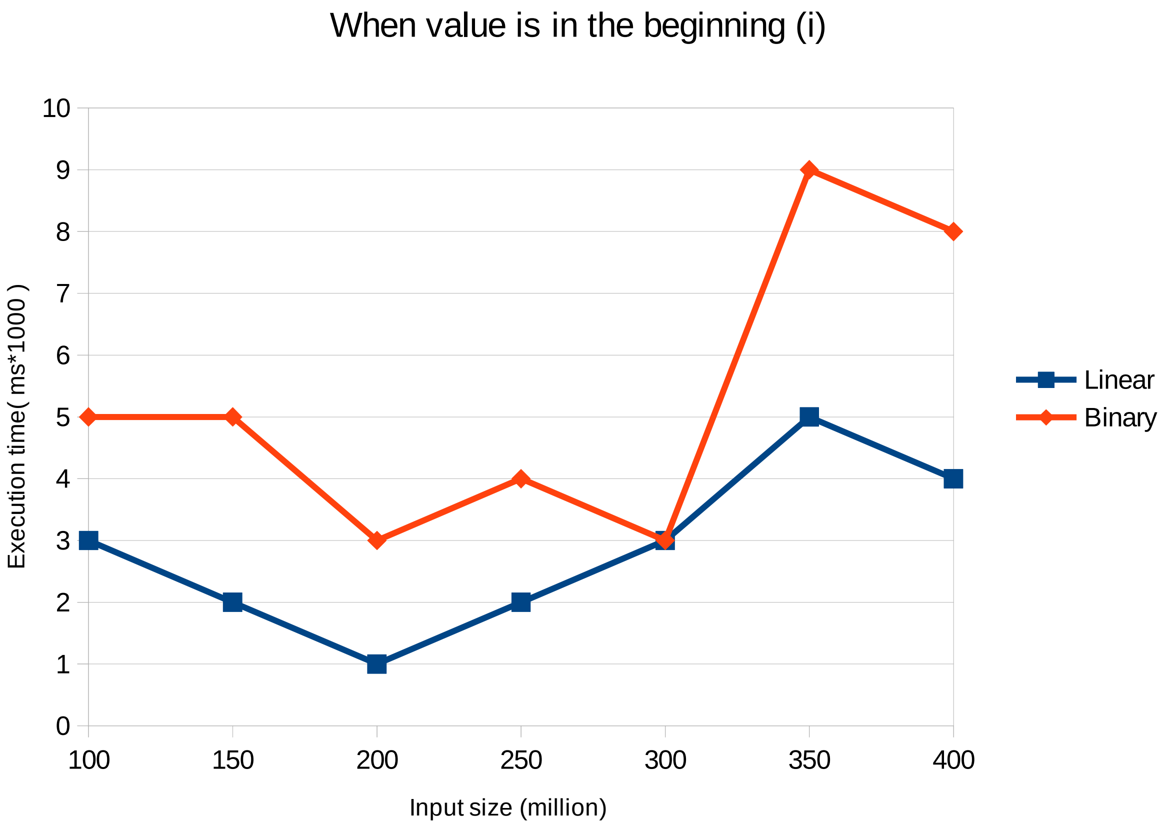
CS201 HW2

Berdan Akyürek
21600904
CS201-2

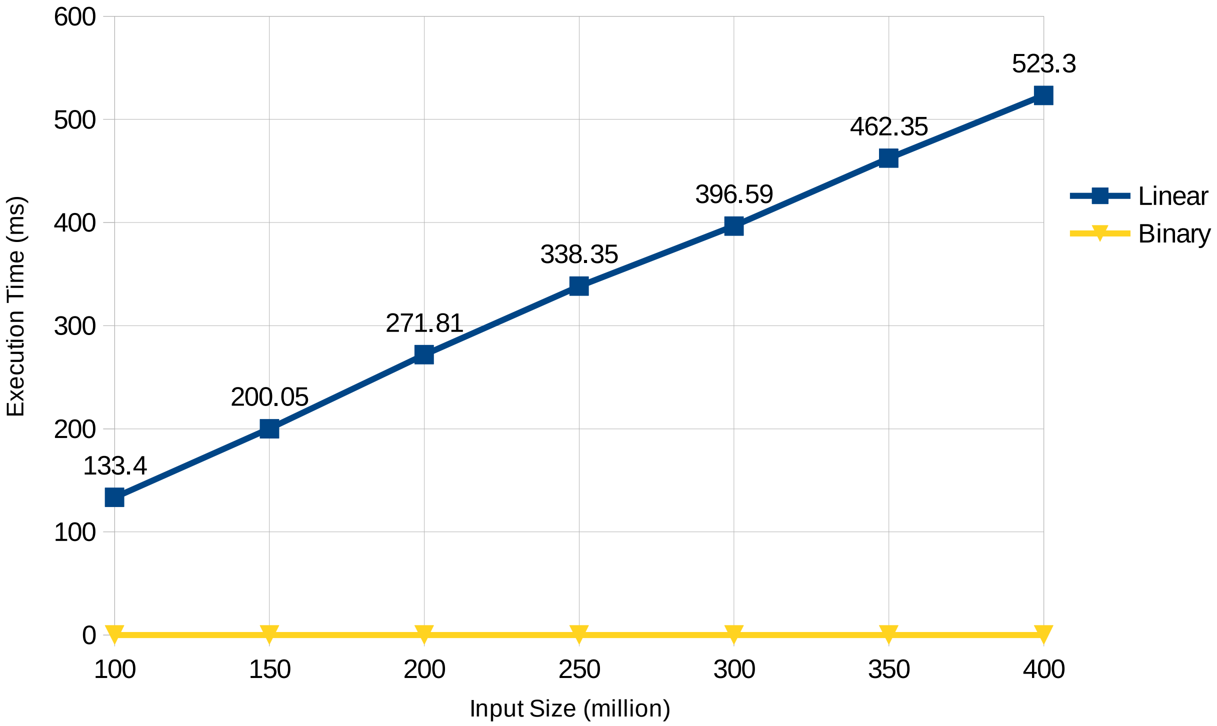
Experiment Results:

N (million)	Execution Time (ms)							
	(i)		(ii)		(iii)		(iv)	
	Linear	Binary	Linear	Binary	Linear	Binary	Linear	Binary
100	0.003	0.005	133.404	0.002	271.274	0.002	256.666	0.002
150	0.002	0.005	200.046	0.002	406.981	0.002	398.088	0.002
