Istanbul Technical University
Faculty of Computer and Informatics
Computer Engineering Department

BLG 335E
Homework 2
Report

İbrahim Ethem Göze - 150190054

December 4th, 2022

Contents

1 Introduction

The code is for finding a data set's mean, standard deviation, minimum and maximum value, first and third quartile, and median efficiently.

2 Algorithms Explained

I used a vector if necessary and sorted it with insertion sort. If the program asked only minimum and maximum value I did not create a vector. Instead I held those values in global variables. If any of mean, standard deviation, first quartile, third quartile or median asked then I created a vector.

2.1 Insertion Sort

I used insertion sort because we needed to sort the values again and again. After some point only a couple of elements needs to get sorted. So insertion sort has almost O(n) complexity in this situation. I even stored the size of the vector last time I sorted it and started the next sorting after that point to save more time.

3 Images and Figures

I did not include output times for finding the maximum value because it was almost identical with finding the minimum value. Similarly finding firstq and thirdq are very similar with finding median so, I did not include them to.

3.1 Mean Time Diagrams

Figure 1: Mean 10

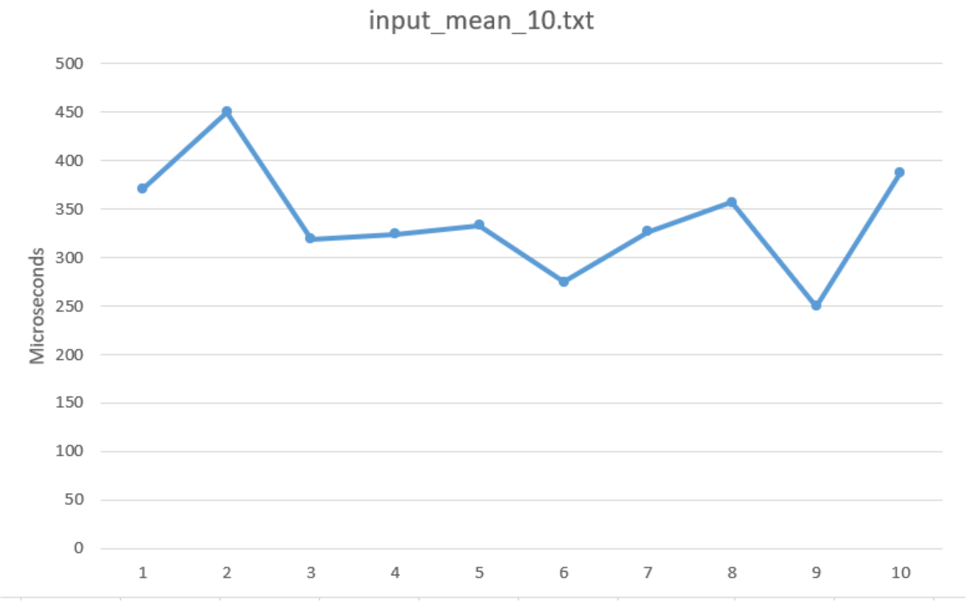


Figure 2: Mean 100

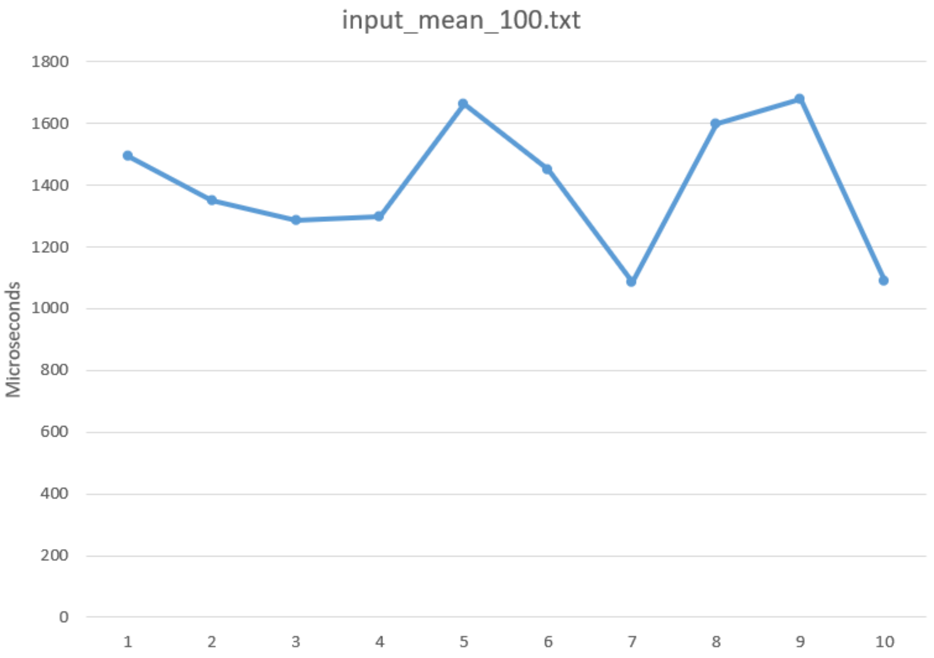


Figure 3: Mean 1000

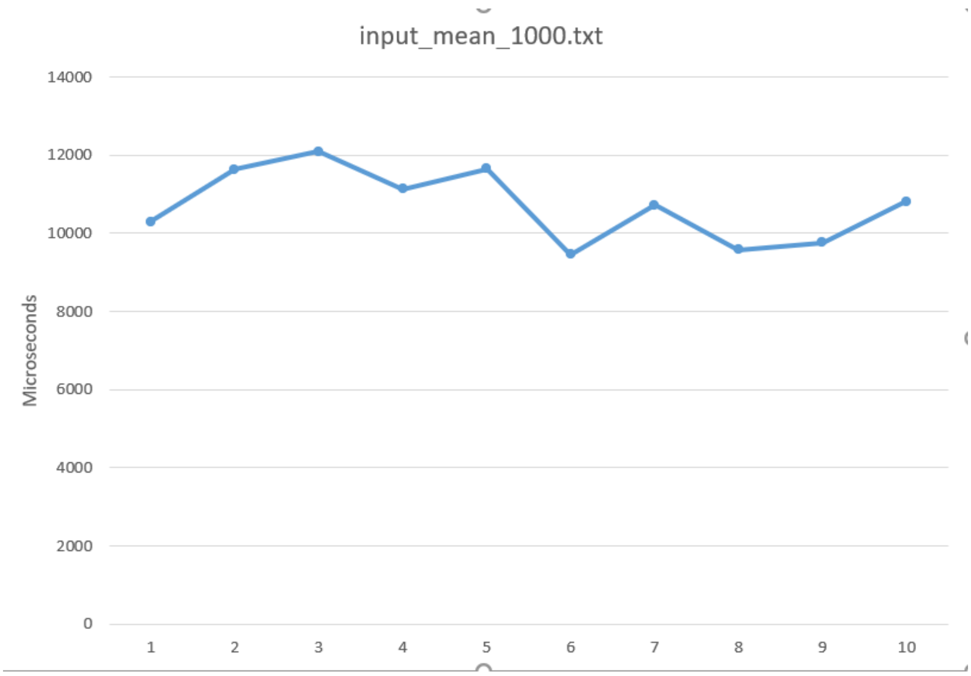


Figure 4: Mean 10000

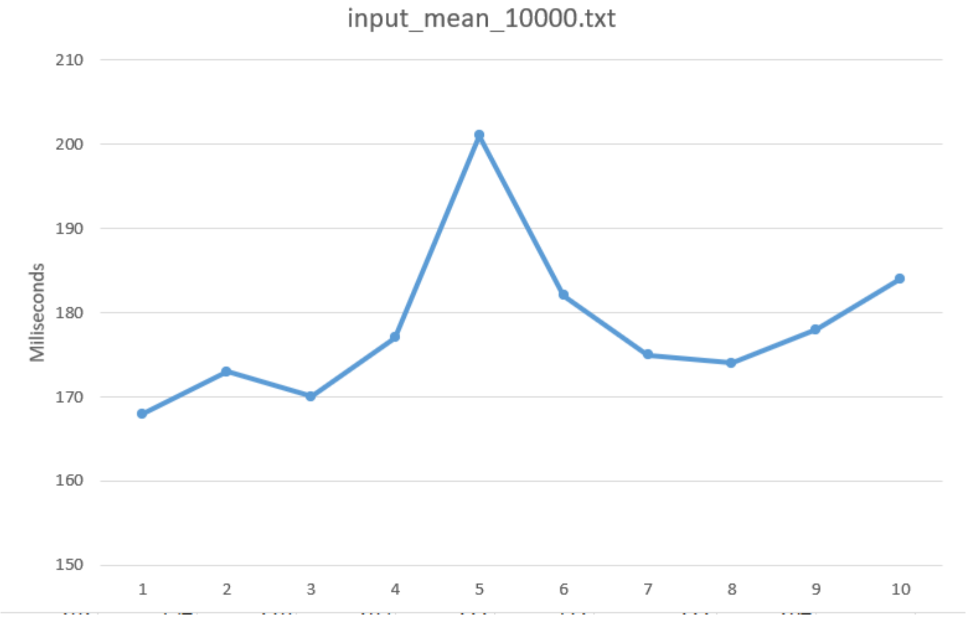
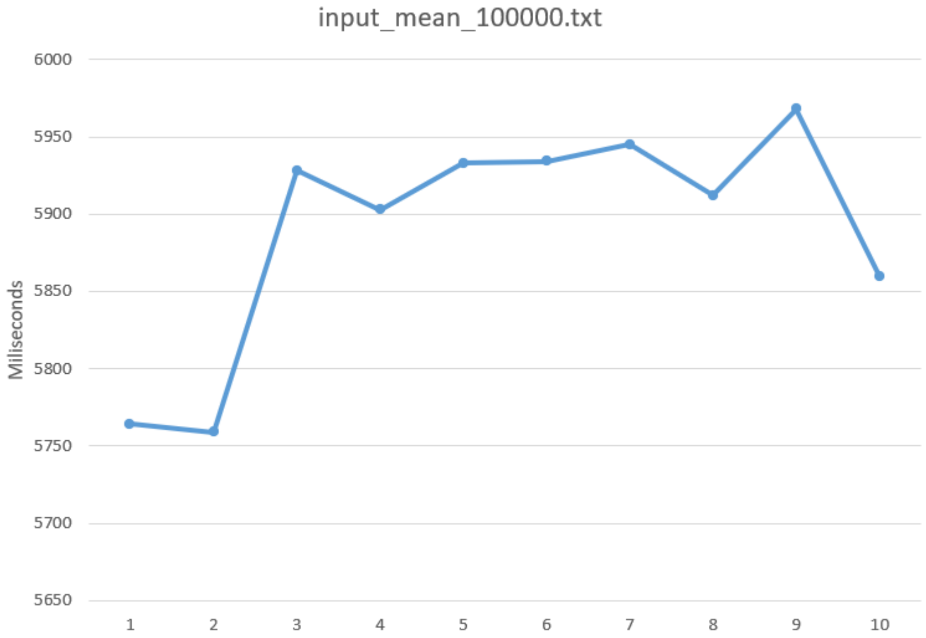


