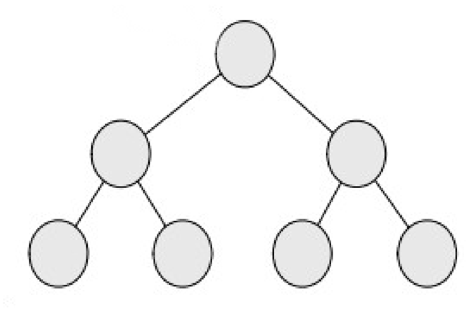
REPORT

1. Please explain your implementation on the PQ operations and the simulation features with their theoretical running times. It is better you support your explanations with comments on your source code file.

A heap structure is smiliar to a nearly complete binary tree. If we want to calculate the deep lenght of the nearly complete binary tree we have to analiyse it first.



In these kinds of trees we can calculate our tree's deep lenght with this:

The relationship between n (node numbers) and h (deep lenght) is:

$n = 1 + 2 + 2^2 + ... + 2^{h-1} + 2^h = 2^{h+1}-1$

And if we calculate h from this:

$h = \log(n+1) - 1$

As we can see that in the lengh of the each node can calculate with this formula, if we use this formula on big O notation

$O(\log(n+1) - 1)$ and if we substute this to $O(\log(n))$ we can find that for each node our time complexity is $\log(n)$.

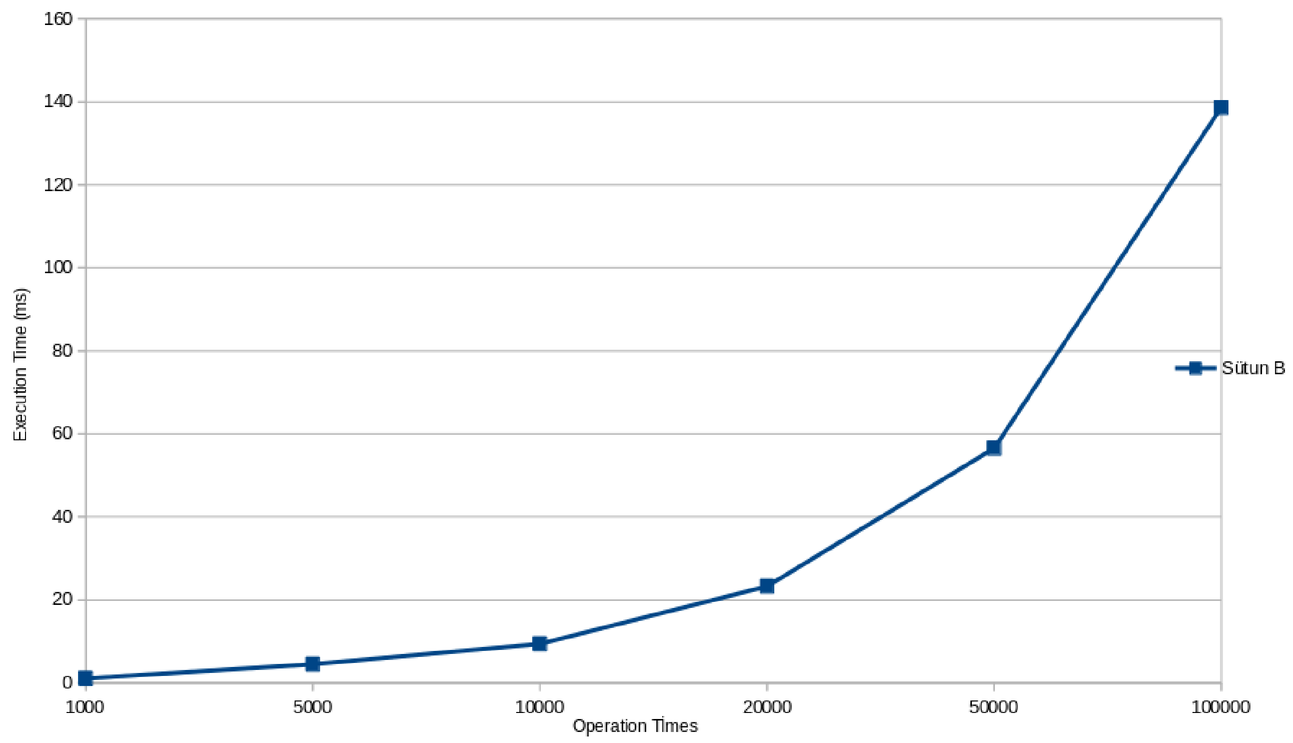
If we use this knowledge on the Min_Heapify function; We can say our each Min_Heapify call we just look at a node and if this node is bigger then one of the left or right leaves we swap them and we can look for another level for this node, and because of that for each node we would have to visit some different levels of deeps, we can say that our Min_Heapify function's time complexity in big O is $O(\log(n))$.

