ANALYSIS OF SYMBOL TABLES

APPROACH:

This is a study of a Data Structure “Symbol Tables”, which are used to store data in key values pairs. We analyze symbol tables by reading in a CSV file (using LookupCSV) and time complexities are calculated.

PROCEDURE:

- ➔ **Step 1:** Consider different sizes of data sets with increment in number for entries. For example 100 entries, 10,000 entries, 1 Lakh entries and so on.

- ➔ **Step 2:** Sequential Search Symbol Table, Binary Search Symbol Table, Binary Search Tree, Red - Black BST, Separate Chain hashing Symbol TAble and Linear Probing Symbol Table are used to perform the analysis and lookupCSV class is used to read CSV files and calculate complexities.

- ➔ **Step 3:** Create an input file to read the CSV file and to take queries. Here for “N” number of inputs, “N” number of queries are given.

- ➔ **Step 4:** Insert all key-value pairs into all Symbols Tables and record the time taken for insertions.

- ➔ **Step 5:** Repeat the same for queries.

- ➔ **Step 6:** Plot graphs with the values obtained to understand the outcomes of the Symbol Table’s complexities.

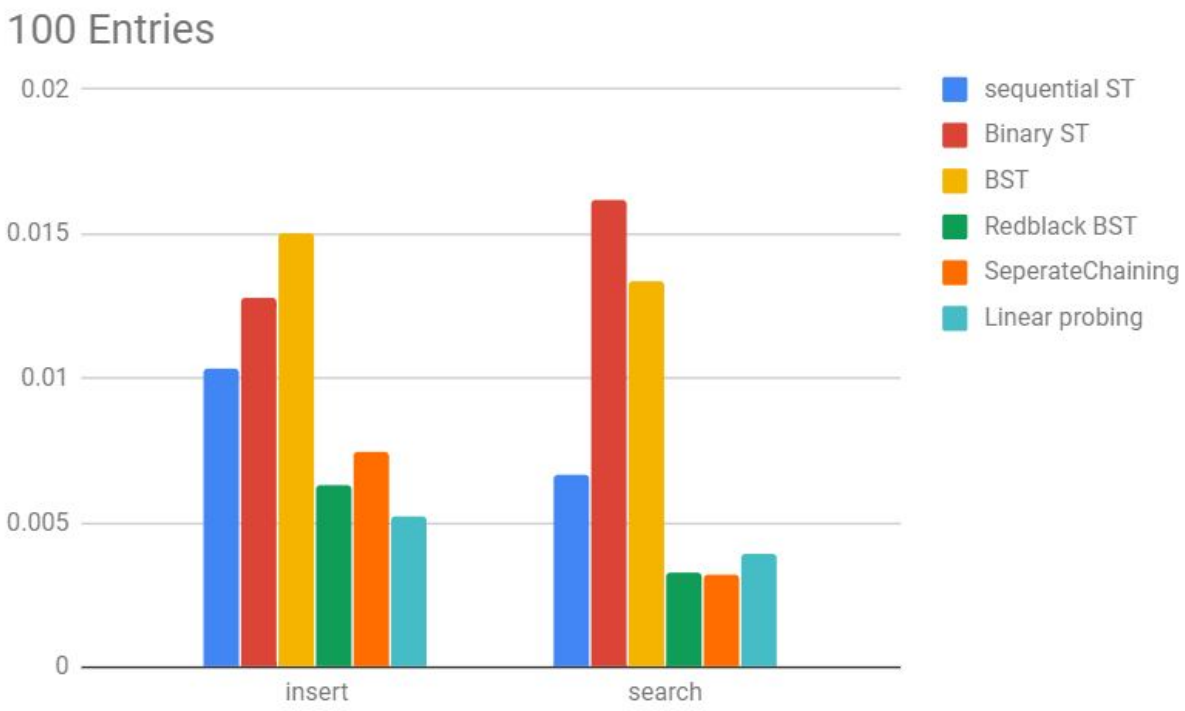
DATA SETS:

Filename	Size	Number of Entries
elements.csv	5kb	100
googleplaystore.csv	1MB	10,000
1Lsales.csv	12MB	1,00,000
2Lsales.csv	30MB	2,00,000

