**Priority Queue ADT**

Ensures only the most important things are output/processed.

The highest priority items are always at the front of the queue.  
ie highest priority = 0  
Can have even negative (-3) priority for emergency/jump queue items.

**Entry**

An entry in a priority queue is simply a key-value pair.  
Priority queues store entries for efficient removal/insertions.

**Reflexive** property: X <= X  
-A key is always less or equal to itself

**Antisymmetric** property: X <= Y ^ Y <= X then X = Y

**Transitive** Property: X <= Y ^ Y <= Z then X <= Z

In a comparable context, all 3 of these must hold.

**Comparator ADT**

You must provide compare(x,y)

We must make sure its ordered in some way.

**Priority Queue Sorting**

-added and removed in order.

Unsorted List  
Insertion: O(1)  
RemoveMin + min: O(n)

Sorted List:  
Insertion: O(n)  
RemoveMin + min: O(1)

**Heap ADT**

Only remove at root, and heap order constraints, we get O(logn)  
  
We keep track of the: Root, last node, insert Node

A heap storing n keys has a height O(logn)

-wattsapp msg

**Heap-Sort**

**Array List-based heap implementation**

-minimize auxiliary space we need.

In the array, from the parent at index i, the index of the:  
left node: 2i  
right node: 2i+1

**Merging two heaps.**

Given 2 heaps, and a key k.  
Create a new heap with root storing k.  
Then have the 1 subtree as the left node, and the other as the right.  
Then we perform downheap to restore the heap-order property.