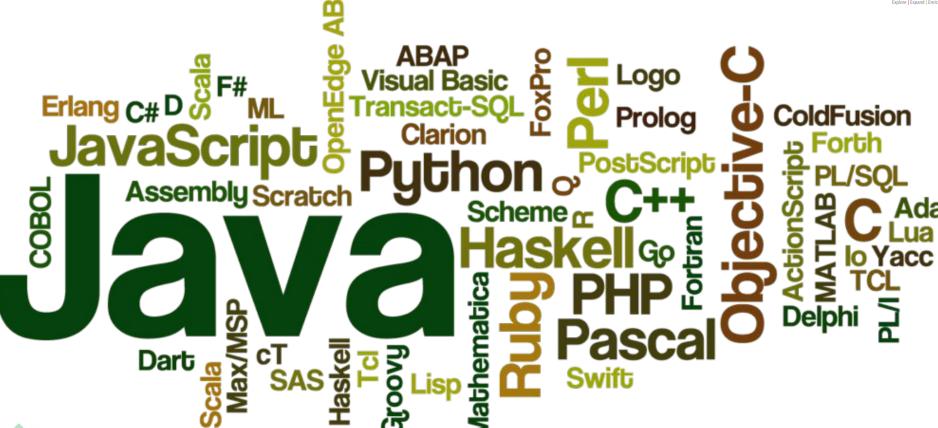


Explore | Expand | Enrich







SEARCHING TECHNIQUES IN JAVA



SEARCHING TECHNIQUES



- Searching is an operation or a technique that helps finds the place of a given element or value in the list
- Any search is said to be successful or unsuccessful depending upon whether the element that is being searched is found or not
- Some of the standard searching technique that is being followed in the data structure is listed below
- Linear Search or Sequential Search
- Binary Search

LINEAR SEARCH



- Match the key element with array element
- If key element is found, return the index position of the array element
- If key element is not found, return -1

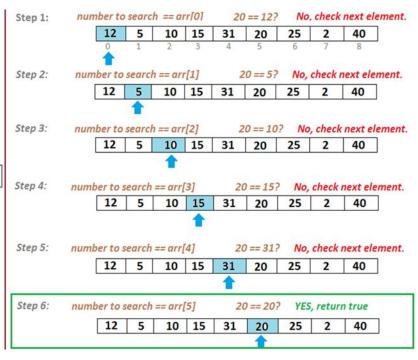


LINEAR SEARCH IMPLEMENTATION



Search 20

12	5	10	15	31	20	25	2	40
0	1	2	3	4	5	6	7	8





LOGIC



```
class Main
                                              public static void main(String args[])
public static int search(int arr[], int x)
                                                  int arr[] = \{ 2, 3, 4, 10, 40 \};
                                                  int x = 10;
    int n = arr.length;
    for(int i = 0; i < n; i++)
                                                  int result = search(arr, x);
                                                  if(result == -1)
        if(arr[i] == x)
                                                      System.out.print("Element is not
            return i;
                                              present in array");
                                                  else
                                                      System.out.print("Element is
    return -1;
                                              present at index " + result);
```



BINARY SEARCH



- Binary Search: binary search or half-interval search algorithm finds the position of a specified value (the input "key") within a sorted array
- In each step, the algorithm compares the input key value with the key value of the middle element of the array
- If the keys match, then a matching element has been found so its index, or position, is returned



IMPLEMENTATIONS



Example:

Search 20

if 20 (number to search) == 15? if yes, We found the element.

(False)

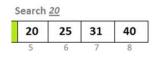
If 20 (number to search) > 15, It means, 20 would be on Right sub array --->

(TRUE, in our case)

40

If 20 (number to search) < 15, It means, 20 would be on Left sub array <---(False)

This time our searching reduce to half as we don't need to look on left sub array before 15 as it contain elements lower than 15. STEP 2:



Middle =
$$low+(high-low)/2 = 5+(8-5)/2 = 5+3/2 = 5+1 = 6$$

40

if 20(number to search) == 25? if yes, We found the element.

(False) (False)

If 20(number to search) > 25, It means, 20 would be on Right sub array ---> If 20(number to search) < 25, It means, 20 would be on Left sub array <---(TRUE, in our case)

This time our searching reduce to half as we don't need to look on right sub array after 25 as it contain elements greater than 25. STEP 3:

Search 20

Middle =
$$low+(high-low)/2 = 5 + (5-5)/2 = 5+0 = 5$$

If 20 (number to search) == 20? YES, We found the element. (TRUE, in our case) STOP Searching, return True

LOGIC



```
class Main{
 public static void binarySearch(int arr[], int first, int last, int
key){
   int mid = (first + last)/2;
   while( first <= last ){</pre>
      if ( arr[mid] < key ){</pre>
        first = mid + 1;
      }else if ( arr[mid] == key ){
        System.out.println("Element is found at index: " + mid);
        break;
      }else{
         last = mid - 1;
      mid = (first + last)/2;
   if ( first > last ){
      System.out.println("Element is not found!");
```

