CMPT 225 Assignment 3 Write Up

Implementation method to create a PQ

In my implementation method, I combined the AVL tree given from the author's code and a min heap. In addition, I created a node class in my PQ.h. Moreover, I used the friend class to keep track of my PQ IDs and AvlTree IDs in sync. This method of implementation seemed the most logical way of implementing a priority queue as we learned in class throughout the term. Additionally, I used this method of implementation since the professor was talking about it in class.

PQ() and constructor PQ()

First I initialized my priority queue by setting the size to 0. I additionally resized my heap since arrays usually start at size 0 but to make the implementation of other functions easier I resized mine to start from 1 as seen in the code. In the constructor, I made sure I was inserting like this, the ID of the task itself then priorities (tasks [i], priorities[i]).

Functions in PQ class

Functions I tested for, insert (x,p), findMin(), deleteMin(), isEmpty(). int size(), and makeEmpty(). The first thing I did was copied the AVL tree from the previous assignment (authors code). I modified the insert function within the AVL tree so that it would work with my priority queue class. After creating the constructors in PQ I worked on the insert function. In my first test, I inserted in ascending order to see if my insert was working properly in which it did. I also made sure my simple size(), and makeEmpty() functions were working properly as well. I did this by first checking the size of the array when the first insertions of IDs and priorities were completed, then I deleted the whole array and checked for the size after the deletion of the array. As seen in my test program it works.

The second thing I tested for was my findMin() and deleteMin() functions. I did this by inserting random IDs and priorities into the array. Then after insertion, I performed findMin() and then printed out the index, priority, and values. I then performed deleteMin(), then performed another find min to see if the minimum value was deleted. As seen in my PQdemo.cpp it works.

The third thing I tested for was my percolate down and up functions. I did this by inserting random items and priorities into my PQ. I then checked the size. Then deleted it twice, checked the min value, and then finally checked the size once again. As seen in my PQdemo it works.