COMPSCI 311: Introduction to Algorithms **Fall 2018**

Homework 3

Released 10/4/2018 Due 10/18/2018 11:59pm in Gradescope

Instructions. You may 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.

Submissions. Please submit a PDF file. You may submit a scanned handwritten document, but a typed submission is preferred. It will be very helpful if in your submission each question starts on a new page.

1. **(20 points) Recurrences** Give asymptotic upper and lower bounds in the following recurrences. Assume $T(n)$ constant for $n \leq 2$. Make your bounds as tight as possible and justify your answers.

   a) $T(n) = T(n/3) + T(n/2) + n$

   b) $T(n) = T(n-1) + \log n$

   c) $T(n) = 2T(n/2) + n/\log n$ Fact: $\sum_{k=1}^{n} 1/k = \Theta(\log n)$

2. **(15 points) Divide and Conquer the Tree**

   Consider the following algorithm to compute a MST for a connected weighted graph $G = (V, E)$: partition V into two subsets differing in size by at most 1; recursively find the MSTs for these two "halves"; join these two MSTs by the smallest-weight edge across the cut. Does this give a valid algorithm? Prove or give a counterexample. What is the complexity?

3. **(20 points) Decimal to Binary**

   We have available a subroutine `fastmultiply`(x,y) that takes two numbers of which the longest has n bits and returns their product, in binary, in time $\Theta(n^{\log_2 3})$. It works like the one discussed in class for base 10. We'll use this 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.

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

   How do you fill in the code? 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.

   ```
   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 $n - n/2$ 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 should the algorithm return? What is the running time of this algorithm?

4. **(15 points) Almost Balanced Trees**

   We'll call a binary tree *almost balanced* if for any of its nodes, its two subtrees differ in height by at most 1. (The height of an empty tree is 0 and the height of a single node is 1). Write an algorithm that determines if a tree is almost balanced, and analyze its complexity. You are not allowed to modify the tree (store anything in its nodes).

5. **(10 points) Discrete Optimization** (K&T Ch5 Ex6). Consider a connected undirected graph G where each node v is labeled with some value that you can obtain by calling a subroutine `query(v)`. No two vertices share the same value. A node is a local minimum if its value is less than that of any of its neighbors.

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

6. **(20 points) Honest or Liar?**

   We have a set of n people. The only way to determine their honesty is to pair them up: they will talk and then each will report about the other one. Since honest people always tell the truth, while liars may or may not, we have the following situations after A and B have talked: If A says "B is honest" and B says "A is honest", we know both are honest or both are liars. For any other combination of answers, we only know at least one of the two is a liar. We know that more than $n/2$ people are honest.

   a) Consider the problem of finding an honest person. Show that using at most $n/2$ tests by pairing is enough to reduce the problem to one of at most size $\lceil n/2 \rceil$.

   b) Show that you can identify all honest people using $\Theta(n)$ tests by pairing. Write the recurrence that describes the number of tests and solve it.

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