing the frequency and entropy moments of the degrees in a multi-graph [CM05b, CCM07]. A couple of papers considering an extension of the streaming model also consider graph problems [ADRR04, DFR06]. We will discuss the work on graph problems in further detail in Chapter 10.

Sequences: Rather than considering only the set of data items in a stream, in some settings the stream naturally defines a string or list, i.e., the i-th element of the stream is the i-th element of the string. In these cases, interesting problems include counting inversions [GZ03, AJKS02], computing edit-distance [CMS01] and estimating the length of the longest increasing subsequence [LNVZ06, SW07, GJKK07, GM07a, GG07]. Recent work on checking the correctness of priority queues [CKM07] is very related to these string problems.

Probabilistic Data: Very recently, researchers have considered the *probabilistic-stream model* where the stream elements encode distributions over a universe $[n] \cup \{\perp\}$ where $\perp$ represents a null element [JKV07]. Such a stream naturally defines a distribution over "deterministic" streams with elements from $[n]$. The existing research has focused on designing algorithms that estimate the expected value of various aggregate functions such as mean, median, and frequency moments [JMMV07, CG07b].

1.2 Our Contributions

The primary goal of this thesis is to investigate general issues in the data-stream model. While it is hoped that the specific problems addressed are of some inherent interest, the primary motivation behind the problems considered is to elucidate more fundamental aspects of the data-stream model. Such issues include the role played by the ordering of a stream and whether there is a substantial difference between average-case ordering and worst-case ordering; what trade-offs arise in the context of multi-pass algorithms; characterizing the notions of "stream-difference" that can be approximated in small space; and identifying the issues that arise when trying to make inferences about the source of the stream rather than the empirical data contained in the stream. We expand on these issues in Section 1.2.1.

The work contained in this thesis also formed the basis for two new directions of data-streams research: *stochastic streams* and *graph streams.*

- *Stochastic Streams:* This involves looking at data-stream problems through a statistical or learning-theoretic lens. For example, consider a stream formed by taking a sequence of independent sample from an unknown distribution. While previous algorithms for ℓ_p distances and frequency moments etc. are applicable in such a scenario, taking a more statistical view of the data motivates us to look at other problems such as comparing distributions using information divergences and estimating the entropy of a distribution. In addition, the question of stream-order naturally arises because of the exchangeability property of a sequence of independent samples. Another example of a stochastic stream could be a stream generated by an evolving k-th order Markov chain. This might be an appropriate model when the stream consists of text.

- *Graph Streams:* In this case the data items in the stream are edges, possibly weighted or directed, and we wish to compute properties of the graph defined by the set of all edges. Such a graph could be the *web-graph* where edges represent hyper-links between web-pages. Alternatively it could be a *call-graph* where edges represent telephone calls made between different phone numbers. In contrast to the majority of previous research that has considered estimating aggregate properties of a stream of numerical values, consideration of graph streams gives rise to a rich set of problems with which to explore some of the model-theoretic questions outlined above.

In Section 1.2.2 we give an overview of our results in both areas.

1.2.1 Questions and Themes

Does it matter how streams are ordered? Many functions that we wish to compute on a stream are invariant under permutation of the stream. Therefore, the ordering of the stream is only important in that it may have a bearing on the resources required to compute the function. In the literature to date, it is usually assumed that the stream is ordered by an adversary who may know the algorithm being used but is unaware of the coin tosses made by the algorithm. A natural complexity question is to quantify the powerful of such an adversary, i.e., how much more resources are required to process a stream that is adversarially ordered rather than one that is randomly ordered? Alternatively, if we impose certain computational constraints on the adversary, a popular idea in cryptography, how does this affect the resources required to compute in the model?

However, our motivation for investigating the effect of order is not purely theoretical. It is impossible to solve many important problems in small space when the stream is ordered adversarially. However, it is useful to know if a problem can be solved under certain conditions, or, in particular, if it is solvable "on average." The assumption that all possible orderings of the data stream are equiprobable gives rise to a natural notion of average that does not require any assumptions about actual set of data items in the stream.

There are also numerous situations in which it is reasonable to assume the stream is not ordered adversarially. The semantics of the data may lead us to believe that the stream is randomly ordered. Alternatively, there are situations in which the order of the stream can be explicitly randomized. We discuss some of these scenarios in the context of database streams in Chapter 3.

The random-order model was first considered by Munro and Paterson [MP80] but, prior to our work, has received little attention. Demaine, López-Ortiz, and Munro [DLOM02] considered the frequency estimation of internet packet streams assuming that the packets arrived in random order. Kannan [Kan01] conjectured that any problem that could be solved in $P/2$ passes of a randomly ordered stream could be solved in at most P passes of an adversarially ordered stream.

What can we learn about the source of a stream? As mentioned earlier, a natural setting in which a data stream would be ordered randomly is if each element of the stream is a sample drawn independently from some unknown distribution. Clearly, no matter what the source distribution, given the m samples taken, each of the $m!$ permutations of the sequence of samples were equally likely. However, if the stream is generated in this way, another important question arises: can we learn something about the source of the samples?

Trying to make such inferences is the problem that underlies much of statistics and machine learning. A fundamental quantity of interest is the *sample complexity* of the property being inferred, i.e., the number of samples that are necessary to learn the property with high probability. Sample complexity is inherently interesting from a statistical point of view. From a computational view point it is also clearly relevant since it is a trivial lower bound on the *time-complexity* of any algorithm that can make a certain inference. However, there is another component of the computational complexity, that of *space-complexity*, i.e., how much memory is required by the learning algorithm. In many applications space complexity may be more important than time complexity. For example, embedded devices, sensor networks or network routers may be required to modify their behavior on the basis of what is learned about the underlying distribution of the data being processed.

Unlike the time complexity, space complexity is not necessarily as large as sample complexity. In particular, if we are presented with a stream of samples, sometimes we make the desired inference without storing all the samples. This is clearly the case when trying to estimate the mean of a distribution. But what about estimating more complicated properties? In particular, could we learn the probability density function of a distribution? Techniques developed for processing data streams could be of great use when solving these types of problems.

To date, almost of all of the streaming algorithms proposed have considered computing empirical quantities determined by the data items rather than trying to make some inference about the source of the stream. One notable exception is the recent work by Chang and Kannan [CK06] that seeks to learn the piecewise-linear probability density function of a stream of samples. An issue that arises in that work is that the number of samples required was substantially larger than the sample-complexity of the problem. Is this generally the case or is it possible to design algorithms that have small space-complexity and sample-complexity?

The problem of learning from independent samples can be generalized to a situation where an algorithm has access to an oracle that supports various queries about a distribution. Such models were introduced by Kearns et al. [KMR$^+$94]. In the *generative model* a `sample`(p) query returns i with probability p_i. In the *evaluative model*, a `probe`(p, i) query returns the value p_i. A natural third model, the *combined model* was introduced by Batu et al. [BDKR05]. In this model both the `sample` and `probe` operations are permissible. In all three models, the complexity of an algorithm is measured by the number of queries to the oracle.

Is it possible to relate these oracle models to the stream model? Accessing the data sequentially limits how adaptive any sampling procedure may be but the severity of this restriction may be ameliorated by permitting an algorithm multiple passes. We discuss this question further in Chapter 3 along with some relevant prior work by Feigenbaum et al. [FKSV02b].

How valuable is an extra pass? How does resource-use trade-off with the number of passes an algorithm may make over the stream? In particular, can we design algorithms that use significantly less space if they are permitted more than a single pass over the data stream?

We just mentioned that relaxing the restriction on the number of passes, allows sampling-based algorithms to be more adaptive. More generally, the number of passes permitted can be viewed as establishing a spectrum between the single-pass data-stream model and the traditional random-access model. Does the computational power grow smoothly along this spectrum or is it ever the case that, given the same space restriction, allowing $P + 1$ passes rather than P passes makes a problem tractable that would otherwise be intractable. We consider this question through-out this thesis but primarily in Chapter 3 and Chapter 10.

What distances can be sketched? There has been considerable success in designing algorithms that quantify the "difference" between two streams. In this thesis we will be interested in characterizing the notions of difference that can be estimated in small space?

We primarily consider "update" streams where each data item is a value in $[n]$. A stream defines a frequency vector $(f_1, f_2, \ldots, f_n)$ where f_i is the number of times i occurs in the stream. This model is essentially a generalization of the *aggregate model*, in which all updates to a given i are grouped together in the ordering of the stream. We are interested in comparing the frequency vectors defined by different streams. Previous work has demonstrated algorithms that approximate distances based on ℓ_p norms [FKSV02a, Ind00, FS01, CDIM03] (note that the Hamming distances is closely related to the ℓ_0 norm.) Are there algorithms for other commonly used distance measures? Can we characterize the set of distances that can be approximated? Are there distances that can be approximated in the aggregate-model that can not be approximated in the update-model? We consider these questions in Chapter 7.

1.2.2 Summary of Results

In this section we give a brief overview of the specific results in this thesis. The algorithmic results in this thesis are summarized in Tables 1.1, 1.2, and 1.3. Lower-bounds are summarized in Table 1.4. We also comment on the relevance of these results to the broader questions outlined in the previous section. Further discussion along with detailed summaries of previous work can be found in Chapters 3, 7, and 10. The full technical details and proofs of the results can be found in the remaining chapters.

Random-Order Streams: We first explore issues of random-order streams using the computation of quantiles as an example. In addition to considering the problem of exact selection we also consider computing approximate quantiles using the following notion of approximation.

Definition 1.1 (Rank and Approximate Selection). *The rank of an item x in a set S is defined as,* $\textsc{Rank}_S(x) = |\{x' \in S | x' < x\}| + 1$. *We say x is an Υ-approximate rank k element in S if,* $\textsc{Rank}_S(x) = k \pm \Upsilon$.

We present a single pass algorithm using $O(1)$ words of space that, given any k, returns an $O(k^{1/2} \log^3 k)$-approximate k-rank element (with high probability) if the stream is in random-order. Our algorithm does not require prior knowledge of the length of the stream. In comparison, we show that an algorithm achieving similar precision when the stream is ordered adversarially would require polynomial space. This immediately demonstrates a separation between the random-order and adversarial-order stream models. Using new techniques we also show that any algorithm that returns an m^{δ}-approximate median of a randomly ordered stream with probability at least 3/4 requires $\Omega(\sqrt{m^{1-3\delta}/\log n})$ space.

We then address the conjecture of Kannan [Kan01] that stated that any problem that could be solved in $P/2$ passes of a randomly ordered stream could be solved in at most P passes of an adversarially ordered stream. We present an algorithm using $O(\operatorname{polylog} m)$ space that returns the exact median of a data stream in $O(\log \log m)$ passes. This was conjectured by Munro and Paterson [MP80] and has been unresolved since 1978. On the other hand we show that any algorithm that returns an m^{δ}-approximate median in P passes of an adversarially ordered stream requires $\Omega(m^{(1-\delta)/P} P^{-6})$ space. In particular, this shows that the conjecture is false.

Lastly, we explore connections between query complexity of problems in the oracle models. We show that, for a wide range of functions f, any combined oracle algorithm making k queries can be transformed into a two-pass algorithm in the adversarial-order streaming model that uses $\tilde{O}(k)$ space. If the stream is in random-order the resulting algorithm only requires one pass.

Space-Efficient Sampling: We initiate the study of *space-efficient sampling* and demonstrate how techniques for processing data streams can be used to design algorithms with small space complexity.

Our main algorithmic result is for learning the probability density function of one-dimensional distributions. We suppose we have access to independent samples from some distribution on the real line with probability density function D that has at most k piece-wise linear segments. We wish to find a probability density function $\hat{D}$ such that $\int_{\mathbb{R}} |D(x) - \hat{D}(x)| \leq \epsilon$. An almost identical problem was considered by Chang and Kannan [CK06]. The important difference is that Chang and Kannan were interested in designing algorithms that could process the samples in any order. Given that we are assuming the stream of samples are independent draws from the source distribution, it is natural to consider this problem in the random-order stream model. Furthermore, while Chang and Kannan assumed that the samples are stored locally and hence the samples can be read numerous times, we are primarily interested in the scenario in which samples are not stored. We present an algorithm for learning D up to ϵ-variational error. The algorithm uses $\tilde{O}(k^2\epsilon^{-4})$ samples and $\tilde{O}(k)$ space. The algorithm capitalizes on the fact that a sequence of independent samples will be in random-order. We also present an algorithm directly comparable to that in [CK06], i.e., the algorithm must process the samples in an arbitrary order but is permitted multiple passes over the stream of samples. Our algorithm takes P (assumed constant) passes over $\tilde{O}(k^2\epsilon^{-4})$ samples and uses $\tilde{O}(k\epsilon^{-2/P})$ space. In comparison, the algorithm of [CK06] takes P passes over $\tilde{O}(k^6\epsilon^{-6})$ samples and uses

$\tilde{O}(k^3\epsilon^{-4/P})$ space.

We also explore the problem of estimating frequency moments in this model. We show that it is possible to (ϵ, δ)-approximate F_k in a randomly ordered stream with $\tilde{O}_\epsilon((n/t)^{1-2/k})$ space when the stream length is $m = \Omega(nt\epsilon^{-3/k}\log n)$. Varying the value of t establishes a trade-off between sample-complexity and space-complexity. In particular, if $m = \Omega(n^2\epsilon^{-3/k}\log n)$ then the space only depends on n polylogarithmically. This provides another demonstration of importance of a stream being in random-order: if the stream were ordered adversarially, the same estimation would require $\Omega(n^{1-2/k})$ space.

Information Divergences and Entropy: As we mentioned above, previous work has demonstrated algorithms that approximate distances based on ℓ_p-norms [Ind00, FS01, FKSV02a, CDIM03]. Experience leads us to conjecture that the only distances that can be approximated in small space are those that are essentially based on norms. We present a partial result that we call the *Shift Invariant Theorem.* This result provides a very general set of conditions under which a distance measure cannot be approximated in small space.

Next we focus on a specific set of distance measures called *information divergences.* These quantify the difference between two distributions in such a way that captures aspects of dissimilarity that are important in statistics and machine learning applications. We consider two main families of information divergences called the *f-Divergences* and *Bregman Divergences.*

Definition 1.2 (f-Divergences and Bregman Divergences)**.** *Let p and q be two n-point distributions. A convex function $f : (0, \infty) \to \mathbb{R}$ such that $f(1) = 0$ gives rise to an f*-divergence*,*

$$\mathcal{D}_f(p, q) = \sum p_i f(q_i/p_i) \ .$$

Similarly, a strictly convex function $F : (0, 1) \to \mathbb{R}$ gives rise to (decomposable)

Bregman Divergence*,*

$$\mathcal{B}_F(p,q) = \sum_i \left[F(p_i) - F(q_i) - (p_i - q_i)F'(q)\right] \ ,$$

where F' denotes the first derivative of F.

Commonly used f-divergences include the ℓ_1 distance, the *Hellinger divergence*, the *Jensen-Shannon divergence* and the *Kullback-Liebler divergence*. The family of Bregman divergences also includes the Kullback-Liebler divergence along with the ℓ_2^2 distance and the *Itakura-Saito divergence*.

We use the Shift Invariant Theorem to characterize a large set of f-divergences and Bregman divergences that can not be multiplicatively approximated in small space. This characterization is successful in the sense that the only f-divergences and Bregman divergences that are not included are ℓ_1 and ℓ_2^2. These are exactly the f-divergences and Bregman divergences for which small-space multiplicative approximation algorithms are known.

Given the lower bounds on multiplicative approximation, we next consider finding additive approximations. We show that any bounded $\mathcal{D}_f$ can be (ϵ,δ)-additive-approximated using $O(\epsilon^{-2}\log\delta^{-1})$ space. For an unbounded $\mathcal{D}_f$ we show that any $(\epsilon, 1/4)$-additive-approximation requires $\Omega(n)$ space for any ϵ. For an bounded $\mathcal{D}_f$ we show that any $(\epsilon, 1/4)$-additive-approximation requires $\Omega(\epsilon^{-2})$ space. Similarly, a Bregman divergence $\mathcal{B}_F$ can be (ϵ,δ)-additive-approximated in $O(\epsilon^{-2}\log\delta^{-1})$ space if F and F'' are bounded. On the other hand if $F(0)$ or $F'(0)$ is unbounded, then any $(\epsilon, 1/4)$-additive-approximation of $\mathcal{B}_F$ requires $\Omega(n)$ space for any constant ϵ.

A fundamental quantity that is closely related to the information divergences is the *entropy* of a distribution. This quantity arises in coding and information theory, learning and statistics literature. It is defined as follows:

Definition 1.3 (Entropy)**.** *The* entropy *of distribution p is $H(p) = \sum_i -p_i \lg p_i$.*

The first attempt to approximate the entropy of a data stream appeared in [GMV06]. This result was improved upon by a sequence of papers [CDM06, LSO$^+$06, BG06]. In this thesis we present a single-pass, $O(\epsilon^{-2}\log(\delta^{-1})\log m)$-space, (ϵ,δ)-approximation algorithm for entropy. This improves upon the previous work. The algorithm uses a novel extension of a method introduced by Alon, Matias, and Szegedy [AMS99] that may have other applications. We show that our algorithm is essentially optimal by proving that any $(\epsilon, 1/4)$-approximation requires $\Omega(\epsilon^{-2}/\log^2(1/\epsilon))$ space. We then show that any $(\epsilon, 1/4)$-approximation of the k-th order entropy ($k > 0$) of a streams requires $\Omega(n/\log n)$ space where the k-th order entropy of a stream is a generalization of the entropy that quantifies how easy it is to predict a character of the stream given the previous k characters. However, we present an (ϵ,δ)-addtive-approximation using $O(k^2\epsilon^{-2}\log(\delta^{-1})\log^2 n\log^2 m)$ space. We also present an (ϵ,δ)-approximation algorithm for estimating the *unbiased random walk entropy*, a natural quantity related to the first order entropy of an unbiased walk on an undirected graph. Our algorithm uses $O(\epsilon^{-4}\log n\log\delta^{-1})$ space. This algorithm can also be implemented in the graph streaming model to be discussed shortly.

Lastly, we present testing algorithms for entropy and information divergences in the oracle models. These include an $(\epsilon, \epsilon/2, \delta)$-tester for all bounded f-Divergences in the combined-oracle model. We prove a matching lower bound thereby showing optimality. We also present a sub-linear tester in the generative model for a range of f-divergences including Hellinger and Jensen-Shannon. The algorithm makes $\tilde{O}(\epsilon^{-4}n^{2/3})$ queries. This answers an open question posed in [BFR$^+$00]. Finally, we present an $(\epsilon, \epsilon/2, \delta)$-tester for entropy in the combined oracle model using $O(\epsilon^{-2}\log n\log\delta^{-1})$ queries. This matches the lower bound in [BDKR05].

Graph Streaming: We now consider streams that define graphs. This will prove an ideal context to explore trade-offs that arise in multiple-pass streaming. This is also the first comprehensive treatment of solving problems on graph streams.

Definition 1.4 (Graph Stream). *For a data stream $A = \langle a_1, a_2, \ldots, a_m\rangle$, with each data item $a_j \in [n]\times[n]$, we define a graph G on n vertices $V = \{v_1, \ldots, v_n\}$ with* *edges* $E = \{(v_i, v_k) : a_j = (i,k) \text{ for some } j \in [m]\}$.

Note that n is no longer the universe size of the data items. We normally assume that each a_j is distinct although this assumption is often not necessary. When the data items are not distinct, the model can naturally be extended to consider multi-graphs, i.e., an edge (v_i, v_k) has multiplicity equal to $|\{j : a_j = (i,k)\}|$. Similarly, we mainly consider undirected graphs but the definition can be generalized to define directed graphs. Sometimes we will consider weighted graphs and in this case $a_j \in [n] \times [n] \times \mathbb{N}$ where the third component of the data item indicates a weight associated with the edge. Note that some authors have also considered a special case of the model, the adjacency-list model, in which all incident edges are grouped together in the stream [BYKS02]; we will be primarily interested in the fully general model.

We start with some negative results that exhibit various trade-offs between space, the number of passes, and accuracy of estimation. The lower-bounds we present complement the algorithms presented in [FKM+05a, Zha05].

We first show that many graph algorithms are inherently unsuitable when computing in the streaming model. In particular, for $\gamma \in (0,1)$ and $l \in [\lfloor 1/\gamma \rfloor]$, computing the first l layers of a breadth-first-search (BFS) tree from a prescribed node requires either $\lfloor (l-1)/2 \rfloor$ passes or $\Omega(n^{1+\gamma})$ space. On the other hand it will be trivial to construct the first l layers of a BFS tree with l passes even in space $O(n \log n)$. Constructing BFS trees is a very common sub-routine in many graph algorithms and this result shows that as we restrict the space, any algorithm for constructing a BFS essentially requires random access to the data.

We next show a trade-off between space and accuracy: any single pass algorithm that approximates the weighted graph distance between two given nodes up to a factor γ^{-1} with constant probability requires $\Omega(n^{1+\gamma})$ space. Furthermore, this bound also applies to estimating the diameter of the graph. Any P-pass algorithm that ascertains whether the length of the shortest cycle is longer than g, requires $\Omega\left(P^{-1}(n/g)^{1+1/(g-5)}\right)$ space. This trade-off between space and passes, in contrast

	Space	Passes	Order
Entropy:			
0-th order (ϵ, δ)-approx.	$O(\epsilon^{-2})$	1	Adversarial
k-th order (ϵ, δ)-additive-approx.	$O(k^2\epsilon^{-2})$	1	Adversarial
Random Walk (ϵ, δ)-approx.	$O(\epsilon^{-4})$	1	Adversarial
Information Distances:			
Bounded $\mathcal{D}_f$ (ϵ, δ)-additive-approx.	$O(\epsilon^{-2})$	1	Adversarial
Certain $\mathcal{B}_F$ (ϵ, δ)-additive-approx.	$O(\epsilon^{-2})$	1	Adversarial
Quantile Estimation:			
$O(\sqrt{m}\log^2 m \log \delta^{-1})$-approx	$O(1)$	1	Random
Exact Selection	$O(1)$	$O(\log\log m)$	Random
Graph Matchings:			
Unweighted $1/2$-approx	$O(n)$	1	Adversarial
Unweighted $(1-\epsilon)$-approx	$O(n)$	$O_\epsilon(1)$	Adversarial
Weighted 0.17-approx	$O(n)$	1	Adversarial
Weighted $(1/2-\epsilon)$-approx	$O(n)$	$O(\epsilon^{-1})$	Adversarial

Table 1.1: Streaming Algorithms. Log terms are omitted in the space column.

to some of the problems discussed above, is smooth and indicates that the only way to get away with using half the amount of space is essentially to make half as much progress in each pass. Lastly we show that testing any of a large class of graph properties, which we refer to as *balanced properties*, in one pass requires $\Omega(n)$ space. This class includes properties such as connectivity and bipartiteness.

We also present multiple pass algorithms for computing weighted and unweighted graph matchings. All these algorithms take linear time and use $O(n \operatorname{polylog} n)$ space. First we present a single pass, 1/2-approximation for maximum cardinality matchings. With successive passes it is possible to "grow" this matching such that, for any $\epsilon > 0$, with $O_\epsilon(1)$ passes it is possible to find a $(1-\epsilon)$-approximation for the maximum cardinality matching. For weighted matching we present a single pass, $1/(3+2\sqrt{2})$-approximation. Again, for any $\epsilon > 0$, using $O_\epsilon(1)$ passes this can be grown into a $(1/2-\epsilon)$-approximation algorithm.

	Sample	Space	Passes	Order
k-piecewise linear pdf	$O(k^2\epsilon^{-4})$	$O(k)$	1	Random
k-piecewise linear pdf	$O((1.25)^{P/2}\ell k^2\epsilon^{-4})$	$O(k\epsilon^{-2/P})$	P	Adversarial
k-th frequency moment	$O(nt\epsilon^{-3/k})$	$O_\epsilon((n/t)^{1-2/k})$	1	Random

Table 1.2: Learning Algorithms in the Data-Stream Model. Log terms are omitted.

	Samples	Probes
$(\epsilon, \epsilon/2, \delta)$-tester for Shannon Entropy	$O(\epsilon^{-1})$	$O(\epsilon^{-1})$
$(\epsilon, \epsilon^2 n^{-1/3}/32, \delta)$-tester for Hellinger, JS, and $\triangle$	$O(\epsilon^{-4}n^{2/3})$	0
$(\epsilon, \epsilon/2, \delta)$-tester for Bounded $\mathcal{D}_f$	$O(\epsilon^{-1})$	$O(\epsilon^{-1})$

Table 1.3: Oracle Algorithms. Log terms are omitted.

	Space	Passes	Order
Entropy:			
0-th order (Shannon)	$\Omega(\epsilon^{-2}/\log^2 \epsilon^{-1})$	1	Adversarial
k-th order	$\Omega(n/\log n)$	1	Adversarial
Information Distances:			
Unbounded $\mathcal{D}_f$	$\Omega(n/P)$	P	Adversarial
Bounded $\mathcal{D}_f$ (Additive)	$\Omega(\epsilon^{-2})$	1	Adversarial
Certain $\mathcal{B}_F$	$\Omega(n/P)$	P	Adversarial
Quantile Estimation:			
m^γ-approx	$\Omega(\sqrt{m^{1-3\gamma}/\log m})$	1	Random
m^γ-approx	$\Omega(m^{(1-\gamma)/P}P^{-6})$	P	Adversarial
Graphs Streaming:			
Depth $2(P+1)$-th BSF Tree	$\Omega(n^{1+1/(2P+2)})$	P	Adversarial
$1/\gamma$ approx. of Diameter	$\Omega(n^{1+\gamma})$	1	Adversarial
Testing if Girth is at most g	$\Omega((n/g)^{1+4/(3g-7)}/P)$	P	Adversarial
Testing Balanced Properties	$\Omega(n)$	1	Adversarial

Table 1.4: Lower-Bounds in the Data-Stream Model. Results are $(\epsilon, 99/100)$-approximations except when noted.

1.3 Organization

The main body of this thesis is split into three parts. The first contains the results pertaining to questions about stream-ordering and oracles. Chapter 4 contains the results on quantiles and random-order streams. Parts of this work appeared in Guha and McGregor [GM06, GM07b]. This is followed by a short model-theoretic result relating the oracle and stream models in Chapter 5. This work first appeared in Guha, McGregor, and Venkatasubramanian [GMV06]. The work on space-efficient sampling, in Chapter 6, first appeared in Guha and McGregor [GM07c].

The second part of the thesis looks at estimating entropy and information divergences in the stream model. More generally we ask which distances are sketchable. Chapter 8 concerns estimating Information Distances in the streaming model and first appeared in Guha, Indyk, and McGregor [GIM07]. In Chapter 9, we present the results on approximating entropy in the streaming model. This work first appeared in Chakrabarti, Cormode, and McGregor [CCM07]. Also included in Chapters 8 and 9 are oracle algorithms for estimating information distances and entropy and these originally appeared in Guha, McGregor, and Venkatasubramanian [GMV06]

In the third part of this thesis we address graph streams. The work in Chapters 11 and 12 are based on Feigenbaum et al. [FKM$^+$05b, FKM$^+$05a] and McGregor [McG05].

Chapter 2

Preliminaries

Chapter Outline: In this chapter we discuss some of the techniques that will be used throughout this thesis. This will include a discussion about various sampling methods and the relationship between communication complexity and computation in the stream model.

2.1 Communication Complexity

The study of communication complexity was initiated by Yao in 1979 [Yao79]. We first consider the basic model in which two separated parties, Alice and Bob, are trying to evaluate a function $f : X \times Y \to Z$ on the input (x, y). Unfortunately Alice only knows x and Bob only knows y. Consequently, if either party is to learn the value of $f(x, y)$, Alice and Bob may need to send messages to one another. The main goal of communication complexity is to quantify how "long" these messages must be if $f(x, y)$ is to be computed correctly. It is usually assumed that these messages are transmitted as binary strings and the length of the communication is simply the length of the string formed by concatenating all the messages.

There are many different natural variants of the basic model. The number of rounds of communication may be limited, i.e., we assume that each party communicates in turn and limit the number of times the parties alternate. A special case would be when Alice sends a single message to Bob upon which Bob is expected to be able to output the correct value of the function. Often we permit the protocol for communication to be randomized and tolerate some small probability that the recipient of the final message does not correctly evaluate function. Further variants and a comprehensive treatment of communication complexity can be found in [KN97].

We will be primarily interested in the *r-round randomized communication complexity* of a function. Formally, we consider a *communication protocol* Π which determines the content of each message given the messages already sent, random coin tosses, and the private information of the party whose turn it is to send the message. Note that at any stage, the set of messages that can be transmitted by a player must be prefix-free or otherwise there is a potential ambiguity over when a player has finished communicating. We define the cost of a protocol, $|\Pi|$, to be the maximum (over all coin tosses and inputs $(x, y) \in X \times Y$) number of bits communicated by the protocol. We say Π has δ probability of failure if the probability (over the random coin tosses) that the recipient of the final message cannot correctly compute $f(x, y)$ is at most $1 - \delta$. Then the r-round randomized communication complexity of f, denoted $R^r_\delta(f)$, is the minimum cost of a protocol with δ probability of failure. If we place no restriction on the number of rounds we write $R_\delta(f)$.

We will also be interested in the *r-round distributed communication complexity* of a function, $D^r_{\mu,\delta}(f)$. This is defined similarly to $R^r_\delta(f)$ except that we insist that the protocols are deterministic and the input for f is chosen from a distribution μ.

Intuitively it makes sense that communication complexity should be related to the stream model. For example, consider reading through a long sequence of values and trying to compute some function of these values. At any point, we have a certain memory of the values we have already read. This memory can be thought of as a communication from your former self to your future self. We formalize this intuition in the next section.

Reductions from Communication Complexity: Many lower-bound proofs in the literature proceed along the lines of the following template. Let (x, y) be an instance of some communication problem f. We suppose that there exists a streaming algorithm $\mathcal{A}$ making P passes over a stream and using W working memory whose output satisfies certain guarantees with probability at least $1 - \delta$. If we can show that $\mathcal{A}$ gives rise to a $(2P - 1)$-round communication protocol with cost $O(PW)$ then we deduce that $W = \Omega(R_\delta^{2P-1}(f)/P)$.

To construct such a reduction we would show how Alice and Bob can construct a set of stream elements $S_A(x)$ and $S_B(y)$ such that the "correct" value of $\mathcal{A}$ on the stream containing $S_A(x) \cup S_B(y)$ determines the value of $f(x, y)$. Alice and Bob can then emulate $\mathcal{A}$: Alice runs $\mathcal{A}$ on $S_A(x)$, communicates the memory state of $\mathcal{A}$, Bob runs $\mathcal{A}$ initiated with this memory state on $S_B(x)$ and communicates the memory state of $\mathcal{A}$ to Alice and so on. The resulting protocol has $(2P - 1)$-rounds and has cost $O(PW)$ as required.

This general methodology has been used extensively and was first used by Alon, Matias, and Szegedy [AMS99] and Henzinger, Raghavan, and Rajagopalan [HRR99]. It will be used throughout this thesis. Sometimes the reduction will be straightforward but more often than not, constructing suitable $S_A(x)$ and $S_B(x)$ will be distinctly non-trivial. In the next section we summarize some of the communication complexity results that we will use in this thesis.

Two-Party Communication Results:

- Set-Disjointness: Let Alice have $x \in \mathbb{F}_2^n$ and Bob have $y \in \mathbb{F}_2^n$ where $\|x\|_1 = \|y\|_1 = \lfloor n/4 \rfloor$. Then define,

$$\textsc{Set-Disjointness}(x, y) = \begin{cases} 1 & \text{if } x.y = 0 \\ 0 & \text{if } x.y \geq 1 \end{cases} .$$

It was shown by Kalyanasundaram and Schnitger [KS92] and Raz [Raz92] that,

$$R_{1/4}(\textsc{Set-Disjointness}) = \Omega(n) \ .$$

- $\textsc{Index}$: Let Alice have $x \in \mathbb{F}_2^n$ and Bob have $j \in [n]$. Then define,

$$\textsc{Index}(x, j) = x_j \ .$$

It can be shown that (e.g., [KN97]),

$$R^1_{1/4}(\textsc{Index}) = \Omega(n) \ .$$

- $\textsc{Gap-Hamdist}$: Let Alice have $x \in \mathbb{F}_2^n$ and Bob have $y \in \mathbb{F}_2^n$ such that either $\|x - y\|_1 \leq n/2$ or $\|x - y\|_1 \geq n/2 + \sqrt{n}$. Then define,

$$\textsc{Gap-Hamdist}(x, y) = \begin{cases} 1 & \text{if } \|x - y\|_1 \leq n/2 \\ 0 & \text{if } \|x - y\|_1 \geq n/2 + \sqrt{n} \end{cases} \ .$$

It was shown by Indyk and Woodruff [IW03, Woo04]) that,

$$R^1_{1/4}(\textsc{Gap-Hamdist}) = \Omega(n) \ .$$

- $\textsc{Prefix}$: Let Alice have $x \in \mathbb{F}_2^n$ and Bob have $y \in \mathbb{F}_2^n$ and $j \in [n]$. Then define,

$$\textsc{Prefix}(x, y, j) = \begin{cases} 1 & \text{if } \forall i \in [j], x_i = y_i \\ 0 & \text{otherwise} \end{cases} \ .$$

It was shown by Chakrabatri et al. [CCM07] (see Chapter 9) that,

$$R^1_{1/4}(\textsc{Prefix}) = \Omega(n/\log n) \ .$$

- Let Alice have $f_A \in F_m$ and Bob have $f_B \in F_m$ where F_m be the set of all functions from $[m]$ to $[m]$. Define k-$\textsc{pointer} : F_m \times F_m \to [m]$ by k-$\textsc{pointer}(f_A, f_B) = g_k(f_A, f_B)$ where $g_0(f_A, f_B) = 1$ and,

$$g_i(f_A, f_B) = \begin{cases} f_A(g_{i-1}(f_A, f_B)) & \text{if } i \text{ even} \\ f_B(g_{i-1}(f_A, f_B)) & \text{if } i \text{ odd} \end{cases} \ .$$

It was shown by Nisan and Wigderson [NW93] that,

$$R^{k-1}_{1/4}(k\text{-POINTER}) = \Omega(m/k^2 - k\log m) \ .$$

In Chapter 11 we generalize this result to consider multi-valued functions. In Chapter 4 we prove a related result in which there are k-players and player i has a function $f_i \in F_m$. The goal is to compute $f_k(f_{k-1}(\ldots f_1(1)\ldots))$.

2.2 Sampling Methods

Reservoir Sampling: Reservoir sampling, due to Vitter [Vit85], is a simple technique that maintains a uniform random sample S of size s from a data stream. At the beginning we add the first s elements of the stream to S. When the j-th data element, a_j, arrives with probability s/j we remove a random element from S and add a_j to S. It is straight-forward to show that, at each step j, S is a uniform random sample from $\{a_i : i \in [j]\}$. As described the algorithm seems to require $O(\log t)$ random coin tosses at each step. However, this can be substantially reduced. See [Vit85] for details.

AMS Sampling: Another important sampling technique that we will use in Chapters 8 and 9 is a method introduced by Alon, Matias and Szegedy [AMS99]. The procedure is designed for processing stream $A = \langle a_1, \ldots, a_m\rangle$ where $a_i \in [n]$. Suppose we wish to estimate $\overline{f}(A) := \frac{1}{m}\sum_{i=1}^{n} f(m_i)$ where f is some function and $m_i = |\{a_j : a_j = i, j \in [m]\}|$ for all $i \in [n]$. Examples of a such a quantity include frequency moments and entropy.

The basic idea is to space-efficiently generate a random variable R defined thus: Pick $J \in [m]$ uniformly at random and let $R = |\{j : a_j = a_J, J \le j \le m\}|$. Then we define $X = f(R) - f(R-1)$. It can easily be shown that $\mathrm{E}[X] = \overline{f}(A)$.

To ensure that $\mathrm{E}[X]$ is "close" to $\overline{f}(A)$ with high probability we can generate many independent copies of X and average them appropriately. We now describe one way to achieve this. For integers c_1 and c_2, we define the random variable

$$\mathrm{Est}_f(R, c_1, c_2) \ := \ \operatorname*{median}_{1 \le j \le c_2} \left(\frac{1}{c_1} \sum_{i=1}^{c_1} X_{ij} \right) . \tag{2.1}$$

where the random variables $\{X_{ij}\}$ are independent and each distributed identically to $X = (f(R) - f(R-1))$. Then, an appropriate combination of Chernoff-Hoeffding and Chebychev bounds yields the following lemma.

Lemma 2.1. *Let $X := f(R) - f(R-1)$, $c_1 \geq (8/\epsilon^2)(\mathrm{Var}[X]/\mathrm{E}[X]^2)$ and $c_2 \geq 4 \lg \delta^{-1}$. Then $\mathrm{E}[X] = \overline{f}(A)$ and the estimator $\mathrm{Est}_f(R, c_1, c_2)$ gives an (ϵ, δ)-approx. for $\overline{f}(A)$ using space $c_1 c_2$ times the space required to maintain R.*

However, in some applications it is more convenient to use a slightly different approach in which we simply take the mean of c instantiations of X. Suppose that X is bounded in the range $[-a, b]$ where $a, b \leq 0$. For an integer c, define the random variable

$$\mathrm{Est}_f(R, c) \ := \ \frac{1}{c} \sum_{i=1}^{c} X_i \, , \tag{2.2}$$

where the random variables $\{X_i\}$ are independent and each distributed identically to $(f(R) - f(R-1))$. Appealing to Chernoff-Hoeffding bounds one can show the following lemma.

Lemma 2.2. *Let $X := f(R) - f(R-1)$, $a, b \geq 0$ such that $-a \leq X \leq b$, and*

$$c \geq 3(1 + a/\mathrm{E}[X])^2 \epsilon^{-2} \ln(2\delta^{-1})(a+b)/(a + \mathrm{E}[X]) \ .$$

Then $\mathrm{E}[X] = \overline{f}(A)$ and, if $\mathrm{E}[X] \geq 0$, the estimator $\mathrm{Est}_f(R, c)$ gives an (ϵ, δ)-approximation to $\overline{f}(A)$ using space c times the space required to maintain R.

This new technique will improve over the previous technique when $a = 0$ and the best bound for $\mathrm{Var}[X]$ is b^2. This alone will sometimes be enough to improve upon previous results. However, in Chapter 9, we will describe a more involved variation of the AMS-sampling technique that we will use to approximate the entropy of a stream. In Chapter 8, we show how to extend the technique to approximating functions of two arguments.

2.3 Concentration Bounds

Given the importance of sampling and other randomized algorithms in this thesis will finish this chapter by summarizing some of the concentration bounds we will use. These concentration bounds include the Chernoff-Hoeffding bound.

Lemma 2.3 (Chernoff-Hoeffding)**.** *Let* $\{X_t\}_{1\leq t\leq m}$ *be independently distributed random variables with (continuous) range* $[0,u]$. *Let* $X = \sum_{1\leq t\leq m} X_t$. *Then for* $\gamma > 0$,

$$\Pr\left[|X - E\left[X\right]| \geq \gamma E\left[X\right]\right] \leq 2\exp\left(\frac{-\gamma^2 E\left[X\right]}{3u}\right) \quad .$$

While Chernoff-Hoeffding bounds are used frequently, at various points we will need to use less common variants of these bounds. In particular we will be interested in sampling without replacement.

Consider a population C consisting of N values $\underline{c} = \{c_1, \ldots, c_N\}$. Let the mean value of the population be denoted $\mu = N^{-1}\sum_{i=1}^{N} c_i$ and let $c^* = \max_{1\leq i\leq N} c_i - \min_{1\leq i\leq N} c_i$. Let $X_1, \ldots, X_n$ denote a random sample without replacement from C and $\bar{X} = n^{-1}\sum_{i=1}^{n} X_i$.

Theorem 2.4 (Hoeffding)**.**

$$\Pr\left[\bar{X} \notin (\mu - a, \mu + b)\right] \leq \exp\left(-2na^2/c^{*2}\right) + \exp\left(-2nb^2/c^{*2}\right) \quad .$$

The following corollary will also be useful.

Corollary 2.5. *Assume that* $c_1 = c_2 = \ldots = c_k = 1$ *and* $c_{k+1} = c_{k+2} = \ldots = c_N = 0$.

$$\Pr\left[\bar{X} \notin (\mu - a, \mu + b)\right] \leq \exp\left(-2n^2a^2/k\right) + \exp\left(-2n^2b^2/k\right) \quad .$$

Part I

Oracles and Ordering

Chapter 3

Introduction

3.1 Random-Order Streams

Many functions that we may wish to compute are invariant under permutation of the data items. For such functions it is natural to attempt to design algorithms that correctly compute the relevant function even when the ordering is chosen adversarially. However this is not always possible. Perhaps the situation would be ameliorated if the stream was in *random order*.

Definition 3.1. *Consider a set of elements* $a_1, \ldots, a_m \in \mathcal{U}$. *Then this set and* $\pi \in \mathrm{Sym}_m$ *defines a stream* $\langle x_{\pi(1)}, \ldots, x_{\pi(m)} \rangle$. *If* π *is chosen uniformly from* Sym_m *then we say the stream is in* random-order.

