**Analysis of all types of symbol tables**

**Problem statement:**

By considering a lookup CSV file to make an analysis of all types of symbol tables and justify which algorithm is best for different scenarios.

**Approach:**

* There are many types of symbol tables to use. In this problem, we are going to solve by taking different kinds of data sets.
* After checking each and every symbol table analysis we have to decide which symbol table performance is best based on the number of key-value pairs.
* For each and every symbol table we are going to check about search and insert time complexities.
* In the end, we can decide which symbol table is efficient.

**Symbol tables:**

* Sequential search symbol table
* Binary search symbol table
* Balanced symbol table
* Red black BST symbol table
* Linear probing hash table
* Separate chaining hash table

**DataSets:**

* 10000 entries - 2 MB size
* 2 lakh entries - 15MB size
* If I am taking the input as 1 Million, it is taking so long time to run for binary search ST and sequential search ST but for remaining symbol tables I am getting the output in minimum time.

**Procedure:**

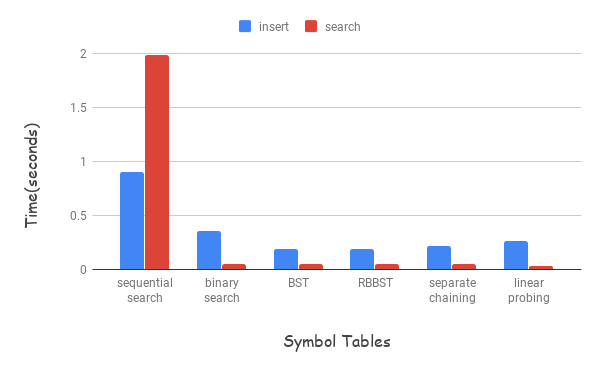
* First of all, write a lookup CSV program to check the performance of each kind of symbol table.
* By giving different types of datasets to each symbol table check how much time it takes for search operation and inserts operation.
* With help of stopwatch class in Java, we can user the timer and analyze the data.

**Analysis:**

**10000 key-value pairs:**

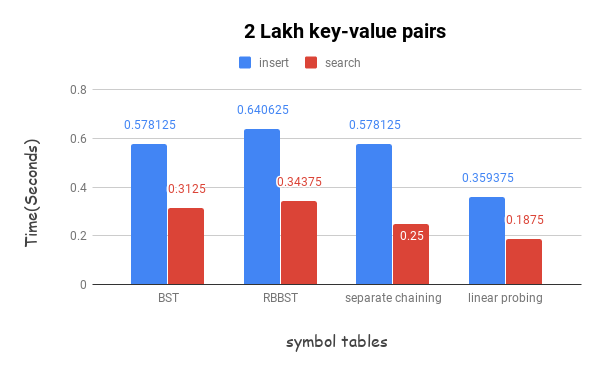
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | sequential search | binary search | BST | RBBST | separate chaining | linear probing |
| insert | 0.90625 | 0.359375 | 0.1875 | 0.1875 | 0.21875 | 0.265625 |
| search | 1.984375 | 0.046875 | 0.046875 | 0.046875 | 0.046875 | 0.03125 |

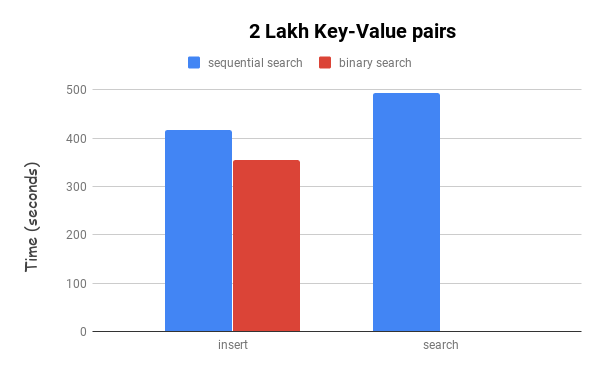
(Time in seconds)



**2 lakh key-value pairs:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | sequential search | binary search | BST | RBBST | separate chaining | linear probing |
| insert | 416.859375 | 354.890625 | 0.578125 | 0.640625 | 0.578125 | 0.359375 |
| search | 493.328125 | 0.265625 | 0.3125 | 0.34375 | 0.25 | 0.1875 |



