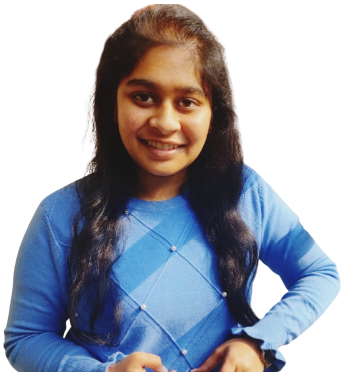
Geeks Man
Algorithms
Lesson 3

Trilok Kaushik

Founder of Geeksman

Aditya Aggarwal



Aditya Gupta

Suhani Mittal

Team Coordinators

Topic :- Searching

Searching means retrieving the information stored in some Data Structures like Array, LinkedList, Trees, Hash Tables etc.

Types of Searching :

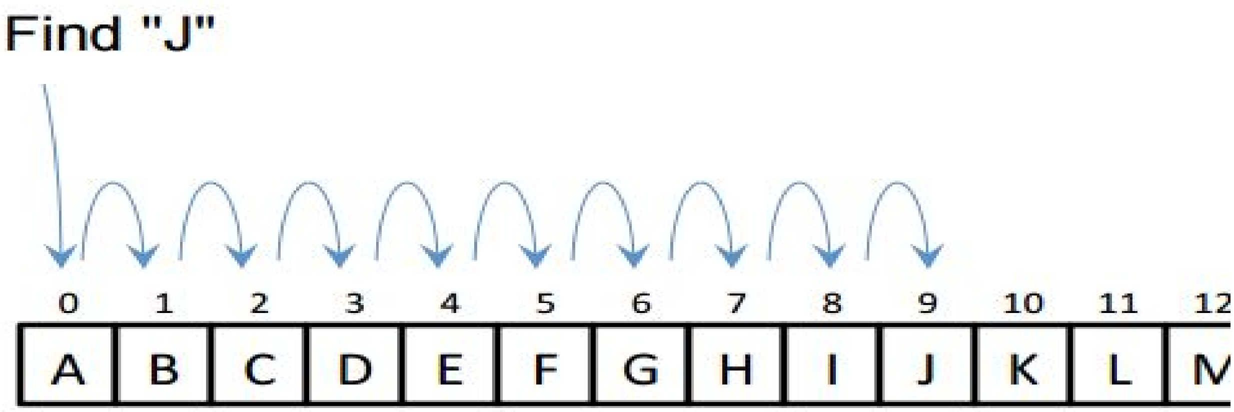
1. *Linear Search*
2. *Binary Search*
3. *Ternary Search*
4. *Jump Search*
5. *Exponential Search etc.*

1. Linear Search

Characteristics :

1. It Sequentially Searches the element.
2. It works on both sorted or unsorted Data.

➔ *Linear Search using Array :*



❑ Iterative Approach :-

Algorithm :

1. Variable Key = User input // storing the element to be searched
2. Set variable i=0 // from where the Searching Starts from
3. If **element at ith position == key element** then **break / return i** ;

   // Search terminates successfully when the element got found
4. i=i+1 // incrementing the variable i
5. If **i>=n** then **return -1** //search terminated unsuccessfully

Code :

```
// arr - user input array
// size - number of elements in array
// key - number to be searched
int linearSearch(int arr[],int size,int key){
    for(int i=0; i<size; i++){
        if(arr[i]==key)
```

```
            return ++i;
    return -1;       // return -1 when element not found in array
}
```

Here is the link for the complete code of linear search https://sapphireengine.com/@/4wrs9p

❑ Recursive Approach :

Algorithm :

1. Creating an array of n-size elements and storing values in it.
2. Storing the search element in a key variable.
3. Calling the linear_search(arr,n-1,key)
4. linear_search(array, n , target) : if element found at nth position then return 'n';
5. Else return linear_search function for n-1 elements.

Code:

```
// arr - input array
// size- size of array
// key - number to be searched
int linearSearch(int arr[],int size,int key){
    if(size<0)        //base case
        return -1;

//checking whether last element equal to key or not
```

