Homework #3

Goals:

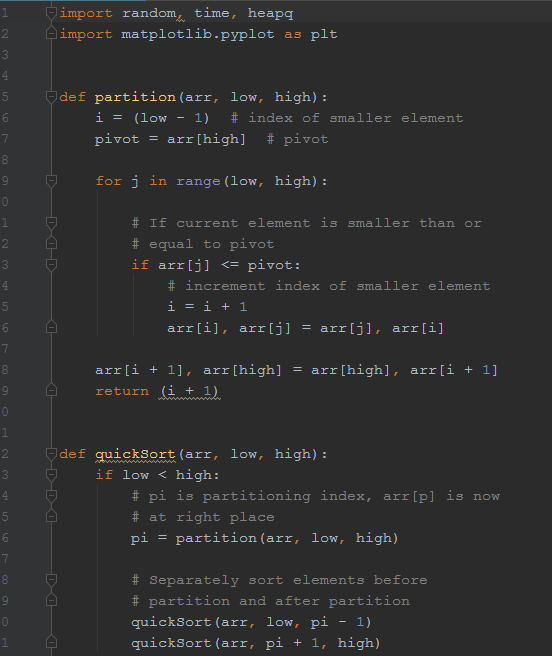
We wanted to make a benchmark analysis in which we compare the time four different functions take to sort the elements of a list with the elements from the smallest one to the biggest.

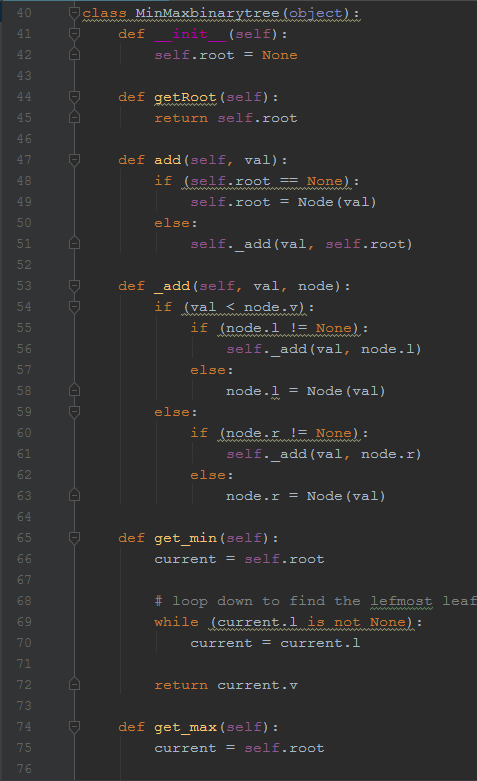
Beginning:

We start by importing three libraries called “random” to produce random numbers; “time” so that we can enable the timing functions we need to measure them and “heapq” to enable the “Heap” function. Then we add another library called “Matplotlib.pyplot” to help us create the graph which we will need at the end.

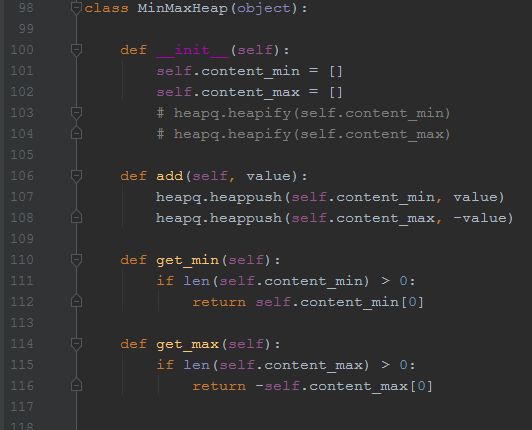
Development:

We then continue our work by implementing the different functions in four different classes. The first one is “quicksort”, which as the name suggest will reveal to be the one of the fastest between these four.

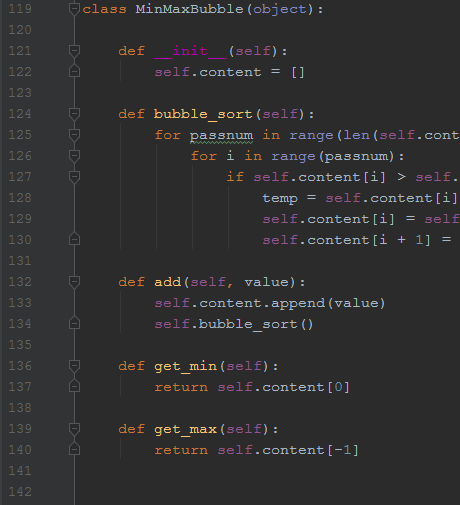


Then we add our second function:”Binary tree”. It uses the divide and conquer tactic to sort all of our elements in a pyramidal scheme of “parents” and children.

