csci-5454-hw7

Jake Krol

October 2024

1 Question 1

1.1 Question 1a

maxWeightNotIncluding(v) is 0 for leaf nodes, since v must be excluded. For 1 child, the optimal solution is the max of $maxWeightIncluding(v_{child})$ or $maxWeightNotIncluding(v_{child})$. For 2 children, take the sum of optimal results for the left and right hand children of v. Where, the optimal result for each side is equivalent to the case for 1 child: $maxWeightIncluding(v_{child})$ or $maxWeightNotIncluding(v_{child})$.

$$maxWeightNotIncluding(v) = \begin{cases} 0 & \text{0 children (v is leaf node)} \\ \\ max(& \text{if non-leaf node with 1 child} \\ \quad maxWeightIncluding(v_{child}), & \\ \quad maxWeightNotIncluding(v_{child}) & \\ ) & \\ \\ max(& \text{if non-leaf node with 2 children} \\ \quad maxWeightIncluding(v_{left-child}), & \\ \quad maxWeightNotIncluding(v_{left-child}) & \\ ) & \\ + & \\ max(& \\ \quad maxWeightIncluding(v_{right-child}), & \\ \quad maxWeightNotIncluding(v_{right-child}) & \\ ) & \end{cases}$$

In all cases of ***maxWeightIncluding(v)***, v must be included, and the children of v must be excluded. The 0 children case is leaves only the root weight, w_v. For 1 child, the child node must be skipped since v must be included, and the remaining objective is equivalent to finding $maxWeightNotIncluding(v_{child})$

for the subtree rooted at v_{child}. Similarly for 2 children, the recurrence is the root w_v plus the $maxWeightNotIncluding$ solutions for both left and right children.

$$maxWeightIncluding(v) = \begin{cases} w_v & \text{0 children (v is leaf node)} \\ w_v + maxWeightNotIncluding(v_{child}) & \text{1 child} \\ w_v + maxWeightNotIncluding(v_{left-child}) + & \text{2 children} \\ maxWeightNotIncluding(v_{right-child}) & \end{cases}$$

1.2 Question 1b

The optimal solution for the full tree with root $r = v_0$ is

$$\mathbf{max(maxWeightNotIncluding(v_0), maxWeightIncluding(v_0) = max(9, 9.2) = 9.2}$$

$$\begin{aligned} maxWeightNotIncluding(v_0) = \\ max(maxWeightIncluding(v_1), maxWeightNotIncluding(v_1)) + \\ max(maxWeightIncluding(v_3), maxWeightNotIncluding(v_3)) = \\ 2 + 7 = 9 \end{aligned}$$

$$\begin{aligned} maxWeightIncluding(v_0) = \\ w_{v_0} + maxWeightNotIncluding(v_1) + maxWeightNotIncluding(v_3) = \\ 1.9 + 1 + 6.3 = 9.2 \end{aligned}$$

1.3 Question 1c

Since *maxWeightIncluding(v_i)* and *maxWeightNotIncluding(v_i)* depend on the subproblems of v_i's child nodes, the tree should be computed in a bottom-up fashion while storing the *maxWeightIncluding()* and *maxWeightNotIncluding()* values for each node in a table. The table's key is the node index and each key stores two values: *maxWeightIncluding(v_i)* and *maxWeightNotIncluding(v_i)*. For a parent node, lookup the children node's values in the table to avoid re-computing subproblems.

1.4 Question 1d

The bottom-up approach handles the recurrence base cases for *maxWeightIncluding(v_i)* and *maxWeightNotIncluding(v_i)* in $O(1)$. Similarly, the 1 child and 2 child(ren) cases only require looking up values of the children in the table, then performing a small number of sum and max operations, which are not a function of N. Therefore, the 1 and 2 child cases are also handled in constant time: $O(1)$. Since the runtime at each node is constant, the total runtime is $O(N)$, for sufficiently large N.

2 Question 2

2.1 Question 2a

The recurrence for $minPalindromeIns(s)$ considers three cases. First, if the string has 0 or 1 characters, then it is already a palindrome and return 0 since no characters need to be inserted. Second, if the first character, $s[0]$, matches the last character, $s[n-1]$, then this is already optimal. The new objective now is to find $minPalindromeIns(s[1] \dots s[n-2])$. Third, if $s[0] \neq s[n-1]$, then 1 character must be inserted at either the start or end. To do this, take the minimum insertions for either $s[1] \dots s[n-1]$ or $s[0] \dots s[n-2]$ and increment by 1 insertion.

$$minPalindromeIns(s) = \begin{cases} 0 & |s| = 0 \text{ or } 1(\text{ base case}) \\ \\ minPalindromeIns(s[1] \dots s[n-2]) & s[0] = s[n-1] \\ \\ 1 + min(& s[0] \neq s[n-1] \\ minPalindromeIns(s[1] \dots s[n-1]), & \\ minPalindromeIns(s[0] \dots s[n-2]) & \\ ) & \end{cases}$$

2.2 Question 2b

Algorithm 1 FILL-MEMO(s)

```
n = LENGTH(s)                                   ▷ size of array
A = [n, n]                                      ▷ initialize n by n table
A = FILL(A, 0)                                  ▷ Fill A with zeros
for i = 0 upto n − 1 do
    for j = i + 1 upto n − 1 do
        if s[i] = s[j] then
            A[i, j] = A[i + 1, j − 1]           ▷ lookup lower left
        else
            A[i, j] = 1 + min(A[i + 1, j], A[i, j − 1])   ▷ min(bottom, left) plus 1
        end if
    end for
end for
return A
```

The memo table, A, has the following properties:

- A has n rows and columns
- For $i > j$, $A[i, j]$ is not applicable; the lower left triangle can be ignored.

- For the diagonal $j = i$, $A[i, j]$ corresponds to the base case and is filled with zeros.
- For $i < j$, $A[i, j] = minPalindromeIns(s[i] \dots s[j])$ where i and j are start and stop character indices of the string, s.

After initialization, the remaining upper right triangle of A is filled by comparing the start and end characters. If $s[i] = s[j]$, then the characters match and $minPalindromeIns(s[i] \dots s[j]) = minPalindromeIns(s[i+1] \dots s[j-1])$, the lower left entry. If $s[i] \neq s[j]$, then the optimal solution is the minimum value at either the bottom or left adjacent entry plus 1. Since, the optimal solution requires adding 1 character to either the sub strings $s[i+1] \dots s[j]$ or the $s[i] \dots s[j-1]$.

Filling A requires $\Theta(\frac{(n-1)n}{2})$ if we consider initialization and zero-filling of the table with size n by n as $\Theta(1)$. Since, only the upper right triangle of an $n-1$ matrix remains to be filled. Or, the code could be written to loop over all i and j manually setting $j > i$ to some NA value and $i = j$ to 0. This would require $\Theta(n^2)$

2.3 Question 2c

The $minPalindromeIns$ solution for the full string $s[0] \dots s[n-1]$ is located the top right entry of the table: $A[0, n-1]$.