Figure 5: Mean 100000



3.2 Standard Deviation Time Diagrams

Figure 6: Std 10

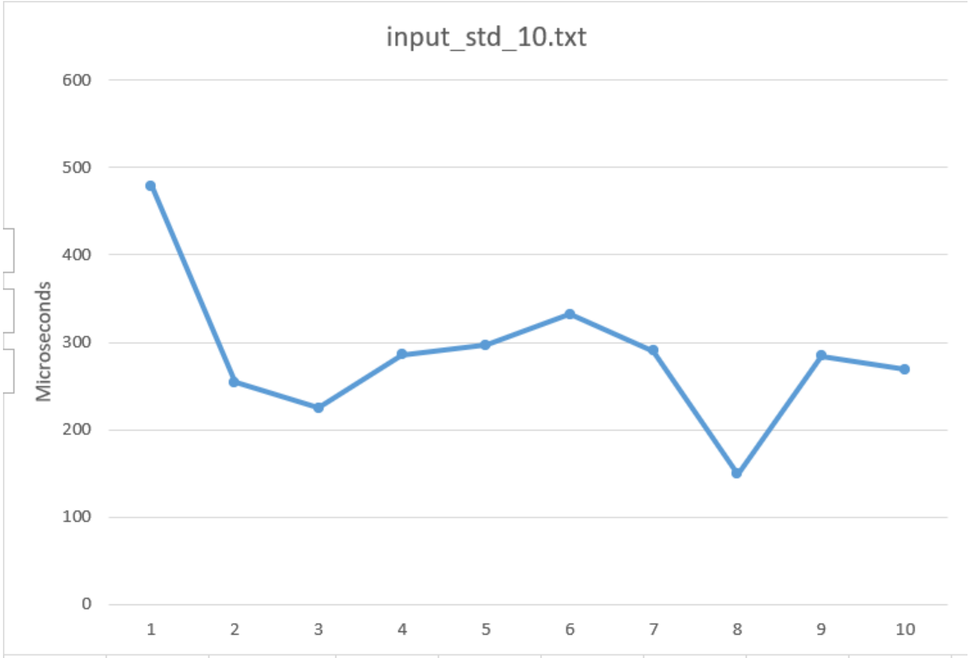


Figure 7: Std 100

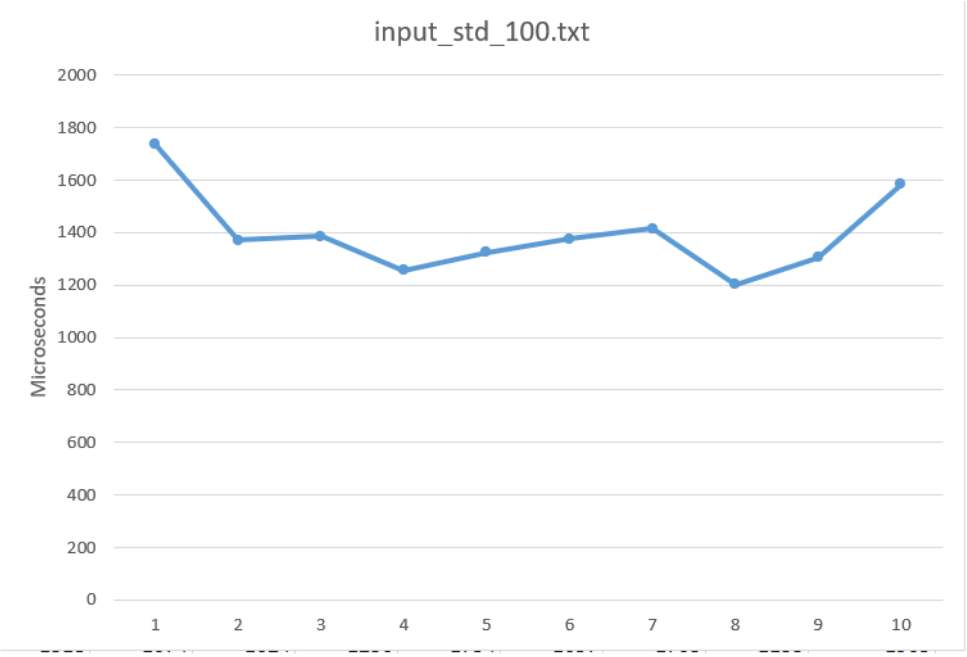


