CMPSCI 311: Introduction to Algorithms **Spring 2018**

Homework 4

Released 3/21/2018 Due 11:59pm 4/4/2016 in Gradescope

Instructions. You make work in groups, but you must individually write your solutions yourself. List your collaborators on your submission.

If you are asked to design an algorithm as part of a homework problem, please provide: (a) the pseudocode for the algorithm, (b) an explanation of the intuition for the algorithm, (c) a proof of correctness, (d) the running time of your algorithm, and (e) justification for your running time analysis.

There are 4 questions, each worth 25 points total.

Submission instructions. This assignment is due by 11:59pm on 4/4/2018 in Gradescope. Please submit a pdf file. You may submit a scanned handwritten document, but a typed submission is preferred.

1. **Dynamic Programming Short Answer**

   (a) **Scheduling.** UMass IT is trying to write a scheduler for two identical machines. They know that there are n jobs and that the ith job takes t_i seconds, regardless of which machine it is assigned to (the machines are identical). Each t_i is a positive integer. Each job must be assigned to one of the two machines, and we want to assign jobs so as to minimize the completion time of the last job to finish.

   More precisely, if we assign a set $S_1 \subseteq \{1, \ldots, n\}$ of jobs to the first machine then we necessarily assign $S_2 = \{1, \ldots, n\} \setminus S_1$ to the second. The finish time is $F_1 = \sum_{i \in S_1} t_i$ for machine one and $F_2 = \sum_{i \in S_2} t_i$ for machine two. The final finish time is then $F = \max\{F_1, F_2\}$.

   Design an efficient algorithm to find the cost of the optimal schedule, which is the smallest value F attainable by an assignment of jobs to machines. You do not need to find the assignment itself. If each t_i is a k-bit integer, what is the running time of your algorithm as a function n and k?

   (b) **Longest Common Substring.** Given two strings $X = x_1 x_2 \ldots x_m$ and $Y = y_1 y_2 \ldots y_n$, we wish to find the length of the *longest common substring*, which is the largest k such that there exists i, j with $x_i x_{i+1} \ldots x_{i+k-1} = y_j y_{j+1} \ldots y_{j+k-1}$. Design an $O(nm)$ algorithm to find both the length of the longest common substring, and the actual substring.

2. **A Strategy Game.** Jesse and Russell found a pile of money and want to split it up amongst them. They decide to play a strategy game using the money, so that the smarter of them gets to keep the most. Here's how the game works:

   There are n bills of various denominations and we line them up in a list $S[1..n]$ so the ith bill has value $S[i]$. On a player's turn, they may take either the first bill or the last bill in the list, and then it becomes the other player's turn. The goal is to collect as much money as possible

   Russell wants to play this game optimally, so he's asking you to design an algorithm that computes an optimal strategy. Your job is to (1) design an $O(n^2)$ time algorithm that given the list $S[1..n]$ precomputes some information to give to Russell before the game, and (2) an $O(1)$ time algorithm that given a configuration of the game and the precomputed information, identifies the best move.

3. **Optimal BST.** A binary search tree (BST) is a data structure that contains n integers organized in a binary tree with the following properties:

   - Each node in the tree is associated with an integer (so there are n total nodes).
   - If a node contains a number y then all nodes in the left subtree, if there are any, must contain some number x such that $x < y$ and all nodes in the right subtree, if there are any, must contain some number z such that $y < z$.

   If a binary search tree has items $x_1 < x_2 < \ldots < x_n$, define the *cost* to lookup an item x to be the number of comparisons made in searching for it in the tree (if the item is in the tree, this is always 1 plus the distance between the item and the root). Thus, the cost for looking up the root is 1. In this question we will only ever do lookups for items in the tree.

   Given items $x_1 < x_2 < \ldots < x_n$, there are many BSTs that we can build on these numbers. If we have a sequence of requests, we can calculate the total cost of the requests of the different BSTs, and thus we can define an *optimal BST* for a request sequence.

   (a) Observe that the cost of a sequence of requests does not depend on the order (since the tree is static), but only on the number of times each key is requested. If $n = 4$ and the request sequence is $[2, 4, 3, 6]$ (which means x_1 is accessed twice, x_2 is accessed 4 times, and so on), what is an optimal BST for this sequence?

   (b) In the general case, if the optimal BST for a given request sequence has x_i at the root, prove that the left subtree must be an optimal BST for the requests on $x_1, \ldots, x_{i-1}$ and the right subtree must be an optimal BST for the requests on $x_{i+1}, \ldots, x_n$.

