CSE 332 DATA STRUCTURES
HOMEWORK 2

Due: Friday, Jan 20th at the beginning of class.

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) Now perform two deleteMin operations on the binary min heap you constructed in part (a). 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.

(c) 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.

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(). In the lecture pseudocode, the objects are the values themselves (rather than being objects that have a value field and a pointer to some other object of a different type); you may follow the lecture's example.

(b) What is the worst case complexity of this algorithm?

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 this corresponds to simple bitshift operations. In d-heaps, the arithmetic is also fairly straightforward, but is no longer necessarily as fast. In this problem you'll figure out exactly what this math is.

(a) If a d-heap is stored as an array, for an entry located at index i, what are the indices of the parent & 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's becoming complicated, you might want to rethink your approach.

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