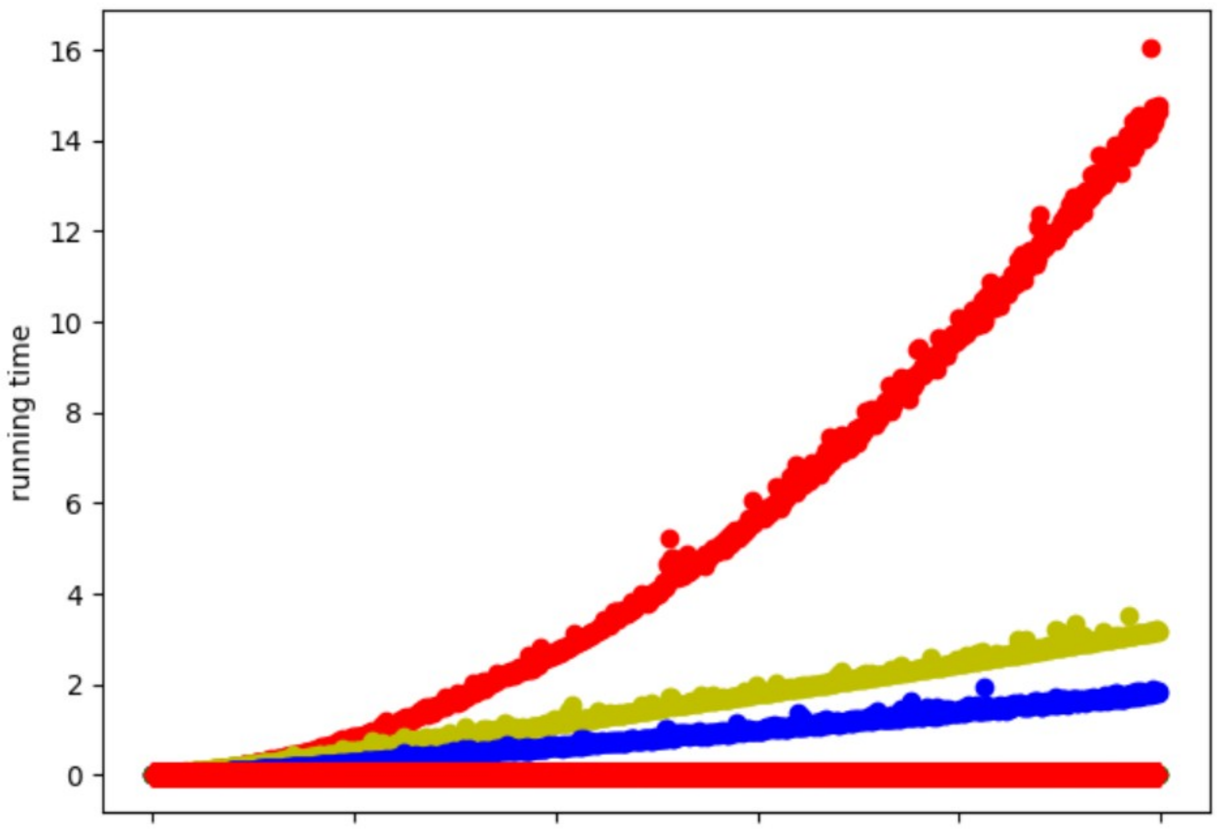
Report:

The goal of the current task was to experimentally estimate the time complexity of different sorting algorithms on different data structures. This report will explain some details of my solution.