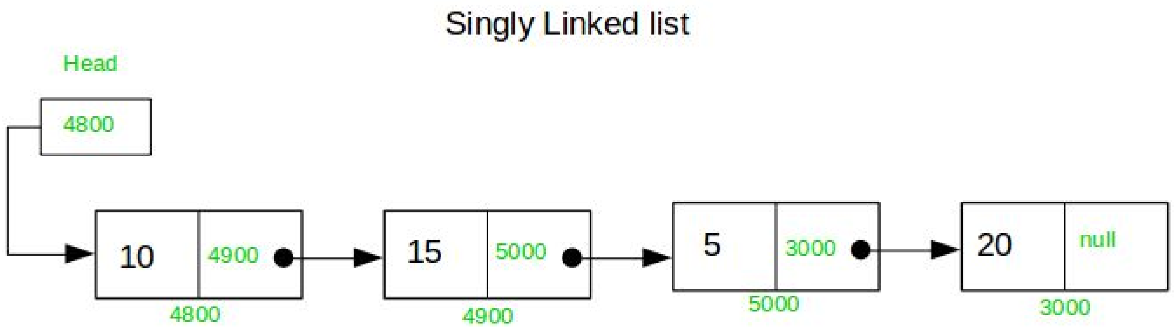
```
  if(arr[size]==key)
      return ++size;

  //calling recursion
  return linearSearch(arr,size-1,key);
}
```

Here is the link for the complete code of linear search using recursion

https://sapphireengine.com/@/1jx7ki

➔ *Linear Search using LinkedList :*



❑ Iterative Approach :

Algorithm :

1. Check if the head is null or not // null means the linked list is empty.
2. Store head address in variable temp;

3. If temp->data == target element then return temp;
4. Else point temp to the next address of the node.

Code :

```
Following is the class structure of the Node class:

       class Node
       {
       public:
            int data;
            Node *next;
            Node(int data)
            {
                 this->data = data;
                 this->next = NULL;
            }
         };
// head - stores the address of first node
// n - data to be searched in linked list
int LinearSearch(Node *head, int key)
{
   if(head==NULL)return false;  //list is empty
   Node *temp=head;      //declaring temporary node
   int j=0;
   while(temp!=NULL){
       j++;
       if(temp->data==key){
           return j;//element found so return j
       }
       temp=temp->next;   //incrementing temp
   }
   return -1;         //n not found in linked list
}
```

Here is the link for the complete code of linear search for searching element in linked list(iterative)

https://sapphireengine.com/@/9k389x

❑ Recursive Approach :

Algorithm :

1. Calling linear_Search (temp): if temp->Data == key element then return head_virtual.
2. Else return linear_search (temp -> next)

Code :

```
// head - stores the address of first node
// key - data to be searched in linked list
bool search(Node *head, int key)
{
    if(head==NULL){
       return false;
   }
   if(head->data==key)
       return true;
   return search(head->next,key);
}
```

Here is the link for the complete code of linear search for searching element in linked list(recursive)

https://sapphireengine.com/@/r2yr2e

Questions

Ques1 :- You are given a list of 5 integers and these integers are in the range from 1 to 6. There are no duplicates in list. One of the integers is missing in the list. Which of the following expression would give the missing number.

^ is bitwise XOR operator.

~ is bitwise NOT operator.

Let elements of list can be accessed as list[0], list[1], list[2], list[3], list[4]

(A) list[0] ^ list[1] ^ list[2] ^ list[3] ^ list[4]

(B) list[0] ^ list[1] ^ list[2] ^ list[3] ^ list[4] ^ 1 ^ 2 ^ 3 ^ 4 ^ 5 ^ 6

(C) list[0] ^ list[1] ^ list[2] ^ list[3] ^ list[4] ^ 1 ^ 2 ^ 3 ^ 4 ^ 5

(D) ~(list[0] ^ list[1] ^ list[2] ^ list[3] ^ list[4])

Ques2 :- The average number of key comparisons done in a successful sequential search in a list of length it is
(A) log n
(B) (n-1)/2
(C) n/2
(D) (n+1)/2

Ques3 :- The average case occurs in the Linear Search Algorithm when:
(A) The item to be searched is in some where middle of the Array
(B) The item to be searched is not in the array
(C) The item to be searched is in the last of the array
(D) The item to be searched is either in the last or not in the array

Ques4 :-Number of comparisons required for an unsuccessful search of an element in a sequential search, organized, fixed length, symbol table of length L is

(A) L

(B) L/2

(C) (L+1)/2

(D) 2L

Ques5 :-

```
#include <stdio.h>

void print(int n, int j)
{
   if (j >= n)
      return;
   if ( n-j > 0 && n-j >= j)
       printf( "%d %d\n", j, n-j);
   print(n, j+1);
}

int main()
{
    int n = 8;
    print(n, 1);
}
```

(A) 1 7

2 6

3 5

4 4

4 4

(B) 1 7

2 6

3 5

4 4

(C) 1 7

2 6

3 5

(D) 1 2

3 4

5 6

7 8