Hande Köksal

22401938

Section 1

CS201 HW-2 REPORT

3. Task

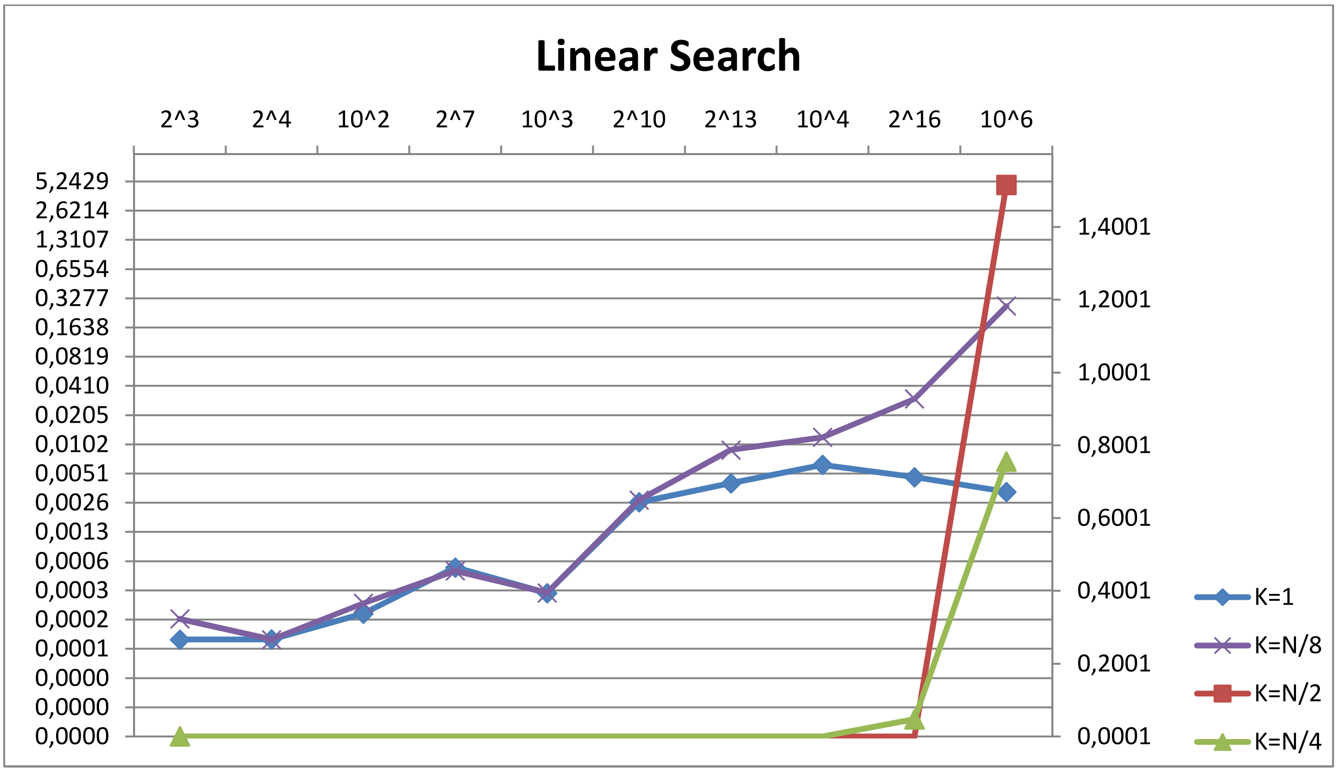
3.4 Table

N	Linear Search				Binary Search				Jump Search			
K	K=1	K=N/2	K=N/4	K=N/8	K=1	K=N/2	K=N/4	K=N/8	K=1	K=N/2	K=N/4	K=N/8
2^3	9.99e-005	9.2e-005	9.99e-005	0.0001632	0.0001027	0.0001006	0.0001308	4.7e-005	0.000149	0.0002014	0.0002045	0.000348
2^4	0.0001	0.0001001	0.0001033	9.98e-005	0.001992	0.0001059	0.0001	0.0002002	0.0001926	0.0001106	0.0002297	0.0002038
10^2	0.000185	0.0001	0.0002728	0.0002367	0.0001668	0.0002441	0.0002416	0.0002541	9.96e-005	0.0003371	0.0002	0.0001999
2^7	0.0005515	0.0006045	0.0005851	0.0005091	8.07e-005	0.0006514	0.0002708	0.0002769	0.0002026	0.0008498	0.0003319	0.0004839
10^3	0.0003006	0.0005104	0.0002534	0.0003027	0.0002274	0.0029248	0.0017369	0.0010759	0.0002235	0.0040619	0.0034909	0.0011528
2^{10}	0.0025859	0.0025719	0.0027021	0.0027322	9.46e-005	0.002191	0.0013183	0.0009483	0.0003129	0.0020906	0.001406	0.0009915
2^{13}	0.0040743	0.0197228	0.0139358	0.0089714	0.0001997	0.0210553	0.0119183	0.0065172	0.0003005	0.0192003	0.0117435	0.0062701
10^4	0.0063083	0.0277047	0.0180098	0.0121205	0.0001811	0.0251239	0.0143969	0.0079521	0.0001998	0.0245471	0.0149932	0.008083
2^{16}	0.0046962	0.0824254	0.0474311	0.0301442	0.0001999	0.0994675	0.059492	0.037316	0.0005056	0.109841	0.0827339	0.0483414
10^6	0.003309	1.5159	0.755926	0.272594	0.0001621	2.08013	1.09078	0.34664	0.0012439	2.06545	1.05641	0.431087

