Data Structures and Analysis of Algorithms — Quiz 2
Sections 5 and 6 – Fadi Zaraket

Name:
Student ID:
Date:

Instructions and advice:

- *This is page 1. The exam has 11 pages*
- *Your time is limited, use it carefully. You have 120 minutes to finish this exam.*
- *Write your name, student id and today's date.*
- You are allowed to have two cheat sheets with your name on it. Photocopies are not allowed.
- The mark on each question is an estimate of how many minutes you should spend on the question.
- Read all questions before you start working on the exam. This will help you know where to start.
- Start with the easiest questions.
- If you feel stuck, you probably misunderstood the question. Read it again. Still stuck, ask for clarification.
- Don't leave early. Check your answers.
- Be confident and do not look around.
- Show what you are doing even if you can do it all in your head. It helps you get partial credit.
- Do not leave a question without an answer, otherwise you leave the grader no choice.
- For multiple choice questions, if you do not know the answer, eliminate the obviously wrong answers, then guess.

Part one. Binary search trees (15 pts.):

a. Write an algorithm that takes the root of a binary tree and returns whether the tree is a binary search tree or not.

b. Show a binary search tree whose preorder and inorder traversals generate the same result.

Part two. Answer four of the five questions below. (15pts.)

a. How many nodes can exist on level h in a binary tree?

b. How many different hash functions can we have to place n items in m positions where $n \leq m$?

c. We can preprocess and turn any binary search tree into a perfect binary tree in linear time. We saw how we can do that with the DSW algorithm. Why then learn and use red black balanced trees?

d. A node in a *full binary tree* has two children or zero children. How many leaves must we have in a full binary tree with n nodes?

Part three. Heaps (15 pts.)

Build a max heap out of the array $A = \{4, 1, 3, 2, 16, 9, 10, 14, 8, 7\}$ using the `BuildHeap` code shown below.

Show the state of the array after each call to `heapify`.

Remember that `heapify` takes a fresh element (pointed to by `i`) and two heaps (considered to be the sub-trees under `i`) and merges the two heaps using `BubbleDown`.

```
BuildHeap(A) {
    HeapSize = length(A);
    i = floor( length(A)/2);
    for(; i >= 1; i--)
        heapify(A,i)
}
```

Part four. Heaps and balanced search trees (15 pts.)

Describe how you would use a red-black balanced search tree to implement a heap; specifically you need to support INSERT and EXTRACT-MIN operations in time $O(\log n)$, where n is the number of items currently in the set.

The $O(\log n)$ complexity of these operations matches the complexity of these operations on a heap. Give three reasons why the heaps we studied in class are superior to the approach based on balanced binary search trees.

Part five. Hashing (8 pts.)

Describe the differences between chaining and open addressing for resolving collisions in a hash table. Stress the relative advantages of each approach.

Part six. Expected depth of BST (15 pts.)

Let S be a set of n random numbers. Let T be a binary search tree that represents S. Explain intuitively why $D(n)$, the depth of T, is expected to be $O(\log n)$. Derive a recurrence equation that computes the expected value of $D(n)$ and attempt to solve it. Hint: use a similar approach to the one we used with quicksort in class.

Part seven. Secret bid (10 pts.)

In a closed-bid auction (similar to ebay auctions), you submit your secret bid before the closed deadline. If your competitor knows your bid, he/she can easily win by bidding 1 dollar higher than your bid. You are worried that your competitors may have access to the data you submit before time.

Think of a solution that can guard bidders from competitors. You may change the rules as long as you keep the deadline fixed and the bid secret before the deadline. Hint: use hashing.

Part eight. Traversing and iterating (10 pts.)

Compare the running time of the in-order traversal of a binary search tree with calling minimum and then successively calling next until we reach the end of the tree. Explain your reasoning.

Part nine. Sorting with red-black balanced trees (10 pts.)

You need to read n numbers and then print them out in sorted order. Suppose you have access to a red-black balanced tree which supports the operations `search, in-order-traverse, insert, delete, minimum, maximum, successor` and `predecessor` each in $O(\log n)$ time.

- How can you sort in $O(n \log n)$ time using only insert and in-order traversal?
- How can you sort in $O(n \log n)$ time using only minimum, successor, and insert?
- How can you sort in $O(n \log n)$ time using only minimum, insert, delete

Part ten. BONUS (10 pts.)

- You are given 12 coins. One of them is lighter or heavier than the others. You can use three weighings. You have to identify the different coin.

- Insert 16 into the following red-black balanced tree. Assume you insert a new element using regular binary search insert first, color it red, then you fix the broken properties of the tree.