EXPERİMENTAL SETUP

AlgorithmSortAll

N	K	RANGE	AVERAGER TİME (SEC)
100,000	50,000	5,000	4.1259
100,000	50,000	10,000	4.14062
100,000	50,000	20,000	4.12618
100,000	50,000	40,000	4.14554
100,000	50,000	80,000	4.146
200,000	100,000	5,000	16.4555
200,000	100,000	10,000	16.545
200,000	100,000	20,000	16.745
200,000	100,000	40,000	17.045
200,000	100,000	80,000	17.467
300,000	150,000	5,000	37.336
300,000	150,000	10,000	37.566
300,000	150,000	20,000	38.233
300,000	150,000	40,000	39.543
300,000	150,000	80,000	39.121
400,000	200,000	5,000	66.1157
400,000	200,000	10,000	66.232
400,000	200,000	20,000	66.232
400,000	200,000	40,000	66.232
400,000	200,000	80,000	67.321
500,000	250,000	5,000	104.227
500,000	250,000	10,000	103.232
500,000	250,000	20,000	104.232
500,000	250,000	40,000	105.213
500,000	250,000	80,000	103.926

AlgorithmSortK

N	K	RANGE	AVERAGE TİME (SEC)
100,000	50,000	5,000	8.60342
100,000	50,000	10,000	8.58070
100,000	50,000	20,000	8.997400
100,000	50,000	40,000	8.61941
100,000	50,000	80,000	8.59974
200,000	100,000	5,000	35.28
200,000	100,000	10,000	35.40
200,000	100,000	20,000	35.6706
200,000	100,000	40,000	36.543
200,000	100,000	80,000	37.1232
300,000	100,000	5,000	80.0039
300,000	100,000	10,000	83.345
300,000	100,000	20,000	83.823432
300,000	100,000	40,000	80.324234
300,000	100,000	80,000	82.3242142
400,000	200,000	5,000	143.675
400,000	200,000	10,000	143.2312
400,000	200,000	20,000	145.232
400,000	200,000	40,000	143.23424
400,000	200,000	80,000	144.232
500,000	250,000	5,000	225.67
500,000	250,000	10,000	225.2123
500,000	250,000	20,000	225.32421
500,000	250,000	40,000	225.213213
500,000	250,000	80,000	225.45

AlgorithmSortHeap

N	K	RANGE	AVERAGE TİME (SEC)
100,000	50,000	5,000	0.0253740
100,000	50,000	10,000	0.0253750
100,000	50,000	20,000	0.0263510
100,000	50,000	40,000	0.026642
100,000	50,000	80,000	0.0273040
200,000	100,000	5,000	0.055087
200,000	100,000	10,000	0.055097
200,000	100,000	20,000	0.055127
200,000	100,000	40,000	0.055235
200,000	100,000	80,000	0.055387
300,000	150,000	5,000	0.081965
300,000	150,000	10,000	0.084322
300,000	150,000	20,000	0.085324
300,000	150,000	40,000	0.086342
300,000	150,000	80,000	0.088324
400,000	200,000	5,000	0.126697
400,000	200,000	10,000	0.123423
400,000	200,000	20,000	0.127787
400,000	200,000	40,000	0.124343
400,000	200,000	80,000	0.129232
500,000	250,000	5,000	0.151866
500,000	250,000	10,000	0.152324
500,000	250,000	20,000	0.152342
500,000	250,000	40,000	0.153240
500,000	250,000	80,000	0.155686

