CSI 33 LECTURE NOTES (Ojakian)

Topic 9: Heaps

OUTLINE
(References: Ch. 13)

1. Heaps
2. Priority Queue

1. Priority Queue

   (a) What it is: See interface on page 445.
   Note: We will ignore the "item" in enqueue.

   PROBLEM 1. *Determine the result of the dequeues:*

```
enqueue(5)
enqueue(2)
enqueue(3)
enqueue(9)

dequeue()
dequeue()
dequeue()
dequeue()
```

   (b) Examples of application: See page 444: Hospital Triage and Computer Operating System Tasks.

   (c) Problems implementing it as a list:

   i. If Enqueue at back, then Dequene is inefficient

   ii. If Enqueue in order, then Enqueue inefficient (finding spot could be fast, but may need to shift)

   PROBLEM 2. *Observe the above two inefficient approaches on the above example.*

   (d) What about Binary Search Tree?
   Only good as long as balanced...

   PROBLEM 3. *Consider the operation of a priority queue using a BST in the following example.*

```
enqueue(2)
enqueue(3)
enqueue(5)
enqueue(9)

dequeue()
dequeue()
dequeue()
dequeue()
```

(e) A response: Heaps! - key point: it will be forced to remain balanced.

2. <u>Heap</u>

   Note: Also called Binary Heap or Max Heap

   (a) Definition (Do via example on page 446):

      i. Binary Tree with ordered labels
      ii. It's complete
      iii. The value at any node is as large or larger than its children
      iv. Note: This is a **MAX heap**. Also exists **MIN heap**
      v.

         PROBLEM 4. *Draw examples. Which are Max Heaps and which are Min Heaps and which are neither?*

   (b) Insertion (do via example on page 448):

      First consider a problematic approach: Start at the top and move down to insertion spot similar to BST What could go wrong?

      i. Place the item at the bottom in the "next" spot
      ii. Swap it up the heap till it is correctly positioned

   (c) Pop Max (do via example on page 447):

      First consider a problematic approach: Successively move larger child up the tree to fill the deleted spot. What could go wrong?

      i. Remove the "last" node in the tree.
      ii. Replace the root (i.e. the max) with this node.
      iii. Swap it down the heap till it is correctly positioned

3. <u>Implementation</u>

(a) Difficulty in using our trees based on link structures: How do you deal with the "last" node.

PROBLEM 5. *Consider how you would find the last node in a linked-based binary tree.*

PROBLEM 6. *Suppose you tried to correct, by maintaining a pointer to the last node. Observe how the last node can change in somewhat complicated ways.*

(b) Alternative- Binary trees represented as an array (See Chapter 7, pages 229-230).

PROBLEM 7. *Draw a (non-complete) binary tree and give its array representation in two ways: starting at index 0 and starting at index 1.*

PROBLEM 8. *Based on the last example, determine formulas (in both cases) for finding the following: parent, left child, right child.*

PROBLEM 9. *Consider the following array: [NONE, 3, 4, 5, 7, 6, 7, 9, 8, 8]*

i. *Draw the binary tree represented by this array representation.*

ii. *What kind of data type have we drawn?*

PROBLEM 10. *Consider the BST we get by the following sequence of inserts: 1, then 2, then 3, then 4.*

i. *Draw the resulting BST*

ii. *Using a linked binary tree, how many Tree Nodes of memory are used?*

iii. *Draw the array representation of this tree.*

PROBLEM 11. *Compare the advantages and disadvantages of representing a binary tree as an array versus a linked structure?*

PROBLEM 12. *Program the binary tree using an array with a start index of 1 (for HW you will modify it to do a start index of 0).*

(c) **Program the Heap with insert and pop max!**