Of course, an algorithm for processing a stream in random-order may have a probability of failure even if it is deterministic. However, as with randomized algorithms, we could hope to design algorithms with arbitrarily small failure probabilities. For a randomized algorithm, reducing the failure probability normally comes at the price of tossing more random coins and incurring a penalty in terms of space and time complexity. When processing a random-order stream there is not the option of "adding more randomness" to reduce the failure probability that is due to the ordering of the stream. But we shall see that it is sometimes possible to reduce this failure probability at the expense of increasing approximation factors.

There are numerous reasons to study random-order streams. In the literature to date, it is usually assumed that the stream is ordered by an adversary that, while it may know the algorithm being used, is unaware of the coin tosses made by the algorithm. A natural complexity question is how powerful is such an adversary, i.e., how much more resources are required to process a stream that is adversarially ordered rather than one that is randomly ordered? Alternatively if we impose certain computational constraints on the adversary, a popular idea in cryptography, how does this affect the resources required to compute in the model?

Another reason is because random-order streams gives rise to a natural notion of average-case analysis. We expand on this in the next section. Lastly, there are numerous scenarios when it is reasonable to assume that the stream is randomly ordered. We detail some such scenarios in Section 3.1.2.

3.1.1 Random-Order Streams as Average Case Analysis

As mentioned above, it is impossible to solve many important problems in small space when the stream is ordered adversarially. However, it is desirable to know under what circumstances these problems can be solved. There is a natural choice regarding how to go about doing this.

When evaluating a permutation invariant function f of a stream, there are two orthogonal components to an instance. Firstly, there is the object $\mathcal{O}$ described by a set of data items $\{a_j : i \in [m]\}$. Since f is permutation invariant, $\mathcal{O}$ determines the value of f. Secondly, there is the permutation of the stream σ that determines the ordering of the stream. One approach when designing algorithms is to make an assumption about $\mathcal{O}$, e.g., to assume that $\{a_j : i \in [m]\}$ are a set of values that are distributed according to, say a Gaussian distribution. Unfortunately it often hard to know the distribution of typical instances. We avoid this pitfall by, rather than making assumptions about $\mathcal{O}$, considering which problems can be solved, with high probability, when the data items are ordered randomly. This approach is an *average case analysis* where σ is chosen uniformly from all possible permutations but $\mathcal{O}$ is chosen worst case.

3.1.2 When are streams randomly ordered?

There are also many scenarios in which it is reasonable to assume the stream is not ordered adversarily. These include scenarios where the stream order is random either because of the semantics of data, by design, or by definition.

Random by Definition: A natural setting in which a data stream would be ordered randomly is if each element of the stream is a sample drawn independently from some unknown distribution. Clearly, no matter what the source distribution, given the m samples taken, each of the $m!$ permutations of the sequence of samples were equally likely. We will discuss this scenario in further detail in Sections 3.2 and 3.3.

Random by Semantics: In other situations, the semantics of the data in the stream may imply that the stream is randomly ordered. For example, consider a database of employee records in which the records are sorted by "surname." We wish to estimate some property of the salaries of the employees given a stream of $\langle$surname, salary$\rangle$ tuples. If we assume there is no correlation between the lexicographic ordering of the surnames and the numerical ordering of salaries then the values in salary field of the tuple are indeed in a random order. We note that several query optimizers make such assumptions.

Random by Design: Lastly, there are some scenarios in which we dictate the order of the stream. Naturally, we can therefore ensure it is non-adversarial! An example is the "backing sample" architecture proposed by Gibbons, Matias, and Poosala [GMP02, GM99] for maintaining accurate estimates of aggregate properties of a database. A large sample is stored in the disk and this sample is used to periodically correct estimates of the relevant properties.

3.1.3 Related Work

The random-order model was first considered by Munro and Paterson [MP80] but, prior to our work, has received little attention. Demaine, López-Ortiz, and Munro [DLOM02] considered the frequency estimation of internet packet streams assuming that the packets arrived in random order. Kannan [Kan01] conjectured that any problem that could be solved in $P/2$ passes of a randomly ordered stream could be solved in at most P passes of an adversarially ordered stream.

We will investigate random-order streams in the context of quantile estimation. To review the previous work we need to formally define what it means to estimate an Υ-approximate k-rank element in a set. The definition is a generalization of the standard definition to deal with multi-sets.

Definition 3.2 (Rank and Approximate Selection)**.** *The rank of an item x in a set S is defined as,* $\textsc{Rank}_S(x) = |\{x' \in S | x' < x\}| + 1$*. Assuming there are no duplicate elements in S, we say x is an Υ-approximate k-rank element in S if,* $\textsc{Rank}_S(x) = k \pm \Upsilon$*. If there are duplicate elements in S then we say x is an Υ-approximate k-rank element if there exists some way of ordering identical elements such that x is an Υ-approximate k-rank element.*

Munro and Paterson considered finding exact quantiles with limited storage in one of the earliest papers on the data stream model [MP80]. They considered the problem in the adversarial-order model and in the random-order model. They show that exact selection in an adversarially-ordered stream of length m is possible with P passes and $O(m^{1/P})$ space. In particular, this show that exact selection is possible in poly-logarithmic space if the algorithm may have $O(\log m)$ passes over the data [MP80]. They also conjectured $O(\log \log m)$ passes would suffice if the stream was in random-order but only showed that with P passes, $O(m^{1/(2P)})$ space is sufficient.

The problem has received a lot of attention in recent years starting from the work of Manku, Rajagopalan, and Lindsay [MRL98, MRL99]. The authors of [MRL98, MRL99] showed that it is possible to find an ϵm-approximate median $O(\epsilon^{-1} \log^2 \epsilon m)$ space. This was improved to a deterministic $O(\epsilon^{-1} \log \epsilon m)$ space algorithm by Greenwald and Khanna [GK01]. This was extended to a model supporting deletions by Gilbert et al. [GKMS02]. Gupta and Zane [GZ03] and Cormode et al. [CKMS06] presented algorithms for finding an ϵk-approximate k-rank element.

3.1.4 Our Results

We present the following results for single pass algorithms.

- A single-pass algorithm using $O(1)$ words of space that, given any k, returns an element of rank $k \pm O(k^{1/2} \log^2 k)$ if the stream is randomly ordered. Our algorithm does not require prior knowledge of the length of the stream. In comparison, we show that an algorithm achieving similar precision when the stream is ordered adversarially would require polynomial space.

- We introduce two notions of the order of the stream being *semi-random*: *t-bounded-adversary random* (t-BAR) and *ϵ-generation random* (ϵ-GR). As the names suggest the first is related to the computational power of an adversary ordering the stream and the second is related to the random process that determines the order. We show how the performance of our algorithm degrades as the "randomness" of the decreases according to either notion. The notion of ϵ-GR will also be important for proving lower bounds in the random-order stream model.

- Any algorithm that returns an m^δ-approximate median of a randomly ordered stream with probability at least 3/4 requires $\Omega(\sqrt{m^{1-3\delta}/\log(n)})$ space.

We present the following results for multiple pass algorithms.

- There exists a $O(\log \log m)$-pass, $O(\text{polylog}(m, 1/\delta))$-space, algorithm that returns the exact median with probability $1 - \delta$. This resolves the conjecture of Munro and Paterson [MP80].

- Any algorithm that returns an m^{δ}-approximate median in P passes of an adversarially ordered stream requires $\Omega(m^{(1-\delta)/P} P^{-6})$ space. This disproves the conjecture of Kannan [Kan01].

3.2 Oracle Models

In this section we return to the idea that we may want to process a stream of samples. In particular, we will be interested in computing something about the empirical distribution defined by the relative frequency of the values in the stream.

Two main oracle models have been used in the property testing literature for testing properties of distributions. These are the *generative* and *evaluative* models introduced by Kearns et al. [KMR+94]. The generative model of a distribution permits only one operation: taking a sample from the distribution. In other words, given a distribution $p = \{p_1, \dots p_n\}$, `sample`(p) returns i with probability p_i. In the evaluative model, a `probe` operation is permitted and `probe`(p, i) returns p_i. A natural third model, the *combined* model was introduced by Batu et al. [BDKR05]. In this model both the `sample` and `probe` operations are permissible. In all three models, the complexity of an algorithm is measured by the number of operations.

In the streaming model, the algorithm will "see" all the data but will only have sub-linear space to remember what has been seen. In the oracle models, only a fraction of the data will be revealed but there is greater flexibility about how this data is accessed and there is no space restriction. It is natural to ask how these models relate to each other.

3.2.1 Related Work

Feigenbaum et al. [FKSV02b] initiated the study of a related problem. They considered testing properties of a length m list of values. They consider processing the list in the data-stream model and in an oracle model in which queries can be made to the value contained in each position of the list. They showed that there exist functions that are easy in their oracle model but hard to test in streams and vice versa. In particular, they show that testing Sorted-Superset, the property that the elements in the first half of the list (which are promised to be sorted) are a permutation of the elements in the second half, requires $\Omega(m)$ space in the streaming model but only requires $O(\log m)$ queries in the oracle model. Conversely, testing Groupedness, the property that all identical values in the list appear consecutively, requires $\Omega(\sqrt{m})$ queries in the oracle while it only requires $O(\log m)$ space in the streaming model.

Given such a result it may appear that the problem of relating oracle models to streaming models is resolved. However, most of the properties considered by [FKSV02b] are dependent on the ordering of the data stream. When considering properties of the empirical distribution defined by a stream, the ordering of the stream is irrelevant. Furthermore, for many properties, the actual values of the data items in the stream are not important in the sense the property is invariant under re-labeling the values in the stream. Such properties include the entropy of the data stream or the ℓ_1 difference between two empirical distributions.

3.2.2 Our Results

We consider estimating *symmetric properties* of a distribution, i.e., properties that are invariant under a relabeling of the data items. For such properties we relate the computational power of the combined oracle model to the data-stream model. In particular, we consider an combined-oracle that "knows" the empirical distribution defined by a stream A. We show that any algorithm $\mathcal{A}$ that makes $O(k)$ queries to the oracle can be transformed into an algorithm $\mathcal{A}'$ for processing A using $O(k)$ space. If A is adversarially ordered then $\mathcal{A}'$ requires two passes over A but if A is randomly ordered then the algorithm only requires a single pass.

3.3 Space-Efficient Sampling

In this section, we continue with the theme of processing a stream of samples. However, unlike the previous section, we will not be interested in computing something about the empirical distribution defined by the stream. Rather, we will be interested in using these samples to make some inference about the source of these samples on the assumption that each sample is drawn independently from the source. This is in stark contrast to almost all the streaming research to date.

This introduces various new issues. *Sample-complexity* is a fundamental measure of complexity in many learning problems. A very general set-up in statistics and machine-learning is that an algorithm may request a sequence of independent samples from some source for the purposes of making some inference about the source. Sample complexity is simply the number of samples that must be requested before the algorithm can make the desired inference with sufficiently high probability. Unfortunately, when we are trying to reason about complicated systems, Vapnik-Chervonenkis arguments, for example, can be used to show that many samples are required. Processing these samples can become expensive in terms of the length of time taken to process these samples and the space necessary to store these samples. While the time-complexity is necessarily at least as large as the sample-complexity, can the space-complexity can be significantly less?

The question we are posing is essentially a computational generalization of the notion of sufficient statistics: do we need to retain all the samples that are generated, or can we maintain small summaries and still make the desired inferences? In other words, can we separate the space and sample complexities of a learning problem.

This is particularly important when we have a fast data source. For example, suppose we are sampling traffic at a router in an attempt to monitor network traffic patterns. We can very quickly gather a large sample; but to store this sample creates a significant problem as a typical monitoring system is only permitted a small memory footprint and writing to disk would slow the system down considerably. Even if we were permitted a reasonably sized footprint, we may be able to make more accurate inferences and predictions if we use the space more efficiently than just storing samples.

Another important issue that arises is the potential trade-off between sample complexity and space complexity. Because we may not be able to store all the samples in the allotted space, our algorithms potentially incur a loss of accuracy. Consequently, we may need to investigate a slightly larger set of samples and thereby offset the loss of accuracy. The aforementioned problem of estimating medias illustrates some of these ideas.

Example 3.3. *We say y is an ϵ-approximate median of a one dimensional distribution with probability density function D if,*

$$\int_{-\infty}^{y} D(x)dx \in 1/2 \pm \epsilon \quad .$$

It can be shown that the sample complexity of finding an ϵ-approximate median is $\Theta(\epsilon^{-2})$. However, it can also be shown that there exists a constant $c(\delta)$ such that, with probability at least $1 - \delta$, any element whose rank is in the range $m/2 \pm \epsilon m/2$ in a set of $m = c/\epsilon^2$ samples is an ϵ-approximate median. But there exist algorithms [GK01] using only $O(\epsilon^{-1} \log \epsilon m)$ space that, when presented with a stream of m values will return an element whose rank is in the range $m/2 \pm \epsilon m/2$. Hence the space complexity of learning an ϵ-approximate median is only $O(\epsilon^{-1} \log \epsilon^{-1})$.

Furthermore, since the samples are in random order, we know that with $O(1)$ words of space we can return an element with rank $m/2 \pm O(\sqrt{m} \ln^2 m \ln \delta^{-1})$. Hence, by increasing the number of samples to $O(\epsilon^{-2} \ln^4 \epsilon^{-1})$ we may decrease the space complexity to $O(1)$ words.

3.3.1 Related Work

The main piece of related work is by Chang and Kannan [CK06]. They consider processing a stream of samples that are drawn from a distribution on the real line with probability generating function D that has at most k piecewise-linear segments. They design an algorithm that finds a probability density function $\hat{D}$ such that $\int_{\mathbb{R}} |D(x) - \hat{D}(x)| \leq \epsilon$. However, this algorithm makes multiple passes over the data stream which necessitates that the samples are stored somewhere. Also the algorithm does not take advantage of the fact that the samples will be in a random-order and therefore the algorithm works even if the stream is reordered. We will discuss the details of the algorithm in relation to our results in the next section.

More generally, the idea of processing a stream of samples is related to on-line algorithms (see, e.g., [BEY98]) and on-line learning (see, e.g., [Blu98]). However in the on-line model, space issues have not been considered widely. Furthermore, there is a notion of an irrevocable decision or guess being made at each step. This does not have a direct analogue in the data stream model, where we are primarily concerned with the accuracy of an estimate of some function after all the data has been seen. However, because of the space restriction, the algorithm is forced to make an irrevocable decision about what information to "remember" (explicitly or implicitly) about the new item and what information currently encoded in the current memory state can be "forgotten."

3.3.2 Our Results

For discrete distributions, over a domain $[n]$, Batu et al. [BFR$^+$00] provide algorithms with sublinear (in n) sample complexity. These algorithms test whether two distributions are almost identical in variational distance or at least ϵ-far apart. We show that the space complexity of their algorithm can be improved with a small blowup in sample complexity. Furthermore, we consider approximating the distance between two distributions.

	Chang and Kannan [CK06]	This Work	
Length	$O(k^6\epsilon^{-6})$	$O(k^2\epsilon^{-4})$	$O(k^2\epsilon^{-4})$
Space	$O(k^3\epsilon^{-4/P})$	$O(k\epsilon^{-2/P})$	$O(k)$
Passes	P	P	1
Order	Adversarial	Adversarial	Random

Table 3.1: Comparison of Results for Learning Distributions. Log terms are omitted and P is even and assumed constant.

Next we consider the problem of learning piecewise-linear probability density function as considered by Chang and Kannan [CK06]. We present an single-pass algorithm using $\tilde{O}(k^2\epsilon^{-4})$ samples and $\tilde{O}(k)$ space. Our algorithm capitalizes on the fact that a sequence of independent samples will be in random-order. We also present an algorithm directly comparable to that in [CK06], i.e., the algorithm must process the samples in an arbitrary order but is permitted multiple passes over the stream of samples. Our algorithm takes P (assumed constant) passes over $\tilde{O}(k^2\epsilon^{-4})$ samples and uses $\tilde{O}(k\epsilon^{-2/P})$ space. In comparison, the algorithm of [CK06] takes P passes over $\tilde{O}(k^6\epsilon^{-6})$ samples and uses $\tilde{O}(k^3\epsilon^{-4/P})$ space. See Table 3.1.

We conclude with a section about the importance of the assumption that the samples in the stream are ordered randomly rather than adversarially. We discuss our ideas in the context of estimating the frequency moments of a discrete distribution.

Chapter 4

Random vs. Adversarial Order

Chapter Outline: We present a single-pass algorithm for computing an approximate quantile of a randomly ordered stream. We present generalizations for semi-randomly ordered streams and show how to compute an exact quantile in multiple passes. Finally, we present lower-bounds for both random-order streams and adversarial-order streams thereby refuting a conjecture of Kannan [Kan01]. For background see Chapter 3.

4.1 Algorithms for Random-Order Streams

In this section we show how to perform approximate selection of the k-th smallest element in a single pass over a randomly-ordered stream S. We will present the algorithm assuming the exact value of the length of the stream, m, is known in advance. In a subsequent section, we will show that this assumption is not necessary. In what follows we will assume that the stream contains distinct values. This can easily be achieved with probability at least $1 - \delta$ by attaching a secondary value $y_i \in_R [m^2\delta^{-1}]$ to each item x_i in the stream. We say $(x_i, y_i) < (x_j, y_j)$ iff $x_i < x_j$ or ($x_i = x_j$ and $y_i < y_j$). Note that breaking the ties arbitrarily results in a stream whose order is not random. We also may assume that $k \leq m/2$ by symmetry.

```
Selection Algorithm:
  1. Let $\Upsilon = 10\ln^2(m)\ln(\delta^{-1})\sqrt{k}$ and $p = 2(\lg(m/\Upsilon) + \sqrt{\ln(3/\delta)\lg(m/\Upsilon)})$
  2. Let $a = -\infty$ and $b = +\infty$
  3. Let $l_1 = m\Upsilon^{-1}\ln(3m^2p/\delta)$ and $l_2 = 2(m-1)\Upsilon^{-1}\sqrt{(k+\Upsilon)\ln(6mp/\delta)}$
  4. Partition the stream as $S = \langle S_1, E_1, \ldots S_p, E_p\rangle$ where $|S_i| = l_1$, $|E_i| = l_2$
  5. Phase $i$:
     (a) Sample sub-phase: If $S_i \cap (a,b) = \emptyset$ return $a$, else let $u \in_R S_i \cap (a,b)$
     (b) Estimate sub-phase: Let $r = \textsc{Rank}_{E_i}(u)$ and $\tilde{r} = \frac{(m-1)(r-1)}{l_2} + 1$
     (c) Update sub-phase: If $\tilde{r} < k - \Upsilon/2$, $a \leftarrow u$, $\tilde{r} > k + \Upsilon/2$, $b \leftarrow u$ else
         return $u$
```

Figure 4.1: An Algorithm for Quantile Estimation

Algorithm Overview: Our algorithm proceeds in phases and each phase is composed of three distinct sub-phases; the *Sample* sub-phase, the *Estimate* sub-phase, and the *Update* sub-phase. At all points we maintain an open interval (a, b) such that we believe that the value of the element with rank k is between a and b In each phase we aim to narrow the interval (a, b). The Sample sub-phase finds a value $u \in (a, b)$. The Estimate sub-phase estimates the rank of u. The Update sub-phase replaces a or b by u depending on whether the rank of u is believed to be less or greater than u. See Fig. 4.1 for the algorithm.

Algorithm Analysis: For the analysis we define the following quantity,

$$\Gamma(a, b) = |S \cap (a, b)| = |\{v \in S : a < v < b\}| \ .$$

Lemma 4.1. *With high probability, for all phases, if* $\Gamma(a, b) \geq \Upsilon$ *then there exists an element* u *in each sample sub-phase, i.e.,*

$$\Pr\left[\forall\ i \in [p], a, b \in S \text{ such that } \Gamma(a, b) \geq \Upsilon; S_i \cap (a, b) \neq \emptyset\right] \geq 1 - \delta/3 \ .$$

Proof. Fix $i \in [p]$ and $a, b \in S$ such that $\Gamma(a,b) \geq \Upsilon$. Then,

$$\Pr\left[S_i \cap (a,b) \neq \emptyset\right] \geq 1 - \left(1 - \frac{\Gamma(a,b)}{m}\right)^{l_1} \geq 1 - \exp(-\Upsilon l_1/m) = 1 - \frac{\delta}{3m^2p} \ .$$

The result follows by applying the union bound over all choices of i, a and b. $\square$

Lemma 4.2. *With high probability, for all phases, we determine the rank of u with sufficient accuracy, i.e.,*

$$\Pr\left[\forall i \in [p], u \in S; \begin{array}{ll} \tilde{r} = \textsc{Rank}_S(u) \pm \Upsilon/2 & \textit{if } \textsc{Rank}_S(u) < k + \Upsilon + 1 \\ \tilde{r} > k \pm \Upsilon/2 & \textit{if } \textsc{Rank}_S(u) \geq k + \Upsilon + 1 \end{array}\right] \geq 1 - \delta/3 \ ,$$

where $\tilde{r} = (m-1)(\textsc{Rank}_{E_i}(u) - 1)/l_2 + 1$.

Proof. Fix $i \in [p]$ and $u \in S$. First we consider u such that $\textsc{Rank}_S(u) < k + \Upsilon + 1$. Let $X = \textsc{Rank}_{E_i}(u) - 1$ and note that $E[X] = l_2(\textsc{Rank}_S(u) - 1)/(m-1)$.

$$\begin{aligned} \Pr\left[\tilde{r} \neq \textsc{Rank}_S(u) \pm \Upsilon/2\right] &= \Pr\left[|X - E[X]| \geq \frac{l_2\Upsilon}{2(m-1)}\right] \\ &\leq 2\exp\left(\frac{-2(l_2\Upsilon/(2(m-1)))^2}{\textsc{Rank}_S(u) - 1}\right) \\ &\leq \frac{\delta}{3mp}. \end{aligned}$$

Now assume that $\textsc{Rank}_S(u) \geq k + \Upsilon + 1$ and note that $\Pr\left[\tilde{r} \geq k + \Upsilon/2\right]$ is minimized for $\textsc{Rank}_S(u) = k + \Upsilon + 1$. Hence,

$$\begin{aligned} \Pr\left[\tilde{r} > k + \Upsilon/2\right] &= 1 - \Pr\left[E[X] - X \geq \frac{l_2\Upsilon}{2(m-1)}\right] \\ &\geq 1 - \exp\left(-\frac{(l_2\Upsilon)^2}{4(k+\Upsilon)(m-1)^2}\right) \\ &= 1 - \frac{\delta}{6mp} \ . \end{aligned}$$

The result follows by applying the union bound over all choices of i and u. $\square$

We now give the main theorem of this section.

Theorem 4.3. *For $k \in [m]$, there exists a single-pass, $O(1)$-space algorithm in the random-order model that returns u such that $\textsc{Rank}_S(u) = k \pm 10\ln^2(m)\ln(\delta^{-1})\sqrt{k}$ with probability at least $1 - \delta$.*

Proof. Consider $\textsc{Gap} = |\{v \in S : a < v < b\}|$ in each phase of the algorithm. By Lemma 4.1 and Lemma 4.2, with probability at least $1 - 2\delta/3$, in every phase $\textsc{Gap}$ decreases (unless $\textsc{Gap}$ is already less than Υ) and $\textsc{Rank}_S(a) \leq k \leq \textsc{Rank}_S(b)$. In particular, with probability 1/2, $\textsc{Gap}$ decreases by at least a factor 2 between each phase. Let X be the number of phases in which $\textsc{Gap}$ halves. If the algorithm does not terminate then $X < \lg(m/\Upsilon)$ since $\textsc{Gap}$ is initially m and the algorithm will terminate if $\textsc{Gap} < \Upsilon$. But,

$$\Pr[X < \lg(m/\Upsilon)] = \Pr\left[X - E[X] < \sqrt{\ln(3/\delta)\lg(m/\Upsilon)}\right] \geq 1 - \delta/3 \ .$$

Hence with probability at least $1 - \delta$ the algorithm returns a value with rank $k \pm \Upsilon$.

The space bound immediately from the fact that the algorithm only stores a constant number of polynomially sized values and maintains a count over $O(m)$ elements. Finally, for sufficiently large m,

$$\begin{aligned} p(l_1 + l_2) &= O\left(\left(\lg\frac{m}{\Upsilon} + \sqrt{\ln(\delta^{-1})\lg\frac{m}{\Upsilon}}\right)\left(m\ln\frac{mp}{\delta} + m\sqrt{(k+\Upsilon)\ln\frac{mp}{\delta}}\right)\Upsilon^{-1}\right) \\ &= O(\ln^2(m)\ln(\delta^{-1})m\Upsilon^{-1}\sqrt{k}) \\ &= O(m) \ . \end{aligned}$$

□

4.1.1 Generalizing to Unknown Stream Lengths

The algorithm in the previous section assumed prior knowledge of n, the length of the stream. We now discuss a simple way to remove this assumption. First we argue that, for our purposes, it is sufficient to only look at half the stream!

Lemma 4.4. *Given a randomly ordered stream S of length m, let S' be a contiguous sub-stream of length $\tilde{m} \geq m/2$. Then, with probability at least $1 - \delta$, if u is the $\tilde{k}$-th smallest element of S', then,*

$$\textsc{Rank}_S(u) = \tilde{k}m/\tilde{m} \pm 2(8\tilde{k}\ln\delta^{-1})^{0.5} \ .$$

Proof. Let $a = \tilde{k}/\tilde{m}$. Let the elements in the stream be $x_1 \le \ldots \le x_m$. Let $X = |\{x_1, \ldots, x_{am+b}\} \cap S'|$ and $Y = |\{x_1, \ldots, x_{am-b-1}\} \cap S'|$ where $b = 2(8\tilde{k} \ln \delta^{-1})^{0.5}$. The probability that the element of rank $\tilde{k} = a\tilde{m}$ in S' has rank in S outside the range $[am - b, am + b]$ is less than,

$$\begin{aligned}
\Pr[X < a\tilde{m} \text{ or } Y > a\tilde{m}] &\le \Pr[X < E[X] - b/2 \text{ or } Y > E[Y] + b/2] \\
&\le 2 \exp\left(\frac{-(b/2)^2}{3(a\tilde{m} + b)}\right) \\
&\le \delta\ .
\end{aligned}$$

The lemma follows. □

To remove the assumption that we know m, we make multiple instantiations of the algorithm. Each instantiation corresponds to a guess of m. Let $\beta = 1.5$. Instantiation i guesses a length of $\lceil 4\beta^i \rceil - \lfloor \beta^i \rfloor + 1$ and is run on the stream starting with the $\lfloor \beta^i \rfloor$-th data item and ending with the $\lceil 4\beta^i \rceil$-th data item. We remember the result of the algorithm until the $2(\lceil 4\beta^i \rceil - \lfloor \beta^i \rfloor + 1)$-th element arrives. We say the instantiation has been canceled at this point.

Lemma 4.5. *At any time there are only a constant number of instantiations. Furthermore, when the stream terminates, at least one instantiation has run on a sub-stream of at least $m/2$.*

Proof. Consider the t-th element of the data stream. By this point there have been $O(\log_\beta t)$ instantiations made. However, $\Omega(\log_\beta t/6)$ instantiations have been canceled. Hence, $O(\log_\beta t - \log_\beta t/6) = O(1)$ instantiations are running. We now show that there always exists an instantiation that has been running on at least half the stream. The i-th instantiation gives a useful result if the length of the stream $m \in U_i = \{\lfloor 4\beta^i \rfloor + 1, \ldots, 2(\lceil 4\beta^i \rceil - \lfloor \beta^i \rfloor + 1)\}$. But $\bigcup_{i \ge 0} U_i = \mathbb{N} \setminus \{0, 1, 2, 3, 4\}$ since for all $i > 1$, $\lfloor 4\beta^i + 1 \rfloor \le 2(\lceil 4\beta^{i-1} \rceil - \lfloor \beta^{i-1} \rfloor + 1)$. □

We can therefore generalize Theorem 4.3 as follows,

Theorem 4.6. *For $k \in [m]$, there exists a single-pass, $O(1)$-space algorithm in the random-order model that returns u such that $\textsc{Rank}_S(u) = k \pm 11 \ln^2(m) \ln(\delta^{-1})\sqrt{k}$ with probability at least $1 - \delta$. The algorithm need not know m in advance.*

4.1.2 Multi-Pass Exact Selection

In this section we consider the problem of exact selection of an element of rank $k = \Omega(m)$. We will later show that this requires $\Omega(\sqrt{m})$-space if an algorithm is permitted only one pass over a stream in random-order. However, if $O(\log\log m)$ passes are permitted we now show that $O(\operatorname{polylog} m)$ space is sufficient.

We use a slight variant of the single-pass algorithm in Section 4.1 as a building block. Rather than returning a single candidate, we output the pair a and b. Using the analysis in Section 4.1, it can be shown that, with probability $1-\delta$, $\textsc{Rank}_S(a) \leq k \leq \textsc{Rank}_S(b)$ and that

$$|\textsc{Rank}_S(a) - \textsc{Rank}_S(b)| \leq O(\sqrt{m} \log^2 m \log \delta^{-1}) \ .$$

In one additional pass, $\textsc{Rank}_S(a)$ and $\textsc{Rank}_S(b)$ can be computed exactly. Hence, after two passes, by ignoring all elements less than a or greater than b, we have reduced the problem to that of finding an element of rank $k - \textsc{Rank}_S(a) + 1$ in a stream of length $O(\sqrt{m} \log^3 m)$[1]. If we repeat this process $O(\log\log m)$ times and then select the desired element by explicitly storing the remaining $O(\operatorname{polylog} m)$-length stream, it would appear that we can perform exact selection in $O(\operatorname{polylog} m)$-space and $O(\log\log m)$-passes. However, there is one crucial detail that needs addressed. In the first pass, by assumption we are processing a data stream whose order is chosen uniformly from Sym_m. However, because the stream-order is not re-randomized between each pass, it is possible that the previous analysis does not apply because of dependences that may arise between different passes. Fortunately, the following straight-forward, but crucial, observation demonstrates that this is not

[1]We will assume that $\delta^{-1} = \operatorname{poly}(m)$ in this section.

the case.

Lemma 4.7. *Consider the set $\{x_1, \ldots, x_m\}$ and $\pi \in \mathrm{Sym}_m$. Let a and b be the bounds returned after a pass of the algorithm on the stream $\langle x_{\pi(1)}, \ldots, x_{\pi(m)} \rangle$. Let $\pi' \in \mathrm{Sym}_m$ satisfy $\pi(i) = \pi'(i)$ for all $i \in [m]$ such that $x_i \notin [a, b]$. Then the algorithm also would return the same bounds after processing the stream $\langle x_{\pi'(1)}, \ldots, x_{\pi'(m)} \rangle$.*

Therefore, conditioned on the algorithm returning a and b, the sub-stream of elements in the range $[a, b]$ are ordered uniformly at random. This leads to the following theorem.

Theorem 4.8. *For $k \in [m]$, there exists an $O(\operatorname{polylog} m)$-space, $O(\log \log m)$-pass algorithm in the random-order model that returns the k-th smallest value of a stream with probability $1 - 1/\operatorname{poly}(m)$.*

4.1.3 Applications to Equi-Depth Histograms

In this section, we overview an application to constructing B-bucket equi-depth histograms. Here, the histogram is defined by B buckets whose boundaries are defined by the items of rank $im/(B + 1)$ for $i \in [B]$. Gibbons, Matias, and Poosala [GMP02] consider the problem of constructing an approximate B-bucket equi-depth histogram of data stored in a backing sample. The measure of "goodness-of-fit" they consider is

$$\mu = m^{-1} \sqrt{B^{-1} \sum_{i \in B} \epsilon_i^2} \ ,$$

where ϵ_i is the error in the rank of the boundary of the i-th bucket. They show that μ can be made smaller than any $\epsilon > 0$ where the space used depends on ϵ. However, in their model it is possible to ensure that the data is stored in random order. As a consequence of the algorithm in Section 4.1, we get the following theorem.

Theorem 4.9. *In a single pass over a backing sample of size m stored in random order we can compute the B-quantiles of the samples using $O(B \log m)$ memory with error $\tilde{O}(m^{-1/2})$.*

Since the error goes to zero as the sample size increases, we have the first consistent estimator for this problem.

4.2 Notions of Semi-Random Order

In this section we consider two natural notions of "semi-random" ordering and explain how our algorithm can be adjusted to process streams whose order is semi-random under either definition. The first notion is stochastic in nature: we consider the distribution over orders which are "close" to the random order in an ℓ_1 sense. This will play a critical role when proving lower bounds.

Definition 4.10 (ϵ-Generated-Random Order). *Given set $\{x_1, \ldots, x_m\}$, $\pi \in \mathrm{Sym}_m$ defines a stream $\langle x_{\pi(1)}, \ldots, x_{\pi(m)} \rangle$. We say the order is* ϵ-Generated Random *(ϵ-GR) if π is chosen according to a distribution ν such that $\|\mu - \nu\|_1 \leq \epsilon$ where μ is the uniform distribution on* Sym_m.

The importance of this definition is captured in the following simple lemma.

Lemma 4.11. *Let $\mathcal{A}$ be a randomized algorithm that succeeds (i.e., returns an estimate of some property with some accuracy guarantee) with probability at least $1 - \delta$ in the random-order model. Then $\mathcal{A}$ succeeds with probability at least $1 - \delta - \epsilon$ when the stream order is ϵ-GR.*

Proof. Let $\Pr_{\mu,\mathrm{coin}}(\cdot)$ denote the probability of an event over the internal coin tosses of $\mathcal{A}$ and the ordering of the stream when the stream-order is chosen according to the uniform distribution μ. Similarly, define $\Pr_{\nu,\mathrm{coin}}(\cdot)$ where ν is any distribution satisfying $\|\mu - \nu\|_1 \leq \epsilon$.

$$\Pr_{\mu,\mathrm{coin}}(\mathcal{A} \text{ succeeds}) = \sum_{\pi \in \mathrm{Sym}_m} \Pr_{\mu}(\pi) \Pr_{\mathrm{coin}}(\mathcal{A} \text{ succeeds} | \pi) \leq \Pr_{\nu,\mathrm{coin}}(\mathcal{A} \text{ succeeds}) + \epsilon \ .$$

The lemma follows since $\Pr_{\mu,\mathrm{coin}}(\mathcal{A} \text{ succeeds}) \geq 1 - \delta$ by assumption. $\square$

The next theorem follows immediately from Theorem 4.3 and Lemma 4.11.

Theorem 4.12. *For $k \in [m]$, there exists a single-pass, $O(\log m)$-space algorithm in the ϵ-GR-order model that returns u such that $\textsc{Rank}_S(u) = k \pm 11 \ln^2(m) \ln(\delta^{-1})\sqrt{k}$ with probability at least $1 - \delta - \epsilon$.*

The second definition is computational in nature. We consider an adversary upstream of our algorithm that can re-order the elements subject to having limited memory to do this reordering.

Definition 4.13 (t-Bounded-Adversary-Random Order)**.** *A* t-bounded adversary *is an adversary that can only delay at most t elements at a time, i.e., when presented with a stream $\langle x_1, \ldots, x_m\rangle$ it can ensure that the received stream is $\langle x_{\pi(1)}, \ldots x_{\pi(m)}\rangle$ if $\pi \in \mathrm{Sym}_m$ satisfies,*

$$\forall i \in [m] \ , |\{j \in [m] : j < i \text{ and } \pi(i) < \pi(j)\}| \leq t \ . \tag{4.1}$$

The order of a stream is t-bounded-adversary-random (t-BAR) if it is generated by a t-bounded adversary acting on a stream whose order is random.

For example, with $t = 2$, $\langle 1, 2, 3, 4, 5, 6, 7, 8, 9\rangle$ can become $\langle 3, 2, 1, 6, 5, 4, 9, 8, 7\rangle$ or $\langle 3, 4, 5, 6, 7, 8, 9, 1, 2\rangle$ but not $\langle 9, 8, 7, 6, 5, 4, 3, 2, 1\rangle$. In particular, in the adversarial-order model the stream order is $(n-1)$-BAR while in the random-order model the order is 0-BAR.

Lemma 4.14. *Consider streams $S = \langle x_1, \ldots, x_m\rangle$ and $S' = \langle x_{\pi(1)}, \ldots, x_{\pi(m)}\rangle$ where π satisfies Eq. 4.1. Let $S_{a,b} = \langle x_{i_1}, x_{i_2}, \ldots\rangle$ and $S'_{a,b} = \langle x_{\pi(i_1)}, x_{\pi(i_2)}, \ldots\rangle$ be the sub-streams of elements in the range (a, b). Then for any $j, w \in [m]$, $|\{x_{i_j}, \ldots, x_{i_{j+w}}\} \cap \{x_{\pi(i_j)}, \ldots, x_{\pi(i_{j+w})}\}| \geq w - t$.*

We assume that $t \leq \sqrt{k}$. Given the above lemma is quite straight-forward to transform the algorithm of the previous section into one that is correct (with prescribed probability) when processing a stream in t-BAR order. In particular, it is sufficient to set $l_1 = O(m\Upsilon^{-1}\ln(3m^2p/\delta) + t\delta^{-1})$ and to choose a random u among $S_i \cap (a,b)$ in each sample-phase. Note that $l_1 < l_2$ for $t \leq \sqrt{k}$. In each estimate-phase a t-bounded adversary can introduce an extra $mt/l_2 \leq t\Upsilon/\sqrt{k} \leq \Upsilon$ error. Hence, the total error is at most 2Υ.

Theorem 4.15. *For $k \in [m]$, there exists a single-pass, $O(1)$-space algorithm in the t-BAR-order model that returns u such that* $\textsc{Rank}_S(u) = k \pm 20\ln^2(m)\ln(\delta^{-1})\sqrt{k}$ *with probability at least $1-\delta$.*

4.3 Random-Order Lower-Bound

In this section we will prove a lower bound on the space required to m^δ-approximate the median in the single-pass random-order model. Our lower-bound will be based on a reduction from the communication complexity of indexing [KN97]. However, the reduction is significantly more involved then typical reductions because different segments of a stream can not be determined independently by different players if the stream is in random order.

Consider two players Alice and Bob where Alice has a binary string σ of length s and Bob has an index $r \in [s]$ where s will be determined later. It is known that for Bob to determine $\textsc{Index}(\sigma, r) = \sigma_r$ after a single message from Alice with probability at least 4/5, then this message must consist of $\Omega(s)$ bits. More precisely,

Theorem 4.16 (e.g., [KN97])**.** $R^1_{1/5}(\textsc{Index}) \geq c^*s$ *for some constant $c^* > 0$.*

We start by assuming that there exists an algorithm $\mathcal{A}$ that computes an m^δ-approximate median in the single-pass, random-order model with probability at least 9/10. We then use this to construct a one-way communication protocol that will allow Alice and Bob to solve their $\textsc{Index}$ problem. They do this by simulating $\mathcal{A}$ on a stream of length m where Alice determines the prefix of the stream and Bob determines the remaining prefix. The stream they determine consists of the union of the following sets of elements:

X: A size x set consisting of $m/2 + m^\delta - 2m^\delta r$ copies of 0.

Y: A size y set consisting of $n/2 - m^\delta - 2m^\delta(s - r)$ copies of $2s + 2$.

Z: A size $z = 2m^\delta s$ set consisting of $2m^\delta$ copies of $\{2i + \sigma_i : i \in [s]\}$.

Note that any m^δ-approximate median of $U = S \cup X \cup Y$ is $2r + \sigma_r$. The difficulty we face is that we may only assume $\mathcal{A}$ returns an m^δ-approximate median of U if U is ordered randomly. Ensuring this seems to require a significant amount of communication between Alice and Bob. How else can Alice determine the balance of elements from X and Y in the prefix of the stream or Bob know the elements of Z that should appear in the suffix of the stream?

In what follows we will argue that by carefully choosing the length of the prefix, suffix, and s, it is possible for Alice and Bob to ensure that the ordering of the stream is 1/20-generated-random while Alice and Bob need only communicate a sufficiently small number of bits with probability at least 19/20. Then, by appealing to Lemma 4.11, we may assume that the protocol is correct with probability at least 4/5.

Generating a Stream in Semi-Random Order: Let A be the set of elements in the part of the stream determined by Alice. Let $B = U \setminus A$ be the set of elements in the part of the stream determined by Bob. Let $p = c^*/(8m^\delta \log m)$ and consider the following protocol:

1. Alice determines $A \cap Z$ and $B \cap Z$ by placing an element from Z into B with probability p and placing it in A otherwise. Alice picks t_0 according to $T_0 \sim \text{Bin}(m/2 - z, 1 - p)$ and t_1 according to $T_1 \sim \text{Bin}(m/2 - z, 1 - p)$. She places t_0 copies of 0 and t_1 copies of 1 into A. She sends a message encoding $B \cap Z$, t_0, t_1 and the memory state of $\mathcal{A}$ ran on a random permutation of A.

2. Bob instantiates $\mathcal{A}$ with memory state sent by Alice and continues running it on a random permutation of $B = (B \cap Z) \cup \{x - t_0 \text{ copies of } 0\} \cup \{(y -$

t_1 copies of 1}. Finally, Bob returns 1 if the output of the algorithm is odd and 0 otherwise.

