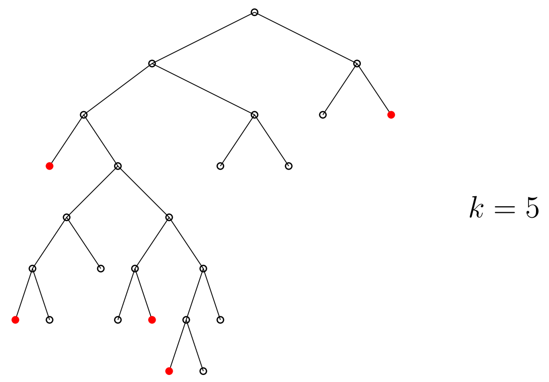
CS/ECE 374 A (Spring 2022)
Homework 7 Solutions

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 $\Delta(S) = \min_{u,v\in S} d(u,v)$, 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 :-)



(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 $j \in \{0,\ldots,n\}$, 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.

Solution:

(a) Let r be the root of the entire tree. The final answer we want is $\max_{j \in \{0,\ldots,n\}} F(r,k,j)$.

Base case. For each leaf v, $F(v,i,j) = -\infty$ for each $i \in \{1,\ldots,k\}$ and $j \in \{0,\ldots,n\}$, except $F(v,1,0) = \infty$. For each internal node v, $F(v,i,0) = -\infty$ for each $i \in \{0,\ldots,n\}$.

Recursive formula. For each internal node v with children v_1 and v_2, and each $i \in \{1,\ldots,k\}$ and $j \in \{1,\ldots,n\}$,

$$F(v,i,j) \;=\; \max \begin{cases} \max\limits_{i' \in \{1,\ldots,i-1\},\, j' \in \{j-1,\ldots,n\}} \min\{F(v_1,i',j'),\; F(v_2,i-i',j-1),\; j+j'+1\} \\ \max\limits_{i' \in \{1,\ldots,i-1\},\, j'' \in \{j-1,\ldots,n\}} \min\{F(v_1,i',j-1),\; F(v_2,i-i',j''),\; j+j''+1\} \\ F(v_1,i,j-1) \\ F(v_2,i,j-1) \end{cases}$$

(Note: the expression can be simplifed without separating out the last two cases, if we add boundary base cases for $i = 0$ and use sentinels, namely, set $F(v,0,-\infty) = \infty$ for leaves v.)

Justification. Consider the optimal solution S corresponding to $F(v,i,j)$. Let S_1 denote the subset of the leaves of S from the left subtree at v_1, and let S_2 denote the subset of the leaves of S from the right subtree at v_2. Suppose we know that $|S_1| = i'$ and $|S_2| = i-i'$. And suppose we know that $\min_{u \in S_1} d(u,v_1) = j'$ and $\min_{u \in S_2} d(u,v_2) = j''$. We must have $\min\{j',j''\}+1 = j$. In other words, we have (i) $j'' = j-1$ and $j' \in \{j-1,\ldots,n\}$, or (ii) $j' = j-1$ and $j'' \in \{j-1,\ldots,n\}$.
Now, $\Delta(S) = \min\{\Delta(S_1), \Delta(S_2), (j'+1)+(j''+1)\}$, since the closest pair in S may have both leaves in S_1 (in which case the distance is $\Delta(S_1)$), or both leaves in S_2 (in which case the distance is $\Delta(S_2)$), or one leaf in S_1 and one leaf in S_2 (in which case the distance is $(j'+1)+(j''+1)$ by routing through the root v).
Thus, the optimal solution has value $\min\{F(v_1,i',j'), F(v_2,i-i',j''), (j'+1)+(j''+1)\}$.
Exceptional cases are when $i' = i$, where the value is just $F(v_1,i,j-1)$, and when $i' = 0$, where the value is just $F(v_2,i,j-1)$.
We don't know i',j',j'' in advance, and so will try all feasible choices and take the maximum.

(b) *Evaluation order.* Any bottom-up ordering of v (e.g., postorder traversal).

