Mohammed S. Yaseen

Fall 2019

Bilkent University

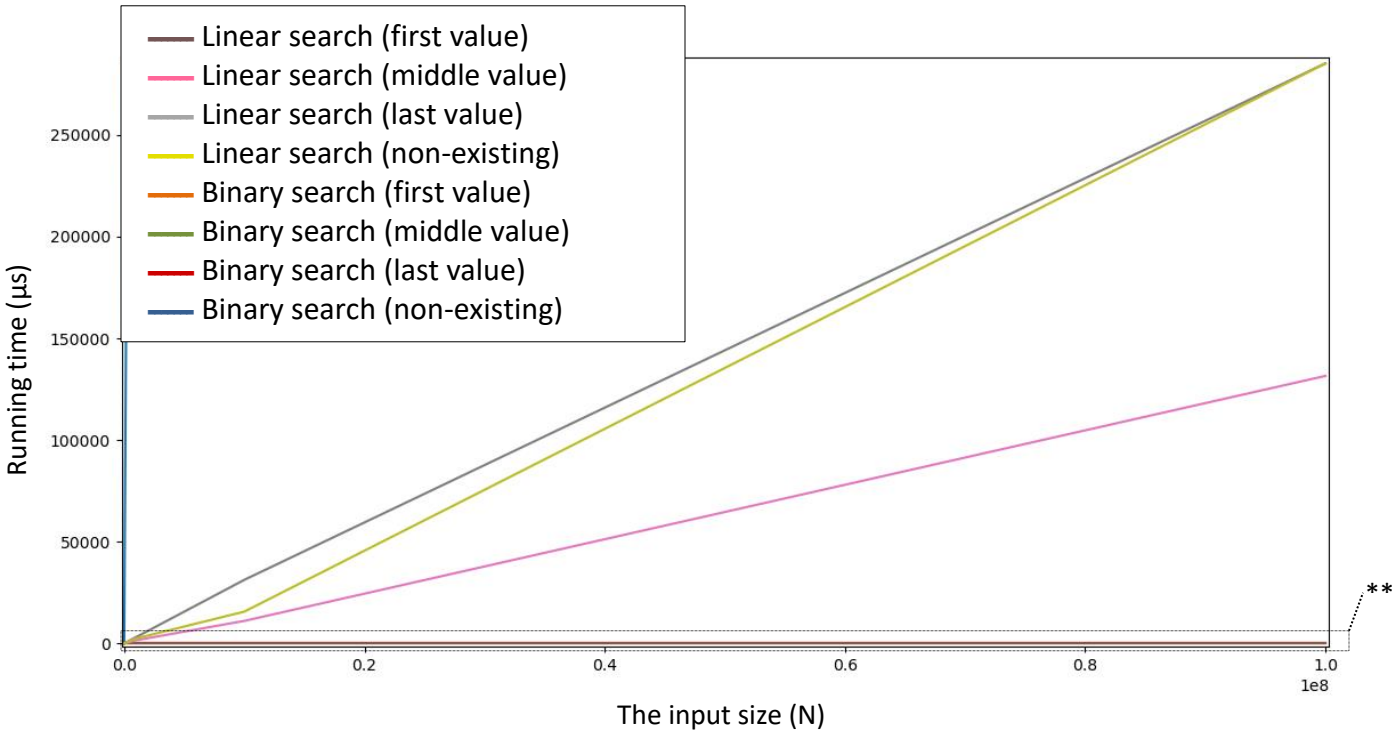
- **Observation**

I applied an algorithm that makes arrays of different sizes (starting with 10 and increasing with powers of 10) and uses both linear search and binary search respectively to look for items located in the beginning, middle, and end of the array, as well as a non-existing item. The run times in microseconds have been recorded in the table 1 and graphs 1, 2, and 3.

