Batch: B4 Roll. No.: 16010122828

Experiment: 8

Grade: AA / AB / BB / BC / CC / CD /DD

Title: Implementation of Searching algorithms

Objective: To understand various searching methods

Expected Outcome of Experiment:

CO	Outcome
CO3	Demonstrate sorting and searching methods.

Websites/books referred:

1. stackoverflow
2. geeksforgeeks
3. Rema Thareja

Abstract: -

(Define Search)

Searching is the process of finding a given value position in a list of values. It decides whether a search key is present in the data or not. It is the algorithmic process of finding a particular item in a collection of items. It can be done on internal data structure or on external data structure.

Searching methods(Linear search, binary search, hash search):

Linear Search: A linear search or sequential search is a method for finding an element within a list. It sequentially checks each element of the list until a match is found or the whole list has been searched.

Binary Search: Binary search, also known as half-interval search, logarithmic search, or binary chop, is a search algorithm that finds the position of a target value within a sorted array. Binary search compares the target value to the middle element of the array.

Hash Search: One of the most common approaches is to use a hash function to transform one or more characteristics of the searched-for item into a value that is used to index into an indexed hash table. Hash-based searching has better average-case performance than the other search algorithms described.

Algorithm Binary Search:

Condition : Only applicable to sorted arrays.

- Compare x with the middle element.
- If x matches with the middle element, return the mid index.
- Else If x > mid element, search in the right half.
- Else search in the left half.

Hashing: (Define hashing, collision and list collision handling methods)

Hashing: Hashing is the process of converting a given key into another value. A hash function is used to generate the new value according to a mathematical algorithm.

Collision: A collision occurs when more than one value to be hashed by a particular hash function hash to the same slot in the table or data structure (hash table) being generated by the hash function.

Collision handling methods:

- Open Hashing (Separate chaining)
- Closed Hashing (Open Addressing)
- Liner Probing
- Quadratic probing
- Double hashing

Code and output screenshots for Binary search:

CODE:

```
#include <bits/stdc++.h>
using namespace std;

int binarySearch(int arr[], int l, int r, int x)
{
    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;
}  int main()
{
    int n;
  cout<<"Enter number of elements: ";
  cin>>n;
  int arr[n];
  cout<<"Enter array elements in sorted order: ";
  for(int i=0;i<n;i++)
  {   cin>>arr[i];
   }
```

```
   int x;
  cout<<"Enter the number to be searched: ";
   cin>>x;
    int result = binarySearch(arr, 0, n - 1, x);
    if(result == -1)
        cout << "Element is not present in array";
        else
        cout << "Element is present at index " << result;
         return 0;
}
```

OUTPUT:

```
Enter number of elements: 5
Enter array elements in sorted order: 3
5
7
9
11
Enter the number to be searched: 0
Element is not present in array
```

```
Enter number of elements: 5
Enter array elements in sorted order: 3
4
7
8
9
Enter the number to be searched: 8
Element is present at index 3
```

Post lab questions-

a. Compare and contrast various collision handling methods.

- In Open Hashing each cell in the array points to a list containg the collisions.

The hashing has produced the same index for all items in the linked list.

- In Closed Hashing you use only one array for everything. You store the collisions in the same array. The trick is to use some smart way to jump from collision to collision until you find what you want. And do this in a reproducible / deterministic way.

Comparison of Open Addressing Techniques-

	Linear Probing	Quadratic Probing	Double Hashing
Primary Clustering	Yes	No	No
Secondary Clustering	Yes	Yes	No
Number of Probe Sequence (m = size of table)	m	m	m^2
Cache performance	Best	Lies between the two	Poor

b. Compare linear search and binary search

Difference Between Linear Search and Binary Search

Definition

Linear search is an algorithm to find an element in a list by sequentially checking the elements of the list until finding the matching element. Binary search is an algorithm that finds the position of a target value within a sorted array. Thus, this is the main difference between linear search and binary search.

Synonyms

Sequential search is another term for linear search whereas half-interval search or logarithmic search refers to the same binary search.

Time complexity

The time complexity of a linear search is O(N) while the time complexity of a binary search is $O(\log_2 N)$. Hence, this is another difference between linear search and binary search. Best Case

Furthermore, the best case in a linear search is to find the element in the first position while the best case in binary search is to find the element in the middle position.

Sorting the Array

In a linear search, it is not required to sort the array before searching the element. However, in a binary search, it is necessary to sort the array before searching the element. Therefore, the prerequisite to sort the array makes a difference between linear search and binary search.

Efficiency

One other difference between linear search and binary search is their efficiency. Binary search is more efficient than linear search.

Simplicity

Besides, binary search is more complex than linear search.

c. Store the given numbers in bucket of size 16, resolve the collisions if any with
   a. Linear probing
   b. Quadratic hashing
   c. Chaining

20, 33, 65, 23, 11, 32, 78, 64, 3, 87, 10, 7

a. Linear probing

0	32	20=1*16+4 1 probe. 33=2*16+1 1 probe. 65=4*16+1 2 probes. 23=1*16+7 1 probe. 11=0*16+11 1 probe. 32=2*16+0 1 probe. 78=4*16+14 1 probe. 64=4*16+0 4 probes. 3=0*16+3 3 probes. 87=5*16+7 2 probes. 10=0*16+10 1 probe. 7=0*16+7 3 probes
1	33	
2	65	
3	64	
4	20	
5	3	
6		
7	23	
8	87	
9	7	
10	10	
11	11	
12		
13		
14	78	
15		

b. Quadratic hashing

0	32	20=1*16+4 4 1 probe. 33=2*16+1 1 1 probe. 65=4*16+1 1 2 2 probes. 23=1*16+7 7 1 probe. 11=0*16+11 11 1 probe. 32=2*16+0 0 1 probe. 78=4*16+14 14 1 probe. 64=4*16+0 0 1 4 9
1	33	
2	65	
3	3	
4	20	
5		
6		
7	23	
8	87	
9	64	

10	10	4 probes. 3=0*16+3 3 1 probe. 87=5*16+7 7 8 2 probes. 10=0*16+10 10 1 probe. 7=0*16+7 7 8 11 0 7 0 11 8 7 8 11 0 7 0 11 8 No spot available.
11	11	
12		
13		
14	78	
15		

c. Chaining

0	32 64	20=1*16+4 33=2*16+1 65=4*16+1 23=1*16+7 11=0*16+11 32=2*16+0 78=4*16+14 64=4*16+0 3=0*16+3 87=5*16+7 10=0*16+10 7=0*16+7
1	33 65	
2		
3	3	
4	20	
5		
6		
7	23 87 7	
8		
9		
10	10	
11	11	
12		
13		
14	78	
15		

Conclusion

Linear search and binary search are two algorithms to search an element in a data structure such as an array. Binary search is efficient and fast than linear search, but it is mandatory to sort the array first before performing the search operation. Thus, the main difference between linear search and binary search is that binary search is more efficient and takes minimum time to search an element, when compared to linear search.