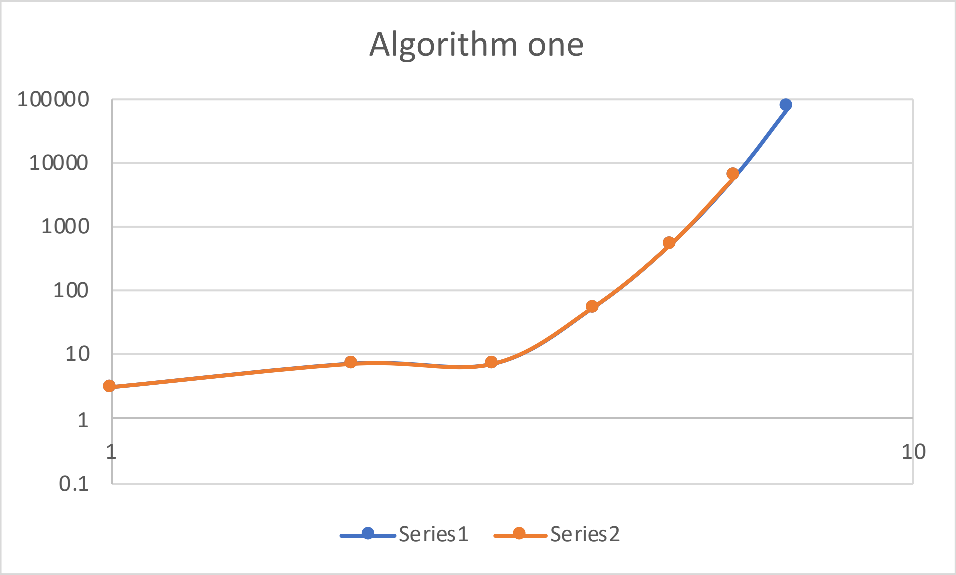
Seth Jacobs

We were given four algorithms and tasked with running experiments to determine the "Big O" time for each algorithm. For my experiment I ran each algorithm in a for loop that doubles the number of inputs each time around. I ran the code with multiple number ranges to determine what numbers are too small to be irrelevant for the test, and what number would be too big as to not allow the test to run in a fair amount of time. After determining the correct input sizes, I ran the tests 3 times and looked at how long in milliseconds it took each input size to complete. The tables below show for each algorithm, the average time, in milliseconds, rounded to the nearest whole number that it took for the algorithms to run. It then shows the ratio of the difference in time between an input size, and the input size half its size. Based on the limit of the ratios I should be able to determine the time it will take to run 2n inputs, n being the last input size calculated.

Algorithm one

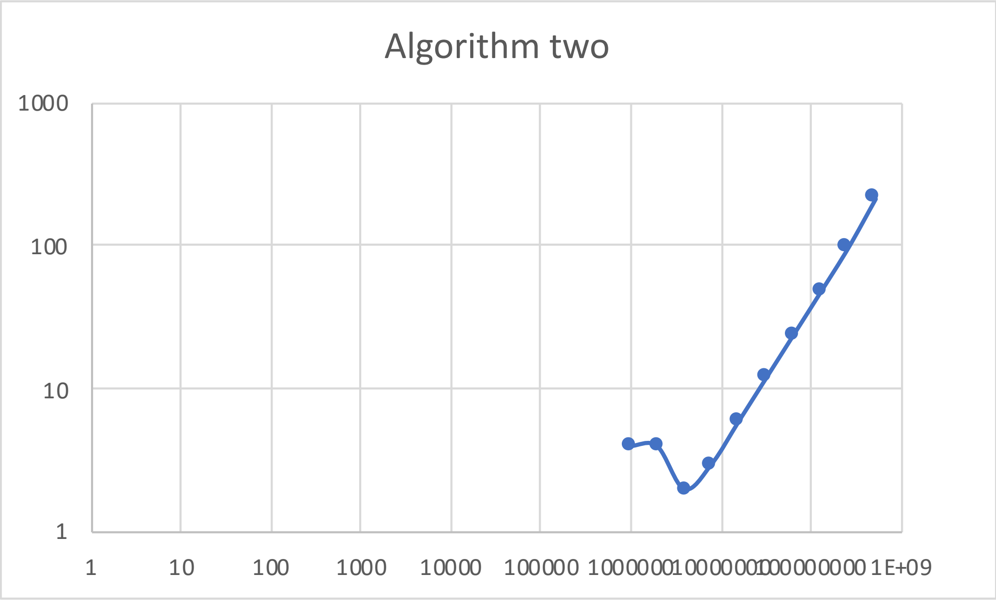
N	Time milliseconds	Ratio
100	**3**	
200	**7**	2.3
400	**7**	1
800	**54**	7.7
1600	**534**	9.7
3200	**6009**	11.44
6400	**72505**	11.58



