



Explore | Expand | Enrich



# SEARCHING TECHNIQUES IN JAVA



- Searching is an operation or a technique that helps find 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 techniques that are being followed in the data structure are 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

Step 1: *number to search == arr[0]      20 == 12?      No, check next element.*

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



Step 2: *number to search == arr[1]      20 == 5?      No, check next element.*

12	5	10	15	31	20	25	2	40
----	---	----	----	----	----	----	---	----



Step 3: *number to search == arr[2]      20 == 10?      No, check next element.*

12	5	10	15	31	20	25	2	40
----	---	----	----	----	----	----	---	----



Step 4: *number to search == arr[3]      20 == 15?      No, check next element.*

12	5	10	15	31	20	25	2	40
----	---	----	----	----	----	----	---	----



Step 5: *number to search == arr[4]      20 == 31?      No, check next element.*

12	5	10	15	31	20	25	2	40
----	---	----	----	----	----	----	---	----



Step 6: *number to search == arr[5]      20 == 20?      YES, return true*

12	5	10	15	31	20	25	2	40
----	---	----	----	----	----	----	---	----



```
class Main
{
    public static int search(int arr[], int x)
    {
        int n = arr.length;
        for(int i = 0; i < n; i++)
        {
            if(arr[i] == x)
                return i;
        }
        return -1;
    }
}
```

```
public static void main(String args[])
{
    int arr[] = { 2, 3, 4, 10, 40 };
    int x = 10;

    int result = search(arr, x);
    if(result == -1)
        System.out.print("Element is not
present in array");
    else
        System.out.print("Element is
present at index " + result);
}
}
```

- 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

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

**STEP 1:** Middle =  $\text{low} + (\text{high} - \text{low}) / 2 = 0 + (8 - 0) / 2 = 4$

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

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)

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

(False)

**STEP 2:** 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.

Search 20

20	25	31	40
5	6	7	8

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

20	25	31	40
5	6	7	8

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

(False)

If 20 (number to search) > 25, It means, 20 would be on Right sub array --->

(False)

If 20 (number to search) < 25, It means, 20 would be on Left sub array <---

(TRUE, in our case)

**STEP 3:** 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.

Search 20

20
5

Middle =  $\text{low} + (\text{high} - \text{low}) / 2 = 5 + (5 - 5) / 2 = 5 + 0 = 5$

20
5

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

```
class Main{
    public static void binarySearch(int arr[], int first, int last, int
key){
        int mid = (first + last)/2;
        while( first <= last ){
            if ( arr[mid] < key ){
                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!");
        }
    }
}
```

THANK YOU

