**Big O of the Functions**

Run time of the different algorithms based on an increasing input size.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Size** | **find1() in ms** | **find2() in ms** | **find3() in ms** | **find4() in ms** |
| 100000 | 0.357259 | 3.001176 | 0.103532 | 0.001642 |
| 110000 | 0.241051 | 3.408694 | 0.072165 | 0.001937 |
| 120000 | 0.388037 | 3.817854 | 0.11052 | 0.001975 |
| 130000 | 0.209313 | 4.092351 | 0.062153 | 0.00187 |
| 140000 | 0.471758 | 4.46882 | 0.139666 | 0.002007 |
| 150000 | 0.213728 | 4.948405 | 0.071389 | 0.002078 |
| 160000 | 0.171858 | 5.347635 | 0.044505 | 0.001866 |
| 170000 | 0.308422 | 5.656596 | 0.086971 | 0.002093 |
| 180000 | 0.136762 | 5.955043 | 0.039374 | 0.001662 |
| 190000 | 0.699818 | 6.486262 | 0.20533 | 0.00189 |
| 200000 | 0.544046 | 6.509009 | 0.15629 | 0.002074 |

Graphically depicts and compares the run time for each function as input size increases

Big O of each function:

* find1(): The graph shows that the find1() function is linear in nature hence the Big O of find1() is O(n). The algorithm has just one for loop that does a linear search which has a complexity of O(n), further backing up the graphs depiction of find1() being Big O O(n).
* find2(): The graph shows that the find2() function is exponentially increasing in nature hence the Big O of find2() is O(n2). The algorithm makes a deep copy of the list which worse-case scenario is O(n2). The built in sort takes O(nlogn) and the binary search takes O(logn). Hence, the overall final Big O complexity is O(n2).
* find3(): The graph shows that the find3() function is highly linear in nature hence the Big O of find3() is O(n). The algorithm used the in built in which iterates through the list hence the Big O of find3() based on the graph and algorithm is O(n).
* find4(): The graph shows that the find4() function is logarithmic in nature hence the Big O of find4() is O(logn). The algorithm performs a binary search on a pre-sorted list, this has a complexity of O(logn). Therefore, based on the graph and analysis of the algorithm, find4() has a Big O of O(logn).