REPORT ON PROGRAM 1

SHREYAS ASERKAR: A1175156

Algorithms used:

1.Quick Sort for First name, Last name, Street Address, City, State.
2. Radix Sort(LSD) for Zip Code.

Reason for the using the above algorithms :

1. Quick Sort: It is one of the best methods with a stable performance. Quick Sort is also a cache friendly sorting algorithm as it has good locality of reference when used for arrays, even with large inputs like in this program, and it uses divide and conquer algorithm making it easy to parallelize. It has time complexity of **O(nlog n)** for its best case and average case. Also the space complexity is very less making it one of the best choices to implement. The algorithm has been modified to sort strings using the Ascii values of the strings.

2. Radix Sort(LSD): Radix sort with least significant digits is efficient when sorting strings of fixed length. It can be used to sort data in a large dataset. The complexity of Radix sort(LSD) is **O(nk)** where n is the number of elements to be sorted and k is the size of each element. The complexity is linear which is the best complexity achievable. It's a fast algorithm. The algorithm has been modified to sort strings using the Ascii values of the strings

Output Analysis :

The output was run on inputs of size records 10,100,500,1000,2000,4000 and the output is analyzed and is mentioned below :

Output for 10 records:

Quick Sort
Last name: 13879749 ns

First name: 10351013 ns

State: 5503921 ns

City: 2759443 ns

Street Address: 4741834 ns

Radix Sort(LSD)
Zip Code: 3503078 ns

Total Time Taken: 107467345 ns or 107467 µs

Output for 100 records:

<u>*Quick Sort*</u>
Last name: 34215303 ns

First name: 24597234 ns

State: 18835885 ns

City: 8338198 ns

Street Address: 6250548 ns

<u>*Radix Sort(LSD)*</u>
`Zip Code:` 12265903 ns

<u>*Total Time Taken*:</u> 247740981 ns or 247740 µs

Output for 500 records:

<u>*Quick Sort*</u>
Last name: 82962906 ns

First name: 39905666 ns

State: 41288595 ns

City: 29889481 ns

Street Address: 32174692 ns

<u>*Radix Sort(LSD)*</u>
Zip Code: 29802674 ns

Total Time Taken: 587404400 ns or 587404 µs

Output for 1000 records:

<u>*Quick Sort*</u>
Last name: 159440927 ns

First name: 68608925 ns

State: 67285863 ns

City: 46748043 ns

Street Address: 65113544 ns

Radix Sort(LSD)
Zip Code: 57114023 ns

Total Time Taken: 881914377 ns or 881914 µs

Output for 2000 records:

Quick Sort
Last name: 183143576 ns

First name: 80140175 ns

State: 94223044 ns

City: 89598741 ns

Street Address: 144057660 ns

Radix Sort(LSD)
Zip Code: 91046241 ns

Total Time Taken: 1338962546 ns or 1338962 µs

Output for 4000 records:

Quick Sort
Last name: 324045816 ns

First name: 226561729 ns

State: 248604777 ns

City: 164034868 ns

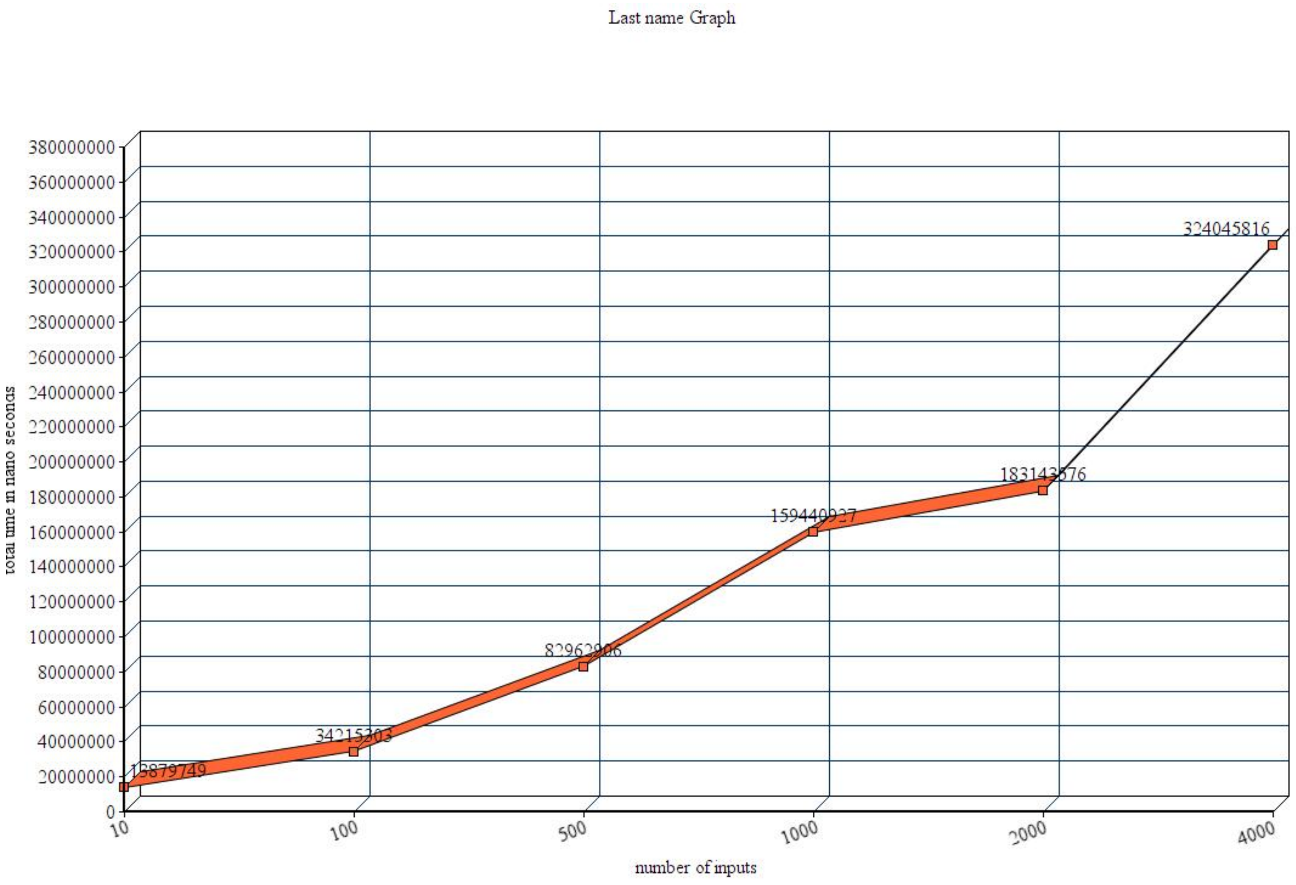
Street Address: 244629605 ns

Radix Sort(LSD)