| Search Input / Input size | The first Value [1] | | Value at the middle [(array Size) / 2]* | | Value at the end [array Size]* | | Non existing value [array Size + 1]* | |
|---|---|---|---|---|---|---|---|---|
| | **Binary** | **Linear** | **Binary** | **Linear** | **Binary** | **Linear** | **Binary** | **Linear** |
| $10^1$ | 0.0163905 | 0.0045972 | 0.007496 | 0.011493 | 0.0213867 | 0.022987 | 0.0186885 | 0.0241868 |
| $10^2$ | 0.0354784 | 0.0044971 | 0.0077952 | 0.154905 | 0.0360827 | 0.2998 | 0.0352782 | 0.302831 |
| $10^3$ | 0.0565655 | 0.004697 | 0.0072957 | 1.49915 | 0.0550689 | 2.8981 | 0.0536669 | 2.9017 |
| $10^4$ | 0.0814499 | 0.0044977 | 0.0073956 | 13.9935 | 0.079951 | 27.988 | 0.0793511 | 28.01 |
| $10^5$ | 0.101238 | 0.0045972 | 0.0073952 | 139.93 | 0.09834 | 279.78 | 0.100837 | 282.707 |
| $10^6$ | 0.140913 | 0.0044971 | 0.0074958 | 1449.2 | 0.116828 | 2897.7 | 0.115728 | 2887.44 |
| $10^7$ | 0.167097 | 0.0046971 | 0.0075955 | 14490 | 0.142017 | 29984 | 0.142712 | 28980 |
| $10^8$ | 0.192182 | 0.0050971 | 0.0078954 | 144909 | 0.161101 | 285826 | 0.160701 | 283829 |

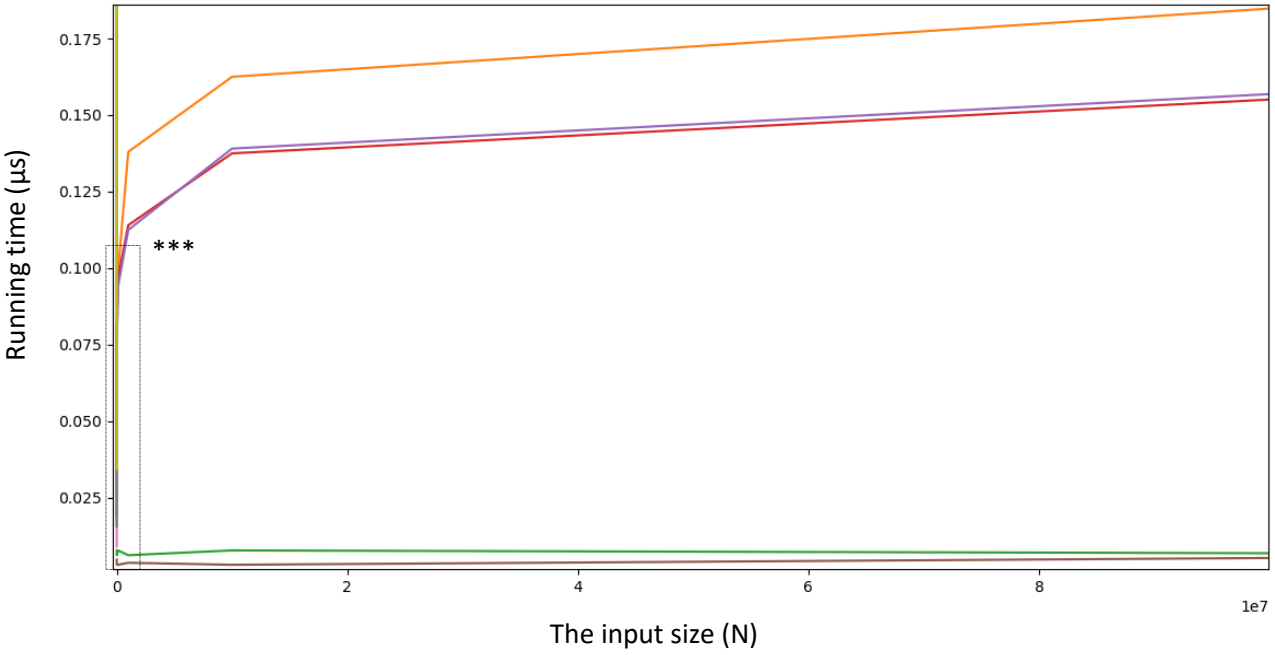
**Table 1 - Binary and linear search growth rates (experimental)**

* The array is populated with consecutive integers starting from 1 and incrementing by 1.