Let ν be the distribution over stream-orders generated by the above protocol. The next lemma establishes that ν is almost uniform. This will be required to prove the correctness of the algorithm.

Lemma 4.17. *If* $z = 10^{-6}\sqrt{pm}$ *then* $\|\mu - \nu\|_1 \le 1/20$ *where* μ *is the uniform distribution on* Sym_m.

Proof. Define the random variables $T_0' \sim \mathrm{Bin}(x, 1-p)$ and $T_1' \sim \mathrm{Bin}(y, 1-p)$ and let $a_0 = x - m/2 + z$ and $a_1 = y - m/2 + z$. Note that $a_0, a_1 \ge 0$ and $a_0 + a_1 = z$. We upper-bound $\|\mu - \nu\|_1$ as follows,

$$\begin{aligned} \|\mu - \nu\|_1 &= \sum_{t_0,t_1} \left|\Pr[T_0 = t_0, T_1 = t_1] - \Pr[T_0' = t_0, T_1' = t_1]\right| \\ &\le \max_{\substack{t_0 \in (1-p)x \pm b^* \\ t_1 \in (1-p)y \pm b^*}} \left|\frac{\Pr[T_0 = t_0, T_1 = t_1]}{\Pr[T_0' = t_0, T_1' = t_1]} - 1\right| + \Pr[A \ge b^* - pz] + \Pr[B \ge b^*] \end{aligned}$$

where $A = \max\{|T_0 - E[T_0]|, |T_1 - E[T_1]|\}$, $B = \max\{|T_0' - E[T_0']|, |T_1' - E[T_1']|\}$, and $b^* = 10\sqrt{pm/2} + pz$. By the Chernoff bound,

$$\Pr[A \ge b^* - pz] + \Pr[B \ge b^*] \le 8 \exp\left(-2(b^* - pz)^2/(3pm)\right)$$

and hence the last two terms are upper-bounded by 1/40 for sufficiently large m.

Let $t_0 = (1-p)x + b_0$ and $t_1 = (1-p)x + b_1$ and assume that $|b_0|, |b_1| \le b^*$. Then, $\Pr[T_0 = t_0, T_1 = t_1] / \Pr[T_0' = t_0, T_1' = t_1]$ equals,

$$\frac{\binom{m/2-z}{t_0}\binom{m/2-z}{t_1}}{\binom{x}{t_0}\binom{y}{t_1}p^z} = \left(\prod_{i \in [a_0]} \frac{xp - i + 1 - b_0}{(x - i + 1)p}\right)\left(\prod_{i \in [a_1]} \frac{yp - i + 1 - b_1}{(y - i + 1)p}\right) ,$$

and therefore,

$$\exp\left(\frac{-zb^*}{p(x-z)} + \frac{-zb^*}{p(y-z)}\right) \le \frac{\Pr[T_0 = t_0, T_1 = t_1]}{\Pr[T_0' = t_0, T_1' = t_1]} \le \exp\left(\frac{2z^2 + zb^*}{p(x-z)} + \frac{2z^2 + zb^*}{p(y-z)}\right).$$

Substituting z establishes that $|\Pr[T_0 = t_0, T_1 = t_1] / \Pr[T_0' = t_0, T_1' = t_1] - 1| \le 1/40$ for sufficiently large m. The lemma follows. $\square$

The next lemma will be necessary to bound the communication of the protocol.

Lemma 4.18. $\Pr[|Z \cap B| \geq c^* s/(2\log m)] \leq 1/20$.

Proof. Note that $E[|Z \cap B|] = pz$. Then, by an application of the Chernoff bound,

$$\Pr[|Z \cap B| \geq c^* s/(2\log m)] = \Pr[|Z \cap B| \geq 2E[|Z \cap B|]] = \exp(-c^* s/(12\log m)) \ .$$

□

Theorem 4.19. *Computing an m^δ-approximate median in the random-order model with probability at least $9/10$ requires $\Omega(\sqrt{m^{1-3\delta}/\log m})$ space.*

Proof. Let Alice and Bob follow the above protocol to solve their instance of Index. Assume $\mathcal{A}$ use M bits of space. By Lemma 4.11 and Lemma 4.17, the protocol is correct with probability at least $9/10 - 1/20 = 17/20$. Furthermore, by Lemma 4.18, with probability at least $19/20$ the protocol requires at most $3c^* s/4 + M$ bits of communication (for sufficiently large m): $c^* s/2$ bits to transmit $Z \cap B$, $2\log m$ bits to transmit t_0 and t_1, and M bits for the memory state of $\mathcal{A}$. Therefore, there exists a protocol transmitting $3c^* s/4 + M$ bits that is correct with probability at least $17/20 - 1/20 = 4/5$. Hence, by Theorem 4.16, $M = \Omega(s) = \Omega(\sqrt{m^{1-3\delta}/\log m})$. □

4.4 Adversarial-Order Lower-Bound

In this section we prove that any P-pass algorithm that returns an m^δ-approximate median in the adversarial-order model requires $\Omega(m^{(1-\delta)/P} P^{-6})$ space. The proof will use a reduction from the communication complexity of a generalized form of pointer-chasing that we now describe.

Definition 4.20 (Generalized Pointer Chasing)**.** *For $i \in [p]$, let $f_i : [V] \to [V]$ be an arbitrary function. Then g_p is defined by*

$$g_p(f_1, f_2, \ldots, f_p) = f_p(f_{p-1}(\ldots f_1(1))\ldots)) \ .$$

S_1	S_2	S_3
$(0,0,0) \times 5(3 - f_A(1))$		
	$(1,0,0) \times (3 - f_B(1))$	
		$(1,1,f_C(1))$
		$(1,2,f_C(2))$
		$(1,3,f_C(3))$
	$(1,4,0) \times (f_B(1) - 1)$	
	$(2,0,0) \times (3 - f_B(2))$	
		$(2,1,f_C(1))$
		$(2,2,f_C(2))$
		$(2,3,f_C(3))$
	$(2,4,0) \times (f_B(2) - 1)$	
	$(3,0,0) \times (3 - f_B(3))$	
		$(3,1,f_C(1))$
		$(3,2,f_C(2))$
		$(3,3,f_C(3))$
	$(4,4,0) \times (f_B(3) - 1)$	
$(4,0,0) \times 5(f_A(1) - 1)$		

Table 4.1: Reduction from Pointer Chasing to Exact Median Finding. A triple of the form (x_2, x_1, x_0) corresponds to the numerical value $x_2 \cdot 5^2 + x_1 \cdot 5^1 + x_0 \cdot 5^0$. Note that $\text{median}(S_1 \cup S_2 \cup S_3) = f_A(1) \cdot 5^2 + f_B(f_A(1)) \cdot 5^1 + f_C(f_B(f_A(1))) \cdot 5^0$

Let the i-th player, P_i, have function f_i and consider a protocol in which the players must speak in the reverse order, i.e., $P_p, P_{p-1}, \ldots, P_1, P_p, \ldots$. We say the protocol has r rounds if P_p communicates r times. Let $R^r_\delta(g_p)$ be the total number of bits that must be communicated in an r round (randomized) protocol for P_1 to learn g_p with probability at least $1 - \delta$.

Note that $R^p_0(g_p) = O(p \log V)$. We will be looking at $(p-1)$-round protocols. The proof of the next result will be deferred to the next section.

Theorem 4.21. $R^{p-1}_{1/10}(g_p) = \Omega(V/p^4 - p^2 \log V)$.

The next theorem is proven by reducing from generalized pointer-chasing to approximate selection.

Theorem 4.22. *Finding an m^δ-approximate median in p passes with probability at least $9/10$ in the adversarial-order model requires $\Omega(m^{(1-\delta)/p}p^{-6})$ space.*

Proof. We will show how a p-pass algorithm $\mathcal{A}$ that computes a t-approximate median of a length m stream gives rise to a p-round protocol for computing g_{p+1} when $V = \left(m/((p+1)(2t+1))\right)^{1/p}/2$. If $\mathcal{A}$ uses M bits of space then the protocol uses at most $(p(p+1)-1)M$ bits. Hence, by Theorem 4.21, this implies that $M = \Omega(V/p^6) = \Omega((m/t)^{1/p}p^{-6})$.

The intuition behind the proof is that any t-approximate median will correspond to a number $g_1g_2g_3\ldots g_{p+1}$ written in base $V+2$. The input of P_1 will first determine the highest order 'bit', i.e., g_1. Then the input of P_2 will determine the g_2 and so on. Specifically, each player P_i will determine a segment of the stream S_i: P_{p+1} determines the first $m_{p+1} = |S_{p+1}|$ elements, P_p determines the next $m_p = |S_p|$, etc. These segments are defined as follows,

$$S_1 = \{\underbrace{0,\ldots\qquad\qquad\ldots,0}_{(V-f_1(1))(2t+1)(2V-1)^{p-1}},\underbrace{(V+1)b^p,\ldots,(V+1)b^p}_{(f_1(1)-1)(2t+1)(2V-1)^{p-1}}\}$$

and for $j \in \{2,\ldots,p\}$,

$$S_j = \bigcup_{x_{p+2-j},\ldots,x_p\in[V]} \{\underbrace{\sum_{i=p+2-j}^{p} x_ib^i,\ldots,\sum_{i=p+2-j}^{p} x_ib^i}_{(V-f_j(x_{p+2-j}))(2t+1)(2V-1)^{p-j}},$$
$$\underbrace{(V+1)b^{p+1-j}+\sum_{i=p+2-j}^{p} x_ib^i,\ldots,(V+1)b^{p+2-j}+\sum_{i=p+2-j}^{p} x_ib^i}_{(f_j(x_{p+2-j})-1)(2t+1)(2V-1)^{p-j}}\}\ ,$$

and finally,

$$S_{p+1} = \bigcup_{x_1,\ldots,x_p\in[V]} \{\underbrace{f_{p+1}(x_1)+\sum_{i=1}^{p} x_ib^i,\ldots,f_{p+1}(x_1)+\sum_{i=1}^{p} x_ib^i}_{2t+1}\}\ ,$$

where $b = V+2$. See Table 4.1 for the case when $p=2$, $t=0$, and $V=3$. Note that $m_{p+1} = (2t+1)V^p$ and for $j \le p$, $m_j = (2t+1)(V-1)(2V-1)^{p-j}V^{j-1} < (2t+1)V^p$.

Hence,

$$\sum_{j=1}^{p+1} m_j \leq (2t+1)(p+1)(2V)^p = m \ ,$$

and that the largest value in the stream is $(V+1)b^p = O(m)$. Any t-approximate median equals $\sum_{i=1}^{p+1} g_i b^{p+1-i}$ and, therefore, if P_1 returns a t-approximate median modulo b then this equals g_{p+1}. This can easily be computed by a protocol in which each player transmits the memory state of the algorithm at the appropriate juncture. This concludes the proof. $\square$

4.4.1 Proof of Theorem 4.21

The proof is a generalization of a proof by Nisan and Widgerson [NW93]. We present the entire argument for completeness. In the proof we lower bound the $(p-1)$-round distributional complexity, $D^{p-1}_{1/20}(g_p)$, i.e., we will consider a deterministic protocol and an input chosen from some distribution. The theorem will then follow by Yao's Lemma [Yao80] since

$$D^{p-1}_{1/20}(g_p) \leq 2R^{p-1}_{1/10}(g_p) \ .$$

Let T be the protocol tree of a deterministic p-round protocol. We consider the input distribution where each f_i is chosen uniformly from F, the set of all V^V functions from $[V]$ to $[V]$. Note that this distribution over inputs gives rise to distribution over paths from the root of T to the leaves. We will assume that in round j, P_i's message include g_{j-1} if $i > j$ and g_j if $i \leq j$. E.g., for $p = 4$ the appended information is shown in the following table where $g_0 = 1$.

	Round 1				Round 2				Round 3			
Player	4	3	2	1	4	3	2	1	4	3	2	1
Appended	g_0	g_0	g_0	g_1	g_1	g_1	g_2	g_2	g_2	g_3	g_3	-

This is possible with only $O(p^2 \log V)$ extra communication. Consequently we may assume that at each node at least $\lg V$ bits are transmitted. We will assume that protocol T requires at most $\epsilon V/2$ bits of communication where $\epsilon = 10^{-4}(p+1)^{-4}$ and derive a contradiction.

Consider a node z in the protocol tree of T corresponding to the jth round of the protocol when it is P_i's turn to speak. Let g_{t-1} be the appended information in the last transmission. Note that $g_0, g_1, \ldots, g_{t-1}$ are specified by the messages so far.

Denote the set of functions $f_1 \times \ldots \times f_p$ that are consistent with the messages already sent be $F_1^z \times \ldots \times F_p^z$. Note that the probability of arriving at node z is $|F|^{-p} \prod_{1 \le j \le p} |F_j^z|$. Also note that, conditioned on arriving at node z, $f_1 \times \ldots \times f_p$ is uniformly distributed over $F_1^z \times \ldots \times F_p^z$.

Definition 4.23. *Let c_z be the total communication until z is reached. We say a node z in the protocol tree is* nice *if, for $\delta = \max\{4\sqrt{\epsilon}, 400\epsilon\}$, it satisfies the following two conditions:*

$$|F_j^z| \ge 2^{-2c_z}|F| \textit{ for } j \in [p] \quad \textit{and} \quad H(f_t^z(g_{t-1})) \ge \lg V - \delta \ .$$

Claim 4.24. *Given the protocol reaches node z and z is nice then,*

$$\Pr[\textit{next node visited is nice}] \ge 1 - 4\sqrt{\epsilon} - 1/V \ .$$

Proof. Let w be a child of z and let $c_w = c_z + a_w$. For $l \neq i$ note that $|F_l^w| = |F_l^z|$ since P_l did not communicate at node z. Hence, the probability that we reach node w given we have reached z is $\prod_{1 \le j \le p} |F_j^w|/|F_j^z| = |F_i^w|/|F_i^z|$. Furthermore, since z is nice,

$$\Pr\left[|F_i^w| < 2^{-2c_w}|F|\right] \le \Pr\left[\frac{|F_i^w|}{|F_i^z|} < 2^{-2a_w}\right] \le \sum_w 2^{-2a_w} \le \frac{1}{V}\sum_w 2^{-a_w} \le \frac{1}{V} \ .$$

where the second last inequality follows from $a_w \ge \lg V$ and the last inequality follows by Kraft's inequality (the messages sent must be prefix free because the receiving player needs to know when they are supposed to send the next message.) Hence, with probability at least $1-1/V$, the next node in the protocol tree satisfies the first condition for being nice.

Proving the second condition is satisfied with high probability is more complicated. Consider two different cases, $i \neq t$ and $i = t$, corresponding to whether or not player i appended g_t. In the first case, since P_t did not communicate, $F_t^z = F_t^w$ and hence $H(f_t^w(g_{t-1})) = H(f_t^z(g_{t-1})) \geq \lg V - \delta$.

We now consider the second case. In this case we need to show that $H(f_{t+1}^w(g_t)) \geq \lg V - \delta$. Note that we can express f_{t+1}^w as the following vector of random variables, $(f_{t+1}^w(1), \ldots, f_{t+1}^w(V))$ where each $f_{t+1}^w(v)$ is a random variables in universe $[V]$. Note there is no reason to believe that components of this vector are independent. By the sub-additivity of entropy,

$$\sum_{v \in [V]} H(f_{t+1}^w(v)) \geq H(f_{t+1}^w) \geq \lg(2^{-2c_w}|F|) = \lg(|F|) - 2c_w \geq V \lg V - \epsilon V$$

using the fact that f_{t+1}^w is uniformly distribution over F_{t+1}^w, $|F_{t+1}^w| \geq 2^{-2c_w}|F|$ and $c_w \leq \epsilon V/2$. Hence if v were chosen uniformly at random from $[V]$,

$$\Pr\left[H(f_{t+1}^w(v)) \leq \log V - \delta\right] \leq \epsilon/\delta \ ,$$

by Markov's inequality. However, we are not interested in a v chosen uniformly at random but rather $v = g_t = f_t^z(g_{t-1})$. However since the entropy of $f_t^z(g_{t-1})$ is large it is almost distributed uniformly. Specifically, since $H(f_t^z(g_{t-1})) \geq \lg V - \delta$ and it follows along the same lines as in [NW93], that for our choice of δ,

$$\Pr\left[H(f_{t+1}^w(g_t)) \leq \log V - \delta\right] \leq \frac{\epsilon}{\delta}\left(1 + \sqrt{\frac{4\delta}{\epsilon/\delta}}\right) \leq 4\sqrt{\epsilon} \ .$$

Hence with probability at least $1 - 4\sqrt{\epsilon}$ the next node satisfies the second condition of being nice. The claim follows by the union bound. $\square$

Note that the height of the protocol tree is $p(p-1)$ and that the root of the protocol tree is nice. Hence the probability of ending at a leaf that is not nice is at most $p(p-1)(1/V + 4\sqrt{\epsilon}) \leq 1/25$. If the final leaf node is nice then $H(g_t)$ is at least $\lg b - \delta$ and hence the probability that g_t is guessed correctly is at most $(\delta+1)/\lg V$ using Fano's inequality. This is less than $1/100$ for sufficiently large V and hence the total probability of P_1 guessing g_p correctly is at most $1 - 1/20$.

Chapter 5

Oracle vs. Stream Models

Chapter Outline: In this chapter, we prove a short result that relates the random-order and adversarial-order stream models with the combined oracle model. For background, see Chapter 3.

5.1 Connecting Oracle and Streaming Models

We say that a function $f : \mathbb{R}^n \to \mathbb{R}$ is *symmetric*, if for all $p_1, \ldots, p_n \in \mathbb{R}, i, j \in [n]$,

$$\begin{aligned} & f(p_1, \ldots, p_{i-1}, p_i, p_{i+1}, \ldots, p_{j-1}, p_j, p_{j+1}, \ldots, p_n) \\ = \quad & f(p_1, \ldots, p_{i-1}, p_j, p_{i+1}, \ldots, p_{j-1}, p_i, p_{j+1}, \ldots, p_n) \ . \end{aligned}$$

Symmetry is a desirable and often-assumed property of functions on distributions, and is a special case of invariance under coordinate re-parameterizations [Č81]. We will show that we can always express an algorithm for the combined oracle model in a canonical form where the algorithm first samples and then probes the samples along with a few other elements. The idea would be to view the original algorithm, after the sampling stages and probing of the samples, as a randomized decision tree that we rewrite as an oblivious decision tree using a result of Bar-Yossef et al. [BYKS01]. Then we could simulate this new decision tree in the random order model. We start with the necessary definitions.

Definition 5.1 (Decision Tree)**.** *A* randomized decision tree *for a function f is a decision tree having three types of nodes; a* query *node that asks for the value of an input parameter and maps the resulting value to a choice of child node to visit, a random choice node, where the child node is chosen at random, and output nodes, where an answer expressed as a function of all queries thus far is returned.*

An oblivious decision tree *is one where the queries are made independent of the input, or the random choices in the algorithm. Formally, suppose we have a tree T with worst-case query complexity u. Then an I*-relabeling *of T by $I = \{i_1, \dots i_u\}$ relabels all query nodes of depth j by the query to i_j, yielding the tree T^I. An oblivious decision tree is then a pair T, Δ_u, where T is a decision tree with worst-case complexity q and Δ_u is a distribution on $[n]^u$. A computation on an oblivious decision tree consists of two steps: (1) sample u elements I from Δ_q, (2) Relabel T to T^I and run it on input x.*

An important step in our argument will be transition between using a randomized decision tree and an oblivious decision tree. We will do this using the following result due to Bar-Yossef [BY02].

Lemma 5.2 (Bar-Yossef [BY02] Lemma 4.17)**.** *Let T be a randomized decision tree that computes an approximation to a symmetric function f with u queries in the worst case and u_E queries in the expected case with the expectation taken over the random choices used by T. Then there is an oblivious decision tree (T, W_u) where W_u is the uniform-without-replacement distribution of worst-case query complexity u and expected query complexity u_E that computes an (equally good) approximation of f.*

The first lemma shows how any combined oracle algorithm can be transformed to one of a canonical form.

Lemma 5.3 (Canonical Form Algorithm)**.** *Let $\mathcal{A}$ be an approximation algorithm for a symmetric function f using (worst-case) t oracle calls. Then there exists a* *canonical algorithm $\mathcal{A}'$ that uses (worst-case) $3t$ oracle calls and achieves an equally good approximation.*

Proof. Note that `sample` does not take a parameter and therefore only the number of samples we make can depend on the outcome of probes we may do. However, we know that there can be at most t samples taken. Hence if we request t samples initially we can assume that we do not need to do any further sampling. Note that we have at most doubled the oracle calls. Let S be the set of i's seen as samples. Obviously taking more samples than were taken by $\mathcal{A}$ can not be detrimental when it comes to correctness. Then, for each value $i \in S$ we perform $\texttt{probe}(p, i)$. This only adds t queries to the complexity.

We now have a randomized algorithm that takes as input the outcome from our t samples and the value p_i for all $i \in S$ and performs a series of further probes. Note that since all samples have already been made, this phase of the computation can be viewed as a randomized decision tree. But now we can appeal to Lemma 5.2 and argue that this randomized decision can be rewritten as an oblivious decision tree. In such a tree, all queries can be decided in advance and we now have an algorithm of the desired canonical form. □

We are now ready to prove the main structural result of this section. The central idea for simulating in two pass regular stream model is to sample in the first pass and then do exact counting in the second pass. For the random order stream result we are able to do both the sampling and exact counting in the same pass by using the prefix of the random order stream as a source for sample oracle queries.

Theorem 5.4. *Let $\mathcal{A}$ be an approximation algorithm for a symmetric function f in the combined oracle model. Then, there exist a single pass random stream algorithm and a two pass regular stream algorithm that use $O(t)$ space that returns a (equally good) approximation for f.*

Proof. We first consider a stream in random order. Consider the following streaming algorithm that uses $O(t)$ space. We store the first t items in the data stream, $\langle\langle p, i_1\rangle, \langle p, i_2\rangle, \ldots \langle p, i_t\rangle\rangle$. Now for each $i \in S_w = \{i_1, i_2, \ldots i_t\}$ we set up a counter that will be used to maintain as exact count of the frequency of i. We now chose t values, S'_w from $k \in [n] \setminus S_w$ uniformly at random (without replacement) and set up a counter for each of these t values. We also maintain a counter to estimate the length of the stream m. At the end of the data stream we claim that we can simulate the oracles calls made by any canonical algorithm. The only difficulty in establishing this claim is showing that we can use the stored prefix to simulate the generative oracle. Ideally we would like to claim that we can just return i_j on the jth query to the generative oracle. This is however not the case. We get around this dependence in the random stream model by doing the following: on the jth generative oracle call we output i_j with probability $(m-j+1)/m$ and otherwise output $i_{j'}$ where j' is chosen uniformly at random from $[j-1]$. We thus can emulate the generative oracle calls made by $\mathcal{A}$. The probes performed by $\mathcal{A}$, can also be emulated because for each i_j we have maintained counters that give us p_{i_j} and for each $k \in S'_w$ we know p_k.

For stream in adversarial order things are simpler. In the first pass we generate our random sample (with replacement) using standard techniques. In the second pass we count the exact frequencies of the relevant i. □

The proof can be generalized to the case of computing a function of two distributions. We say that such a function $f : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is *symmetric*, if for all $p_1, \ldots, p_n, q_1, \ldots, q_n \in \mathbb{R}, i, j \in [n]$,

$$\begin{aligned} & f(p_1, \ldots, p_i, \ldots, p_j, \ldots, p_n, q_1, \ldots, q_i, \ldots, q_j, \ldots, q_n) \\ = \; & f(p_1, \ldots, p_j, \ldots, p_i, \ldots, p_n, q_1, \ldots, q_j, \ldots, q_i, \ldots, q_n) \ . \end{aligned}$$

The only caveat is that we need $m(p) = \Theta(m(q))$ in order that, with high probability, there are t elements of the form $\langle p, \cdot\rangle$ and t elements of the form $\langle q, \cdot\rangle$ in the first $O(t)$ data items.

Chapter 6

Sample vs. Space Complexity

Chapter Outline: In this chapter, we consider the problem of learning properties of a distribution given independent samples but with only limited space. We start with a simple algorithmic result for learning discrete distributions. In the next section, we give our main result: a space-efficient learning algorithm for determining the probability density function of a piecewise-linear distribution. We conclude with a brief discussion about learning frequency moments and the important of random order when trading-off sample and space complexity. For background see Chapter 3.

6.1 Discrete Distributions

There exists an algorithm that test whether the ℓ_1 distance between two discrete distributions on n points, is greater than ϵ or less than $\epsilon/(4\sqrt{n})$ using $O(\epsilon^{-4}n^{2/3})$ samples [BFR$^+$00]. Here we describe a method that takes more samples but only uses $O(\epsilon^{-2}\log n)$ space. Furthermore, our algorithm will actually ϵ-additively approximate the distance. It will be an integral part of an algorithm in the next section.

We start by quoting a result by Indyk [Ind00] that will be used in this section.

Theorem 6.1 (Indyk [Ind00]). *Let A be a stream defining two n-point empirical* *distributions p and q. There exists a one-pass, $O(\epsilon^{-2}\log(n)\log(\delta^{-1}))$-space, (ϵ,δ)-approximation of $|p-q|$. We call this algorithm ℓ_1-Sketch.*

Theorem 6.2. *Consider two discrete distributions p and q on n points. Given a stream containing $m \geq 12n\log(2n/\delta)/\epsilon^2$ samples from each distribution it is possible to find an estimate T, of $|p-q|$ such that, with probability at least $1-\delta$,*

$$(1+\gamma)^{-1}(|p-q|-2\epsilon) \leq T \leq (1+\gamma)(|p-q|+2\epsilon) \ ,$$

using $O(\gamma^{-2}\log(n\log(1/\delta)/\epsilon^2)\log(1/\delta))$ space.

Proof. After taking m samples define f_i to be the number of samples equal to i. This define the empirical distribution $\hat{p}_i = f_i/m$. First we show that $|\hat{p}-p|$ is small. Note that $E[f_i] = mp_i$ and

$$\Pr\left[|\hat{p}_i - p_i| \leq \max\left\{\frac{\epsilon p_i}{2}, \frac{\epsilon}{2n}\right\}\right] = \Pr\left[|f_i - E[f_i]| \leq m\max\left\{\frac{\epsilon p_i}{2}, \frac{\epsilon}{2n}\right\}\right] \leq \frac{\delta}{n} \ .$$

Hence with probability at least $1-\delta$, for all $i \in [n]$, $|\hat{p}_i - p_i| \leq \max\{\epsilon/(2n), \epsilon p_i/2\}$. Therefore $|\hat{p}-p| \leq \epsilon$.

We can prove $|\hat{q}-q|$ is small in an identical way. Hence $\big||\hat{p}-\hat{q}|-|p-q|\big| \leq 2\epsilon$. We can approximate $|\hat{p}-\hat{q}|$ upto a multiplicative factor of $1+\gamma$ in $O(\gamma^{-2}\log(n\epsilon^{-2}\log\delta^{-1}))$ space using the result in Theorem 6.1. □

One interesting corollary of this result is as follows.

Corollary 6.3. *Let D be a member of a finite family $\mathcal{F}$ of hypothesis distributions on $[n]$. There exists a one-pass, $O(\log(n\log(|\mathcal{F}|\delta^{-1})/\epsilon^2)\log(|\mathcal{F}|\delta^{-1}))$-space algorithm that finds $F \in \mathcal{F}$ such that $\Pr[|D-F| \leq \epsilon] \geq 1-\delta$ given $O(n\log(n|\mathcal{F}|\delta^{-1})/\epsilon^2)$ samples.*

This follows by setting $\gamma = 1$ and making the error probability sufficiently small that the ℓ_1 difference between the stream and each hypothesis distribution is accurately estimated.

```
Algorithm Linear$(J, X, \beta, \delta)$
1.  $\eta \leftarrow \beta/8k$
2.  Partition range $[l,u)$ into $[l+(i-1)(u-l)\eta, l+i(u-l)\eta)$, $i \in [1/\eta]$
3.  Using the algorithm $\ell_1$-Sketch, let $T$ be a 2-approximation to
        $$\sum_{1\leq i\leq 1/\eta} |\tilde{d}_i - l_i|$$
    where
        $$\tilde{d}_i = \frac{|X \cap [l+(i-1)(u-l)\eta, l+i(u-l)\eta)|}{|X|} \text{ and } l_i = (a(2i-1)/2+b)\eta$$
    where $a$ and $b$ satisfying $l_1 = \tilde{d}_1$ and $a/2+b=1$
4.  If $T \leq \beta/4$ then accept otherwise reject
```

Figure 6.1: An Algorithm for Testing if a Distribution is Linear

6.2 Continuous Distributions

Consider a distribution on the real line with probability generating function D that has at most k piece-wise linear segments. Alternatively D can be viewed as a mixture of $O(k)$ linear distributions. We wish to find a k-linear probability density function $\hat{D}$ such that

$$\int_{\mathbb{R}} |D(x) - \hat{D}(x)| dx \leq \epsilon \ .$$

Let us assume we know the range of the distribution. Note that this can be learned with error at most ϵ by taking the maximum and minimum values of the first $O(\epsilon^{-1}\log(1/\delta))$ samples. By scaling we will assume that the range of the distribution is $[0,1)$.

The Linear Algorithm: An important sub-routine in our algorithm will be an algorithm that tests whether a stream of samples from a k-piecewise linear distribution on the range $[l,u)$ is linear. The algorithm is presented in Fig. 6.1. The intuition behind the algorithm is to quantize the samples to $1/\eta$ equally spaced values between

```
Algorithm Learning-Piecewise-Linear-Distributions$(X, \epsilon, \delta)$
1.  Define the following set of values,

        $t_1 \leftarrow k, t_2 \leftarrow 2$ and $\alpha \leftarrow 1/42$
        $d_p^l \leftarrow (1-2\alpha)^{-p} t_1 t_2^{p-1}$ and $d_p^u \leftarrow (1+2\alpha)^{-p} t_1 t_2^{p-1}$ for $p \in [\ell]$
        $\ell \leftarrow \lceil \log(2kt_2\epsilon^{-1}/t_1)/\log(t_2/(1+2\alpha)) \rceil$ and $\delta_1 \leftarrow \delta/(6\ell k)$

2.  Partition the stream: $X = X_{1,1}, X_{1,2}, X_{2,1}, X_{2,2}, \ldots, X_{\ell,1}, X_{\ell,2}$ where

            $|X_{1,1}| = m_{\text{qua}}(t_1, \alpha, \delta_1)$
            $|X_{p,1}| = 3m_{\text{qua}}(t_2, \alpha, \delta_1) d_p^l \log(\delta_1^{-1})$ for $p \in [\ell]$
            $|X_{p,2}| = 3m_{\text{lin}}(k, \epsilon d_p^u/(\ell k), \delta_1) d_p^l \log(\delta_1^{-1})$ for $p \in [\ell]$

    where $m_{\text{qua}}(\cdot,\cdot,\cdot)$ and $m_{\text{lin}}(\cdot,\cdot,\cdot)$ are defined in Eq. 6.1 and Eq. 6.3.
3.  $\mathcal{J}_0 \leftarrow \{[0,1)\}$
4.  for $p \in [\ell]$, $J = [a_0, a_t) \in \mathcal{J}_{p-1}$:
5.      do if $p = 1$ then $t = t_1$ else $t = t_2$
6.          Using Quantiles, find $(a_i)_{i \in [t-1]}$ with $\textsc{Rank}_{X_{p,1}}(a_i) = \frac{i}{t} \pm \frac{\alpha}{2t}$.
7.          Let partit$(J) = \{[a_0, a_1), \ldots, [a_{t-1}, a_t)\}$.
8.          for $J' \in$ partit$(J)$:
9.              do if Linear$(J', J' \cap X_{p,2}, \frac{\epsilon d_p^u}{2\ell k}, \delta_1)$ rejects then $\mathcal{J}_p \leftarrow \{J'\} \cup \mathcal{J}_p$
```

Figure 6.2: An Algorithm for Piecewise-Linear Distributions

l and u. The algorithm computes a linear distribution that, if the k-piecewise linear distribution is linear, is close to the distribution. This distance is computed with the algorithm ℓ_1-*Sketch*.

Theorem 6.4 (The *Linear* Algorithm)**.** *Let X be a stream consisting of*

$$m_{lin}(k, \beta, \delta) := ck\beta^{-3} \log(ck\beta^{-1}\delta^{-1}) \tag{6.1}$$

(for some sufficiently large constant c) samples drawn from the distribution D_J, the distribution formed by conditioning D on the interval J. Then, with probability $1-\delta$, it will accept if D_J is linear on J and will reject if D_J is not within ℓ_1

distance β of a linear distribution on J. Furthermore, if D_J is linear then the algorithm determines a linear distribution at most β from D_J. The algorithm uses $O((\log(1/\beta) + \log k)\log(1/\delta))$ space.

Proof. Without loss of generality, we may assume that the range of J is $[l, u) = [0, 1)$. Let L be a linear probability density function on the range $[0, 1)$ of the form $ay + b$ where a and b will be determined later.

Let $\eta = \beta/(8k)$. Let $d_i = \int_{(i-1)\eta}^{i\eta} D_J(y)dy$ and $l_i = \int_{(i-1)\eta}^{i\eta} L(y)dy = (a(2i-1)/2 + b)\eta$ where a and b are determined by $l_1 = \tilde{d}_1$ and $a/2 + b = 1$ (recall from Fig. 6.1 that $\tilde{d}_i$ is the probability mass observed in the interval $[(i-1)\eta, i\eta))$. First note that $l_i \leq 2\eta$ for $i \in [1/\eta]$. Then,

$$\int_{(i-1)\eta}^{i\eta} |D_J(y) - L(y)|dy \leq l_i + d_i \leq 2l_i + |d_i - l_i| \leq 4\eta + |d_i - l_i| \ ,$$

and $\int_{(i-1)\eta}^{i\eta} |D_J(y) - L(y)|dy \geq |d_i - l_i|$. Because D_J has at most k linear segments there are at most k values of i such that $\int_{(i-1)\eta}^{i\eta} |D_J(y) - L(y)|dy \neq |l_i - d_i|$. Therefore,

$$\int_0^1 |D_J(y) - L(y)|dy \leq \beta/2 + \sum_{i\in[1/\eta]} |d_i - l_i| \ . \tag{6.2}$$

At this point we consider the discrete distributions $(d_1, \ldots, d_{1/\eta})$ and $(l_1, \ldots, l_{1/\eta})$. Appealing to Theorem 6.2, it is possible to approximate $\sum_{i\in[1/\eta]} |d_i - l_i|$ up to an additive term of $\epsilon = \beta/10$ and multiplicative term of $\gamma = 11/10$ with probability at least $1 - \delta/10$ using $O(\log(1/\delta)\log(\beta^{-2}\eta^{-1}))$ space. Therefore, the estimate T satisfies,

$$\frac{10}{11}\sum |d_i - l_i| - \frac{\beta}{11} \leq T \leq \frac{11}{10}\sum |d_i - l_i| + \frac{11\beta}{100} \ .$$

Combining this with Eq. 6.2, we get that,

$$\frac{10}{11}\int_0^1 |D_J(y) - L(y)|dy - \frac{6\beta}{11} \leq T \leq \frac{11}{10}\int_0^1 |D_J(y) - L(y)|dy + \frac{66\beta}{100} \ .$$

Hence, if D_J is β-far from all linear density functions then, D_J is β-far from L. Hence $T \geq 4\beta/11$. Now suppose D_J is linear. By an application of the Chernoff-Hoeffding bounds we know that with probability at least $1-\delta/10$, $|\tilde{d}_1 - d_1| < \beta\eta/10$ and therefore $\int_0^1 |D_J(y) - L(y)|dy \leq \beta/10$. In this case $T \leq 22\beta/100$. Hence *Linear* accepts D_J if it is linear. $\square$

The Learning Algorithm: Our algorithm uses the same template as an algorithm of Chang and Kannan [CK06]. The algorithm operates in ℓ phases. Each phase p generates a set of intervals $\mathcal{J}_p$. Intuitively these intervals are those on which D does not appear to be close to linear. The union of these intervals is a subset of the union of intervals in $\mathcal{J}_{p-1}$ where $\mathcal{J}_0$ contains only one interval, the range of the distribution $[0,1)$. Consequently the algorithm can be viewed as if "zooming" in on the sub-intervals of $[0,1)$ on which D is not linear. In the first phase of the algorithm the range is partitions $[0,1)$ into t_1 intervals of roughly equal probability mass using the algorithm *Quantiles*. Each of these intervals are tested in parallel to see if the distribution is linear when restricted to that interval. This step uses the algorithm *Linear*, given earlier. In subsequent phases, intervals that do not appear to be linear are further subdivided into t_2 intervals of roughly equal probability mass. Each of the sub-intervals are tested and, if they are not linear, are further sub-divided. This process continues for ℓ phases. At the end of the ℓ-th phase there will be at most k sub-intervals that remain and appear not to be linear. However these sub-intervals will contain so little mass that approximating then by the uniform distribution will not contribute significantly to the error.

The algorithm is given in Fig. 6.2. In the streaming model each iteration of Lines 4 and 8 are performed in parallel. Note that the algorithm potentially finds $2\ell k$ intervals upon which the distribution is linear. Given these, a good k-piecewise-linear representation $\hat{D}$, can be found by dynamic programming.

We will first prove an important lemma that establishes how well we can sub-divide intervals.

Lemma 6.5. *Let X be set of*

$$m_{qua}(1/\gamma, \alpha, \delta) := c\gamma^{-2}\alpha^{-2}\log(c\delta^{-1}\gamma^{-1}) \tag{6.3}$$

(for some sufficiently large constant c) samples from a distribution D and, for $i \in [1/\gamma]$, let $x_i \in X$ be an element whose relative rank (with respect to X) is in the range $[\gamma(i-\alpha/2), \gamma(i+\alpha/2)]$. Then with probability at least $1-\delta$, for all $i \in [1/\gamma]$, x_i has relative rank (with respect to D) in the range $[i\gamma - \gamma\alpha, i\gamma + \gamma\alpha]$.

Proof. Let a and b be such that $\int_{-\infty}^{a} D(x)dx = \gamma - \gamma\alpha$ and $\int_{b}^{\infty} D(x)dx = (1-\gamma)-\gamma\alpha$. Consider the set X of n samples. Let Y_a (Y_b) be the number of elements in X that are less (greater) than a (b). Then the probability that an element whose relative rank (with respect to X) is in the range $[\gamma - \alpha\gamma/2, \gamma + \alpha\gamma/2]$ does not have relative rank (with respect to D) in the range $[\gamma - \gamma\alpha, \gamma + \gamma\alpha]$ is bounded above by,

$$\begin{aligned}
& \Pr\left[Y_a > (\gamma - \frac{\alpha\gamma}{2})m\right] + \Pr\left[Y_b > (1-\gamma-\frac{\alpha\gamma}{2})m\right] \\
= \; & \Pr\left[Y_a > \left(1 + \frac{\alpha\gamma}{2(\gamma-\alpha\gamma)}\right)E\left[Y_a\right]\right] + \Pr\left[Y_b > \left(1+\frac{\alpha\gamma}{2(1-\gamma-\alpha\gamma)}\right)E\left[Y_b\right]\right] \\
\leq \; & \exp\left(\frac{-\alpha^2\gamma^2 m}{12(\gamma-\alpha\gamma)}\right) + \exp\left(\frac{-\alpha^2\gamma^2 m}{12(1-\gamma-\alpha\gamma)}\right) \\
\leq \; & 2\exp(-m\alpha^2\gamma^2/12) \ .
\end{aligned}$$

Setting $m = 12\gamma^{-2}\alpha^{-2}\log(\delta/(2\gamma))$ ensures that this probability is less than $\delta\gamma$. The lemma follows by the union bound. □

Our algorithm will use an algorithm of Greenwald and Khanna [GK01] that finds elements of approximate *relative rank* where the relative rank of x in a set X is defined as,

$$\textsc{Rank}_X(x) := |X|^{-1}|\{y \in X, y \leq x\}| \ .$$

Theorem 6.6 (Greenwald and Khanna [GK01]). *Consider a stream X of length m. There exists a single-pass algorithm Quantiles using $O(\epsilon^{-1}\log\epsilon m)$ space that constructs a synopsis of the data such that, for any $k \in [m]$ this synopsis can return an x with $\textsc{Rank}_X(x) = k/m \pm \epsilon$.*

We now prove the main properties of the algorithm.