Zip Code: 133865295 ns

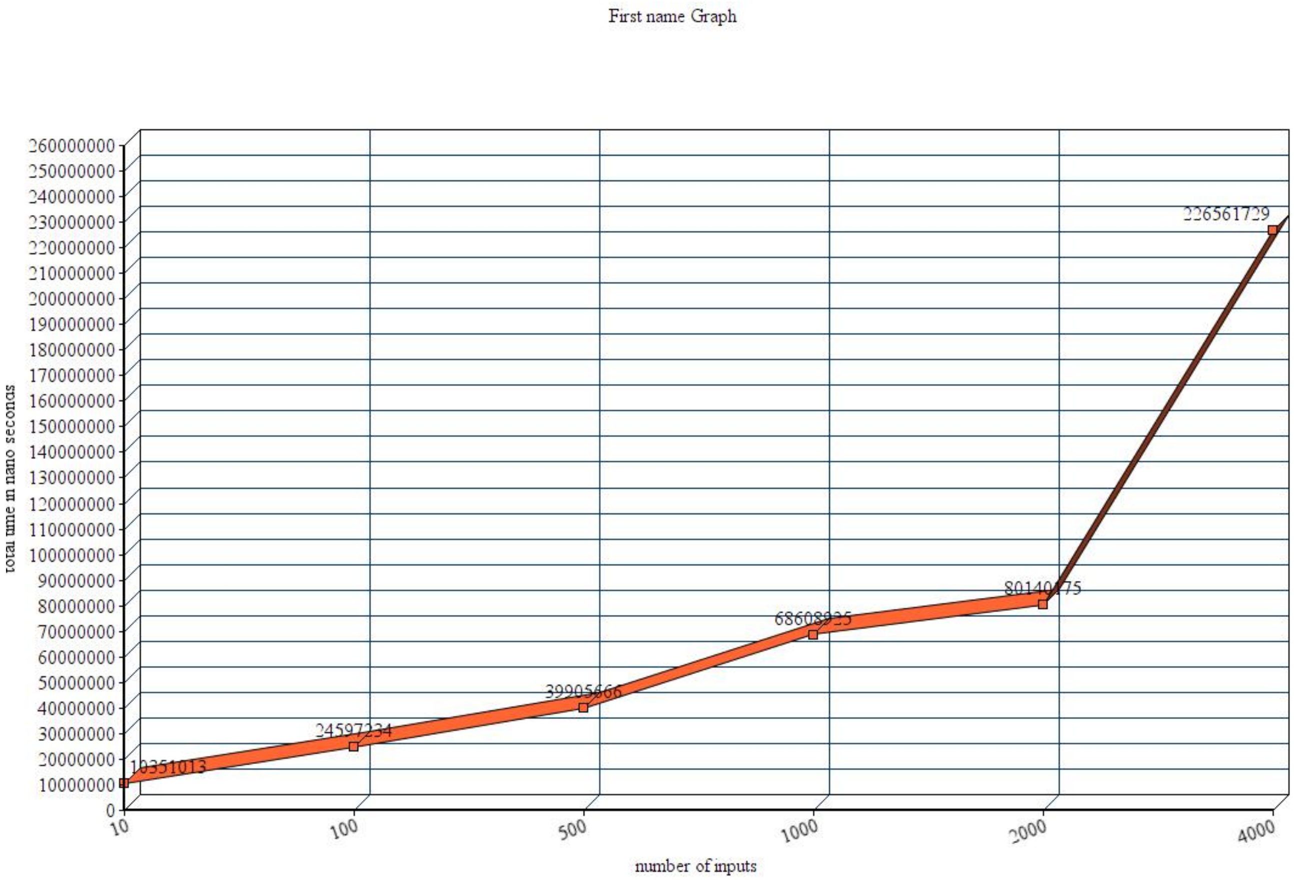
Total Time Taken: 2361781729 ns or 2361781 µs

PERFORMANCE GRAPHS

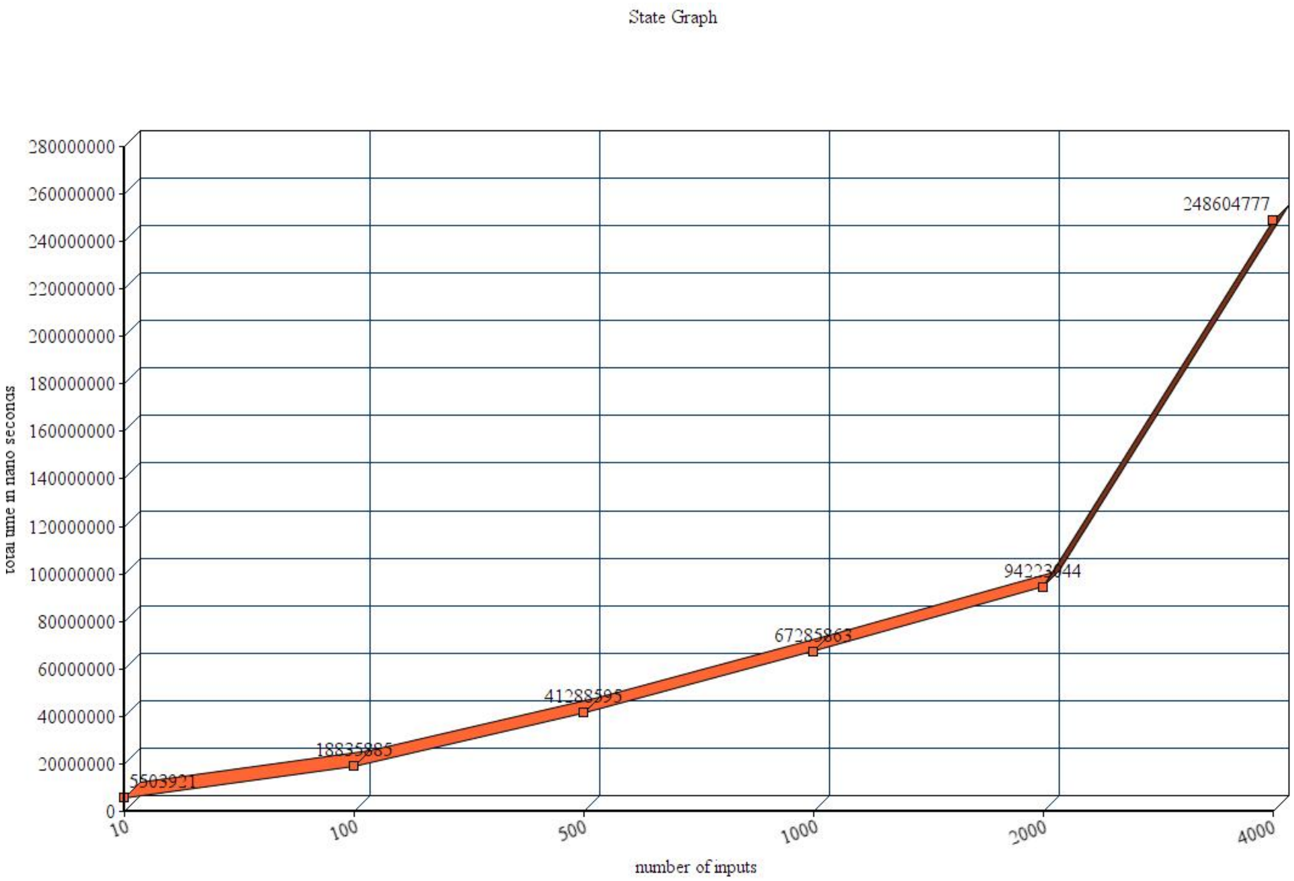
Last Name graph:



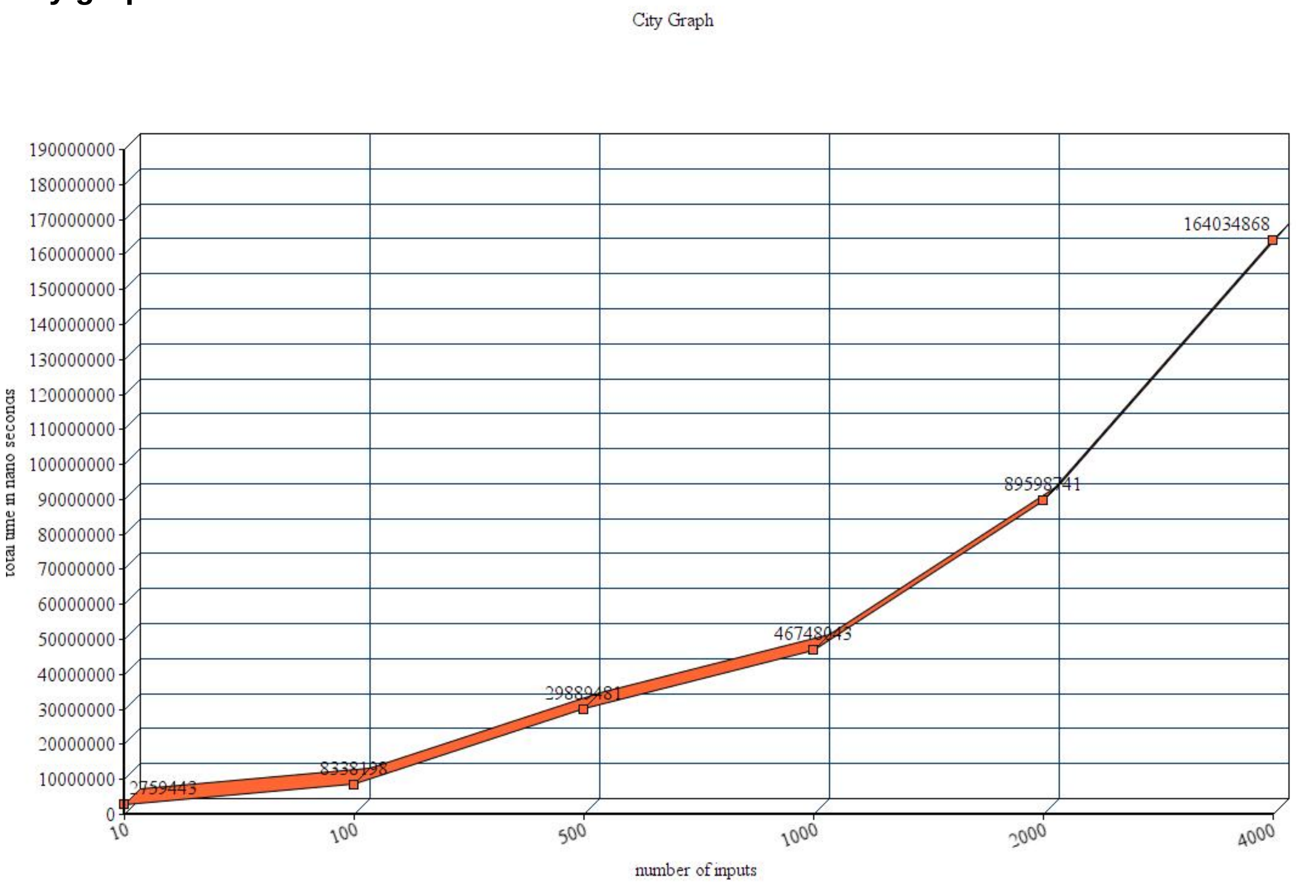
First Name graph:



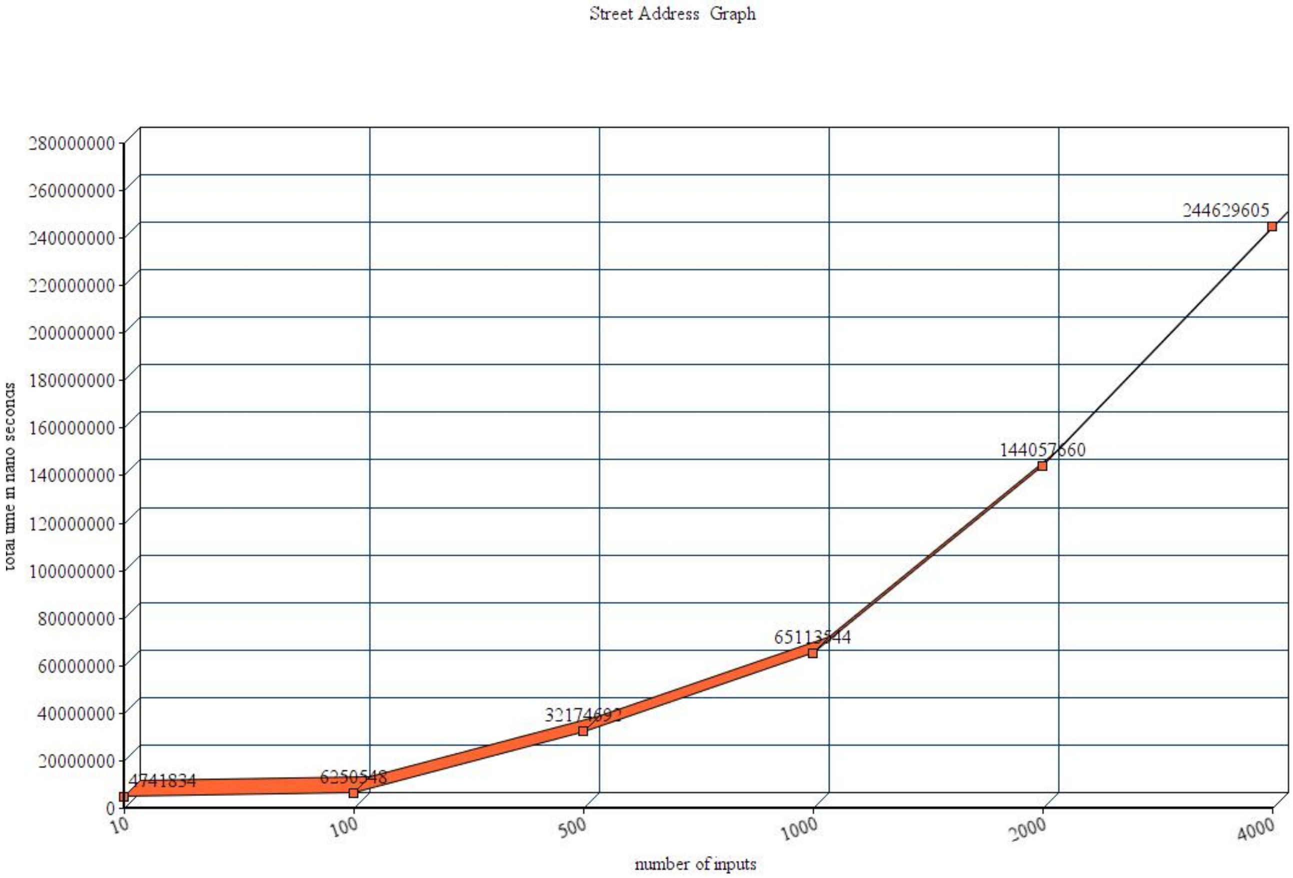
State graph :