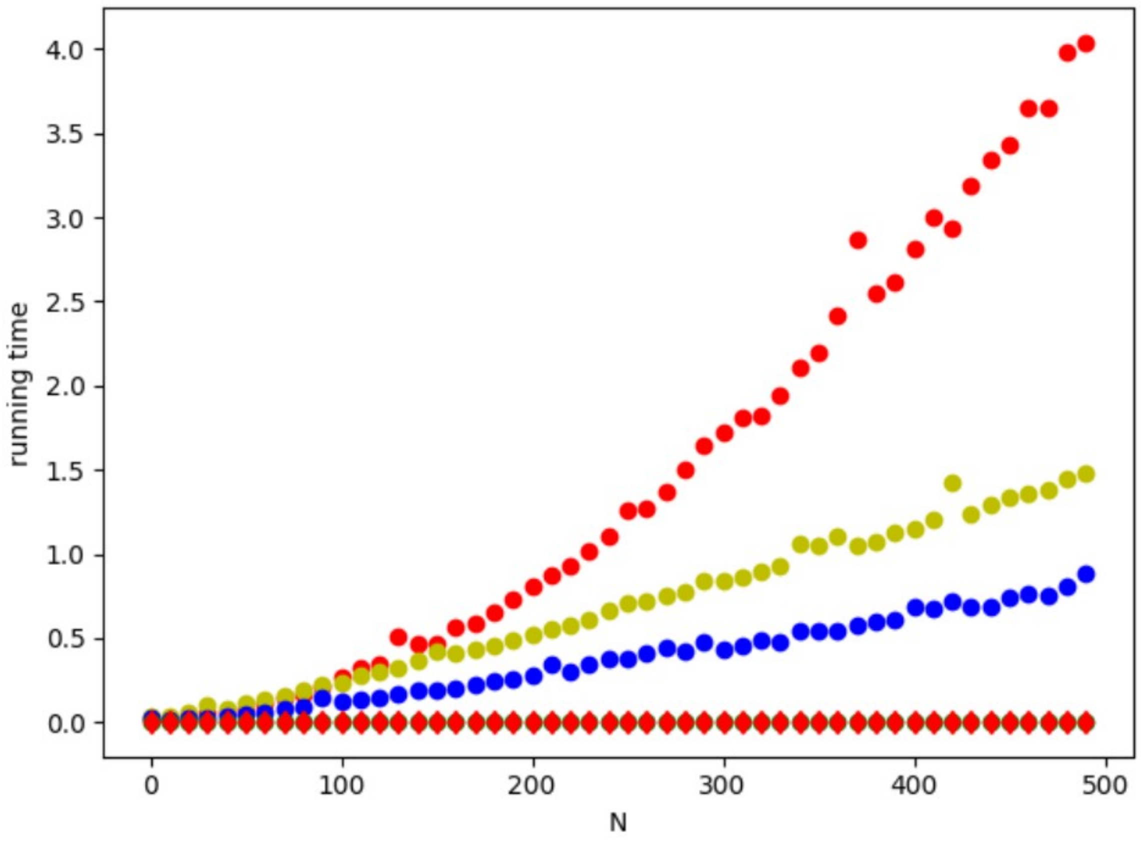
After having implemented the asked 5 sorting algorithms it was finally time to test it. I have done several:

Assume that **n** is the upper bound of the total number of elements either in a std::vector<> or in a doubly linked list.

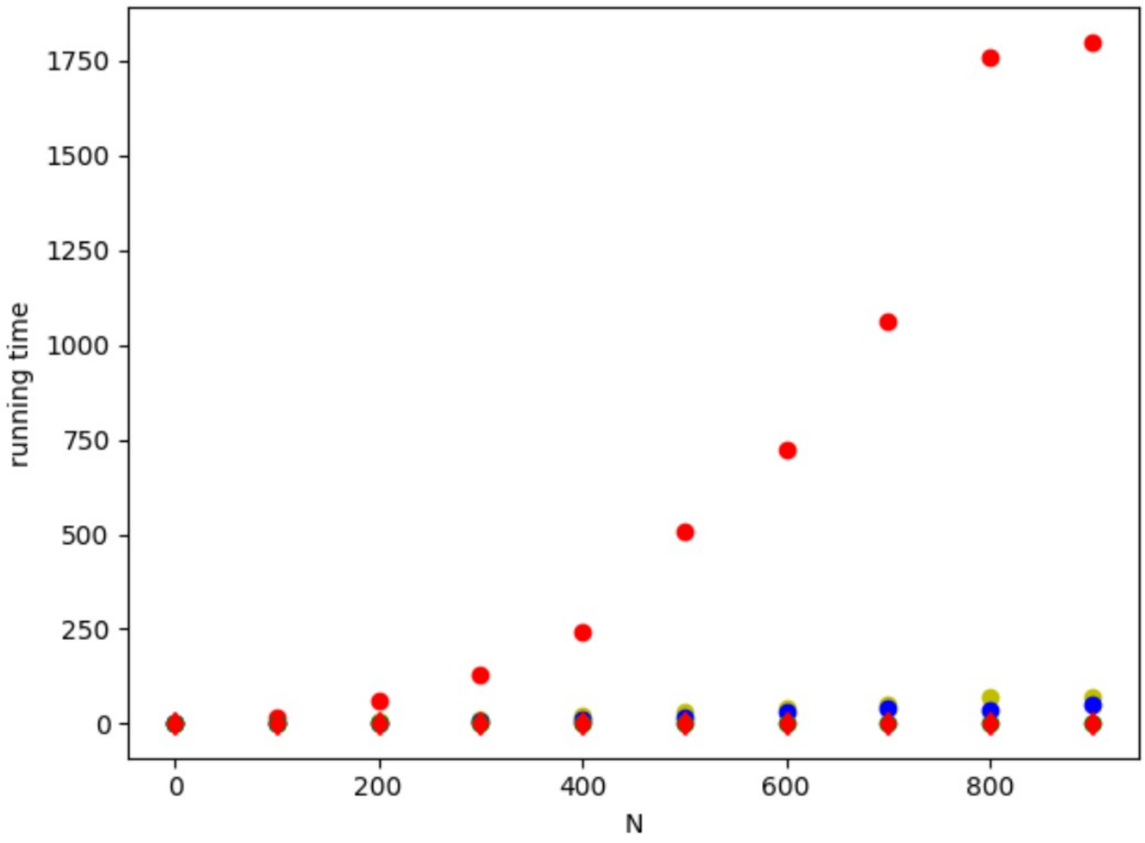
Plot #1 – n = 1000, step = 1:



