CMP_SC: Homework 2
Due: 1:45 pm, September 20, 2016

Answers to the following problems must be handed in class. Unless otherwise stated, the running time of an algorithm is its running time in the worst case. You must explain why your algorithm runs in the stated time bounds in order to get full credit. There are a total of 20 points and will contribute towards 2% of the final grade.

1. Given a binary tree T and a simple path p of T from the root to a leaf, we shall call $\mathsf{deg\_2\_length}(p)$ to be the number of nodes in p that have degree 2. Let $p_1, p_2, \ldots, p_k$ be all the simple paths from the root of T to a leaf. For T, we shall call $\mathsf{deg\_2\_height}(T)$ to be the maximum number amongst the sequence $\mathsf{deg\_2\_length}(p_1), \ldots, \mathsf{deg\_2\_length}(p_k)$.

   For example, consider the tree T below. There are 3 paths from the root to a leaf: $p_1 = 2 \to 3 \to 5$, $p_2 = 2 \to 3 \to 7$ and $p_3 = 2 \to 6 \to 8 \to 1$. $\mathsf{deg\_2\_length}(p_1) = 2, \mathsf{deg\_2\_length}(p_2) = 2$ and $\mathsf{deg\_2\_length}(p_3) = 1$. Therefore $\mathsf{deg\_2\_height}(T) = 2$.

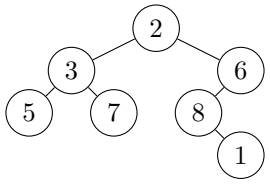


Figure 1: Binary tree T

   Consider the following computational problem.

   Input: A binary tree T.
   Output: $\mathsf{deg\_2\_height}(T)$.

   (a) Give a recursive algorithm that solves the above problem in $O(n)$ running time. (2 points)

   (b) Give a non-recursive algorithm that solves the above problem in $O(n)$ running time. (3 points)

   You may assume that the attribute $T.root$ is a pointer to the root node and that each node n of the tree has at least two attributes $n.left$ which points

to the left subtree of n, and $n.right$ which points to the right subtree. If you want to store more attributes, that will be fine, just explain what is the attribute stored in the node.

2. A sequence of integers $s = a_1, a_2, \ldots, a_k$ is said to be a *strictly increasing* sequence of length k if each number in the sequence is strictly smaller than the next one. For example, $1, 2, 3, 4$ is a strictly increasing sequence, $1, 2, 3, 2$ is not a strictly increasing sequence, and $1, 2, 3, 3$ is also not a strictly increasing sequence of numbers. Given a sequence of numbers $s = a_1, a_2, \ldots, a_k$ a sequence of consecutive elements $s_1 = a_{i+1}, a_{i_2}, \ldots, a_{a_{i_j}}$ from s is said to be a strictly increasing contiguous subsequence of s if s_1 is strictly increasing. For example if $s = 1, 2, 5, 3, 5, 6, 7$ then $1, 2, 5$ and $3, 5, 6, 7$ are both strictly increasing contiguous subsequences of s.

   Consider the following computational problem.

   Input: An array A of size n, with each entry in the array.
   Output: Array B of size n such that for each i, $B[i]$ is the length of the longest strictly increasing contiguous subsequence of the sequence $A[1], A[2], \ldots, A[i]$.

   For example, if $n = 11$, and A is the array

1	2	5	3	5	6	7	8	4	8	9

   then B is the array

1	2	3	3	3	3	4	5	5	5	5

   .

   Give an algorithm that solves the above computational problem in $O(n)$ running time. (3 points).

3. For this problem, we shall define the first-quartile of i integers $a_1, \ldots, a_i$ to be the $\lceil \frac{i}{4} \rceil$-th smallest number amongst them. For example, if the sequence of numbers is $5, 3, 4, 6, 7, 2, 8, 2$ then its first-quartile is 2.

   We want to solve the following computational problem.

   Input: An integer array A of size n.
   Output: Array B such that $B[i]$ is the first-quartile of the first i integers of A.

   For example, if A is the array

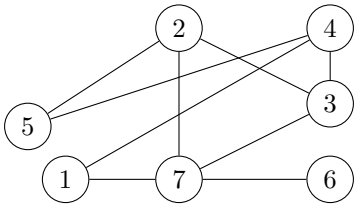
5	3	4	6	7	2	8	2

   then B is the array

5	3	3	3	4	3	3	2

   Give an algorithm that solves the above problem in $O(n \log n)$ running time. (5 points).

4. Consider the following undirected graph G:



(a) Give the adjacency matrix representation of G. (2 points)

(b) Give the adjacency list representation of G. (2 points)

(c) Give the distances and the BFS tree generated by running the Breadth First Search algorithm on G with 3 as the source vertex. (3 points)