200	0.001	0.003	271.805	0.002	528.813	0.002	538.214	0.002
250	0.002	0.004	338.349	0.002	658.398	0.002	669.635	0.002
300	0.003	0.003	396.586	0.003	792.005	0.001	802.146	0.002
350	0.005	0.009	462.35	0.009	932.684	0.002	934.53	0.003
400	0.004	0.008	523.296	0.009	1091.22	0.001	1102.88	0.009

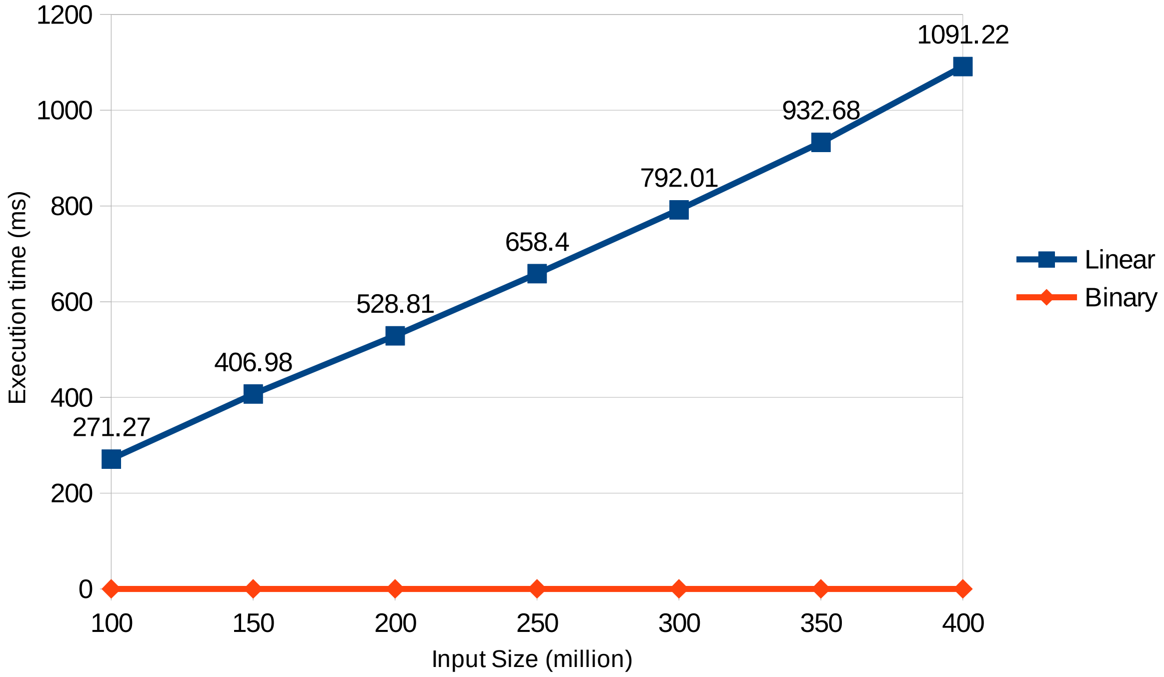
I couldn’t run the program for input sizes greater than 400 million due to technical problems.

