Deniz Yüksel

21600880 09.04.2018

CS 201 – Spring - Homework 2

The experiment is done in my personal computer which has the following properties:

- Intel® Core™ i7-6700HQ CPU @ 2.60GHz 2.60GHz
- RAM of 16.0 GB
- 64-bit Operating System, x64- based processor

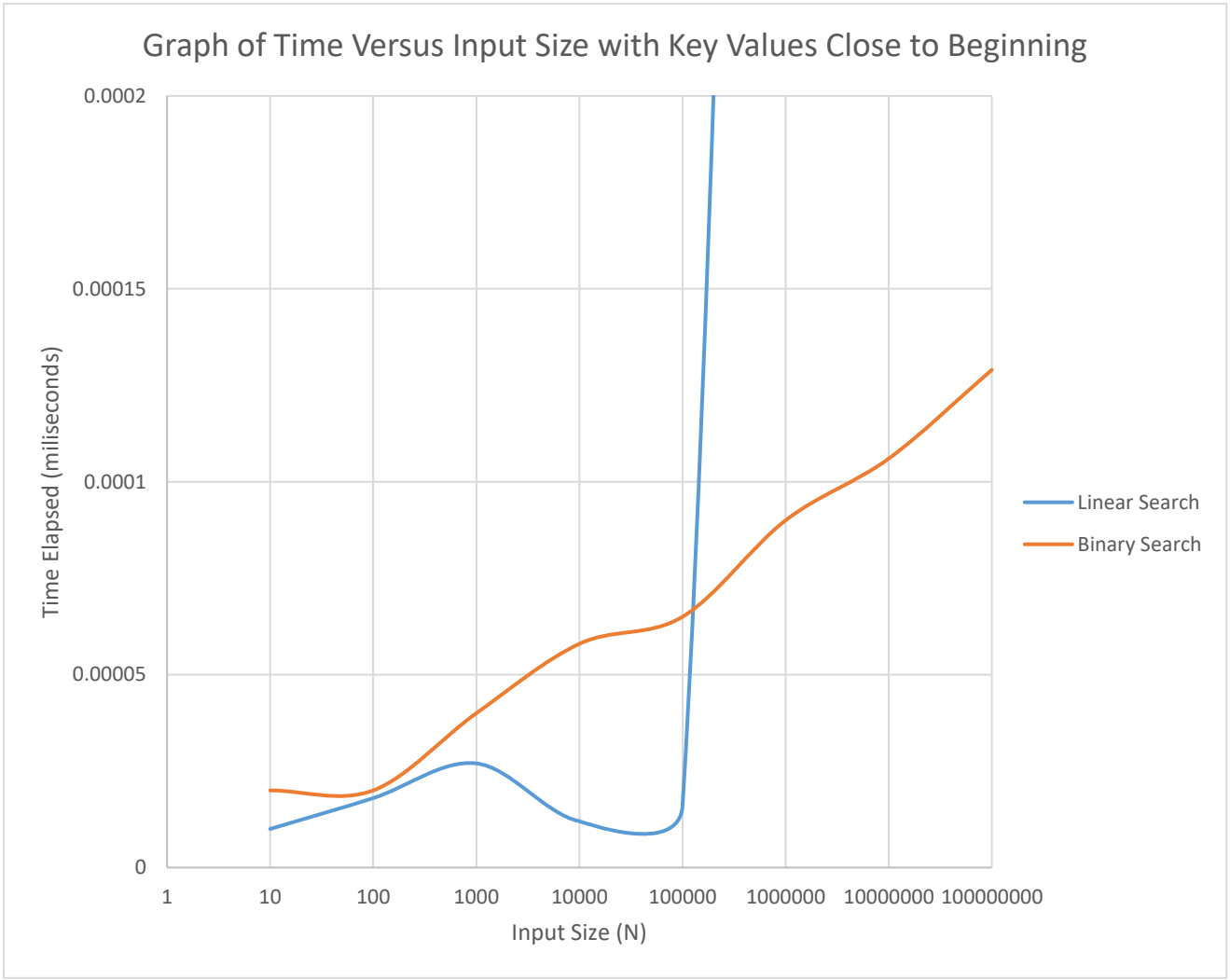
The data are gathered using Codeblocks IDE. All the graphs for this homework are drawn with Microsoft Excel 2016. In order to draw the graphs in Excel, the Turkish notation for double value types are written with "," rather than ".". This is because Excel cannot plot if double value types are written with "."s. Thus, in the graphs, "," is used.

The table below shows the runtime for each cases for the key: close to beginning, around middle, close to end, and not existing. The table shows these data with respect to input sizes, and for both algorithms, linear and binary search.

