Priority Queues and Heaps

Reading LDC Ch 18

Priority “Queue”

queue

priority queue
How to implement a Priority Queue?

- Keep them sorted!
  (Haven’t we implemented it already?)
  - Appropriate if the number of items is small

- Sorted Array-based implementation

- Linked List-based implementation

- Binary search tree implementation

---

But then... what kind of tree is this?
Heaps, MaxHeaps, MinHeaps

- A minheap is a complete binary tree in which each node’s element is less than or equal to both of its children
- A minheap keeps the smallest element readily available

- A maxheap is __________________________
- A maxheap keeps the ________ element readily available

Maxheap, Minheap, or Neither?

Maxheap: 110, 115, 331, 230, 307
Minheap: zebra, tiger, moose, panda, lion, koala
A minheap is a complete binary tree in which each node’s element is less than or equal to both of its children. A minheap keeps the smallest element readily available.

A maxheap is a complete binary tree in which ________ is less than or equal to both of its children. A maxheap keeps the _______ element readily available.

Which Data Structure to extend to create a heap?

Three primary operations for maxheaps:

MaxHeap extends BinaryTree

```java
package javafoundations;

public interface MaxHeap<T extends Comparable<T>> extends BinaryTree<T> {
    // Adds the specified object to the heap.
    public void add(T obj);

    // Returns a reference to the element with the highest value in the heap.
    public T getMax();

    // Removes and returns the element with the highest value in the heap.
    public T removeMax();
}
```
Adding an Element to a MaxHeap: example

- Insert new course: 349 in a maxheap!
- Strategy
  - Insert newItem into the bottom of the tree
  - “Trickle up” new item to appropriate spot in the tree

Removing the Max from a MaxHeap: example

- Remove old course: 331.
- Step 1: Delete (& remember to return) the item at the root
  - Results in disjoint heaps
- Step 2: Copy the item from the last node into the root, resulting in a “semiheap”
Heaps: heapify

- Step 3: Transform the semiheap back into a heap by “trickling down” the smallest-of-three element
  - Performed by the recursive calls to heapify

How many times do we need to call heapify?

11

Heap Implementations

- Using LinkedBinaryTree
- Using ComputedLinkArrayTree
- Which is better?
Can we use a Heap to sort?

- Algorithm:

- Efficiency?

- More efficient strategy
  - The second half of the array represents a bunch of (one-node) heaps
  - Use heapify() to fix the first-half nodes

(Max)Heapsort: example

Initial input array

Initial heap.  
Next: heapify(2)  
After heapify(2)

6 3 5 9 2 10

6
3
9 2 10

6
3
9 2 5

6
3
9
10

6
3
9
2
5

10
3
9
2
6

Final heap tree

Next: heapify(0)  
After heapify(1)

Efficiency?
Next: Use LinkedMaxHeap to implement PQs

```java
package javafoundations;

public interface MaxHeap<T extends Comparable<T>>
    extends BinaryTree<T>
{
    // Adds the specified object to the heap.
    public void add (T obj);

    // Returns a reference to the element with the highest value in
    // the heap.
    public T getMax ();

    // Removes and returns the element with the highest value in the
    // heap.
    public T removeMax ();
}
```

Since PriorityQueue implements Queue...

```java
package javafoundations;

public interface Queue<T>
{
    // Adds the specified element to the rear of the queue.
    public void enqueue (T element);

    // Removes and returns the element at the front of the queue.
    public T dequeue();

    // Returns a reference to the element at the front of the queue
    // without removing it.
    public T first();

    // Returns true if the queue contains no elements and false
    // otherwise.
    public boolean isEmpty();

    // Returns the number of elements in the queue.
    public int size();

    // Returns a string representation of the queue.
    public String toString();
}
```
/**
 * A data structure that works like a queue, but instead of
 * FIFO, always dequeues the item with the highest priority.
 * Uses a maxheap to store and sort items.
 */
package javafoundations;
import javafoundations.exceptions.*;

public class PriorityQueue<T> extends Comparable<T> implements Queue<T> {
    private LinkedMaxHeap<T> heap;

    public PriorityQueue() {
        heap = new LinkedMaxHeap<T>();
    }

    public T first() {
        return heap.getMax();
    }

    public boolean isEmpty() {
        return heap.isEmpty();
    }

    public int size() {
        return heap.size();
    }

    public String toString() {
        return heap.toString();
    }
}

public void enqueue(T element) {
    heap.add(element);
}

public T dequeue() {
    try {
        T temp = heap.removeMax();
        return temp;
    } catch (EmptyCollectionException ece) {
        System.out.println(ece);
    }
    return null;
}

{more...}