Lemma 6.7. *With probability at least* $1-\delta$*, for all* $p \in [\ell]$*,*

1. $|\{\mathcal{J}_p\}| \le k$ *and for each* $J \in \mathcal{J}_p$*,* $\frac{1}{d_p^l} \le D(J) \le \frac{1}{d_p^u}$*.*

2. *For each* $J \in \mathcal{J}_p, J' \in \text{partit}(J)$*, in Line 8 the call to Linear "succeeds", i.e., accepts if* D *is linear on* $X_{p,2} \cap J'$ *and rejects if the distribution formed by conditioning* D *on* J' *is* $\epsilon d_p^u/(2\ell k)$*-far from linear.*

Proof. The proof is by induction on p. Clearly $|\{\mathcal{J}_1\}| \le k$. Since $|X_{1,1} \cap J| = m_{\text{qua}}(t_1, \alpha, \delta_1)$ for all $J \in \{\mathcal{J}_0\}$, by Lemma 6.5,

$$\forall J \in \mathcal{J}_1,\ \frac{1-2\alpha}{t_1} \le D(J) \le \frac{1+2\alpha}{t_1}$$

with probability at least $1-\delta_1$. Appealing to the Chernoff bound and the union bound, with probability at least $1-\delta_1 k$,

$$\forall J \in \mathcal{J}_0, J' \in \text{partit}(J),\ |X_{1,2} \cap J'| \ge m_{\text{lin}}\left(k, \frac{\epsilon d_1^u}{2\ell k}, \delta_1\right) .$$

Hence, by Theorem 6.4, with probability $1-\delta_1 k$, each call to *Linear* in the first phase succeeds.

Assume the conditions hold for phase $p-1$. Appealing to the Chernoff bound and union bound, with probability at least $1-\delta_1 k$, $|X_{p,1} \cap J| \ge m_{\text{qua}}(t_2, \alpha, \delta_1)$ for all $J \in \mathcal{J}_{p-1}$. Hence by Lemma 6.5, $\forall J \in \mathcal{J}_{p-1}, J' \in \text{partit}(J)$,

$$\frac{1-2\alpha}{t_2} D(J) \le D(J') \le \frac{1+2\alpha}{t_2} D(J)$$

with probability at least $1-k\delta_1$. Similarly, with probability at least $1-2\delta_1 k$,

$$\forall J \in \mathcal{J}_{p-1}, J' \in \text{partit}(J),\ |X_{1,2} \cap J'| \ge m_{\text{lin}}\left(k, \frac{\epsilon d_p^u}{2\ell k}, \delta_1\right) .$$

Hence, by Theorem 6.4, with probability $1-2\delta_1 k$, each call to *Linear* in the p-th phase succeeds.

Hence, with probability at least $1-6k\ell\delta_1 = 1-\delta$ the conditions hold for all $p \in [\ell]$. □

Theorem 6.8. *With probability at least $1-\delta$, it is possible to compute an approximation to D within ℓ_1 distances ϵ using a single pass over $m = O(k^2\epsilon^{-4})$ samples with $\tilde{O}(k)$ space.*

Proof. Assume that the conditions in Lemma 6.7 hold. When *Linear* determines that an interval is close enough to linear in level p there is the potential of incurring $(\epsilon d^u_p/(2\ell k))/d^u_p = \epsilon/(2\ell k)$ error. This can happen to at most $k\ell$ intervals and hence contributes at most $\epsilon/2$ error.

The only other source of errors is the fact that there might be some intervals remaining at the last phase when $p = \ell$. However the probability mass in each interval is at most $\epsilon/(2k)$. There will be at most k of these intervals and hence the error incurred in this way is at most $\epsilon/2$.

The space complexity of the algorithm is $\tilde{O}(k)$ because at most $t_2|\{\mathcal{J}_p\}| \leq 2k$ instances of *Linear* are run in parallel. The sample complexity is (see Eq. 6.1, Eq. 6.3, and Fig. 6.2 for the definitions),

$$\sum_{p\in[\ell]}(|X_{p,1}|+|X_{p,2}|) \leq \tilde{O}\left(d^l_\ell + \max_{p\in[\ell]}\frac{d^l_p k}{(\epsilon d^u_p/k)^3}\right) \leq \tilde{O}\left(\frac{k}{\epsilon}+\frac{k^2}{\epsilon^3}\right)\left(\frac{1+2\alpha}{1-2\alpha}\right)^\ell \leq \tilde{O}\left(\frac{k^2}{\epsilon^4}\right).$$

□

Remark: The above algorithm can be made to work in the case where the stream of samples is stored and is ordered adversarially, but with the proviso that the algorithm may make $P = 2\ell$ passes over the samples. This is easy to observe; reconsider the algorithm in Figure 6.2. Assume $P = 2\ell$ and set $t_1 = 10k\epsilon^{-1/\ell}$ and $t_2 = 10\epsilon^{-1/\ell}$. Each phase can be simulated in two passes. Thus $P = 2\ell$ passes is sufficient. This yields the following theorem.

Theorem 6.9. *With probability at least $1-\delta$, it is possible to compute an approximation to D within ℓ_1 distances ϵ using a single pass over $m = \tilde{O}((1.25)^\ell \ell k^2\epsilon^{-4})$ samples with $\tilde{O}(k\epsilon^{-1/\ell})$ space.*

As such, it is a strict improvement over the result in [CK06]. The basis for this improvement is primarily a tighter analysis (particularly in Theorem 6.4) and the use of the quantile estimation algorithm of [GK01]. Note that the authors of [CK06] show that $O(k^2/\epsilon^2)$ samples (and space) is sufficient for one pass, but there is a significant increase in the sample complexity in extending the algorithm to multiple passes. In our analysis this increase is much smaller, as well as the space complexity is better, which is a central point.

6.3 Importance of Random Order

One of the most important aspects of processing a stream of samples is that, as opposed to the usual streaming model, the data elements arrive in a random order. Many properties of a stream are provably hard to estimate in small space when the order of the stream is chosen adversarially. This begs the question whether some problems are not amenable to solutions in the data stream model because such streaming algorithms are necessarily forgetful or because they have to be resilient to an adversary that is potentially out to thwart them. While we have no definitive answer to this question, it does seem that relaxing the assumption that there exists an adversary ordering the data does significantly increase the range of problems that can be tackled in small space. Estimating Frequency Moments [AMS99, IW05], a classic problem from the literature in the streaming model, illustrates this point. The k-th frequency moment of a discrete distribution on n points is the k-th power of ℓ_k norm of the probability vector, i.e., $F_k = \sum_{i \in [n]} p_i^k$. The following theorem demonstrates the enormous power achieved by streaming points in a random order rather than an adversarial order.

Theorem 6.10. *It is possible to (ϵ, δ)-approximate F_k in a randomly ordered stream with $\tilde{O}((n/t)^{1-2/k})$ space when the stream length is $m = \Omega(ntk^2\epsilon^{-2-1/k}\log(n\delta^{-1}))$. In particular, if $m = \Omega(n^2k^2\epsilon^{-2-1/k}\log(n\delta^{-1}))$ then the space only depends on n*

poly-logarithmically. If the stream was ordered adversarially, the same estimation requires $\Omega(n^{1-2/k})$ *space.*

Proof. The lower bound is a generalization of the results in [CKS03]. The algorithm is simple "serialization" of [IW05] as follows. For $i = 1$ to t, let $\hat{p}_j$ be the fraction of occurrences of j among the items occurring between position $1 + (i-1)m'$ and im' where

$$m' = \Theta(nk^2(\ln(1+\epsilon/10))^{-2}\epsilon^{-1/k}\log(2n\delta^{-1})) \ .$$

Use [IW05] to compute B_i, a $(1+\epsilon/3)$ multiplicative estimate to $A_i = \sum_{j=1+(i-1)n/t}^{in/t} \hat{p}_j^k$ with probability at least $1-\delta/2$. Return $\sum_{i\in[t]} B_i$. Note that this sum can be computed incrementally rather than by storing B_i for $i \in [t]$.

We now analyze this algorithm. We will show that $\sum_{i\in[t]} A_i$ is a $(1+\epsilon/3)$ approximation to F_k with probability at least $1-\delta/2$. Subsequently it will be clear that $\sum_{i\in[t]} B_i$ is a $(1+\epsilon/3)^2 \leq (1+\epsilon)$ approximation (assuming $\epsilon < 1/4$) with probability at least $1-\delta$. By the Chernoff bound and the union bound, with probability at least $1-\delta/2$,

$$\forall j \in [n],\ |p_j - \hat{p}_j| \leq \ln\left(1+\frac{\epsilon}{10}\right) k^{-1} \max\left(p_j, \frac{(\epsilon/10)^{1/k}}{n}\right) \ .$$

Therefore, with probability at least $1-\delta/2$,

$$\frac{F_k - n^{1-k}\epsilon/10}{(1+\epsilon/10)} \leq \sum_{i\in[t]} A_i \leq (1+\epsilon/10)\left(F_k + n^{1-k}\epsilon/10\right) \ .$$

By convexity, $F_k \geq \sum_{i\in[n]}(1/n)^k = n^{1-k}$ and therefore,

$$F_k(1+\epsilon/10)^{-2} \leq \sum_{i\in[t]} A_i \leq F_k(1+\epsilon/10)^2 \ .$$

Hence $\sum_{i\in[t]} A_i$ is a $(1+\epsilon/3)$ approximation. □

We remark that [BYKS01] showed that any streaming algorithm that only samples (possibly adaptively) from the stream and returns an (ϵ,δ)-approx. of F_k, must take at least $O(n^{1-1/k}\epsilon^{-1/k}\log\delta^{-1})$ samples.

Part II

Entropy and Information Divergences

Chapter 7

Introduction

7.1 Which distances are sketchable?

One of the most basic and well studied problems in the data-stream model is the estimation of distances between two objects that are specified by the stream. A typical application would be trying to estimate the difference between the network traffic observed at two different routers. We primarily consider *update* streams where each data item is a value in the range $[n]$. A stream defines a frequency vector $(m_1, m_2, \ldots, m_n)$ where m_i is the number of times i occurs in the stream. Note that this model essentially captures the model in which a single data item can encode multiple copies of a value. This model is a generalization of the *aggregate model*, in which all updates to a given i are grouped together in the ordering of the stream.

We are interested in comparing the frequency vectors defined by different streams. We can also think of these frequency vectors defining *empirical probability distributions.*

Definition 7.1 (Empirical Distributions). *For a data stream $S = \langle a_1, \ldots, a_m \rangle$ where $a_i \in \{p, q\} \times [n]$ we define* empirical distributions *p and q as follows. Let $m(p)_i = |\{j : a_j = \langle p, i \rangle\}|$, $m(p) = |\{j : a_j = \langle p, \cdot \rangle\}|$ and $p_i = m(p)_i / m(p)$. Similarly for q.*

Estimation of distances also allows us to construct approximate representations such as histograms, wavelet decompositions, and Fourier summaries. This is because these problems can often be reduced to finding the "closest" representation in a suitable class.

7.1.1 Related Work

One of the cornerstone results in this context has been the result of Alon, Matias, and Szegedy [AMS99] that showed that it is possible to compute an (ϵ, δ)-approximation of the second frequency moment over a data stream, with insertion and deletions, defined over a domain $[n]$ using a $\text{poly}(\epsilon, \log n)$ size space. This update-able summary has come to be referred to as a "sketch" of the data. This result of estimating the ℓ_2 norm under insertion and deletion immediately allows us to estimate the ℓ_2 distance between streams. Feigenbaum et al. [FKSV02b] presented an algorithm to estimate ℓ_1 distances in the aggregate model when all updates to each p_i or q_i are grouped together. Indyk [Ind00] extended the result all ℓ_p-measures with $0 < p \leq 2$ using stable distributions. Over a sequence of papers [SS02, BYJKS02, CKS03, IW05, BGKS06], frequency moments and the associated ℓ_p distances have become well understood. Cormode et al. [CDIM03] presented algorithms for estimating Hamming distances. This work again uses stable distributions and relies on the fact that the Hamming norm is related to the ℓ_p norm for sufficiently small p. Unfortunately the stable-distribution technique and the hashing approach of Feigenbaum et al. seem to be inherently unsuited for distances that are not of the form $g(\sum_{i \in [n]} f(p_i - q_i))$ for some functions f and g.

Experience leads us to conjecture that the only distances that can be approximated in small space are those that are essentially based on norms. Several methods of creating summary representations of streams have been proposed for a variety of applications [BCFM00, CCFC02, CM05a]; in terms of distances they can be adapted to compute the Jaccard coefficient (symmetric difference over union) for two sets.

However, these methods do not usually lead to (ϵ, δ)-approximations. Are there algorithms for other commonly used distance measures? Can we characterize the set of distances that can be approximated? Are there distances that can be approximated in the aggregate-model which can not be approximated in the update-model?

7.1.2 Our Results

In this paper we take several steps towards a characterization of the distances that can be sketched. We consider "decomposable" distances, i.e., distances $d : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^+$ for which there exists a $\phi : \mathbb{R} \times \mathbb{R} \to \mathbb{R}^+$ such that $d(x, y) = \sum_{i \in [n]} \phi(x_i, y_i)$. We prove the *Shift Invariant Theorem* for non-estimability of arbitrary decomposable distances over multiple passes. This shows that there is a fundamental difficulty in approximating any difference measure which is not "flat" in the dense that $d(x, y)$ can be very different for $d(x + v, y + v)$. That condition does not apply to distances induced by *norms*, where the distance is a function of the difference of the two vectors; hence the positive results for the ℓ_1 and ℓ_2 norms.

7.2 The Information Divergences.

Applications in pattern matching, image analysis, statistical learning, etc., use distances which are not ℓ_p norms. Several distances[1] such as the Kullback-Leibler and Hellinger divergences are widely used to estimate the distances between distributions, and have a long history of study in statistics and information theory literature. We will discuss two very broad classes of distance measures (1) f-divergences, which are central to statistical tests and, (2) Bregman Divergences, which are used for finding optimal models using mathematical programming.

We first discuss the Ali-Silvey distances or f-divergences, discovered independently by Csiszár [Csi91], and Ali and Silvey [AS66]. These are defined as follows.

[1]Several of the "distances" used are not metric and, consequently, are usually referred to as divergences.

ℓ_1 distance:	$f(u) = \|1-u\|$
Kullback-Liebler (KL) divergence:	$f(u) = u \ln u$
Jensen-Shannon (JS) divergence:	$f(u) = \ln(2/(1+u)) + u\ln(2u/(1+u))$
Hellinger divergence:	$f(u) = (\sqrt{u}-1)^2$
χ^2 divergence:	$f(u) = (u-1)^2$
Triangle divergence:	$f(u) = (u-1)^2/(u+1)$.

Table 7.1: Common f-divergences

Definition 7.2 (f-Divergences or Ali-Silvey-Csiszár divergences)**.** *A convex function $f : (0,\infty) \to \mathbb{R}$ such that $f(1) = 0$ gives rise to an* f-divergence*,*

$$\mathcal{D}_f(p, q) = \sum p_i f(q_i/p_i)$$

where $f(0) = \lim_{t\to 0} f(t)$, $0f(0/0) = 0$, and $0f(a/0) = a \lim_{u\to\infty} f(u)/u$.

The family of f-divergences include many commonly used information theoretic distances, e.g., the Kullback-Liebler (KL) divergence and its symmetrization, the Jensen-Shannon (JS) divergence; Matsusita's divergence or the squared Hellinger divergence; theχ^2 divergence and its symmetrization, the Triangle divergence. See Table 7.1.

The quantity q_i/p_i is called the "likelihood ratio" and a fundamental aspect of these measures is that these divergences are tied to "ratio tests" in Neyman-Pearson style hypothesis testing [CT91]. Several of these divergences appear as exponents of error probabilities for optimal classifiers, e.g., in Stein's Lemma. Results of Csiszár [Csi91], Liese and Vajda [LV87], and Amari [Ama85, AN00], show that f-divergences are the unique class of distances on distributions that arise from a fairly simple set of axioms, e.g., permutation invariance, non-decreasing local projections, certain direct-sum theorems etc. In many ways these divergences are "natural" to distributions and statistics, much the same way that ℓ_2 is a natural measure for points in $\mathbb{R}^n$. Moreover, all of these distances are related to each other (via the Fisher information matrix) [Č81] in a way that other plausible measures (most notably ℓ_2) are not.

A major reason for investigating these f-divergences lies in loss functions used in statistical learning. The ℓ_1 distance captures the "hinge loss" and the other divergences are geared towards non-linear losses. To understand the connection better, we need to also discuss the connections between f-divergences and Bregman divergences. The general family of "arcing" [Bre99] and "AnyBoost" [MBBF99] family of algorithms fall into a constrained convex programming framework introduced earlier by Bregman [Bre67]. Friedman, Hastie and Tibshirani [FHT00] established the connection between boosting algorithms and logistic loss, and subsequently over a series of papers [LPP97, Laf99, KW99, CSS02], the study of Bregman divergences and information geometry has become the method of choice for studying exponential loss functions. The connection between loss functions and f-divergences are investigated have been recently by Nguyen, Wainright, and Jordan [NWJ05].

Definition 7.3 (Bregman Divergences)**.** *A strictly convex function $F : (0, 1] \to \mathbb{R}$ gives rise to (decomposable)* Bregman Divergence*,*

$$\mathcal{B}_F(p, q) = \sum_i \left[F(p_i) - F(q_i) - (p_i - q_i)F'(q)\right] \;\; ,$$

where F' denotes the first derivative of F.

Note that Bregman divergences are typically extended to the positive cone of $\mathbb{R}^n$, beyond the probability simplex. The most well-known Bregman divergence is ℓ_2^2. The Kullback–Leibler divergence is also a Bregman divergence, as well as the Itakura–Saito divergence. See Table 7.2.

The main use of Bregman divergences is in finding optimal models. Given a distribution q we are interested in finding a p that best matches the data, and this is posed as a convex optimization problem $\min_p \mathcal{B}_F(p, q)$. It is easy to verify that a linear combination of Bregman divergences is a Bregman divergence and that the Bregman balls are convex in the first argument (but often not in the second). Furthermore there is a natural convex duality between the optimum representation

Kullback-Liebler (KL) divergence:	$F(z) = z \log z$
ℓ_2^2 divergence:	$F(z) = z^2$
Itakura–Saito divergence:	$F(z) = -\log z$

Table 7.2: Common Bregman divergences

p^* under $\mathcal{B}_F$, and the divergence $\mathcal{B}_F$. This connection to convex optimization is one of the many reasons for the increasing use of Bregman divergences in the learning literature.

Giving the important of both these information divergences, it is natural to ask whether they can be approximated in the data stream model.

7.2.1 Our Results

Unfortunately, our first results are negative but they help us understand why the ℓ_1 and ℓ_2 distances are special among the f- and Bregman-divergences in the context of streaming.

- For all f-divergences with f twice-differentiable and f'' strictly positive, no polynomial factor approximation of $\mathcal{D}_f(p, q)$ is possible in sub-linear space. Note that for ℓ_1, which can be sketched, is realized by the function $f(u) = |u - 1|$. Hence, the lower-bound does not apply to since f'' is not defined at 1.

- For all Bregman divergences $\mathcal{B}_F$ where F is twice differentiable and there exists $\rho, z_0 > 0$ such that,

  $$\forall\, 0 \le z_2 \le z_1 \le z_0,\ \frac{F''(z_1)}{F''(z_2)} \ge \left(\frac{z_1}{z_2}\right)^\rho \text{ or } \forall\, 0 \le z_2 \le z_1 \le z_0,\ \frac{F''(z_1)}{F''(z_2)} \le \left(\frac{z_2}{z_1}\right)^\rho .$$

  then no polynomial factor approximation of $\mathcal{B}_F$ is possible in sub-linear space. This condition effectively states that $F''(z)$ vanishes or diverges monotonically, and polynomially fast, as $z \to 0$. Note that for ℓ_2^2, which can be sketched, the function $F(z) = z^2$ and F'' is constant.

An interesting specific case of the above results is that the f-divergence Hellinger can not be approximated up to any polynomial factor. However, if the stream had been an aggregate stream in which identical data items are grouped together than an (ϵ, δ)-approximation is possible in poly-logarithmic space via an algorithm for estimating ℓ_2 [AMS99].

We next consider additive approximations.

- There exists an (ϵ, δ)-additive-approx. for $\mathcal{D}_f$ in $O(\epsilon^{-2} \log \delta^{-1})$ space if $\mathcal{D}_f$ is bounded. Furthermore, $\Omega(\epsilon^{-2})$ space can be shown to be necessary. Alternatively, if $\mathcal{D}_f$ is unbounded then any $(\epsilon, 1/4)$-additive-approx. requires $\Omega(n)$ space for any constant $\epsilon > 0$.

- There exists an (ϵ, δ)-additive-approx. for $\mathcal{B}_F$ in $O(\epsilon^{-2} \log \delta^{-1})$ space if F and F'' are bounded in the range $(0, 1]$. If $F(0)$ or $F'(0)$ is unbounded, then any $(\epsilon, 1/4)$-additive-approx. requires $\Omega(n)$ space for any constant ϵ.

Lastly we present testing algorithms information divergences in the oracle models introduced in Chapter 1.

- There exists an $(\epsilon, \epsilon/2, \delta)$-tester for estimating any bounded f-divergence in the combined-oracle model. We prove a matching lower bound thereby showing optimality.

- We also present a sub-linear tester in the generative model for a range of f-divergences including Hellinger and Jensen-Shannon. The algorithm makes $\tilde{O}(\epsilon^{-4} n^{2/3})$ queries. This answers an open question posed in [BFR+00].

7.3 Entropy

Closely related to the information divergences is the *entropy* of a distribution.

Definition 7.4 (Entropy). *The entropy of a distribution is defined as*

$$H(p) = \sum_i -p_i \lg p_i \ .$$

For example, it is related to the Jensen-Shannon and Kullback-Liebler divergences;

$$\begin{aligned} \mathrm{JS}(p,q) &= \ln 2 \left(2H\left(\frac{p+q}{2}\right) - H(p) - H(q) \right) \\ \mathrm{KL}(p,u) &= \ln 2 \left(\lg n - H(p) \right) \end{aligned}$$

where u is the uniform distribution and $(p+q)/2$ is the distribution whose i-th component is $(p_i + q_i)/2$.

Entropy is quantity that was originally introduced by Shannon [Sha48]. It captures the "information content" of a random event. For example, it can be used to lower-bound the compressibility of data and plays a fundamental role in the coding and information theory. Recently, it has been used in networking applications [GMT05, WP05, XZB05] where it can be useful when trying to detect anomalous behavior.

7.3.1 Related Work

Guha, McGregor and Venkatasubramanian [GMV06] gave a constant factor approximation for constant H as well as (ϵ, δ)-approximations for H, using space that depends on the value of H. Chakrabarti, Do Ba and Muthukrishnan [CDM06] gave a one pass algorithm for approximating H with sublinear but polynomial in m space, as well as a two-pass algorithm requiring only poly-logarithmic space.

In the networking world, the problem of approximating the entropy of a stream was considered in Lall et al. [LSO$^+$06]. They focused on estimating the entropy norm,

$$F_H := \sum_{i=1}^{n} m_i \lg m_i \ .$$

Clearly F_H and H are closely related and we can write $F_H = m \lg m - mH$. However, [LSO$^+$06] made certain assumptions about the distribution defined by the stream and these ensured that computing H based on their estimate of F_H would give accurate results. Both this paper and [CDM06], use the AMS-sampling procedure described in Chapter 2.

Most recently, Bhuvanagiri and Ganguly [BG06] described an algorithm that can approximate H in poly-logarithmic space in a single pass. The algorithm is based on the same ideas and techniques as recent algorithms for optimally approximating frequency moments [IW05, BGKS06], and can tolerate streams in which previously observed items are removed. The exact space bound is

$$O\left(\epsilon^{-3}(\log^4 m)(\log \delta^{-1})\frac{\log m + \log n + \log \epsilon^{-1}}{\log \epsilon^{-1} + \log\log m}\right) ,$$

which is suboptimal in its dependency on ϵ, and has high cost in terms of $\log m$.

7.3.2 Our Results

Our results include the following results for processing adversarially ordered streams.

- There exists a single-pass, $O(\epsilon^{-2}\log(\delta^{-1})\log m)$-space, (ϵ, δ)-approximation for entropy. This improves over previous work on this problem [CDM06, GMV06, LSO$^+$06, BG06]. The algorithm uses a novel extension of a method introduced by Alon, Matias, and Szegedy [AMS99]. In effect, this extension allows us to transform the two-pass algorithm of Chakrabarti, Do Ba and Muthukrishnan [CDM06] into a single pass algorithm. We believe that this technique may have other applications. We show that our algorithm is essentially optimal by proving that any $(\epsilon, 1/4)$-approximation requires $\Omega(\epsilon^{-2}/\log^2 \epsilon^{-1})$ space.

- Any $(\epsilon, 1/4)$-approximation of the k-th order entropy ($k > 0$) of a streams requires $\Omega(n/\log n)$ space where the k-th order entropy of a stream is a generalization of the entropy that quantifies how easy it is to predict a character of the stream given the previous k characters. However, we present an (ϵ, δ)-addtive-approximation using $O(k^2 \epsilon^{-2} \log(\delta^{-1}) \log^2 n \log^2 m)$ space.

- There exists an (ϵ, δ)-approximation algorithm for estimating the *unbiased random walk entropy*, a natural quantity related to the first order entropy of an unbiased walk on an undirected graph. Our algorithm uses $O(\epsilon^{-4} \log n)$ space. This algorithm can also be implemented in the graph streaming model.

Lastly, we present the following result for testing entropy in the combined-oracle model.

- There exists an $(\epsilon, \epsilon/2, \delta)$-tester for entropy in the combined oracle model using $O(\epsilon^{-1} \log n)$ queries. This matches the lower bound in [BDKR05] and is therefore optimal.

Chapter 8

Information Divergences

Chapter Outline: In this chapter, we present the technical details behind the results related to approximating the information divergence between two empirical distributions defined by a stream. We start by establishing some preliminary results about the geometry of the relevant divergences. We then consider multiplicative approximation and prove the "Shift-Invariant Theorem" which characterizes a set of distance measures that can not be approximated in the streaming model with small space. In the next section, we present algorithms and lower-bounds for additive approximation. Finally, we present algorithms and lower-bounds for testing f-divergences in various oracle models. For background see Chapter 7.

8.1 Geometric Preliminaries

In this section, we present some simple geometric results that will allow us to make certain useful assumptions about the f or F defining an f-divergence or Bregman divergence.

We start by defining a *conjugate* $f^*(u) = uf(1/u)$ and note that we can write any f-divergence as,

$$D_f(p,q) = \sum_{i:p_i>q_i} p_i f(q_i/p_i) + \sum_{i:q_i>p_i} q_i f^*(p_i/q_i) \ .$$

The following lemma that demonstrates that we may assume that $f(u) \in [0, f(0)]$ and $f^*(u) \in [0, f^*(0)]$ for $u \in [0,1]$ where both $f(0) = \lim_{u\to 0} f(u)$ and $f^*(0) = \lim_{u\to 0} f^*(u)$ exist if f is bounded.

Lemma 8.1. *Let f be a real-valued function that is convex on $(0,\infty)$ and satisfies $f(1) = 0$. Then there exists a real-valued function g that is convex on $(0,\infty)$ and satisfies $g(1) = 0$ such that*

1. *$\mathcal{D}_f(p,q) = \mathcal{D}_g(p,q)$ for all distributions p and q.*
2. *g is decreasing in the range $(0,1]$ and increasing in the range $[1,\infty)$. In particular, if f is differentiable at 1 then $g'(1) = 0$.*
3. *$g(0) = \lim_{u\to 0} g(u)$ and $g^*(0) = \lim_{u\to 0} g^*(u)$ exist if $\mathcal{D}_f$ was bounded.*

Proof. Note that D_f bounded implies that $f(0) = \lim_{u\to 0} f(u)$ exists. Otherwise the

$$D_f((1/2,1/2),(0,1)) = 1/2(f(0) + f(2))$$

would not be finite. Similarly $f^*(0) = \lim_{u\to 0} f^*(u) = \lim_{u\to 0} uf(1/u)$ exists because otherwise

$$D_f((0,1),(1/2,1/2)) = 0.5 \lim_{u\to\infty} f(u)/u + f(1/2) = 0.5 \lim_{u\to 0} uf(1/u)/u + f(1/2)$$

would not be finite. Let $c = -\lim_{u\to 1^-} f(u)/(1-u)$. This limit exists because f is convex and defined on $(0,\infty)$.Then $g(u) = f(u) - c(u-1)$ satisfies the necessary conditions. □

For example, the Hellinger divergence can be realized by either $f(u) = (\sqrt{u}-1)^2$ or $f(u) = 2 - 2\sqrt{u}$. The next lemma shows that if we are willing to tolerate an additive approximation we may make certain assumptions about the derivative of f.

Lemma 8.2. *Given a bounded $\mathcal{D}_f$ and $\epsilon \in (0,1)$, let u_0 satisfy*

$$\min\left(f(u_0)/f(0), f^*(u_0)/f^*(0)\right) \geq 1-\epsilon \ ,$$

and define g as follows:

$$g(u) = \begin{cases} f(u) & \text{for } u \in (u_0, 1/u_0) \\ f(0) - u(f(0)-f(u_0))/u_0 & \text{for } u \in [0, u_0] \\ uf^*(0) - (f^*(0) - f^*(u_0))/u_0 & \text{for } u \in [1/u_0, \infty) \end{cases}$$

Then,

1. *$\mathcal{D}_g(P,Q)(1-\epsilon) \leq \mathcal{D}_f(P,Q) \leq \mathcal{D}_g(P,Q)$.*

2. $\max_u |g'(u)| \leq \max\left(\frac{f(0)-f(u_0)}{u_0}, f^*(0)\right)$.

3. $\max_u |g^{*\prime}(u)| \leq \max\left(\frac{f^*(0)-f^*(u_0)}{u_0}, f(0)\right)$.

Proof. Because f, g, f^*, and g^* are decreasing in the range $[0,1]$, $1 \leq g(u)/f(u) \leq f(0)/f(u_0)$ and $1 \leq g^*(u)/f^*(u) \leq f^*(0)/f^*(u_0)$ for $u \in [0,1]$. The first claim follows by the assumption on u_0. To bound the derivatives, note that $g(u)$ and $g^*(u)$ are convex and hence the absolute value of the derivative in maximized at $u = 0$ or $u \to \infty$. The remaining claims follow by taking the derivative at these points. $\square$

Note that $\lim_{u\to 0} |g'(u)|$ is bounded whereas $\lim_{u\to 0} |f'(u)|$ need not be bounded. For the Hellinger divergence, $f(u) = (\sqrt{u}-1)^2$ and $f'(u) = (\sqrt{u}-1)/\sqrt{u}$. Similar to Lemma 8.1, the following lemma demonstrates that, without loss of generality, we may make various assumptions about the F that defines a Bregman divergence.

Lemma 8.3. *Let F be a differentiable, real valued function that is strictly convex on $(0,1]$ such that $\lim_{u\to 0^+} F(u)$ and $\lim_{u\to 0^+} F'(u)$ exist. Then, there exists a differentiable, real valued function G that is strictly convex on $(0,1]$ and,*

1. *$\mathcal{B}_F(p,q) = \mathcal{B}_G(p,q)$ for all distributions p and q.*

2. *$G(z) \geq 0$ for $x \in (0,1]$ and G is increasing in the range $(0,1]$.*

3. $\lim_{u\to 0^+} G'(u) = 0$ *and* $\lim_{u\to 0^+} G(u) = 0$.

Proof. The function $G(z) = F(z) - F'(0)z - F'(0)$ satisfies the conditions. $\square$

8.2 Multiplicative Approximations

We start with the central theorem of this section, the *Shift Invariance Theorem.* This theorem characterizes a large class of divergences that are not sketchable.

Theorem 8.4 (Shift Invariance Theorem)**.** *Let* $\phi : [0,1]^2 \to \mathbb{R}^+$ *satisfy* $\phi(x,x) = 0$ *for all* $x \in [0,1]$ *and there exists* $n_0, a, b, c \in \mathbb{N}$ *such that for all* $n \geq n_0$,

$$\max\left(\phi\left(\frac{a}{m}, \frac{a+c}{m}\right), \phi\left(\frac{a+c}{m}, \frac{a}{m}\right)\right) > \frac{\alpha^2 n}{4}\left(\phi\left(\frac{b+c}{m}, \frac{b}{m}\right) + \phi\left(\frac{b}{m}, \frac{b+c}{m}\right)\right)$$

where $m = an/4 + bn + cn/2$*. Then any algorithm that returns an* α *approximation of* $d(p,q) = \sum_{i\in[5n/4]} \phi(p_i, q_i)$ *with probability at least* $3/4$ *where* p *and* q *are defined by a stream of length* $O((a+b+c)n)$ *over* $[5n/4]$ *requires* $\Omega(n)$ *space.*

Proof. We refer the reader to the lower bounds template given in Chapter 2. Assume that n is divisible by 4 and $n > n_0$. Let $(x,y) \in \mathbb{F}_2^n \times \mathbb{F}_2^n$ be an instance of Set-Disjointness where $\sum_i x_i = \sum_i y_i = n/4$. Alice and Bob determine the prefix of a stream $S_A(x)$ and the suffix $S_B(y)$ respectively. We first assume that $\phi(a/m, (a+c)/m) \geq \phi((a+c)/m, a/m)$.

$$\begin{aligned} S_A(x) &= \bigcup_{i\in[n]} \{ax_i + b(1-x_i) \text{ copies of } \langle p, i\rangle \text{ and } \langle q, i\rangle\} \\ &\qquad \cup \bigcup_{i\in[\frac{n}{4}]} \{b \text{ copies of } \langle p, i+n\rangle \text{ and } \langle q, i+n\rangle\} \\ S_B(y) &= \bigcup_{i\in[n]} \{cy_i \text{ copies of } \langle q, i\rangle\} \cup \bigcup_{i\in[\frac{n}{4}]} \{c \text{ copies of } \langle p, i+n\rangle\} \end{aligned}$$

Observe that $m(p) = m(q) = an/4 + bn + cn/2$ and

$$\mathcal{D}_f(p,q) = (x.y)\phi\left(\frac{a}{m}, \frac{a+c}{m}\right) + (n/4 - x.y)\phi\left(\frac{b}{m}, \frac{b+c}{m}\right) + (n/4)\phi\left(\frac{b+c}{m}, \frac{b}{m}\right) .$$

Therefore,

$$\begin{aligned} x.y = 0 &\Leftrightarrow \mathcal{D}_f(p,q) = (n/4)(\phi(b/m,(b+c)/m) + \phi((b+c)/m, b/m)) \\ x.y = 1 &\Leftrightarrow \mathcal{D}_f(p,q) \geq \alpha^2(n/4)(\phi(b/m,(b+c)/m) + \phi((b+c)/m, b/m)) \end{aligned}$$

Therefore any α-approximation would determine the value of $x.y$ and hence an α-approximation requires $\Omega(n)$ space [Raz92]. If $\phi(a/m,(a+c)/m) \leq \phi((a+c)/m, a/m)$ then the proof follows by reversing the roles of p and q. $\square$

The above theorem suggests that unless $\phi(x_i, y_i)$ is some function of $x_i - y_i$ then the distance is not sketchable. The result holds even if the algorithm may take a constant number of passes over the data. We also mention a simpler result that can be proved using similar ideas to those employed above. This states that if there exist $a, b, c \in \mathbb{N}$ such that

$$\max\left(\frac{\phi(a+c,a)}{\phi(b+c,b)}, \frac{\phi(a,a+c)}{\phi(b,b+c)}\right) > \alpha^2 \ ,$$

then any single-pass α-approximation of $\sum_{i\in[n]} \phi(m(p)_i, m(q)_i)$ requires $\Omega(n)$ space.

We next present two corollaries of Theorem 8.4. These characterize the f-divergences and Bregman divergences that can be not be sketched. Note that ℓ_1 and ℓ_2^2, which can be sketched, are the only commonly used divergences that do not satisfy the relevant conditions.

Corollary 8.5 (f-Divergences)**.** *Given an f-divergence $\mathcal{D}_f$, if f is twice differentiable and f'' is strictly positive, then no polynomial factor approximation of $\mathcal{D}_f$ is possible in sub-linear space.*

Proof. We first note that by Lemma 8.1 we may assume $f(1) = f'(1) = 0$. Let $a = c = 1$ and $b = \alpha^2 n(f''(1)+1)/(8f(2))$ where α is an arbitrary polynomial in n. Note that $f(2) > 0$ because f is strictly convex.

We start by observing that,

$$\phi(b/m,(b+c)/m) = (b/m)f(1+1/b) = (b/m)\left[f(1) + \frac{1}{b}f'(1) + \frac{1}{2!b^2}f''(1+\gamma)\right]$$

for some $\gamma \in [0, 1/b]$ by Taylor's Theorem. Since $f(1) = f'(1) = 0$ and $f''(t)$ is continuous at $t = 1$ this implies that for sufficiently large n, $f''(1+\gamma) \leq f''(1) + 1$ and so,

$$\phi(b/m, (b+c)/m) \leq \frac{f''(1)+1}{2mb} = \frac{f''(1)+1}{2f(2)b} m^{-1} f(2) \leq \frac{8}{\alpha^2 n} \phi(a/m, (a+c)/m) \ .$$

Similarly we can show that for sufficiently large n,

$$\phi((b+c)/m, b/m) \leq \frac{8}{\alpha^2 n} \phi(a/m, (a+c)/m) \ .$$

Then appealing to Theorem 8.4 we get the required result.

□

Corollary 8.6 (Bregman Divergences)**.** *Given a Bregman divergences $\mathcal{B}_F$, if F is twice differentiable and there exists $\rho, z_0 > 0$ such that,*

$$\forall\ 0 \leq z_2 \leq z_1 \leq z_0,\ \frac{F''(z_1)}{F''(z_2)} \geq \left(\frac{z_1}{z_2}\right)^\rho \ \textit{or}\ \forall\ 0 \leq z_2 \leq z_1 \leq z_0,\ \frac{F''(z_1)}{F''(z_2)} \leq \left(\frac{z_2}{z_1}\right)^\rho$$

then no polynomial factor approximation of $\mathcal{B}_F$ is possible in sub-linear space.

This condition effectively states that $F''(z)$ vanishes or diverges monotonically, and polynomially fast, as $z \to 0$.

Proof. By the Mean-Value Theorem, for any $m, r \in \mathbb{N}$, there exists $\gamma(r) \in [0,1]$ such that, $\phi(r/m, (r+1)/m) + \phi((r+1)/m, r/m) = m^{-2} F''((r + \gamma(r))/m)$. Therefore, for any $a, b \in \mathbb{N}, c = 1$ and $m = an/4 + bn + n/2$,

$$\frac{\max\left(\phi\left(\frac{a}{m}, \frac{a+c}{m}\right), \phi\left(\frac{a+c}{m}, \frac{a}{m}\right)\right)}{\phi\left(\frac{b+c}{m}, \frac{b}{m}\right) + \phi\left(\frac{b}{m}, \frac{b+c}{m}\right)} \geq \frac{1}{2} \frac{F''((a+\gamma(a))/m)}{F''((b+\gamma(b))/m)} \ .$$

If $\forall\ 0 \leq z_2 \leq z_1 \leq z_0,\ F''(z_1)/F''(z_2) \geq (z_1/z_2)^\rho$ then set $a = (\alpha^2 n)^{1/\rho}$ and $b = 1$ where α is an arbitrary polynomial in n. If $\forall\ 0 \leq z_2 \leq z_1 \leq z_0,\ F''(z_1)/F''(z_2) \geq (z_2/z_1)^\rho$ then set $a = 1$ and $b = (\alpha n)^{1/\rho}$. In both cases we deduce that the RHS of Eqn. 8.1 is greater than $\alpha^2 n/4$. Hence, appealing to Theorem 8.4, we get the required result. □

8.3 Additive Approximations

In this section we focus on additive approximations. As mentioned earlier, the probability of misclassification using ratio tests is often bounded by $2^{-\mathcal{D}_f}$, for certain $\mathcal{D}_f$. Hence, an additive ϵ approximation translates to a multiplicative 2^ϵ factor for computing the error probability.

Our goal is the characterization of divergences that can be approximated additively. We first present a general algorithmic result and then prove two general lower-bounds. In the subsequent sections, we consider f-divergences and Bregman divergences in particular.

Theorem 8.7. *There exists an (ϵ, δ)-additive-approximation for*

$$d_\phi(p,q) = \sum_{i \in [n]} \phi(p_i, q_i)$$

(we assume that $\phi(0,0) = 0$) using $O(\tau\epsilon^{-2}\log\delta^{-1})$ space where

$$\tau = 4 \max_{x,y \in [0,1]} \left(\left| \frac{\partial}{\partial x}\phi(x,y) \right| + \left| \frac{\partial}{\partial y}\phi(x,y) \right| \right) .$$

The algorithm does not need to know $m(p)$ or $m(q)$ in advance.

Proof. We will describe a basic estimator that can be computed in small space without prior knowledge of $m(p)$ or $m(q)$. We will then argue that the estimator is correct in estimation. Finally, we show that, by averaging a small number of independent basic estimators, we may return a sufficiently accurate estimator with the necessary probability.

Let $d \in_R \{p, q\}$ and $j_d \in_R [m(d)]$. Let $a_j = \langle d, k \rangle$ be the j_d-th element in the stream of the form $\langle d, \cdot \rangle$ and compute

$$\begin{aligned} r &= I[d = p] \cdot r_0 + |\{\ell > j : a_k = \langle p, k \rangle\}| \cdot m(q) \\ s &= I[d = q] \cdot s_0 + |\{\ell > j : a_k = \langle q, k \rangle\}| \cdot m(p) \end{aligned}$$

where $r_0 \in_R [m(q)]$ and $s_0 \in_R [m(p)]$. We are now ready to define the basic estimator

$$X(r,s) = 2m^* \begin{cases} \phi(r/m^*, s/m^*) - \phi(r/m^* - 1/m^*, s/m^*) & \text{if } d = p \\ \phi(r/m^*, s/m^*) - \phi(r/m^*, s/m^* - 1/m^*) & \text{if } d = q \end{cases}$$

where $m^* = m(p)m(q)$.