ANALYSIS:

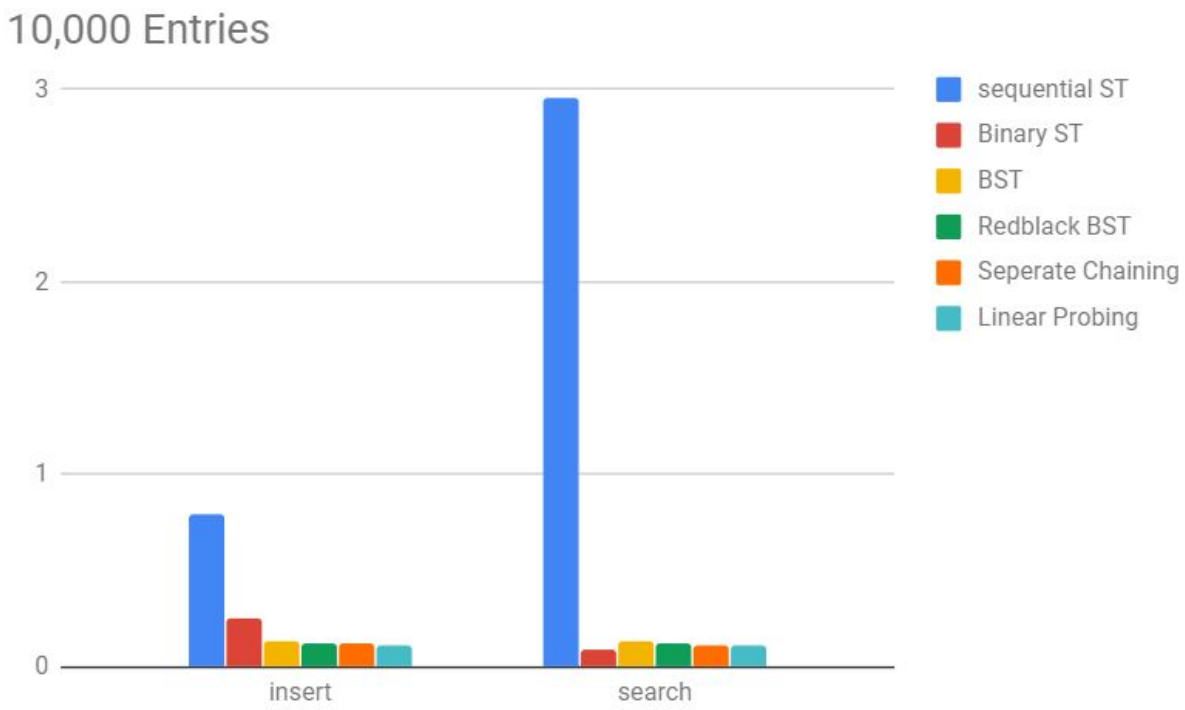
- **100 entries :**

	sequential ST	Binary ST	BST	Redblack BST	Separate Chaining	Linear probing
insert	0.010324308	0.012783039	0.01499451	0.006331118	0.007472323	0.005234121
search	0.006631327	0.016206141	0.013361353	0.003297671	0.003226218	0.003930989



