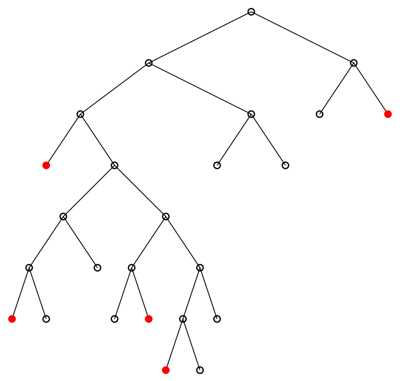
CS/ECE 374 A (Spring 2022)
Homework 7 (due March 24 Thursday at 10am)

Instructions: As in previous homeworks.

Problem 7.1: *(Social distancing for koalas?)* We are given a binary tree T with n nodes, and a number k. (You may assume that every non-leaf node has exactly 2 children.) We want to pick a subset S of k leaves that maximizes the value[1] $\Delta(S) = \min_{u,v \in S} d(u,v)$, where $d(u,v)$ denotes the distance between u and v, i.e., the length of the (unique) path from u to v in T. (Intuitively, $\Delta(S)$ represents the amount of "separation" between the leaves in S.) You will design an efficient dynamic programming algorithm to solve this problem.

In the example below, one optimal subset S is shown in red, with $\Delta(S) = 5$. (Turn the picture upside down if you are a koala :-)



$k = 5$

(a) Let T_v denote the subtree rooted at a node v. Consider the following definition of subproblems: for each node v and each $i \in \{0, \ldots, k\}$ and $j \in \{0, \ldots, n\}$,[2] let $F(v, i, j)$ be the maximum of $\Delta(S)$ over all subsets S of the leaves in T_v, subject to the following two constraints:

- S contains exactly i leaves, and
- $\min_{u \in S} d(u, v) = j$ (i.e., the closest distance from S to the root of T_v is j).

(As usual, if no feasible solution exists, the maximum is $-\infty$.)

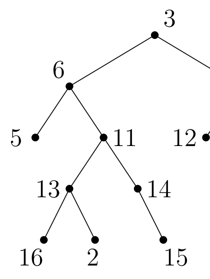
First, describe a recursive formula for $F(v, i, j)$. Include appropriate base cases and justification of your formula, and indicate how the final optimal value to the original problem can be expressed in terms of $F(\cdot, \cdot, \cdot)$.

(b) Next, specify the evaluation order, write pseudocode to solve the problem, and analyze the running time as a function of n and k. Your algorithm only needs to output the optimal value.

[1] In the minimum, we require $u \neq v$.
[2] It's ok if you ignore the $i = 0$ case.

Problem 7.2: *(Turning a sequence into a tree)* We are given a sequence of n numbers $\langle a_1, \ldots, a_n\rangle$. We want to compute a binary tree T with nodes $a_1, \ldots, a_n$ such that the *inorder traversal* is precisely $\langle a_1, \ldots, a_n\rangle$, while minimizing the cost function $c(T) = \sum_{(a_i,\, a_j) \text{ is an edge of } T} |a_i - a_j|$. (Here, an internal node of T may have degree 1 or 2.) You will design two efficient dynamic programming algorithms to solve this problem.

For example, for the input sequence $\langle 5, 6, 16, 13, 2, 11, 14, 15, 3, 12, 8, 7\rangle$, one feasible solution is the following, which has cost $|3-6| + |3-8| + |6-5| + |6-11| + |8-12| + |8-7| + |11-13| + |11-14| + |13-16| + |13-2| + |14-15| = 39$ (but we are not claiming that this is optimal).



(a) Consider the following definition of subproblems: for each i, j, m with $1 \le i \le m \le j \le n$, let $C(i, j, m)$ be the minimum of $c(T)$ over all binary trees T with nodes $a_i, a_{i+1}, \ldots, a_j$ such that

- the inorder traversal of T is $\langle a_i, a_{i+1}, \ldots, a_j\rangle$, and
- the root of T is a_m.

First, describe a recursive formula for $C(i, j, m)$. Include appropriate base cases and justification of your formula, and indicate how the final optimal value to the original problem can be expressed in terms of $C(\cdot, \cdot, \cdot)$.

(b) Next, specify the evaluation order, and analyze the running time of the resulting dynamic programming algorithm. (Your algorithm only needs to compute the optimal cost.) For this problem, there is no need to give pseudocode (if your descriptions of the recursive formula and evaluation order are precise enough).

(c) Consider a different definition of subproblem: for each i, j, p with $1 \le i \le j \le n$ and $1 \le p \le n$, let $C'(i, j, p)$ be the minimum of $c(T) + |a_p - r(T)|$ over all binary trees T with nodes $a_i, a_{i+1}, \ldots, a_j$ such that the inorder traversal of T is $\langle a_i, a_{i+1}, \ldots, a_j\rangle$, where $r(T)$ denotes the root of T.

Describe a new recursive formula for $C'(i, j, p)$. Include appropriate base cases, justification of your formula, and indicate how the final optimal value to the original problem can be expressed in terms of $C'(\cdot, \cdot, \cdot)$.

(d) Next, using (c), specify the evaluation order, and analyze the running time of the resulting dynamic programming algorithm. (Your algorithm only needs to compute the optimal cost.) Again, there is no need to give pseudocode (if your descriptions of the recursive formula and evaluation order are precise enough). Your algorithm in (d) should be more efficient than your algorithm in (b).[3]

[3]... assuming that you didn't do anything too clever in (b) (otherwise, (b) and (d) could be equally efficient).