CSE 101 Fall 2018
Homework 6
Due: Tuesday, November 20, 2018 at 11:59pm

Instructions

This homework assignment may be completed in groups of size 1-4. The solutions must be **typed** (using a computer.) Figures and graphs can be handdrawn. For algorithm descriptions we require a high-level English description AND an implementation description. If you find it necessary to also include a pseudocode to help understand your description then feel free to include it.

Please refer to the course page for requirements in writing up answers for algorithm questions.

---

1. (25 points) Construct a divide and conquer recursive algorithm that takes as input a binary tree T and computes, for each pair of nodes x and y in T, the deepest node that is an ancestor of both x and y, and stores it in an array $LCA[x, y]$. The main idea is that if x is in the left sub-tree of the root, and y is in the right sub-tree, then the only common ancestor of x and y is the root. Otherwise, the least common ancestor is in the subtree that contains both x and y.

   Each non-leaf in T, x, has left-child $x.left$, and right child $x.right$, and each non-root has parent $x.parent$. (Child pointers at leaves and the parent pointer at the root return NULL). For any searches, use depth-first search procedure DFS that is linear-time in the size of the sub-tree and returns the list of nodes in the sub-tree.
   (Give the recursive algorithm - 10 points, Give the recursion for runtime indicated by your algorithm in case of a complete binary tree - 5 points, Solve this recursion for runtime analysis - 10 points)

2. (25 points) Consider the following problem:

   We are given an array of real numbers $V[1..n]$. We wish to find a subset of array positions, $S \subseteq [1...n]$ that maximizes $\sum_{i \in S} V[i]$ subject to no two consecutive array positions being in S. For example, say $V = [10, 14, 12, 6, 13, 4]$, the best solution is to take elements $1, 3, 5$ to get a total of $10 + 12 + 13 = 35$. If instead, we try to take the 14 in position 2, we must exclude the 10 and 12 in positions 1 and 3, leaving us with the second best choice $2, 5$ giving a total of $14 + 13 = 27$. Design a divide and conquer algorithm that is based on a case analysis, do we pick position $n/2$ or not? Your algorithm should just find the best sum, not the set of positions.
   (Note: your algorithm should run in $O(n^2)$ time.)

3. (25 points) Your input list of length n consists of of n-1 consecutive non-negative integers in the range of 1 to n+1 and one of the integers is missing in the list.
   for example:
   Input : ar: [1, 2, 3, 4, 6, 7, 8]
   Assume that there are no duplicates in list. Suggest the most efficient divide and conquer algorithm to find the missing number in this list.
   (Algorithm description - 10 points, runtime recursion - 5 points, recursion solution and algorithm efficieny - 10 points)

4. (25 points)

a) Consider the following recursive algorithm: We are given a binary tree T. In addition, at each node x except the root, we are given a positive real value $d(x)$ called the distance between x and its parent, We want to find the distances between every two nodes x, y in the tree and store it in an array $D[x, y]$.

Let L_x represent the left sub-tree rooted at x and R_x the right sub-tree. The algorithm to do so is as follows:

(a) Distances(root){compute all distances between pairs of nodes in the sub-tree rooted at *root*. }

(b) If $root = NIL$ then halt.

(c) ELSE do:

(d) begin;{else}

(e) Distances(lc(root));

(f) Distances(rc(root));

(g) For each $I \in L_{root}$, $D[I, root] := D[I, lc(root)] + d(lc(root))$.

(h) For each $J \in R_{root}$, $D[J, root] := D[J, rc(root)] + d(rc(root))$.

(i) For each $I \in L_{root}$ do:

(j)   For each $J \in R_{root}$ do:

(k) $D[I, J] = D[I, root] + D[J, root]$.

(l) end;{else}

Consider the case when the tree is almost perfectly balanced, i.e, for every x, $|L(x)| - 1 \leq |R(x)| \leq |L(x)| + 1$. Give a recurrence relation for the time the algorithm takes in this case, and solve it to get the order of the time. (Finding correct recursion- 10 points, solving it to get correct run time - 10 points)

b) Consider the recurrence :

$T(n) = 8T(n/2) + n^3/log(n)$

What does the master theorem suggest as a solution to this problem ? (5 points)