Al-Imam Mohammad Ibn Saud Islamic University
Collage of Computer and Information Sciences
Department of Computer Sciences

جامعة الإمام محمد بن سعود الإسلامية
كلية علوم الحاسب والمعلومات
قسم علوم الحاسب

Design and Analysis of Algorithms

CS215

Course Project

By

Yasser Abdalmohsen alboraidi (437017295)

Supervisor

DR. Hashemi bin Ahmed bin Nasser

1. Introduction

In this project, we will create algorithms to search and compare between them, the goal of this comparison is to display the worst and best times for each algorithm by graphing each

algorithm and determining the best algorithm to search for the elements. And we're going to compare these algorithms (linear search, binary search, TSA search)

2. Theoretical part

- **The design of the TSA algorithm**

```
def TSA (start, end, key, array):
    if (end >= start)
        mid1 -> start + (start - end) /3
        mid2 -> end - (end -start) /3
        if (array[mid1] = key)
            return mid1
        if (array[mid2] = key)
            return mid2
        if (key < array[mid1]):
            return TSA (start, mid1 - 1, key, array)
        else if (key > array[mid2]):
            return TSA (mid2 + 1, end, key, array)
        else:
            return TSA (mid1 + 1, mid2 - 1, key, array)
    return -1
```

- **The recurrence relation of the TSA algorithm**

  The recurrence relation of the TSA algorithm is “Iterative method”

- **The analysis of the TSA algorithm**

  the associated recurrence equation

  $T(n) = T(n/3) + O(1)$

3. Practical part

- **The implementation of linear search algorithm**

  Code :

```
public class LinearSearch{
public static int linearSearch(int[] arr, int key){
        for(int i=0;i<arr.length;i++){
            if(arr[i] == key){
                return i;
            }
        }
        return -1;
    }
    public static void main(String a[]){
        int[] al= {10,20,30,40,50,60,70,80,90};
        int key = 50;
        System.out.println(key+" is found at index: "+linearSearch(al, key));
    }
}
```

  The run screenshots :

```
run:
50 is found at index: 4
BUILD SUCCESSFUL (total time: 0 seconds)
```

- **The implementation of binary search algorithm**

Code :

```
package binarysearch;
class BinarySearch {
        int binarySearch(int arr[], int x)
        {
                int l = 0, r = arr.length - 1;
                while (l <= r) {
                        int m = l + (r - l) / 2;

                        if (arr[m] == x)
                                return m;

                        if (arr[m] < x)
                                l = m + 1;
                        else
                                r = m - 1;
                }
                return -1;
        }
        public static void main(String args[])
        {
                BinarySearch ob = new BinarySearch();
                int arr[] = {10,20,30,40,50,60,70,80,90 };
                int n = arr.length;
                int x = 50;
                int result = ob.binarySearch(arr, x);
                if (result == -1)
                        System.out.println("Element not present");
                else
                        System.out.println("Element found at "
                                                + "index " + result);
        }
}
```

The run screenshots :

```
run:
Element found at index 4
BUILD SUCCESSFUL (total time: 0 seconds)
```

- **The implementation of the TSA algorithm**

  Code :

```
class TSA {
    static int totalComparison = 0;
    static int TSA(int s, int e, int key, int ar[]) {
        if (e >= s) {
            totalComparison++;
            int mid1 = s + (e - s) / 3;
            int mid2 = e - (e - s) / 3;

            if (ar[mid1] == key) {
```

The run screenshots :

```
run:
Enter the length of array you want to generate: 10


4 12 20 28 36 44 52 60 68 76

Enter the number you want to search: 36
Total Comparison by TSA: 3
Index of 36 is 4
BUILD SUCCESSFUL (total time: 18 seconds)
```

4. The analysis

Table A : For each strategy (TSA, BS, or LS) and each size (100, 200, …, 1000) of the array, provide the average number of comparisons performed by the strategy.

Strategy	N=100	N=200	N=300	N=400	N=500	N=600	N=700	N=800	N=900	N=1000
TSA	15	17	20	22	21	23	23	24	25	27
BS	9	12	13	14	15	16	16	16	16	16
LS	53	119	167	233	304	361	351	477	541	625

Table B: For each strategy and each size of the array, provide the best case of the strategy (case where the minimum number of comparisons performed by the strategy is reached).

Strategy	N=100	N=200	N=300	N=400	N=500	N=600	N=700	N=800	N=900	N=1000
TSA	2	5	6	6	9	11	9	5	11	12
BS	9	9	7	15	9	7	9	11	7	11
LS	3	4	71	13	8	15	25	10	12	28

