Homework 2

Zhehao Wang 404380075 (Dis 1D)

Apr 10, 2016

1 Number of inversions for any permutation

2 Number of intersection and inversions

3 Celebrity iterative

4 Diameter of tree

(a)

Define the **height** of a rooted directed tree as the number of edges on the longest path from the root to a leaf. Algorithm is given in Alg 5.

Algorithm 1 Diameter of a rooted directed tree's underlying undirected tree, recursive

function FINDHEIGHTORDIAMETER(root, findHeight, prevRoot)
    if $degree(root) = 1$ **then**
        return 0
    $heights \leftarrow []$
    for each $\{n | n \in V, (n, root) \in E, n \neq prevRoot\}$ **do**
        $heights.push(1 + findHeightOrDiameter(n, true, root))$
    if $findHeight$ **then**
        return $max(heights)$
    else
        return $max(heights) + 2^{nd} highest(heights)$

This recursive algorithm takes in the root of a tree and produces the height of the tree, by each time removing the root and finding the maximum height among all resulting sub trees. The diameter of the tree would be the sum of the heights of two highest subtrees. Initial call to the algorithm should look like $findHeightOrDiameter(root, false, nil)$. This algorithm is $O(n)$, where n is the number of nodes in the tree, because each node in the tree will be visited exactly once.

(b)

The iterative version of the algorithm is given in Alg 6.

This algorithm is $O(n)$, where n is the number of nodes in the tree, because each node in the tree will be visited exactly once.

Algorithm 2 Diameter of a rooted directed tree's underlying undirected tree, iterative

function FINDHEIGHTORDIAMETER(root)
    $queue \leftarrow [root]$
    $height0 \leftarrow 0$
    $height1 \leftarrow 0$
    while True **do**
        $nodeCount \leftarrow queue.size()$
        if $nodeCount = 0$ **then**
            return $height0 + height1$
        $height \leftarrow height + 1$
        while $nodeCount > 0$ **do**
            $r \leftarrow queue.dequeue()$
            $r.visited \leftarrow true$
            if $degree(r) = 1$ **then**
                if $height > height0$ **then**
                    $height0 \leftarrow height$
                    $height1 \leftarrow height0$
                else if $height > height1$ **then**
                    $height1 \leftarrow height$
            else
                for each $\{n | n \in V, (n, r) \in E, n.visited = false\}$ **do**
                    $queue.enqueue(n)$
            $nodeCount \leftarrow nodeCount - 1$