O(n^3) Since the limit of the ratios approaches 8, this algorithm is cubic. (The average of the ratios is 7.88)

Algorithm Two

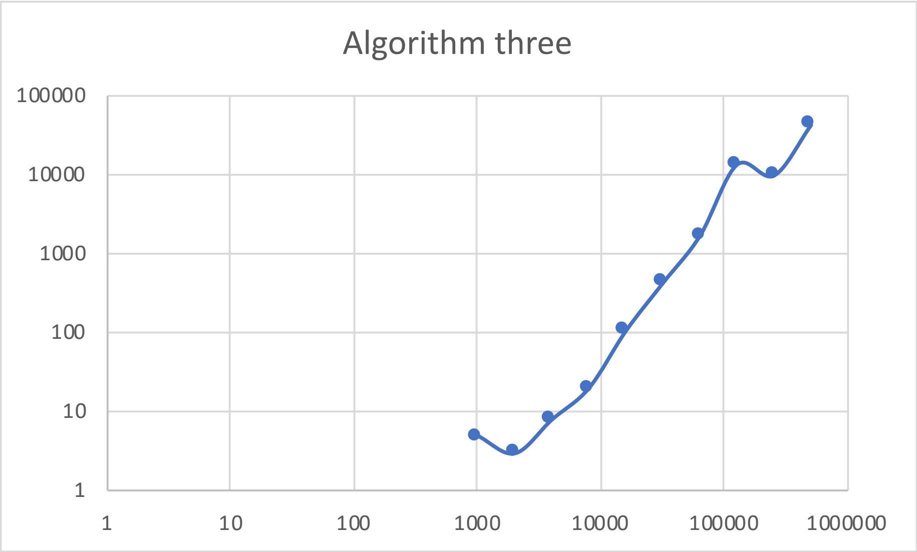
N	Time milliseconds	Ratio
1000000	**4**	
2000000	**4**	1
4000000	**2**	.5
8000000	**3**	1.5
16000000	**6**	2
32000000	**12**	2
64000000	**24**	2
128000000	**48**	2
256000000	**97**	2.02



O(n)
Since the limit of the ratios approaches 2, this algorithm is linear. If that is true then the input of size 512000000 which is double the last test should be around 194, double the last tests time. It was 213. Which is 97*2.2. That is very close to the ratio of 2.

Algorithm three

N	Time milliseconds	Ratio
1000	**5**	
2000	**3**	.6
4000	**8**	2.6
8000	**20**	2.5
16000	**106**	5.3
32000	**427**	4.02
64000	**1688**	3.95
128000	**13360**	7.9
256000	**9753**	7.9
512000	**42275**	4.33