Note that $\Pr[k = i] = (p_i + q_i)/2$ and that

$$E[X(r,s)|k=i] = 2\left(\frac{m^*\phi(m(p)_i m(q)/m^*, m(q)_i m(p)/m^*)}{m(p)m(q)_i + m(q)m(p)_i}\right) = \frac{2\phi(p_i,q_i)}{p_i+q_i}$$

therefore $E[X(r,s)] = \sum_i \phi(p_i,q_i)$ as required.

$$|X(r,s)| \le 2\max\left\{\max_{x\in\left[\frac{r-1}{m^*},\frac{r}{m^*}\right]}\left|\frac{\partial}{\partial x}\phi(x,s/m^*)\right|, \max_{y\in\left[\frac{s-1}{m^*},\frac{s}{m^*}\right]}\left|\frac{\partial}{\partial y}\phi(r/m^*,y)\right|\right\} \le \tau$$

Hence, averaging $O(\tau\epsilon^{-2}\log\delta^{-1})$ independent basic estimators results in an (ϵ,δ)-additive-approx. □

Theorem 8.8. *Any $(\epsilon, 1/4)$-additive approx.*

$$d_\phi(p,q) = \sum_{i\in[n]} \phi(p_i,q_i)$$

requires $\Omega(\epsilon^{-2})$ space if,

$$\exists a,b>0, \forall x,\ \phi(x,0) = ax, \phi(0,x) = bx, \textit{and } \phi(x,x) = 0.$$

Proof. We refer the reader to the lower bounds template given in Chapter 2. Let $(x,y) \in \mathbb{F}_2^n \times \mathbb{F}_2^n$ be an instance of Gap-Hamdist where $n = \lfloor \epsilon^{-2} \rfloor$ and the weight of x and y is cn. Then define p and q by the following stream elements.

$$S_A(x) = \{x_i \text{ copies of } \langle p,i\rangle \text{ for } i \in [n]\}$$
$$S_B(y) = \{y_i \text{ copies of } \langle q,i\rangle \text{ for } i \in [n]\}$$

Then

$$d_\phi(p,q) = \frac{a|\{i : x_i = 1, y_i = 0\}|}{cn} + \frac{b|\{i : x_i = 0, y_i = 1\}|}{cn} = d_H(x,y)\frac{a+b}{cn},$$

where $d_H(x,y)$ is the Hamming distance between x and y. Therefore a $\frac{(a+b)\epsilon}{2c}$-additive approximate determines whether $d_H(x,y) \le n/2$ or $d_H(x,y) \ge n/2 + \sqrt{n}$. □

Theorem 8.9. *Any $(\epsilon, 1/4)$-additive approx. of $d_\phi(p,q) = \sum_{i\in[n]} \phi(p_i, q_i)$ requires $\Omega(n)$ space if either $\phi(x,0)$ or $\phi(0,x)$ is unbounded for all $x > 0$. This applies even if one of the distributions is known to be uniform.*

Proof. We refer the reader to the lower bounds template given in Chapter 2. Let $(x,y) \in \mathbb{F}_2^n \times \mathbb{F}_2^n$ be an instance of SET-DISJOINTNESS. Then define q by the following stream elements.

$$S_A(x) = \{1 - x_i \text{ copies of } \langle q, i\rangle \text{ for } i \in [n]\}$$

$$S_B(y) = \{1 - y_i \text{ copies of } \langle q, i\rangle \text{ for } i \in [n]\}$$

Let p be the uniform distribution. If $\phi(1/n, 0)$ is unbounded then $d_\phi(p,q)$ is finite iff $x.y = 0$. If $\phi(1/n, 0)$ is unbounded then $d_\phi(p,q)$ is finite iff $x.y = 0$. □

8.3.1 Additive Approximation for f-divergences

In this section we show that $\mathcal{D}_f(p,q)$ can be additively approximated up to any additive $\epsilon > 0$ if and only if $\mathcal{D}_f$ is bounded.

Theorem 8.10. *There exists a one-pass, $O(\epsilon^{-2}\log\delta^{-1})$-space, (ϵ,δ)-additive-approx. for any bounded f-divergence. The algorithm does not need to know $m(p)$ or $m(q)$ in advance.*

Proof. We appeal to Theorem 8.7 and note that,

$$\max_{x,y\in[0,1]} \left(\left|\frac{\partial}{\partial x}\phi(x,y)\right| + \left|\frac{\partial}{\partial y}\phi(x,y)\right| \right) = \max_{x,y\in[0,1]} \left(|f(y/x) - (y/x)f'(y/x)| + |f'(y/x)|\right)$$
$$= \max_{u\in[0,1]} \left(\left|f^{*\prime}(u)\right| + |f'(u)|\right)$$

By Lemma 8.2 we may assume this is independent of m and n. The result follows. □

We complement the above theorem with the following lower-bound that follows from Theorem 8.9 and Theorem 8.8.

Theorem 8.11. *Any $(\epsilon, 1/4)$-additive-approximation of an unbounded $\mathcal{D}_f$ requires $\Omega(n)$ space. This applies even if one of the distributions is known to be uniform. Any $(\epsilon, 1/4)$-additive-approximation of a bounded $\mathcal{D}_f$ requires $\Omega(\epsilon^{-2})$ space.*

8.3.2 Additive Approximation for Bregman divergences

In this section we proof a partial characterization of the Bregman divergences that can be additively approximated.

Theorem 8.12. *There exists a one-pass, $O(\epsilon^{-2}\log\delta^{-1})$-space, (ϵ, δ)-additive-approx. of a Bregman divergence if F and F'' are bounded in the range $[0, 1]$. The algorithm does not need to know $m(p)$ or $m(q)$ in advance.*

Proof. We appeal to Theorem 8.7 and note that,

$$\max_{x,y\in[0,1]}\left(\left|\frac{\partial}{\partial x}\phi(x,y)\right| + \left|\frac{\partial}{\partial y}\phi(x,y)\right|\right) = \max_{x,y\in[0,1]}\left(|F'(x) - F'(y)| + |x-y|F''(y)\right) .$$

We may assume this is constant by convexity of F and the assumptions of the theorem. The result follows. □

The next theorem follows immediately from Theorem 8.9.

Theorem 8.13. *If $F(0)$ or $F'(0)$ is unbounded then an $(\epsilon, 1/4)$-additive-approx. of $\mathcal{B}_F$ requires $\Omega(n)$ space even if one of the distributions is known to be uniform.*

8.4 Testing f-Divergences in the Oracle Models

In this section, we consider approximation and testing of the f-divergence between two distributions in the oracle models introduced in Chapter 1. Recall that the *generative model* supports a `sample`(p) query that returns i with probability p_i. In the *evaluative model*, a `probe`(p, i) query returns the value p_i. Lastly, the *combined model* supports both the `sample` and `probe` operations. In all three models, the complexity of an algorithm is measured by the number of oracle queries.

8.4.1 f-Divergences Testing (Generative Oracle)

In this section we consider testing in the generative model for various f-divergences. We will present the results for the Hellinger distance. However, the Jensen-Shannon and triangle divergences are constant factor related to the Hellinger distance:

$$\text{Hellinger}(p,q)/2 \le \Delta(p,q)/2 \le JS(p,q) \le \ln(2)\ \Delta(p,q) \le 2\ln(2)\ \text{Hellinger}(p,q)\ . \tag{8.1}$$

Parts of Eqn. 8.1 are proved in [Top00] and the other inequalities follow from the AM-GM inequality. Therefore a result for Hellinger naturally implies analogous results forJensen-Shannon and triangle. Our algorithm is similar to that in [BFR$^+$00], and is presented in Figure 8.1. It relies on an ℓ_2 tester given in [BFR$^+$00]. Central to the analysis are the following inequalities.

$$\frac{\ell_2^2(p,q)}{2(\ell_\infty(p)+\ell_\infty(q))} \le \text{Hellinger}(p,q) \le \ell_1(p,q) \le \sqrt{n}\ell_2(p,q)\ . \tag{8.2}$$

Lemma 8.14 (ℓ_2 Testing [BFR$^+$00])**.** *There exists an $(\epsilon,\epsilon/2,\delta)$-tester for $\ell_2(p,q)$ using $O(\epsilon^{-4}(b^2+\epsilon^2\sqrt{b})\log\delta^{-1})$ samples where $b=\max(\ell_\infty(p),\ell_\infty(q))$.*

The first prove two preliminary lemmas.

Lemma 8.15. *Define $\tilde{p}_i = m_i^p/m$ and $\tilde{q}_i = m_i^q/m$. Let $\gamma\in(0,1)$. With $m = O(\gamma^{-4}n^\alpha\log(n\delta^{-1}))$ samples, with probability $1-\delta/2$, the following two conditions hold:*

$$\begin{aligned}&\forall i\notin S,\ p_i,q_i \le 2n^{-\alpha}\\ &\forall i\in S,\ |(\sqrt{p_i}-\sqrt{q_i})^2-(\sqrt{\tilde{p}_i}-\sqrt{\tilde{q}_i})^2| \le \gamma\max\{p_i,q_i\}/100\ .\end{aligned}$$

Proof. By applying Chernoff-Hoeffding bounds it is straight-forward to show that with probability at least $1-\delta/2$,

$$\forall i\in[n],\ |\tilde{p}_i-p_i| \le \max\left(\frac{\gamma p_i}{1000},\frac{\gamma^2 n^{-\alpha}}{1000}\right) \text{ and } |\tilde{q}_i-q_i| \le \max\left(\frac{\gamma q_i}{1000},\frac{\gamma^2 n^{-\alpha}}{1000}\right)\ .$$

```
Algorithm Hellinger-Test$(p, q, \epsilon)$
1.  $m_i^p, m_i^q \leftarrow 0$ for all $i \in [n]$
2.  for $t = 1$ to $m$:
3.      do $i \leftarrow$ sample$(p)$ and $m_i^p \leftarrow m_i^p + 1$
4.         $i \leftarrow$ sample$(q)$ and $m_i^q \leftarrow m_i^q + 1$
5.  return FAIL if
        $$\sum_{i \in S} \left( \sqrt{m_i^p/m} - \sqrt{m_i^q/m} \right)^2 > \epsilon/10$$
    where $S = \{i : \max\{m_i^p, m_i^q\} \geq mn^{-\alpha}\}$
6.  return $\ell_2$-Tester$(p', q', \epsilon n^{-1/2})$ where $p'$ is the distribution formed by the
    following sampling procedure:
        $i \leftarrow$ sample$(p)$
        sample$(p') \leftarrow$ ($i$ if $i \notin S$ and $j \in_R [2n] \setminus [n]$ otherwise)
    and $q'$ is defined analogously.
```

Figure 8.1: Hellinger-Testing (Generative Oracle Model)

Therefore $i \notin S$ implies that $p_i, q_i \leq 2n^{-\alpha}$ as required. Also, if $i \in S$ and $p_i > q_i$ then $p_i \geq n^{-\alpha}/2$ and hence $|\tilde{p}_i - p_i| \leq (\gamma/1000)p_i$. Let $i \in S$. Without loss of generality assume that $p_i \geq q_i$. Therefore,

$$\begin{aligned} |(\sqrt{\tilde{p}_i} - \sqrt{\tilde{q}_i})^2 + (\sqrt{p_i} - \sqrt{q_i})^2| &\leq |\tilde{p}_i - p_i| + |\tilde{q}_i - q_i| + 2|\sqrt{\tilde{p}_i\tilde{q}_i} - \sqrt{p_i q_i}| \\ &\leq \gamma p_i/500 + 2|\sqrt{\tilde{p}_i\tilde{q}_i} - \sqrt{p_i q_i}| \ . \end{aligned}$$

First assume that $q_i \geq \gamma^2 n^{-\alpha}$. Therefore,

$$2|\sqrt{\tilde{p}_i\tilde{q}_i} - \sqrt{p_i q_i}| \leq 2\sqrt{p_i q_i}|\sqrt{\tilde{p}_i\tilde{q}_i/(p_i q_i)} - 1| \leq 2\gamma p_i/1000 \ .$$

Alternatively, if $q_i \leq \gamma^2 n^{-\alpha}$ then,

$$2|\sqrt{\tilde{p}_i\tilde{q}_i} - \sqrt{p_i q_i}| \leq 2(\gamma n^{-\alpha}/1000)\sqrt{\max\{\tilde{p}_i, p_i\}} \leq (\gamma/250)p_i\sqrt{(1+\gamma)} \leq 2\gamma p_i/250 \ .$$

In either case, $|(\sqrt{\tilde{p}_i} - \sqrt{\tilde{q}_i})^2 + (\sqrt{p_i} - \sqrt{q_i})^2| \leq \gamma p_i/100$ as required. □

Lemma 8.16. *Let p and q be two distributions on $[n]$ and let $S \subset [n]$. Define a distribution p',*

$$p'_i = \begin{cases} p_i & \text{if } i \in [n] \setminus S \\ 0 & \text{if } i \in S \\ (\sum_{j\in S} p_j)/n & \text{if } i \in [2n] \setminus [n] \end{cases} .$$

Let q' be defined analogously. Then,

$$\sum_{i \notin S} (\sqrt{p_i} - \sqrt{q_i})^2 \leq \text{Hellinger}(p', q') \leq \text{Hellinger}(p, q) \ .$$

Proof. The first inequality is immediate because all terms are positive. To bound the second term we need to show that,

$$\sum_{i\in S} (\sqrt{p_i} - \sqrt{q_i})^2 \geq n \left(\sqrt{\frac{\sum_{i\in S} p_i}{n}} - \sqrt{\frac{\sum_{i\in S} p_i}{n}} \right)^2 = \left(\sqrt{\sum_{i\in S} p_i} - \sqrt{\sum_{i\in S} q_i} \right)^2 \ .$$

We will first show that $(\sqrt{p_i} - \sqrt{q_i})^2 + (\sqrt{p_j} - \sqrt{q_j})^2 \geq (\sqrt{p_i + p_j} - \sqrt{q_i + q_j})^2$. This is because,

$$\begin{array}{rrcl} & (\sqrt{p_j q_i} - \sqrt{p_i q_j})^2 & \geq & 0 \\ \Rightarrow & (p_i + p_j)(q_i + q_j) & \geq & p_i q_i + p_j q_j + 2\sqrt{p_i p_j q_i q_j} \\ \Rightarrow & 2\sqrt{(p_i + p_j)(q_i + q_j)} & \geq & 2\sqrt{p_i q_i} + 2\sqrt{p_j q_j} \\ \Rightarrow & (\sqrt{p_i} - \sqrt{q_i})^2 + (\sqrt{p_j} - \sqrt{q_j})^2 & \geq & (\sqrt{p_i + p_j} - \sqrt{q_i + q_j})^2 \ . \end{array}$$

Therefore, by "merging" the probability mass on all indices in S we decrease the Hellinger distance as required. □

Theorem 8.17 (Hellinger Testing)**.** *There exists an $(\epsilon, \epsilon^2/(32n^{1-\alpha}), \delta)$-tester for Hellinger$(p, q)$ using $O(\max\{\epsilon^{-2} n^\alpha \log n, \epsilon^{-4}(n^{-2\alpha+2} + \epsilon^2 n^{1-\alpha/2})\} \log \delta^{-1})$ samples.*

Proof. Let $A = \sum_{i\in S} (\sqrt{p_i} - \sqrt{q_i})^2$ and $B = \sum_{i\notin S} (\sqrt{p_i} - \sqrt{q_i})^2$. By Lemma 8.15, we estimate A with an additive error of $\epsilon/10$.

1. If $\text{Hellinger}(p, q) > \epsilon$ then either A is bigger than $\epsilon/2$ or B is bigger than $\epsilon/2$. If A is bigger than $\epsilon/2$ then our estimate of A is bigger than $\epsilon(1/2 - 2/10)$ and in which case $\sum_{i \in S}(\sqrt{\tilde{p}_i} - \sqrt{\tilde{q}_i})^2 > \epsilon/10$ and we fail. Otherwise if B is bigger than $\epsilon/2$. Therefore, appealing to Eq. 8.2 and Lemma 8.16 (not that the p' and q' are on base $[2n]$) we deduce that,
$$\epsilon/2 \leq \text{Hellinger}(p', q') \leq \sqrt{2n}\ell_2(p', q') \ .$$
Consequently the ℓ_2 test fails.

2. If $\text{Hellinger}(p, q) < \epsilon^2/(32n^{1-\alpha})$ then $A < \epsilon^2/n^{1-\alpha}$ and we pass the first test because our estimate of A is at most $\epsilon^2/n^{1-\alpha} + \epsilon/100 < \epsilon/10$ (for sufficiently large n.) By Lemma 8.15, $\max(\ell_\infty(p'), \ell_\infty(q')) \leq 2n^{-\alpha}$. Therefore, appealing to Eq. 8.2 and Lemma 8.16,
$$n^\alpha \ell_2^2(p', q')/8 \leq \text{Hellinger}(p', q') \leq A + B < \epsilon^2/(32n^{1-\alpha})$$
implies that the second test passes since $n^\alpha \ell_2^2(p', q') \leq \epsilon^2/(4n^{1-\alpha})$ and thus $\ell_2(p', q') \leq \epsilon/(2\sqrt{n})$.

□

Observe that setting $\alpha = 2/3$ yields an algorithm with sample complexity $\tilde{O}(n^{2/3}/\epsilon^4)$. For distributions such that either $p_i = q_i$ or one of p_i, q_i is 0, $\Delta(p, q) = \ell_1(p, q) = \text{Hellinger}(p, q) = \text{JS}(p, q)$. Batu et al. [BFR+00] discuss lower bounds for ℓ_1 distance property testing.

8.4.2 f-Divergences Testing (Combined Oracle)

In this section we consider property testing in the combined oracle model for all bounded f-divergences.

Theorem 8.18. *There exists an (ϵ, δ)-approximation algorithm for any τ-bounded D_f in the combined oracle model making $O(\tau\epsilon^{-2}\log(\delta^{-1})/D_f(p, q)))$ queries.*

```
Algorithm Combined Oracle Distance Testing
1.  $E \leftarrow 0$
2.  for $t = 1$ to $m$:
3.      do $i \leftarrow \texttt{sample}(p)$ and $x = \frac{\texttt{probe}(q,i)}{\texttt{probe}(p,i)}$
4.          If $x > 1$ then $a \leftarrow f(x)$ else $a \leftarrow 0$
5.          $j \leftarrow \texttt{sample}(q)$ and $x = \frac{\texttt{probe}(q,j)}{\texttt{probe}(p,j)}$
6.          If $x < 1$ then $b \leftarrow f^*(1/x)$ else $b \leftarrow 0$
7.          $E \leftarrow (a+b)/2\tau + E$
8.  return $2\tau E/m$
```

Figure 8.2: Distance-Testing (Combined Oracle Model)

Proof. Consider the value $(a+b)/(2\tau)$ added to E in each iteration. This is a random variable with range $[0,1]$ and mean $(D_f(p,q))/(2\tau)$. By applying the Chernoff-Hoeffding bounds,

$$\Pr\left[\left|E - m\frac{D_f(p,q)}{2\tau}\right| < \epsilon m\frac{D_f(p,q)}{2\tau}\right] \leq 2e^{-\epsilon^2 D_f(p,q)m/6\tau} \leq 1-\delta \ .$$

Therefore, E is an (ϵ,δ)-approximation for $mD_f(p,q)/2\tau$. Hence, $2\tau E/m$ is an (ϵ,δ)-approximation for $D_f(p,q)$ as required. □

Using a slightly different analysis, the algorithm also yields an $(\epsilon,\epsilon/2,\delta)$-tester.

Corollary 8.19. *There exists an $(\epsilon,\epsilon/2,\delta)$-tester algorithm for any bounded D_f in the combined oracle model making $O(\epsilon^{-1}\log\delta^{-1})$ queries.*

We now prove a corresponding lower-bound that shows that our algorithm is tight. Note that while it is relatively simple to see that there exists two distributions that are indistinguishable with less than $o(1/\ell_1)$ oracle calls, it requires some work to also show a lower bound with a dependence on ϵ. Further note that the proof below also gives analogous results for JS, Hellinger and Δ. (This follows from the remarks made at the end of the previous section.)

Theorem 8.20. *Any $(\epsilon,1/4)$-approximation algorithm of ℓ_1 in the combined oracle model requires $\Omega(\epsilon^{-2}/\ell_1)$ queries.*

Proof. Let p and q^r be the distributions on $[n]$ described by the following two probability vectors:

$$\begin{aligned} p &= (1 - 3a/2, \overbrace{3a\epsilon/2k, \dots, 3a\epsilon/2k}^{k/\epsilon}, 0, \dots, 0) \\ q^r &= (1 - 3a/2, \overbrace{0, \dots 0}^{r}, \overbrace{3a\epsilon/2k, \dots, 3a\epsilon/2k}^{k/\epsilon}, 0, \dots, 0) \end{aligned}$$

Then $\ell_1(p, q^{k/3\epsilon}) = a$ and $\ell_1(p, q^{k/3\epsilon+k}) = a(1 + 3\epsilon)$. Hence to $1 + \epsilon$ approximate the distance between p and q^r we need to distinguish between the cases when $r = k/3\epsilon (=: r_1)$ and $r = k/3\epsilon + k (=: r_2)$. Consider the distributions p' and $q^{r\prime}$ formed by arbitrarily permuting the base sets of the p and q^r. Note that the ℓ_1 distance remains the same. We will show that, without knowledge of the permutation, it is impossible to estimate this distance with $o(1/(\epsilon^2 a))$ oracle calls. We reason this by first disregarding the value of any "blind probes", i.e., a probe $\texttt{probe}(p', i)$ or $\texttt{probe}(q', i)$ for any i that has not been returned as a sample. This is the case because, by choosing $n \gg k/(a\epsilon^2)$ we ensure that, with arbitrarily high probability, for any $o(1/(\epsilon^2 a))$ set of i's chosen from any $n - o(1/(a\epsilon^2))$ sized subset of $[n]$, $p_i{}' = q_i^{r\prime} = 0$. This is the case for both r_1 and r_2. Let $I = \{i : p_i \text{ or } q_i^r = 3a\epsilon/2k\}$ and $I_1 = \{i \in I : p_i \neq q_i^r\}$. Therefore determining whether $r = r_1$ or r_2 is equivalent to determining whether $|I_1|/|I| = 1/2$ or $1/2 + \frac{9\epsilon}{8+6\epsilon}$. We may assume that every time an algorithm sees i returned by $\texttt{sample}(p)$ or $\texttt{sample}(q)$, it learns the exact values of p_i and q_i for free. Furthermore, by making k large ($k = \omega(1/\epsilon^3)$ suffices) we can ensure that no two $\texttt{sample}$ oracle calls will ever return the same $i \in I$ (with high probability.) Hence distinguishing between $|I_1|/|I| = 1/2$ and $1/2 + \frac{9\epsilon}{8+6\epsilon}$ is analogous to distinguishing between a fair coin and a $\frac{9\epsilon}{8+6\epsilon} = \Theta(\epsilon)$ biased coin. It is well known that the latter requires $\Omega(1/\epsilon^2)$ samples. Unfortunately only $O(1/a)$ samples return an $i \in I$ since with probability $1 - 3a/2$ we output an $i \notin I$ when sampling either p or q. The bound follows. $\square$

Chapter 9

Entropy

Chapter Outline: In this chapter we present a single-pass algorithm for estimating the entropy, $-\sum_{i\in[n]} p_i \lg p_i$, of the empirical distribution defined by a data stream. We demonstrate that the algorithm is near-optimal by showing an almost matching lower bound. In the next two sections, we present algorithms and lower-bounds for estimating related quantities, the k-order entropy and the entropy of a random-walk on an undirected graph. Lastly, we present an algorithm for estimating entropy in the combined oracle model. For background see Chapter 7.

9.1 Computing the Entropy of a Stream

For a real-valued function f such that $f(0) = 0$, let us define $\overline{f}_m(A) := \frac{1}{m}\sum_{i=1}^{n} f(m_i)$. We base our approach on the method of Alon, Matias and Szegedy [AMS99] to estimate quantities of the form $\overline{f}_m(A)$: note that the empirical entropy of A is one such quantity with $f(m_i) = m_i \log(m/m_i)$.

Definition 9.1. *Let $\mathcal{D}(A)$ be the distribution of the random variable R defined thus: Pick $J \in [m]$ uniformly at random and let $R = |\{j : a_j = a_J,\ J \le j \le m\}|$.*

The core idea is to space-efficiently generate a random variable $R \sim \mathcal{D}(A)$. For an integer c, define the random variable

$$\mathrm{Est}_f(R,c) \ := \ \frac{1}{c}\sum_{i=1}^{c} X_i \,, \tag{9.1}$$

where the random variables $\{X_i\}$ are independent and each distributed identically to $(f(R)-f(R-1))$. Appealing to Chernoff-Hoeffding bounds one can show that by increasing c, $\mathrm{Est}_f(R,c)$ can be made arbitrarily close to $\overline{f}_m(A)$. This is formalized in the lemma below.

Lemma 9.2. *Let $X := f(R) - f(R-1)$, $a, b \geq 0$ such that $-a \leq X \leq b$, and*

$$c \geq 3(1 + a/\mathrm{E}[X])^2\epsilon^{-2}\ln(2\delta^{-1})(a+b)/(a+\mathrm{E}[X]) \ .$$

Then $\mathrm{E}[X] = \overline{f}_m(A)$ and, if $\mathrm{E}[X] \geq 0$, the estimator $\mathrm{Est}_f(R,c)$ gives an (ϵ,δ)-approximation to $\overline{f}_m(A)$ using space c times the space required to maintain R.

Proof. $\mathrm{E}[X] = \overline{f}_m(A)$ follows by straightforward calculation of the expectation. Consider the random variable $Y := (X+a)/(a+b)$. First note that $Y \in [0,1]$ and that $\mathrm{E}[Y] = (\overline{f}_m(A)+a)/(a+b)$. Therefore, Chernoff-Hoeffding bounds imply that, if $\{Y_i\}$ are independent and each distributed identically to Y, then

$$\begin{aligned}
&\Pr\left[\left|\frac{1}{c}\sum_{i\in[c]} Y_i - \frac{\overline{f}_m(A)+a}{a+b}\right| > \frac{\epsilon}{1+a/\mathrm{E}[X]}\frac{\overline{f}_m(A)+a}{a+b}\right]\\
&= \ \Pr\left[\left|\frac{1}{c}\sum_{i\in[c]} Y_i - \mathrm{E}[Y]\right| > \frac{\epsilon}{1+a/\mathrm{E}[X]}\,\mathrm{E}[Y]\right]\\
&\leq \ \exp\left(-c\left(\frac{\epsilon}{1+a/\mathrm{E}[X]}\right)^2\frac{\overline{f}_m(A)+a}{2(a+b)}\right) + \exp\left(-c\left(\frac{\epsilon}{1+a/\mathrm{E}[X]}\right)^2\frac{\overline{f}_m(A)+a}{3(a+b)}\right)\\
&\leq \delta \ .
\end{aligned}$$

Consequently $\mathrm{Est}'_f(R,c) = c^{-1}\sum_{i\in[c]} Y_i$ is an $(\epsilon/(1+a/\mathrm{E}[X]),\delta)$-approximation to $(\overline{f}_m(A)+a)/(a+b)$. Note that, $\mathrm{Est}'_f(R,c) = \left(\mathrm{Est}_f(R,c)+a\right)/(a+b)$. This implies that,

$$\begin{aligned} &\Pr\left[|\operatorname{Est}_f(R,c) - \overline{f}_m(A)| > \epsilon \overline{f}_m(A)\right] \\ = \;& \Pr\left[\left|\operatorname{Est}'_f(R,c) - \frac{\overline{f}_m(A)+a}{a+b}\right| > \frac{\epsilon}{1+a/\operatorname{E}[X]}\frac{\overline{f}_m(A)+a}{a+b}\right] \le \delta \ . \end{aligned}$$

Therefore, $\operatorname{Est}_f(R,c)$ gives an (ϵ,δ)-approximation to $\overline{f}_m(A)$ as claimed. □

Overview of the technique: We now give some of the intuition behind our algorithm for estimating $H(p)$. Let A' denote the substream of A obtained by removing from A all occurrences of the most frequent token (with ties broken arbitrarily) and let $R' \sim \mathcal{D}(A')$. A key component of our algorithm (see Algorithm *Maintain-Samples* below) is a technique to simultaneously maintain R and enough extra information that lets us recover R' when we need it. Let $p_{\max} := \max_i p_i$. Let λ be given by

$$\lambda(x,m) \ := \ x\lg(m/x)\,, \text{ where } \lambda(0,m) \ := \ 0\,, \tag{9.2}$$

so that $\overline{\lambda}_m(A,m) = H(p)$. Define $X = \lambda(R,m) - \lambda(R-1,m)$ and $X' = \lambda(R',m) - \lambda(R'-1,m)$. If $p_{\max}$ is bounded away from 1 then we can show that $1/\operatorname{E}[X]$ is "small," so $\operatorname{Est}_\lambda(R,c)$ gives us our desired estimator for a "small" value of c, by Lemma 9.2. If, on the other hand, $p_{\max} > \frac{1}{2}$ then we can recover R' and can show that $1/\operatorname{E}[X']$ is "small." Finally, by our analysis we can show that $\operatorname{Est}_\lambda(R',c)$ and an estimate of $p_{\max}$ can be combined to give an (ϵ,δ)-approximation to $H(p)$. This logic is given in Algorithm *Entropy-Estimator* below.

Thus, our algorithm must also maintain an estimate of $p_{\max}$ in parallel to Algorithm *Maintain-Samples*. There are a number of ways of doing this and here we choose to use the Misra-Gries algorithm [MG82] with a sufficiently large number of counters. This (deterministic) algorithm takes a parameter k — the number of counters — and processes the stream, retaining up to k pairs $(i,\hat{m}_i)$, where i is a token and the counter $\hat{m}_i$ is an estimate of its frequency m_i. The algorithm starts out holding no pairs and implicitly setting each $\hat{m}_i = 0$. Upon reading a token, i, if a pair $(i, \hat{m}_i)$ has already been retained, then $\hat{m}_i$ is incremented; else, if fewer than k pairs have been retained, then a new pair $(i, 1)$ is created and retained; else, $\hat{m}_j$ is decremented for each retained pair $(j, \hat{m}_j)$ and then all pairs of the form $(j, 0)$ are discarded. The following lemma summarizes the key properties of this algorithm; the proof is simple (see, e.g., [BKMT03]) and we omit it.

Lemma 9.3. *The estimates $\hat{m}_i$ computed by the Misra-Gries algorithm using k counters satisfy $\hat{m}_i \leq m_i$ and $m_i - \hat{m}_i \leq (m - m_i)/k$.* □

We now describe our algorithm more precisely with some pseudocode. By abuse of notation we use $\mathrm{Est}_\lambda(r, c)$ to also denote the algorithmic procedure of running in parallel c copies of an algorithm that produces r and combining these results as in Eq. 9.1.

```
Algorithm Maintain-Samples
1.  for $a \in A$
2.      do Let $t$ be a random number in the range $[m^3]$
3.         if $a = s_0$
4.            then if $t < t_0$ then $(s_0, t_0, r_0) \leftarrow (a, t, 1)$ else $r_0 \leftarrow r_0 + 1$
5.            else if $a = s_1$ then $r_1 \leftarrow r_1 + 1$
6.                 if $t < t_0$
7.                    then $(s_1, t_1, r_1) \leftarrow (s_0, t_0, r_0); (s_0, t_0, r_0) \leftarrow (a, t, 1)$
8.                    else if $t < t_1$ then $(s_1, t_1, r_1) \leftarrow (a, t, 1)$

Algorithm Entropy-Estimator
1.  $c \leftarrow 16\epsilon^{-2} \ln(2\delta^{-1}) \lg(me)$
2.  Run the Misra-Gries algorithm on $A$ with $k = \lceil 7\epsilon^{-1} \rceil$ counters, in parallel
    with Maintain-Samples
3.  if Misra-Gries retains a token $i$ with counter $\hat{m}_i > m/2$
4.     then $(i_{\max}, \hat{p}_{\max}) \leftarrow (i, \hat{m}_i/m)$
5.          if $a_0 = i_{\max}$ then $r \leftarrow r_1$ else $r \leftarrow r_0$
6.          return $(1 - \hat{p}_{\max}) \cdot \mathrm{Est}_\lambda(r, c) + \hat{p}_{\max} \lg(1/\hat{p}_{\max})$
7.     else return $\mathrm{Est}_\lambda(r_0, c)$
```

Figure 9.1: An Algorithm for Approximating Entropy.

Maintaining Samples from the Stream: We show a procedure that allows us to generate R and R' with the appropriate distributions. For each token a in the stream, we draw t, a random number in the range $[m^3]$, as its label. We choose to store certain tokens from the stream, along with their label and the count of the number of times the same token has been observed in the stream since it was last picked. We store *two* such tokens: the token s_0 that has achieved the least t value seen so far, and the token s_1 such that it has the least t value of all tokens not equal to s_0 seen so far. Let t_0 and t_1 denote their corresponding labels, and let r_0 and r_1 denote their counts in the above sense. Note that it is easy to maintain these properties as new items arrive in the stream, as Algorithm *Maintain-Samples* illustrates.

Lemma 9.4. *Algorithm Maintain-Samples satisfies the following properties. (i) After processing the whole stream A, s_0 is picked uniformly at random from A and $r_0 \sim \mathcal{D}(A)$. (ii) For $a \in [n]$, let $A \setminus a$ denote the stream A with all occurrences of a removed. Suppose we set s and r thus: if $s_0 \neq a$ then $s = s_0$ and $r = r_0$, else $s = s_1$ and $r = r_1$. Then s is picked uniformly from $A \setminus a$ and $r \sim \mathcal{D}(A \setminus a)$.*

Proof. To prove (i), note that the way we pick each label t ensures that (w.h.p.) there are no collisions amongst labels and, conditioned on this, the probability that any particular token gets the lowest label value is $1/m$.

We show (ii) by reducing to the previous case. Imagine generating the stream $A \setminus a$ and running the algorithm on it. Clearly, picking the item with the smallest t value samples uniformly from $A \setminus a$. Now let us add back in all the occurrences of a from A. One of these may achieve a lower t value than any item in $A \setminus a$, in which case it will be picked as s_0, but then s_1 will correspond to the sample we wanted from $A \setminus a$, so we can return that. Else, $s_0 \neq a$, and is a uniform sample from $A \setminus a$. Hence, by checking whether $s_0 = a$ or not, we can choose a uniform sample from $A \setminus a$. The claim about the distribution of r is now straightforward: we only need to observe from the pseudocode that, for $j \in \{0, 1\}$, r_j correctly counts the number of occurrences of s_j in A from the time s_j was last picked. $\square$

Analysis of the Algorithm: We now analyse our main algorithm, given in full in Algorithm *Entropy-Estimator*.

Theorem 9.5. *Algorithm Entropy-Estimator uses*

$$O(\epsilon^{-2} \log(\delta^{-1}) \log m(\log m + \log n)) \text{ bits}$$

and gives an (ϵ, δ)*-approximation to* $H(p)$*.*

Proof. To argue about the correctness of Algorithm *Entropy-Estimator*, we first look closely at the Misra-Gries algorithm used within it. By Lemma 9.3, $\hat{p}_i := \hat{m}_i/m$ is a good estimate of p_i. To be precise, $|\hat{p}_i - p_i| \leq (1-p_i)/k$. Hence, by virtue of the estimation method, if $p_i > \frac{2}{3}$ and $k \geq 2$, then i must be among the tokens retained and must satisfy $\hat{p}_i > \frac{1}{2}$. Therefore, in this case we will pick $i_{\max}$ — the item with maximum frequency — correctly, and $p_{\max}$ will satisfy

$$\hat{p}_{\max} \;\leq\; p_{\max} \quad \text{and} \quad |\hat{p}_{\max} - p_{\max}| \;\leq\; \frac{1-p_{\max}}{k}\,. \tag{9.3}$$

Let A, A', R, R', X, X' be as before. Suppose $\hat{p}_{\max} \leq \frac{1}{2}$. The algorithm then reaches Line 7. By Part (i) of Lemma 9.4, the returned value is $\mathrm{Est}_\lambda(R, c)$. Now (9.3), together with $k \geq 2$, implies $p_{\max} \leq \frac{2}{3}$. Lemma 3.3 from [CDM06] states that the minimum entropy is obtained when all other tokens are identical, giving us $H(p) \geq \frac{2}{3}\lg\frac{3}{2} + \frac{1}{3}\lg\frac{3}{1} > 0.9$. Note that $-\lg e \leq X \leq \lg m$. This follows because,

$$\frac{d}{dx} x \lg\left(\frac{m}{x}\right) = \lg\left(\frac{m}{x}\right) - \lg e$$

and so $\frac{d}{dx}\lambda(x) - \lambda(x-1) = \lg(1-1/x)$ shows $\lambda(x) - \lambda(x-1)$ is decreasing in the range 1 to m. The maximum value is $\lambda(1) - \lambda(0) = \lg m$ and the minimum is $\lambda(m) - \lambda(m-1) = -\lg e$. Hence Lemma 9.2 implies that c is large enough to ensure that the return value is a $(\frac{3}{4}\epsilon, \delta)$-approximation to $H(p)$.

Now suppose $\hat{p}_{\max} > 1/2$. The algorithm then reaches Line 6. By Part (ii) of Lemma 9.4, the return value is $(1-\hat{p}_{\max}) \cdot \mathrm{Est}_\lambda(R', c) + \hat{p}_{\max} \lg(1/\hat{p}_{\max})$, and (9.3) implies that $p_{\max} > 1/2$. Assume, w.l.o.g., that $i_{\max} = 1$. Then

$$\mathrm{E}[X'] \;=\; \overline{\lambda}(A') \;=\; \frac{1}{m-m_1} \sum_{i=2}^{n} \lambda(m_i) \;\geq\; \lg \frac{m}{m-m_1} \;\geq\; 1\,,$$

where the penultimate inequality follows by convexity arguments. As before, $-\lg e \leq X \leq \lg m$, and hence Lemma 9.2 implies that c is large enough to ensure that $\mathrm{Est}_\lambda(R', c)$ is a $(\frac{3}{4}\epsilon, \delta)$-approximation to $\overline{\lambda}(A')$.

Next, we show that $\hat{p}_1 \lg(1/\hat{p}_1)$ is a $(\frac{2}{k}, 0)$-approximation to $p_1 \lg(1/p_1)$, as follows:

$$\frac{|p_1 \lg(1/p_1) - \hat{p}_1 \lg(1/\hat{p}_1)|}{p_1 \lg(1/p_1)} \;\leq\; \frac{|\hat{p}_1 - p_1|}{p_1 \lg(1/p_1)} \max_{p \in [\frac{1}{2}, 1]} \left| \frac{d}{dp}(p \lg(1/p)) \right| \;\leq\; \frac{(1-p_1)}{k\, p_1 \lg(1/p_1)} \cdot \lg e,$$

and this bounded by $2/k$ because $g(p) := (1-p)/(p \ln(1/p))$ is non-increasing in the interval $[\frac{1}{2}, 1]$, so $g(p) \leq g(\frac{1}{2}) < 2$. To see this, note that $1 - p + \ln p \leq 0$ for all positive p and that $g'(p) = (1 - p + \ln p)/(p \ln p)^2$. Now observe that

$$H(p) \;=\; (1-p_1)\overline{\lambda}_{m-m_1}(A', m) + p_1 \lg(1/p_1)\,. \tag{9.4}$$

From (9.3) it follows that $(1-\hat{p}_1)$ is an $(\frac{1}{k}, 0)$-approximation to $(1-p_1)$. Note that $\frac{1}{7}\epsilon + \frac{3}{4}\epsilon + \frac{3}{28}\epsilon^2 \leq \epsilon$ for $\epsilon \leq 1$. Thus, setting $k \geq \lceil 7\epsilon^{-1} \rceil$ ensures that that $(1-\hat{p}_1) \cdot \mathrm{Est}_\lambda(R', c)$ is a (ϵ, δ)-approximation to $(1-p_1)\overline{\lambda}(A')$, and $\hat{p}_1 \lg(1/\hat{p}_1)$ is a (better than) $(\epsilon, 0)$-approximation to $p_1 \lg(1/p_1)$. Thus, we have shown that in this case the algorithm returns a (ϵ, δ)-approximation to $H(p)$, since both terms in (9.4) are approximated with relative error.

The claim about the space usage is straightforward. The Misra-Gries algorithm requires $O(k) = O(\epsilon^{-1})$ counters and item identifiers. Each run of Algorithm *Maintain-Samples* requires $O(1)$ counters, labels, and item identifiers, and there are $c = O(\epsilon^{-2} \log(\delta^{-1}) \log m)$ such runs. Everything stored is either an item from the stream, a counter that is bounded by m, or a label that is bounded by m^3, so the space for each of these is $O(\log m + \log n)$ bits. □

9.1.1 Variations on the Algorithm

Randomness and Stream Length: As described, our algorithm seems to require $O(m \log m)$ bits of randomness, since we require a random number in the range $[m^3]$ for each item in the stream. This randomness requirement can be reduced to $O(\log^{O(1)} m)$ bits by standard arguments invoking Nisan's pseudorandom generator [Nis92]. An alternate approach is to use a hash function from a min-wise independent family on the stream index to generate t [Ind01]. This requires a modification to the analysis: the probability of picking any fixed item changes from $1/m$ to a value in the interval $[(1-\epsilon)/m, (1+\epsilon)/m]$. One can show that this introduces a $1+O(\epsilon)$ factor in the expressions for expectation, and does not significantly affect the range of the estimators, and so does not affect the overall correctness; an additional $O(\log n \log \epsilon^{-1})$ factor in space would also be incurred to store the descriptions of the hash functions.

The algorithm above also seems to require prior knowledge of m, although an upper bound clearly suffices (we can compute the true m as the stream arrives). But we only need to know m in order to choose the size of the random labels large enough to avoid collisions. Should the assumed bound be proven too low, it suffices to extend the length of labels t_0 and t_1 by drawing further random bits in the event of collisions to break ties. Invoking the principle of deferred decisions, it is clear that the correctness of the algorithm is unaffected.

Sliding Window Computations: In many cases it is desirable to compute functions not over the whole semi-infinite stream, but rather over a sliding window of the last W updates. Our method accommodates such an extension with an $O(\log^2 W)$ expansion of space (with high probability). Formally, define the sliding window count of i as $m_i^w = |\{j | a_j = i, i > m - w\}|$. The empirical probability is $p_i^w = m_i^w / w$, and the empirical entropy is $H(p^w) = \sum_{i=1}^{n} -p_i^w \lg p_i^w$.

Lemma 9.6. *We can approximate $H(p^w)$ for any $w < W$ in space bounded by*

$O(\epsilon^{-2}\log(\delta^{-1})\log^3 W)$ machine words with high probability.

Proof. We present an algorithm that retains sufficient information so that, after observing the stream of values, given $w < W$ we can recover the information that Algorithm *Entropy-Estimator* would have stored had only the most recent w values been presented to it. From this, the correctness follows immediately. Thus, we must be able to compute $s_0^w, r_0^w, s_1^w, r_1^w, i_{\max}^w$ and $p_{\max}^w$, the values of $s_0, r_0, s_1, r_1, i_{\max}$ and $p_{\max}$ on the substreams defined by the sliding window specified by w.

For $i_{\max}^w$ and $p_{\max}^w$, it is not sufficient to apply standard sliding window frequent items queries [AM04]. To provide the overall accuracy guarantee, we needed to approximate $p_{\max}$ with error proportion to $\epsilon'(1-p_{\max}^w)$ for a parameter ϵ'. Prior work gives guarantees only in terms of $\epsilon' p_{\max}^w$, so we need to adopt a new approach. We replace our use of the Misra-Gries algorithm with the Count-Min sketch [CM05a]. This is a randomized algorithm that hashes each input item to $O(\log\delta^{-1})$ buckets, and maintains a sum of counts within each of a total of $O(\epsilon^{-1}\log\delta^{-1})$ buckets. If we were able to create a CM-sketch summarizing just the most recent w updates, then we would be able to find an (ϵ,δ) approximation to $(1-p_{\max}^w)$, and hence also find $p_{\max}^w$ with error $\epsilon(1-p_{\max}^w)$. This follow immediately from the properties of the sketch proved in [CM05a]. In order to make this valid for arbitrary sliding windows, we replace each counter within the sketch with an Exponential Histogram or Deterministic Wave data structure [DGIM02, GT02]. This allows us to $(\epsilon, 0)$ approximate the count of each bucket within the most recent $w < W$ timesteps in space $O(\epsilon^{-1}\log^2 W)$. Combining these and rescaling ϵ, one can build an (ϵ,δ) approximation of $(1-p_{\max}^w)$ for any $w < W$. The space required for this estimation is $O(\epsilon^{-2}\log^2 W\log\delta^{-1}(\log m+\log n))$ bits.

For s_0^w, r_0^w, s_1^w and r_1^w, we can take advantage of the fact that these are defined by randomly chosen tags t_0^w and t_1^w, and for any W there are relatively few possible candidates for all the $w < W$. Let t_j be the random tag for the jth item in the stream. We maintain the following set of tuples,

$$S_0 = \{(j, a_j, t_j, r_j) : j = \underset{m-w<i\leq m}{\operatorname{argmin}}\ t_j, r_j = |\{k|a_k = a_j, k \geq j\}|, w < W\}$$

This set defines $j_0^w = \operatorname{argmin}_{m-w<i\leq m} t_j$ for $w < W$. We maintain a second set of tuples,

$$S_1 = \{(j, a_j, t_j, r_j) : j = \underset{\substack{m-w<i\leq m \\ i\neq j_0^w}}{\operatorname{argmin}}\ t_j, r_j = |\{k|a_k = a_j, k \geq j\}|, w < W\}$$

and this set defines $j_1^w = \operatorname{argmin}_{m-w<i\leq m} t_j$ for $w < W$. Note that it is straightforward to maintain S_0 and S_1. Then, for any $w < W$, we set,

$$(s_0^w, r_0^w) \leftarrow (a_{j_0^w}, r_{j_0^w}) \quad \text{and} \quad (s_1^w, r_1^w) \leftarrow (a_{j_1^w}, r_{j_1^w}) \ .$$

We now bound the sizes of S_0 and S_1. The size of S_0 can be bounded by observing that if we build a treap over the sequence of timestamp, label pairs where we order by timestamp and heapify by label, the members of S_0 correspond to precisely the right spine of the treap. As argued in [BDM02], this approach yields a strong bound on $|S_0|$, since with high probability the height of a treap with randomly chosen priorities such as these (i.e., a random binary search tree) is logarithmic. Further, we can observe that members of S_1 correspond to nodes in the treap that are also on the right spine, are left children of members of S_0, or the right descendants of left children. Thus, if the treap has height h, the size of S_1 is $O(h^2)$. For windows of size at most W, the implicit treap has height $O(\log W)$ with high probability.

Thus, we need to store a factor of $O(\log^2 W)$ more information for each instance of the basic estimator. The total space bound is therefore $O(\epsilon^{-2} \log(\delta^{-1}) \log^3 W(\log m + \log n))$ bits, since now the estimator is bounded by $\log W$, not $\log m$. $\square$

9.1.2 Extensions to the Technique

We observe that the method we have introduced here, of allowing a sample to be drawn from a modified stream with an item removed may have other applications.

The method naturally extends to allowing us to specify a set of k items to remove from the stream after the fact, by keeping the $k+1$ distinct items achieving the smallest label values. In particular, Lemma 9.4 can be extended to give the following.

Lemma 9.7. *There exists an algorithm $\mathcal{A}$, using $O(k)$ space, that returns k pairs $(s_i, r_i)_{i \in [k+1]}$ such that s_i is picked uniformly at random from $A \setminus \{s_1, \ldots, s_{i-1}\}$ and $r \sim \mathcal{D}(A \setminus \{s_1, \ldots, s_{i-1}\})$. Consequently, given a set S of size at most k and the output of $\mathcal{A}$ it is possible to sample (s, r) such that s is picked uniformly at random from $A \setminus S$ and $r \sim \mathcal{D}(A \setminus S)$.*

This may be of use in applications where we can independently identify "junk" items or other undesirable values which would dominate the stream if not removed. For example, in the case in which we wish to compute the quantiles of a distribution after removing the k most frequent items from the distribution. Additionally, the procedure may have utility in situations where a small fraction of values in the stream can significantly contribute to the variance of other estimators.

9.1.3 Lower Bound

We now show that the dependence of the above space bound on ϵ is nearly tight. To be precise, we prove the following theorem.

Theorem 9.8. *Any one-pass randomized $(\epsilon, \frac{1}{4})$-approximation for $H(p)$ requires $\Omega(\epsilon^{-2}/\log^2(\epsilon^{-1}))$ space.*

Proof. Let GAP-HAMDIST denote the following (one-way) communication problem. Alice receives $x \in \{0,1\}^N$ and Bob receives $y \in \{0,1\}^N$. Alice must send a message to Bob after which Bob must answer "near" if the Hamming distance $\|x-y\|_1 \leq N/2$ and "far" if $\|x-y\|_1 \geq N/2 + \sqrt{N}$. They may answer arbitrarily if neither of these two cases hold. The two players may follow a randomized protocol that must work correctly with probability at least $\frac{3}{4}$. It is known [IW03, Woo04] that GAP-HAMDIST has one-way communication complexity $\Omega(N)$.

We now reduce GAP-HAMDIST to the problem of approximating $H(p)$. Suppose $\mathcal{A}$ is a one-pass algorithm that (ϵ, δ)-approximates $H(p)$. Let N be chosen such that $\epsilon^{-1} = 3\sqrt{N}(\lg N + 1/2)$ and assume, w.l.o.g., that N is an integer. Alice and Bob will run $\mathcal{A}$ on a stream of tokens from $[N] \times \{0, 1\}$ as follows. Alice feeds the stream $\langle(i, x_i)\rangle_{i=1}^{N}$ into $\mathcal{A}$ and then sends over the memory contents of $\mathcal{A}$ to Bob who then continues the run by feeding in the stream $\langle(i, y_i)\rangle_{i=1}^{N}$. Bob then looks at the output $\mathrm{out}(\mathcal{A})$ and answers "near" if

$$\mathrm{out}(\mathcal{A}) \;<\; \lg N + \frac{1}{2} + \frac{1}{2\sqrt{N}}$$

and answers "far" otherwise. We now prove the correctness of this protocol.

Let $d := \|x - y\|_1$. Note that the stream constructed by Alice and Bob in the protocol will have $N - d$ tokens with frequency 2 each and $2d$ tokens with frequency 1 each. Therefore,

$$H(p) \;=\; (N-d) \cdot \frac{2}{2N} \lg \frac{2N}{2} + 2d \cdot \frac{1}{2N} \lg \frac{2N}{1} \;=\; \lg N + \frac{d}{N}\,.$$

Therefore, if $d \leq N/2$, then $H(p) \leq \lg N + \frac{1}{2}$ whence, with probability at least $\frac{3}{4}$, we will have

$$\mathrm{out}(\mathcal{A}) \leq (1+\epsilon) H(p) \leq \left(1 + \frac{1}{3\sqrt{N}(\lg N + 1/2)}\right)\left(\lg N + \frac{1}{2}\right) < \lg N + \frac{1}{2} + \frac{1}{2\sqrt{N}}$$

and Bob will correctly answer "near." A similar calculation shows that if $d \geq N/2 + \sqrt{N}$ then, with probability at least $\frac{3}{4}$, Bob will correctly answer "far." Therefore the protocol is correct and the communication complexity lower bound implies that $\mathcal{A}$ must use space at least $\Omega(N) = \Omega(\epsilon^{-2}/\log^2(\epsilon^{-1}))$.

□

9.2 Higher-Order Entropy

The kth order entropy is a quantity defined on a sequence that quantifies how easy it is to predict a character of the sequence given the previous k characters. We start with a formal definition.

Definition 9.9. *For a data stream $A = \langle a_1, a_2, \ldots, a_m \rangle$, with each token $a_j \in [n]$, we define*

$$m_{i_1 i_2 \ldots i_t} \;:=\; |\{j \leq m-k : \, a_{j-1+l} = i_l \text{ for } l \in [t]\}| \, ; \quad p_{i_t|i_1,i_2,\ldots,i_{t-1}} \;:=\; \frac{m_{i_1 i_2 \ldots i_t}}{m_{i_1 i_2 \ldots i_{t-1}}} \, ,$$

for $i_1, i_2, \ldots, i_t \in [n]$. The (empirical) kth order entropy *of A is defined as*

$$H_k(A) := -\sum_{i_1} p_{i_1} \sum_{i_2} p_{i_2|i_1} \ldots \sum_{i_{k+1}} p_{i_{k+1}|i_1 \ldots i_k} \lg p_{i_{k+1}|i_1 \ldots i_k} \, .$$

Unfortunately, unlike empirical entropy, H_0, there is no small space algorithm for multiplicatively approximating H_k. This is even the case for H_1 as substantiated in the following theorem.

Theorem 9.10. *Any $(\epsilon, 9/10)$-approximation of $H_1(A)$ requires $\Omega(m/\log m)$ bits of space for any ϵ.*

Proof. Let PREFIX denote the following (one-way) communication problem. Alice has a string $x \in \{0,1\}^N$ and Bob has a string $y \in \{0,1\}^{N'}$ with $N' \leq N$. Alice must send a message to Bob, and Bob must answer "yes" if y is a prefix of x, and "no" otherwise. The one-way probabilistic communication complexity of PREFIX is $\Omega(N/\log N)$, as the following argument shows. Suppose we could solve PREFIX using C bits of communication. Repeating such a protocol $O(\log n)$ times in parallel reduces the probability of failure from constant to $O(1/n)$. But by posing $O(n)$ PREFIX queries in response to Alice's message in this latter protocol, Bob could learn x with failure probability at most a constant. Therefore, we must have $C \log n = \Omega(n)$.

Consider an instance (x, y) of PREFIX. Let Alice and Bob jointly construct the stream $A = \langle a_1, a_2, \ldots, a_N, \, b_1, b_2, \ldots, b_{N'} \rangle$, where $a_i = (i, x_i)$ for $i \in [N]$ and $b_i = (i, y_i)$ for $i \in [N']$. Note that,

$$H_1(A) \;=\; -\sum_i p_i \sum_j p_{j|i} \lg p_{j|i} \;=\; 0$$

if x is a prefix of y. But $H_1(A) \neq 0$ if x is not a prefix of y. This reduction proves that any multiplicative approximation to H_1 requires $\Omega(N/\log N)$ space, using the same logic as that in the conclusion of the proof of Theorem 9.8. Since the stream length $m = N + N' = \Theta(N)$, this translates to an $\Omega(m/\log m)$ lower bound. □

Since the above theorem effectively rules out efficient multiplicative approximation, we now turn our attention to additive approximation. The next theorem (and its proof) shows how the algorithm in Section 2 gives rise to an efficient algorithm that additively approximates the kth order entropy.

Theorem 9.11. *There exists a one-pass, $O(k^2\epsilon^{-2}\log\delta^{-1}\log^2 n\log^2 m)$-space, (ϵ,δ)-additive-approx. for $H_k(A)$.*

Proof. We first rewrite the kth order entropy as follows.

$$\begin{aligned}
H_k(A) &= -\sum_{i_1,i_2,\ldots,i_{k+1}} p_{i_1}p_{i_2|i_1}\ldots p_{i_{k+1}|i_1 i_2\ldots i_k}\lg p_{i_{k+1}|i_1 i_2\ldots i_k} \\
&= \sum_{i_1,i_2,\ldots,i_{k+1}} \frac{m_{i_1\ldots i_{k+1}}}{m-k}\lg\frac{m_{i_1\ldots i_k}}{m_{i_1\ldots i_{k+1}}} \\
&= -\sum_{i_1,i_2,\ldots,i_k} \frac{m_{i_1\ldots i_k}}{m-k}\lg\frac{m-k}{m_{i_1\ldots i_k}} + \sum_{i_1,i_2,\ldots,i_{k+1}} \frac{m_{i_1\ldots i_{k+1}}}{m-k}\lg\frac{m-k}{m_{i_1\ldots i_{k+1}}} \\
&= H(p^{k+1}) - H(p^k)
\end{aligned}$$

where p^k is the distribution over n^k points with $p^k_{i_1 i_2\ldots i_k} = m_{i_1 i_2\ldots i_k}/(m-k)$ and p^{k+1} is the distribution over n^{k+1} points with $p^k_{i_1 i_2\ldots i_{k+1}} = m_{i_1 i_2\ldots i_{k+1}}/(m-k)$. Since $H(p^k)$ is less than $k\lg n$, if we approximate it to a multiplicative factor of at most $(1+\epsilon/(2k\lg n))$ then we have an additive $\epsilon/2$ approximation. Appealing to Theorem 9.5 this can be done in $O(k^2\epsilon^{-2}\log(\delta^{-1})\log^2(n)\log(m))$ space. We can deal with $H(p^{k+1})$ similarly and hence we get an ϵ additive approximation for $H_k(A)$. Directly implementing these algorithms, we need to store strings of k characters from the input stream as a single kth order character; for large k, we can hash these strings onto the range $[m^2]$. Since there are only $m-k$ substrings of length k, then there are no collisions in this hashing w.h.p., and the space needed is only $O(\log m)$ bits for each stored item or counter.

□

9.3 Entropy of a Random Walk

In Theorem 9.10 we showed that it was impossible to multiplicatively approximate the first order entropy, H_1, of a stream in sub-linear space. In this section we consider a related quantity H_G, the *unbiased random walk entropy*. We will discuss the nature of this relationship after a formal definition.

Definition 9.12. *For a data stream $A = \langle a_1, a_2, \ldots, a_m\rangle$, with each token $a_j \in [n]$, we define an undirected graph $G(V, E)$ on n vertices, where $V = [n]$ and $E = \{\{u, v\} \in [n]^2 : u = a_j, v = a_{j+1}$ for some $j \in [m-1]\}$. Let d_i be the degree of node i. Then the* unbiased random walk entropy *of A is defined as,*

$$H_G \ := \ \frac{1}{2|E|}\sum_{i\in[n]} d_i \lg d_i \,.$$

Consider a stream formed by an unbiased random walk on an undirected graph G, i.e., if $a_i = j$ then a_{i+1} is uniformly chosen from the d_j neighbors of j. Then H_G is the limit of $H_1(A)$ as the length of this random walk tends to infinity:

$$\frac{1}{2|E|}\sum_{i\in[n]} d_i \lg d_i \ = \ \lim_{m\to\infty}\sum_{i\in[n]}\frac{m_i}{m}\sum_{j\in[n]}\frac{m_{ij}}{m_i}\lg\frac{m_i}{m_{ij}} \ = \ \lim_{m\to\infty} H_1(\langle a_1, a_2, \ldots, a_m\rangle)\,,$$

since $\lim_{m\to\infty}(m_{ij}/m_i) = 1/d_i$ and $\lim_{m\to\infty}(m_i/m) = d_i/(2|E|)$ as the stationary distribution of a random walk on an undirected graph is

$$(d_1/(2|E|), d_2/(2|E|), \ldots, d_n/(2|E|)) \ .$$

See Section 4.3 of Cover and Thomas [CT91], for example, for more context. We focus on computing H_G rather than on computing the entropy of a sample walk, since this gives greater flexibility: it can be computed on arbitrary permutations of the edges, for example, and only requires that each edge be observed at least once.

For the rest of this section it will be convenient to reason about a stream E' that can be easily transduced from A. E' will consist of $m-1$, not necessarily distinct, edges on the set of nodes $V = [n]$, $E' = \langle e_1, e_2, \ldots, e_{m-1} \rangle$ where $e_i = (a_i, a_{i+1})$. Note that E is the set produced by removing all duplicate edges in E'.

Overview of the algorithm: Our algorithm uses the standard AMS-Estimator as described in Section 9.1. However, because E' includes duplicate items which we wish to disregard, our basic estimator is necessarily more complicated. The algorithm combines ideas from multi-graph streaming [CM05b] and entropy-norm estimation [CDM06] and uses min-wise hashing [Ind01] and distinct element estimators [BYJK$^+$02].

Ideally the basic estimator would sample a node w uniformly from the multi-set in which each node u occurs d_u times. Then let r be uniformly chosen from $\{1, \ldots, d_w\}$. If the basic estimator were to return $g(r) = f(r) - f(r-1)$ where $f(x) = x \lg x$ then the estimator would be correct in expectation:

$$\sum_{w \in [n]} \frac{d_w}{2|E|} \sum_{r \in [d_w]} \frac{1}{d_w}(f(r) - f(r-1)) = \frac{1}{2|E|} \sum_{w \in [n]} d_w \lg d_w \ .$$

To mimic this sampling procedure we use an ϵ-min-wise hash function h [Ind01] to map the distinct edges in E' into $[m]$. It allows us to pick an edge $e = (u, v)$ (almost) uniformly at random from E by finding the edge e that minimizes $h(e)$. We pick w uniformly from $\{u, v\}$. Note that w has been chosen with probability proportional to $(1 \pm \epsilon)\frac{d_w}{2|E|}$. Let $i = \max\{j : e_j = e\}$ and consider the r distinct edges among $\{e_i, \ldots, e_m\}$ that are incident on w. Let $e^1, \ldots, e^{d_w}$ be the d_w edges that are incident on w and let $i_k = \max\{j : e_j = e^k\}$ for $k \in [d_w]$. Then r is distributed as $|\{k : i_k \geq i\}|$ and hence takes a value from $\{1, \ldots, d_w\}$ with probability $(1 \pm \epsilon)/d_w$.

Unfortunately we cannot compute r exactly unless it is small. If $r \leq \epsilon^{-2}$ then we maintain an exact count, by keeping the set of distinct edges. Otherwise we compute an (ϵ, δ)-approximation of r using a distinct element estimation algorithm, e.g. [BYJK+02]. Note that if this is greater than n we replace the estimate by n to get a better bound. This will be important when bounding the maximum value of the estimator. Either way, let this (approximate) count be $\tilde{r}$. We then return $g(\tilde{r})$. The next lemma demonstrates that using $g(\tilde{r})$ rather than $g(r)$ only incurs a small amount of additional error.

Lemma 9.13. *Assuming* $\epsilon < 1/4$, $\Pr[|g(r) - g(\tilde{r})| \le O(\epsilon) g(r)] \ge 1 - \delta$.

Proof. If $r \le \epsilon^{-2}$, then $r = \tilde{r}$, and the claim follows immediately. Therefore we focus on the case where $r > \epsilon^{-2}$. Let $\tilde{r} = (1+\gamma)r$ where $|\gamma| \le \epsilon$. We write $g(r)$ as the sum of the two positive terms,

$$g(r) = \lg(r-1) + r\lg(1 + 1/(r-1))$$

and will consider the two terms in the above expression separately.

Note that for $r \ge 2$, $\frac{\tilde{r}-1}{r-1} = 1 \pm 2\epsilon$. Hence, for the first term, and providing the distinct element estimation succeeds with its accuracy bounds,

$$|\lg(\tilde{r}-1) - \lg(r-1)| \ = \ \left|\lg\frac{\tilde{r}-1}{r-1}\right| \ = \ O(\epsilon) \ \le \ O(\epsilon)\lg(r-1)\,.$$

where the last inequality follows since $r > \epsilon^{-2}$, $\epsilon < \frac{1}{4}$, and hence $\lg(r-1) > 1$.

Note that for $r \ge 2$, $r\lg\left(1 + \frac{1}{r-1}\right) \ge 1$. For the second term,

$$\left| r\lg\left(1 + \frac{1}{r-1}\right) - \tilde{r}\lg\left(1 + \frac{1}{\tilde{r}-1}\right)\right| \le O(\epsilon) r\lg\left(1 + \frac{1}{r-1}\right)\,.$$

Hence $|g(r) - g(\tilde{r})| \le O(\epsilon)g(r)$ as required. □

Theorem 9.14. *There is a one-pass, $\tilde{O}(\epsilon^{-4})$-space, (ϵ, δ)-approx. for H_G.*

Proof. Consider the expectation of the basic estimator:

$$\mathrm{E}[X] = \sum_{w\in[n]} \frac{(1 \pm O(\epsilon))d_w}{2|E|} \sum_{r\in[d_w]} \frac{1 \pm O(\epsilon)}{d_w}(f(r) - f(r-1)) = \frac{1 \pm O(\epsilon)}{2|E|}\sum_{w\in[n]} d_w \lg d_w\,.$$

Note that since the graph G is revealed by a random walk, this graph must be connected. Hence $|E| \geq n-1$ and $d_w \geq 1$ for all $w \in V$. But then $\sum_w d_w = 2|E| \geq 2(n-1)$ and therefore,

$$\frac{1}{2|E|}\sum_{w\in[n]} d_w \lg d_w \;\geq\; \lg\frac{2|E|}{n} \;\geq\; \lg 2(1-1/n)\,.$$

The maximum value taken by the basic estimator is,

$$\max[X] \leq \max_{1\leq r\leq n}(f(r)-f(r-1)) \leq \left(n\lg\frac{n}{n-1}+\lg(n-1)\right) < (2+\lg n)\,.$$

Therefore, by appealing to Lemma 9.2, if we take c independent copies of this estimator we can get a (ϵ,δ)-approximation to $\mathrm{E}[X]$ if $c \geq 6\epsilon^{-2}(2+\lg n)\ln(2\delta^{-1})/(\lg 2(1-1/n))$. Hence, with probability $1-O(\delta)$, the value returned is $(1\pm O(\epsilon))H_G$.

The space bound follows because for each of the $O(\epsilon^{-2}\log n\log\delta^{-1})$ basic estimators we require an ϵ min-wise hash function using $O(\log n\log\epsilon^{-1})$ space [Ind01] and a distinct element counter using $O((\epsilon^{-2}\log\log n+\log n)\log\delta_1^{-1})$ space [BYJK+02] where $\delta_1^{-1}=O(c\delta^{-1})$. Hence, rescaling ϵ and δ gives the required result. □

Our bounds are independent of the length of the stream, m, since there are only n^2 distinct edges, and our algorithms are not affected by multiple copies of the same edge. Note that our algorithm is still correct if the multi-set of edges E' arrives in an arbitrary order, i.e., it is not necessary that (u,v) is followed by (v,w) for some w. Hence, it also belongs to the graph-stream paradigm discussed in Chapter 10.

9.4 Testing Entropy (Combined Oracle)

In this section we present a simple algorithm that achieves the optimal bounds for estimating the entropy in the combined oracle model. This algorithm improves upon the previous upper bound of Batu et al. [BDKR05] by a factor of $\Omega(\log n/H)$ where H is the entropy of the distribution. The authors of [BDKR05] showed that their algorithms were tight for $H=\Omega(\log n)$; we show that the upper and lower bounds

```
Algorithm Combined Oracle Entropy Testing
1.  $E \leftarrow 0$
2.  for $t = 1$ to $m$:
3.      do $i \leftarrow$ sample($p$)
4.          $p_i \leftarrow$ probe($p, i$)
5.          if $p_i \geq n^{-3}$ then $a \leftarrow \lg(1/p_i)/(3\lg n)$ else $a \leftarrow 0$
6.          $E \leftarrow a + E$
7.  return $3E\lg n/m$
```

Figure 9.2: Entropy-Testing (Combined Oracle Model)

match for arbitrary H. The algorithm is presented in Figure 9.2. It is structurally similar to the algorithm given in [BDKR05] but uses a cutoff that will allow for a tighter analysis via Chernoff bounds.

The next lemma estimates the contribution of the unseen elements and that leads to the main theorem about estimating entropy in the combined oracle model.

Lemma 9.15. *For any $S \subset [n]$, $\lg 1/p_i \leq \lg(n/\sum_{i\in S} p_i)\sum_{i\in S} p_i$.*

Theorem 9.16. *There exists an (ϵ, δ)-approximation algorithm for $H(p)$ making $O(\epsilon^{-2}H^{-1}\log(n)\log(\delta^{-1}))$ combined-oracle queries.*

Proof. We restrict our attention to the case when $H(p) > 1/n$ and $\epsilon > 1/\sqrt{n}$ since otherwise we can trivially find the entropy exactly in $O(1/\epsilon^2 H(p))$ time by simply probing each $i \in [n]$. Consider the value a added to E in each iteration. This is a random variable with range $[0, 1]$ since $p_i \geq 1/n^3$ guarantees that $-\lg(1/p_i)/(3\lg n) \leq 1$. Now, the combined mass of all p_i such that $p_i < 1/n^3$ is at most $1/n^2$. Therefore, by Lemma 9.15 the maximum contribution to the entropy from such i is at most $3n^{-2}\lg n \leq \epsilon H(p)/2$ for sufficiently large n. Hence,

$$(1 - \epsilon/2)H(p)/(3\lg n) \leq E[a] \leq H(p)/(3\lg n),$$

and therefore, if we can $(\epsilon/2, \delta)$-approximate $E[a]$ then we are done. By applying the Chernoff-Hoeffding bounds, this is achieved for the chosen value of m. Therefore with $O(\epsilon^{-2}H^{-1}\log(n)\log(\delta^{-1}))$ samples/probes the probability that we do not $1+\epsilon/2$ approximate $E[a]$ is at most δ. $\square$

Part III

Graph Streams

Chapter 10

Introduction

10.1 Graph Streams

In this section of the thesis we present the first comprehensive treatment of streaming problems in which the stream defines an undirected, unweighted, simple graph in the following way.

Definition 10.1 (Graph Stream)**.** *For a data stream $A = \langle a_1, a_2, \ldots, a_m \rangle$, with each data item $a_j \in [n] \times [n]$, we define a graph G on n vertices $V = \{v_1, \ldots, v_n\}$ with edges $E = \{(v_i, v_k) : a_j = (i, k) \text{ for some } j \in [m]\}$.*

We normally assume that each a_j is distinct although this assumption is often not necessary. When the data items are not distinct, the model can naturally be extended to consider multi-graphs, i.e., an edge (v_i, v_k) has multiplicity equal to $|\{j : a_j = (i, k)\}|$. Similarly, we mainly consider undirected graphs but the definition can be generalized to define directed graphs. Sometimes we will consider weighted graphs and in this case $a_j \in [n] \times [n] \times \mathbb{N}$ where the third component of the data item indicates a weight associated with the edge. Note that some authors have also considered a special case of the model, the adjacency-list model, in which all incident edges are grouped together in the stream [BYKS02], we will be primarily interested in the fully general model.

Massive graphs arise naturally in many real world scenarios. Two examples are the *call-graph* and the *web-graph*. In the call-graph, nodes represent telephone numbers and edges correspond to calls placed during some time interval. In the web-graph, nodes represent web pages, and the edges correspond to hyper-links between pages. Also, massive graphs appear in structured data mining, where the relationships among the data items in the data set are represented as graphs. When processing these graphs it is often appropriate to use the streaming model. For example, the graph may be revealed by a web-crawler or the graph may be stored on external memory devices and being able to process the edges in an arbitrary order improves I/O efficiency. Indeed, the authors of [KRRT99] argue that one of the major drawbacks of standard graph algorithms, when applied to massive graphs such as the web, is their need to have random access to the edge set.

10.1.1 Related Work

Prior to the work contained in this thesis there was only a few results concerning graph streams [HRR99, BYKS02, BGW03]. Subsequently, the area has attracted more interest [JG05, CM05b, EZ06, BFL$^+$06, Zel06, GS06, Bas06, Elk07] and a couple of other papers have considered graph problems in a various extensions of the streaming model [ADRR04, DFR06]. Much of this work is of a statistical nature, e.g. counting triangles [BFL$^+$06, BYKS02, JG05] or estimating frequency and entropy moments of the degrees in a multi-graph [CM05b, CCM07]. It seemed that more "complicated" computation was not possible in this model. For example, Buchsbaum et al. [BGW03] demonstrated the intrinsic difficultly of computing common neighborhoods in the streaming model with small space.

In general it seems that most graph algorithms need to access the data in a very adaptive fashion. Since we can not store the entire graph (this would require $O(m)$ space), emulating a traditional algorithm may necessitate an excessive number of passes over the data. One possible alternative is to consider algorithms that use

$O(n \operatorname{polylog} n)$ space, i.e., space proportional to the number of nodes rather than the number of edges. For a massive dense graph this requires considerably less space than storing the entire graph. This restriction was identified as an apparent "sweet-spot" for graph streaming in a survey article by Muthukrishnan [Mut06] and dubbed the *semi-streaming* space restriction in Feigenbaum et al. [FKM$^+$05b]. This spurred further research on designing algorithm for solving graph problems in the streaming model such as distance estimation [FKM$^+$05b, FKM$^+$05a, EZ06, Bas06, Elk07], matchings [FKM$^+$05b, McG05] and connectivity [FKM$^+$05b, Zel06]. We will provide further discussion on the results on distance estimation and matching in the next two sections.

A related model is the *semi-external model*. This was introduced by Abello et al. [ABW02] for computations on massive graphs. In this model the vertex set can be stored in memory, but the edge set cannot. However, this work addresses the problems in an external memory model in which random access to the edges, although expensive, is allowed. Lastly, graph problems have been considered in a model that extends the stream model by allowing the algorithm to write to the stream during each pass [ADRR04, DFR06]. These *annotations* can then be utilized by the algorithm during successive passes. [ADRR04] goes further and suggests a model in which *sorting passes* are permitted in which the data stream is sorted according to a key encoded by the annotations.

10.2 Distances

In this section we present results pertaining to the estimation of graph distances in the data stream model. Graph distance is a natural quantity when trying to understand properties of massive graphs such as the diameter of the world-wide-web [AJB99]. We start with a formal definition of the relevant terms.

Definition 10.2 (Graph Distance, Diameter, and Girth). *For an undirected, unweighted graph $G = (V, E)$ we define a distance function $d_G : V \times V \to \{0, \ldots, n-1\}$ where $d_G(u, v)$ is the length of the shortest path in G between u and v. The diameter of G is the length of the longest shortest path, i.e.,*

$$\mathrm{Diam}(G) = \max_{u,v \in V} d_G(u, v) \ .$$

The girth of G is the length of the shortest cycle in G, i.e.,

$$\mathrm{Girth}(G) = 1 + \min_{(u,v) \in E} d_{G \setminus (u,v)}(u, v) \ .$$

The above definitions extend naturally to weighted graphs.

10.2.1 Related Work

Designing algorithms for computing graph distances is a well studied problem in computer science. Classic algorithms such as Dijkstra's algorithm, the Bellman-Ford algorithm and the Floyd-Warshall algorithm are taught widely [CLRS01]. Recent research has focused on computing approximate graph distances [ABCP98, Elk01, TZ01, BS03]. Unfortunately these algorithms are inherently unsuitable for computing distances in the streaming model. In particular, an important sub-routine of many of the existing algorithms is the construction of Breadth-First-Search (BFS) trees. One of our main results is a lower-bound on the number of passes required by an algorithm that computes a BFS tree.

Algorithms for approximating distances in the streaming model were presented in [FKM+05a, FKM+05b, EZ06, Bas06, Elk07]. These algorithms approximate distance by constructing *spanners*.

Definition 10.3 (Spanners). *A subgraph $H = (V, E')$ is a (α, β)-*spanner *of $G = (V, E)$ if, for any vertices $x, y \in V$,*

$$d_G(x, y) \le d_H(x, y) \le \alpha \cdot d_G(x, y) + \beta \ .$$

In [FKM$^+$05a], we present a randomized streaming algorithm that constructs a $(2t+1, 0)$-spanner in one pass. With probability $1 - 1/n^{\Omega(1)}$, the algorithm uses $O(tn^{1+1/t} \log^2 n)$ bits of space and processes each edge in $O(t^2 n^{1/t} \log n)$ time. Using this spanner, all distances in the graph can be approximated up to a factor of $(2t+1)$. This improves upon an algorithm that can be found in [FKM$^+$05b]. That construction had similar space requirements but was much slower taking $O(n)$ time to process each edge. Both algorithms can be generalized to weighted graphs. Recent results have further improved this construction [Bas06, Elk07]. [EZ06] present algorithms for constructing (α, β)-spanners. However, these algorithms require multiple passes over the stream.

10.2.2 Our Results

In these section we focus on negative results. The lower-bounds we present complement the algorithms presented in [FKM$^+$05a, Zha05]

1. *Connectivity and Balanced Properties:* We show that testing any of a large class of graph properties, which we refer to as *balanced properties*, in one pass requires $\Omega(n)$ space. This class includes properties such as connectivity and bipartite-ness. This result provides a formal motivation for the *semi-streaming* space restriction where algorithms are permitted $O(n \operatorname{polylog} n)$ space.

2. *Graph Distances and Graph Diameter:* We show that any single pass algorithm that approximates the (weighted) graph distance between two given nodes up to a factor $1/\gamma$ with probability at least $3/4$ requires $\Omega(n^{1+\gamma})$ bits of space. Furthermore, this bound also applies to estimating the diameter of the graph. The lower-bound shows that the algorithms in [FKM$^+$05a, Zha05] are close to optimal.

3. *Breadth-First-Search Trees:* Let γ be a constant in the range $(0, 1)$ and $l \in [\lfloor 1/\gamma \rfloor]$. Computing the first l layers of a BFS tree from a prescribed node

requires either $\lceil (l-1)/2 \rceil$ passes or $\Omega(n^{1+\gamma})$ bits of space. On the other hand, it will be trivial to construct the first l layers of a BFS tree with l passes even in space $O(n \log n)$. Constructing BFS trees is a very common sub-routine in many graph algorithms. Hence this result demonstrates the need for substantially different algorithmic techniques when computing in the streaming model.

4. *Girth:* Any P-pass algorithm that ascertains whether the length of the shortest cycle is longer than g, requires $\Omega\left(P^{-1}(n/g)^{1+4/(3g-4)}\right)$ bits of space.

Trade-offs: The above results indicate various trade-offs between model parameters and accuracy. This include the smooth trade-off between the space a single-pass algorithm is permitted and the accuracy achievable when estimating graph distances. For multiple-pass algorithms, a smooth trade-off between passes and space is evident when trying to compute the girth of a graph. This trade-off is, in a sense. fundamental as it indicates that the only way to get away with using half the amount of space is essentially to make half as much progress in each pass. The trade-off between space and passes when computing BFS-trees indicates that as we restrict the space, no algorithm can do much better than emulating a trivial traditional graph algorithm and will consequently require an excessive number of passes.

10.3 Matchings

In this section we present algorithms for computing *maximum cardinality matching* and *maximum weighted matching*. We start with basic definitions for these problems before reviewing related work.

Definition 10.4 (Matching). *Given a graph $G = (V, E)$, the* Maximum Cardinality Matching *(MCM) problem is to find the largest set of edges such that no two adjacent edges are selected. More generally, for an edge-weighted graph, the* Maximum

Weighted Matching *(MWM) problem is to find the set of edges whose total weight is maximized subject to the condition that no two adjacent edges are selected.*

10.3.1 Related Work

Computing large matchings is classic graph problem. Exact polynomial solutions are known [Edm65, Gab90, HK73, MV80] for MCM and MWM. The fastest of these algorithms solves the maximum weighted matching problem with running time $O(nm + n^2 \log n)$ where $n = |V|$ and $m = |E|$.

However, for massive graphs in real world applications, the above algorithms can still be prohibitively slow. Consequently there has been much interest in faster algorithms, typically of $O(m + n)$ complexity, that find good approximate solutions to the above problems. For MCM, a linear time approximation-scheme was given by Kalantari and Shokoufandeh [KS95]. The first linear time approximation algorithm for MWM was introduced by Preis [Pre99]. This algorithm achieved a 1/2 approximation ratio. This was later improved upon by the $(2/3 - \epsilon)$ linear time[1] approximation algorithm given by Drake and Hougardy [DH03]. A simplified version of this result was given by Pettie and Sanders [PS04].

In addition to concerns about time complexity, when computing with massive graphs it is no longer reasonable to assume that we can store the entire input graph in *random access memory.* In this case the above algorithms are not applicable as they require random access to the input. With this in mind, we consider the problem in the graph stream model.

MCM and MWM have previously been studied under similar assumptions by Feigenbaum et al. [FKM$^+$05b]. The best previously attained results were a 1/6 approximation to MWM and for $\epsilon > 0$ and a $(2/3 - \epsilon)$-approximation to MCM on the assumption that the graph is a bipartite graph. We improve upon both of these results.

[1]Note that here, and throughout this section, we assume that ϵ is an arbitrarily small *constant.*

10.3.2 Our Results

We present the following $O(m)$-time, $O(n \operatorname{polylog} n)$-space algorithms:

1. *Maximum Cardinality Matching:* A single pass, 1/2-approximation for maximum cardinality matchings. For any $\epsilon > 0$, $O_\epsilon(1)$ pass $(1-\epsilon)$-approximation.

2. *Maximum Weighted Matching:* A single pass, $1/(3+2\sqrt{2})$-approximation for maximum weighted matchings. For $\epsilon > 0$, a $O_\epsilon(1)$ pass $(1/2-\epsilon)$-approximation.

Trade-offs: The trade-offs that are implicit in our algorithms are more positive than those discussed in the previous section. While the single pass algorithms will be relatively straightforward, the main challenge in improving upon the matchings that can be found in one pass will be how to fully utilize the successive passes. For both the weighted and unweighted case we will show how to "grow" a matching using a small number of passes and thereby achieve the claimed approximation factors.

Chapter 11

Graph Distances

Chapter Outline: In this chapter we present the technical details behind the results related to graph distance in the data-stream model. We start by showing that testing a range of basic graph properties such as whether a graph is connected or bipartite requires space proportional the number of nodes. This justifies the semi-streaming model. We then present a lower-bounds on the space required to approximate graph distances or to construct breadth-first-search trees. We conclude with a lower-bound on the space required to determine the girth of a graph. For background see Chapter 10.

