CS201-2

NAME: HAMMAD KHAN MUSAKHEL
ID: 21801175
HW2

Computer Specifications:

MacBook Pro (Retina, 15-inch, Mid 2015)
Processor: 2.2 GHz Quad-Core Intel Core i7
Memory: 16 GB 1600 MHz DDR3
Startup Disc: Macintosh HD
Graphics: Intel Iris Pro 1536 MB

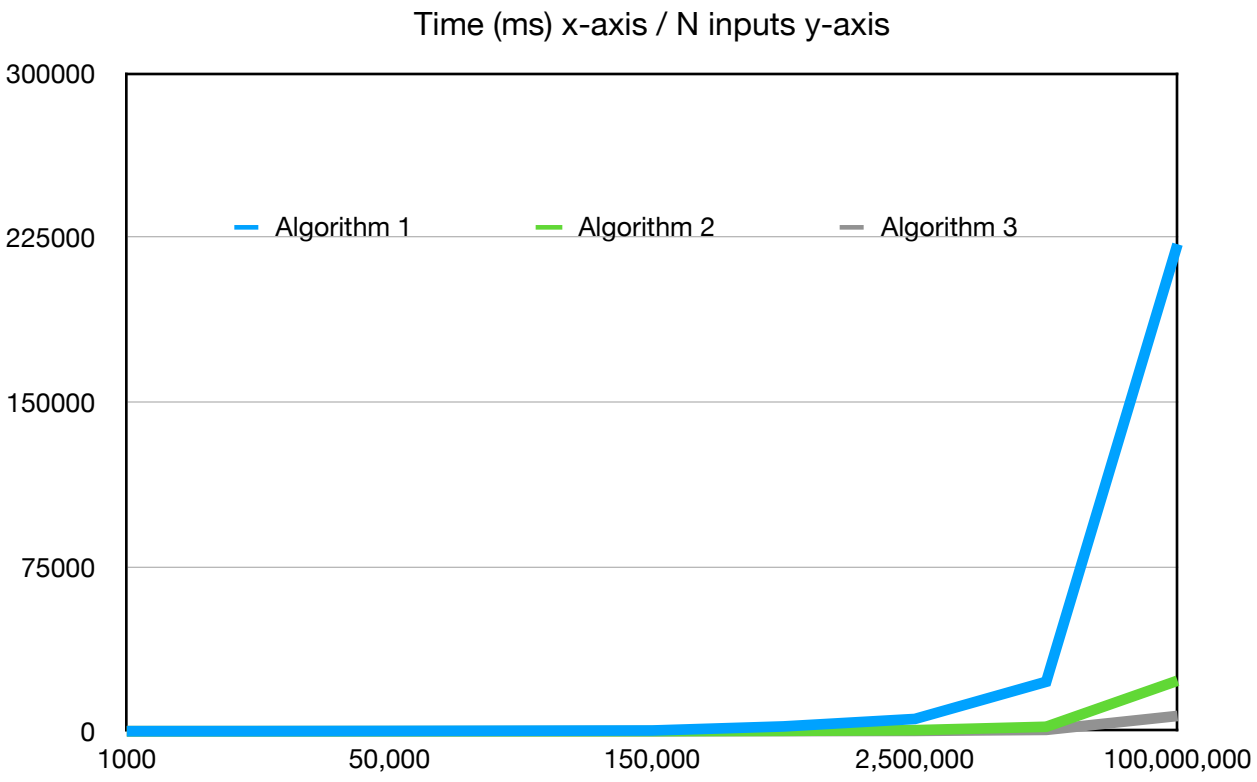
k = 1000. And Input Size (N) is varied.

Experimental Values of the 3 Algorithms

K is kept constant at 1000, and N is varied

Below is the graph for Time taken in Milliseconds (ms) for each Algorithm over the range of N input values:

N, the size of input	Algorithm 1, O(k.N)	Algorithm 2, O(NlogN)	Algorithm 3, O(N)
1000	1.246	0.095	0.087
10,000	22.11	1.20	0.577
50,000	103.48	6.975	2.88
100,000	215.54	14.9	6.10
150,000	320.51	23.30	7.81
1,000,000	2,181.54	180.91	64.38
2,500,000	5,582.41	468.295	142.4
10,000,000	22,528.20	2,030.55	606.5
100,000,000	221,950.30	23,025.5	6,954.69