Figure 8: Std 1000

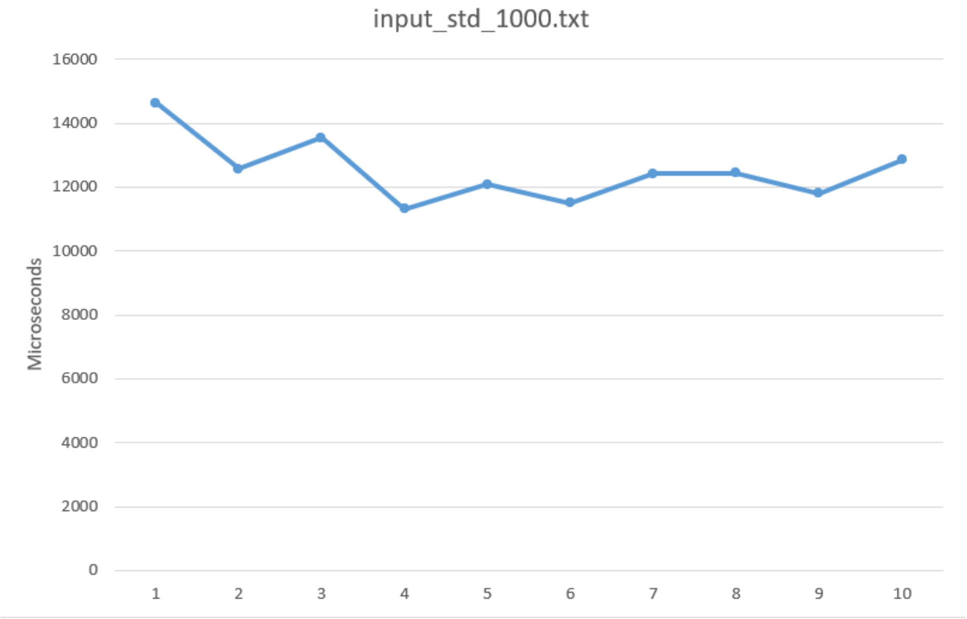


Figure 9: Std 10000

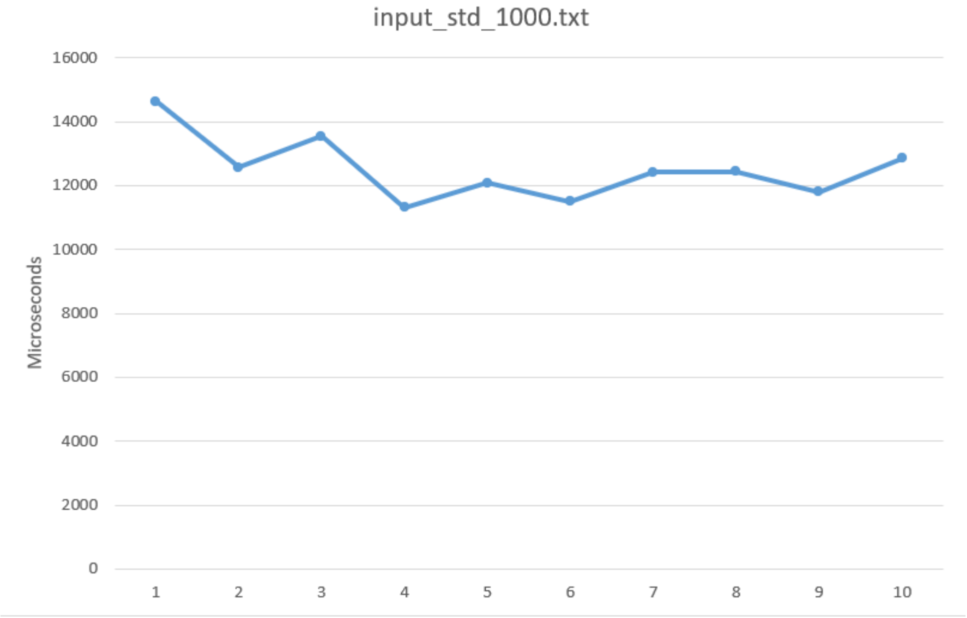
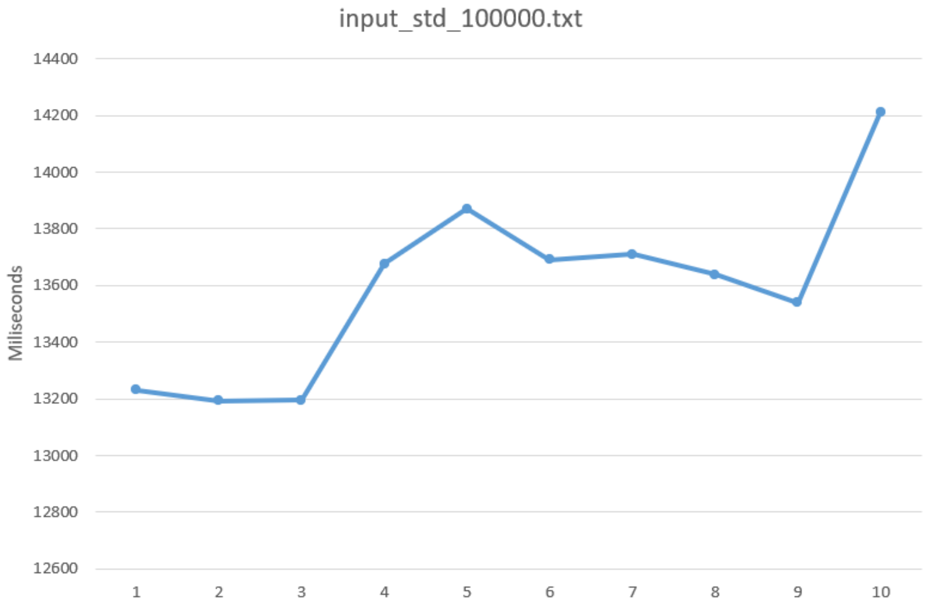