Pseudocode. We use recursion to evaluate in bottom-up order. (This is still dynamic programming—we will use a table to memoize $F(\cdot)$.)

```
Evaluate($v$):
1. if $v$ is a leaf then
2.     for $i = 1$ to $k$ do
3.         for $j = 0$ to $n$ do
4.             $F[v,i,j] = -\infty$
5.     $F[v,1,0] = \infty$
6.     return
```

```
 7. let $v_1, v_2$ be the children of $v$
 8. Evaluate($v_1$)
 9. Evaluate($v_2$)
10. for $i = 1$ to $k$ do
11.     $F[v, i, 0] = -\infty$
12.     for $j = 1$ to $n$ do
13.         $F[v, i, j] = \max\{F[v_1, i, j-1], F[v_2, i, j-1]\}$
14.         for $i' = 1$ to $i - 1$ do
15.             for $j' = j - 1$ to $n$ do
16.                 $F[v, i, j] = \max\{F[v, i, j],\ \min\{F[v_1, i', j'],\ F[v_2, i - i', j - 1],\ j + j' + 1\}\}$
17.             for $j'' = j - 1$ to $n$ do
18.                 $F[v, i, j] = \max\{F[v, i, j],\ \min\{F[v_1, i', j - 1],\ F[v_2, i - i', j''],\ j + j'' + 1\}\}$
```

Algorithm:

```
19. let $r$ be the root
20. Evaluate($r$)
21. return $\min_{j=0,\ldots,n} F[r, k, j]$
```

Analysis. Lines 1–6 take $O(kn)$ time. Lines 10–18 take $O(k \cdot n \cdot k \cdot n) = O(k^2 n^2)$ time. Thus, over all n nodes, Evaluate() takes $O(k^2 n^2 \cdot n) = O(k^2 n^3)$ time. Line 21 takes $O(n)$ additional time. Total running time: $O(k^2 n^3)$.

(Note: A better bound can be obtained if the tree is balanced: the running time is actually $O(k^2 h^2 n)$ if the tree has height h, since j and j'' only need to go up to h.)

Alternate Solution for (b):

Here is an alternate way to write pseudocode that completely avoids recursion:

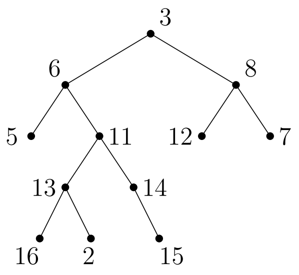
```
 1. first compute a postorder traversal
 2. for each node $v$ in postorder do {
 3.     if $v$ is a leaf then
 4.         for $i = 1$ to $k$ do
 5.             for $j = 0$ to $n$ do
 6.                 $F[v, i, j] = -\infty$
 7.         $F[v, 1, 0] = \infty$
 8.         break
 9.     let $v_1, v_2$ be the children of $v$
10.     for $i = 1$ to $k$ do
11.         $F[v, i, 0] = -\infty$
12.         for $j = 1$ to $n$ do
13.             $F[v, i, j] = \max\{F[v_1, i, j-1], F[v_2, i, j-1]\}$
14.             for $i' = 1$ to $i - 1$ do
15.                 for $j' = j - 1$ to $n$ do
16.                     $F[v, i, j] = \max\{F[v, i, j],\ \min\{F[v_1, i', j'],\ F[v_2, i - i', j - 1],\ j + j' + 1\}\}$
17.                 for $j'' = j - 1$ to $n$ do
18.                     $F[v, i, j] = \max\{F[v, i, j],\ \min\{F[v_1, i', j - 1],\ F[v_2, i - i', j''],\ j + j'' + 1\}\}$
```

```
19. let $r$ be the root
20. return $\min_{j=0,\ldots,n} F[r,k,j]$
```

The analysis is similar.

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).

Solution:

(a) The final answer we want is $\min_{m\in\{1,\ldots,n\}} C(1,n,m)$.

Base case. For each $i \in \{1,\ldots,n\}$, $C(i,i,i) = 0$.

