# Analysis

The different data structures were compared according to the time taken to add the element at the start. It has been compared with a large amount of data, ranging from 10000 to 1000000. The different data structures tested here are ArrayList, LinkedList, TreeSet, and HashSet. Please note that the performance was only analyzed by adding the elements at the start.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Amount of Data | Array List | LinkedList | Tree Set | HashSet |
| 10000 | 0.005 | 0.001 | 0.012 | 0.02 |
| 30000 | 0.038 | 0.03 | 0.015 | 0.013 |
| 100000 | 0.422 | 0.007 | 0.018 | 0.013 |
| 300000 | 4.579 | 0.017 | 0.059 | 0.036 |
| 1000000 | 79.343 | 0.018 | 0.198 | 0.139 |

Table 1: Comparison of time taken by different data structures to add elements at the start.

In the table above, the first column shows the amount of data. The other columns show the amount of time taken by the data structure to add elements at the start. All the time was entered on the first try, which could also bring some variations in the result, as it has been noticed that it is much faster for HashSet, TreeSet, and LinkedList on the second try. Note that the situation would be completely different if the elements were added in the middle.

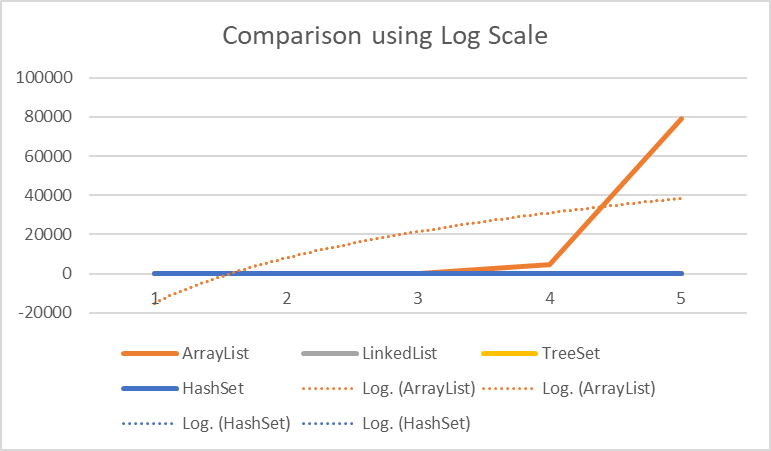


Figure 1: Graph generated using the table.

In the graph, the y-axis shows the amount of data and the x-axis shows the time in seconds. From the graph, it is clear that the ArrayList took the most time to add elements from all four data structures. There were also difficulties in generating the graph because when the amount of data was 1000000, the array list took about 79 seconds to add elements at the start. This made the X-axis, the time-axis, quite long. While all other data structures took only 0.1 seconds at most, which now seems quite irrelevant.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Amount of Data | ArrayList | LinkedList | TreeSet | HashSet |
| 100000 | 0.003 | 0.005 | 0.000 | 0.000 |
| 300000 | 0.006 | 0.005 | 0.000 | 0.000 |
| 1000000 | 0.017 | 0.013 | 0.000 | 0.000 |

Table 2: Comparison of time taken to find objects

From the table, it is clear that the HashSet and TreeSet do not require any time to find objects. Therefore, they would be the most suitable for the scenario of finding objects.

In conclusion, the most efficient and fastest data structure for adding elements was LinkedList. But theoretically, the HashSet would also work quite efficiently because it does not depend on the amount of data. Overall, HashSet would be the most efficient if we needed to work with both operations: Add and Find.