CSE 332 Data Abstractions, Winter 2013
Homework 2

Due: **Friday, January 25, 2013** at the BEGINNING of lecture. Your work should be readable as well as correct. You should refer to the written homework guidelines on the course website for a reminder about what is acceptable pseudocode. This assignment has **3**, Whoa!! THREE questions to go with a three day weekend! Have fun!

Problem 1. Binary Min-Heaps

This problem will give you some practice with the basic operations on binary min heaps.

(a) Starting with an empty binary min heap, show the result of inserting, in the following order, 13, 9, 3, 8, 5, 6, 14, 1, 12, 10, and 2, one at a time (using percolate up each time), into the heap. Be sure to draw the result after **every** insertion. By show here we mean draw the resulting binary tree with the values at "each node." In addition, give the array representation of your **final** answer.

(b) Instead of inserting the elements in part (a) into the heap one at a time, suppose that you use Floyd's algorithm. Show the resulting binary min heap tree. (It would help if you showed the intermediate trees so if there are any bugs in your solution we will be better able to assign partial credit, but this is not required). In addition, give the array representation of your final answer.

(c) Now perform two deleteMin operations on the binary min heap you constructed in part (b). Show the binary min heaps that result from these successive deletions ("draw the resulting binary tree with values at each node"). Be sure to draw the result after **every** deletion. In addition, give the array representation of your **final** answer.

Problem 2. Alternate remove() Algorithm for Heaps

As discussed in class, one way to remove an object from a heap is to decrease its priority value to negative infinity, percolate it up to the root of the heap, and then call deleteMin(). An alternative is to simply remove it from the heap, thus creating a hole, and then repair the heap.

(a) Write pseudocode for an algorithm that will perform the remove operation according to the alternative approach described above. Your pseudocode should implement the method call remove(int index), where index is the index into the heap array for the object to be removed. Your pseudocode can make calls to the following methods described in lecture: insert(), deleteMin(), percolateUp(), and percolateDown(). Like in lecture, you may assume that objects are just integer priority values (we will ignore the data associated with the priorities).

(b) What is the worst case complexity of the algorithm you wrote in part (a)?

Problem 3. d-Heap Arithmetic

Binary heaps implemented using an array have the nice property of finding children and parents of a node using only multiplication and division by 2 and incrementing by 1. This arithmetic is often very fast on most computers, especially the multiplication and division by 2 since these correspond to simple bitshift operations. In d-heaps, the arithmetic is also fairly straightforward, but is no longer necessarily as fast. In this problem you will figure out how the arithmetic works in those heaps.

(a) We will begin with considering a 3-heap (a heap where each node has ≤ 3 children. If a 3-heap is stored as an array, for an entry located at index i, what are the indices of its parent and its children? You may find it convenient to place the root at index 0 instead of 1 to simplify calculations (be sure to specify if you make this change).

Hint: the solution should be very concise. If it is becoming complicated, you might want to rethink your approach.

(b) Generalize your solution from (a) to work for d-heaps in general. If a d-heap is stored as an array, for an entry located at index i, what are the indices of its parent and its children?

(c) For what values of d will these operations be implementable with bit shifts instead of divisions and multiplications?

(d) If a d-heap has n nodes, what will its height be? (give an exact expression, not something in big O or theta etc.)

(e) If a d-heap has height h, what is the maximum number of nodes that it can contain? What is the mininum? (again, give an exact expression, NOT something in big-O or theta etc.)