CS21003 - Tutorial 1

August 3rd, 2018

1. Put the following functions in order from lowest to highest in terms of their Θ classes. (Some of the functions may be in the same Θ class. Indicate that on your list also.)
   (a) $f_1(n) = nlogn$
   (b) $f_2(n) = n^{3/2}$
   (c) $f_3(n) = 1000$
   (d) $f_4(n) = \sqrt{n}(n + logn)$
   (e) $f_5(n) = 3^n$
   (f) $f_6(n) = 2^{n+2}$
   (g) $f_7(n) = 0.00001$
2. Give an example of two positive real valued functions $f(n)$ and $g(n)$ of natural numbers that satisfy the property that $f(n)$ is not $O(g(n))$ and $g(n)$ is also not $O(f(n))$.
3. Assume that each node in a binary tree T contains only a positive integer value and two child pointers (left and right). No parent pointers or additional values can be stored in the nodes. Let r be the root of the tree, and v any node in the tree T. The weight of v is defined as the sum of all the values stored on the unique $r - v$ path. Your task is to locate the maximum of the weights of all the nodes in T in $O(n)$ time. Assume that all the values are positive.
4. The vertex set of a binary tree T on eight nodes is $\{a, b, c, d, e, f, g, h\}$. The inorder listing of the vertices of T is *bfdcgeha*, and the postorder listing is *fdghecba*. Reconstruct the tree T. Explain the relevant steps.
5. How many distinct binary search trees can be created out of 4 distinct keys? [**Note:** Try solving it for the general case of n keys and use that to find the solution for $n = 4$.]
6. **Prove or disprove:** You are given a sequence of integers $a_1, a_2, \ldots, a_n$ in an array. This can lead to a BST having the maximum possible height only if the integers are in sorted order.
7. You are given a sequence of integers $a_1, a_2, \ldots, a_n$ in an array. You need to decide whether inserting these integers in that sequence leads to a height of $n - 1$ of the binary search tree. Propose a worst-case $O(n)$-time algorithm to solve the problem. Note that if you actually build the tree, you end up in a $\Theta(n^2)$ running time in the worst case.
8. Let us define a *relaxed red-black tree* as a binary search tree that satisfies red-black properties 1, 3, 4 and 5. Thus, the root can be either *red* or *black*. Consider a relaxed red-black tree T whose root is red. If we color the root of T black but make no other changes to T, is the resulting tree a *red-black* tree?