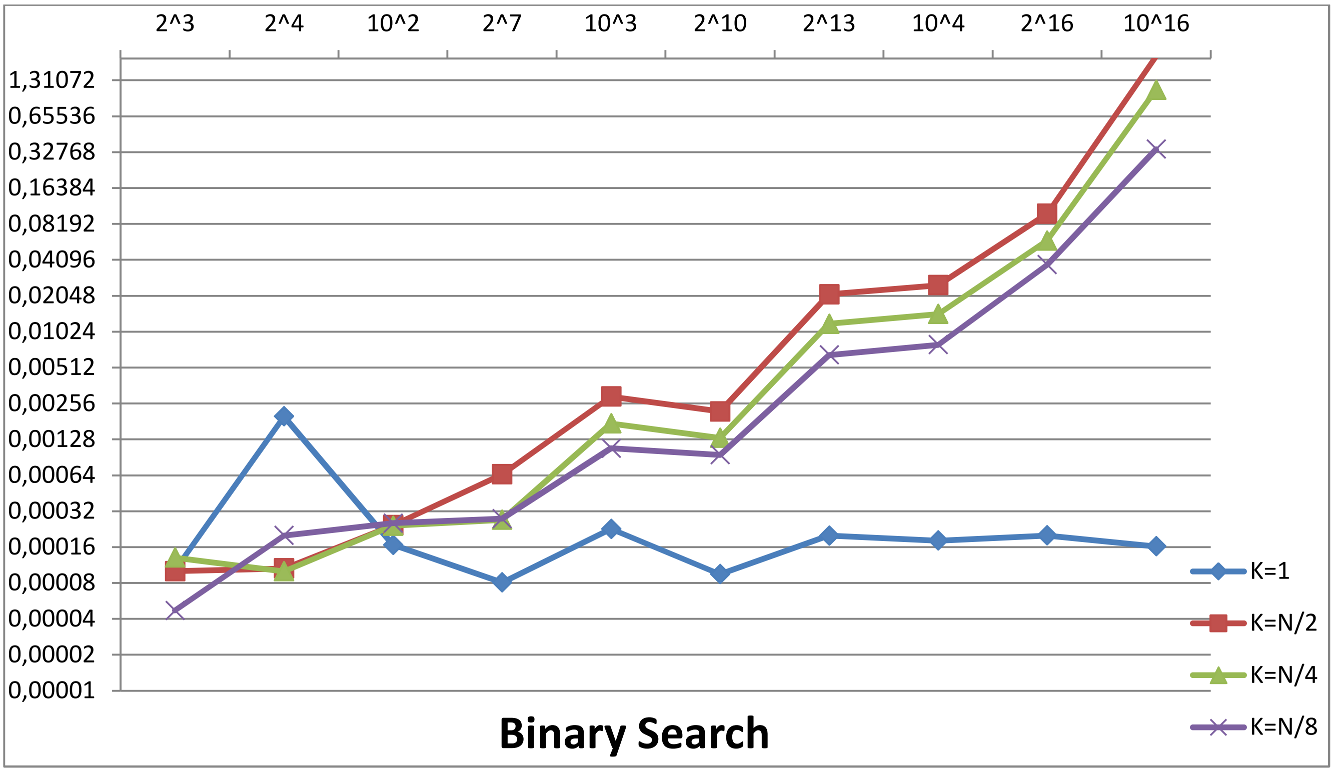
Table 1

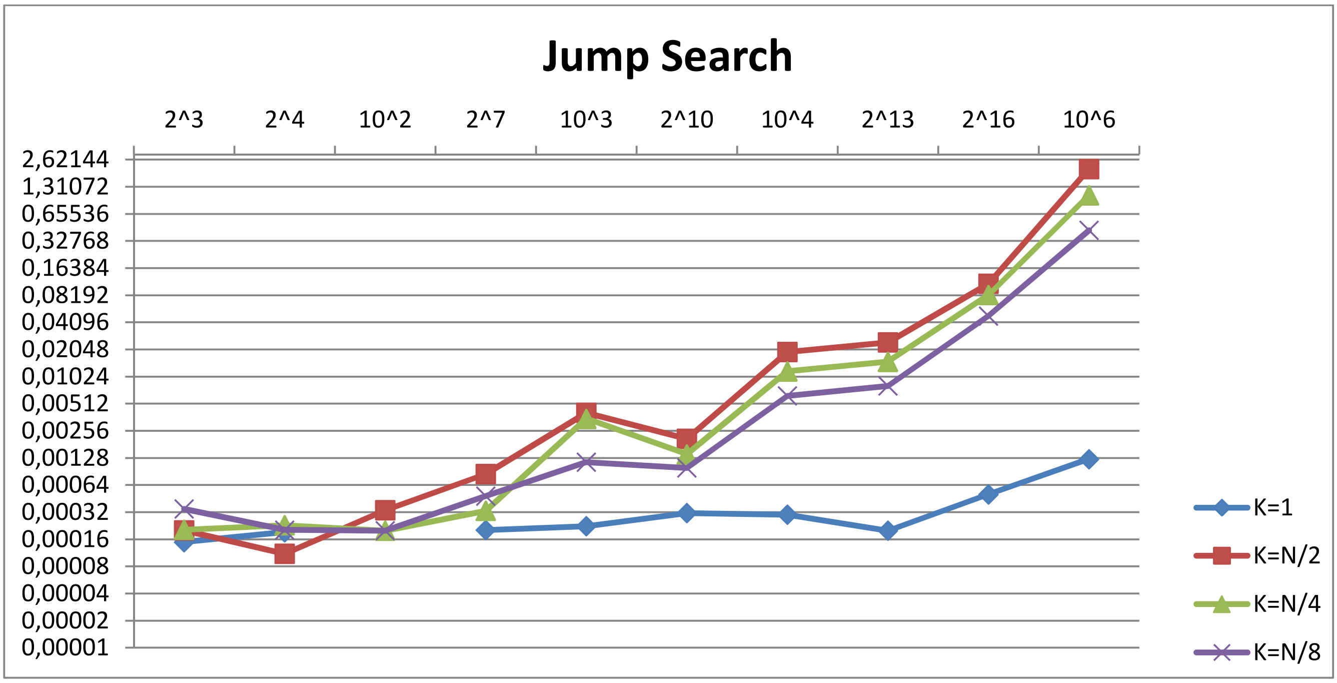
Table 1 Remarks: Every time value is in milliseconds. All search algorithms with different N and K values were put in a for loop to prevent getting 0 milliseconds. The for loop executed for 10000 times. After that, the total time was divided by 10000 to get the right values.

