

Binary Search - Theory + Implementation

17 August 2021 08:10 AM

Binary Search - We take a sorted array and check the middle element. If element to be found is the middle element then return it else if it is less than middle then start will be same but end will be middle-1. Similarly if element to be searched is greater than mid then start will be mid+1 and end will be same.

Complexity - $O(\log_2 N)$

NOTE : $\text{mid} = (\text{start} + \text{end}) / 2$

If the values of start and end are very large, then mid will exceed the limit of int.

Better way : $\text{mid} = \text{start} + (\text{end} - \text{start}) / 2$

```
class BinarySearch {
    // Returns index of x if it is present in arr[l..
    // r], else return -1
    int binarySearch(int arr[], int l, int r, int x)
    {
        if (r >= l) {
            int mid = l + (r - l) / 2;

            // If the element is present at the
            // middle itself
            if (arr[mid] == x)
                return mid;

            // If element is smaller than mid, then
            // it can only be present in left subarray
            if (arr[mid] > x)
                return binarySearch(arr, l, mid - 1, x);

            // Else the element can only be present
            // in right subarray
            return binarySearch(arr, mid + 1, r, x);
        }
    }
}
```

```

        // We reach here when element is not present
        // in array
        return -1;
    }

    // Driver method to test above
    public static void main(String args[])
    {
        BinarySearch ob = new BinarySearch();
        int arr[] = { 2, 3, 4, 10, 40 };
        int n = arr.length;
        int x = 10;
        int result = ob.binarySearch(arr, 0, n - 1, x);
        if (result == -1)
            System.out.println("Element not present");
        else
            System.out.println("Element found at index " +
result);
    }
}

```

Order Agnostic Binary Search :

When we don't know how the array is sorted i.e. ascending or descending.

So in this, we need to make a check of first and last element.

If they are same then the array contains all elements equal.

```

// find whether the array is sorted in ascending or descending
boolean isAsc = arr[start] < arr[end];

```

```


if (arr[mid] == target) {
    return mid;
}

```

```

if (isAsc) {
    if (target < arr[mid]) {

```



```
if (isAsc) {  
    if (target < arr[mid]) {  
        end = mid - 1;  
    } else {  
        start = mid + 1;  
    }  
} else {  
    if (target > arr[mid]) {  
        end = mid - 1;  
    } else {  
        start = mid + 1;  
    }  
}
```