CMPSCI 311: Introduction to Algorithms **Fall 2016**

Homework 3

Released 10/7/2016 Due 8:00pm 10/21/2016 in Moodle

Note: *LaTeX template courtesy of UC Berkeley EECS dept.*

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.

Submission instructions. This assignment is due by 8:00pm on 10/21/2016 in Moodle. Please submit a pdf file. You may submit a scanned handwritten document, but a typed submission is preferred.

1. **(10 points) Minimizing waiting time.** You are working at a restaurant and have n customers waiting to be served. Unfortunately, you only have one table, so you can only serve one customer at a time. Each customer i takes t_i minutes to finish eating, at which point you can seat another customer. All of the customers arrived at exactly the same time, so you plan to seat them to minimize the total time spent in the restaurant. For example, if the customers are numbered $1, \ldots, n$ and you seat them in this order, then customer i spends $\sum_{j=1}^{i} t_i$ time in the restaurant, because she first waits for everyone before her to finish eating and then has to finish her own meal.

   We wish to minimize the total time spent in the restaurant, which is,

   $$T = \sum_{i=1}^{n} (\text{time spent by customer } i).$$

   Design an efficient algorithm that produces an ordering that minimizes this quantity.

2. **(10 points) Spanning Trees K&T Ch4.Ex9.** Let $G = (V, E)$ be a graph with edge weights w_e for each e. A tree T is a minimum-bottleneck spanning tree of G if the largest edge weight in T is no larger than the largest edge weight of any other tree T'. That is if we define $\text{cost}(T) = \max_{e \in T} w_e$, then T is a minimum-bottleneck spanning tree (MBST) if $\text{cost}(T) \leq \text{cost}(T')$ for any other tree T'.

   (a) If T is a MBST of G, is it also a minimum spanning tree? Prove or give a counterexample.

   (b) If T is a MST of G, is it also a MBST? Prove or give a counterexample.

3. **(20 points) Degree Statistics. K&T Ch4.Ex30.** Given a sequence of n natural numbers $d_1, d_2, \ldots, d_n$, design an algorithm that decides, in polynomial time, whether there exists an undirected graph $G = (V, E)$ whose node degrees are precisely $d_1, \ldots, d_n$. So if $V = \{v_1, \ldots, v_n\}$, then we should have $\deg(v_1) = d_1, \deg(v_2) = d_2$ and so on. The graph can not contain self-loops or multiple edges between the same pair of vertices. Note your algorithm need not construct the graph, it just needs to decide if one exists.

4. **(20 points) Fast Matrix Multiplication.** Given two $n \times n$ matrices X, Y of integers, their product is another $n \times n$ matrix Z with,

   $$Z_{ij} = \sum_{k=1}^{n} X_{ik} Y_{kj}.$$

   Naively, computing Z seems to take $O(n^3)$ time since there are n^2 entries, and computing the value for a single entry involves n addition operations. In this problem, we'll develop a faster matrix multiplication algorithm. For simplicity, assume that n is a power of 2.

(a) First, define eight $n/2 \times n/2$ matrices A, B, C, D, E, F, G, H so that,

$$X = \begin{bmatrix} A & B \\ C & D \end{bmatrix}, Y = \begin{bmatrix} E & F \\ G & H \end{bmatrix}$$

Prove that,

$$Z = XY = \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

(b) This suggests a divide and conquer algorithm for matrix multiplication. Describe the algorithm in words, write down the running time as a recurrence relation, and solve the recurrence.

(c) The above algorithm uses eight recursive calls, but a more clever decomposition due to Strassen can achieve the same result using only seven recursive calls of similar form. What is the recurrence for such an algorithm and what would the running time be?

5. **(20 points) Decimal to Binary** Recall that in class we designed an algorithm that takes two n digit numbers x, y and returns their product xy, in base 10, in time $O(n^a)$ where $a = \log_2 3$. In this problem, we'll use a subroutine `fastmultiply`(x,y) that takes two n *bit* numbers and returns their product, in binary, in time $O(n^a)$ with $a = \log_2 3$. We'll use fast binary multiplication to convert numbers from decimal to binary.

   As representation, we will represent decimal numbers as strings and binary numbers using bits as usual. Given this representation, you can index decimal numbers, but are unable to multiply two decimal numbers together, without first converting to binary.

   (a) We'll first design an algorithm to convert the decimal number 10^n to binary. Assume that n is a power of 2.

   def `pwr2bin`(n):
   If $n = 1$: return 1010 (decimal 10 in binary)
   Else:
   $z =$ /* FILL ME IN */
   Return `fastmultiply`(z,z).

   What is the appropriate value for z? What is the running time of the algorithm?

   (b) The next procedure is supposed to convert a decimal integer x with n digits into binary. Assume that n is a power of 2.

   def `dec2bin`(x):
   If `length`(x) = 1: return `binary`(x)
   Else:
   Split x into x_L the leading $n/2$ digits and x_R, the trailing $n/2$ digits.
   Return /* FILL ME IN */.

   The subroutine `binary`(x) performs a lookup into a table containing the binary value of all decimal numbers $0, \ldots, 9$. What are we supposed to return? What is the running time of this algorithm?

6. **(20 points) Discrete Optimization K&T Ch5.Ex6 and Ex7.** In this problem we will consider graphs where each vertex v is labeled with a real number $x(v)$. The $x(v)$ are all distinct (no two vertices share the same value). A node v is a *local minimum* if $x(v) < x(u)$ for all of v's neighbors u. In this problem, we aim to design efficient algorithms for finding a local minima.

   The labels are initially unknown to you. You can obtain the value $x(v)$ of a vertex v by calling a subroutine `query(v)`. We would like to design algorithms that make few calls to this subroutine.

   (a) Suppose that G is a complete n-node binary tree, so it has $2^d - 1$ vertices for some d. Design an algorithm that finds a local minima of G using $O(\log_2 n)$ calls to the subroutine `query`.

(b) Suppose now that G is a $n \times n$ grid graph, in other words it is a graph with nodes associated with pairs of natural numbers $(i, j), 1 \leq i, j \leq n$ and edges between vertices (i, j) and (k, l) if and only if $|i - k| + |j - l| = 1$. Design an algorithm to find a local minima of G using $O(n)$ calls to the subroutine `query`. Note that this graph has n^2 vertices.

7. **(0 points).** How long did it take you to complete this assignment?