If we look at the Exctract_Min function there is not much to say it's time complexity is the same with Min_Heapify because we just call Min_Heapify from it.

The running time of the Decrease_Key funciton is $O(\log(n))$ because in this function we just compare a node with it's parent and if it is smaller then it's parent we just swap them and continue the comparison. In this case we just compare a node for each level of the tree. Therefore we can say $O(\log(n))$ is our running time.

If we look at the Insert function this function's running time is $O(\log(n))$ becuse we do not do anything other than the calling Decrease_Key Function.

2. (17 points) Demonstrate (using a graph or a table) of the effect of the m choice on the running time. You should run the simulation for different values of m between 1000 and 100000 for a constant p (0.2). Also, discuss the demonstration by answering following questions: Is the graph as what you have expected? Does it match to the theoretical running times?



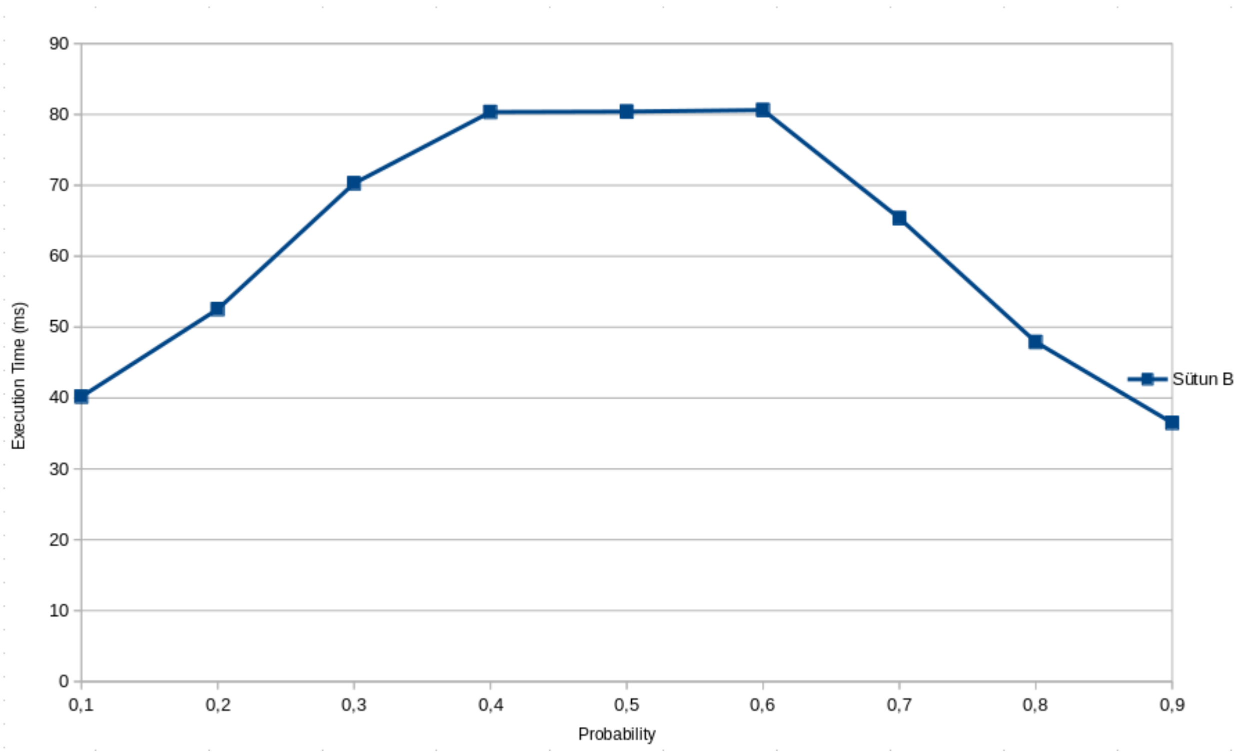
Is the graph as what you have expected?

Yes, As I expected the graph is a smiliar graph to nlogn graph. It grows slower than n^2 graph.

Does it match to the theoretical running times?

Yes Our theoretical running time for each function was nlogn, and we can see in this function that our graph is a similar graph to nlogn graph.

3.



As we can see our changes between 35 ms and 85 ms wecan say that our time does not change much.

Is the running time affected by p?

It is effected but not that much.

Why do you think it is affected or not effected?

If we say it is effected enough it can be beacuse of the simulation direction, we add more taxis it can be more operation for each node but when we add taxis we also decrease the other operation times, if they are nearly equal both addition and update is going to be in balance probability because of that our time increased.

BURAK ŞEN
150170063