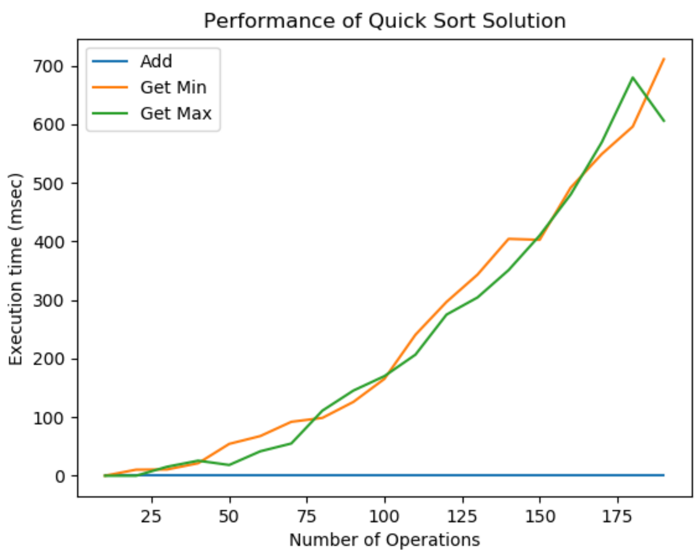
CLASSES PERFORMANCE REPORT

The aim of this report is to explain what has been done for this assignment. What is asked to do? Create four different classes of objects that keep track of the maximum and minimum number in a random list of numbers.

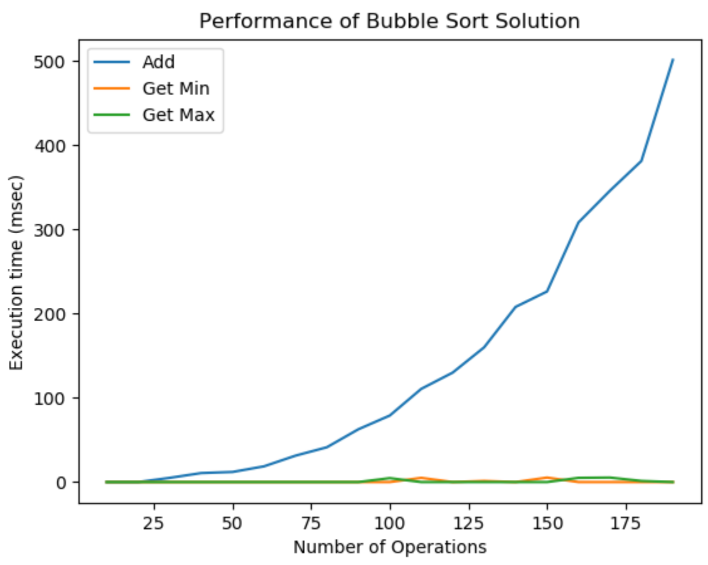
QUICK SORT CLASS

Quicksort is an algorithm, based on 'divide and conquer' strategy, that is used to sort provided lists of numbers. I have created a new class object, called QuickAlwaysSorted that is based on lists object, but somehow implemented, because the get_min and get_max methods contain the Quicksort algorithm, in order to return the sorted list. This class keeps track of the maximum and minimum number in a list of numbers. Quicksort takes O(nlog(n)) average runtime, so indirectly, add method takes O(nlog(n)) average runtime. Big O notation is a mathematical notation that describes the limiting behavior of a function when the argument tends towards a particular value or infinity. The graph, produced with plot methods, shows the performance of QuickAlwaysSorted object in adding numbers, finding minimum and maximum numbers. The plot shows that the add method is constant, equal to 0, while get min and get max methods are increasing with speed O(nlog(n)).



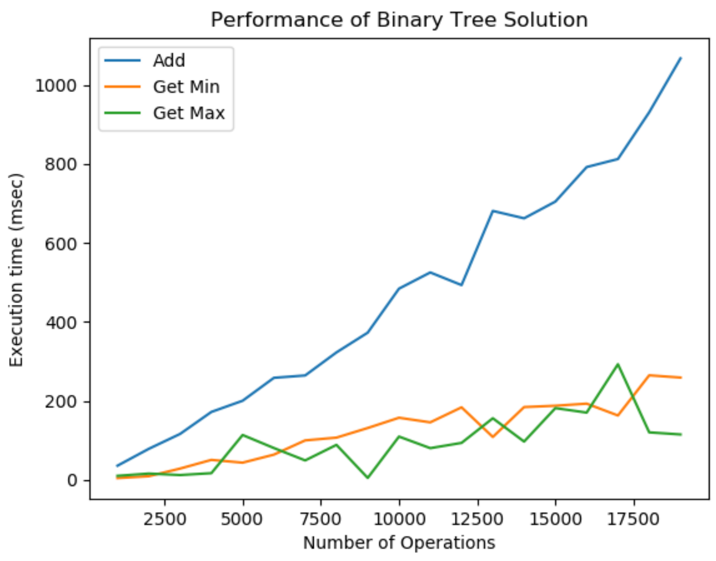
BUBBLE SORT CLASS

Bubblesort is an algorithm that is used to sort provided lists of numbers. It is based on sorting of couples of elements, that are compared and inverted, if in the wrong order. I have created a new class object, called BubbleAlwaysSorted that is based on lists object, but somehow implemented, because the add method contains the sorting algorithm, in order to return the sorted list. This class keeps track of the maximum and minimum numbers in a list of numbers. Bubblesort takes O(n^2) average runtime, so indirectly, add takes O(n^2) average runtime. Big O notation is a mathematical notation that describes the limiting behavior of a function when the argument tends towards a particular value or infinity. The graph, produced with plot methods, shows the performance of BubbleAlwaysSorted object in adding numbers, finding minimum and maximum numbers. The plot shows that the get min and get max methods are constant, equal to 0, while add method is increasing with speed O(n^2).



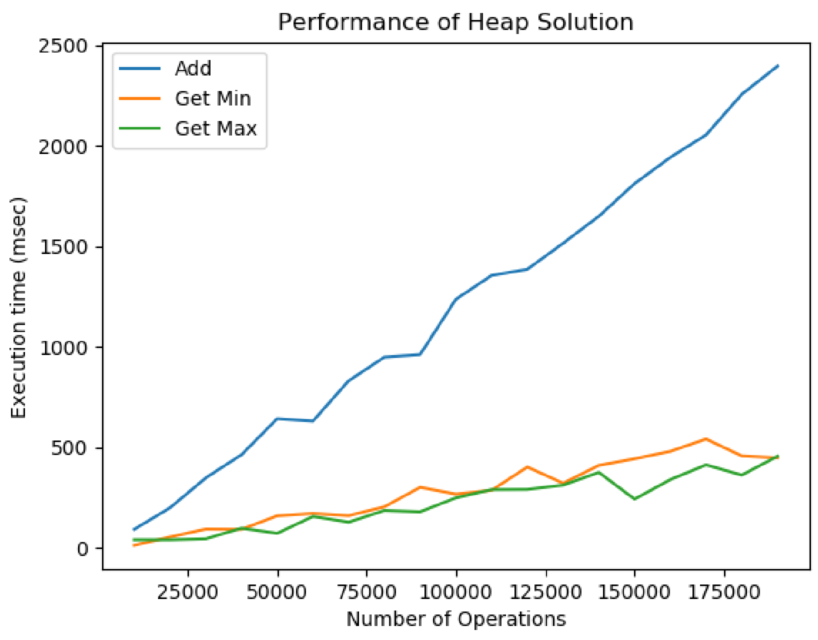
BINARY SEARCH TREE CLASS

A binary tree is a tree data structure in which each node has at most two children, which are referred to as the left child and the right child. I have created two new classes: one for nodes, called TreeNode, the other that is the actual tree, called BinarySearchTree. Differently from previous classes, this class, by definition is sorted. In fact, the tree is built by separating numbers bigger than the root node on the right and small ones on the left: the root node is bigger than his left child, but smaller than its right child. So the smallest number would be the left most and the bigger one the right most. The graph, produced with plot methods, shows the performance of BinarySearchTree object in adding numbers, finding minimum and maximum number. The plot shows that the get min and get max methods are slowly increasing at O(n) speed, while add method is increasing with speed O(log(n)) . Big O notation is a mathematical notation that describes the limiting behavior of a function when the argument tends towards a particular value or infinity.



HEAP CLASS

A heap is a specialized tree-based data structure which is essentially an almost complete tree that satisfies the heap property: if P is a parent node of C, then the key (the value) of P is either greater than or equal to (in a max heap) or less than or equal to (in a min heap) the key of C. The node at the "top" of the heap (with no parents) is called the root node. In a heap, the highest (or lowest) priority element is always stored at the root. I have created the MinMaxHeap class object that is actually creating two heaps: a min heap and a max heap. Why so? Because in case of min heap, the minimum number- node is the root node; in case of max heap, the maximum number- node is the root node. The max heap is built by adding the opposite (negative) value to the heap structure. The graph, produced with plot methods, shows the performance of MinMaxHeap object in adding numbers, finding minimum and maximum number. The plot shows that the get min and get max methods are slowly increasing at O(n) speed, while add method is increasing with speed O(1). Big O notation is a mathematical notation that describes the limiting behavior of a function when the argument tends towards a particular value or infinity.



MEASURING CLASSES PERFORMANCES

To compare the performance of the four classes, I have to import ‘time’ package. I create a function called ‘measure the time’ which takes as arguments the random list and one of the empty class object. It uses the ‘time’ method: it takes track of the initial and ending time, of the machine on which is run, of the required process: add method, get min method and get max method. Finally, I keep this time values in three variables.

Then I create three lists, in which all the time values will be kept. How is produced the random list? I have to import ‘random’ package and use the ‘randint’ method to append random numbers, within a range, to a list that is used then in the measuring. The function ‘measure the time’ is run three times, according to the number of repetitions, and then an average of these three repetitions is done. Finally, I convert the total time for each method from seconds into milliseconds, and append it in some variables, lists, that are used then for the plotting.

PLOT

The first thing to do is to import 'matplotlib' package: for simplification, I call it 'plt'. Then to have the graphs I use the 'plot' method, which takes two inputs, which are the x and y coordinates of the points, and a different default input that define a characteristic of the graphs: I modify the label of each graph. Then I labelled the x and y axis, which respectively represent the number of operations and the execution time, expressed in milliseconds. In the end, I put a title to my plot and I use the 'show' method to visualize the complete plot.

CONCLUSIONS

By looking at the plots present in this report and the formulas of average running time, heap class can be considered the one with fastest performance, as regards tree based data structure; while quick sort class can be considered the one with fastest performance, as regards list objects.