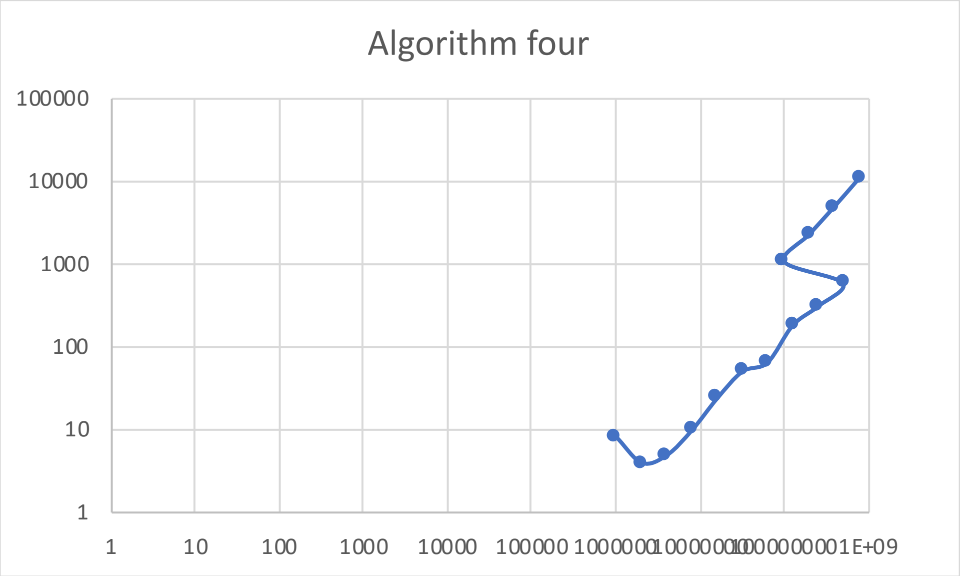


O(n^2)

Since the limit of the ratios approaches 4, this algorithm is quadratic. If that is true, the result of 1024000 should be 42275*4. It was 191595. Which is 42275*4.5. Close to the ratio of 4.

Algorithm four

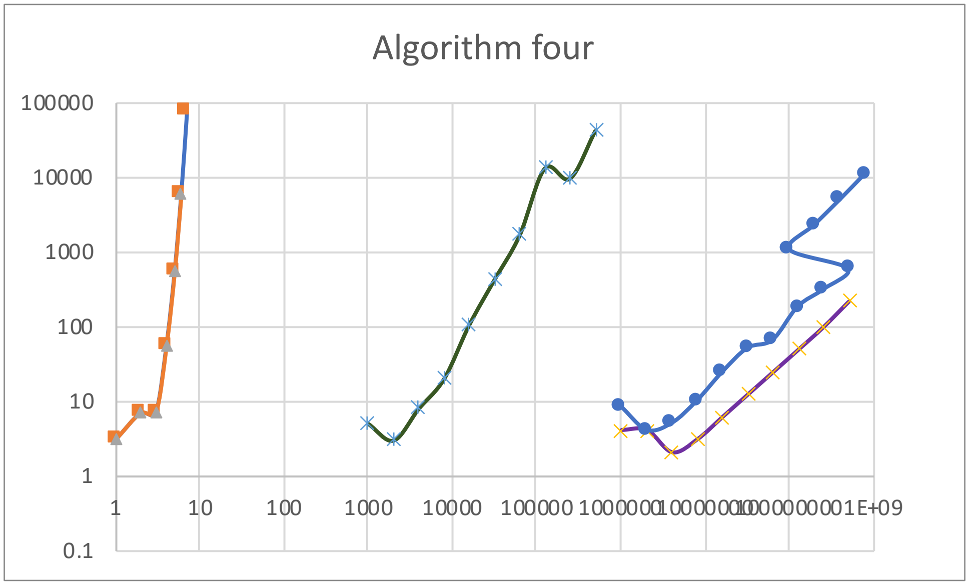
N	Time milliseconds	Ratio
1000000	**8**	
2000000	**4**	.5
4000000	**5**	1.25
8000000	**10**	2
16000000	**24**	2.4
32000000	**50**	2.08
64000000	**64**	1.28
128000000	**178**	2.78
256000000	**300**	1.68
512000000	**574**	1.91
102400000	**1033**	1.79
204800000	**2233**	2.16
409600000	**4799**	2.14



O(n)

Since the limit of the ratios approaches 2, this algorithm is linear. If that were true then the result of 819200000 should be close 4799*2. It was 10616 which is 4799*2.21.

With all 4 algorithms on one graph you can easily tell the difference in efficiency between them.
Algorithm one is in orange.
Algorithm two is in purple.
Algorithm three is in green.
Algorithm four in in black.



One can clearly see from the graph that for “large” input sizes algorithms two and four are the most efficient, then algorithm three, and the least efficient is algorithm one.