3.5 Plots



Linear Search Graph: K=1 and K=N/8 are on the left y-axis. The other K values are on the right y-axis. This was done in order to minimize the overlap between the lines.





Linear, Binary and Jump Search Graph: x-axis values are on the top of the graph because y-axis values are in logarithmic scale. The numbers are extremely small and are very close to each other, doing it the other way results in the lines overlapping. It makes it difficult to see the changes as well because they all get packed into the bottom of the graph.

4. Comments

4.1 What are the specifications of your computer? (Processor, RAM, operating system, etc.)

Device Name: DESKTOP-CILIM76

Processor: Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz, 2.11 GHz

Installed RAM: 8.00 GB (usable: 7.81 GB)

Operating System: Windows 10 Pro

Storage: 932 GB HDD TOSHIBA MQ04ABF100, 119 GB SSD LITEON CV8-8E128-HP

Graphics Card: NVIDIA GeForce MX110 (2 GB), Intel(R) UHD Graphics (128 MB)

System Type: 64-bit operating system, x64-based processor

4.2 Which of the search algorithms described in Section 2 work, and which do not, when the data is unsorted?

Linear search works, binary and jump search don’t work when the data is unsorted. That is because in linear search, the algorithm looks at each element one by one. The order doesn’t matter. However binary and jump search don’t look at every element in the data. Instead, these algorithms look into specific places and continue accordingly to the element that is currently being looked into. After that, these algorithms plan the next move by looking at the relation between the current element and the target. This relation being whether they are the same or if one of them is smaller or bigger than the other.

4.3 What are the theoretical worst, average, and best cases for each search algorithm and how do they change when they are adapted to finding the nearest K values as described in Section 3.2? What are the corresponding running times for each scenario for each algorithm? (K is defined as a function of N)

1. Linear Search

Search Algorithm

Best case: O(1): Target is at index 0

Average case: O(N): Target is in the middle, at index (N-1)/2

Worst case: O(N): Target is at index N-1 or target is not in the data

Search Algorithm Adapted to K

After finding target, it takes O(K) to get K closest values.

Best case: O(K)

Average case: O(N+K)

Worst case: O(N+K)

2. Binary Search

Search Algorithm

Best case: O(1): Target is the middle, at index (N-1)/2

Average case: O(logN)

Worst case: O(logN)

Search Algorithm Adapted to K

After finding target, it takes O(K) to get K closest values.

Best case: O(K)

Average case: O(logN+K)

Worst case: O(logN+K)

3. Jump Search

Search Algorithm

Best case: O(1): Target is at index 0

Average case: O($\sqrt{N}$): Target is in the middle, at index (N-1)/2

Worst case: O($\sqrt{N}$): Target is at an index in last part [N-1-jump, N-1]

Search Algorithm Adapted to K

After finding target, it takes O(K) to get K closest values.