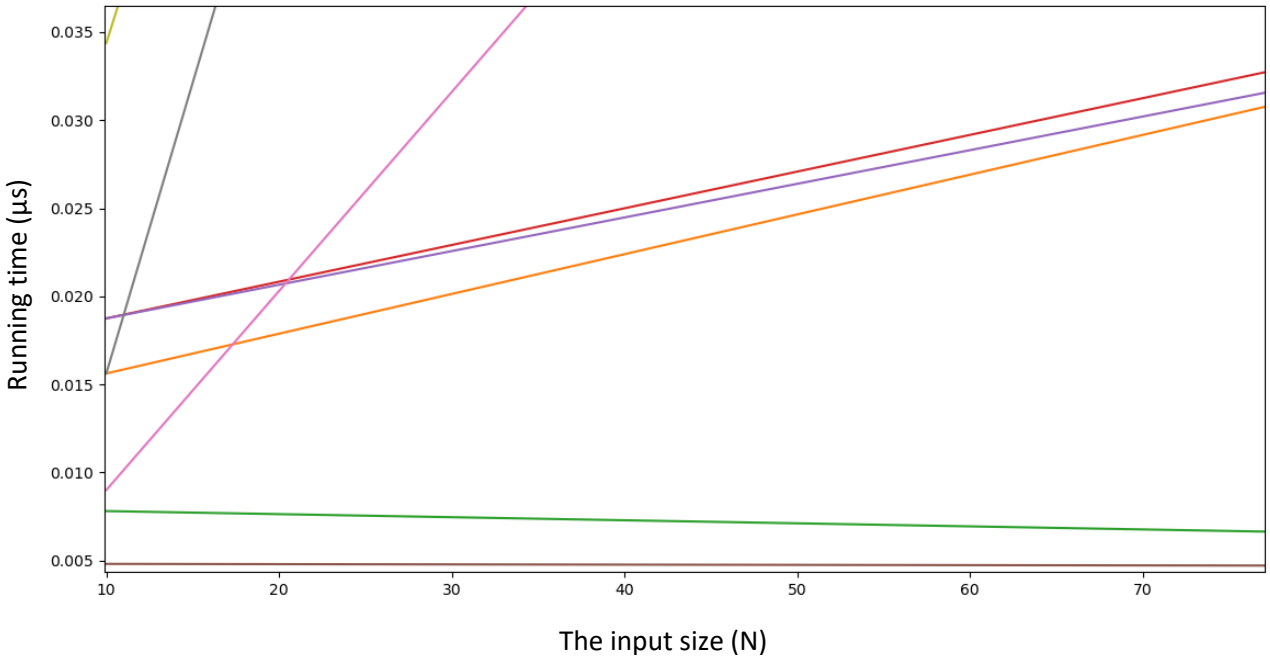

**Graph 1 - Binary and linear search growth rates (experimental)**

** This portion of the graph is magnified in graph 2




**Graph 2 – The starred portion of graph 1**

*** This portion of the graph is magnified in graph 3




**Graph 3 – The starred portion of graph 2**

Looking at the table and graphs we notice that in general linear search tends to run slower than the binary search, also in most of the cases the run time of linear search increases linearly, hence the name; however, binary search increases logarithmically.

Analyzing the worst case (light grey and yellow lines for linear; orange, red, and blue for binary,) we notice that it is different between the two search types; having the item at the end of the array or non-existing are the worst cases for the linear search, whereas having the items at the first index, the last index, or non-existing are the worst cases for the binary search. On the other hand, we notice that the run time complexities are linear and logarithmic for linear search and binary search respectively.

In terms of best case (dark grey line for linear; green line for binary,) we notice that they are again different: the first item for binary search and the item in the middle for binary search (only if the array size was odd.) Here both run at approximately the same speed, which is constant regardless of the size of the array. That is because in linear search, it is always the first item that is being retrieved; on the other hand, in binary if the array size was odd and is being divided by 2, then if the item being searched is the same as the item left in the middle after division, it is returned without any other iteration.