   (c) Use the above observation to give a general algorithm for constructing an optimal BST for a given sequence $x_1 < x_2 < \ldots < x_n$ and their counts $c_1, \ldots, c_n$ (where c_i is the number of times x_i is accessed). The running time of your algorithm should be $O(n^3)$.

4. **Piecewise Regression.** In this problem, we will design an algorithm for a form of regression. We are given a list of n points x_i and we want to output another list y_i, which is close to x, but which has very few *changepoints*. A changepoint in y is an index i such that $y_i \neq y_{i+1}$.

   More formally for a vector y, the *segments* S of y are the sets of contiguous indices for which y has the same value. For example, the vector $y = (0, 0, 1, 2, 2, 1)$ has segments $S = \{\{1, 2\}, \{3\}, \{4, 5\}, \{6\}\}$. Notice that the segments form a partition of the set $\{1, \ldots, n\}$. We use $s \in S$ to denote a segment and we must have $y_i = y_j$ for $i, j \in s$. We use $|s|$ to denote the number of elements in the segment.

   Intuitively if y has few changepoints, then it should have few segments, or relatedly, it should have large segments. This question formulates two optimization problems based on this intuition.

   (a) For a fixed list x, define the cost of a vector y, with segments S as

   $$\text{cost}_1(y) = \sum_{j=1}^{n} (y_j - x_j)^2 - \sum_{s \in S} |s|$$

   Give an $O(n)$ algorithm that takes as input x and finds the vector y that minimizes this cost.

   (b) Redefine the cost:

   $$\text{cost}_2(y) = \sum_{j=1}^{n} (y_j - x_j)^2 - \sum_{s \in S} |s|^2$$

   Give an $O(n^3)$ algorithm that takes as input x and finds the vector y that minimizes this cost.

   Fact. In both cases, if s is a segment in the optimal solution, then the optimal y has $y_i = \frac{1}{|s|} \sum_{j \in s} x_j$ for all $i \in s$. This minimizes the squared error, since all ys must take the same value in the segment. You may use this fact without proof. Assume that computing the mean of k numbers takes $O(k)$ time.

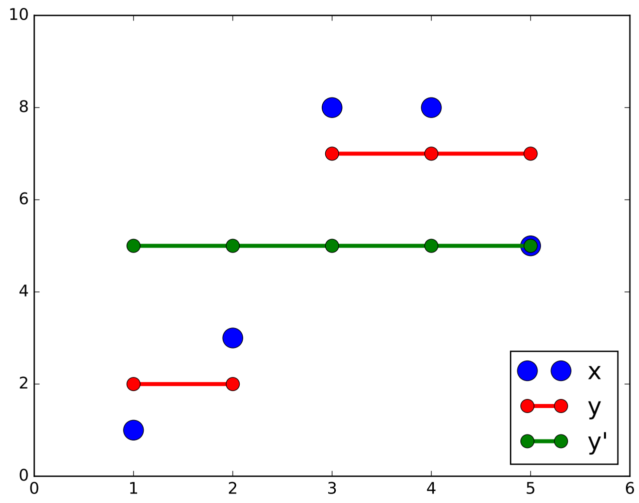

Figure 1.1: Example of piecewise regression.

Example. (See Figure 1.1) Suppose that $x_1 = 1, x_2 = 3, x_3 = 8, x_4 = 8, x_5 = 5$. Consider two segmentations: $S = \{\{1,2\},\{3,4,5\}\}$ and $S' = \{\{1,2,3,4,5\}\}$. Under S, the optimal y is $y_1 = 2, y_2 = 2, y_3 = 7, y_4 = 7, y_5 = 7$. Under S', the optimal y' is $y'_1 = y'_2 = y'_3 = y'_4 = y'_5 = 5$. The values of y and y' are computed by taking the average of the x values in each segment. So y_1 and y_2 are both equal to $(x_1 + x_2)/2$. The costs are

$$\begin{aligned} \text{cost}_1(y) = 8 - (2+3) = 3 \quad &\text{and} \quad \text{cost}_2(y) = 8 - (4+9) = -5 \\ \text{cost}_1(y') = 38 - 5 = 33 \quad &\text{and} \quad \text{cost}_2(y') = 38 - 25 = 13 \end{aligned}$$

So in both cases the segmentation S with vector y is better.

5. **(0 points).** How long did it take you to complete this assignment?