Plot #2 – n=500, step = 10:

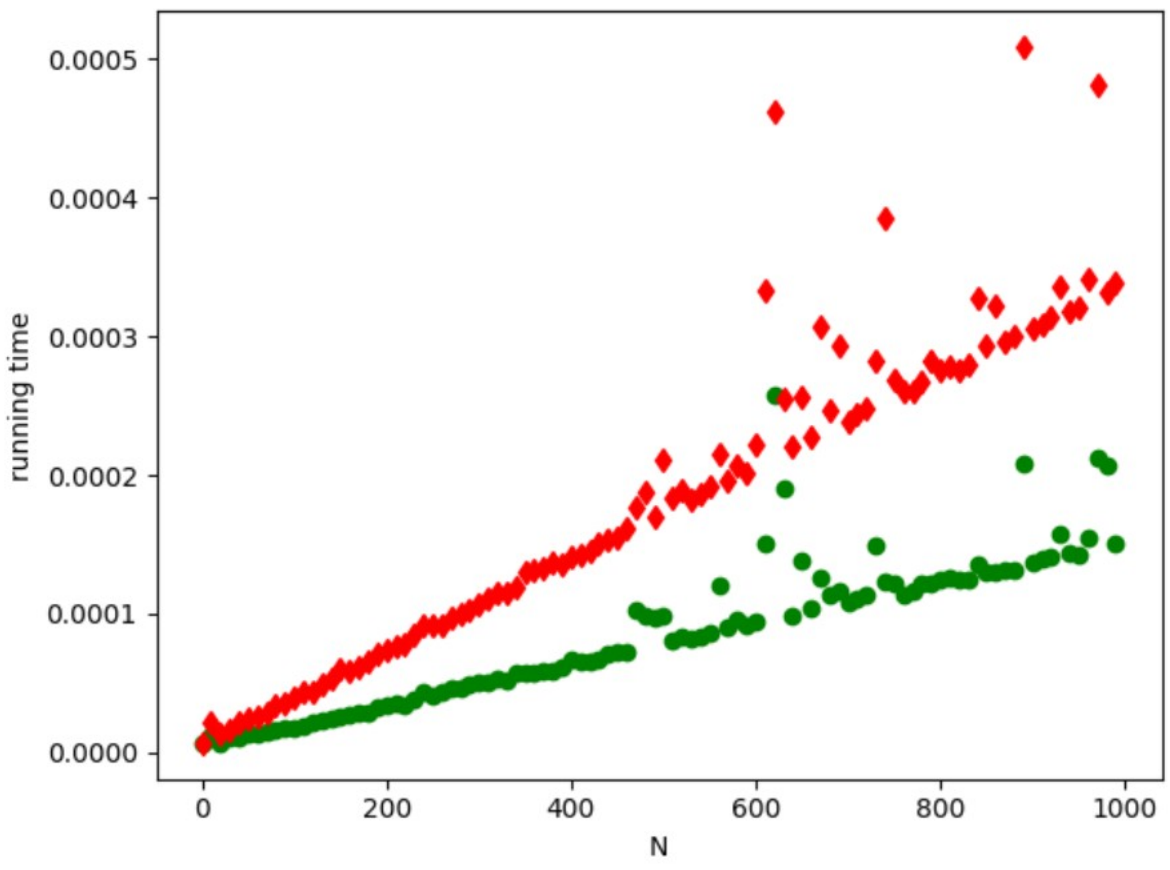


Plot #3 – n=10 000, step = 1000:



- The last plot is more informative in terms that it clearly illustrates the fact that Insertion sort has complexity $O(n^2)$ and shows a drastic difference with all the other algorithms;
- The first two plots show that both Quick and Merge sort have complexity O(nlogn). Moreover, the coefficient for the Merge sort complexity is slightly greater than the one for the Quick sort;
- One can easily deduce from the first three graphs that the complexity of the Counting and Radix sorting algorithms is O(n). Nevertheless, in order to get some clarity, let us consider just these two algorithms in the following plot:

  Plot #4 – n = 1000, step = 10:



Regarding this case, the Radix soring algorithm has a higher coefficient that the Counting one.

Remark: However, that might be the case because of a quite small n. Unfortunately, I do not have any kind of access to a machine which will give me the opportunity to have those estimations less than in a day and without blowing up my laptop.

Overall, we come up to the following conclusion (in terms of time complexity) :

Counting < Radix < Quick < Merge < Insertion.