Binary Search Algorithm

What is Binary Search Algorithm?

Binary search is a search algorithm used to find the position of a target value within a **sorted** array. It works by repeatedly dividing the search interval in half until the target value is found or the interval is empty. The search interval is halved by comparing the target element with the middle value of the search space.

Conditions to apply Binary Search Algorithm in a Data Structure:

To apply Binary Search algorithm:

- The data structure must be sorted.
- Access to any element of the data structure should take constant time.

Binary Search Algorithm:

Below is the step-by-step algorithm for Binary Search:

- Divide the search space into two halves by **finding the middle index "mid"**.
- Compare the middle element of the search space with the **key**.
- If the **key** is found at middle element, the process is terminated.
- If the **key** is not found at middle element, choose which half will be used as the next search space.
  - If the **key** is smaller than the middle element, then the **left** side is used for next search.
  - If the **key** is larger than the middle element, then the **right** side is used for next search.
- This process is continued until the **key** is found or the total search space is exhausted.

How does Binary Search Algorithm work?

To understand the working of binary search, consider the following illustration:

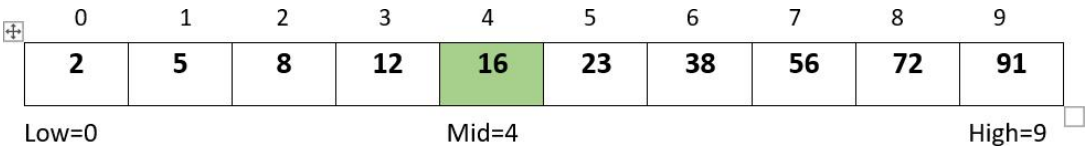
Consider an array **arr[] = {2, 5, 8, 12, 16, 23, 38, 56, 72, 91}**, and the **target = 23.**

1. Since arr[4] == 16, the target is found at index 4.

2. *Calculate the **mid** and compare the mid element with the key. In the example the* **Initial range:** low = 0, high = 9, mid = 5, check arr[5] = 23.

- *If the key is less than **mid** element, move to **left** and if it is greater than the **mid** then move search space to the **right**.*

- *Key (i.e., 23) is greater than current **mid** element (i.e., 16). The search space moves to the **right**.*



How to Implement Binary Search Algorithm?

The **Binary Search Algorithm** can be implemented in the following two ways

- Iterative Binary Search Algorithm
- Recursive Binary Search Algorithm

Iterative Binary Search Algorithm:

Here we use a while loop to continue the process of comparing the key and splitting the search space in two halves.

```
using System;

class BSIterative {

    // Returns index of x if it is present in arr[]
    static int binarySearch(int[] arr, int x)
    {
        int low = 0, high = arr.Length - 1;
        while (low <= high) {
            int mid = low + (high - low) / 2;

            // Check if x is present at mid
            if (arr[mid] == x)
                return mid;

            // If x greater, ignore left half
            if (arr[mid] < x)
                low = mid + 1;

            // If x is smaller, ignore right half
            else
                high = mid - 1;
        }

        // If we reach here, then element was
        // not present
        return -1;
    }

    // Driver code
    public static void Main()
    {
        int[] arr = { 2, 3, 4, 10, 40 };
        int n = arr.Length;
        int x = 10;
        int result = binarySearch(arr, x);
        if (result == -1)
            Console.WriteLine(
                "Element is not present in array");
        else
```

```
Element is present at index 3
```

```
    }
}
```

Recursive Binary Search

```
// C# implementation of recursive Binary Search
using System;

class BSRecursive {

    // Returns index of x if it is present in
    // arr[low..high], else return -1
    static int binarySearch(int[] arr, int low, int high, int x)
    {
        if (high >= low) {
            int mid = low + (high - low) / 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, low, mid - 1, x);

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

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

    // Driver code
    public static void Main()
    {

        int[] arr = { 2, 3, 4, 10, 40 };
        int n = arr.Length;
        int x = 10;

        int result = binarySearch(arr, 0, n - 1, x);

        if (result == -1)
            Console.WriteLine(
                "Element is not present in arrau");
        else
            Console.WriteLine("Element is present at index "
                              + result);
        Console.ReadKey();
    }
}
```

```
Element is present at index 3
```