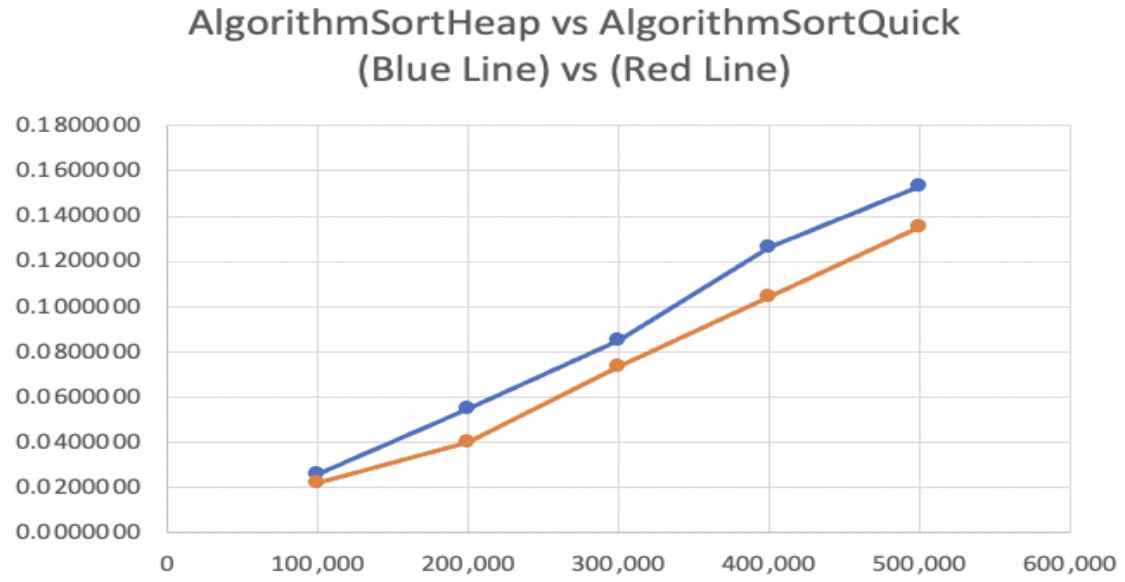
ALgorithmSortQuick

N	K	RANGE	AVERAGE TİME (SEC)
100,000	50,000	5,000	0.02230
100,000	50,000	10,000	0.02131
100,000	50,000	20,000	0.02321
100,000	50,000	40,000	0.021232321
100,000	50,000	80,000	0.023213
200,000	100,000	5,000	0.04323
200,000	100,000	10,000	0.04232
200,000	100,000	20,000	0.03213
200,000	100,000	40,000	0.04232
200,000	100,000	80,000	0.04123
300,000	150,000	5,000	0.07232
300,000	150,000	10,000	0.07432
300,000	150,000	20,000	0.07343
300,000	150,000	40,000	0.07312
300,000	150,000	80,000	0.07321
400,000	200,000	5,000	0.10321
400,000	200,000	10,000	0.10213
400,000	200,000	20,000	0.10232
400,000	200,000	40,000	0.10234
400,000	200,000	80,000	0.11343
500,000	250,000	5,000	0.13232
500,000	250,000	10,000	0.13131
500,000	250,000	20,000	0.13769
500,000	250,000	40,000	0.13769
500,000	250,000	80,000	0.13658