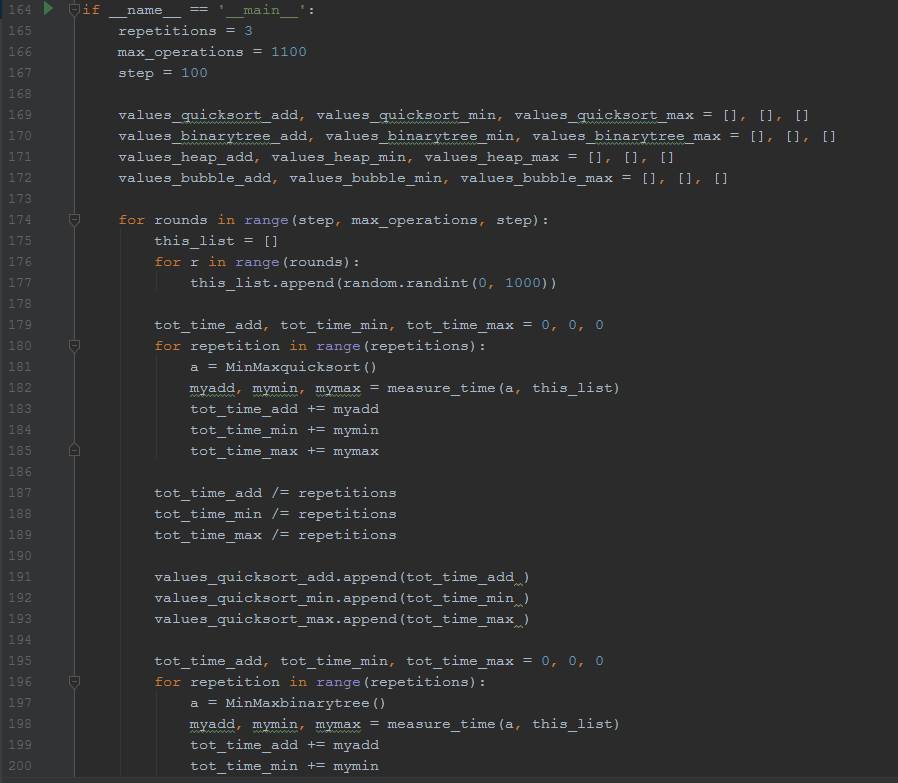
The third one we have is the “heap” method which differentiates from the binary tree because of the fact that heap is the opposite version of “binary tree” when it comes to “parents” and “children” in the “branches” of the function.



Last AND least of this list is “bubble sort”. This program is the least recommended when it comes to sorting as it takes an enormous amount of time to compile the program and overall severely slowed down the rest of the programs.



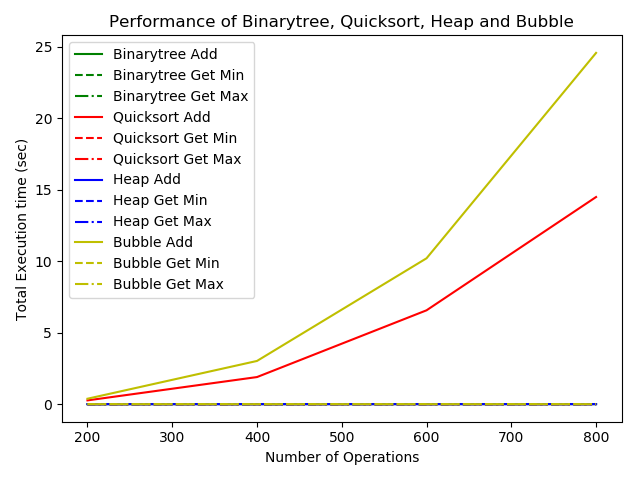
After putting there the 4 programs it was time to measure them by creating some empty lists, and then filling them with a specific number of random values with repetitions, steps and range of the values still dictated by us.(\*All code showed in the “benchmark” file\*)



After that we print all the results so that we can see some actual values to the codes we just wrote. In the “README.md” file there is attached a table with our result perfectly organized so that conclusions can be easily seen.

Conclusions:

The values show the clear result but some image representation done with some graphs can help to visualize better the conclusions of this experiment and clear all doubts that might appear from the data table.



This one is the most reiterative one which shows a perception of how the different graphs do under our three different tasks.

(\*for a variety of more detailed graphs on this topic we can run our “benchmark” program on any python 3 reader”\*)

Python version : 3.7

compiler :PyCharm 2018 2.4 (64 bit)

system : Windows

release : 7

machine : HP EliteBook

processor : Intel Core i7-2820 QM @2.30 GHz Genuine