CS/ECE 374 P20

Pengxu Zheng, Jiawei Tang

TOTAL POINTS

20 / 100

QUESTION 1

1 20. 20 / 100

+ **100 pts** Correct

+ **60 pts** Correct Recurrence: English description (10) + final answer (10) + base cases (10) + recursive cases (30).

+ **40 pts** DP implementation details: data structure (10) + evaluation order (20) + running time analysis (10)

✓ **+ 10 pts** **English description: correct and clear English description of the variables/what the algorithm is computing. If this is missing, extra 20 points off (see below)**

+ **10 pts** Final answer: how to call your algorithm to get the final answer or which variable value (on which parameters) to return

✓ **+ 10 pts** **Correct base case(s)**

+ **30 pts** Correct recursive case(s). If recursive case is wrong, no credits for the DP implementation details.

+ **10 pts** Correct memoization data structure

+ **20 pts** Correct evaluation order

+ **10 pts** Correct running time analysis

+ **25 pts** IDK

+ **0 pts** Incorrect; Not understanding the question (see comments below)

- **20 pts** Uses code rather than pseudocode, contains wall of text, or is otherwise hard to read.

- **20 pts** Extra penalty for not having English description

+ **0 pts** Suboptimal running time. For c > 1, credit for a solution running in Omega(n^c) time is scaled by (130-20c)%. See manual point adjustment.

- **10 pts** Typos

Incorrect base case and recursion

HW Solution

CS/ECE 374: Algorithms & Models of Computation, Spring 2019 Version: **1.0**

Submitted by:

- **≪Jiawei Tang≫**: **≪jiaweit2≫**
- **≪Pengxu Zheng≫**: **≪pzheng5≫**

19

Solution:

20. We want to label all of the vertices and create an array M for memorization. We have a function $f(v)$ to find the maximum mass of the set under vertex v as we treat v as the new root of the tree. We want to find the answer $f(v)$. We have the base case: $f(v_1) = 0$ if v_1 is a leaf that doesn't have any child. Assume we can directly access child or grandchild or parent.

We have our recursive formula defined as, in the optimal solution,

$$\text{f(v)} = \begin{cases} \sum_{\text{all grandchildren j of v}} M[j] & \text{if not use v} \\ 1 + \sum_{\text{all child j of v}} & \text{if use v} \end{cases}$$

My plan is to traverse the tree in postorder, from bottom up. Every time we will check for current vertex's parent's mass to be either equal to 0 or larger than 0. The parent will be 0 by default. But once we decide to use the current vertex, we will temporarily set its parent to have a mass that is the same as current vertex's. The reason why I do this is once we decide to use the current vertex, we can't use the vertices that are on the same level of the current vertex and share the same parent with the current vertex. Assume that if a vertex doesn't have a grandchild or child, we set the value to be 0. Assume V is the set that contain all of the vertices when the root is set to be v.

$f(v)$:
Every element in M is initialized as 0
for all $v \in V$ in postorder **do**
  if $M[v.parent] == 0$ **then**
    if $1 + \sum_{\text{all child j of v}} > \sum_{\text{all grandchildren j of v}}$ **then**
      $M[v] = 1 + \sum_{\text{all child j of v}}$//Use v
      $M[v.parent] = M[i]$
    else
      $M[v] = \sum_{\text{all grandchildren j of v}}$
    end if
  else
    $M[v] = M[\text{a child of v}]$
  end if
end for

The final answer would be $f(v) = M[v]$ where v is given and is the new root the tree by assumption. We will have $O(n)$ subproblems, each with $O(1)$. Thus, the total time is $O(n)$.

1 20. **20 / 100**

+ **100 pts** Correct

+ **60 pts** Correct Recurrence: English description (10) + final answer (10) + base cases (10) + recursive cases (30).

+ **40 pts** DP implementation details: data structure (10) + evaluation order (20) + running time analysis (10)

✓ **+ 10 pts English description: correct and clear English description of the variables/what the algorithm is computing. If this is missing, extra 20 points off (see below)**

+ **10 pts** Final answer: how to call your algorithm to get the final answer or which variable value (on which parameters) to return

✓ **+ 10 pts Correct base case(s)**

+ **30 pts** Correct recursive case(s). If recursive case is wrong, no credits for the DP implementation details.

+ **10 pts** Correct memoization data structure

+ **20 pts** Correct evaluation order

+ **10 pts** Correct running time analysis

+ **25 pts** IDK

+ **0 pts** Incorrect; Not understanding the question (see comments below)

- **20 pts** Uses code rather than pseudocode, contains wall of text, or is otherwise hard to read.

- **20 pts** Extra penalty for not having English description

+ **0 pts** Suboptimal running time. For c > 1, credit for a solution running in Omega(n^c) time is scaled by (130-20c)%. See manual point adjustment.

- **10 pts** Typos

Incorrect base case and recursion