11.1 Connectivity and other Balanced Properties

Our first result shows that a large class of problems can not be solved by a single pass streaming algorithm in small space. Specifically, we identify a general type of graph property[1] and show that testing any such graph property requires $\Omega(n)$ space.

Definition 11.1 (Balanced Properties). *We say a graph property $\mathcal{P}$ is* balanced *if there exists a constant $c > 0$ such that for all sufficiently large n, there exists a graph*

[1] A graph property is simply a boolean function whose variables are the elements of the adjacency matrix of the graph but whose value is independent of the labeling of the nodes of the graph.

$G = (V, E)$ with $|V| = n$ and $u \in V$ such that:

$$\min\{|\{v : (V, E \cup \{(u, v)\}) \text{ has } \mathcal{P}\}|, |\{v : (V, E \cup \{(u, v)\}) \text{ has } \neg\mathcal{P}\}|\ \} \geq cn\ .$$

Note that many interesting properties are balanced including connectivity, bipartiteness, and whether there exists a vertex of a certain degree.

Theorem 11.2. *Testing any balanced graph property $\mathcal{P}$ with probability $2/3$ requires $\Omega(n)$ space.*

Proof. Let c be a constant, $G = (V, E)$ be a graph on n vertices and $u \in V$ be a vertex satisfying the relevant conditions.

The proof is by a reduction to the communication complexity of Index. Let $(x, j) \in \mathbb{F}_2^{cn} \times [cn]$ be an instance of Index. Let $G(x)$ be a relabeling of the vertices of G such that $u = v_n$ and for $i \in [cn]$, $(V, E \cup \{(v_n, v_i)\})$ has $\mathcal{P}$ iff $x_i = 1$. Such a relabeling is possible because $\mathcal{P}$ does not depend on the labeling of the vertices. Let $e(j) = (v_j, v_n)$. Hence the graph determined by the edges of $G(x)$ and $e(j)$ has $\mathcal{P}$ iff $x_j = 1$. Therefore, any single pass algorithm for testing $\mathcal{P}$ using M bits of work space gives rise to a one-message protocol for solving Index. Therefore $M = \Omega(cn)$. □

For some balanced graph properties the above theorem can be generalized. For example it is possible to show that any p-pass algorithm that determines if a graph is connected requires $\Omega(np^{-1})$ bits of space [DFR06].

11.2 Graph-Distances and Graph-Diameter

In this section we show a lower-bound on the amount of space required to approximate the length of the shortest path between two nodes. Our bound also applies to estimating the diameter of the graph. Integral to our proof is the notion of an edge being *k-critical.*

Definition 11.3. *In a graph $G = (V, E)$, an edge $e = (u, v) \in E$ is* k-critical *if $d_{G \setminus (u,v)}(u, v) \geq k$.*

In Lemma 11.4 we show the existence of a graph G with a large subset of edges E' such that each edge in E' is k-critical but the removal of all edges in E' still leaves a graph with relatively small diameter. The proof is by a probabilistic argument.

Lemma 11.4. *For any $\gamma > 0$, $k = \lfloor 1/\gamma - \epsilon \rfloor$ (for some arbitrarily small constant $\epsilon > 0$) and sufficiently large n, there exists a set E of edges partitioned into two disjoint sets E_1 and E_2 on a set of n nodes V such that,*

1. $|E_2| = n^{1+\gamma}/64$.
2. *Every edge in E_2 is k-critical for $G = (V, E)$.*
3. $\mathrm{Diam}(G_1) \leq 2/\gamma$ *where $G_1 = (V, E_1)$.*

Proof. Consider choosing a random graph $G' = (V, E')$ on n nodes where each edge is present with probability $p = 1/(2n^{1-\gamma})$. This is commonly denoted as $G' \sim \mathcal{G}_{n,p}$. We will then construct $G_1 = (V, E_1)$ by deleting each edge in G' with probability $1/2$. We will show that with non-zero probability the sets E_1 and

$$E_2 = \{e \in E' \setminus E_1 : e \text{ is } k\text{-critical for } G'\},$$

satisfy the three required properties. Hence, there exist sets with the required properties.

The second property is satisfied by construction. It follows from the fact that if an edge is k-critical in a graph G, then it is also k-critical in any subgraph of G. We now argue that the third property is satisfied with probability at least $9/10$. First note that the process that generates G_1 is identical to picking $G_1 \sim \mathcal{G}_{n,p/2}$. It can be shown that with high probability, the diameter of such a graph is less than $2/\gamma$ for sufficiently large n [Bol85, Corollary 10.12].

We now show that the first property is satisfied with probability at least $9/10$. Applying the Chernoff bound and the union bound proves that with probability at least $99/100$, the degree of every vertex in G' is between $n^\gamma/4$ and n^γ.

Now consider choosing a random graph and a random edge in that graph simultaneously, i.e., $G' = (V, E') \sim \mathcal{G}_{n,p}$ and an edge $(u, v) \in_R E'$. We now try to lower-bound the probability that(u, v) is k-critical in G'. Let $\Gamma_i(v) = \{w \in V : d_{G'\setminus(u,v)}(v, w) \leq i\}$. For sufficiently large n,

$$|\Gamma_k(v)| \leq \sum_{0 \leq i \leq k} n^{i\gamma} \leq 2n^{k\gamma} \leq (n-1)/100 \ .$$

As G' varies over all possible graphs, by symmetry, each vertex is equally likely to be in $\Gamma_k(v)$. Thus the probability that u is not in this set is at least 99/100. By Markov's inequality,

$$\Pr\left[|\{(u, v) \in E' : d_{G'\setminus(u,v)}(u, v) \geq k\}| \geq |E'|/2\right] \geq 98/100 \ .$$

Note that if the degree of every vertex in G' is at least $n^{\gamma}/4$ then $|E'| \geq n^{1+\gamma}/8$. Hence,

$$\Pr\left[|\{(u, v) \in E' : d_{G'\setminus(u,v)}(u, v) \geq k\}| \geq n^{1+\gamma}/16\right] \geq 97/100 \ .$$

Given that each edge in E' is independently deleted with probability 1/2 to form E_1, by a further application of the Chernoff bound we deduce that,

$$\Pr\left[|\{(u, v) \in E' \setminus E_1 : d_{G'\setminus(u,v)}(u, v) \geq k\}| \geq n^{1+\gamma}/64\right] \geq 96/100 \ .$$

From this set of k-critical edges we can certainly choose a subset whose size is exactly $n^{1+\gamma}/64$ as required by statement 1.

Therefore all three properties hold with probability at least $1 - 2/10 = 4/5$. $\square$

Theorem 11.5. *For any constant $\gamma > 0$, any single pass algorithm that, with probability at least 3/4, returns $\tilde{D}$ such that for an arbitrarily small $\epsilon > 0$,*

$$\text{Diam}(G) \leq \tilde{D} \leq (\lfloor 1/\gamma - \epsilon \rfloor - 1)\, \text{Diam}(G)$$

where G is a weighted graph on n nodes, requires $\Omega(n^{1+\gamma})$ space.

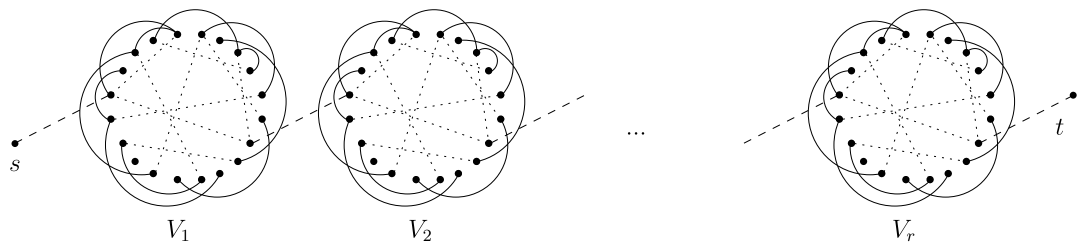


Figure 11.1: Diameter Lower-Bound Construction. Edges $E_x/E_j/E_m$ are dotted/dashed/solid.

Proof. Let $(x, j) \in \mathbb{F}_2^t \times [t]$ by an instance of INDEX. We will show how to transform an algorithm $\mathcal{A}$ for approximating the diameter of a graph into a protocol for INDEX.

Let $G = (V, E = E_1 \cup E_2)$ be a graph on $n' = (64t)^{1/(1+\gamma)}$ nodes with the properties listed in Lemma 11.4. G is hardwired into the protocol. An enumeration of the edges in E_2, $e_1, \ldots, e_t$ is also hardwired into the protocol.

Alice forms the graph $G_x = (V, E_m \cup E_x)$ where, $E_x = \{e_i \in E_2 : x_i = 1\}$ and $E_m = E_1$. She then creates the prefix of a stream by taking r (to be determined later) copies of G_x, i.e., a graph on $n'r$ vertices $\{v_1^1, \ldots, v_{n'}^1, v_1^2, \ldots, v_{n'}^2, v_1^3, \ldots, v_{n'}^r\}$ and with edge set, $\{(v_j^i, v_k^i) : i \in [r], (v_j, v_k) \in E_x\}$. All these edges have unit weight.

Let j be the index in the instance of INDEX and let $e_j = (a, b)$. Bob determines the remaining edges E_j as follows: $r-1$ edges of zero weight, $\{(v_b^i, v_a^{i+1}) : i \in [r-1]\}$, and two edges of weight $2k+2$, (s, v_a^1) and (v_b^r, t). See Fig. 11.1.

Regardless of the values of x and j, the diameter of the graph described by the stream equals $d_G(s, t)$. Note that $x_j = 1$ implies $d_G(s, t) = r + 4k + 3$. However, if $x_j = 0$ then $d_G(s, t) \geq k(r-1) + 4k + 4$. Hence for $r = 1 + (4k+4)(k-2)$, the ratio between $k(r-1) + 4k + 4$ and $r + 4k + 3$ is at least $k - 1$. Hence any single-pass algorithm that approximates the diameter to within a factor $k - 1$ gives rise to a one-way protocol for solving INDEX. Therefore, any such algorithm requires $\Omega(n^{1+\gamma})$ bits of space since the total number of nodes in the construction is $n = O((64t)^{1/(1+\gamma)} k^2)$. □

11.3 Constructing BFS-Trees

In this section we prove a lower bound on the number of passes required to construct the first l layers of breadth first search tree in the streaming model. The result is proved using a reduction from the communication-complexity problem "multi-valued pointer chasing." This is a naturally defined generalization of the pointer-chasing problem considered by Nisan and Wigderson [NW93]. We prove the first results on the multi-round communication complexity of the problem.

Overview of Proof: Nisan and Wigderson [NW93] considered the problem where Alice and Bob have functions f_A and f_B respectively, mapping $[m]$ to $[m]$. The k-round pointer chasing problem is to output the result of starting from 1 and alternatively applying f_A and f_B a total of k times, starting with f_A. Nisan and Wigderson proved that if Bob speaks first the communication complexity of any k-round communication protocol to solve this problem is $\Omega(m/k^2 - k \log m)$. Jain, Radhakrishnan, and Sen [JRS03] gave a direct sum extension showing that if there are t pairs of functions and the goal is to perform k-round pointer chasing as above on each pair, the communication complexity lower bound is approximately t times the bound of [NW93]. More precisely, they showed a lower bound of $\Omega(tm/k^3 - tk \log m - 2t)$ for the problem.

We show how the lower bound of [JRS03] also implies a lower bound on the communication complexity of pointer chasing with t-valued functions. If f_A and f_B are such functions, then the result of pointer chasing starting from 1 produces a set of size at most t^k. The key difference between this problem and the problem of [JRS03] is that in the problem of [JRS03] we are only concerned with chasing "like" pointers. In other words, if we get to an element j using the function f_A^i, then we can only continue with f_B^i. Nevertheless, we show by our reduction that the two problems have fairly similar communication complexity.

Finally, we create a layered graph with l layers in which alternate layers have edges corresponding to d-valued functions f_A and f_B. In order to construct the breadth first search tree, we must solve the l-round, d-valued pointer chasing problem and the lower bound above applies. This will lead to the following theorem.

Theorem 11.6 (BFS Lower Bound)**.** *Let γ be a constant in the range $(0, 1)$ and let $l \in \{1, \ldots, 1/\gamma\}$. Computing the first l layers of a BFS from a prescribed node with probability at least $2/3$ requires $(l-1)/2$ passes or $\Omega(n^{1+\gamma})$ space.*

Formal Argument: We now present the above argument formally. In the following definition, for a function $f : [m] \to [m]$ and set $A \subset [m]$ we denote,

$$f(A) := \{j : f(i) = j \text{ for some } i \in A\} \ .$$

Definition 11.7 (d-valued Pointer Chasing)**.** *Let F_d be the set of all d-valued functions from $[m]$ to $[m]$. Define $g_{d,k} : F_d \times F_d \to \mathcal{P}([m])$ by $g_{d,0}(f_A, f_B) = 1$ and,*

$$g_{d,i}(f_A, f_B) = \begin{cases} f_A(g_{d,i-1}(f_A, f_B)) & \text{if } i \text{ odd} \\ f_B(g_{d,i-1}(f_A, f_B)) & \text{if } i \text{ even} \end{cases} ,$$

where $\mathcal{P}([m])$ denotes the power-set of $[m]$. Note that $g_{d,k}$ is a set of size at most d^k. Let $\bar{g}_{d,k} : F_d \times F_d \to \mathcal{P}([m])^k$ be the function,

$$\bar{g}_{d,k}(f_A, f_B) = \langle g_{d,1}(f_A, f_B), \ldots, g_{d,k}(f_A, f_B)\rangle \ .$$

Let $g^t_{1,k} : F^t_1 \times F^t_1 \to \mathcal{P}([m])^t$ be the t-fold direct sum of $g_{1,k}$, i.e.,

$$g^t_{1,k}(\langle f^1_A, \ldots, f^t_A\rangle, \langle f^1_B, \ldots, f^t_B\rangle) = \langle g_{1,k}(f^1_A, f^1_B), \ldots, g_{1,k}(f^t_A, f^t_B)\rangle \ .$$

Let Alice have function f_A and Bob have function f_B. Let $R^r_\delta(g_{d,k})$ be the r-round randomized communication complexity of $g_{d,k}$ where Bob speaks first, i.e., the number of bits sent in the worst case (over all inputs and random coin tosses) by the best r-round protocol Π in which, with probability at least $1-\delta$, both Alice and Bob learn $g_{d,k}$.

Theorem 11.8 (Nisan, Wigderson [NW93]). $R^k_{\mu_1,1/3}(g_{1,k}) = \Omega(m/k^2 - k\log m)$

The following *direct-sum* theorem for $g_{1,k}$ is proved by [JRS03] using the notion of the information complexity.

Theorem 11.9 (Jain et al. [JRS03]). $R^k_{1/4}(g^t_{1,k}) = \Omega(tmk^{-3} - tk\log m - 2t)$.

We use the above Theorem 11.9 to prove the main communication complexity of this section.

Theorem 11.10. $R^{k-1}_{1/8}(\bar{g}_{d,k}) = \Omega(dm/k^3 - dk\log m - 2d - 12d^k \lg m - km)$.

Proof. The proof will be using a reduction from $g^d_{1,k}$. Let $(\langle f^1_A, \ldots, f^d_A\rangle, \langle f^1_B, \ldots, f^d_B\rangle)$ be an instance of $g^d_{1,k}$ Define f^*_A and f^*_B by,

$$f^*_A(j) := \{f^i_A(j) : i \in [d]\} \text{ and } f^*_B(j) := \{f^i_B(j) : i \in [d]\} \ .$$

Assume there exists a $(k-1)$-round protocol Π for $\bar{g}_{d,k}$ that fails with probability at most $1/4$ and communicates $o(dm/k^3 - dk\log m - 2d - 12d^k \lg m - km)$ bits in the worst case. We will show how to transform Π into a protocol Π' for $g^d_{1,k}$ that fails with probability at most $1/4$ and communicates $o(dm/k^3 - dk\log m - 2d)$ bits in the worst case. This will be a contradiction by Theorem 11.9 and hence there was no such protocol for $\bar{g}_{d,k}$.

If Π is successful then the player who sends the last message, m_{k-1}, of Π knows

$$\bar{g}_{d,k}(f^*_A, f^*_B) = \langle g_{d,1}(f^*_A, f^*_B), \ldots, g_{d,k}(f^*_A, f^*_B)\rangle \ .$$

Assume this message is sent my Alice. In Π' we append m_{k-1} with $\bar{g}_{d,k}$ and the following set of triples,

$$\{\langle i, j, f^i_A(j)\rangle : i \in [d], j \in \bigcup_{r \in \{0\} \cup [k-1]:\text{even}} g_{d,r}(f^*_A, f^*_B)\} \ .$$

Sending $\bar{g}_{d,k}(f^*_A, f^*_B)$ adds at most an extra km bits of communication. Sending the triples adds at most an extra $6d^k \lg m$ bits of communication since $|g_{d,r}(f^*_A, f^*_B)| \le d^r$.

The final message of Π' is the set of triples,

$$\{\langle i, j, f_B^i(j)\rangle : i \in [d], j \in \bigcup_{r\in[k-1]:\text{odd}} g_{d,r}(f_A^*, f_B^*)\} \ .$$

Again, sending the triples adds at most an extra $6d^k \lg m$ bits of communication. Hence Π' communicates $o(dm/k^3 - dk \log m - 2d)$ bits in the worst case. $\square$

We are now ready to prove Theorem 11.6.

Proof of Theorem 11.6. We do a reduction from d-valued pointer chasing. Let $m = n/(l+1)$ and let $d = m^\gamma$. By Thm 11.10, $R_{1/8}^{l-1}(\bar{g}_{d,l}) = \Omega(n^{1+\gamma})$ since l is constant.

Consider an instance (f_A, f_B) of $\bar{g}_{d,l}$. The graph described by the stream is on the following set of $n = (l+1)m$ nodes,

$$V = \bigcup_{1\le i\le l+1} \{v_1^i, \ldots, v_m^i\} \ .$$

For $i \in [l]$ we define a set of edges $E(i)$ between $\{v_1^i, \ldots, v_m^i\}$ and $\{v_1^{i+1}, \ldots, v_m^{i+1}\}$ in the following way:

$$E(i) = \begin{cases} \{(v_j^i, v_k^{i+1}) : k \in f_A(j)\} & \text{if } i \text{ is odd} \\ \{(v_j^i, v_k^{i+1}) : k \in f_B(j)\} & \text{if } i \text{ is even} \end{cases} \ .$$

Suppose there exists an algorithm $\mathcal{A}$ that computes the first l layers of the breadth first search tree from v_1^1 in p passes using memory M. Let L_r be set of nodes that are exactly distance r from v_1^1. Note that for all $r \in [l]$,

$$g_{d,r} = L_r \cap \{v_1^{r+1}, \ldots, v_m^{r+1}\} \ .$$

Hence by simulating $\mathcal{A}$ on a stream starting with $\bigcup_{i\in[l]:\text{even}} E(i)$ and concluding with $\bigcup_{i\in[l]:\text{odd}} E(i)$ in the natural way we deduce there exists an $2p$ round communication protocol for $g_{d,l}$ that uses only $2pM$ communication. Hence either $2p > l-1$ or $M = \Omega(n^{1+\gamma})$. $\square$

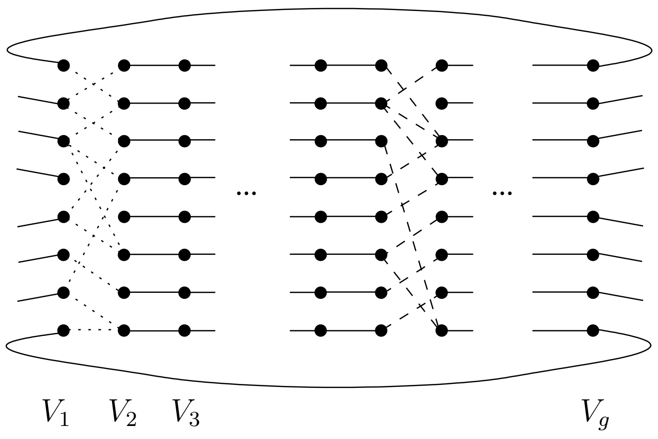


Figure 11.2: Girth Lower-Bound Construction. The edges E_x are dotted, E_y are dashed, and E_m are solid.

11.4 Girth Estimation

In this section we prove a lower bound on the space required by a multi-pass algorithm that tests whether a graph has girth at most g. We shall make use of the following result from [LUW95],

Lemma 11.11 (Lazebnik, Ustimenko, and Woldar [LUW95]). *Let $k \geq 1$ be an odd integer, $t = \left\lfloor \frac{k+2}{4} \right\rfloor$ and q be a prime power. There exists a bipartite, q-regular graph with at most $2q^{k-t+1}$ nodes and girth at least $k+5$.*

Theorem 11.12. *For $g \geq 5$, any p-pass algorithm that tests if the girth of an unweighted graph is at most g, requires $\Omega\left(p^{-1}(n/g)^{1+4/(3g-4)}\right)$ space. If g is odd this can be strengthened to $\Omega\left(p^{-1}(n/g)^{1+4/(3g-7)}\right)$ space.*

Proof. Let q be a prime power and let $k = g-4$ if g is odd and $k = g-3$ if g is even. Let $t = \left\lfloor \frac{k+2}{4} \right\rfloor$. Therefore,

$$k - t + 1 \leq k - \frac{k+2}{4} + 3/4 + 1 \leq \begin{cases} (3g-7)/4 & \text{if } g \text{ is odd} \\ (3g-4)/4 & \text{if } g \text{ is even} \end{cases} .$$

Then Lemma 11.11 implies that there exists a q-regular graph $G' = (L \cup R, E')$ with at most $2n' \leq 2q^{k-t+1}$ nodes and girth at least $g+1$. We denote $L = \{l_1, \ldots, l_{n'}\}$ and $R = \{r_1, \ldots, r_{n'}\}$ and, for each $i \in [n']$, let $D_i = \Gamma(l_i)$.

Let $(x, y) \in \mathbb{F}_2^r \times \mathbb{F}_2^r$ by an instance of Set-Disjointness where $r = n'q$. It will be convenient to write $x = x^1 \ldots x^{n'}$ and $y = y^1 \ldots y^{n'}$ where $x^i, y^j \in \mathbb{F}_2^q$. We will show how to transform a p-pass algorithm $\mathcal{A}$ for testing if the girth of a graph is at most g into a protocol for Set-Disjointness. If $\mathcal{A}$ uses M bits of working memory then the protocol will use $\Omega(pM)$. Hence $M = \Omega(p^{-1}n'q)$.

Alice and Bob construct a graph G based upon G', x, and y as follows. For $i \in [g]$, let $V_i = \{v_1^i, \ldots, v_{n'}^i\}$. For each $i \in [n']$, let $D_i(x) \subset D_i$ be the subset of D_i whose characteristic vector is x^i. $D_i(y)$ is defined similarly. There are three sets of edges on these nodes;

$$\begin{aligned}
E_m &= \bigcup_{j \in [g] \setminus \{1, \lfloor g/2 \rfloor\}} \{(v_i^j, v_i^{j+1}) : i \in [n']\}, \\
E_x &= \{(v_i^1, v_j^2) : j \in D_i(x), i \in [n']\}, \text{ and} \\
E_y &= \{(v_j^{\lfloor g/2 \rfloor}, v_i^{\lfloor g/2 \rfloor + 1}) : j \in D_i(y), i \in [n']\} \ .
\end{aligned}$$

See Fig. 11.2 for a diagram of the construction.

Note that the Girth$(G) = g$ if there exists i such that $D_i(x) \cap D_i(y) \neq \emptyset$, i.e., x and y are not disjoint. However if x and y are disjoint then the shortest cycle is at least length $4 + 2\left\lfloor \frac{g-2}{2} \right\rfloor \geq g + 1$. Hence, determining if the girth is at most g determines if x and y are disjoint. $\square$

Chapter 12

Graph Matching

Chapter Outline: In this chapter we present the technical details behind our results on finding maximum matchings in the data-stream model. We first present a $(1-\epsilon)$-approximation in the unweighted case. We then demonstrate a $(1/2-\epsilon)$-approximation in the weighted case. For background see Chapter 10.

12.1 Unweighted Matchings

In this section we describe a streaming algorithm that, for $\epsilon > 0$, computes a $(1-\epsilon)$-approximation to the maximum cardinality matching (MCM) of the streamed graph. The algorithm will use a $O_\epsilon(1)$ number of passes. We start by giving some basic definitions common to many matching algorithms.

Definition 12.1 (Basic Matching Theory Definitions). *Given a matching M in a graph $G = (V, E)$, we call a vertex* free *if it does not appear as the end point of any edge in M. A length $2i+1$* augmenting path *is a path $u_1 u_2 \ldots u_{2i+2}$ where u_1 and u_{2i+2} are free and $(u_j, u_{j+1}) \in M$ for even j and $(u_j, u_{j+1}) \in E \setminus M$ for odd j.*

Note that if M is a matching and P is an augmenting path then $M \triangle P$ (the symmetric difference of M and P) is a matching of size strictly greater than M. Our algorithm will start by finding a maximal matching and then, by finding short augmenting paths, increase the size of the matching by making local changes. Note that finding a maximal matching can easily achieved in one pass: we select an edge iff we have not already selected an adjacent edge. Finding maximal matchings in this way will be an important sub-routine of our algorithm and we will make repeated use of the fact that the maximum matching has cardinality at most twice that of any maximal matching.

The following lemma establishes that, when there are few short augmenting paths, the size of the matching found can be lower-bound in terms of the size of the maximum cardinality matching $\textsc{Opt}$.

Lemma 12.2. *Let M be a maximal matching and $\textsc{Opt}$ be a matching of maximum cardinality. Consider the connected components of $\textsc{Opt}\triangle M$. Let $\alpha_i|M|$ be the number of connected components with exactly i edges from M and $i+1$ edges from $\textsc{Opt}$. Then,*

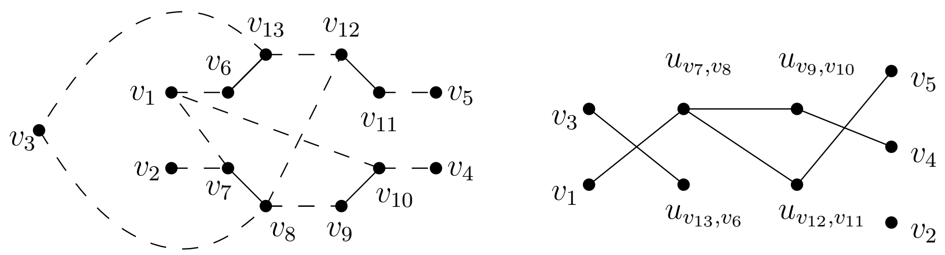
$$\left(\max_{1\le i\le k} \alpha_i \le \frac{1}{2k^2(k+1)}\right) \Rightarrow \left(|M| \ge \frac{\textsc{Opt}}{1+1/k}\right) \ .$$

Proof. First note that $\sum_{i\ge k+1}\alpha_i \le 1/(k+1)$ because $\sum_i i\alpha_i|M| \le |M|$. In each connected component of $\textsc{Opt}\triangle M$ with i edges from M there are either i or $i+1$ edges from $\textsc{Opt}$. Therefore,

$$\frac{\textsc{Opt}}{|M|} \le (1-\sum_i i\alpha_i) + \sum_i (i+1)\alpha_i = 1 + \sum_i \alpha_i \le 1 + k(\max_{1\le i\le k}\alpha_i) + \frac{1}{k+1} \le 1 + \frac{1}{k}.$$

□

So, if there are $\alpha_i|M|$ components in $\textsc{Opt}\triangle M$ with $i+1$ edges from $\textsc{Opt}$ and i edges from M, then there are at least $\alpha_i|M|$ length $2i+1$ augmenting paths for M. Finding an augmenting path allows us to increase the size of M. Hence, if $\max_{1\le i\le k}\alpha_i$ is small we already have a good approximation to $\textsc{Opt}$ whereas, if $\max_{1\le i\le k}\alpha_i$ is large then there exists $i\in[k]$ such that there are many length $2i+1$ augmenting paths.



(a) Current Matching. (b) Layered Graph.

Figure 12.1: A schematic of the procedure for finding length 5 augmenting paths.

Description of the Algorithm: Now we have defined the basic notion of augmenting paths, we are in a position to give an overview of our algorithm. We have just reasoned that, if our matching is not already large, then there exists augmenting paths of some length no greater than $2k + 1$. Our algorithm looks for augmenting paths of each of the k different lengths separately. Consider searching for augmenting paths of length $2i + 1$. See Fig. 12.1 for a schematic of this process when $i = 2$. Fig. 12.1(a) depicts the graph G with heavy solid lines denoting edges in the current matching. To find length $2i + 1$ we randomly "project" the graph (and current matching) into a set of graphs, $\mathcal{L}_i$, which we now define.

Definition 12.3. *Consider a graph whose n nodes are partitioned into $i + 2$ layers $L_{i+1}, \ldots L_0$ and whose edge set is a subset of*

$$\bigcup_{1 \le j \le i+1} \{(u, v) : u \in L_j, v \in L_{j-1}\} \ .$$

*Let $\mathcal{L}_i$ denote graphs of this type and call a path $u_l, \ldots, u_0$ with $u_j \in L_j$ an l-*path.

The random projection is performed as follows. The image of a free node in G is a node in either L_{i+1} or L_0. The image of a matched edge $e = (v, v')$ is a node, either $u_{(v,v')}$ or $u_{(v',v)}$, in one of $L_i, \ldots, L_1$ chosen at random. The edges in the projected graph G' are those in G that are "consistent" with the mapping of the free nodes and the matched edges, i.e., there is an edge between a node $u_{(v_1,v_2)} \in L_j$ and $u_{(v_3,v_4)} \in L_{j-1}$ if there is an edge $(v_2, v_3) \in E$. Now note that an $(i + 1)$-path in G' corresponds to a $2i + 1$ augmenting path in G. Unfortunately the converse is not true, there may be $2i + 1$ augmenting paths in G that do not correspond to $(i+1)$-paths in G' because we only consider consistent edges. However, we will show later that a constant fraction of augmenting paths exist (with high probability) as $(i + 1)$-paths in G'. Fig. 12.1(b) depicts G', the layered graph formed by randomly projecting G into $\mathcal{L}_i$.

We now try to find a nearly maximal set of node disjoint $(i + 1)$-paths in a graph G'. See algorithm *Find-Layer-Paths* in Fig. 12.2. The algorithm finds node disjoint $(i+1)$-paths by doing something akin to a depth first search. Finding a maximal set of node disjoint $(i+1)$-paths can easily be achieved in the RAM model by actually doing a DFS, deleting nodes of found $(i+1)$-paths and deleting edges when back-tracking. Unfortunately this would necessitate too many passes in the streaming model as each back-track potentially requires another pass of the data. Our algorithm in essence blends a DFS and BFS in such a way that we can substantially reduce the number of backtracks required. This will come at the price of possibly stopping prematurely, i.e., when there may still exist some $(i + 1)$-paths that we have not located.

The algorithm first finds a maximal matching between L_{i+1} and L_i. Let S' be the subset of nodes L_i involved in this first matching. It then finds a maximal matching between S' and L_{i-1}. We continue in this fashion, finding a matching between $S'' = \{u \in L_{i-1} : u \text{ matched to some } u' \in L_i\}$ and L_{i-2}. One can think of the algorithm as growing node disjoint paths from left to right. If the size of the maximal matching between some subset S of a level L_j and L_{j-1} falls below a threshold we declare all vertices in S to be dead-ends and conceptually remove them from the graph (in the sense that we never again use these nodes while try to find $(i + 1)$-paths.) At this point we start back-tracking. It is the use of this threshold that ensures a limit on the amount of back-tracking performed by the algorithm. However, because of the threshold, it is possible that a vertex may be falsely declared to be a dead-end, i.e., there may still be a node disjoint path that uses this vertex. With this in mind we want the threshold to be low such that this does not happen often and we can hope to find all but a few of a maximal set of node disjoint $(i+1)$-paths. When we grow some node disjoint paths all the way to L_0, we remove these paths and recurse on the remaining graph. For each node v, the algorithm maintains a tag indicating if it is a "Dead End" or, if we have found a $i+1$ path involving v, the next node in the path.

It is worth reiterating that in each pass of the stream we simply find a maximal matching between some set of nodes. The above algorithm simply determines within which set of nodes we find a maximal matching.

Our algorithm is presented in detail in Fig. 12.2. Here we use the notation $s \in_R S$ to denote choosing an element s uniformly at random from a set S. Also, for a matching M, let $\Gamma_M(u) = v$ if $(u,v) \in M$ and $\emptyset$ otherwise.

Correctness and Running Time Analysis: We first argue that the use of thresholds in *Find-Layer-Paths* ensures that we find all but a small number of a maximal set of $(i+1)$-paths.

Lemma 12.4 (Running Time and Correctness)**.** *Given $G' \in \mathcal{L}_i$, Find-Layer-Paths algorithm finds at least $(\gamma-\delta)|M|$ of the $(i+1)$-paths where $\gamma|M|$ is the size of some maximal set of $(i+1)$-paths. Furthermore, the algorithm takes a constant number of passes.*

Proof. First note that *Find-Layer-Paths*$(\cdot,\cdot,\cdot,l)$ is called with argument $\delta^{2^{i+1-l}}$. During the running of *Find-Layer-Paths*$(\cdot,\cdot,\cdot,l)$ when we run line 15, the number of $(i+1)$-paths we rule out is at most $2\delta^{2^{i+1-l}}|L_{l-1}|$ where the factor 2 comes from the fact that a maximal matching is at least half the size of a maximum matching. Let E_l be the number of times *Find-Layer-Paths*$(\cdot,\cdot,\cdot,l)$ is called: $E_{i+1}=1$, $E_l \leq E_{l+1}/\delta^{2^{i+1-l}}$ and therefore $E_l \leq \delta^{-\sum_{0\leq j\leq i-l} 2^j} = \delta^{-2^{i-l+1}+1}$. Hence, we remove at most $2E_l\delta^{2^{i+1-l}}|L_l| \leq 2\delta|L_l|$. Note that when nodes are labeled as dead-ends in a call to *Find-Layer-Paths*$(\cdot,\cdot,\cdot,1)$, they really are dead-ends and declaring them

```
Algorithm Find-Matching$(G,\epsilon)$
1.  Find a maximal matching $M$
2.  $k \leftarrow \lfloor \epsilon^{-1}+1 \rfloor$ and $r \leftarrow 4k^2(8k+10)(k-1)(2k)^k$
3.  for $j=1$ to $r$:
4.          for $i=1$ to $k$: $M_i \leftarrow$ Find-Aug-Paths$(G,M,i)$
5.          $M \leftarrow \text{argmax}_{M_i}|M_i|$
6.  return $M$

Algorithm Find-Aug-Paths$(G,M,i)$
(* Finds length $2i+1$ augmenting paths for a matching $M$ in $G$ *)
1.  $G' \leftarrow$Create-Layer-Graph$(G,M,i)$
2.  $\mathcal{P}$ =Find-Layer-Paths$(G', L_{i+1}, \frac{1}{r(2k+2)}, i+1)$
3.  return $M \triangle \mathcal{P}$

Algorithm Create-Layer-Graph$(G,M,i)$
(* Randomly constructs $G' \in \mathcal{L}_i$ from a graph $G$ and matching $M$ *)
1.  if $v$ is a free vertex then $l(v) \in_R \{0, i+1\}$
2.  if $e=(u,v) \in M$ then $(l(e),l(v),l(u)) \in_R \{(j,ja,jb),(j,jb,ja): j \in [i]\}$
3.  $E_0 \leftarrow (l^{-1}(1b), l^{-1}(0)) \cap E$ and $E_i \leftarrow (l^{-1}(i+1), l^{-1}(ia)) \cap E$
4.  for $j=0$ to $i+1$: $L_j \leftarrow l^{-1}(j)$
5.  for $j=1$ to $i-1$: $E_j \leftarrow \{(u,v) \in E : (l(u),l(v))=((j+1)b, ja)\}$
6.  return $G'=(L_{i+1} \cup L_i \cup \ldots \cup L_0, E_i \cup E_{i-1} \cup \ldots \cup E_0)$

Algorithm Find-Layer-Paths$(G',S,\delta,j)$
(* Finds many $j$-paths from $S \subset L_j$ *)
1.  Find maximal matching $M'$ between $S$ and untagged vertices in $L_{j-1}$
2.  $S' \leftarrow \{v \in L_{j-1} : \exists u, (u,v) \in M'\}$
3.  if $j=1$
4.     then if $u \in \Gamma_{M'}(L_{j-1})$ then $t(u) \leftarrow \Gamma_{M'}(u)$, $t(\Gamma_{M'}(u)) \leftarrow \Gamma_{M'}(u)$
5.          if $u \in S \setminus \Gamma_{M'}(L_{j-1})$ then $t(u) \leftarrow$ "Dead End "
6.          return
7.  repeat
8.      Find-Layer-Paths$(G',S',\delta^2,j-1)$
9.      for $v \in S'$ such that $t(v) \neq$"Dead End": $t(\Gamma_{M'}(v)) \leftarrow v$
10.     Find maximal matching $M'$ between untagged vertices in $S$ and $L_{j-1}$.
11.     $S' \leftarrow \{v \in L_{j-1} : \exists u, (u,v) \in M'\}$
12. until $|S'| \leq \delta|L_{j-1}|$
13. for $v \in S$ untagged: $t(b) \leftarrow$ "Dead End".
14. return
```

Figure 12.2: An Algorithm for Finding Large Cardinality Matchings.

such rules out no remaining $(i+1)$-paths. Hence, the total number of paths not found is at most $2\delta \sum_{1\leq j\leq i} |L_j| \leq 2\delta|M|$. The number of invocations of the recursive algorithm is,

$$\sum_{1\leq l\leq i+1} E_l \leq \sum_{1\leq l\leq i+1} \delta^{-2^{i+1-l}+1} \leq \delta^{-2^{i+1}} \ .$$

i.e., $O(1)$ and each invocation requires one pass of the data stream to find a maximal matching. □

When looking for length $(2i+1)$ augmenting paths for a matching M in graph G, we randomly create a layered graph $G' \in \mathcal{L}_{i+1}$ using *Create-Layer-Graph* such that $(i+1)$-paths in G' correspond to length $(2i+1)$ augmenting paths. We now need to argue that a) many of the $(2i+1)$ augmenting paths in G exist in G' as $(i+1)$-paths and b) that finding a maximal, rather that a maximum, set of $(i+1)$-paths in G' is sufficient for our purposes.

Theorem 12.5. *If G has $\alpha_i M$ length $2i+1$ augmenting paths, then the number of length $(i+1)$-paths found in G' is at least $(b_i\beta_i - \delta)|M|$ where $b_i = 1/(2i+2)$ and β_i is a random variables distributed as* $\mathrm{Bin}(\alpha_i|M|, 1/(2(2i)^i))$.

Proof. Consider a length $(2i+1)$ augmenting path $P = u_0 u_1 \ldots u_{2i+1}$ in G. The probability that P appears as an $(i+1)$-path in G' is at least,

$$2\Pr[l(u_0)=0]\Pr[l(u_{2i+1})=i+1]\prod_{j\in[i]}\Pr[l(u_{2j})=ja, l(u_{2j-1})=jb] = \frac{1}{2(2i)^i} \ .$$

Given that the probability of each augmenting path existing as a $(i+1)$-path in G' is independent, the number of length $(i+1)$-paths in G' is distributed as $\mathrm{Bin}(\alpha_i|M|, 1/(2(2i)^i))$. The size of a maximal set of node disjoint $(i+1)$-paths is at least a $1/(2i+2)$ fraction of the maximum size node-disjoint set $(i+1)$-paths. Combining this with Lemma 12.4 gives the result. □

Finally, we argue that we only need to try to augment our initial matching a constant number of times.

Theorem 12.6 (Correctness)**.** *With probability $1-f$ by running $O(\log f^{-1})$ copies of the algorithm Find-Matching in parallel we find a $1-\epsilon$ approximation to the matching of maximum cardinality.*

Proof. We show that the probability that a given run of *Find-Matching* does not find a $(1-\epsilon)$-approximation is bounded above by e^{-1}.