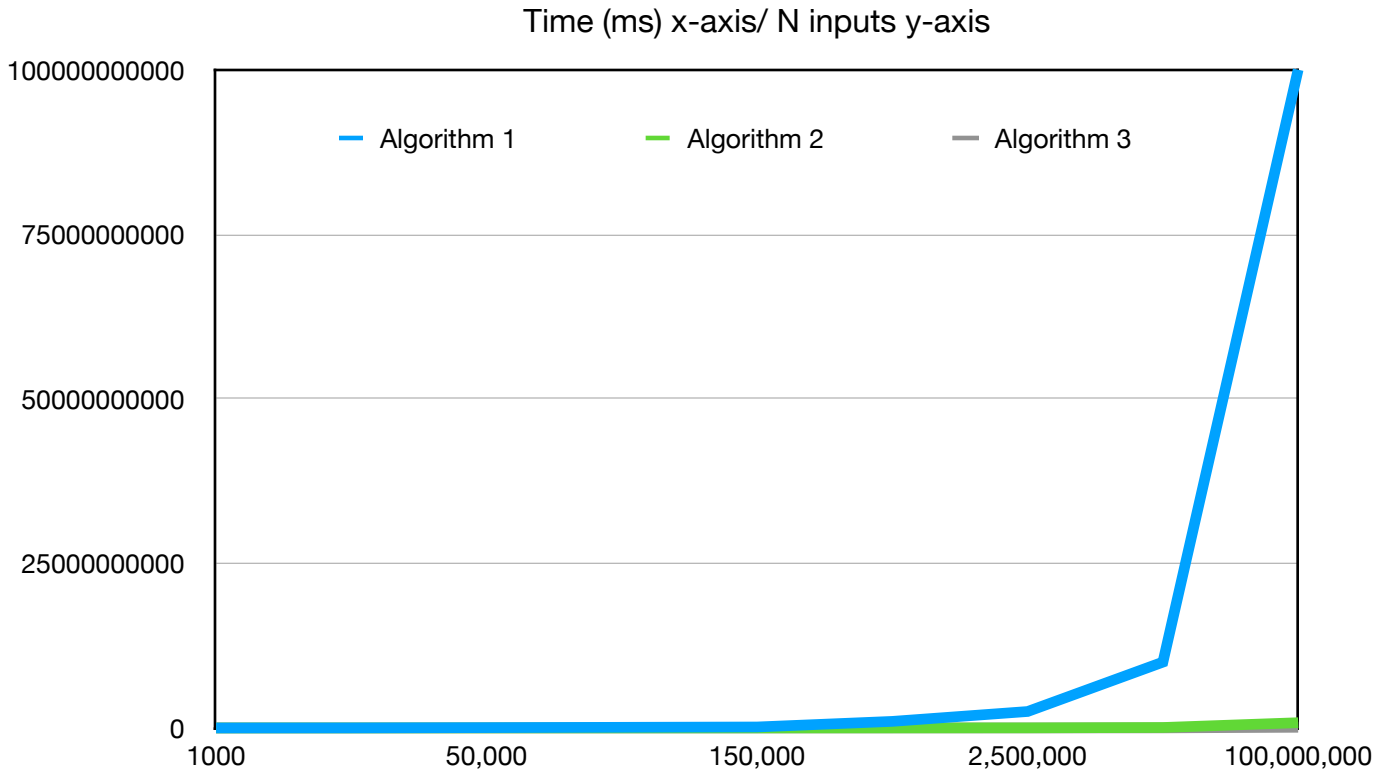


Theoretical Values of the 3 Algorithms

K is kept at 1000 and N is varied

Below is the graph for Time taken in Milliseconds (ms) for each Algorithm over the range of N input values:

N, the size of input	Algorithm 1, O(k.N)	Algorithm 2, O(NlogN)	Algorithm 3, O(N)
1000	1 x 10^6	3,000	1000
10,000	1 x 10^7	40,000	10,000
50,000	5 x 10^7	234,948	50,000
100,000	1 x 10^8	500,000	100,000
150,000	1.5 x 10^8	776,413.69	150,000
1,000,000	1 x 10^9	7,000,000	1,000,000
2,500,000	2.5 x 10^9	15,994,850.02	2,500,000
10,000,000	1 x 10^10	70,000,000	10,000,000
100,000,000	1 x 10^11	800,000,000	100,000,000



Analysis of the two sets of information:

Normally when we evaluate algorithms, we look at their order. This way, we can predict how the running time of the algorithm will increase when the input size (n) increases. We set out to find k largest integers in an array of varying size with three different algorithms (solutions) which had different time complexities. As the theoretical and experimental sets of information can testify to the fact that amongst the three provided algorithms, algorithm 3 is the fastest with an upper bound of O(N). The other two algorithms also have upper bounds respectively; algorithm 1 has an upper bound of O(k.N) and algorithm 2 has an upper bound of O(NlogN).

Ultimately, my values for the experimental readings and plotting are very similar to the theoretical ones; both sets of information follow the general expected trend. This testifies to the objective of the project: time complexities vary based on algorithms and their respective input size. It is not certainly expected to have the exact same values for algorithm 3 or 2 or 1 as received in the theoretical table/graph. Rather, it is a blue print for the general trend expected to be followed by algorithms having same upper bounds. It is obvious from the upper bounds that if I had chosen K to be 1 in algorithm 1 (O(k.N)), I would have observed the same time complexity as algorithm 3 (O(N)). The observed graph from the experiment has the similar trend for each of the algorithms in the theoretical graph. The running time for all the algorithms increases as n, the input size, increases, however the running time for O(N) doesn't increases as fast as that of O(NlogN) or O(k.N) increases, hence making linear growth fastest.

Inaccuracy is expected in practice with the same program having same values ran over and over again gave different yet similar time elapses for each searching algorithm. It can also be understood that the practical and theoretical plots do not follow an obvious pattern. This is

because the time taken to execute completely by an algorithm is also affected by other factors such as the programming language being used to implement the code and the machine being used. However, it allows us to observe how an algorithm increases as a function.