Average cases also differ: for binary being any item between the middle and the end exclusive (if size is even) and excluding the middle item as well (if size is odd), and for linear search being all items excluding the first and last. Here, the complexity is shown to be linear for linear search and logarithmic for binary search.

| Search Input / Input size | The first Value [1] | | Value at the middle [(array Size) / 2]* | | Value at the end [array Size]* | | Non existing value [array Size + 1]* | |
|---|---|---|---|---|---|---|---|---|
| | **Binary** | **Linear** | **Binary** | **Linear** | **Binary** | **Linear** | **Binary** | **Linear** |
| $10^1$ | 2.30259 | 1 | 2.30259 | 2.3979 | 1 | 10 | 10 | 11 |
| $10^2$ | 4.60517 | 1 | 4.60517 | 4.61512 | 1 | 100 | 100 | 101 |
| $10^3$ | 6.90776 | 1 | 6.90776 | 6.90875 | 1 | 1000 | 1000 | 1001 |
| $10^4$ | 9.21034 | 1 | 9.21034 | 9.21044 | 1 | 10000 | 10000 | 10001 |
| $10^5$ | 11.5129 | 1 | 11.5129 | 11.5129 | 1 | 100000 | 100000 | 100001 |
| $10^6$ | 13.8155 | 1 | 13.8155 | 13.8155 | 1 | 1e+006 | 1.00E+06 | 1e+006 |
| $10^7$ | 16.1181 | 1 | 16.1181 | 16.1181 | 1 | 1e+007 | 1.00E+07 | 1e+007 |
| $10^8$ | 18.4207 | 1 | 18.4207 | 18.4207 | 1 | 1e+008 | 1.00E+08 | 1e+008 |

**Table 2 - Binary and linear search growth rates (theoretical)**

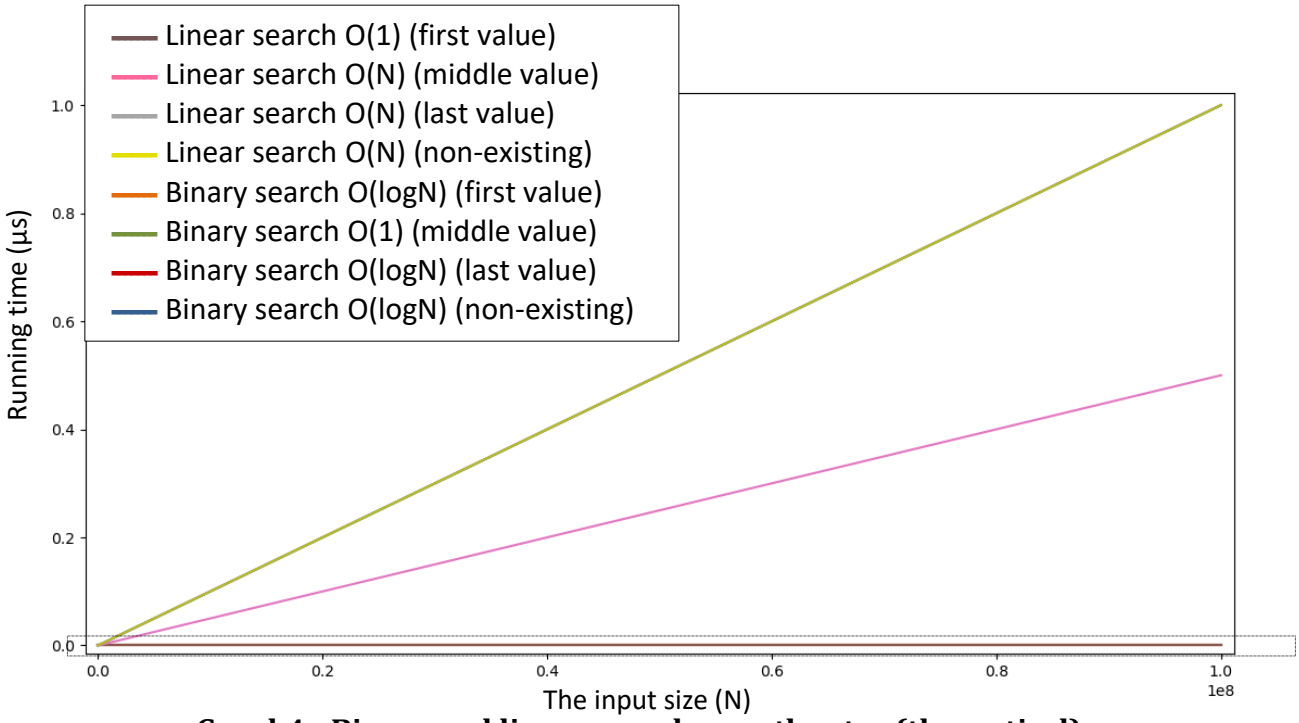
In fact, the results obtained experimentally are in accordance with the expected (theoretical) values shown in figure 2 and graphs 4, 5, and 6. The theoretical values are calculated using the following functions:

**Speed = N** *for average and worst cases of the linear search*
**Speed = logN** *for average and worst cases of the binary search*
**Speed = 1** *for best cases of the linear and binary search*

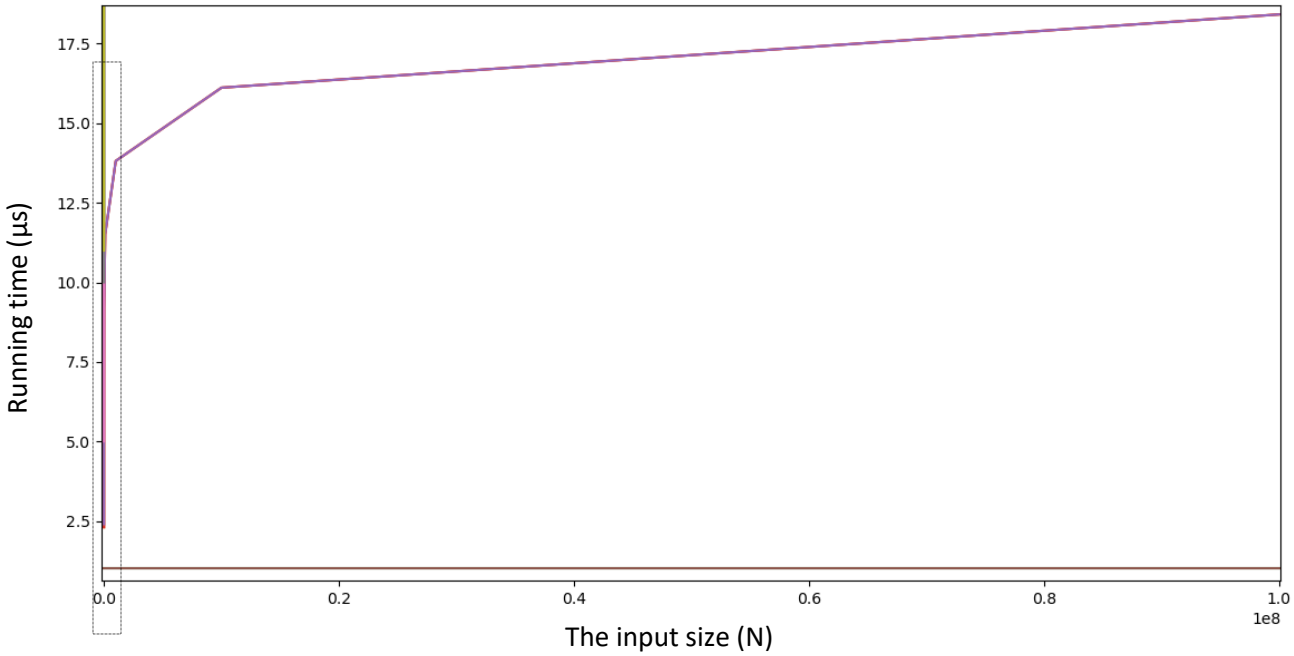
We can notice that there is a slight difference between the expected and the theoretical, especially for average cases since we tend to ignore the small differences in run time between the cases and make generalizations to achieve a general equation that explains the graph; therefore we

don’t get precise values for average cases. However, for worst cases we see that the theoretical matches almost exactly the experimental values.

