Name: Georges Hatem

CS 590 Homework 4: Priority Queues and Heaps Creativity Exercises

Due Date: February 20, 2022

Problem 5.7.24:

In this problem, root of tree contains the minimum key, so for every single time we have to make a comparaison between our desired target value and the root value. If the desired key is greater than the key in the root we perform the pop out operation and report that key in the root. In order to end the loop, we have to see that either there are no nodes at all or key in the root is greater than the target key. Self-restructuring of heap will allow the root value to be minimum for the whole time being. For any ideal case like each node is having only right child in the tree every pop out operation right child of the root is always the new root of the tree. This gives us O(1) time. We assumed that we have to report k numbers. Hence, the running time should be O(k)

The Algorithm is below:

Algorithm HeapTreeSearch(key, v)
Input: The key of the Heap Tree is the key we need to use to compare (the one that was given in the Exercise) and v is any node in the Heap Tree
Output: The elements less than the key in any node v in the Heap Tree will be printed

```
If v is an external node then
    return true
else if (node.element <= key) then
    print(node.element)
    HeapTreeSearch(key, v.leftChild)
    HeapTreeSearch(key, v.rightChild)
```

O(k) is the running time for this Algorithm. The reason for this is that there is no node in tree, which stores a key larger than key that has descendants storing a key less than key.