Name: Georges Hatem

# CS 590 Homework 4: Priority Queues and Heaps Application Exercises

Due Date: February 20, 2022

## Problem 5.7.29:

## Describe a system that can process upgrade requests and cancellations in O(log(n)):

To process upgrades and request cancellation in O(log(n)), the best case scenario is to use a Heap Tree.

In this Heap Tree, each node represents a frequent flyer and his or her priority flyers. The key in the Heap Tree will be the priority flyer status so that the tree would start from low priority to high priority and the element associated with each priority node is the name of the frequent flyer. Any user that cancels his or her upgrade request will be removed from the Heap Tree following the analogy used in Section 5.4. Also, any frequent flyer that upgraded will be inserted into the Heap Tree based using the same analogy in Section 5.4.

**As described in Section 5.4, insertion, removal, and upgrading running time for Heap Trees is O(log(n)). Below is the Algorithm that should be used in this Exercise for insertion:**

**<u>For Insertion:</u>**

**Algorithm HeapInsert(k,e):**
**Input: k represents the key, which is in this case the priority flyer status and e represents the element, which is the name of the frequent flyer in this case.**
**Output: A node (k,e) will be inserted in the Heap Tree**

**n ← n + 1**
**A[n] ← (k,e)**
**i ← n**

**while i > 1 and A[[i/2]] > A[i] do**
**Swap A[[i/2]] and A[i]**
**i ← [i/2]**

**and can determine the k highest-priority flyers on the waiting list in O(klog(n)) time, where n is the number of frequent flyers on the waiting list.**

To determine, the k highest-priority flyers on the waiting list, we need to search for them from the Heap Tree. As stated above, the Heap Tree is structured based on the priority flyer status. The Heap Tree is shaped from lower priority frequent flyers to higher priority frequent flyers. To search for higher priority frequent flyers, we need to traverse the whole tree since the higher priority frequent flyers are on the bottom of the tree. The running time of this process is O(log(n)). Choosing each frequent flyer will take O(1). Since there is n frequent flyers, this will take O(n). So, choosing all frequent flyers will take O(n). So, the whole process will take O(nlog(n)). We are choosing only k frequent flyers with the highest priority

for 1st class seats. **Therefore, the running time of the determination of k flyers with highest priority status will be O(klog(n)).**