Best case: O(K)

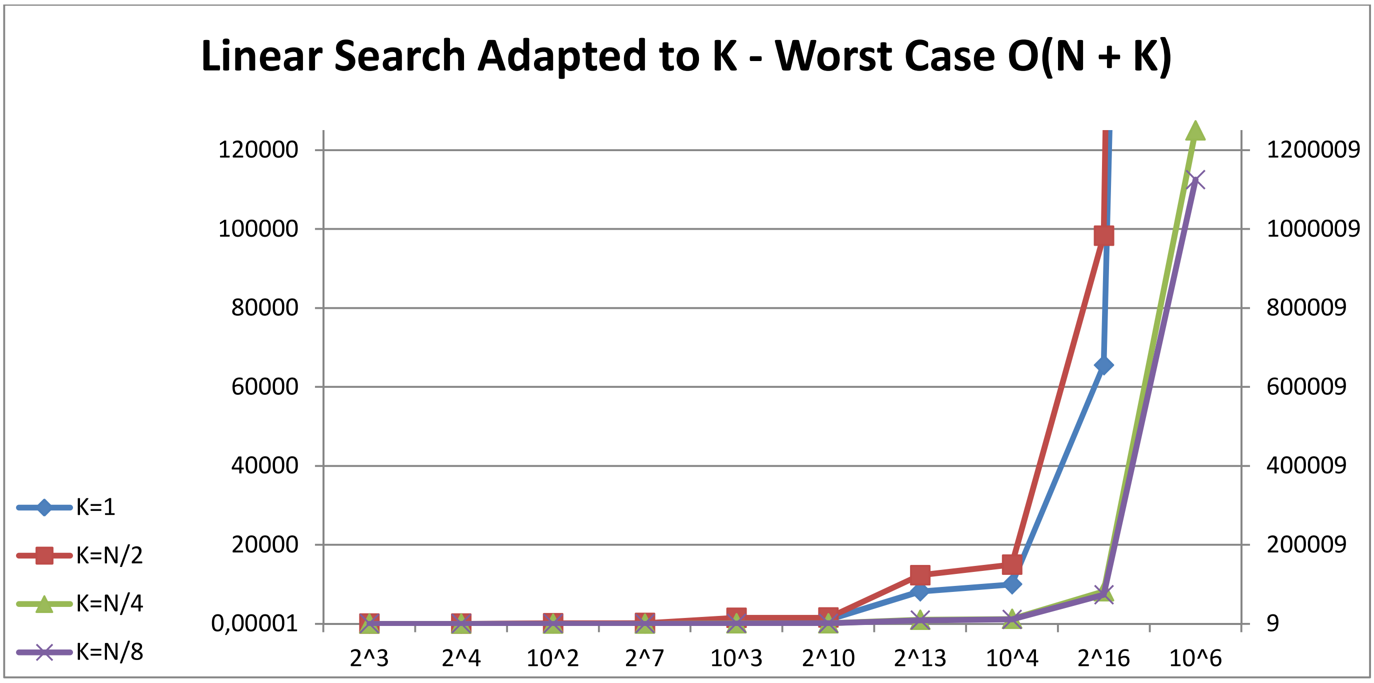
Average case: O($\sqrt{N}$+K)

Worst case: O($\sqrt{N}$+K)

4.4 What are the worst, average, and best cases for each algorithm based on your table and your plots? Do the theoretical and observed results agree? If not, give your best guess as to why that might have happened.

