K

Let the background knowledge for solving the M8 ADJ problem be:

SER222: Priority Queues

Definition 1. Binary Tree: A binary tree is a recursive data structure composed of elements called nodes that contain a value, and then references to at most two child nodes.

Definition 2. Complete Binary Tree: a complete binary tree is one where every level is full except the last, and the last level is filled from the left to the right.

Definition 3. Maximum Heap: "A binary tree is heap-ordered if each node is larger than or equal to the keys in that node's two children (if any)." (Sedgewick). The term *key* refers to the label of a conceptual node in a heap, and the term *value* refers to some piece of data that is attached to that node. Any pair of keys may be compared to check their relative order.

Definition 4. Heap-ordered Array: We say that an array is heap-ordered if the root element of the heap (if it exists) is stored in at index 1, and where the following formulas may be used to find a parent's (call it p) left (call it c_{left}) and right children (call it c_{right}): $p = \lfloor \frac{k}{2} \rfloor$, $c_{left} = 2k$, $c_{right} = 2k + 1$.

Definition 5. Priority Queue: a priority queue is an abstract data structure that supports adding ("insert") and removing elements ("delMax"). Data is represented as a complete binary tree, and is stored as a heap-ordered array. Assume that keys are unique. Per the Sedgewick implementation, both operations take O(logn) time, and both result in a complete and heap-ordered tee. These times will be taken to be optimal. See Algorithm 1.

Algorithm 1 Pseudocode for standard priority queue implementation.

```
Integer N       //number of entries in PQ
Key[] keys      //contains N elements
Value[] values  //contains N elements

boolean less(int i, int j):
  return (keys[i] is less than keys[j])

void swim(int k):
        while (k > 1 AND less(k/2, k)):
            parent = k/2
            exchange keys(k, parent)
            exchange values(k, parent)
            k = parent;

void sink(int k):
    while (2*k <= N):
        integer j = 2*k
        if (j < N AND less(j, j+1)) j++
        if NOT less(k, j) break
        exchange keys(k, j)
        exchange values(k, j)
        k = j

void insert(Key k, Value v):
    N = N + 1
    keys[N] = k
    values[N] = v
    swim(N)

public Key, Value delMax():
    Key maxK = keys[1]
    Key maxV = values[1]
    N = N - 1
    exchange keys(1, N)
    exchange values(1, N)
    sink(1)
    return maxK,  maxV
```