**Priority Queue**

A priority queue is one in which the maximum value is always on top. Thus, if we store the elements in a heap and use a max heap, we can get the top element in linear time.

There are a few operations that can be performed on priority queues. First, extraction of the maximum value.

extractMax (A, &heapSize)  
 if heapSize<1 *//error* else  
 max = A[1];  
 A[1] = A[heapSize];  
 heapSize--;  
 maxHeapify(A, 1);  
 return max;

PSEUDOCODE

This operation has a time complexity of O(log n) since it performs maxHeapify on the root one time.

Next, changing the value of a key at index i. Notice that, if the value decreases, we simply need to perform maxHeapify on that node. If the value increases, we need to keep going up, checking against parents.

heapChangeKey (A, i, key)  
 if key<A[i]  
 A[i]=key;  
 maxHeapify(A, i);  
 else  
 A[i] = key;  
 while (A[i]>A[parent(i)] && i>1)  
 swap(A[i], A{parent(i)];  
 i = parent(i);

PSEUDOCODE

Lastly, we look at inserting a key. Here, we can use a trick. We can set a very low value, and then use the existing heapChangeKey function.

Insert (A, key)  
 heapSize++;  
 A[heapSize] = -∞;  
 heapChangeKey (A, heapSize, key);

PSEUDOCODE