Define a *phase* of the algorithm to be one iteration of the loop started at line 4 of *Find-Matching*. At the start of phase p of the algorithm, let M_p be the current matching. In the course of phase p of the algorithm we augment M_p by at least $|M_p|(\max_{1\leq i\leq k}(b_i\beta_{i,p})-\delta)$ edges where $\beta_{i,p}|M_p| \sim \mathrm{Bin}(\alpha_{i,p}|M_p|, 1/(2(2i)^i))$. Let A_p be the value of $|M_p|\max_i(b_i\beta_{i,p})$ in the p-th phase of the algorithm. Assume that for each of the r phases of the algorithm $\max \alpha_{i,p} \geq \alpha^* := 1/(2k^2(k-1))$. (By Lemma 12.2, if this is ever not the case, we already have a sufficiently sized matching.) Therefore, A_p dominates $b_k \,\mathrm{Bin}(\alpha^*|M_1|, 1/(2(2k)^k))$. Let $(X_p)_{1\leq p\leq r}$ be independent random variables, each distributed as $b_k\,\mathrm{Bin}(\alpha^*|M_1|, 1/(2(2k)^k))$. Therefore,

$$\begin{aligned}
\Pr\left[\prod_{1\leq p\leq r}(1+\max\{0,\max_{1\leq i\leq k}(b_i\beta_{i,p})-\delta\})\geq 2\right] &\geq \Pr\left[\sum_{1\leq p\leq r}\max_{1\leq i\leq k} b_i\beta_{i,p}\geq 2+r\delta\right]\\
&\geq \Pr\left[\sum_{1\leq p\leq r} X_p \geq |M_1|\frac{2+r\delta}{b_k}\right]\\
&= \Pr\left[Z\geq |M_1|(4k+5)\right] \ ,
\end{aligned}$$

for $\delta = b_k/r$ where $Z=\mathrm{Bin}(\alpha^*|M_1|r, 1/(2(2k)^k))$. Finally, by an application of the Chernoff bound,

$$\Pr\left[Z\geq |M_1|(4k+5)\right] = 1-\Pr\left[Z<E\left[Z\right]/2\right] > 1-e^{-2(8k+10)|M_1|}\geq 1-e^{-1} \ ,$$

for $r=2(2k)^k(8k+10)/\alpha^*$. Of course, since M_1 is already at least half the size of the maximal matching this implies that with high probability, at some point during the r phases our assumption that $\max\alpha_{i,p}\geq\alpha^*$, became invalid and at this point we had a sufficiently large matching. □

12.2 Weighted Matching

We now turn our attention to finding maximum weighted matchings. Here each edge $e \in E$ of our graph G has a weight $w(e) > 0$. For a set of edges S let $w(S) = \sum_{e \in S} w(e)$. We seek to maximize $w(S)$ subject to the constraint that S contains no two adjacent edges.

Consider the algorithms in Fig. 12.3. The algorithm greedily collects edges as they stream past and maintains a matching, M, at all points. On seeing an edge e, if $w(e) > (1+\gamma)w(\{e' | e' \in M, e' \text{ and } e \text{ share an end point}\})$ then the algorithm removes any edges in M sharing an end point with e and adds e to M. The algorithm *Find-Weighted-Matching-Multipass* generalizes this to a multi-pass algorithm that repeats the one pass algorithm until the improvement achieved falls below some threshold. We start by introducing some notation.

Definition 12.7. *In a given pass of the graph stream, we say that an edge e is* born *if $e \in M$ at some point during the execution of the algorithm. We say that an edge is* bumped *if it was born but subsequently removed from M by a newer edge. This new edge is a* bumper. *We say an edge is a* survivor *if it is born and never bumped. For each survivor e, let the* Replacement Tree *be the set of edges $T(e) = C_1 \cup C_2 \cup \ldots$, where $C_0 = \{e\}$, $C_1 = \{$the edges bumped by $e\}$, and $C_i = \cup_{e' \in C_{i-1}} \{$the edges bumped by $e'\}$.*

Lemma 12.8. *For a given pass let the set of survivors be S. The weight of the matching found at the end of that pass is therefore $w(S)$.*

1. $w(T(S)) \leq w(S)/\gamma$

2. $w(\text{OPT}) \leq (1+\gamma)\left(w(T(S)) + 2w(S)\right)$

Proof. We prove each statement separately.

1. For each bumper e, $w(e)$ is at least $(1+\gamma)$ the total cost of bumped edges, and an edge has at most one bumper. Hence, for all i, $w(C_i) \geq (1+\gamma)w(C_{i+1})$ and

therefore $(1+\gamma)w(T(e)) = \sum_{i\geq 1}(1+\gamma)w(C_i) \leq \sum_{i\geq 0} w(C_i) = w(T(e))+w(e)$. The first point follows by summing over the survivor edges.

2. Consider the optimal solution that includes edges $\textsc{Opt} = \{o_1, o_2, \ldots\}$. We are going to charge the costs of edges in $\textsc{Opt}$ to the survivors and their replacement trees, $\cup_{e\in S}T(e) \cup \{e\}$. We hold an edge e in this set *accountable to* $o \in \textsc{Opt}$ if either $e = o$ or else o was not born because e was in M when o arrived. Note that, in the second case, it is possible for two edges to be accountable to o. If only one edge is accountable for o then we charge $w(o)$ to e. If two edges e_1 and e_2 are accountable for o, then we charge $\frac{w(o)w(e_1)}{w(e_1)+w(e_2)}$ to e_1 and $\frac{w(o)w(e_2)}{w(e_1)+w(e_2)}$ to e_2. In either case, the amount charged by o to any edge e is at most $(1+\gamma)w(e)$.

   We now redistribute these charges as follows: (for distinct u_1, u_2, u_3) if $e = (u_1, v)$ gets charged by $o = (u_2, v)$, and e subsequently gets bumped by $e' = (u_3, v)$, we transfer the charge from e to e'. Note that we maintain the property that the amount charged by o to any edge e is at most $(1+\gamma)w(e)$ because $w(e') \geq w(e)$. What this redistribution of charges achieves is that now every edge in a replacement tree is only charged by one edge in $\textsc{Opt}$. Survivors can, however, be charged by two edges in $\textsc{Opt}$. We charge $w(\textsc{Opt})$ to the survivors and their replacement trees, and hence

$$w(\textsc{Opt}) \leq \sum_{e\in S}(1+\gamma)w(T(e)) + 2(1+\gamma)w(e) \ .$$

□

Hence, in one pass we achieve an $1/(1/\gamma + 3 + 2\gamma)$ approximation ratio since

$$\textsc{Opt} \leq (1+\gamma)(w(T(S)) + 2w(S)) \leq (3 + 1/\gamma + 2\gamma)w(S) \ .$$

The maximum of this function is achieved for $\gamma = 1/\sqrt{2}$ giving approximation ratio $1/(3+2\sqrt{2})$. This represents only a slight improvement over the 1/6 ratio attained previously. However, a much more significant improvement is realized in the multi-pass algorithm *Find-Weighted-Matching-Multipass*.

```
Algorithm Find-Weighted-Matching($G, \gamma$)
(* Finds Large Weighted Matchings in One Pass *)
1.  $M \leftarrow \emptyset$
2.  for each edge $e \in G$
3.      do $M_e \leftarrow \{e' | e' \in M,\ e'$ and $e$ share an end point$\}$
4.          if $w(e) > (1+\gamma)w(M_e)$ then $M \leftarrow M \cup \{e\} \setminus M_e$ return M

Algorithm Find-Weighted-Matching-Multipass($G, \epsilon$)
(* Finds Large Weighted Matchings *)
1.  $\gamma \leftarrow 2\epsilon/3$
2.  $\kappa \leftarrow \gamma^3/((1+\gamma)^2 - \gamma^3)$
3.  Find a $1/(3+2\sqrt{2})$ weighted matching, $M$
4.  repeat
5.      $S \leftarrow w(M)$
6.      for each edge $e \in G$
7.          do $M_e \leftarrow \{e' | e' \in M,\ e'$ and $e$ share an end point$\}$
8.              if $w(e) > (1+\gamma)w(M_e)$ then $M \leftarrow M \cup \{e\} \setminus M_e$
9.  until $w(M)/S \leq 1+\kappa$
10. return M
```

Figure 12.3: An Algorithm for Finding Large Weighted Matchings

Theorem 12.9. *The algorithm Find-Weighted-Matching-Multipass finds a* $(1/2-\epsilon)$*-approximation to the maximum weighted matching in* $O(\epsilon^{-1})$ *passes.*

Proof. First we prove that the number of passes is as claimed. We increase the weight of our solution by a factor $1+\kappa$ each time we do a pass and we start with a $1/(3+2\sqrt{2})$ approximation. Hence, if we take, $\log_{1+\kappa}(3/2+\sqrt{2})$ passes we have already found a maximum weighted matching. Substituting in $\kappa = \gamma^3/((1+\gamma)^2-\gamma^3)$ establishes the bound on the number of passes.

Let M_i be the matching constructed after the i-th pass. Let $B_i = M_i \cap M_{i-1}$. Now, $(1+\gamma)(w(M_{i-1}) - w(B_i)) \leq w(M_i) - w(B_i)$ and so,

$$\frac{w(M_i)}{w(M_{i-1})} = \frac{w(M_i)}{w(M_{i-1}) - w(B_i) + w(B_i)} \geq \frac{(1+\gamma)w(M_i)}{w(M_i) + \gamma w(B_i)} \ .$$

If $w(M_i)/w(M_{i-1}) < (1+\kappa)$, then we deduce that $w(B_i) \geq w(M_i)(\gamma-\kappa)/(\gamma+\gamma\kappa)$.

Appealing to Lemma 12.8, this means that, for all i,

$$OPT \leq (1/\gamma + 3 + 2\gamma)(w(M_i) - w(B_i)) + 2(1+\gamma)w(B_i) \ ,$$

since edges in B_i have empty replacement trees. So if $w(B_i) \geq w(M_i)(\gamma - \kappa)/(\gamma + \gamma\kappa)$ we deduce that,

$$\begin{aligned} OPT &\leq (1/\gamma + 3 + 2\gamma)(w(M_i) - w(B_i)) + 2(1+\gamma)w(B_i) \\ &\leq \left(1/\gamma + 3 + 2\gamma - (1/\gamma + 1)\frac{\gamma - \kappa}{\gamma + \gamma\kappa}\right) w(M_i) \\ &\leq (2 + 3\gamma)w(M_i) \ . \end{aligned}$$

Since $\gamma = 2\epsilon/3$ the claimed approximation ratio follows. $\square$

Bibliography

[ABB+03] Arvind Arasu, Brian Babcock, Shivnath Babu, Mayur Datar, Keith Ito, Rajeev Motwani, Itaru Nishizawa, Utkarsh Srivastava, Dilys Thomas, Rohit Varma, and Jennifer Widom. STREAM: The Stanford stream data manager. *IEEE Data Eng. Bull.*, 26(1):19–26, 2003.

[ABCP98] Baruch Awerbuch, Bonnie Berger, Lenore Cowen, and David Peleg. Near-linear time construction of sparse neighborhood covers. *SIAM J. Comput.*, 28(1):263–277, 1998.

[ABW02] James Abello, Adam L. Buchsbaum, and Jeffery Westbrook. A functional approach to external graph algorithms. *Algorithmica*, 32(3):437–458, 2002.

[ACÇ+03] Daniel J. Abadi, Donald Carney, Ugur Çetintemel, Mitch Cherniack, Christian Convey, Sangdon Lee, Michael Stonebraker, Nesime Tatbul, and Stanley B. Zdonik. Aurora: a new model and architecture for data stream management. *VLDB J.*, 12(2):120–139, 2003.

[AdBHZ07] Mohammad Ali Abam, Mark de Berg, Peter Hachenberger, and Alireza Zarei. Streaming algorithms for line simplification. In *Symposium on Computational Geometry*, pages 175–183, 2007.

[ADRR04] Gagan Aggarwal, Mayur Datar, Sridhar Rajagopalan, and Matthias Ruhl. On the streaming model augmented with a sorting primitive.

IEEE Symposium on Foundations of Computer Science, pages 540–549, 2004.

[AHPV04] Pankaj K. Agarwal, Sariel Har-Peled, and Kasturi R. Varadarajan. Approximating extent measures of points. *J. ACM*, 51(4):606–635, 2004.

[AJB99] Reka Albert, Hawoong Jeong, and Albert-Laszlo Barabasi. The diameter of the world wide web. *Nature*, 401:130, 1999.

[AJKS02] Miklós Ajtai, T. S. Jayram, Ravi Kumar, and D. Sivakumar. Approximate counting of inversions in a data stream. In *ACM Symposium on Theory of Computing*, pages 370–379, 2002.

[AM04] Arvind Arasu and Gurmeet Singh Manku. Approximate counts and quantiles over sliding windows. In *ACM Symposium on Principles of Database Systems*, pages 286–296, 2004.

[Ama85] Shun-Ichi Amari. *Differential-geometrical methods in statistics.* Springer-Verlag, New York, 1985.

[AMP$^+$06] Deepak Agarwal, Andrew McGregor, Jeff M. Phillips, Suresh Venkatasubramanian, and Zhengyuan Zhu. Spatial scan statistics: approximations and performance study. In *ACM International Conference on Knowledge Discovery and Data Mining*, pages 24–33, 2006.

[AMS99] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58(1):137–147, 1999.

[AN00] Shun-Ichi Amari and Hiroshi Nagaoka. *Methods of Information Geometry.* Oxford University and AMS Translations of Mathematical Monographs, 2000.

[AS66] S. M. Ali and S. D. Silvey. A general class of coefficients of divergence of one distribution from another. *J. of Royal Statistical Society, Series B*, 28:131–142, 1966.

[AY07] Pankaj K. Agarwal and Hai Yu. A space-optimal data-stream algorithm for coresets in the plane. In *Symposium on Computational Geometry*, pages 1–10, 2007.

[Bas06] Surender Baswana. Faster streaming algorithms for graph spanners, 2006.

[BBD$^+$02] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and issues in data stream systems. *ACM Symposium on Principles of Database Systems*, pages 1–16, 2002.

[BCFM00] Andrei Z. Broder, Moses Charikar, Alan M. Frieze, and Michael Mitzenmacher. Min-wise independent permutations. *J. Comput. Syst. Sci.*, 60(3):630–659, 2000.

[BDKR05] Tuğkan Batu, Sanjoy Dasgupta, Ravi Kumar, and Ronitt Rubinfeld. The complexity of approximating the entropy. *SIAM J. Comput.*, 35(1):132–150, 2005.

[BDM02] Brian Babcock, Mayur Datar, and Rajeev Motwani. Sampling from a moving window over streaming data. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 633–634, 2002.

[BEY98] Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, Cambridge, 1998.

[BFL$^+$06] Luciana S. Buriol, Gereon Frahling, Stefano Leonardi, Alberto Marchetti-Spaccamela, and Christian Sohler. Counting triangles in data streams. In *ACM Symposium on Principles of Database Systems*, pages 253–262, 2006.

[BFR$^+$00] Tuğkan Batu, Lance Fortnow, Ronitt Rubinfeld, Warren D. Smith, and Patrick White. Testing that distributions are close. In *IEEE Symposium on Foundations of Computer Science*, pages 259–269, 2000.

[BG06] Lakshminath Bhuvanagiri and Sumit Ganguly. Estimating entropy over data streams. In *ESA*, pages 148–159, 2006.

[BGKS06] Lakshminath Bhuvanagiri, Sumit Ganguly, Deepanjan Kesh, and Chandan Saha. Simpler algorithm for estimating frequency moments of data streams. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 708–713, 2006.

[BGW03] Adam L. Buchsbaum, Raffaele Giancarlo, and Jeffery Westbrook. On finding common neighborhoods in massive graphs. *Theor. Comput. Sci.*, 1-3(299):707–718, 2003.

[BKMT03] Prosenjit Bose, Evangelos Kranakis, Pat Morin, and Yihui Tang. Bounds for frequency estimation of packet streams. In *SIROCCO*, pages 33–42, 2003.

[Blu98] Avrim Blum. On-line algorithms in machine learning. In *Developments from a June 1996 seminar on Online algorithms*, pages 306–325, London, UK, 1998. Springer-Verlag.

[Bol85] Bela Bollobás. *Random Graphs.* Academic Press, London, 1985.

[Bre67] Lev M. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *U.S.S.R. Computational Mathematics and Mathematical Physics*, 7(1):200–217, 1967.

[Bre99] Leo Breiman. Prediction games and arcing algorithms. *Neural Computation*, 11(7):1493–1517, 1999.

[BS03] Surender Baswana and Sandeep Sen. A simple linear time algorithm for computing a $(2k-1)-$spanner of $O(n^{1+1/k})$ size in weighted graphs. In *International Colloquium on Automata, Languages and Programming*, pages 384–296, 2003.

[BY02] Ziv Bar-Yossef. *The Complexity of Massive Data Set Computations.* PhD thesis, University of California at Berkeley, 2002.

[BYJK$^+$02] Ziv Bar-Yossef, T.S. Jayram, Ravi Kumar, D. Sivakumar, and Luca Trevisan. Counting distinct elements in a data stream. In *Proc. 6th International Workshop on Randomization and Approximation Techniques in Computer Science*, pages 1–10, 2002.

[BYJKS02] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. In *IEEE Symposium on Foundations of Computer Science*, pages 209–218, 2002.

[BYKS01] Ziv Bar-Yossef, Ravi Kumar, and D. Sivakumar. Sampling algorithms: lower bounds and applications. *ACM Symposium on Theory of Computing*, pages 266–275, 2001.

[BYKS02] Ziv Bar-Yossef, Ravi Kumar, and D. Sivakumar. Reductions in streaming algorithms, with an application to counting triangles in graphs. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 623–632, 2002.

[CC05] Timothy M. Chan and Eric Y. Chen. Multi-pass geometric algorithms. In *Symposium on Computational Geometry*, pages 180–189, 2005.

[CCD$^+$03] Sirish Chandrasekaran, Owen Cooper, Amol Deshpande, Michael J. Franklin, Joseph M. Hellerstein, Wei Hong, Sailesh Krishnamurthy, Samuel Madden, Vijayshankar Raman, Frederick Reiss, and Mehul A. Shah. TelegraphCQ: Continuous dataflow processing for an uncertain world. In *CIDR*, 2003.

[CCFC02] Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. In *International Colloquium on Automata, Languages and Programming*, pages 693–703, 2002.

[CCFM04] Moses Charikar, Chandra Chekuri, Tomás Feder, and Rajeev Motwani. Incremental clustering and dynamic information retrieval. *SIAM J. Comput.*, 33(6):1417–1440, 2004.

[CCM07] Amit Chakrabarti, Graham Cormode, and Andrew McGregor. A near-optimal algorithm for computing the entropy of a stream. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 328–335, 2007.

[CDIM03] Graham Cormode, Mayur Datar, Piotr Indyk, and S. Muthukrishnan. Comparing data streams using hamming norms (how to zero in). *IEEE Trans. Knowl. Data Eng.*, 15(3):529–540, 2003.

[CDM06] Amit Chakrabarti, Khanh Do Ba, and S. Muthukrishnan. Estimating entropy and entropy norm on data streams. In *Symposium on Theoretical Aspects of Computer Science*, pages 196–205, 2006.

[CDTW00] Jianjun Chen, David J. DeWitt, Feng Tian, and Yuan Wang. Niagaracq: A scalable continuous query system for internet databases. In Weidong Chen, Jeffrey F. Naughton, and Philip A. Bernstein, editors, *ACM International Conference on Management of Data*, pages 379–390. ACM, 2000.

[CG07a] Graham Cormode and Sumit Ganguly. On estimating frequency moments of data streams. In *International Workshop on Randomization and Approximation Techniques in Computer Science*, 2007.

[CG07b] Graham Cormode and Minos Garofalakis. Sketching probabilistic data streams. In *ACM International Conference on Management of Data*, 2007.

[CGL$^+$05] A. Robert Calderbank, Anna C. Gilbert, Kirill Levchenko, S. Muthukrishnan, and Martin Strauss. Improved range-summable random variable construction algorithms. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 840–849, 2005.

[Cha06] Timothy M. Chan. Faster core-set constructions and data-stream algorithms in fixed dimensions. *Comput. Geom.*, 35(1-2):20–35, 2006.

[CJSS03] Charles D. Cranor, Theodore Johnson, Oliver Spatscheck, and Vladislav Shkapenyuk. Gigascope: A stream database for network applications. In *ACM International Conference on Management of Data*, pages 647–651, 2003.

[CK06] Kevin L. Chang and Ravi Kannan. The space complexity of pass-efficient algorithms for clustering. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 1157–1166, 2006.

[CKM07] Matthew Chu, Sampath Kannan, and Andrew McGregor. Checking and spot-checking of heaps. In *International Colloquium on Automata, Languages and Programming*, 2007.

[CKMS06] Graham Cormode, Flip Korn, S. Muthukrishnan, and Divesh Srivastava. Space- and time-efficient deterministic algorithms for biased quantiles over data streams. In *ACM Symposium on Principles of Database Systems*, pages 263–272, 2006.

[CKS03] Amit Chakrabarti, Subhash Khot, and Xiaodong Sun. Near-optimal lower bounds on the multi-party communication complexity of set disjointness. In *IEEE Conference on Computational Complexity*, pages 107–117, 2003.

[CLRS01] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms.* MIT Press and McGraw-Hill, New York, NY, USA, 2001.

[CM05a] Graham Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *J. Algorithms*, 55(1):58–75, 2005.

[CM05b] Graham Cormode and S. Muthukrishnan. Space efficient mining of multigraph streams. In *ACM Symposium on Principles of Database Systems*, pages 271–282, 2005.

[CMS01] Graham Cormode, S. Muthukrishnan, and Süleyman Cenk Sahinalp. Permutation editing and matching via embeddings. In *International Colloquium on Automata, Languages and Programming*, pages 481–492, 2001.

[COP03] Moses Charikar, Liadan O'Callaghan, and Rina Panigrahy. Better streaming algorithms for clustering problems. In *ACM Symposium on Theory of Computing*, pages 30–39, 2003.

[CS06] Timothy M. Chan and Bashir S. Sadjad. Geometric optimization problems over sliding windows. *Int. J. Comput. Geometry Appl.*, 16(2-3):145–158, 2006.

[Csi91] Imre Csiszár. Why least squares and maximum entropy? an axiomatic approach to inference for linear inverse problems. *Ann. Statist.*, pages 2032–2056, 1991.

[CSS02] Michael Collins, Robert E. Schapire, and Yoram Singer. Logistic regression, adaboost and bregman distances. *Machine Learning*, 48(1-3):253–285, 2002.

[CT91] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory.* Wiley Series in Telecommunications. John Wiley & Sons, New York, NY, USA, 1991.

[DFK$^+$04] Petros Drineas, Alan M. Frieze, Ravi Kannan, Santosh Vempala, and V. Vinay. Clustering large graphs via the singular value decomposition. *Machine Learning*, 56(1-3):9–33, 2004.

[DFR06] Camil Demetrescu, Irene Finocchi, and Andrea Ribichini. Trading off space for passes in graph streaming problems. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 714–723, 2006.

[DGIM02] Mayur Datar, Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Maintaining stream statistics over sliding windows. *SIAM J. Comput.*, 31(6):1794–1813, 2002.

[DH03] Doratha E. Drake and Stefan Hougardy. Improved linear time approximation algorithms for weighted matchings. In *RANDOM-APPROX*, pages 14–23, 2003.

[DKM04a] Petros Drineas, Ravi Kannan, and Michael Mahoney. Fast monte carlo algorithms for matrices I. *SIAM Journal on Computing (To Appear)*, 2004.

[DKM04b] Petros Drineas, Ravi Kannan, and Michael Mahoney. Fast monte carlo algorithms for matrices II. *SIAM Journal on Computing (To Appear)*, 2004.

[DKM04c] Petros Drineas, Ravi Kannan, and Michael Mahoney. Fast monte carlo algorithms for matrices III. *SIAM Journal on Computing (To Appear)*, 2004.

[DLOM02] Erik D. Demaine, Alejandro López-Ortiz, and J. Ian Munro. Frequency estimation of internet packet streams with limited space. In *European Symposium on Algorithms*, pages 348–360, 2002.

[DRVW06] Amit Deshpande, Luis Rademacher, Santosh Vempala, and Grant Wang. Matrix approximation and projective clustering via volume sampling. *ACM-SIAM Symposium on Discrete Algorithms*, pages 1117–1126, 2006.

[Edm65] Jack Edmonds. Maximum matching and a polyhedron with 0,1-vertices. *J. Res. Nat. Bur. Standards*, 69(B):125–130, 1965.

[Elk01] Michael L. Elkin. Computing almost shortest paths. In *Proc. 20th ACM Symposium on Principles of Distributed Computing*, pages 53–62, 2001.

[Elk07] Michael Elkin. A near-optimal fully dynamic distributed algorithm for maintaining sparse spanners. In *International Colloquium on Automata, Languages and Programming*, 2007.

[EZ06] Michael Elkin and Jian Zhang. Efficient algorithms for constructing $(1+\epsilon, \beta)$-spanners in the distributed and streaming models. *Distributed Computing*, 18(5):375–385, 2006.

[FHT00] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28:337–407, 2000.

[FIS05] Gereon Frahling, Piotr Indyk, and Christian Sohler. Sampling in dynamic data streams and applications. In *Symposium on Computational Geometry*, pages 142–149, 2005.

[FKM$^+$05a] Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. Graph distances in the streaming model: the value of space. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 745–754, 2005.

[FKM$^+$05b] Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theoretical Computer Science*, 348(2-3):207–216, 2005.

[FKSV02a] Joan Feigenbaum, Sampath Kannan, Martin Strauss, and Mahesh Viswanathan. An approximate L^1 difference algorithm for massive data streams. *SIAM Journal on Computing*, 32(1):131–151, 2002.

[FKSV02b] Joan Feigenbaum, Sampath Kannan, Martin Strauss, and Mahesh Viswanathan. Testing and spot-checking of data streams. *Algorithmica*, 34(1):67–80, 2002.

[FKV04] Alan M. Frieze, Ravi Kannan, and Santosh Vempala. Fast monte-carlo algorithms for finding low-rank approximations. *J. ACM*, 51(6):1025–1041, 2004.

[FKZ04] Joan Feigenbaum, Sampath Kannan, and Jian Zhang. Computing diameter in the streaming and sliding-window models. *Algorithmica*, 41(1):25–41, 2004.

[FM85] Philippe Flajolet and G. Nigel Martin. Probabilistic counting algorithms for data base applications. *J. Comput. Syst. Sci.*, 31(2):182–209, 1985.

[FS82] M. Fischer and S. Salzberg. Finding a majority among n votes. *Journal of Algorithms*, 3(4):362–380, 1982.

[FS01] Jessica H. Fong and Martin Strauss. An approximate L^p-difference algorithm for massive data streams. *Discrete Mathematics and Theoretical Computer Science*, 4(2):301–322, 2001.

[FS05] Gereon Frahling and Christian Sohler. Coresets in dynamic geometric data streams. In *ACM Symposium on Theory of Computing*, pages 209–217, 2005.

[Gab90] Harold N. Gabow. Data structures for weighted matching and nearest common ancestors with linking. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 434–443, 1990.

[GG07] Anna Gal and Parikshit Gopalan. Lower bounds on streaming algorithms for approximating the length of the longest increasing subsequence. In *IEEE Symposium on Foundations of Computer Science*, 2007.

[GGI$^+$02a] Anna C. Gilbert, Sudipto Guha, Piotr Indyk, Yannis Kotidis, S. Muthukrishnan, and Martin Strauss. Fast, small-space algorithms for approximate histogram maintenance. In *ACM Symposium on Theory of Computing*, pages 389–398, 2002.

[GGI$^+$02b] Anna C. Gilbert, Sudipto Guha, Piotr Indyk, S. Muthukrishnan, and Martin Strauss. Near-optimal sparse fourier representations via sampling. In *STOC*, pages 152–161, 2002.

[GH06] Sudipto Guha and Boulos Harb. Approximation algorithms for wavelet transform coding of data streams. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 698–707, 2006.

[GIM07] Sudipto Guha, Piotr Indyk, and Andrew McGregor. Sketching information divergences. In *Conference on Learning Theory*, pages 424–438, 2007.

[GIMS02] Sudipto Guha, Piotr Indyk, S. Muthukrishnan, and Martin Strauss. Histogramming data streams with fast per-item processing. In *International Colloquium on Automata, Languages and Programming*, pages 681–692, 2002.

[GJKK07] Parikshit Gopalan, T.S. Jayram, Robert Krauthgamer, and Ravi Kumar. Estimating the sortedness of a data stream. In *ACM-SIAM Symposium on Discrete Algorithms*, 2007.

[GK01] Michael Greenwald and Sanjeev Khanna. Efficient online computation of quantile summaries. In *ACM International Conference on Management of Data*, pages 58–66, 2001.

[GKMS01] Anna C. Gilbert, Yannis Kotidis, S. Muthukrishnan, and Martin Strauss. Surfing wavelets on streams: One-pass summaries for approximate aggregate queries. In *International Conference on Very Large Data Bases*, pages 79–88, 2001.

[GKMS02] Anna C. Gilbert, Yannis Kotidis, S. Muthukrishnan, and Martin Strauss. How to summarize the universe: Dynamic maintenance of quantiles. In *International Conference on Very Large Data Bases*, pages 454–465, 2002.

[GKS06] Sudipto Guha, Nick Koudas, and Kyuseok Shim. Approximation and streaming algorithms for histogram construction problems. *ACM Trans. Database Syst.*, 31(1):396–438, 2006.

[GM99] Phillip B. Gibbons and Yossi Matia. Synopsis data structures for massive data sets. *DIMACS Series in Discrete Mathematics and Theoretical*

Computer Science: Special Issue on External emory Algorithms and Visualization, A:39–70, 1999.

[GM06] Sudipto Guha and Andrew McGregor. Approximate quantiles and the order of the stream. In *ACM Symposium on Principles of Database Systems*, pages 273–279, 2006.

[GM07a] Sudipto Guha and Andrew McGregor. A general approach to multi-pass stream lower-bounds. *Manuscript*, 2007.

[GM07b] Sudipto Guha and Andrew McGregor. Lower bounds for quantile estimation in random-order and multi-pass streaming. In *International Colloquium on Automata, Languages and Programming*, 2007.

[GM07c] Sudipto Guha and Andrew McGregor. Space-efficient sampling. In *AISTATS*, pages 169–176, 2007.

[GMMO00] Sudipto Guha, Nina Mishra, Rajeev Motwani, and Liadan O'Callaghan. Clustering data streams. In *IEEE Symposium on Foundations of Computer Science*, pages 359–366, 2000.

[GMP02] Phillip B. Gibbons, Yossi Matias, and Viswanath Poosala. Fast incremental maintenance of approximate histograms. *ACM Trans. Database Syst.*, 27(3):261–298, 2002.

[GMT05] Yu Gu, Andrew McCallum, and Don Towsley. Detecting anomalies in network traffic using maximum entropy estimation. In *Internet Measurement Conference*, page 345350, 2005.

[GMV06] Sudipto Guha, Andrew McGregor, and Suresh Venkatasubramanian. Streaming and sublinear approximation of entropy and information distances. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 733–742, 2006.

[GS06] Sumit Ganguly and Barna Saha. On estimating path aggregates over streaming graphs. In *ISAAC*, pages 163–172, 2006.

[GT02] Phillip B. Gibbons and Srikanta Tirthapura. Distributed streams algorithms for sliding windows. In *ACM Symposium on Parallel Algorithms and Architectures*, pages 63–72, 2002.

[GZ03] Anupam Gupta and Francis Zane. Counting inversions in lists. *ACM-SIAM Symposium on Discrete Algorithms*, pages 253–254, 2003.

[HK73] John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.*, 2(4):225–231, 1973.

[HRR99] Monika R. Henzinger, Prabhakar Raghavan, and Sridhar Rajagopalan. Computing on data streams. *External memory algorithms*, pages 107–118, 1999.

[HS03] John Hershberger and Subhash Suri. Convex hulls and related problems in data streams. In *Workshop on Management and Processing of Data Streams*, 2003.

[Ind00] Piotr Indyk. Stable distributions, pseudorandom generators, embeddings and data stream computation. *IEEE Symposium on Foundations of Computer Science*, pages 189–197, 2000.

[Ind01] Piotr Indyk. A small approximately min-wise independent family of hash functions. *J. Algorithms*, 38(1):84–90, 2001.

[Ind03] Piotr Indyk. Better algorithms for high-dimensional proximity problems via asymmetric embeddings. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 539–545, 2003.

[IW03] Piotr Indyk and David P. Woodruff. Tight lower bounds for the distinct elements problem. *IEEE Symposium on Foundations of Computer Science*, pages 283–288, 2003.

[IW05] Piotr Indyk and David P. Woodruff. Optimal approximations of the frequency moments of data streams. In *ACM Symposium on Theory of Computing*, pages 202–208, 2005.

[JG05] Hossein Jowhari and Mohammad Ghodsi. New streaming algorithms for counting triangles in graphs. In *International Conference on Computing and Combinatorics*, pages 710–716, 2005.

[JKV07] T.S. Jayram, Satyen Kale, and Erik Vee. Efficient aggregation algorithms for probabilistic data. In *ACM-SIAM Symposium on Discrete Algorithms*, 2007.

[JMMV07] T. S. Jayram, Andrew McGregor, S. Muthukrishnan, and Erik Vee. Estimating statistical aggregates on probabilistic data streams. In *ACM Symposium on Principles of Database Systems*, pages 243–252, 2007.

[JRS03] Rahul Jain, Jaikumar Radhakrishnan, and Pranab Sen. A direct sum theorem in communication complexity via message compression. In *International Colloquium on Automata, Languages and Programming*, pages 300–315, 2003.

[Kan01] Sampath Kannan. Open problems in streaming. *DIMACS Workshop on Streaming Data Analysis and Mining (Slides: `http://dimacs.rutgers.edu/Workshops/Streaming/abstracts.html`)*, 2001.

[KCC$^+$03] Sailesh Krishnamurthy, Sirish Chandrasekaran, Owen Cooper, Amol Deshpande, Michael J. Franklin, Joseph M. Hellerstein, Wei Hong, Samuel Madden, Frederick Reiss, and Mehul A. Shah. TelegraphCQ: An architectural status report. *IEEE Data Eng. Bull.*, 26(1):11–18, 2003.

[KMR$^+$94] Michael J. Kearns, Yishay Mansour, Dana Ron, Ronitt Rubinfeld, Robert E. Schapire, and Linda Sellie. On the learnability of discrete distributions. In *ACM Symposium on Theory of Computing*, pages 273–282, 1994.

[KN97] Eyal Kushilevitz and Noam Nisan. *Communication Complexity.* Cambridge University Press, 1997.

[KRRT99] Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew Tomkins. Extracting large-scale knowledge bases from the web. In *International Conference on Very Large Data Bases*, pages 639–650, 1999.

[KS92] Bala Kalyanasundaram and Georg Schnitger. The probabilistic communication complexity of set intersection. *SIAM J. Discrete Math.*, 5(4):545–557, 1992.

[KS95] Bahman Kalantari and Ali Shokoufandeh. Approximation schemes for maximum cardinality matching. Technical Report LCSR–TR–248, Laboratory for Computer Science Research, Department of Computer Science. Rutgers University, August 1995.

[KSP03] Richard M. Karp, Scott Shenker, and Christos H. Papadimitriou. A simple algorithm for finding frequent elements in streams and bags. *ACM Trans. Database Syst.*, 28:51–55, 2003.

[KW99] Jyrki Kivinen and Manfred K. Warmuth. Boosting as entropy projection. In *Conference on Learning Theory*, pages 134–144, 1999.

[Laf99] John D. Lafferty. Additive models, boosting, and inference for generalized divergences. In *Conference on Learning Theory*, pages 125–133, 1999.

[LNVZ06] David Liben-Nowell, Erik Vee, and An Zhu. Finding longest increasing and common subsequences in streaming data. *J. Comb. Optim.*, 11(2):155–175, 2006.

[LPP97] John D. Lafferty, Stephen Della Pietra, and Vincent J. Della Pietra. Statistical learning algorithms based on bregman distances. *Proc. of. Canadian Workshop on Information Theory*, 1997.

[LSO$^+$06] Ashwin Lall, Vyas Sekar, Mitsunori Ogihara, Jun Xu, and Hui Zhang. Data streaming algorithms for estimating entropy of network traffic. In *ACM SIGMETRICS*, 2006.

[LUW95] Felix Lazebnik, Vasiliy A. Ustimenko, and Andrew J. Woldar. A new series of dense graphs of high girth. *Bulletin of the AMS*, 32(1):73–79, 1995.

[LV87] F. Liese and F. Vajda. Convex statistical distances. *Teubner-Texte zur Mathematik, Band 95, Leipzig*, 1987.

[MAA05] Ahmed Metwally, Divyakant Agrawal, and Amr El Abbadi. Efficient computation of frequent and top-k elements in data streams. In *ICDT*, pages 398–412, 2005.

[MBBF99] Llew Mason, Jonathan Baxter, Peter Bartlett, and Marcus Frean. Functional gradient techniques for combining hypotheses. In *Advances in Large Margin Classifiers*. MIT Press, 1999.

[McG05] Andrew McGregor. Finding graph matchings in data streams. In *APPROX-RANDOM*, pages 170–181, 2005.

[MG82] Jayadev Misra and David Gries. Finding repeated elements. *Sci. Comput. Program.*, 2(2):143–152, 1982.

[Mor78] Robert Morris. Counting large numbers of events in small registers. *CACM*, 21(10):840–842, 1978.

[MP80] J. Ian Munro and Mike Paterson. Selection and sorting with limited storage. *Theor. Comput. Sci.*, 12:315–323, 1980.

[MRL98] Gurmeet Singh Manku, Sridhar Rajagopalan, and Bruce G. Lindsay. Approximate medians and other quantiles in one pass and with limited memory. In *ACM International Conference on Management of Data*, pages 426–435, 1998.

[MRL99] Gurmeet Singh Manku, Sridhar Rajagopalan, and Bruce G. Lindsay. Random sampling techniques for space efficient online computation of order statistics of large datasets. In *ACM International Conference on Management of Data*, pages 251–262, 1999.

[Mut06] S. Muthukrishnan. Data streams: Algorithms and applications. *Now Publishers*, 2006.

[MV80] Silvio Micali and Vijay V. Vazirani. An $O(\sqrt{V}E)$ algorithm for finding maximum matching in general graphs. In *IEEE Symposium on Foundations of Computer Science*, pages 17–27, 1980.

[Nis92] Noam Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12:449–461, 1992.

[NW93] Noam Nisan and Avi Wigderson. Rounds in communication complexity revisited. *SIAM J. Comput.*, 22(1):211–219, 1993.

[NWJ05] XuanLong Nguyen, Martin J. Wainwright, and Michael I. Jordan. Divergences, surrogate loss functions and experimental design. *Proceedings of NIPS*, 2005.

[Pre99] Robert Preis. Linear time 1/2-approximation algorithm for maximum weighted matching in general graphs. In *Symposium on Theoretical Aspects of Computer Science*, pages 259–269, 1999.

[PS04] Seth Pettie and Peter Sanders. A simpler linear time 2/3-ϵ approximation for maximum weight matching. *Inf. Process. Lett.*, 91(6):271–276, 2004.

[Raz92] Alexander A. Razborov. On the distributional complexity of disjointness. *Theor. Comput. Sci.*, 106(2):385–390, 1992.

[RRR$^+$07] Sofya Raskhodnikova, Dana Ron, Ronitt Rubinfeld, Amir Shpilka, and Adam Smith. Sublinear algorithms for approximating string compressibility and the distribution support size. In *International Workshop on Randomization and Approximation Techniques in Computer Science*, 2007.

[SBAS04] Nisheeth Shrivastava, Chiranjeeb Buragohain, Divyakant Agrawal, and Subhash Suri. Medians and beyond: new aggregation techniques for sensor networks. In *SenSys*, pages 239–249, 2004.

[Sha48] Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423 and 623–656, July and October 1948.

[SS02] Michael E. Saks and Xiaodong Sun. Space lower bounds for distance approximation in the data stream model. *ACM Symposium on Theory of Computing*, pages 360–369, 2002.

[SW07] Xiaoming Sun and David Woodruff. The communication and streaming complexity of computing the longest common and increasing subsequences. In *ACM-SIAM Symposium on Discrete Algorithms*, 2007.

[Top00] Flemming Topsøe. Some inequalities for information divergence and related measures of discrimination. *IEEE Transactions on Information Theory*, 46(4):1602–1609, 2000.

[TZ01] Mikkel Thorup and Uri Zwick. Approximate distance oracles. In *ACM Symposium on Theory of Computing*, pages 183–192, 2001.

[Č81] Nikolaĭ Nikolaevich Čencov. Statistical decision rules and optimal inference. *Transl. Math. Monographs, Am. Math. Soc. (Providence)*, 1981.

[Vit85] Jeffrey Scott Vitter. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11(1):37–57, 1985.

[Woo04] David P. Woodruff. Optimal space lower bounds for all frequency moments. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 167–175, 2004.

[WP05] Arno Wagner and Bernhard Plattner. Entropy based worm and anomaly detection in fast IP networks. In *IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises*, pages 172–177, 2005.

[XZB05] Kuai Xu, Zhi-Li Zhang, and Supratik Bhattacharyya. Profiling internet backbone traffic: behavior models and applications. In *SIGCOMM*, pages 169–180, 2005.

[Yao79] Andrew Chi-Chih Yao. Some complexity questions related to distributive computing (preliminary report). *ACM Symposium on Theory of Computing*, pages 209–213, 1979.

[Yao80] Andrew Chi-Chih Yao. Lower bounds by probabilistic arguments. In *IEEE Symposium on Foundations of Computer Science*, pages 420–428, 1980.

[ZC06] Hamid Zarrabi-Zadeh and Timothy M. Chan. A simple streaming algorithm for minimum enclosing balls. In *Canadian Conference on Computational Geometry*, pages 139–142, 2006.

[Zel06] Mariano Zelke. k-connectivity in the semi-streaming model. *CoRR*, cs/0608066, 2006.

[Zha05] Jian Zhang. *Massive Data Streams in Graph Theory and Computational Geometry.* PhD thesis, Yale University, 2005.