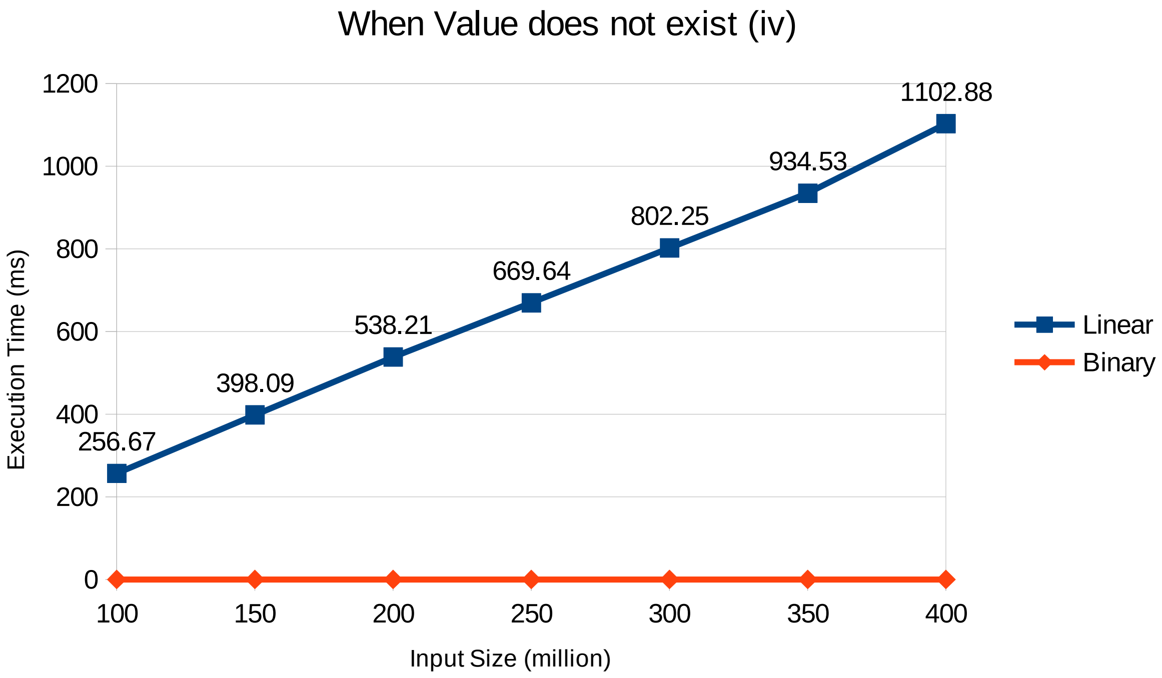


When Value is in the middle (ii)



When Value is in the end (iii)





Best, average, worst case:

Linear Search:

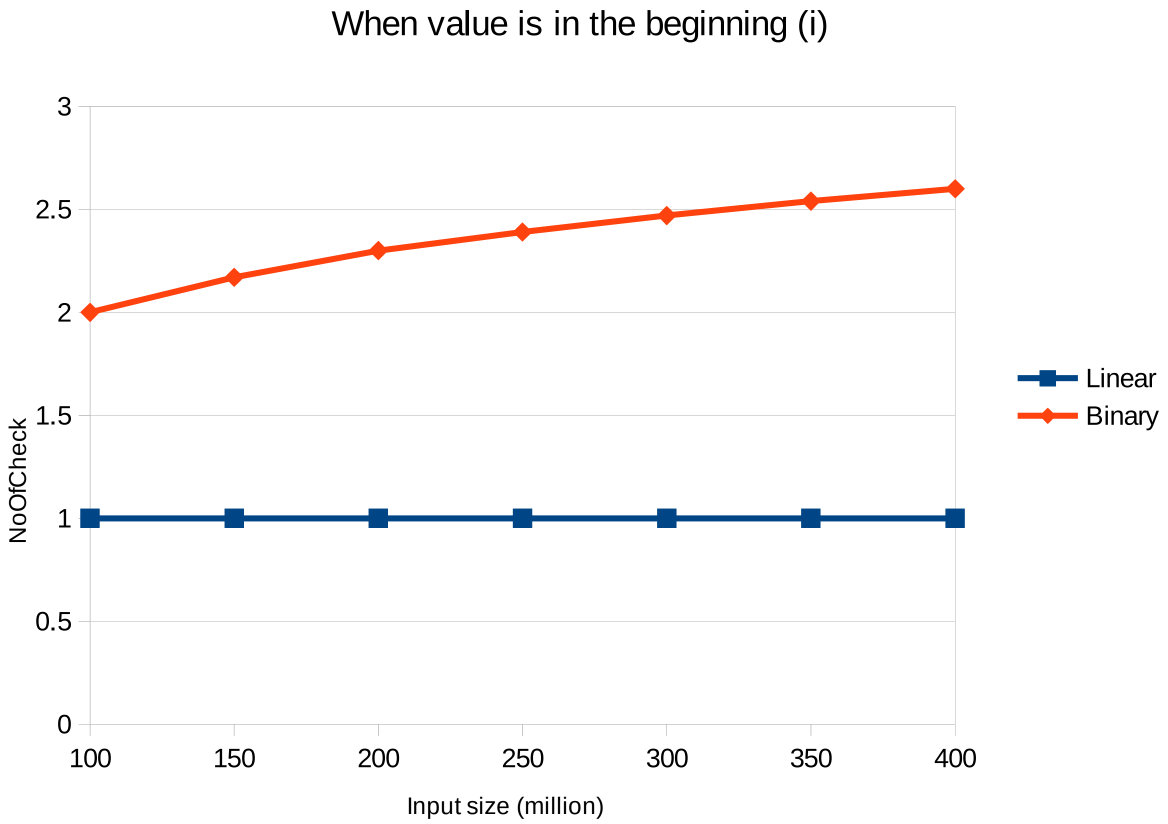
Best case (when value is around beginning): O(1)
Average case (when the value is around end): O(ln)
Worst case (when the value does not exist): O(n)

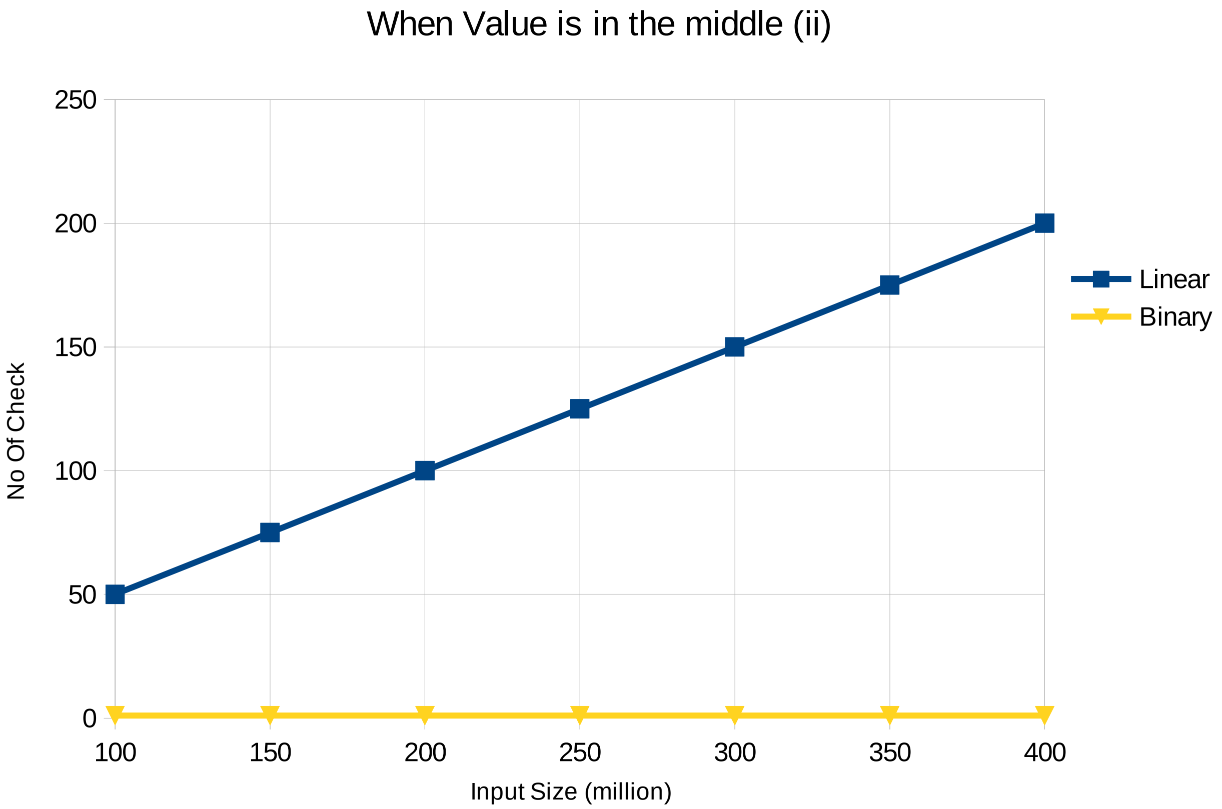
Binary Search:

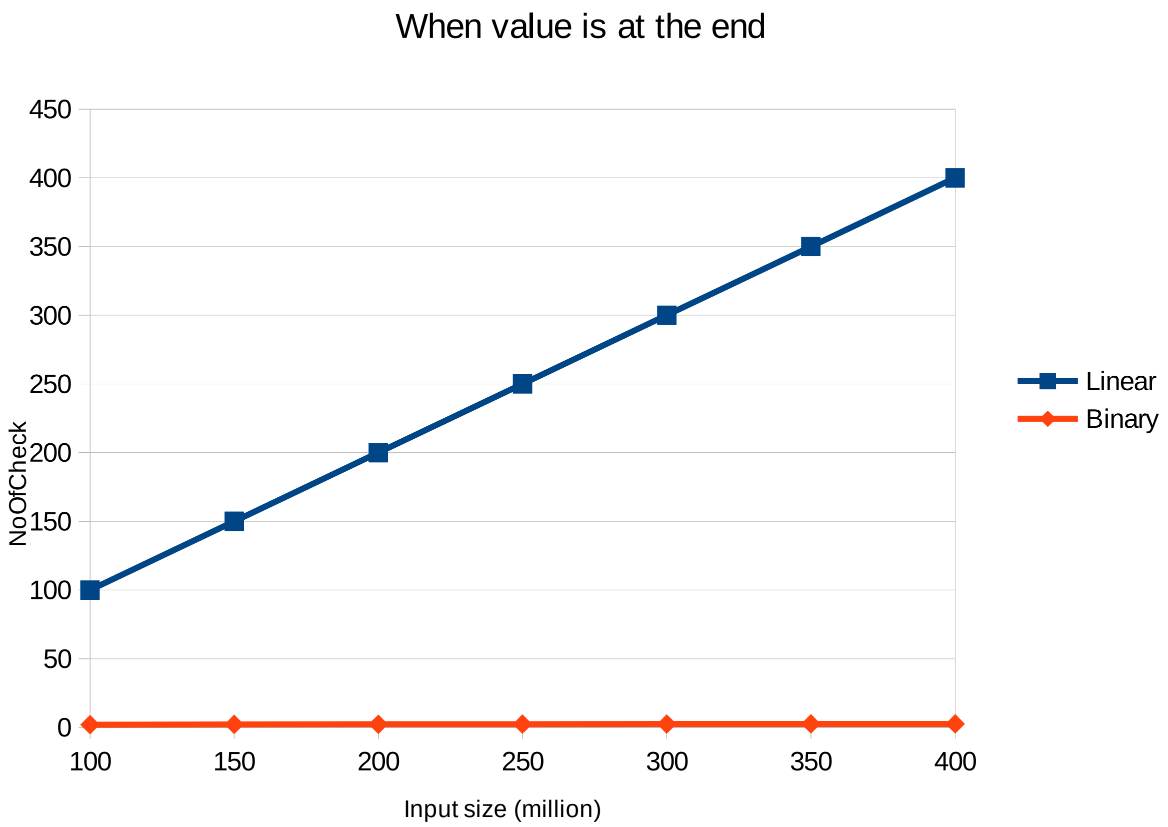
Best case(when the value is around middle): O(1)
Average case (when the value is around beginning or end): O(log n)
Worst case (when the value does not exist): O(log n)

