Priority Queue Operations:

Array pq[ ] holds the elements.

Field size stores number of elements that are currently stored. size ≤ pq.size - 1. pq[0] is unused.

Insert (x) / Add (x) RT = O(log n).

```
if size = pq.size - 1 then
    resize()  // utility function to resize array
              // when it is full
size++
pq[size]  ← x

PercolateUp (size)
```

PercolateUp (size)

```
PercolateUp (i)   // pq[i] may violate heap order
                  // with its parent
    pq[0] ← pq[i]   // saves edge cases
    while (pq[i/2] > pq[0]) do
        pq[i] ← pq[i/2]
        i ← i/2
    pq[i] ← pq[0]
```

DeleteMin ( ) / Remove ( ) // RT = O(log n)

```
min ← pq[1]
pq[1] ← pq[size--]
percolateDown (1)
Return min
```

PercolateDown (i)

```
PercolateDown (i)  // pq[i] may violate heap order
                   // with its children
    x ← pq[i]
    while (2*i <= size) do
        if (2*i = size)   // one child
            if (x > pq[size]) then
                pq[i] ← pq[size]
                i ← size
            else break;
        else   // 2 children
            if (pq[2*i] <= pq[2*i+1]) then
                schild ← 2*i
            else schild ← 2*i+1
            if x > pq[schild] then
                pq[i] ← pq[schild]
                i ← schild
            else
                break
    pq[i] ← x
    return
```