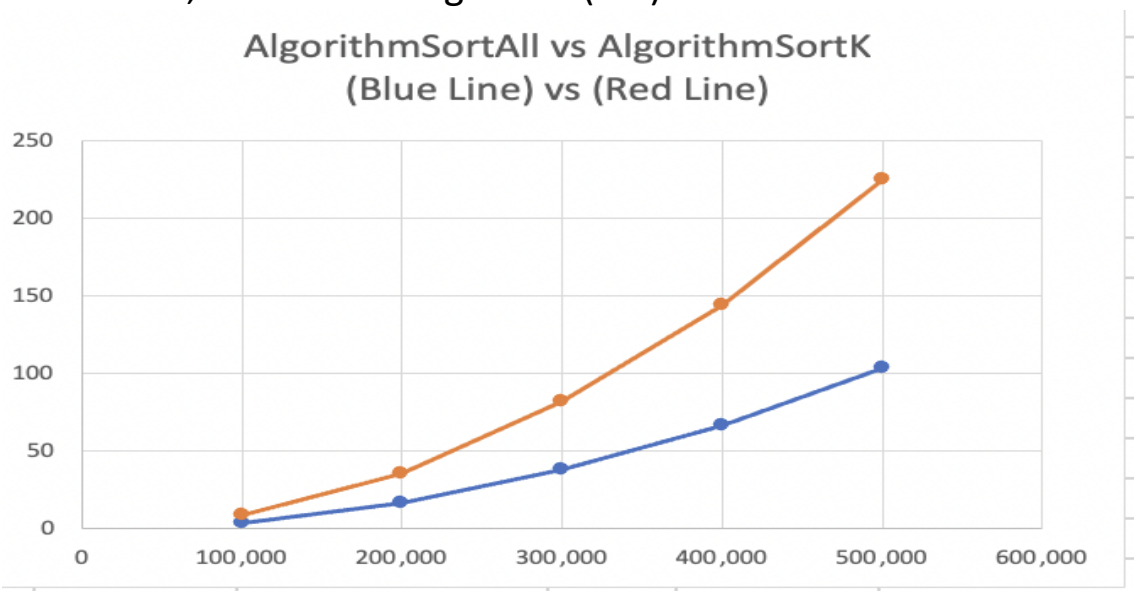
RESULTS

	AlgorithSortAll	AlgorithmSortk	Algorithmsortheap	AlgorithmSortQuick
N	Average Time	Average Time	Average Time	Average Time
100,000	4.136848	8.680134	0.0262092	0.02225
200,000	16.8515	36.00336	0.0551866	0.04025
300,000	38.360	81.96415604	0.085255548	0.07328
400,000	66.426606	143.920888	0.126296504	0.10469
500,000	104.166	225.3737446	0.153091643	0.13512

X axis = N , Y axis = Average Time(sec)



X axis = N , Y axis = Average Time(sec)



DİSCUSSİON

First, I set my N values from 100000 to 500000 and my K values as my n values divided by 2, respectively. I determined my range by doubling from 5000 to 80000. In the first part, I entered these values separately for each of my algorithms and found the average elapsed time in seconds. In the second part, I created a separate table by calculating the average of the working speed of the algorithm at different Range for an N value by considering these calculated values again for each algorithm separately. According to the values of this table I created, I created a comparative graph of AlgorithmSortAll and AlgorithmSortK, and AlgorithmSortHeap and AlgorithmSortQuick together.

Secondly, the average algorithm execution speed of AlgorithmSortAll and AlgorithmSortK is O(n^2). If we go to the comparison of AlgorithmSortAll and AlgorithmSortK, the results of my code are not the expected results, as it can be seen in the table. Because AlgorithmSortAll and AlgorithmSortK use the same sorting algorithm, and AlgorithmSortK ignores unnecessary inputs to reduce processing overhead. But when we look at my results, we see that AlgorithmSortAll runs faster, which may be due to the fact that they both have O(n^2) working speed, and my AlgorithmSortK wastes more time ignoring unnecessary inputs. Also, when we look at the graphics, we see that AlgorithmSortAll is faster than AlgorithmSortK. And we see that the difference in working speed increases as the size of the inputs increases. In summary, my conclusion based on the data is that the reason AlgorithmSortAll is faster than AlgorithmSortK is that AlgorithmSortK wastes more time while sifting through the inputs.

Lastly, according to my input, AlgorithmSortHeap and AlgorithmSortQuick have very, very small operational speed differences. AlgorithmSortHeap works with O(nlogn) algorithm speed, while AlgorithmSortQuick works with O(n) algorithm speed. As we can see when we look at the graphs, AlgorithmSortQuick is linear while AlgorithmSortHeap is nlogn. The expected result was that

AlgorithmSortHeap runs slower than AlgorithmSortQuick, so my data match with the expected result. In addition, if the size of my inputs here were much larger, for example, like 10 million, the difference between the operating speeds of AlgorithmSortHeap and AlgorithmSortQuick, the very, very small differences in the operating speeds of small-sized inputs would no longer be valid, and I think the differences between the operating speeds would reach immeasurable proportions.