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

Homework 3

Released 2/21/2018 Due 11:59pm 3/7/2018 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.

Submission instructions. This assignment is due by 11:59pm on 3/7/2018 in Gradescope. Please submit a pdf file. You may submit a scanned handwritten document, but a typed submission is preferred. It will be extremely helpful if in your submission, you start each question on a new page.

1. **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.

2. **More Spanning Trees K&T Ch4.Ex10.** Let $G = (V, E)$ be an (undirected) graph with weights $w_e \geq 0$ on the edges $e \in E$. Assume you are given a minimum spanning tree T in G. Now assume that a new edge is added to G, connecting two nodes $u, v \in V$ with cost w.

   (a) Give an efficient algorithm to test if T remains the minimum-cost spanning tree with the new edge added to G (but not to the tree T). Make your algorithm run in time $O(|E|)$. Can you do it in $O(|V|)$ time? Please note any assumptions you make about what data structure is used to represent the tree T and the graph G.

   (b) Suppose T is no longer the minimum-cost spanning tree. Give a linear-time algorithm (time $O(|E|)$) to update the tree T to the new minimum-cost spanning tree.

3. **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? You do not have to come up with the algorithm, but rather your task is to analyze the running time of an algorithm like the one above, but that only makes 7 recursive calls.

4. **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?

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