Heap notes

Prof Bill, Mar 2020

Wikipedia: Heap, the data structure, en.wikipedia.org/wiki/Heap_(data_structure)

Read: Sedgewick 2.4 Priority Queues, algs4.cs.princeton.edu/24pq

Animation: click on “Heaps”, www.cs.usfca.edu/~galles/visualization/Algorithms.html

---

Quickly:

- **Priority queues** are often referred to as "heaps", regardless of how they may be implemented.
- In a heap, the highest (or lowest) priority element is always stored at the root.
- A heap is not a sorted structure; it can be regarded as being partially ordered.
- Java's library contains a **PriorityQueue** class, which implements a min-priority-queue.

Heap is a binary tree where this **Heap property** is maintained:

> Each node is smaller than its children.

Priority Queue

Priority Queue ADT - Insert based on a user-specified priority rather than order of insertion like a regular, old queue. Operations include:

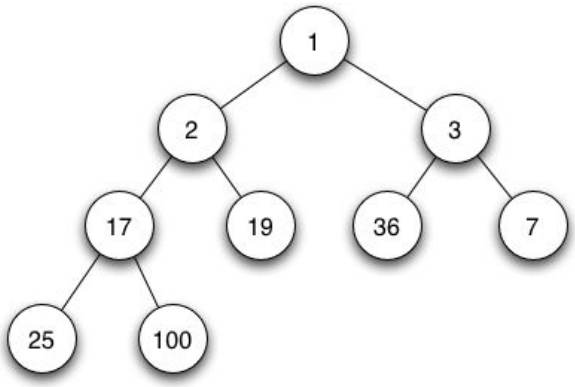
```
insert( item)

item removeMin()

boolean isEmpty()
```

For sure, we would like to insert() and removeMin() efficiently. Read on...

PQ is popular, tons of applications: www.geeksforgeeks.org/applications-priority-queue

Heap

Heap property (again) - Each node is smaller than its children.



Terms:

- ➔ **full binary tree** - every node other than leaves has two children
- ➔ **complete binary tree** - every level except the last is full, all nodes as far to the left as possible
- ➔ **binary tree depth** - num nodes from root to leaf; important: this is **log N** for a full or complete tree!

Term help: web.cecs.pdx.edu/~sheard/course/Cs163/Doc/FullvsComplete.html

JCF **PriorityQueue** holds **Comparable** objects. You can use a **Comparator** as well.
docs.oracle.com/javase/8/docs/api/java/util/PriorityQueue.html

In JCF, flip from min to max in Comparator for PQ to get removeMax().

Heapsort - add items to heap, then removeMin() them for sorted order.

Some pseudocode:

```
// insert item into the heap
insert( item)
        add item as next leaf node
        while heap property is not met
                swap node with parent

// remove and return the smallest item from the heap
item removeMin()
        minItem = root
        put last leaf as root        // restore heap
        while head property is not met
                swap node with smallest child
        return minItem
```

Heap performance is **log N** (excellent!):

- insert is O(log n) because the tree depth is log n.
- removeMin is O(log n)... the min is right there, O(1), but you have to swap and restore the heap, which is O(log n)

Heap as an array

The magic: store your heap as an array!

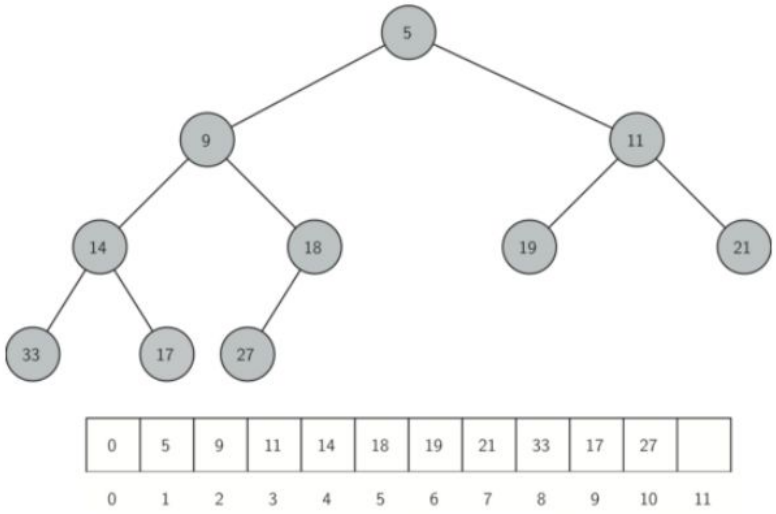
- ➢ This works because it is a complete binary tree

From a tree, fill the array top to bottom, left to right.

Array representation for Page 2 example:

1	2	3	17	19	36	7	25	100

Another one:



Source: stackoverflow.com/questions/38626816/how-to-check-if-array-is-a-min-heap

Yup - use an ArrayList.

(cool) Equations for accessing nodes in array of a heap:

root of tree = A[0]
parent of node A[k] = A[(k-1)/2]
left child of node A[k] = A[2k+1]
right child of node A[k] = A[2k + 2] // left child + 1