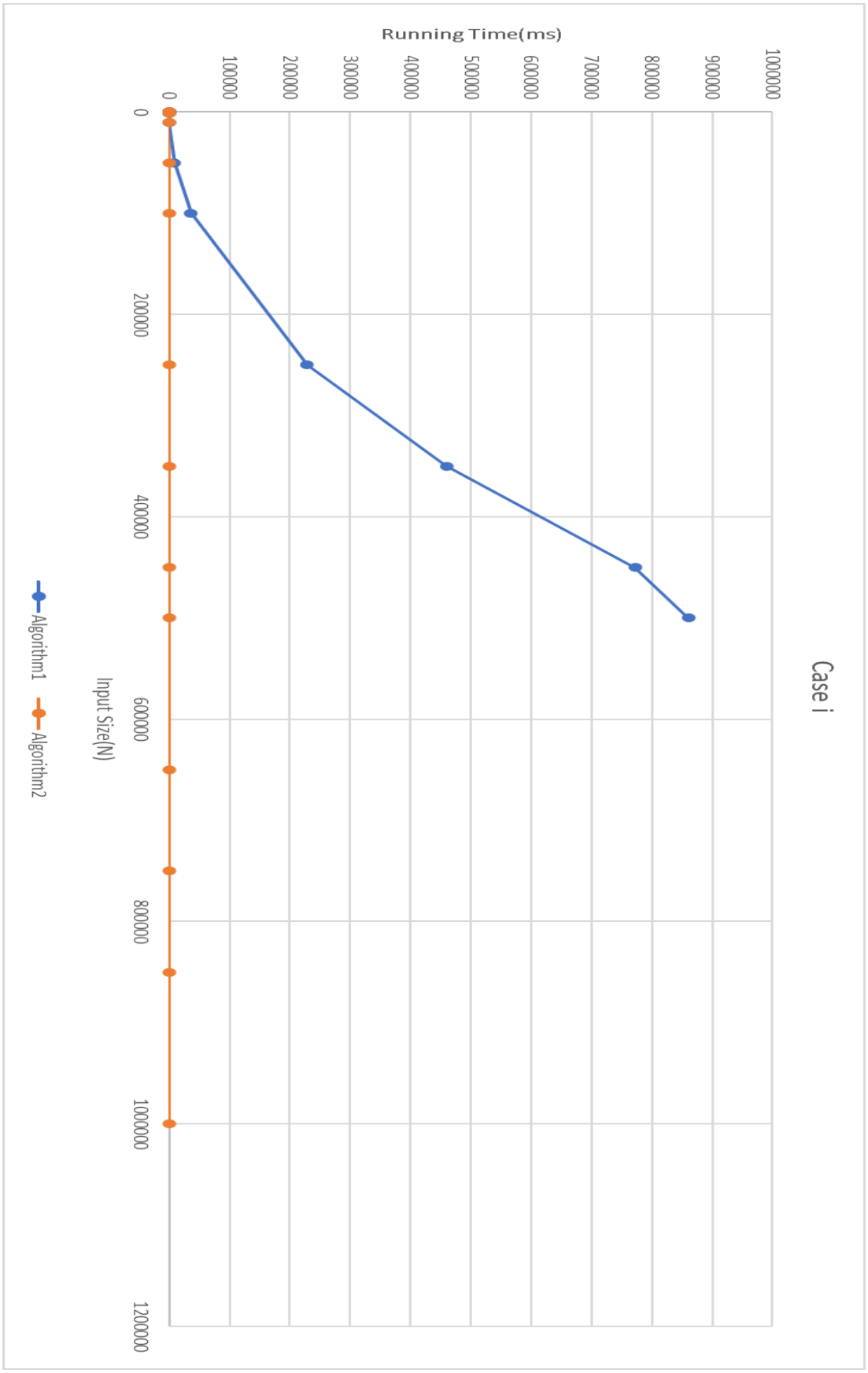
