CSCI 104L Lecture 18: Priority Queues

In a Priority Queue ADT, you may perform the following operations:

- Add an item (with a priority)
- Return the item of highest priority
- Delete the item of highest priority

Our class signature will probably look something like this:

```cpp
template <class T>
class PriorityQueue {
    void add(const T& item);
    T peek() const;
    void remove();
    bool isEmpty();
    void changePriority(T& data);
};
```

**Question 1.** What would be the runtime of add/peek remove, using a...

- unsorted array / linked list?
- sorted array?

![Figure 1: A tree with values in the nodes. Consider this figure when answering the following questions.](image)

**Question 2.** What kind of tree is this?

**Question 3.** Do we really need to store it as a tree, or is there a more compact representation?

**Question 4.** If we are at index \(x\), which index is “above” it in the tree?

**Question 5.** If we are at index \(x\), which index is “below it to the left”? “below it to the right”?

We say that a tree has the **heap property** if, for each item in our “tree,” it will have higher (or equal) priority to anything below it.
Here are some of the functions within the PriorityQueue class:

T PriorityQueue::peek() const {
    return a[0];
}

void PriorityQueue::add(const T& data) {
    a[size] = data;
    bubbleUp(size);
    size++;
}

void PriorityQueue::bubbleUp(int pos) {
    if (pos > 0 && a[pos] > a[(pos-1)/2]) {
        a.swap(pos, (pos-1)/2);
        bubbleUp((pos-1)/2);
    }
}

void PriorityQueue::remove() {
    a.swap(0, size-1);
    size--;
    trickleDown(0);
}

void PriorityQueue::trickleDown(int pos) {
    int child = 2*pos+1;
    if (child < size) {
        if (child+1 < size && a[child] < a[child+1]) child++;
        if (a[child] > a[pos]) {
            a.swap(child, pos);
            trickleDown(child);
        }
    }
}

Question 6. What is the runtime for each of the following PriorityQueue operations?:

- Peek
- Add
- Remove

Question 7. In a priority queue, we generally insert, peek, and remove each element exactly once. A sequence of $n$ inserts, peeks, and removals. What is the overall time for a Priority Queue when using a:

- Unsorted Array/Linked List
- Sorted Array
- Heap
We can also use a heap to sort an array. The idea is to insert each element of the array into a heap, and then use that to find the smallest element that has yet to be placed in its correct spot.

```c
void HeapSort(int *a, int size) {
    Heap h;
    for (int i = 0; i < size; i++) {
        h.add(a[i]);
    }
    for (int i = 0; i < size; i++) {
        a[i] = h.peek();
        h.remove();
    }
}
```

**Question 8.** What’s the runtime of Heapsort?

**Question 9.** Is it stable?