Muhammed Salih Yıldız
150180012

BLG 335E – Analysis of Algorithms I

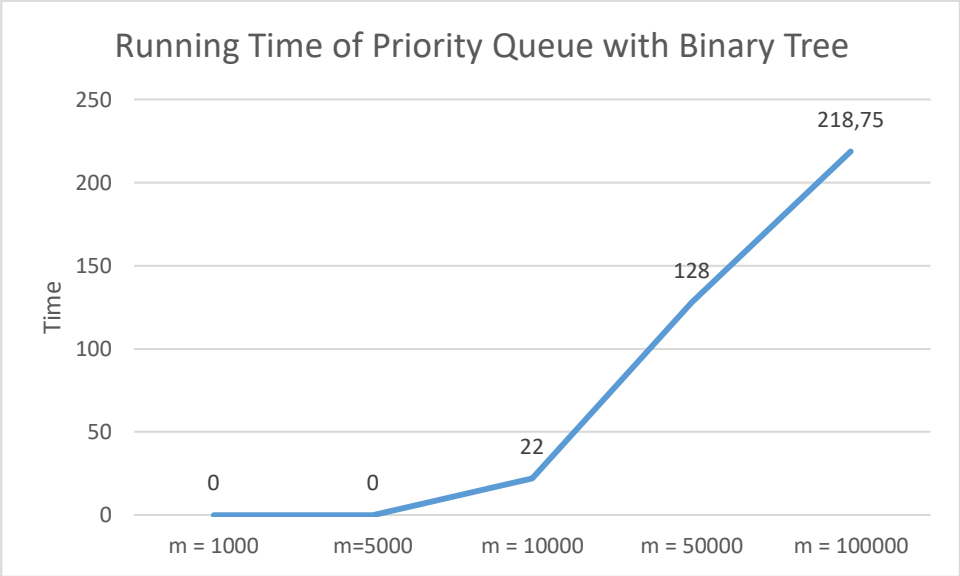
Fall-2020 Homework 2

Report

1)

I used priority queue with min heap in my implementation. There were three operations which are call a taxi, insert a taxi and update a taxi. Call a taxi operation prints the lowest value in the binary tree which is root of the tree. In every call, the program prints the root of the tree which takes O(1) running time and sorts the tree with the Heapify function which takes O(log(n)) running time. Moreover, in the insert operation the program reads a value from text file and appends the calculated value to the tree. Once, it pushes the value to the leaf of the tree then, calls the goUp function which carries the given value to the appropriate position in the upper direction in O(logn) running time. Finally, the update function picks a random index and decreases 0,01 the value of that index. Then, it calls Heapify function which takes O(log(n)) running time to sort the binary tree.

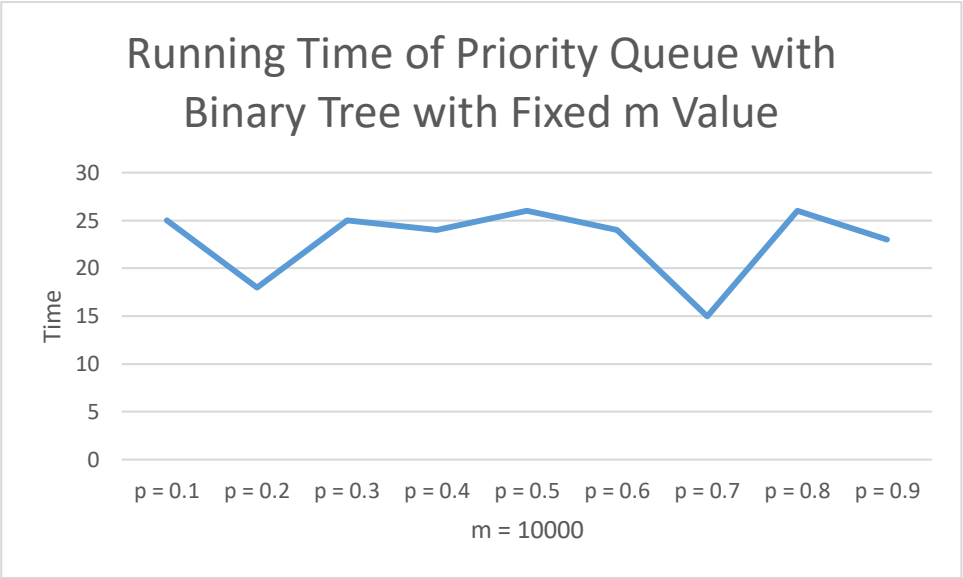
2)



As it shown in the table increasing rate is very similar to O(n log(n)) running time. So, it can be said that running time of the program is O(n log(n)) which is expected. Because in every operation the program sorts the binary tree and this process takes log(n) time. The program makes n operations, so total running time becomes O(n log(n)).

Muhammed Salih Yıldız
150180012

3)



The running time does not change too much. Because p value determines the distribution of the update and insert operations. However, in the both operations the program runs for O(log(n)) time. Since, when updating a node program calls Heapify function which sorts the tree in O(log(n)) time and inserting a new node calls goUp function which runs in O(log(n)) time too. Thus, weights of the update and insert operation does not change total running time.