****

**References:**

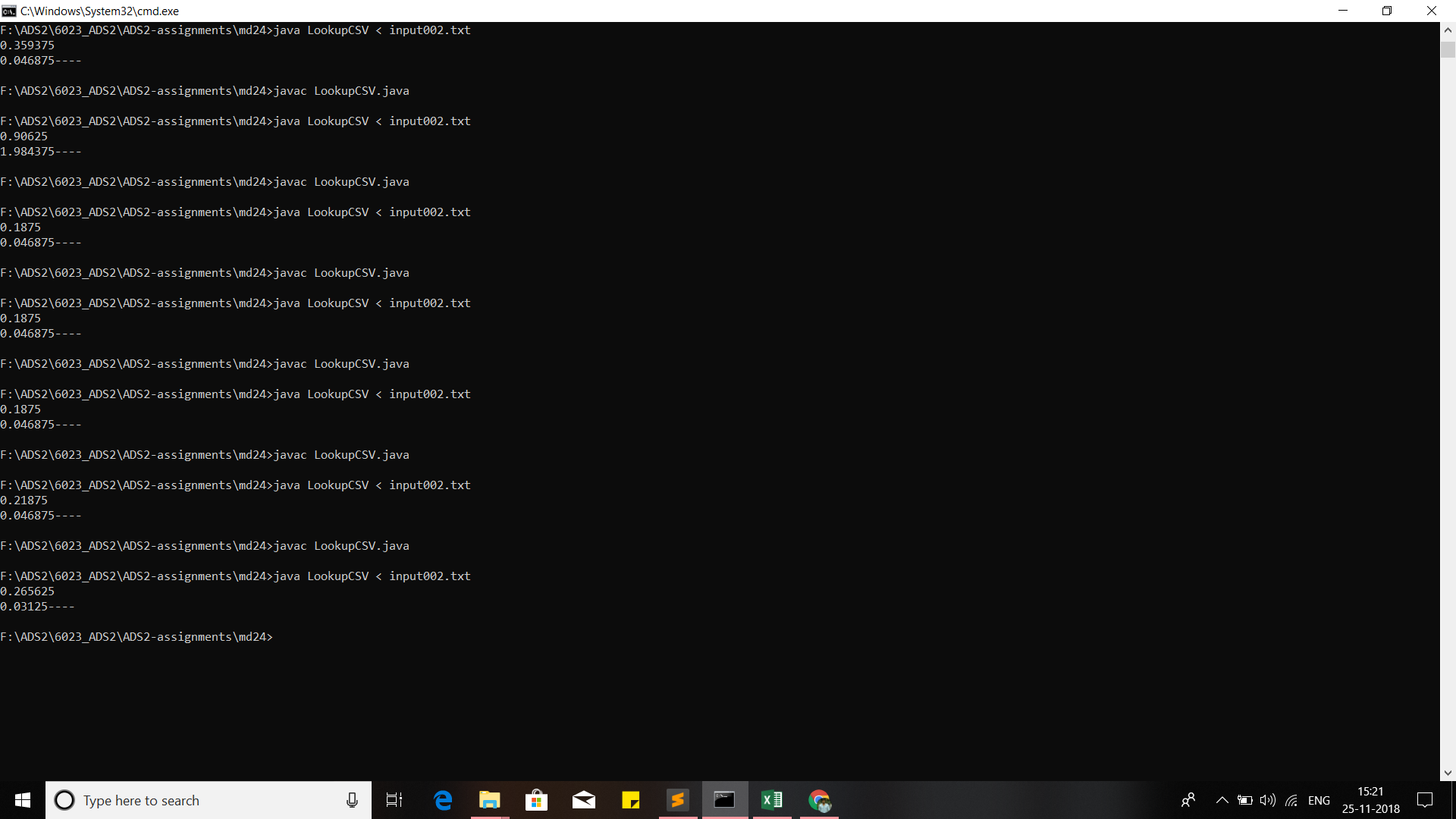
* Algs4.jar

**Conclusion:**

From the above analysis, we can determine that Red black BST and linear probing hash table more advantageous and runs faster when compared to other symbol tables. In the worst case time complexity for Red black BST is O(log N) and for the linear probing hash table, the worst case complexity is O(N).

**Screenshots:**

For 10000 entries-



For 2 Lakh inputs:

