Google Developer Student Club
Meerut Institute of Engineering and Technology

DSA BOOTCAMP

SESSION 3

18FEB2023

Google Developer Student Club
Meerut Institute of Engineering and Technology

TODAY'S TOPIC:

SEARCHING

- 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.

TYPES OF SEARCHING

01 LINEAR SEARCH

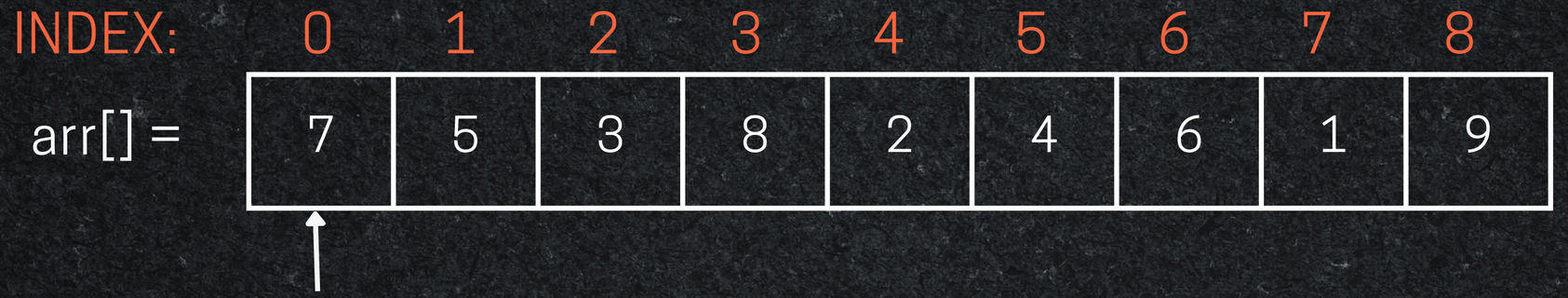
02 BINARY SEARCH

LINEAR SEARCH

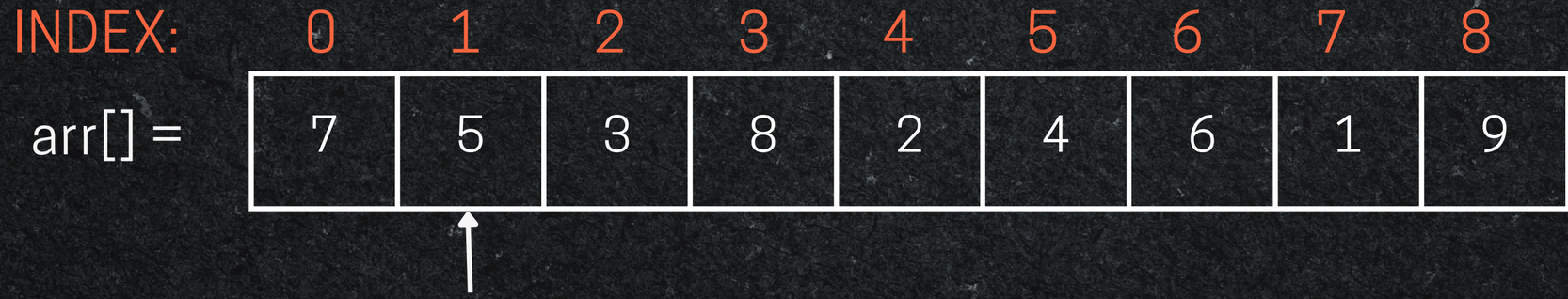
(SEQUENTIAL SEARCH)

This search method searches for an element by visiting all the elements sequentially until the element is found or the array finishes.