Figure 10: Std 100000



3.3 Minimum Value Time Diagrams

Figure 11: Min 10

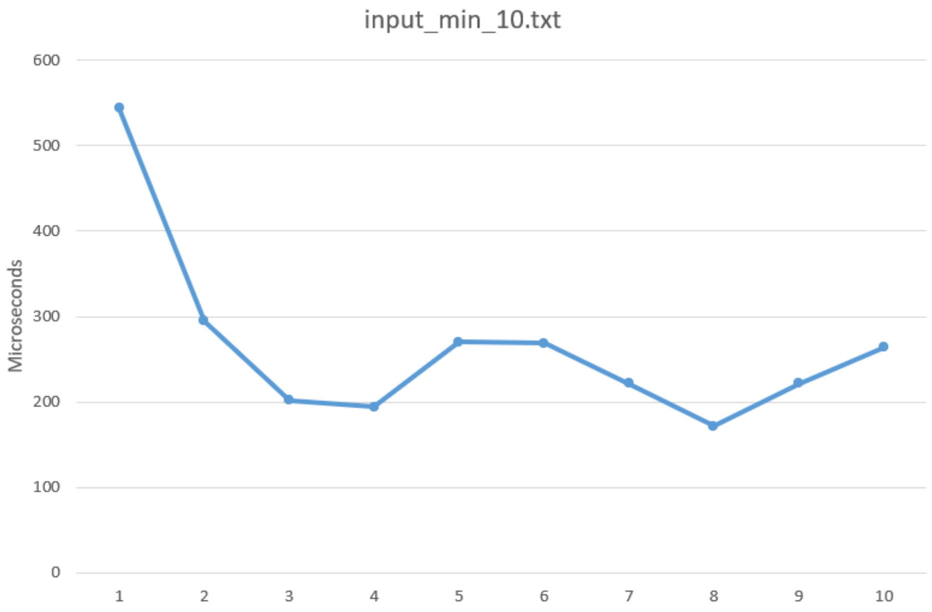


Figure 12: Min 100

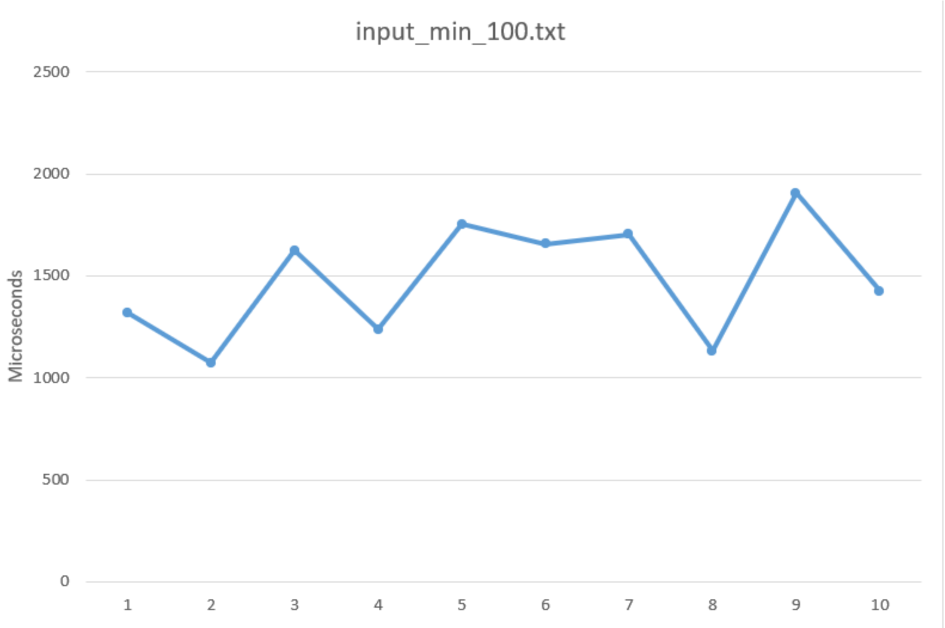


Figure 13: Min 1000

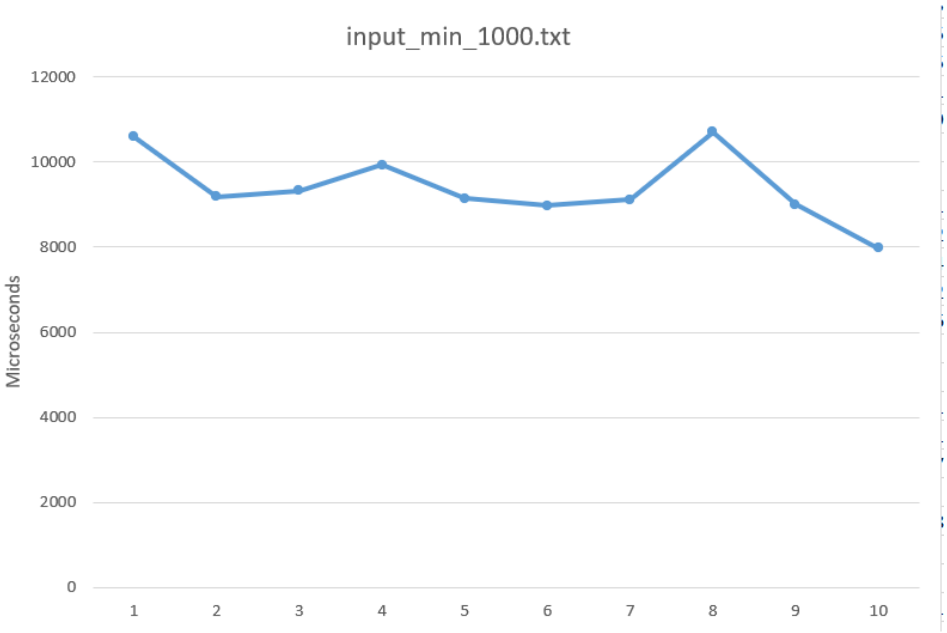


Figure 14: Min 10000

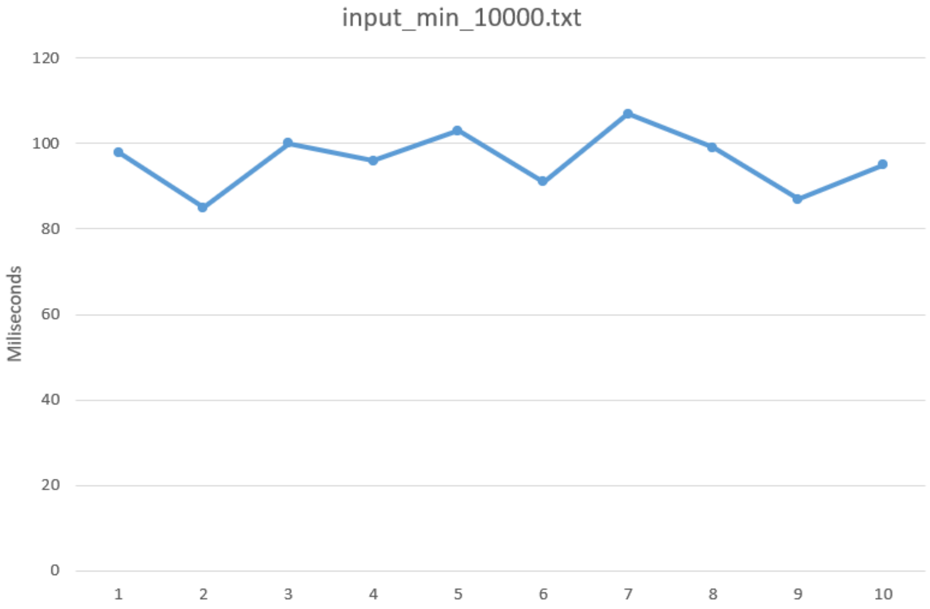
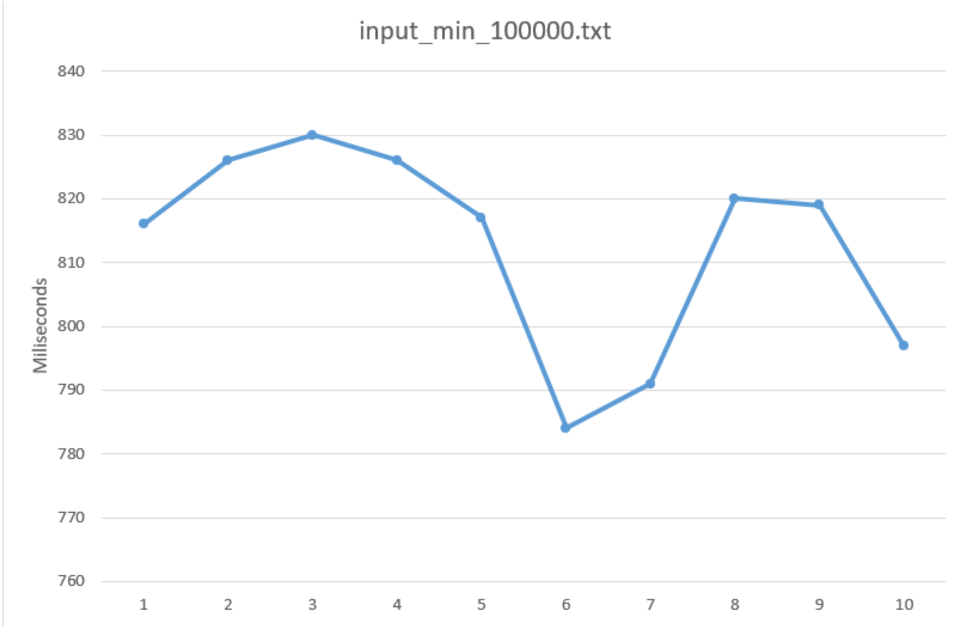


