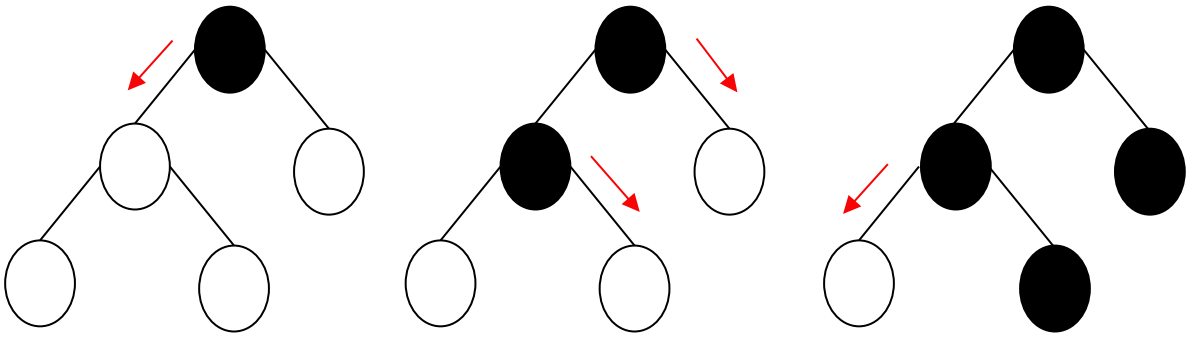
CSCI-GA.1170-003/004 Fundamental Algorithms April 2, 2020

Solutions to Problem 3 of Homework 7 (13 points)

Name: Rahul Zalkikar (rz1567) *Due: 8 pm on Thursday, April 2*

Collaborators: NetID1, NetID2

Suppose we need to distribute a message to all the nodes in a rooted (not necessarily binary) tree. Initially, only the root node knows the message. In a single round, any node that knows the message can forward it to at most one of its children. For example, the minimum number of rounds it takes to distribute the message in the tree given below is 3.



Note that the order in which the messages are distributed matters. For example, in the above tree, if the root node sends the message to the right child in the first round, then the number of rounds will be 4.

Assume that a tree T is given with nodes labeled $\{0, 1, 2, \ldots, n-1\}$ and the node 0 is the root of the tree. Further there is a two-dimensional $n \times n$ array $Child[][]$, where $k = Child[i][0]$ is the number of children of the node labeled i and $Child[i][1], Child[i][2], \ldots, Child[i][k]$ denote the labels corresponding to the children of node i. The remaining entries in the array are -1.

(a) (5 points) Let ROUNDS(i) be a function computing the minimum number of rounds it takes to distribute the message from node i to all nodes in the subtree rooted at i. Give a recursive formula to compute ROUNDS(i) and argue its correctness.

(**Hint**: Suitable subproblems would be the children. However, what is the order in which each of the children should get the message? Use the example to identify this.)

Solution:

ROUNDS(i) = 1 If node i has no children (or $Child[i][0] = 0$)

ROUNDS(i) = max(ROUNDS($Child[i][1]$, ...,ROUNDS($Child[i][k]$)) If node i has children (or $Child[i][0] > 0$) and only one maximum ROUNDS($Child[i][1...k]$)

ROUNDS(i) = max(ROUNDS($Child[i][1]$, ...,ROUNDS($Child[i][k]$))+1 If node i has children (or $Child[i][0] > 0$) and more than one maximum ROUNDS($Child[i][1...k]$)

If a node has no children we return 1 since there are no more nodes to distribute the message to. If a node has children and only one maximum of the minimum number of rounds it takes to distribute the message to its children's children then we return this max. If a node has

children and more than one maximum of the minimum number of rounds it takes to distribute the message to its children's children we return one plus this max. Thus, our formula will always return the minimum number of rounds it takes to distribute the message from a node i to to all nodes in the subtree rooted at i.

□

(b) (4 points) Design an efficient dynamic programming **top-down** recursive algorithm to compute ROUNDS(0).

Solution:

```
1 ROUNDS(i):
2     k = Child[i][0]
3     if k == 0:
4           return 1
5     max = 0
6     countMax = 0
7     for j = 1 to k:
8           if ROUNDS(Child[i][j]) > max:
9                 max =ROUNDS(Child[i][j])
10                countMax = 1
11          else if ROUNDS(Child[i][j]) == max:
12                countMax = countMax + 1
13    if countMax > 1:
14          return max + 1
15    else :
16          return max
```

□

(c) (4 points) Analyze the running time of your algorithm.

Solution:

The height of a tree with n nodes is $\log n$, and $k = Child[i][0] = O(n)$ for some node i. So, the run time of the algorithm is $O(n \log n)$. □