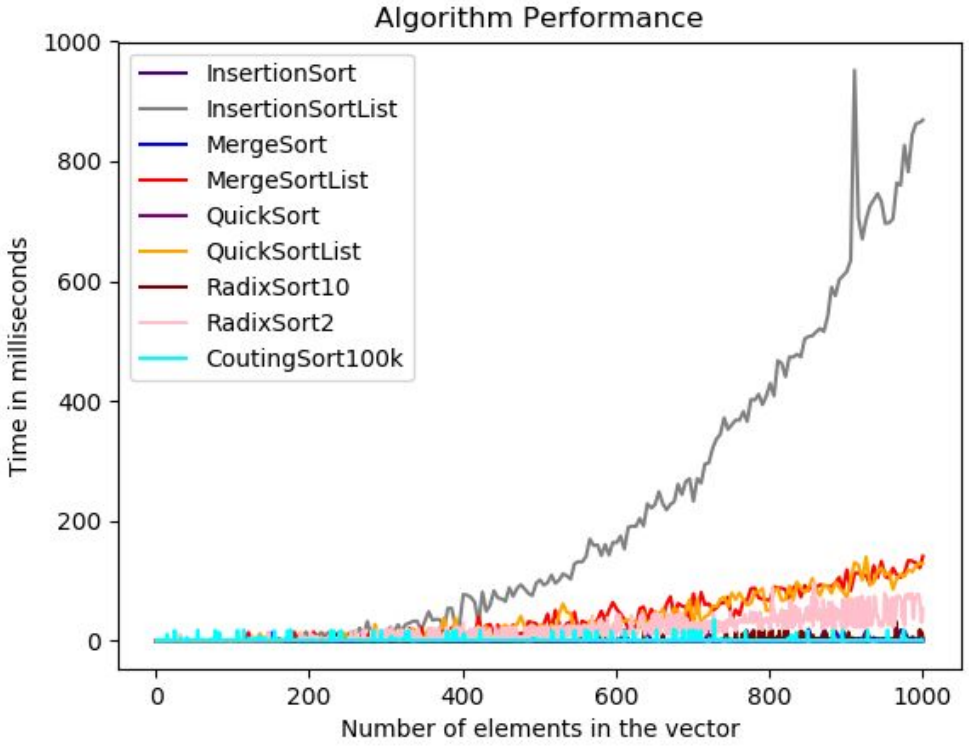
Problem 2 report

This task required to realize Insertion, Merge, Quick sort of vectors or doubly linked lists of integers or strings and required to realize Counting, Radix (base 2, 10) sort for vectors of integers. Then, one has to collect time data via implemented functions and visualize it using pyplot.
This is the example of input data, vector/list of integers and vector/list of strings.

```
clang version 7.0.0-3~ubuntu0.18.04.1 (tags/RELEASE_700/final)
> clang++-7 -pthread -std=c++11 -o main main.cpp
> ./main
11393 60666 27654 96011 55996 -4389 46339 61824 29172 91201 85966 198 22610 55550 77335
04peev g a RDdlTC y2u4h r9V O6qZFV1d yiVYGr gNcsfr H UJ 36XFH 2R dMBZv iPbI
```

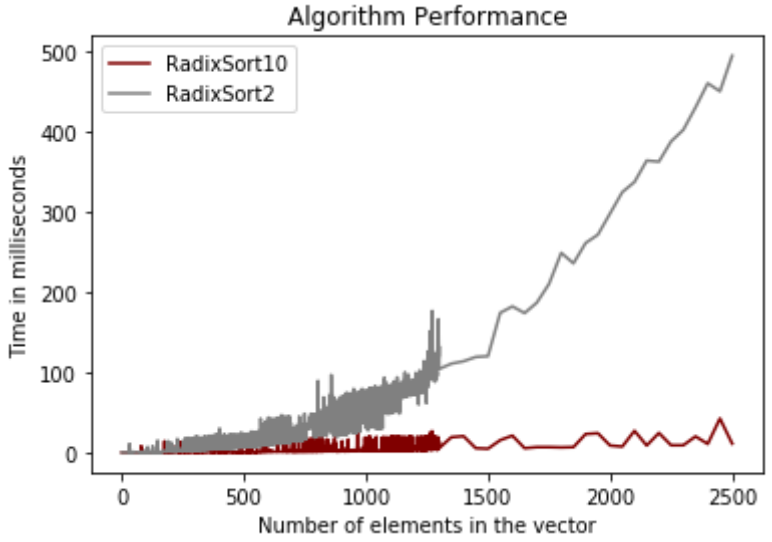
I decided that it's relevant to compare sorting algorithms for vector/doubly linked lists of integers and algorithms for vector/doubly linked lists of strings separately as one usually searches for fastest sort for exact data type.

Units were measured in milliseconds as sorting worked really fast.



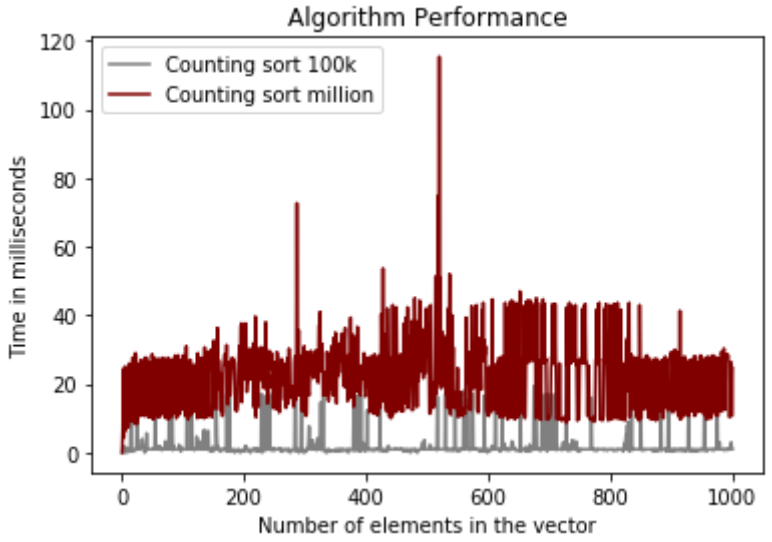
This is graph which includes all sortings for integers. One may notice that InsertionSort for list works really slow. It's evident, lists are extremely uncomfortable because there is no fast way to get some element (node) that is not beginning or ending of the list by it's index, I even implemented method called get(int index), however it doesn't solve problem of slow access to elements. MergeSort and QuickSort of list work significantly faster but not as fast as same functions for vectors.

Now I would like to talk about RadixSort, which was my favourite algorithm during the task, I really like it's idea and my realization of it. It's speed depends more on the



length of biggest number, as it's checks each digit to distribute elements between buckets. RadixSort2 is slower as it checks bits instead of digits, there are more of them. These sorting were tested on numbers from 1 to 100 000 000.

Next sort was CountingSort which is not really fast when there is big difference between minimum and maximum numbers in vector.



These functions were tested for difference equalled 100 000 and 1 000 000. I refused to test it on bigger difference as it worked too long. The second issue is that even if there are 5-10 elements in vector, but there is the smallest possible int and the biggest one, it would not work as it's impossible to create vector of this size to perform sorting.

The last graph is for sorting strings. My program includes generator of random strings of given size. I tested algorithms on strings from 1 to 5 symbols, it wasn't written in task conditions and i was told to choose the size that i want. As for time, the situation is almost the same as for integers, sortings work slower for lists, faster for vectors. InsertionSort for list is also the worst one here.