Figure 15: Min 100000



3.4 Median Time Diagrams

Figure 16: Median 10

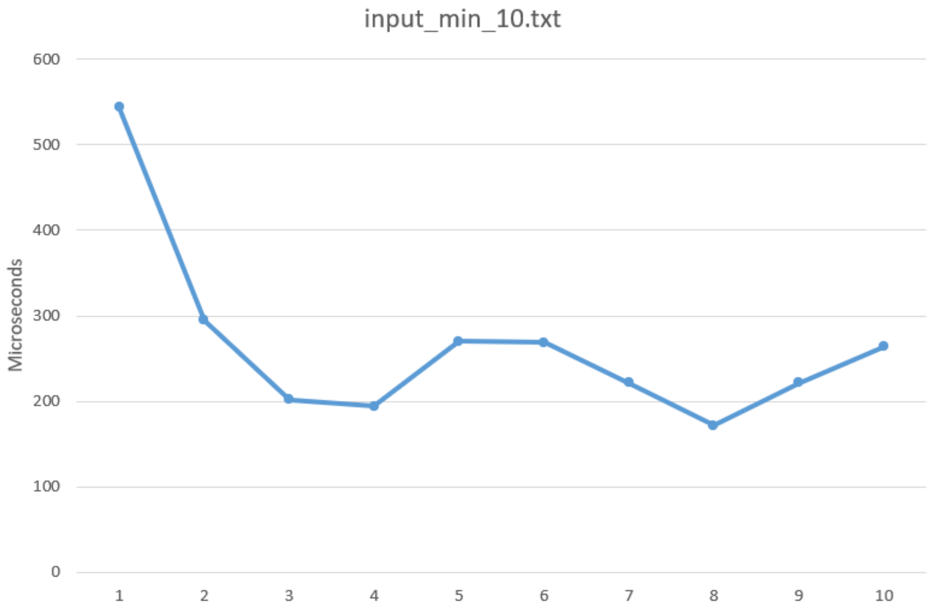


Figure 17: Median 100

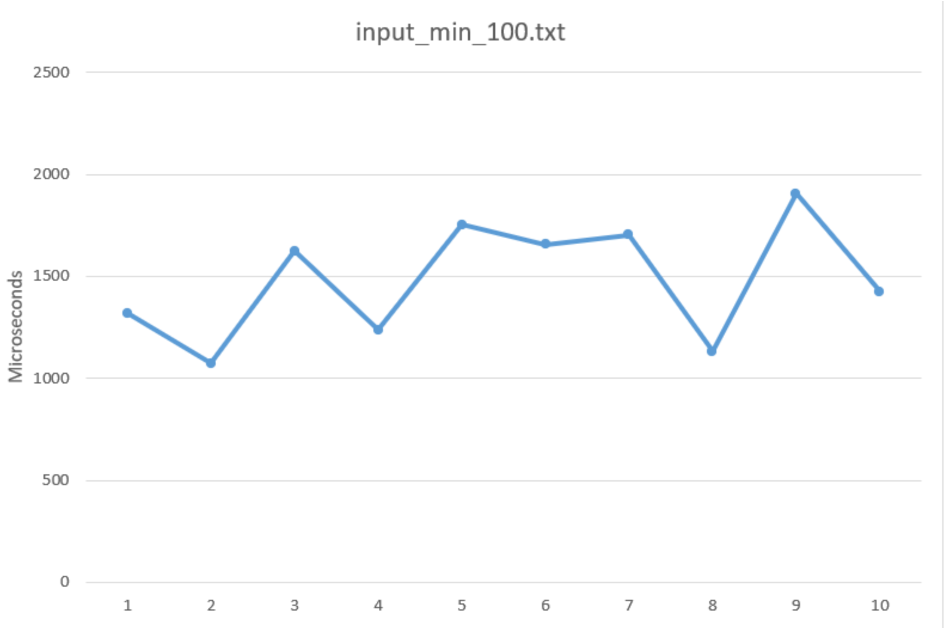


Figure 18: Median 1000

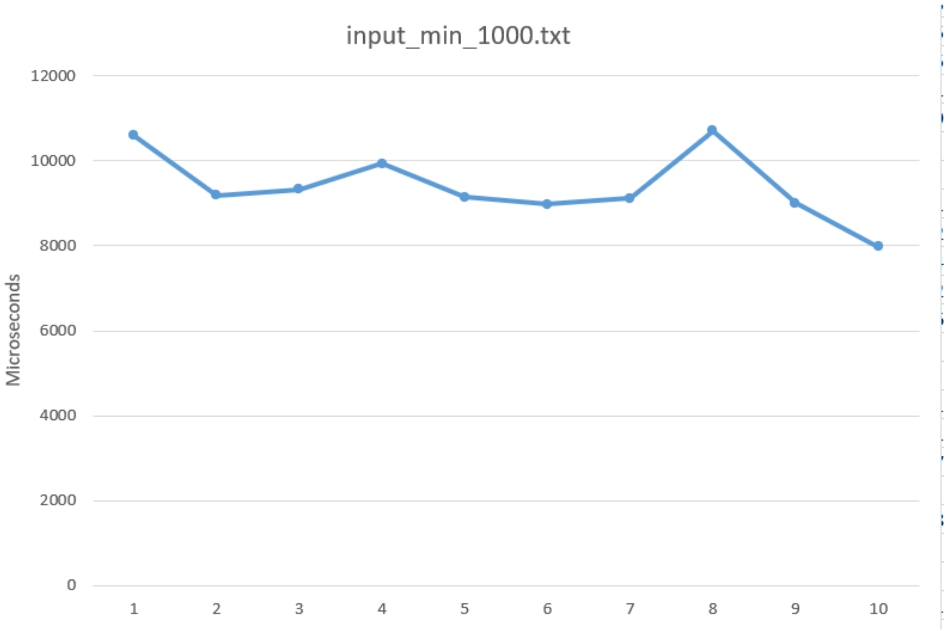


Figure 19: Median 10000

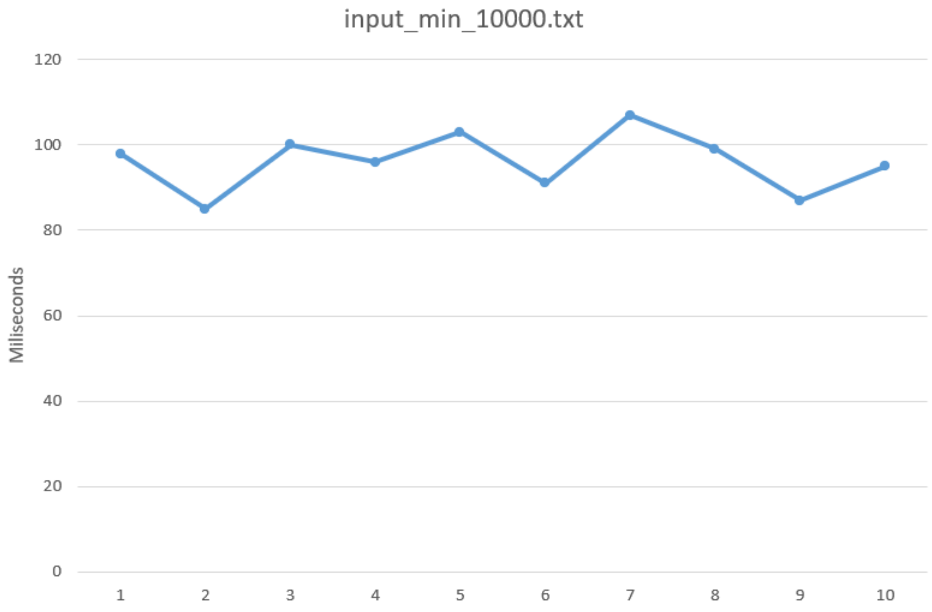
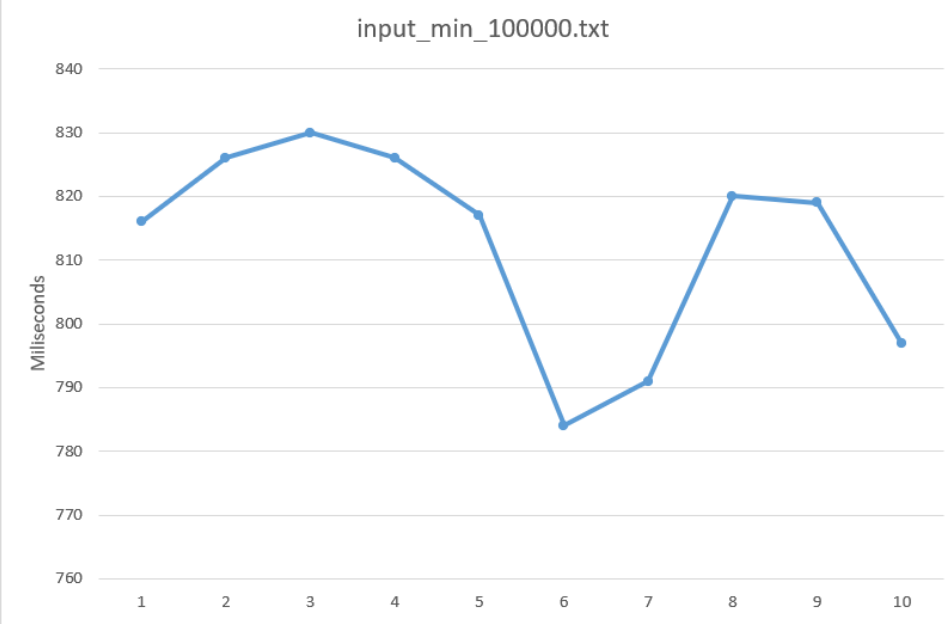


Figure 20: Median 100000



4 Inserting Code Pieces

4.1 Real Code

```
void insertionSort(Value &sample)
{
    int size = sample.values.size();
    int i, j;
    double key;

    bool a_bool = false;

    //we can start the sorting index from a point the last time we sorted t
    //in other words array is sorted up until a point so we will pass thoso
    for (i = SORTED_UNTIL; i < size; i++)
    {
        key = sample.values[i];
        j = i - 1;

        while (j >= 0 && sample.values[j] > key)
        {
            sample.values[j + 1] = sample.values[j];
            j = j - 1;
        }
        sample.values[j + 1] = key;
    }
    SORTED_UNTIL = size - 1;
}
```