CMPS 2200 Introduction to Algorithms – Fall 14

9/9/14

2. Homework

Due **9/18/14** at the beginning of class

Remember, you are allowed to turn in homeworks in groups of two. One writeup, with two names.

1. **Min-Heaps (6 points)**
   (a) Where is the maximum element located in a min-heap? How can you compute it, and what is the runtime?
   (b) Can you give an example of a binary tree that is both a min-heap and a binary search tree? Can you find such an example for every $n > 0$, where n is the number of elements in the tree/heap?
   (c) Give a worst-case example of a min-heap that will cause **heapify_down(A,n,0)** to run in $\Omega(\log n)$ time. Justify your answer.

2. **d-Heaps (7 points)**
   A *d-ary min-heap*, d-heap for short, is the generalization of a binary heap to a d-ary tree. The tree still has to be almost complete, and for every child of a parent the child's value is greater or equal than the parent's value.
   (a) (2 points) Suppose a d-heap is stored in an array (that begins with index 0). For an entry located at index i in which location is its parent and in which locations are its children?
   (b) (1 point) What is the height of a d-heap that contains n elements? Give your answer in Θ-notation, and shortly justify why it is correct (no formal proof needed). The height should be a function of n and d.
   (c) (2 points) Shortly explain how the insertion procedure works for d-heaps (you do not have to give pseudocode). What is the runtime of inserting an element into a d-heap of n elements? The runtime should be a function of n and d.
   (d) (2 points) Shortly explain how the extract_min procedure works for d-heaps (you do not have to give pseudocode). What is the runtime in terms of n and d, where n is the number of elements in the heap?

FLIP OVER TO BACK PAGE $\Longrightarrow$

3. **Red-black tree rotations (6 points)**
   Find a sequence of numbers which, when incrementally inserted into a red-black tree, causes the following sequence of rotations:

   right, left, right.

   You may start with an initially non-empty tree, and you may insert numbers that do not cause any rotations. But there should not be any additional rotations performed.

   Draw the sequence of trees that you obtain after each insertion. For each such tree indicate the node that violates the red-black tree condition, indicate the nodes that participate in the rotation, the type of the rotation, and the subtrees that correspond to each other before and after the rotation.
   Hint: Use a red-black tree demo from the web.

4. **Rotation (4 points)**
   Give pseudo-code for a right-rotation as outlined in the figure below. Assume that a tree node is an object that stores the data, a pointer or reference to the left tree node, and a pointer or reference to the right tree node. Use `parent` as an initial reference into the tree, and rotate at its left child. Explain each step of your pseudo-code using pictures of the current state of the tree.