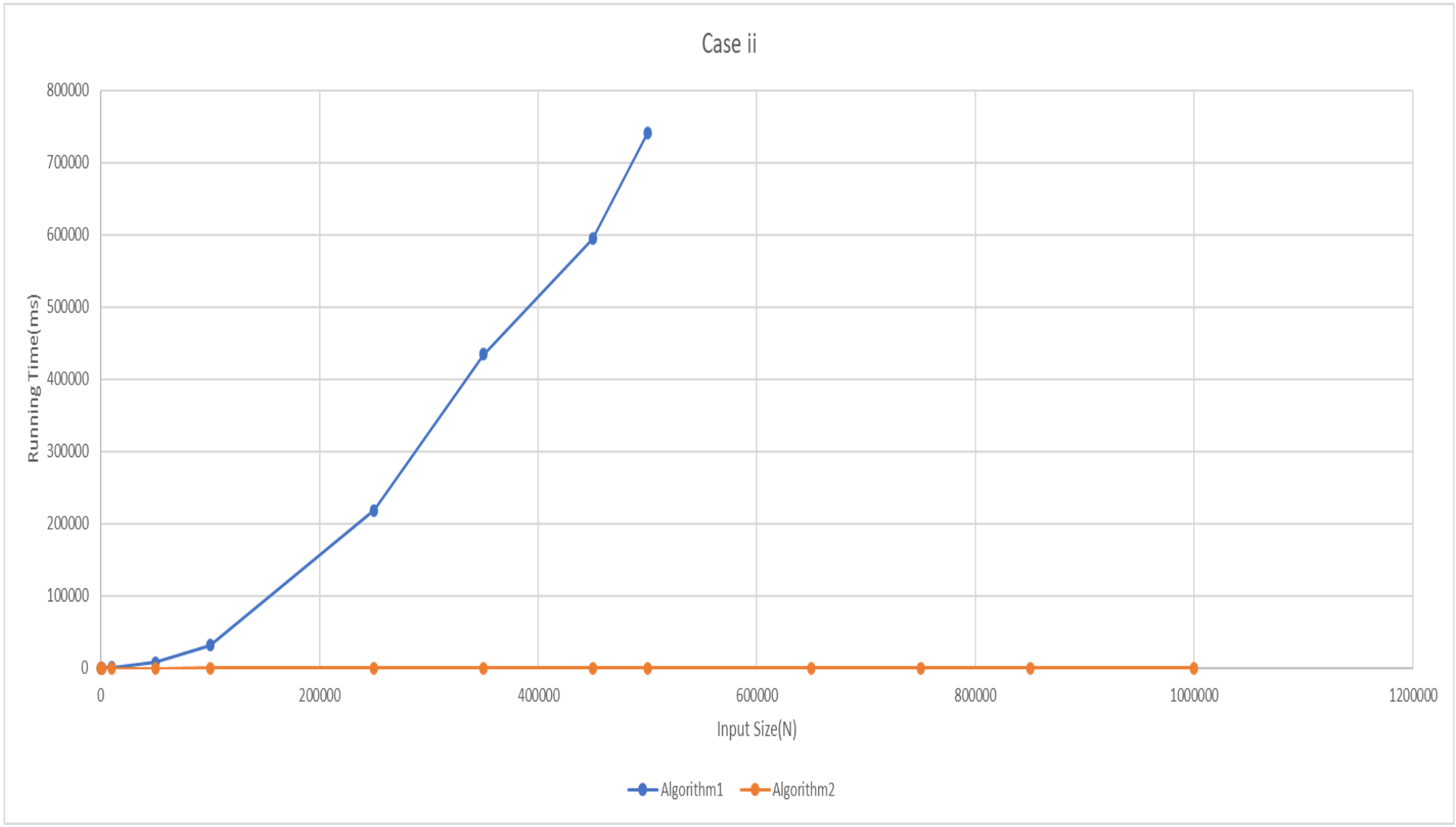
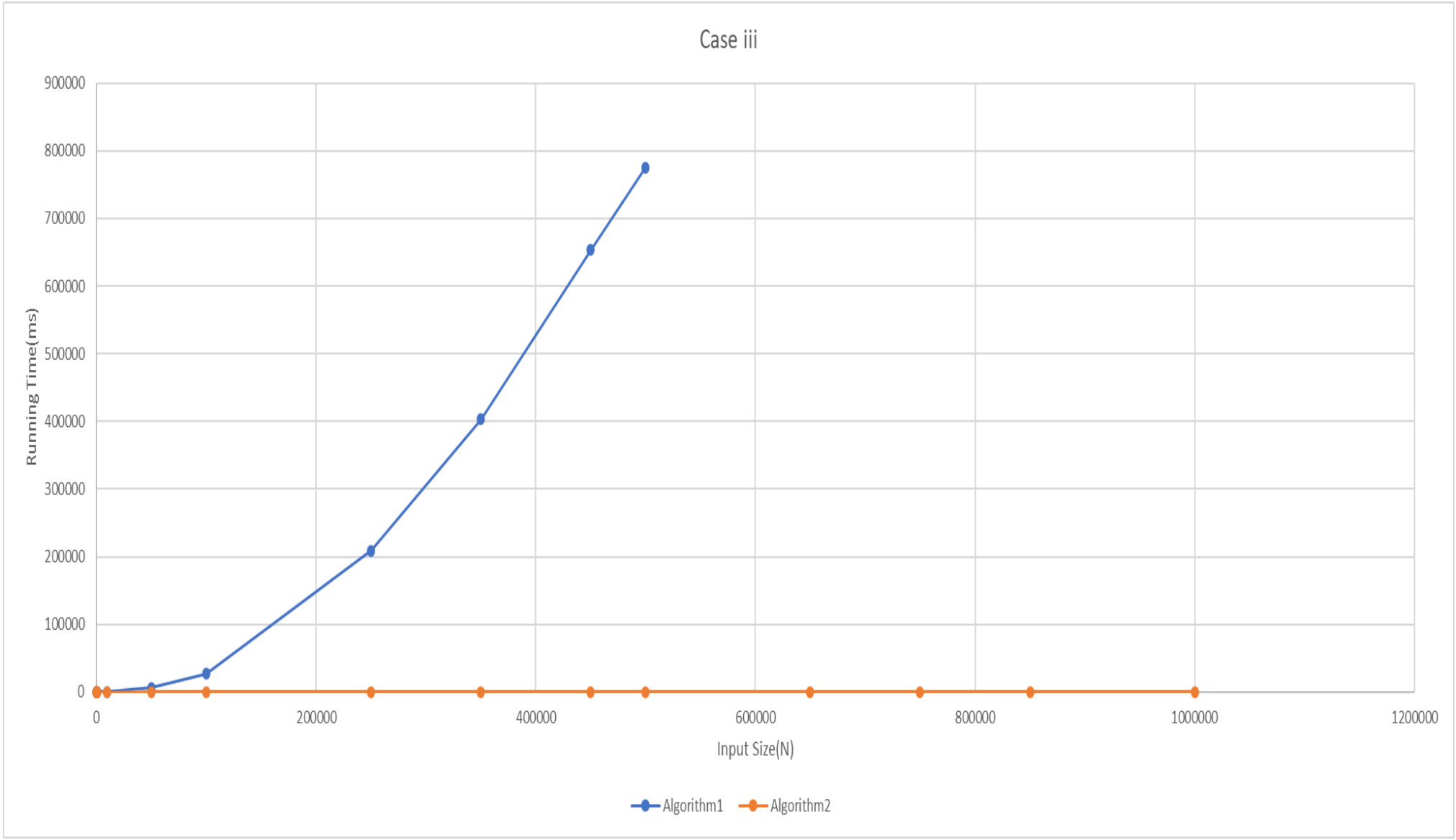
Arman Engin SUCU/21801777/Sec-02

