A priority queue is a data structure of items with keys which supports two basic operations: insert a new item and remove the item with the largest key.
public class PQ_arr<E> {
    private int curSZ; private E theQueue[];
    PQ_arr (int maxSZ) {
        theQueue = (E[])(new Object[maxSZ]); curSZ = 0;
    }
    PQ_arr (E items[], int nitem) {
        theQueue = items; curSZ = nitem;
    }
    
    public boolean isEmpty () { return (curSZ == 0) ; }
    public void insert (E item) {
        theQueue[curSZ++] = item; return ;
    }
    // ... ... // some code omitted
} // end of [PQ_arr]
public class PQ_arr<E> {
    // ... ... // some code omitted
    interface Compare<E> {
        public int f(E x, E y);
    }
    public E delMax(Compare<E> cmp) { // [delMax] is O(n)
        if (curSZ == 0) return null;
        int i0 = 0; E item0 = theQueue[0];
        for (int i = 1; i < curSZ; i += 1) {
            E item = theQueue[i];
            int sgn = cmp.f(item, item0);
            if (sgn > 0) { i0 = i; item0 = item; }
        } // end of [for]
        exch(theQueue, i0, --curSZ); return item0;
    } // end of [delMax]
} // end of [PQ_arr]
Given a total ordering, a binary tree is **heap-ordered** according to the ordering if each path in the tree is ordered according to the ordering.
A heap is a set of nodes with keys arranged in a complete heap-ordered binary tree, represented as an array.

Suppose that a heap $H$ is represented as an array $A$. For each node $n$ in $H$, if $n$ is stored at $A[k]$ for some $k$, then the left child of $n$, if it exists, is stored at $A[2 \times k]$ and the right child of $n$, if it exists, is stored at $A[2 \times k + 1]$.

Note: following the convention, the index of $A$ starts from 1 (instead of 0).
private void swim (int k)
{
    while (k > 1 && less (k/2, k))
    {
        exch (k, k/2); k = k/2;
    } // end of [while]
    return ;
} // end of [swim]
private void sink (int k, int N) {
    int j = 2*k;
    while (j <= N) {
        if (j < N && less (j, j+1)) j += 1;
        if (!less (k, j)) break ;
        exch (k, j);
        k = j; j = 2*k;
    } // end of [while]
} // end of [sink]
Let us go over the Java code!
The End