# Michael Chillemi

# 07/02/2023

# CS 590 - Algorithms

# M4.B4:Programming Assignment Solution Description

I needed to first construct some helper classes in order for this solution to function. A frequent flyer class is the first class I developed. The information about each frequent flyer on the waiting list will be stored in this class. The status for this class is an enumeration and can be super, platinum, gold, or silver. The class will also keep track of additional data including the name of the frequent flyer, their confirmation number, and their current wait time. In order to compare one frequent flyer to another, I also had to build functions for equality, greater than, and less than.

I must create a max heap class, which symbolizes a priority queue, as the following assistance class. A size, a maximum size, and finally a frequent flyer vector will all be stored in this class. GetParent, getLeftChild, and getRightChild are a few of the helpful methods I created for this class. The output of these methods is an int that represents the index of the associated function. I also made the isNodeALeaf function, a convenience method that returns true if the node that is passed in is a leaf. In order to make it easier to switch components in my max heapify, remove max, and insert functions, I also developed a swap method. A function called max heapify is made to take a location and produce a max heap binary tree, which is the vector kept in the class. The remove max function, which I had to write, had the responsibility of removing and returning the node with the highest priority. Once the root has been eliminated, max heapify will be called to arrange the heap in the correct order. My most recent function was a straightforward insert method for adding an element to the priority queue.

The pseudo code for maxHeapify and removeMax will be provided below.

Algorithm maxHeapify(int pos):

Input: pos

If isNodeALeaf(pos)

return

If priorityqueue[pos] < priorityqueue[getLeftChild(pos)] || priorityqueue[pos] < priorityqueue[getRightChild(pos)]

If priorityqueue[getLeftChild(pos) > priorityqueue[getRightChild(pos)]

swap(pos,getLeftChild(pos) )

maxHeapify(getLeftChild(pos))

else

swap(pos,getRightChild(pos))

maxHeapify(getRightChild(pos))

Algorithm removeMax():

Output: Max Priority Frequent Flyer

returned \_flyer

If size > 1 && size < priorityqueue.size()

returned \_flyer.swap( priorityqueue[0])

priorityqueue[0].swap(priorityqueue[--size])

maxHeapifiy(0)

Return returned \_flyer

The last class I created was the NewUnknownAirlineUpgradeSystem class. This class will represent the airline processing system. This class will contain a priority queue for the waiting list and a returned passengers list. Which is the list our algorithm will populate of k highest priority frequent flyers that will get on the flight. One more variable we have a cancel counter to keep track of the amount of cancellations. In this class there are two main functions: insertRequests and systemUpgrade. The insertRequests method is a basic function I created to take an unorganized list of people on the waiting list and convert it to a priority queue. The next function systemUpgrade is the function where i processes the upgrade and cancelation requests. The way this function works is that it will take in the number requests and calculations. A while loop will be implemented and in that while loop we have three basic checks. The first is if the highest priority flyer list size is equal to the number of upgrade requests and the cancel counter is equal to the number of cancellations then it will break out of the loop indicating we met all requirements. In the next if statement it will check if the cancel counter is less than the number of cancellations. If so then we will call remove max to remove the root of the list. The last if statement checks if the size of the highest priority flyer list size is less than the number of upgrade requests. If so then the function will call remove max to remove the root node and add it to the k highest priority frequent flyer list.

The pseudo code for systemUpgrade will be provided below.

Algorithm systemUpgrade(request\_num, cancellation\_num):

while (true)

If highest\_priority\_frequent\_flyer\_list.size() == request\_num && cancel\_counter == cancellation\_num

Break;

If cancel\_counter < cancellation\_num

waiting\_list.removeMax()

Cancel\_countrer++

If highest\_priority\_frequent\_flyer\_list.size() < request\_num

highest\_priority\_frequent\_flyer\_list.emplace\_back(waiting\_list.removeMax())

As for my test cases which all are found in the main.cpp file I wrote out 4 test cases. The first two are a copy of the test cases that Venkata provided in the email. This is to prove that those test cases work with my algorithm. The other two test cases I came up with myself. The expected output will be provided below.(Run the NUASystem.exe in the command terminal to view the results)

$ ./NUASystem.exe

\*\*First Test\*\*

Highest-Priority Flyers:

Flyer Name: Jess

Confirmation Number: 1

Flyer Status: 2

\*\*Second Test\*\*

Highest-Priority Flyers:

Flyer Name: Kyle

Confirmation Number: 5

Flyer Status: 3

Flyer Name: Sammy

Confirmation Number: 1

Flyer Status: 2

\*\*Third Test\*\*

Highest-Priority Flyers:

Flyer Name: Mark

Confirmation Number: 5

Flyer Status: 3

Flyer Name: Andrew

Confirmation Number: 4

Flyer Status: 1

Flyer Name: Dan

Confirmation Number: 1

Flyer Status: 1

\*\*Forth Test\*\*

Highest-Priority Flyers:

Flyer Name: David

Confirmation Number: 5

Flyer Status: 3

Flyer Name: Michelle

Confirmation Number: 4

Flyer Status: 1