PLOTS:




Case ii
800000
700000
600000
500000
400000
300000
200000
100000
0
Running Time(ms)
0
200000
400000
600000
800000
1000000
1200000
Input Size(N)
Algorithm1
Algorithm2


Case iii
900000
800000
700000
600000
500000
400000
300000
200000
100000
0
Running Time(ms)
0
200000
400000
600000
800000
1000000
1200000
Input Size(N)
Algorithm1
Algorithm2

WORST-CASE, BEST-CASE, AND AVERAGE-CASE ANALYSES

Algorithm1:

- Worst-Case: When the second arrays's all item's are same and they are all bigger than first array's all items except the last item in the first array.
  f(N) = O(N) +O(N^3) = O(N^3)
- Average-Case: f(N) = O(N^2)
- Best-Case: When the second array has 1 item and it is bigger than first array's all items.
  F(N) = O(N) + O(N) = O(N)

Algorithm2:

- Worst-Case:When first array's all itmes are bigger than second array, algorithm has to check 2 more if statements.
  f(N) = O(N)
- Average-Case: f(N) = O(N)
- Best-Case: When second array's all itmes are bigger than first array's all items.
  f(N) = O(N)

➢ Worst-Case, Average-Case, Best-Case all of them have O(N) time complexity because in every condition algorithm has to complete the while statement(because there is no break or return in the statement, every condition are same)

COMPUTER SPECIFICATIONS

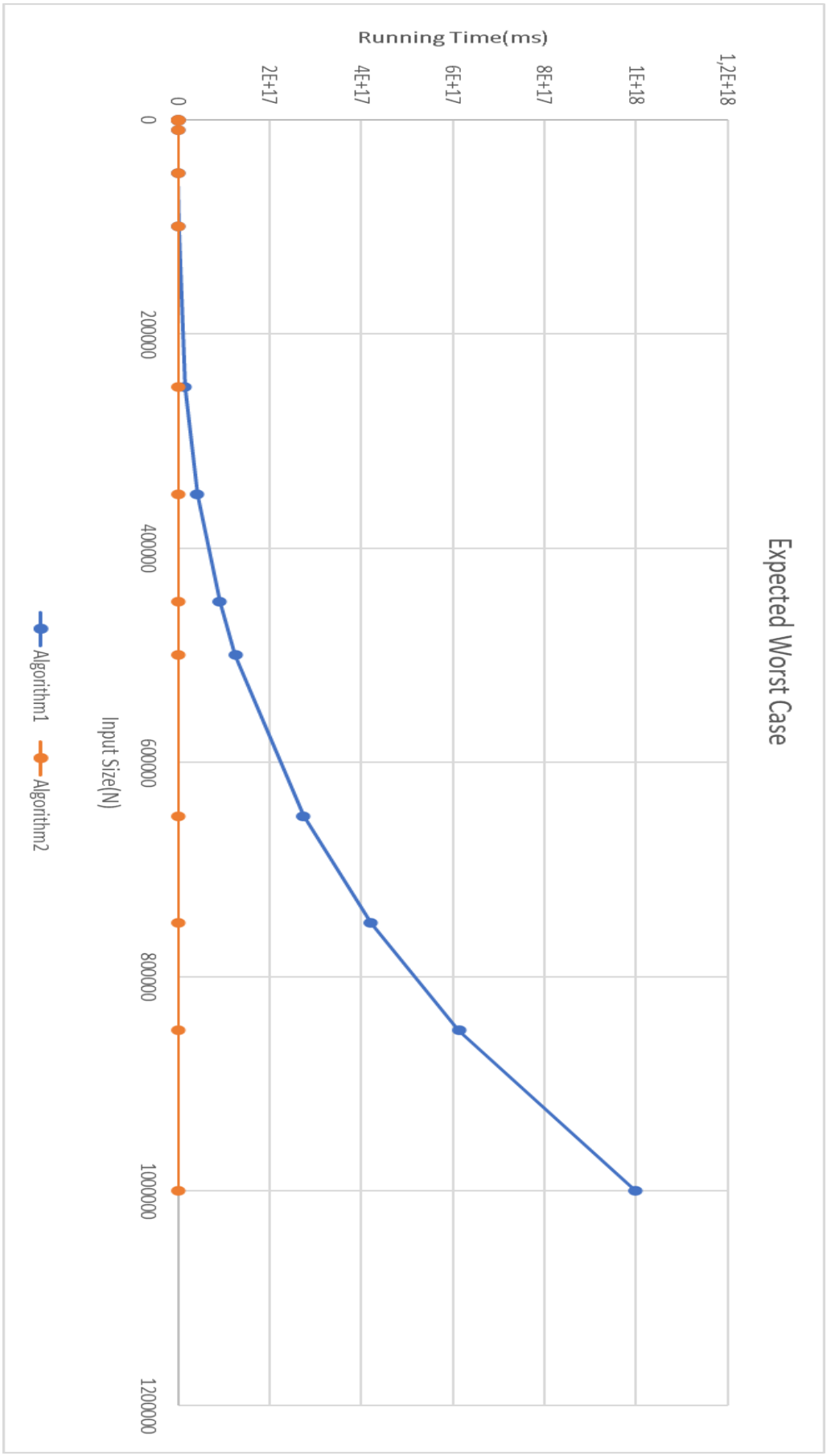
CPU: Intel® Core™ i7-7700HQ CPU @ 2.80GHz