Recursive formula. For each i,j,m with $1 \le i < m < j \le n$,

$$C(i,j,m) = \min_{m_1\in\{i,\ldots,m-1\},\ m_2\in\{m+1,\ldots,j\}} (C(i,m-1,m_1)+C(m+1,j,m_2)+|a_m-a_{m_1}|+|a_m-a_{m_2}|)$$

And for each i,j with $1 \le i < j \le n$,

$$C(i,j,j) = \min_{m_1\in\{i,\ldots,j-1\}} C(i,j-1,m_1) + |a_j - a_{m_1}|$$

$$C(i,j,i) = \min_{m_2\in\{i+1,\ldots,j\}} C(i+1,j,m_2) + |a_i - a_{m_2}|$$

Justification. Consider the optimal solution corresponding to $C(i,j,m)$, which is a tree rooted at a_m. Suppose we know that the left subtree has root a_{m_1} and the right subtree has root a_{m_2}. The left subtree should have inorder traversal $\langle a_i,\ldots,a_{m-1}\rangle$, and the right subtree should have inorder traversal $\langle a_{m+1},\ldots,a_j\rangle$. Thus, the optimal cost is $C(i,m-1,m_1) + C(m+1,j,m_2) + |a_m - a_{m_1}| + |a_m - a_{m_2}|$.
Exceptional cases are when $m = j$ (the right subtree is empty), where the optimal cost is $C(i,j-1,m_1) + |a_j - a_{m_1}|$, and when $m = i$ (the left subtree is empty), where the optimal cost is $C(i+1,j,m_2) + |a_i - a_{m_2}|$.
We don't know m_1, m_2 in advance, and so will try all choices and take the minimum.

(b) *Evaluation order.* Increasing order of j; and for subproblems with the same j, decreasing order of i.
(Increasing order of $j - i$ would also work.)

Analysis. There are $O(n^3)$ subproblems or table entries, indexed by (i,j,m). Each table entry can be evaluated in $O(n^2)$ time using the above recursive formula, since we take min over $O(n^2)$ choices of (m_1,m_2). Thus, the total running time is $O(n^5)$.

(Note: Actually, it can be improved to $O(n^4)$, by rewriting the formula as

$$C(i,j,m) = \min_{m_1\in\{i,\ldots,m-1\}} (C(i,m-1,m_1)+|a_m-a_{m_1}|) + \min_{m_2\in\{m+1,\ldots,j\}} (C(m+1,j,m_2)+|a_m-a_{m_2}|)$$

since the two parts can be minimized separately. This way, the loops for m_1 and for m_2 can be done separately. This idea is similar to the next solution in (c)–(d)...)

(c) The final answer we want is $\min_{m\in\{1,\ldots,n\}} (C'(1, m-1, m) + C'(m+1, n, m))$.

Base case. For each $i, p \in \{1, \ldots, n\}$, $C'(i, i, p) = |a_p - a_i|$.

Recursive formula. For each i, j with $1 \le i < j \le n$ and $p \in \{1, \ldots, n\}$,

$$C'(i,j,p) = \min \begin{cases} \min_{m\in\{i+1,\ldots,j-1\}} (C'(i, m-1, m) + C'(m+1, j, m) + |a_p - a_m|) \\ C'(i, j-1, j) + |a_p - a_j| \\ C'(i+1, j, i) + |a_p - a_i| \end{cases}$$

(Note: the expression can be simplifed, if we add boundary base cases for $C'(i, i-1, p)$.)

Justification. Consider the optimal solution T corresponding to $C'(i, j, p)$. Suppose we know that the root is $r(T) = a_m$. The left subtree should have inorder traversal $\langle a_i, \ldots, a_{m-1}\rangle$, and the right subtree should have inorder traversal $\langle a_{m+1}, \ldots, a_j\rangle$. Thus, the cost $c(T)+|a_p-r(T)|$ is $C'(i, m-1, m)+C'(m+1, j, m)+|a_p-a_m|$ (since $C'(i, m-1, m)$ accounts for the cost of the left subtree plus the edge of its root to a_m, and $C'(i, m-1, m)$ accounts for the cost of the right subtree plus the edge of its root to a_m, and these two parts are independent and can be optimized separately).
Exceptional cases are when $m = j$ (the right subtree is empty), where the optimal cost is $C'(i, j-1, j) + |a_p - a_j|$, and when $m = i$ (the left subtree is empty), where the optimal cost is $C'(i+1, j, i) + |a_p - a_i|$.
We don't know m in advance, and so will try all choices and take the minimum.

(d) *Evaluation order.* Increasing order of j; and for subproblems with the same j, decreasing order of i.
(Increasing order of $j - i$ would also work.)

Analysis. There are $O(n^3)$ subproblems or table entries, indexed by (i, j, p). Each table entry can be evaluated in $O(n)$ time using the above recursive formula, since we take min over $O(n)$ choices of m. Thus, the total running time is $O(n^4)$.