Advanced Data Structures

Chapter 16

Priority Queues

- Collection of elements each of which has a priority.
- Does not maintain a first-in, first-out discipline
- Elements are retrieved according to their priority.
- New items can be inserted in any order.
- Whenever an item is removed, that item has highest priority.
- Customary to give low values to high priorities
  - 1 denotes the highest priority
- Priority Queue extracts the minimum element from the queue.

Priority Queue Example

```
PriorityQueue<WorkOrder> q = new
                                    PriorityQueue<WorkOrder>();
q.add(new WorkOrder(3, "Shampoo carpets"));
q.add(new WorkOrder(1, "Fix overflowing sink"));
q.add(new WorkOrder(2, "Order cleaning supplies"));
```

Priority Queue

- When calling q.remove() for the first time, the work order with priority 1 is removed.
  - Fix overflowing sink
- The next call to q.remove() removes the next highest priority
  - Order cleaning supplies
- Standard Java library supplies a PriorityQueue class
- This is an abstract data type.
- You don’t know how it is implemented.

Heaps

- Binary tree with 2 special properties
  - It is almost completely full
    - All nodes are filled in except the last level may have some missing nodes toward the right
  - The tree fulfills the heap property
    - All nodes store values that are at most as large as the values stored in their descendants.
      - Smallest element is stored in the root.
- Difference between heaps and Binary Search Trees
  - The shape of the heap is very regular.
  - Binary Search Trees can have arbitrary shapes.
  - In the heap the right and left subtrees both store elements that are larger than the root
  - Binary Search Tree , smaller elements stored in the left subtree and larger elements are stored in the right subtree.

Insert New Element in Heap

1. Add a vacant slot to the end of the tree
2. Demote the parent of the empty slot if it is larger than the element to be inserted.
   a) Move the parent into the vacant slot
   b) Move the vacant slot up
   c) Repeat this demotion as long as the parent of the vacant slot is larger than the element to be inserted.
3. At this point, either the vacant slot is at the root or the parent of the vacant slot is smaller than the element to be inserted.
4. Insert the element into the vacant slot.

Removing the Minimum Value From the Heap

1. Extract the root node value.
2. Move the values of the last node of the heap into the root node
3. Remove the last node.
4. At this point the heap property may be violate for the root node, because one or both of its children may be smaller.
5. Promote the smaller child of the root node.
6. At this point the root node is ok.
7. Repeat this process with the demoted child.
   a) Promote the smaller of its children
   b) Continue until the demoted child has no smaller children

- Steps 5-7 are called “fixing the heap”

Heaps

- Advantage
  - Inserting and removing is efficient – O(log(n)) steps.
  - Because of the potential irregular shape of the binary search tree worst case insertion and removal are O(n) operations
  - Easy to use an array or array list to store the node values
    - Store the first layer, then the second, etc.
    - Leave the first element of the array empty.
    - The child nodes of the node with index I have index 2 *i and 2*i + 1
    - The parent node of the node with index i has index i/2.

Heapsort Algorithm

1. Insert all elements to be sorted into the heap
2. Keep extracting the minimum

- Efficiency O(n log(n)). Each insertion and removal is O(log(n)) and these steps are repeated n times.
- Can begin with simply inserting all the elements into an array and use the process of fixing the smallest the heap