2.80GHz

RAM: 16.0GB

OPERATING SYSTEM: Windows 10 Pro

EXPECTED WORST-CASE CHARTS



DISCUSSION:

To make it clear, Algorithm1 slower than Algorithm2.

The worst-case growth rate from the above plot and theoretical data from step 3 are coherent. From the above plot, it can be seen that algorithm1 has O(N^3) growth rate as expected and algorithm2 has O(N) growth rate. Because operations for O(N^3) is very high O(N) looks like a straight line with 0 tangents which should have a tangent bigger than 0.

Algorithm1 is slower in case1 compare to case2. However, my expectation was reverse, because in case2 algorithm1 runs 2 nested for loops however in case 2 it is 3 nested for loop. The underlying reason for this problem is my break statement. Although in case 2 it is 3 for loops when the program finds the wanted case it breaks, however in case 2 break does not work. Therefore, it totally runs 2 nested for loops.

Moreover, in Linux machine running algorithms take more time compare to run them in visual studio

TABLES

These are the tables to compare the datas clearly.

Case i

Size	Algorithm1(ms)	Algorithm2(ms)
10	0	0
100	0	0
1000	3	0
10000	394	0
50000	9445	1
100000	36649	1
250000	228955	2
350000	459836	3
450000	772609	4
500000	861064	4
650000		5
750000		8
850000		6
1000000		8

Case ii

Size	Algorithm1(ms)	Algorithm2(ms)
10	0	0
100	0	0
1000	3	0
10000	318	0
50000	7372	0
100000	31464	1
250000	218253	4
350000	434657	3

450000	594293	4
500000	740803	6
650000		6
750000		6
850000		8
1000000		10

Case iii

Size	Algorithm1(ms)	Algorithm2(ms)
10	0	0
100	0	0
1000	3	0
10000	300	0
50000	6457	1
100000	26660	2
250000	208860	5
350000	403293	7
450000	653244	9
500000	775596	9
650000		8
750000		10
850000		9
1000000		8

WorstCase

Size	Algorithm1(operations)	Algorithm2(operations)
10	1000	10
100	1000000	100
1000	1000000000	1000
10000	1E+12	10000
50000	1,25E+14	50000
100000	1E+15	100000
250000	1,5625E+16	250000
350000	4,2875E+16	350000
450000	9,1125E+16	450000
500000	1,25E+17	500000
650000	2,74625E+17	650000
750000	4,21875E+17	750000
850000	6,14125E+17	850000
1000000	1E+18	1000000