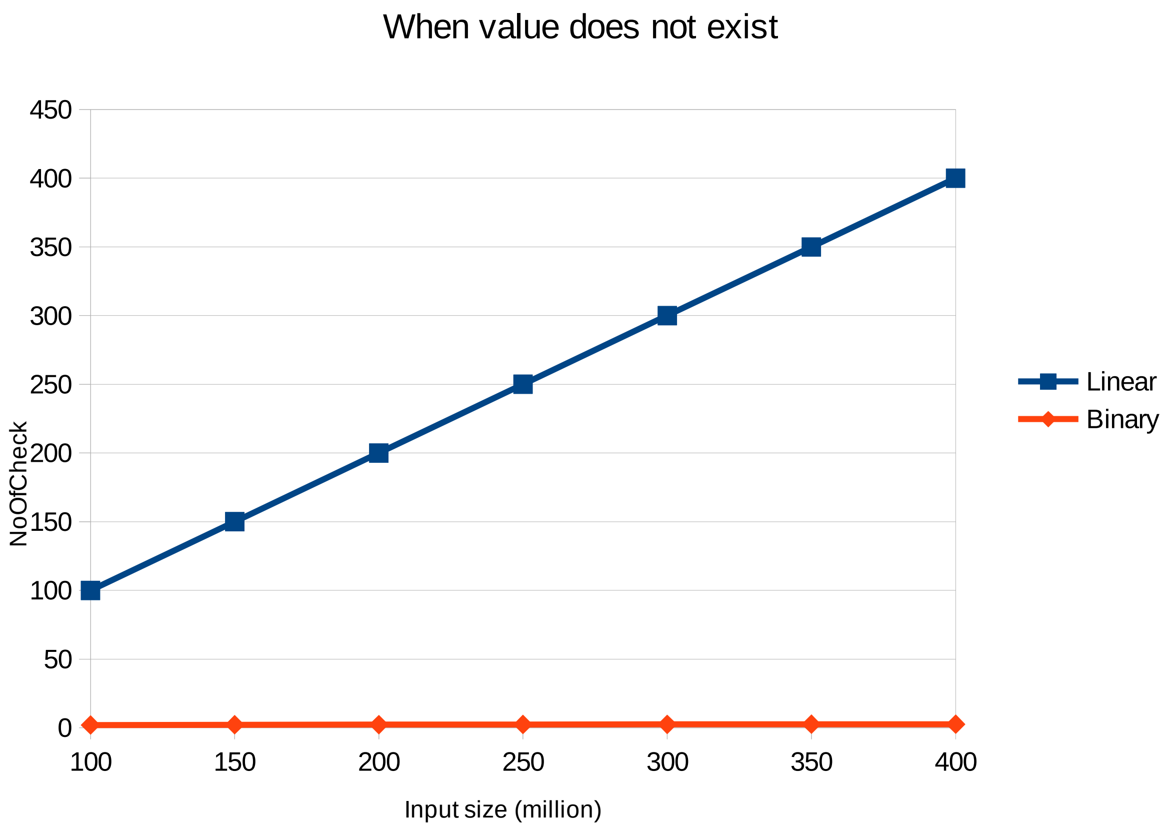
Specifications of the computer that is used for this assignment:
RAM: 4 GB
CPU: Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz
OS: Ubuntu 18.04.3 LTS 64-bit
Compiler: g++

Theoretical Results






When value is at the end
NoOfCheck
450
400
350
300
250
200
150
100
50
0
100
150
200
250
300
350
400
Input size (million)
Linear
Binary


When value does not exist
NoOfCheck
450
400
350
300
250
200
150
100
50
0
100
150
200
250
300
350
400
Input size (million)
Linear
Binary

Comparison of Expected and Experimental Results

The results and graphs were close to expected results for (ii), (iii) and (iv) but it was different for (i) because the graph of that is very specifically generated to see the difference. As seen, most of the time Binary Search is a more efficient algorithm than Linear Search algorithm. But if the searched value is close to the beginning, Linear Search is a better algorithm. The difference between these two algorithms’ efficiencies are getting greater when input size getting greater. So while input size getting greater, the importance of choosing a better algorithm increases too. In other words, if someone is working with big input sizes, he/she needs to find the best algorithm.