Project Report

Design and Analysis of Algorithms CSE5311

Implementation & comparison of search algorithms

Prof. Negin Fraidouni

By,

Name: Eldho Joy

UTA ID: 1001986525

Project Abstract:

The objective of this project is to implement various search algorithms
The algorithm will give the search element index or returns if the element is present in the list. The algorithm searches through the list or files which is given as an input.

Efficiency is one of the main features of an algorithm, this includes efficiency in memory and runtime. This project is designed to display the important properties of different search techniques including their complexity, stability, and memory constraints. I articulated time complexity analysis on various search techniques

Implementation:

I implemented various search algorithms:

1. Linear Search
2. Binary Search
3. Binary Search Tree
4. Red Black Tree

The following data structures are used to implement the project – Array List, Set (LinkedHashSet) and Tree

Algorithm	Data Structure
Linear Search	ArrayList
Binary Search	ArrayList
Binary Search Tree (BST)	ArrayList, Tree
Red Black Tree	ArrayList, Tree

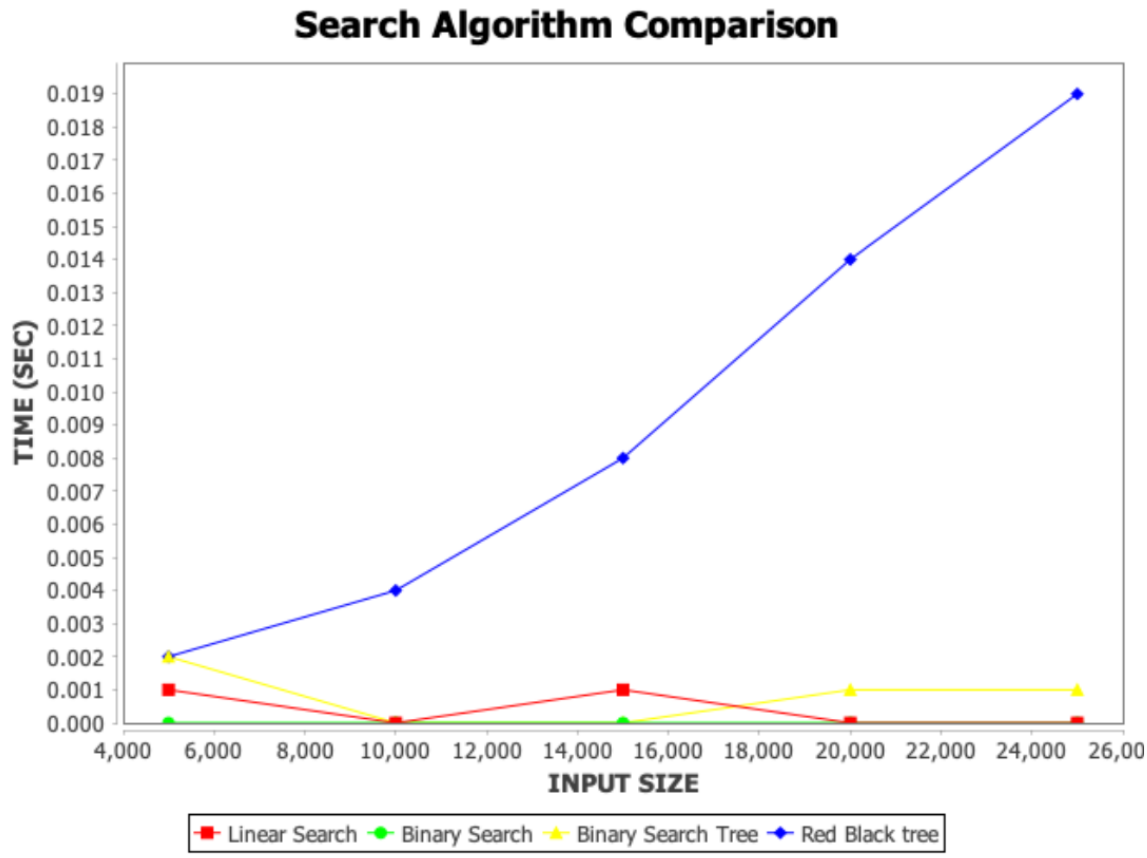
Components:

Search Algorithms are implemented in separate java file and these algorithms are compared for finding time complexity. System time is used for finding the time the algorithms took to execute the project and Java free chart is used to plot the time-complexity of each algorithm.

Inputs are generated using random generator. The inputs are then fed into different algorithms for searching. To avoid duplicates I used set data structure for the input.

