

**“AZƏRBAYCAN HAVA YOLLARI” CJSC NATIONAL AVIATION ACADEMY”**

**Individual Work №: 5**

**Topic: Logarithmic time complexity. Binary search.**

**Subject: OOP**

**Teacher: Mamed Shahmaliyev**

**Group: 1459i Student:IlahaAliyeva**

**Baku 2022**

***Input:****arr[] = {10, 20, 30, 50, 60, 80, 110, 130, 140, 170}, x = 110****Output:****6****Explanation:****Element x is present at index 6*

***Input:****arr[] = {10, 20, 30, 40, 60, 110, 120, 130, 170}, x = 175****Output:****-1****Explanation:****Element x is not present in arr[].*

**Linear Search Approach**: A simple approach is to do a [**linear search**](https://www.geeksforgeeks.org/linear-search/)**.** The time complexity of the Linear search is O(n). Another approach to perform the same task is using *Binary Search*.

**Binary Search Approach:**

***Binary Search****is a*[*searching algorithm*](https://www.geeksforgeeks.org/searching-algorithms/)*used in a sorted array by****repeatedly dividing the search interval in half****. The idea of binary search is to use the information that the array is sorted and reduce the time complexity to O(Log n).*

**Binary Search Algorithm:** The basic steps to perform Binary Search are:

* Begin with an interval covering the whole array.
* If the value of the search key is less than the item in the middle of the interval, narrow the interval to the lower half.
* Otherwise, narrow it to the upper half.
* Repeatedly check until the value is found or the interval is empty.

[Recommended: Please solve it on “***PRACTICE***” first, before moving on to the solution.](https://practice.geeksforgeeks.org/problems/binary-search/1)

**Step-by-step Binary Search Algorithm:** We basically ignore half of the elements just after one comparison.

1. Compare x with the middle element.
2. If x matches with the middle element, we return the mid index.
3. Else If x is greater than the mid element, then x can only lie in the right half subarray after the mid element. So we recur for the right half.
4. Else (x is smaller) recur for the left half.

# Python3 Program for recursive binary search.

# Returns index of x in arr if present, else -1

def binarySearch(arr, l, r, x):

    # Check base case

    if r >= l:

        mid = l + (r - l) // 2

        # If 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

        elif arr[mid] > x:

            return binarySearch(arr, l, mid-1, x)

        # Else the element can only be present

        # in right subarray

        else:

            return binarySearch(arr, mid + 1, r, x)

    else:

        # Element is not present in the array

        return -1

# Driver Code

arr = [2, 3, 4, 10, 40]

x = 10

# Function call

result = binarySearch(arr, 0, len(arr)-1, x)

if result != -1:

    print("Element is present at index % d" % result)

else:

    print("Element is not present in array")

**Output**

Element is present at index 3

# Python3 code to implement iterative Binary

# Search.

# It returns location of x in given array arr

# if present, else returns -1

def binarySearch(arr, l, r, x):

    while l <= r:

        mid = l + (r - l) // 2

        # Check if x is present at mid

        if arr[mid] == x:

            return mid

        # If x is greater, ignore left half

        elif arr[mid] < x:

            l = mid + 1

        # If x is smaller, ignore right half

        else:

            r = mid - 1

    # If we reach here, then the element

    # was not present

    return -1

# Driver Code

arr = [2, 3, 4, 10, 40]

x = 10

# Function call

result = binarySearch(arr, 0, len(arr)-1, x)

if result != -1:

    print("Element is present at index % d" % result)

else:

    print("Element is not present in array")

**Output**

Element is present at index 3