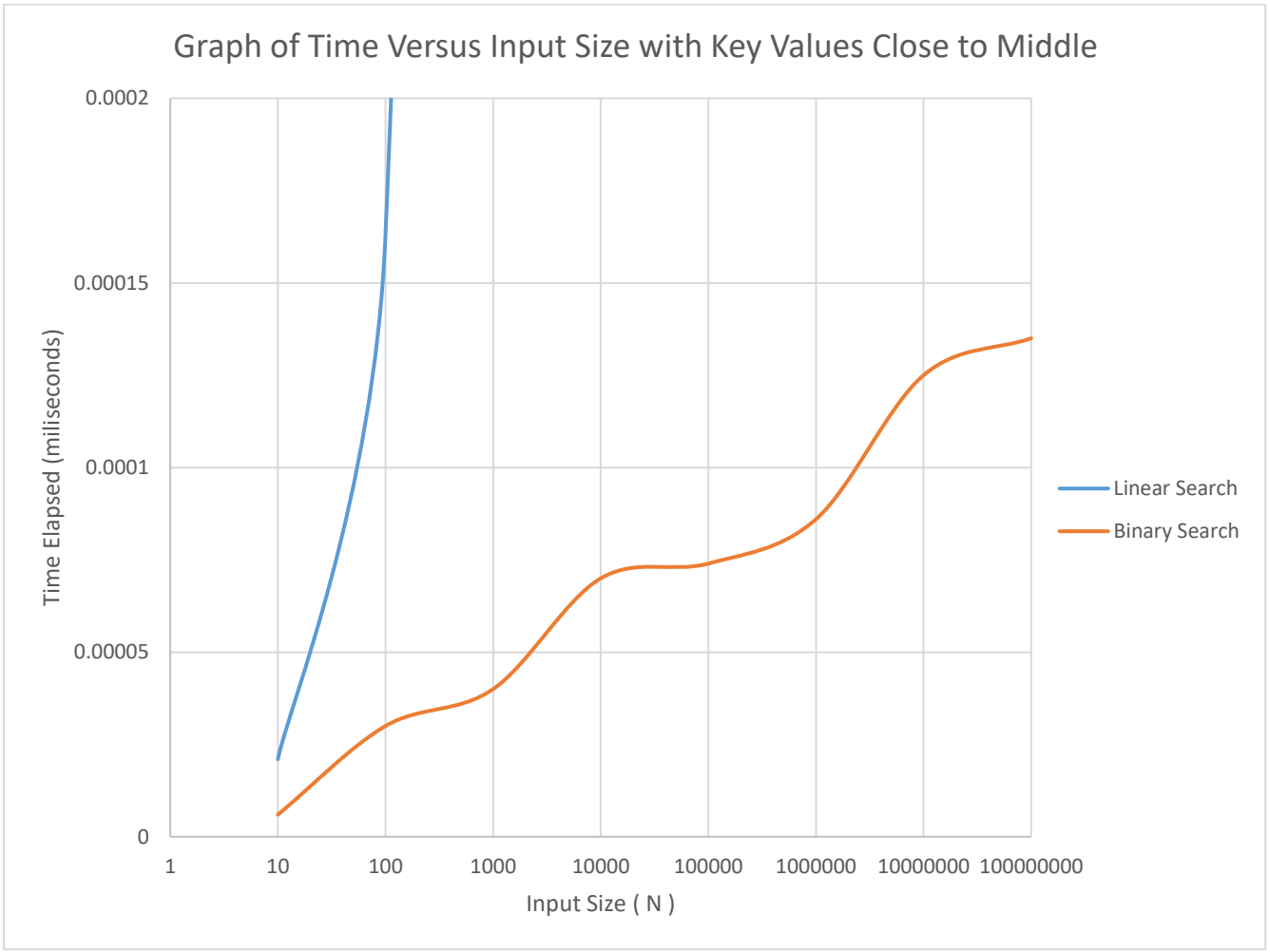
The two tables below indicates the runtimes of the two algorithms, based on input sizes and the position of the searched key.

	Input Size (N)	Runtime at Key Beginning	Runtime at Key Middle	Runtime at Key at End	Runtime at Key Not Existing
ALGORITHM 1: LINEAR SEARCH O(N)	10	0.000010	0.000021	0.000023	0.000035
	100	0.000018	0.000163	0.000275	0.000272
	1000	0.000027	0.001412	0.002891	0.002974
	10000	0.000012	0.013501	0.028482	0.028685
	100000	0.000016	0.132762	0.262445	0.262756
	1000000	0.000865	1.319604	2.383002	2.860000
	10000000	0.002002	13.42005	27.80000	27.87000
	100000000	0.010023	143.0000	261.4000	262.8000
ALGORITHM 2: BINARY SEARCH O(logN)	10	0.000020	0.000006	0.000013	0.000014
	100	0.000020	0.000030	0.000028	0.000027
	1000	0.000040	0.000040	0.000048	0.000038
	10000	0.000058	0.000070	0.000056	0.000054
	100000	0.000065	0.000074	0.000070	0.000067
	1000000	0.000090	0.000086	0.000084	0.000104
	10000000	0.000106	0.000125	0.000097	0.000109
	100000000	0.000129	0.000135	0.000109	0.000110