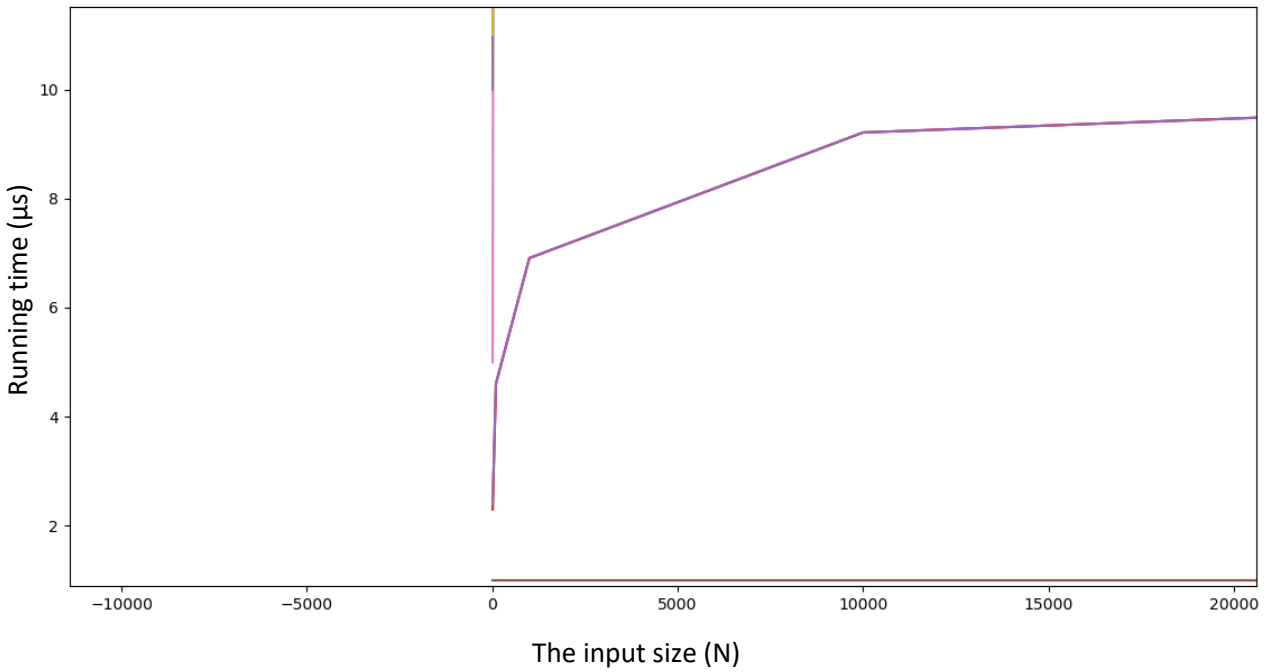
note: some lines show up like one line since they have the exact same values, according to table 2.




**Graph4 - Binary and linear search growth rates (theoretical)**




**Graph 5 – The starred portion of graph 4**




**Graph 6 – The starred portion of graph 5**

- **Computer Specifications:**

OS Name: Microsoft Windows 10 Pro

OS Version: 10.0.18362 N/A Build 18362

System Manufacturer: Hewlett-Packard

System Model: HP Pavilion g6 Notebook PC

System Type: x64-based PC

Processor(s): 1 Processor(s) Installed.

[01]: Intel64 Family 6 Model 58 Stepping 9 GenuineIntel ~2501 Mhz

BIOS Version: Insyde F.26, 6/3/2014

Total Physical Memory: 8,090 MB

Available Physical Memory: 3,010 MB

Virtual Memory: Max Size: 16,181 MB

Virtual Memory: Available: 10,873 MB

Virtual Memory: In Use: 5,308 MB