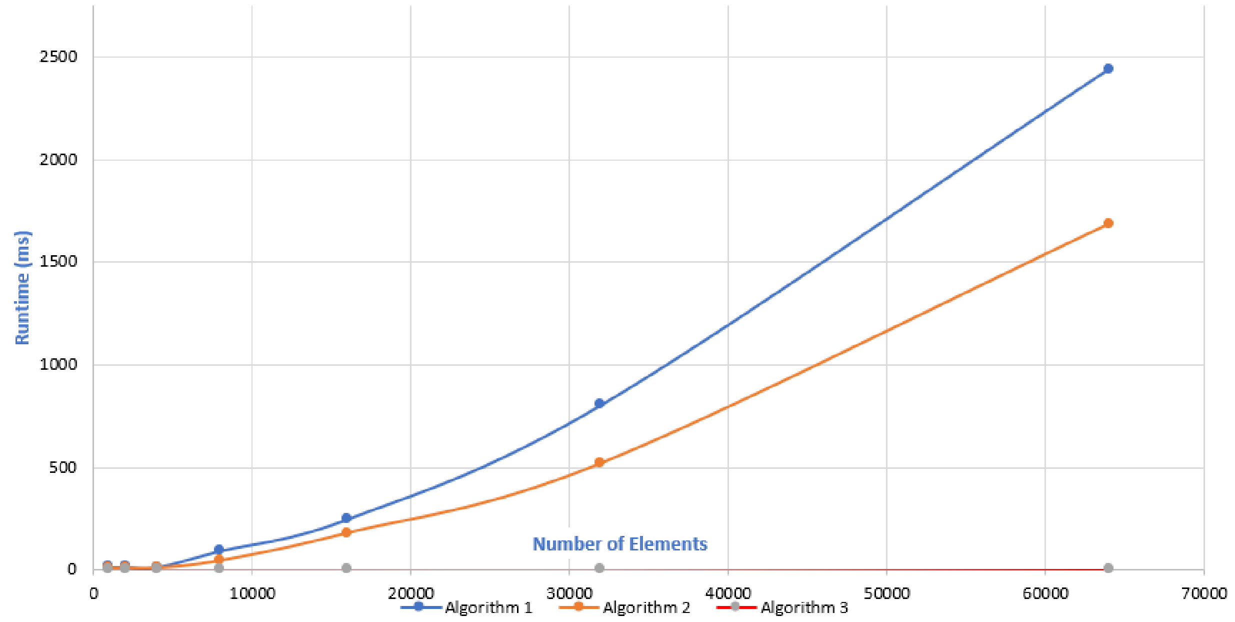
James Sherwood

3/6/2022

HW#8 Extra Credit

Professor Teredesai

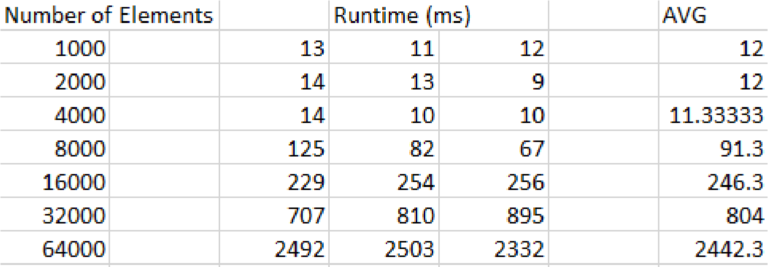
Three algorithms were tested side by side with the same data set, and runtimes were recorded. Prior to calling the methods containing the algorithms, a data set would be created for testing containing 1,000 elements, and this data set would be doubled at the end of each iteration of testing until it contained 64,000 elements. Run times were measured using java's nanoTime() method, taken immediately before an algorithm was called and immediately after. The run time was then recorded as being the difference of these two times converted into milliseconds.



Each algorithm was tested three times, and averaged, as follows:

Algorithm #1

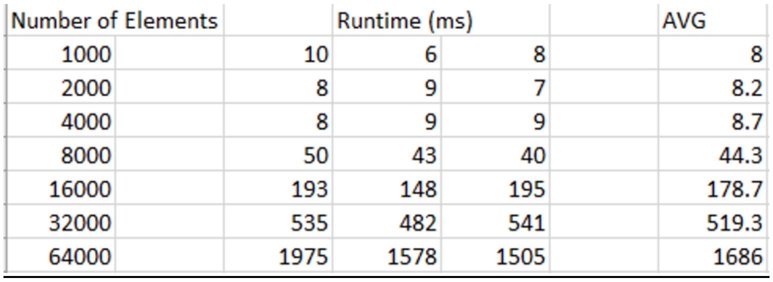
This algorithm was a simple double for loop, comparing each element of the array against every other element in the array. This algorithm did not discriminate with regard to already compared pairs and would even compare to itself. We see that in all tests it was the slowest and least efficient of the algorithms.



Number of Elements		Runtime (ms)		AVG
1000	13	11	12	12
2000	14	13	9	12
4000	14	10	10	11.33333
8000	125	82	67	91.3
16000	229	254	256	246.3
32000	707	810	895	804
64000	2492	2503	2332	2442.3

Algorithm #2

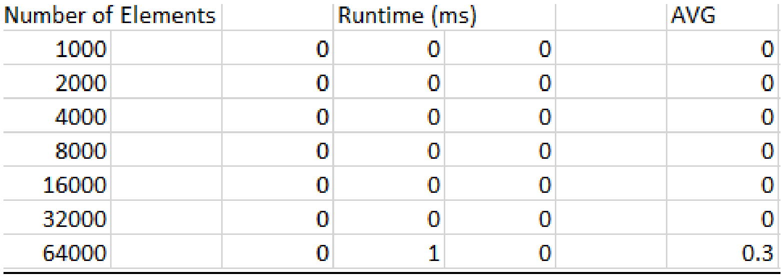
This algorithm improved upon the previous one by maintaining the double for loop, but avoiding duplicate comparisons by changing the parameters of the second loop to be one iteration ahead of the previous loop. We see that as values of N become larger, the efficiency of this algorithm is increased attaining nearly twice the speeds of algorithm #1 at our maximum value of 64,000 elements. We would expect this trend to continue at even larger values of N.



Number of Elements		Runtime (ms)		AVG
1000	10	6	8	8
2000	8	9	7	8.2
4000	8	9	9	8.7
8000	50	43	40	44.3
16000	193	148	195	178.7
32000	535	482	541	519.3
64000	1975	1578	1505	1686

Algorithm #3

Algorithm #3 is easily our most efficient algorithm, executing at speeds that aren't conveniently measured in milliseconds at the tested values of N. By eliminating the second for loop, and the mathematical equation at each step, and only looking for the largest and smallest values of the data set we've removed a lot of inefficiency and unnecessary computing while maintaining accuracy in the reported results.



Number of Elements		Runtime (ms)		AVG
1000	0	0	0	0
2000	0	0	0	0
4000	0	0	0	0
8000	0	0	0	0
16000	0	0	0	0
32000	0	0	0	0
64000	0	1	0	0.3