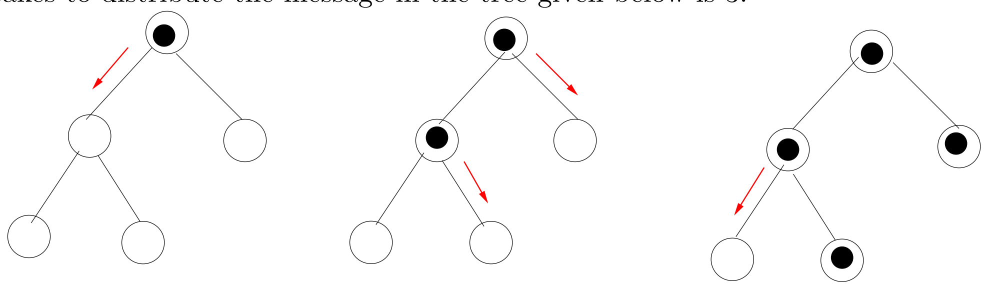
CSCI-GA.1170-001/002 Fundamental Algorithms	November 10, 2015

Solutions to Problem 3 of Homework 7 (8 (+6) points)

Name: GOWTHAM GOLI (N17656180) *Due: Tuesday, November 10*

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) (4 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) as a function of Rounds(j_1), …, Rounds(j_k), where $j_1, \ldots, j_k$ denote the children of node i.

HINT: What is the order in which each of the children get the message?

Solution:

For any leaf node i, we have

$$Rounds(i) = 0$$

For any non-leaf node i, let $j_{\pi_1}, \ldots, j_{\pi_k}$ be a permutation of $j_1, \ldots, j_k$ such that

$$\begin{aligned} Rounds(j_{\pi_1}) \geq Rounds(j_{\pi_2}) \geq \ldots \geq Rounds(j_{\pi_k}) \\ \implies Rounds(i) = Max\{1 + Rounds(j_{\pi_1}), 2 + Rounds(j_{\pi_2}), \ldots, k + Rounds(j_{\pi_k})\} \end{aligned}$$

$\square$

(b) (4 points) Write the pseudocode for the recursion with memorization dynamic programming procedure for computing ROUNDS(0).

Solution:

In the following psuedocode *Rounds* is an array of size n which holds the value of Rounds at every node in the tree and the base call is EVALROUNDS(0)

```
Algorithm: EvalRounds(i)
if Rounds(i) is not NULL then
    Return Rounds[i]
end
if Child[i][0] is 0 then
    Rounds[i] ← 0
    Return Rounds[i]
end
ChildrenRounds ← Vector(int)
for j ← 1 to n do
    if Child[i][j] is not -1 then
        Rounds[Child[i][j]] ← EvalRounds(Child[i][j])
        ChildrenRounds.Add(Rounds(Child[i][j]))
    end
end
DescendingSort(ChildrenRounds)
Rounds[i] ← −∞
for j ← 1 to |ChildrenRounds| do
    Rounds[i] ← Maximum(Rounds[i], j + ChildrenRounds[j])
end
Return Rounds[i]
```

Algorithm 5: Dynamic Programming Algorithm to calculate *Rounds*

□

(b) (4 points (**Extra credit**)) Analyze the running time of your algorithm.

Solution:

To evaluate $Rounds(0)$, we start the base recursive call from the root node 0 which recursively calls each of it's child and when the recursive call is returned back from it's child, store the returned value into the appropriate entry of *Rounds* (used for memorization) corresponding to the respective child.

Each recursive call visits every node exactly once. Therefore, this takes $O(n)$ time in total. Then we sort the *Rounds* of the node's children and then find the maximum of $j + ChildrenRounds[j]$. If every node i has k_i children, then the total time this step takes will be $\sum_{i=0}^{n-1} O(k_i \log k_i) + O(k_i)$ where $\sum_{i=0}^{n-1} k_i = n$

Therefore, the total time taken to evaluate $Rounds(0)$ will be $O(n) + \sum_{i=0}^{n-1} O(k_i \log k_i)$ where $\sum_{i=0}^{n-1} k_i = n$ □