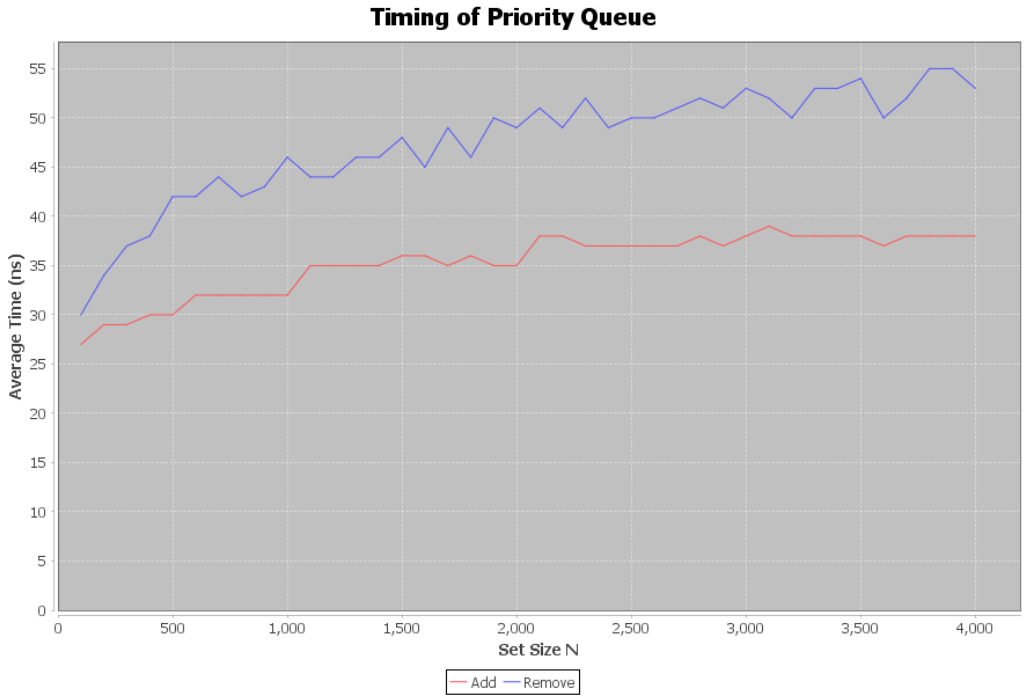
1. I generated this graph by creating a priority queue of size N. The queue was filled from N to 1 so the ordering wouldn't be so nice. I then record the time it took to add 0 (worst case) and remove the minimum.



   The graph roughly shows that both methods are O(log(N)). This is because as you move an item down or up the heap you eliminate half of the comparisons. The reason remove is slower is because you may have to compare to two items to determine if an item should go down the heap. Add only has to compare one.

2. Of the three methods I had to implement, findMin() is O(c) because it just returns the first element. DeleteMin() is O(log(N)) because it has to take the last item in the heap move it to the top and then swap it's way down. Add() is also O(log(N)) because it has to add it to the last place and swap it's way up.
3. One important application of a priority queue that we have talked about in class is finding shortest paths in graphs with edge costs. Something more specific would be a protocol in networking called OSPF (open shortest path first) in which the graph's nodes are routers/switches and the edge costs are ping/bandwidth. OSPF makes sure packets are transported on the fastest path.
4. I spent about 3 hours on this assignment.