State Graph
total time in nano seconds
280000000
260000000
240000000
220000000
200000000
180000000
160000000
140000000
120000000
100000000
80000000
60000000
40000000
20000000
0
5503921
18835885
41288595
67285863
94223044
248604777
10
100
500
1000
2000
4000
number of inputs

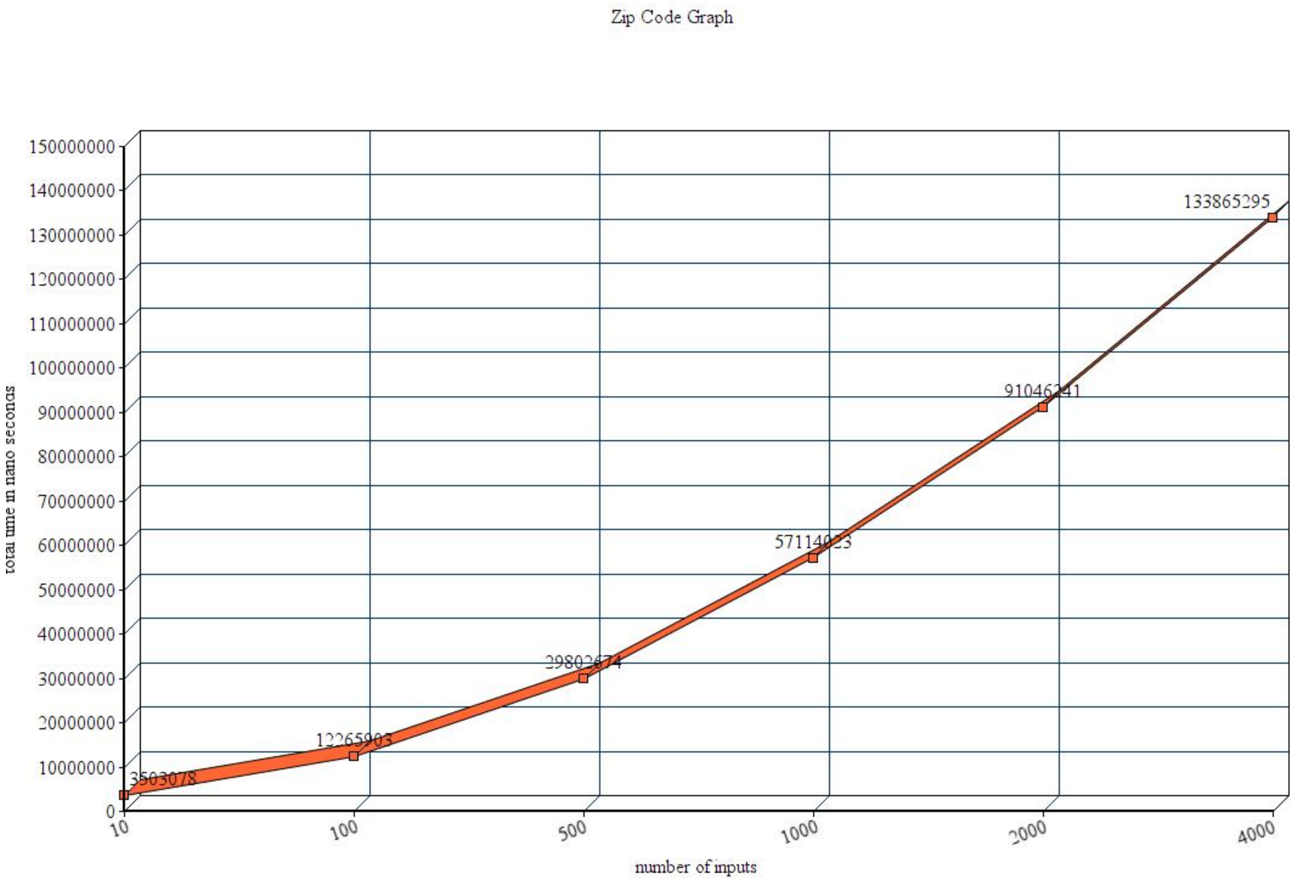
City graph:


City Graph
190000000
180000000
170000000
160000000
150000000
140000000
130000000
120000000
110000000
100000000
90000000
80000000
70000000
60000000
50000000
40000000
30000000
20000000
10000000
0
total time in nano seconds
2759443
8338198
29889481
46748043
89598741
164034868
10
100
500
1000
2000
4000
number of inputs

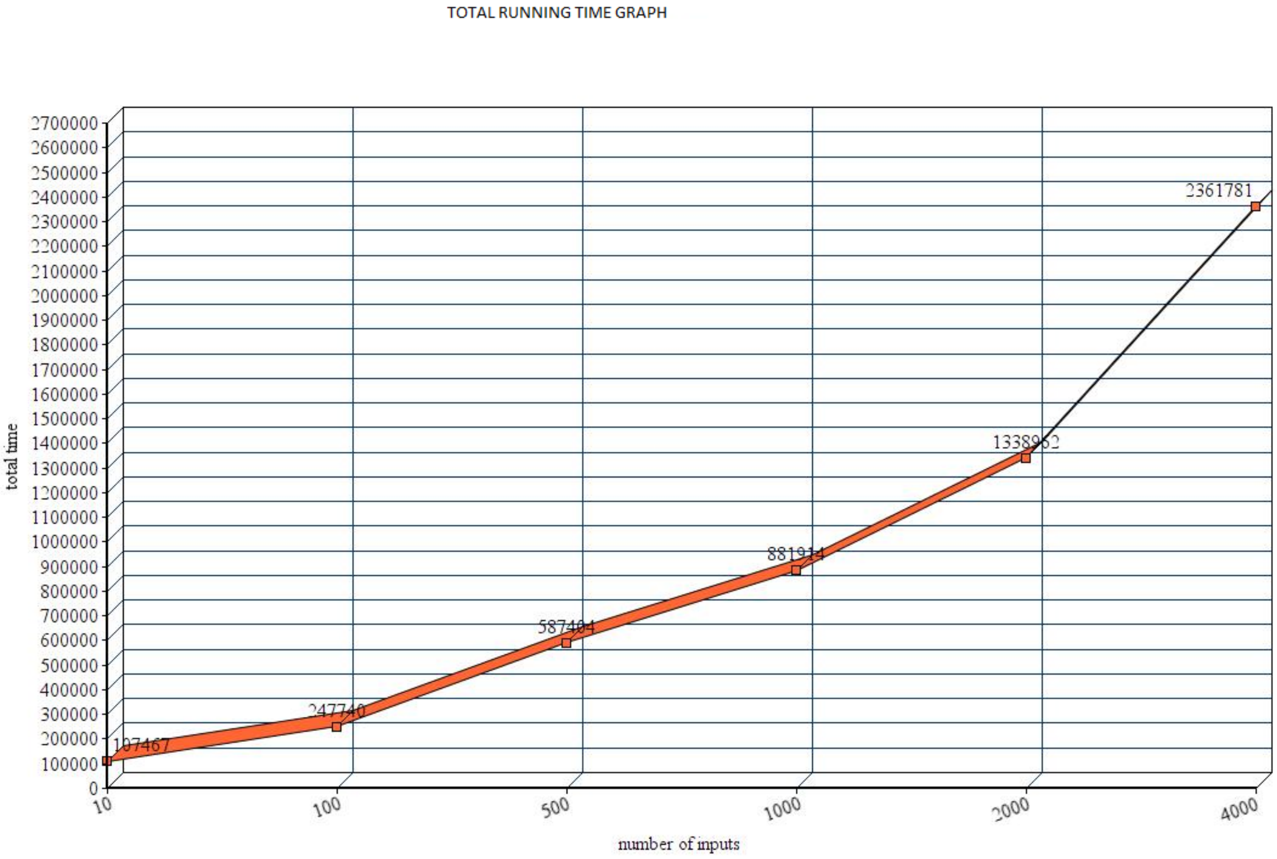
Street Address Graph:



Zip code graph :



Total running time graph:


TOTAL RUNNING TIME GRAPH
total time
number of inputs