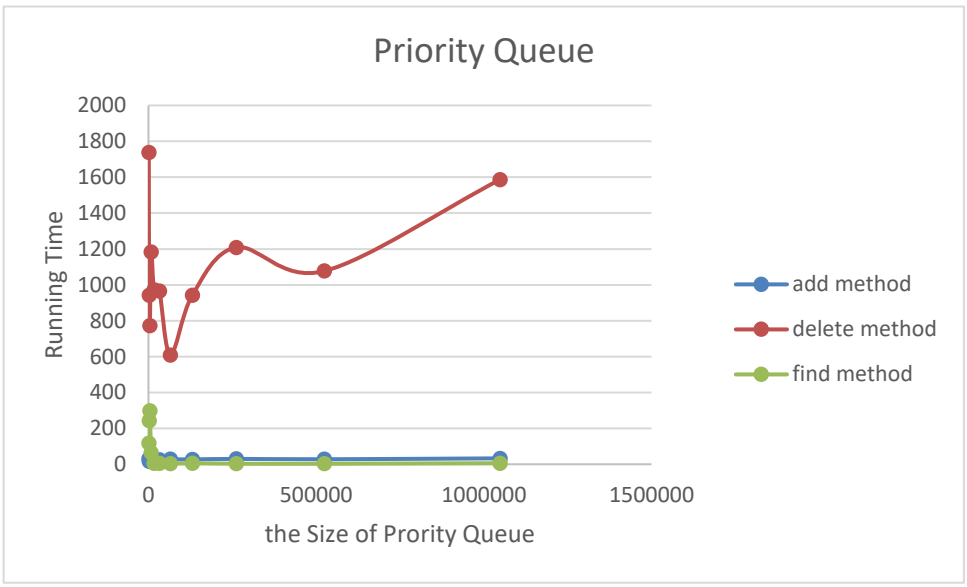
Name: Yixiong Qin

Uid : u0900071

CS 2420

Analysis for Assignment 11

1. I used TimingExperiment08 as we talked in lab 3 to assess the running-time efficiency of priority queue. At the beginning, I declared an "ArrayList<Integer>" and a "PriorityQueue<Integer>". Then I used a "Random" class to generate some random numbers and add them into the "ArrayList". The size of "ArrayList" changed from 2^10 to 2^20. Finally, I called the "addAll" method to add these numbers from the "ArrayList" into the PriorityQueue to get the average time of adding.

2. 

The complexity of "add" method and "find" method in ProrityQueue is O (1), and it's O (logN) of "deleteMin" method. For the "add" method, I used "if-statement"

to assess whether the current element is the first element in the "ProrityQueue", it doesn't have to call the "percolateUp" method if it is, we should move the element to the correct position by using the "PercolateUp" method if it's not. For "percolateUp" method, I used the current element to compare with the root. It needs to switch with the root element if the return number of compare method is less than 0. If the parent of the element is not root, it will need to do a recursion part to compare the element with its parent element. Switch if the return number of compare method is less than 0, return if it is not. In the average case, it just needs to do 1.6 comparisons every time new element insertion. But for worst case, it needs to compare with parent element every time, so the process of it is O (logN). For "deleteMin" method, the implemental logic is similar with the "add" method. It makes sense using the last element to substitute the first element, and then to use the "percolateDown" method to compare with the child element until it gets the correct position.

3. We know that Dijkstra algorithm is a way to find out the cheapest path in graph. We need to use a ProrityQueue which implemented by minimum binary heap in this algorithm. The possible node of graph to the goal will store in the queue, and the cheapest path will be pop if it hits the goal.

4. I almost spent 6 hours on this assignment.