Q. Search for element 2 in the given array

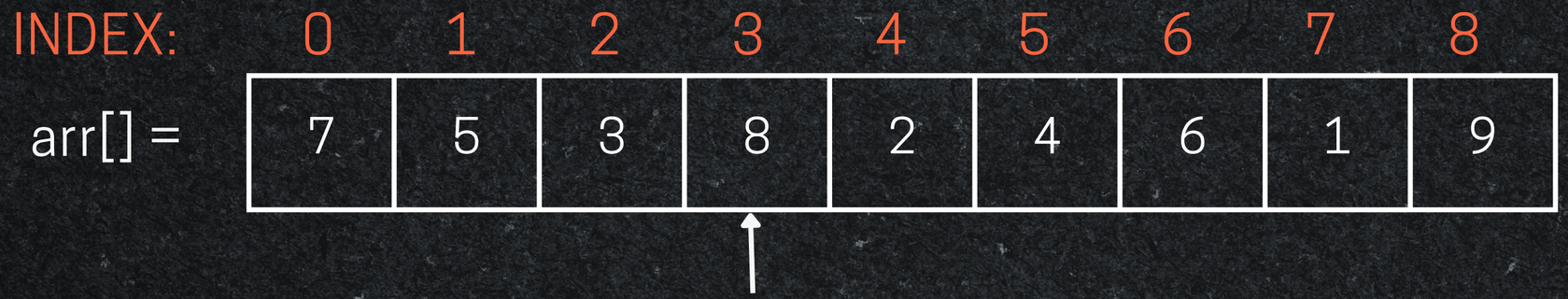


ITERATION 1

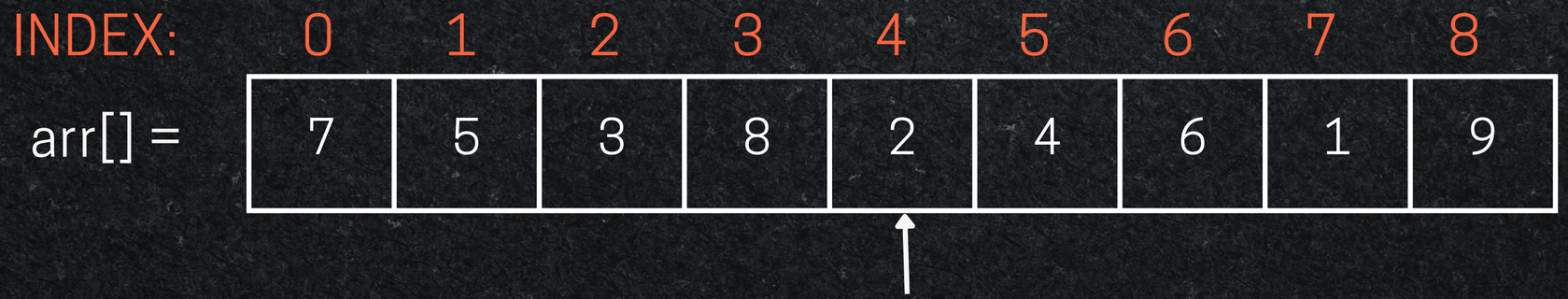


ITERATION 2

ITERATION 3



ITERATION 4



ITERATION 5

CODE FOR LINEAR SEARCH

```
int linearSearch(int arr[],  int element){

    //loop to traverse the whole array
   for (int i = 0; i < arr.size(); i++)
     //compare each element with the element to be searched
   {   if(arr[i]==element){
     //returns the index i where the element is found
         return i;

     }
   }

   return -1;
}
```

BINARY SEARCH

(INTERVAL SEARCH)

- This search method searches for an element by breaking the search space into half each time it finds the wrong element.
- This method is limited to a sorted array.
- The search continues towards either side of the mid, based on whether the element to be searched is lesser or greater than the mid element of the current search space.

```
int binarySearch(int arr[], int size, int element){
    int low, mid, high;
    low = 0;
    high = size-1;
    // Keep searching until low <= high

    while(low<=high){
        mid = (low + high)/2;
        if(arr[mid] == element){ //CASE1
            return mid;
        }
        if(arr[mid]<element){ //CASE2
            low = mid+1;
        }
        else{ //CASE 3
            high = mid -1;
        }
    }
    return -1;
}
```

CODE FOR BINARY SEARCH

Q. Search for element 3 in the given array

mid = (low + high)/2
=(0 + 8)/2
=4

CASE 3: arr[mid] > element

high = mid-1

mid = (low + high)/2
=(0 + 3)/2
=1

CASE 2: arr[mid] < element

low = mid+1

mid = (low + high)/2
=(2 + 3)/2
=2

CASE 1: arr[mid] == element

so, element 3 is present at index mid(i.e. 2)

LINEAR SEARCH

- sequential approach.
- Works on both sorted and unsorted arrays
- preferrable for the small-sized data sets.
- O(n) Complexity

BINARY SEARCH

- divide and conquer approach.
- Works only on sorted arrays
- preferrable for the large-size data sets.
- O(log n) Complexity

Google Developer Student Club
Meerut Institute of Engineering and Technology

Thank You

@cpTeamGDSC_MIET