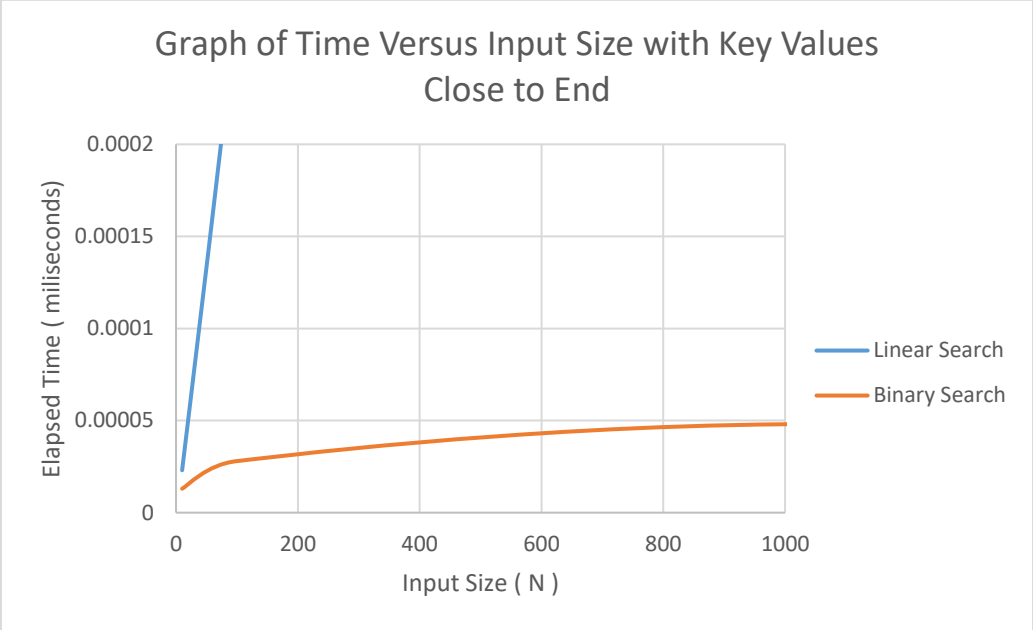
The first graph below shows the rate of two algorithms with respect to elapsed time versus input sizes. This graph, in particular is interested in key values that are very close to beginning. Because the key value is very close to the beginning, for small inputs linear search is faster than binary search. However, as the input size increases as it's seen from the graph around 100000, linear search rockets to infinity with respect to the timescale. This is because the term " close to beginning" can be interpreted differently. For instance, in my case, close to beginning for input size of 10 is 2, and for a size of 100000, it can be 100 instead of 2. If I had chosen key values such as 5 or 10 for every query, linear search would be always faster because the value would be at the very start each time. For small key values, it is user-dependent to say which algorithm is faster. But, after key values that are still close to the beginning, binary search gets faster than linear search.



The second graph below shows the rate of two algorithms with respect to elapsed time versus input sizes. This graph, in particular is according to the key values that are around middle. The timescale and input size scale are similar because linear search is still visible in the graph. For values that are in the middle, linear search is visible until an input size of 100. This is because binary search is much faster than linear search as the key value increases. Linear search has to scan all the array from the beginning until the key value, however the binary search is much faster because each time, it divides the possibility to two. For this case, binary search predicts the key value starting from middle, and acts accordingly if the middle value is smaller or larger than the key value. For input sizes after 100, in this timescale linear search is invisible, thus it is so slow to compete with binary search.



The third graph shows the rate of two algorithms with respect to elapsed time versus input sizes. This graph, focuses on the key values that are close to end. Predictibly, linear search would be much more slower than binary search in this case because as the array gets larger, the values at the end are much further away from the beginning. For this case, although the timescale is same, the input size scale has to be dropped. This is because if input size would be from 0 to 10^8, the curve of linear search would be impossible to distinguish. It is because binary search is much faster compared to linear search for larger input sizes when given very large key values.



The fourth graph below shows two curves for linear and binary search again, with respect to elapsed time and input size. This time, the graph is for key values that are not in the array. Thus, linear search algorithm has to scan through the whole array to know if the value is present or not. In my case, there is not much difference fort he third and fourth graph. This is because I picked key values that are very close to the end. Again like in the first graph, the phrases, “close to the beginning” or “close to the end” can have various interpretations. A conclusion that can be drawn from this homework is that as the key value increases, the runtime of binary and linear search go to a much larger difference, assuring that input size is also increasing. For small input sizes, linear search can sometimes even be faster. However, for the general case, as we take the limit of N and LogN, we can observe that N is much larger compared to LogN for large N values.