Quiz 2 Practice Questions

Algorithmic Questions

Topic : Binary Search Trees

[BST1]
Suppose you are given a completely balanced binary search tree (a completely balanced tree means that each path from root to leaf is exactly the same). What is the time required to find the median of the elements of such a tree ?

[BST2]
Suppose you are given a completely balanced binary search tree (a completely balanced tree means that each path from root to leaf is exactly the same). What is the time required to find the max of the elements of such a tree ?

[BST3]
Suppose some one gave you a binary tree and claimed that it is a BST. Design and analyze an efficient algorithm to verify if it is indeed the case.

[BST4]
Given a BST T and two integers a and b, print all elements in T between a and b.

[BST5]
Suppose you are given two Binary Search Trees T_1 and T_2. Design an efficient algorithm to determine if they contain the same elements.

[BST6]
Design an algorithm to convert a *sorted* array to a *balanced* binary tree.

Red-Black Trees

[RBT1]
Insert the following numbers into an initially empty red-black tree: $8, 2, 4, 7, 5, 3, 1, 6$. (To solve this problem correctly, you will need to refer to the text-book for details about Red-Black trees in addition to the main ideas discussed in class).

[RBT2]

Design a “best-case” red-black tree with 10 nodes, i.e., a red-black tree with the shortest possible path from the root to a leaf.

[RBT3]

Design a “worst-case” red-black tree with 10 nodes, i.e., a red-black tree with the longest possible path from the root to a leaf.

[RBT4]

What is minimum possible number of nodes in a red-black tree which contains two black nodes from every root-to-leaf path?

[RBT5]

Suppose we define a red-black tree where along each path the number of black nodes is the same, and there cannot be more than three consecutive red nodes (but there may be two consecutive red nodes).

(a) What is the ratio of the longest possible path length to the shortest possible path length in this tree?

(b) For a tree that has b black nodes along each path, what ratio of the maximum possible number of nodes to the minimum possible number of nodes in the tree?

Binary Heaps

[Heap1]

Given a heap and a number k, design an efficient algorithm that outputs the top-k largest element from the heap. What is the running time of the algorithm? Can you design an algorithm that runs faster than $O(k \log n)$?

[Heap2]

Instead of binary heaps, suppose you had to implement ternary heaps (i.e., where each node has up to three children). Explain how you would implement such heaps using arrays, and how you can determine child and parent pointers. What are the advantages/disadvantages of ternary heaps over binary heaps?

[Heap3]

Can you use a binary search tree to simulate heap operations? What are the advantages/disadvantages of doing so?

[Heap4]

Design an algorithm to convert a given binary search tree into a heap efficiently. What is the running time of your algorithm

[Heap5]

Assume a heap is arranged such that the largest element is on the top. Suppose you are given two heaps S and T, each with m and n elements respectively. What is the running time of an efficient algorithm to find the 10^{th} largest element of $S \cup T$?

Union-Find

[UF1]

Can you use any of the previously studied data structures (e.g. heaps, red-black trees) for the Union-Find problem? Explain your answer.

[UF2]

What is the the worst-case performance of the Union-Find data structure we discussed (with union by rank and path-compression)?

[UF3]

Suppose we are working with a Union-Find data structure as described in class (assume no path compression is applied). Suppose we consider a set of 10 items that are eventually joined into a single set via 9 union operations

(a) Describe the sequence of union operations that will result in the root having the largest number of children. What is the final degree of the root in this case?

(b) Can you have a sequence of union operations that will result in the root having three children? Explain why or why not.

[UF4]

Suppose we are working with a Union-Find data structure as described in class (this time path compression is allowed for Find operations). Node z is at a distance of 4 from the root r of its tree (i.e. 4 edges away). We now perform a Find on z.

(a) By how much will the degree of r change? Explain your answer.

(b) By how much will the degree of z change? Explain your answer

[UF5]

Recall the definition of the "iterated logarithm function" $\log^*(n)$.

(a) Similarly, try and define the "iterated square root function" $sqrt^*(n)$.

(b) What is the value of $sqrt^*(2^3 2)$?

[UF6]

Determine the incorrect statement below concerning the Union-Find data structure we discussed in class (with union by rank and path-compression).

(a) The union operation sometimes may not increase the height of the resultant tree

(b) The union operation can at most increase the height of the resultant tree by one

(c) The union operation always increases the height of the resultant tree by one

(d) The find operation sometimes may not increase the degree of the root of the resultant tree

(e) The find operation requires two traversals from node to root