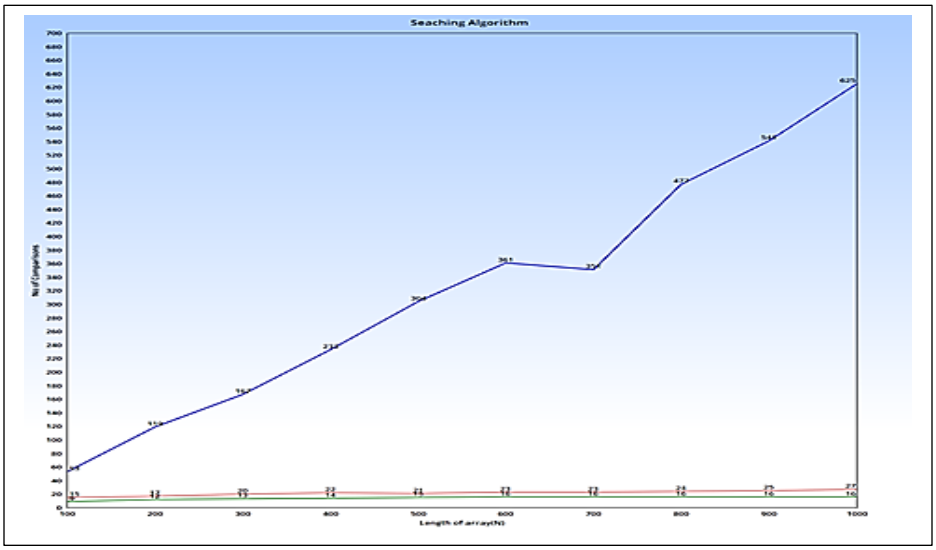
Table C: For each strategy and each size of the array, provide the worst case (case where the maximum number of comparisons performed by the strategy is reached).

Strategy	N=100	N=200	N=300	N=400	N=500	N=600	N=700	N=800	N=900	N=1000
TSA	21	21	25	23	22	25	21	24	26	25
BS	13	15	17	17	17	19	19	19	19	19
LS	99	199	267	386	494	591	691	727	882	951

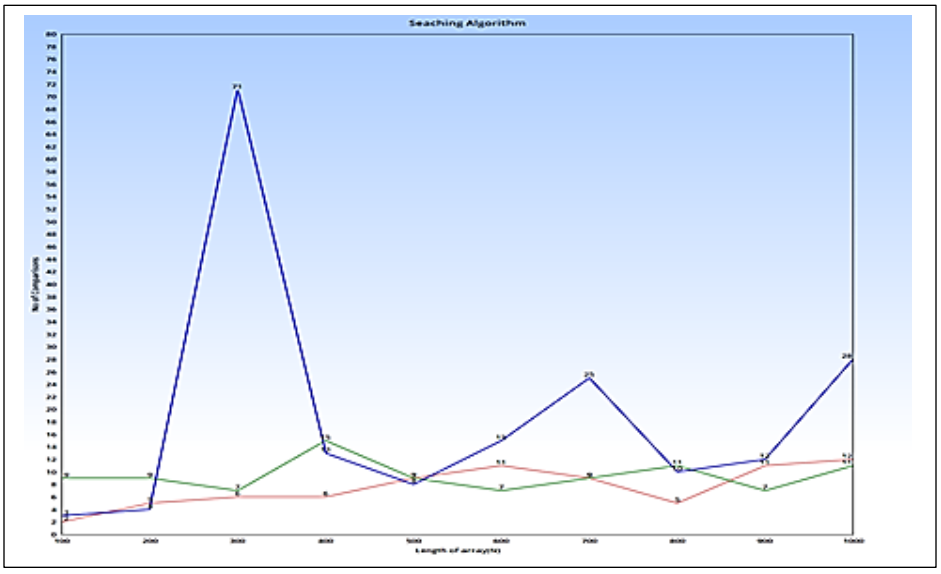
- **The figures A,B,C.**

The illustrative figures from the three tables A, B and C showing the behaviors of the three strategy when the size of the arrays N is increased; where X-axis represents the sizes of the arrays (N) and the Y-axis represents the number of comparisons of the strategy (#NC).

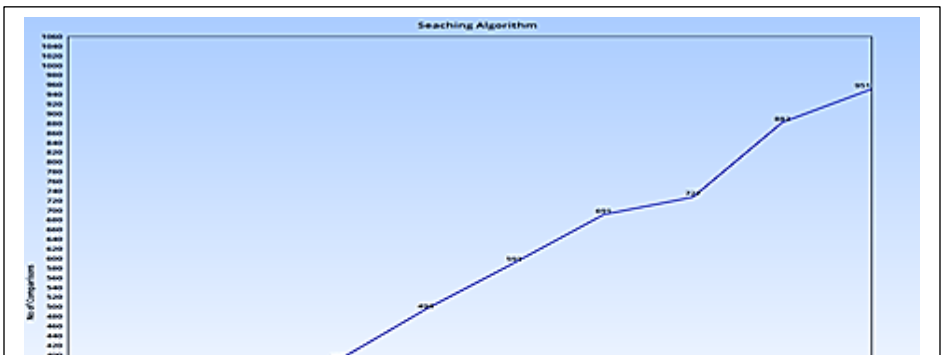
A


Seaching Algorithm
No of Comparisons
Length of array(N)

B


Seaching Algorithm
No of Comparisons
Length of array(N)

C


Seaching Algorithm
No of Comparisons

- **Interpret and analyze your results; compare and explain the behavior of the strategies.**

If we talk about linear search it takes a lot more time as compared to binary search and TSA because it is a sort of brute force algorithm which is too much expensive in respect of computation. On the other hand binary search and TSA both use almost same approach for searching and hence they're more efficient then linear search but binary search is still more efficient as compared to ternary search as there are fewer number of comparisons required in case of bigger array. In our case TSA is working more efficiently then binary search but in case of bigger arrays binary search is more efficient.

- **Check whether the theoretical and practical analysis are correlated .**
  for this purpose, we can draw the asymptotic function of the TSA strategy (part I, 1.b) and compare it with our results.

  Complexity = O(log3 n))

5. Conclusion

Finally, after comparing these algorithms (linear search, binary search, TSA search), we noticed that the algorithm TSA search is better for searching for items due to shorter time and less effort in searching