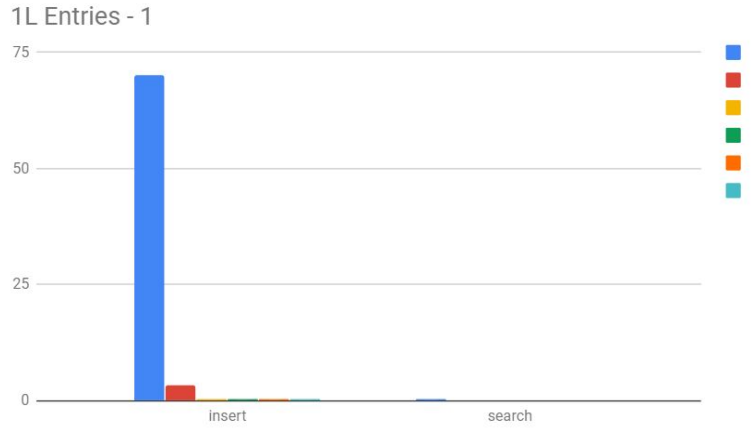
- **1000 entries:**

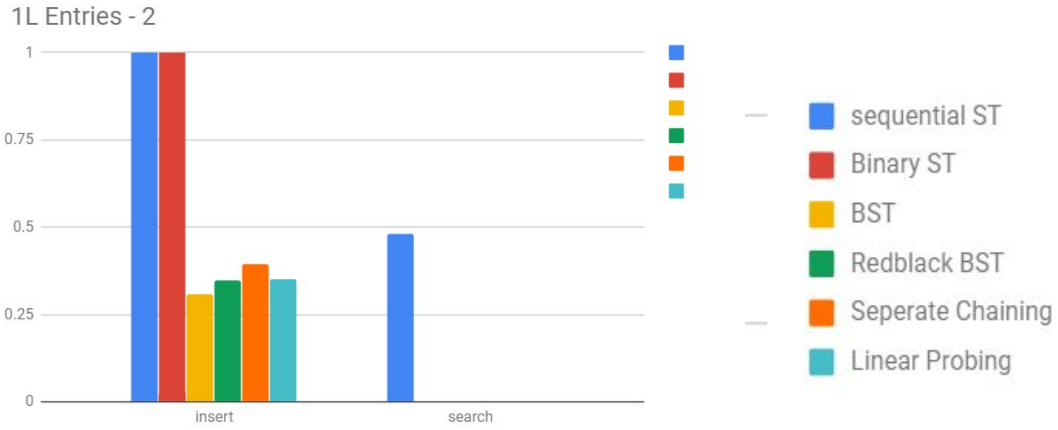
	sequential ST	Binary ST	BST	Redblack BST	Separate Chaining	Linear Probing
insert	0.790692701	0.2501259439	0.131131664	0.126236819	0.118896609	0.116029203
search	2.957555395	0.094446033	0.130524049	0.124292658	0.116564849	0.114640736

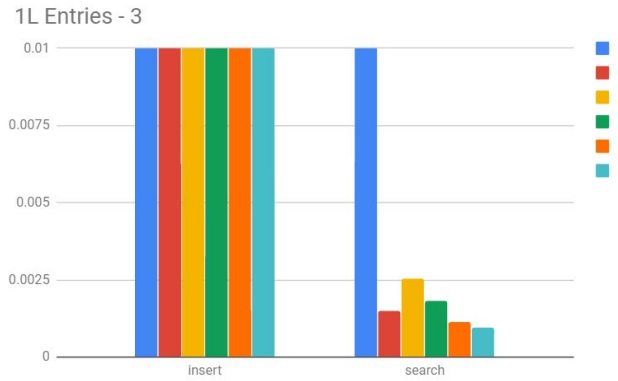


- **1 Lakh Entries:**

	sequential ST	Binary ST	BST	Redblack BST	Separate Chaining	Linear Probing
insert	70.12472858	3.314956162	0.308729396	0.348381133	0.39346182	0.350184444
search	0.480837521	0.001498988	0.002544579	0.001802281	0.001120129	9.67E-04







- **2 Lakh Entries :**

	sequential ST	Binary ST	BST	Redblack BST	Separate Chaining	Linear Probing
insert	84.13386747	4.076218108	0.57778598	0.440315487	0.837414769	0.481089409
search	0.452490911	0.005813977	0.002594443	0.002223808	0.002104033	1.69E-03

- For less number of entries, we can observe same time complexities for all symbol tables. Hence any symbol table, according to the application can be used
- As the size increases, we can see that Separate Chaining and Linear probing are less in time complexity. In an average case, insertion and deletion can happen in constant time.
- The major difference can be observed from the values plotted by 1 lakh entries graph.

REFERENCES:

- https://algs4.cs.princeton.edu/cheatsheet/ - time complexties
- https://stackoverflow.com/questions/5204051/how-to-calculate-the-running-time-of-my-program - to Calculate execution time

CONCLUSION:

Separate Chaining and Linear probing are the efficient symbol tables for large inputs. Red black BST's are better in case of worst-case inputs. BST, Binary Search ST, and Sequential Search ST can be used with fewer data sizes.