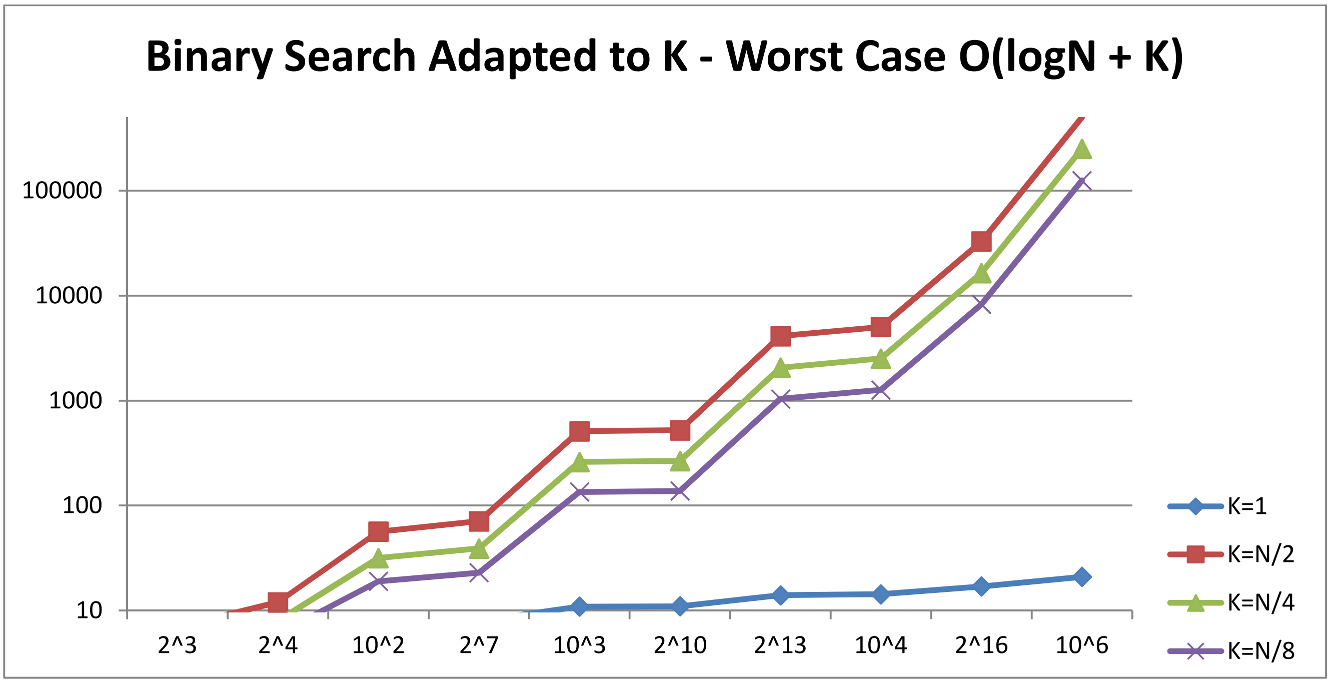
For the tables and graphs, the best case is when N=2^3 and K=1. The worst case is when N=10^{16} and K=N/2. The average case is when K=N/4 and K=N/8 for binary and jump search. The average case for linear search is when N is taken as a middle value in the range, like N=10^3, 2^{10}, or 2^{13}. The best and worst cases of theoretical and observed results agree. However, for the average case, they don't agree. This could be because theoretical result looks at a fixed problem, it eliminates factors and different situations we might encounter when tested in a real world. The data and the targets are always chosen randomly and the big-Oh notation doesn't show the terms smaller than the most significant N. For example, in my code, there are a lot of printing to terminal which takes O(1) time. Operations like this change the elapsed time in the real world but we ignore them in the theoretical results. For these reasons, the elapsed time for each scenario may change. The best and worst cases eliminate this uncertainty because we generally look at the first, middle or the last indexes; they are fixed or can be found with minimum/maximum steps. On the other hand, things are always changing for the average case. Also, I have different cases (methods) in my code for when the target is smaller than the number at index 0, when the target is larger than the number at index N-1, and when the target falls in between two numbers in the data. The time it takes to find K closest values is different for these cases. This is another factor that changes the elapsed time. For those reasons, theoretical and observed results may not always agree.

4.5 Plot the theoretical running times of each algorithm for the different K values described in Section 3.3 and compare them to the plots generated in 3.5. Comment on consistencies and inconsistencies. These plots should have the same organization as plots in 3.5.

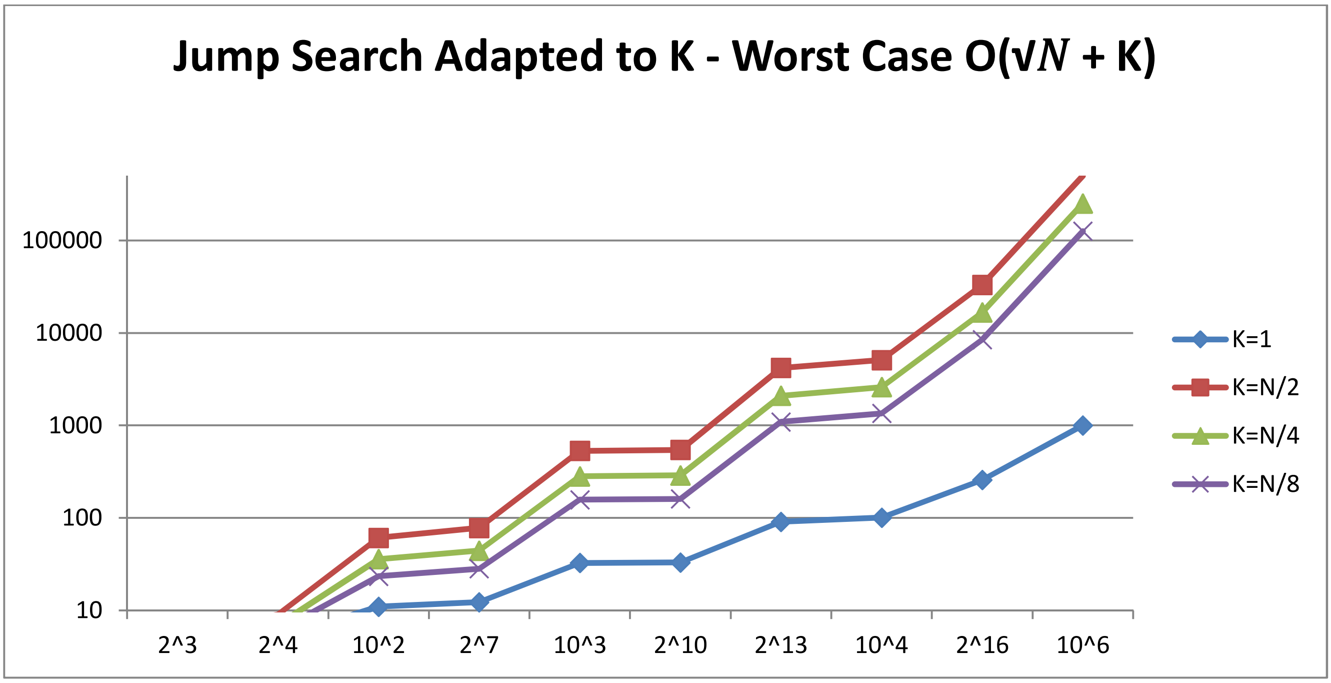


Linear Search Graph: K=1 and K=N/2 are on the left y-axis. The other K values are on the right y-axis. This was done in order to minimize the overlap between the lines.

Comments: My K=N/2 and K=N/4 lines are consistent with the theoretical running time. K=1 and K=N/8 lines are consistent with large N values but they might come off as inconsistent with small N values. That is because of the y-axis values for the graphs. In the theoretical case, these y values are not in milliseconds and they are not specifically small numbers. They show the value based on the O() function. Still, when we look at my graph, it can be observed that for both K=1 and K=N/8 values, the smaller N's generate small elapsed times whereas larger N's rapidly generate big elapsed times.



Comments: My graph is pretty consistent based on the overall looks. Because it's binary search, the values grow slowly when N gets larger and larger. However, the K=1 line is not very consistent. There may have been some factors that changed my K=1's corresponding values, again my graph is not ideal, it is bound to have some errors due to all the random allocations and the many factors that are not calculated in the theoretical cases. But still, I would say that the changes of the lines are consistent with the theoretical cases.



Comments: My graph, like binary search, is consistent with the theoretical graph. Unfortunately, my values were much smaller than these values. Because of this, I can't compare small values between N = 2^3 and N= 2^7. This is also the case for binary search. Overall, the changes in the lines are consistent and both binary and jump search grow much slower than linear search.