I implemented the project using java programming language. I used Eclipse IDE, PHP, HTML and XAMPP for comparison and showing it in GUI. Sytem time is used to calculate the time of execution of each algorithm.

Run Time vs Input size:



Observation:

The run time of different algorithms with different size of input are recorded below

Algorithm	i/p size 5000 nos	i/p - 15000 nos	i/p with 1,50,0000	i/p - 15,00,000
Linear Search	0.001	0.001	0.002	0.006
Binary Search	0.001	0.001	0.001	0.001
Binary Search Tree	0.0	0.002	0.016	0.093
Red Black Tree	0.001	0.005	0.039	0.677

The best, average and worst case are as follows

Algorithm	Best	Average	Worst
Linear Search	$\Omega(1)$	$\theta(N)$	$0(N)$
Binary Search	$\Omega(1)$	$\theta(N)$	$0(N)$

Binary Search Tree	$\Omega(1)$	$\theta(N)$	$0(N)$
Red Black Tree	$\Omega(1)$	$\theta(N)$	$0(N)$

Comparing all the algorithms we can see that binary search is the best searching technique in terms of its time complexity. For small inputs Binary search tree and red black will work faster and for larger inputs the algorithms will take more time for execution. The algorithm which takes more time is Red Black tree as it takes more time to form the tree and perform rotations in order to maintain the balance

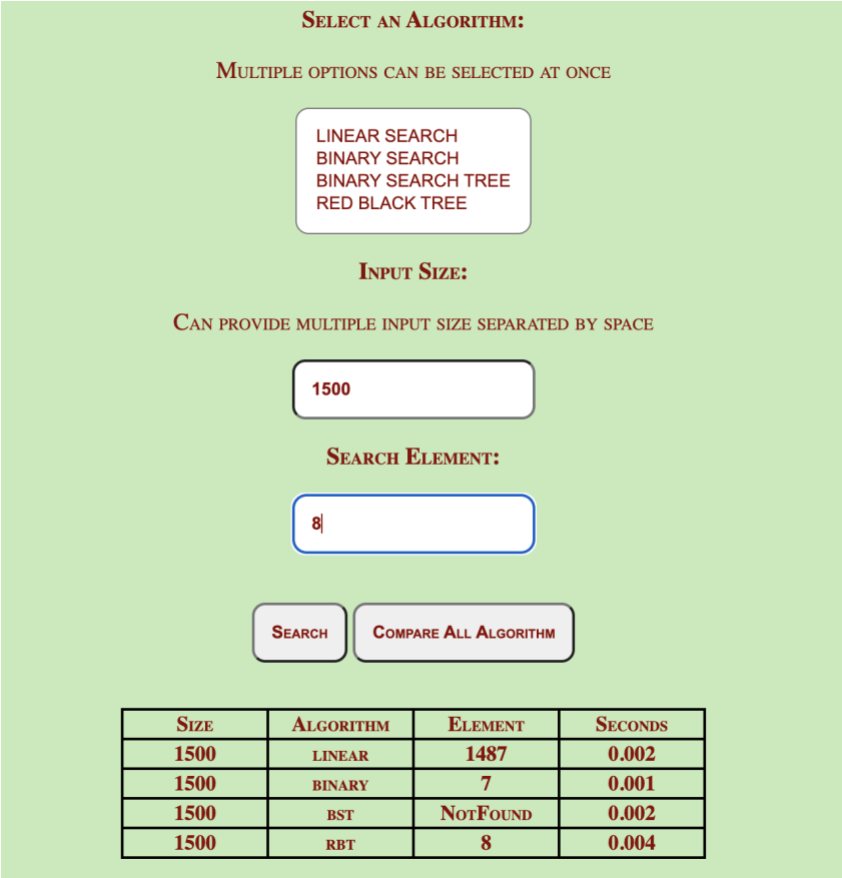
Tools and Languages Used:

Java, HTML, PHP, XAMPP

GUI:

GUI is built using HTML and PHP and we can select different types of algorithm and multiple input size for comparison. The run time will be displayed after running the algorithm and a graph is also generated and saved

Attached GUI image



Conclusion:

The project gives an insight on various search algorithms and its corresponding best case, average case and worst case time complexities

By analysing and plotting the Run-time Vs Input-Size graph, we come to know the which algorithm to be used given a scenario, by analysing input size and time taken.

Future Enhancement:

We can optimize linear searching algorithm by sorting it and then performing searching. For the case of Red Black Tree we will sort it and then insert the elements to the tree so that we can avoid the case of element not found.