**Report on sorting methods**

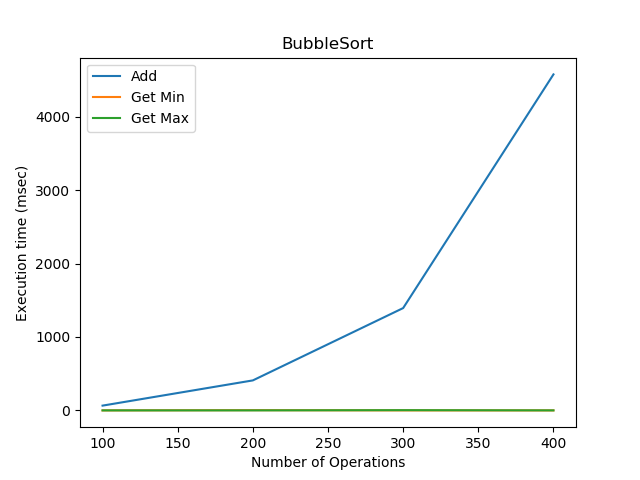
The submitted codes, which are BubbleSort, QuickSort, Heap and BinaryTree, are all used to sort a group of numbers, using different methods. They are also able to find the minimum and the maximum value in the list. Now, let’s analyze each of them.

**Bubble Sort**

This function, in order to sort a list of numbers, compares a couple of number next to each other and either leaves them as they are or exchanges them respectively if they are or not in the correct order. Then it checks the next couple and so on until the entire list is sorted.

The best case for bubble sort is if the list is already sorted and the number of operation will correspond to the number of elements in the list (n). In the worst case, which is when the list is order in the opposite way, instead, the number of operation will be the result of n².

As we can understand from the graph, ‘Add’ takes a lot of time, while ‘Get Min’ and ‘Get Max’ are very fast.



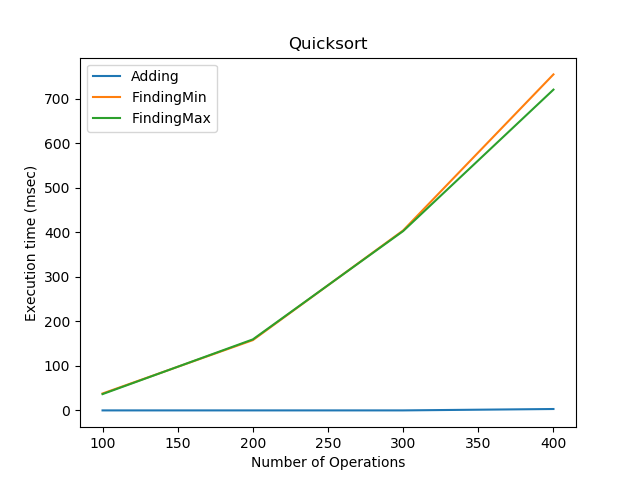
**Quick Sort**

The function QuickSort takes a number, called pivot, and then checks if the other numbers in the list are smaller or bigger than the pivot, moving them accordingly to the left or to the right of the pivot, repeating this operation until it gets a sorted list.

The best case for the QuickSort function is when the pivot element divides the list into two equal halves by being exactly in the middle position and so we make only log₂n calls before we reach a list of size 1.

The worst case, instead, is when the list is arranged in the opposite order with respect to the type of order that we want (either increasing or decreasing); in this case, the number of operations increases to n².

As we can see in the graph, ‘Adding’ is really fast, while ‘finding the Max’ and ‘finding the Min’ take more time.

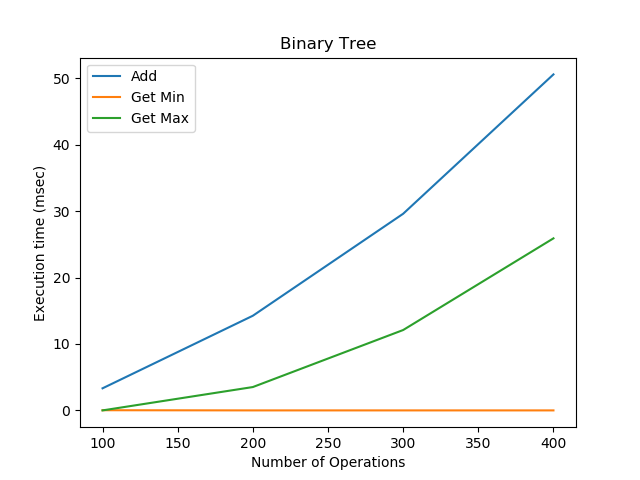


**Binary Tree**

A binary tree is a tree-shaped data structure in which each node has at most two children, the left child and the right child. So sorting a list with this type of function means that each number will have two ‘children’ according to a criteria, which in this case is a numerical order.

In the best case, binary trees with n nodes have a height of log n. While, in the worst case, they have a height of n, which means only one child for each node (degenerate tree).

Adding depends on the number of nodes and so ‘Add’ is the operation that takes more time, while, as it can be seen, the ‘Get Max’ and even more the ‘Get Min’ are very fast, so the maximum and the minimum value are easy to find.

****

**Heap**

The HeapSort function divides its input into a sorted and an unsorted sub list, and it iteratively reduces the unsorted region by taking the largest element and moving it to the sorted region. Similarly to the binary tree, this function organizes a list of numbers by creating nodes and the parent node can only be greater than the children nodes. In this way it is easier to find the maximum and the minimum values.

The number of operations required depends only on the number of levels the new element must rise to satisfy the heap property, thus the insertion operation has a worst case time complexity of O(log n) but an average-case complexity of O(1).

As we can see from the graph, the functions ‘findingMax’ and ‘findingMin’ are definitely faster than the ‘Adding’ function.

