Binary Search

In this tutorial, you will learn how Binary Search sort works. Also, you will find working examples of Binary Search in C, C++, Java and Python.

Binary Search is a searching algorithm for finding an element's position in a sorted array.

In this approach, the element is always searched in the middle of a portion of an array.

> Binary search can be implemented only on a sorted list of items. If the elements are not sorted already, we need to sort them first.

Binary Search Working

Binary Search Algorithm can be implemented in two ways which are discussed below.

1. Iterative Method
2. Recursive Method

The recursive method follows the divide and conquer (/dsa/divide-and-conquer) approach.

Let `x = 4` be the element to be searched.

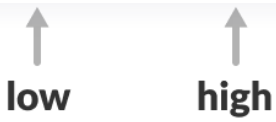
2. Set two pointers low and high at the lowest and the highest positions respectively.

Setting pointers

3. Find the middle element `mid` of the array ie. `arr[(low + high)/2] = 6`.

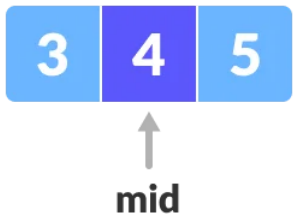
Mid element

4. If x == mid, then return mid.Else, compare the element to be searched with m.



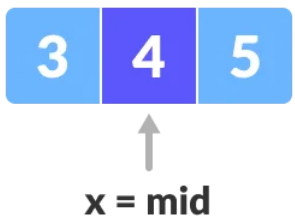
Finding mid element

7. Repeat steps 3 to 6 until low meets high.



Mid element

8. `x = 4` is found.



Found

```
        high = mid - 1
```

Recursive Method

```
binarySearch(arr, x, low, high)
    if low > high
        return False
    else
        mid = (low + high) / 2
        if x == arr[mid]
            return mid
        else if x > arr[mid]         // x is on the right side
            return binarySearch(arr, x, mid + 1, high)
        else                                 // x is on the left side
            return binarySearch(arr, x, low, mid - 1)
```

Python, Java, C/C++ Examples (Iterative Method)

Python Java C C++

```
        if array[mid] == x:
            return mid

        elif array[mid] < x:
            low = mid + 1

        else:
            high = mid - 1

    return -1


array = [3, 4, 5, 6, 7, 8, 9]
x = 4

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

if result != -1:
```

Python, Java, C/C++ Examples (Recursive Method)

Python Java C C++

```
        if array[mid] == x:
            return mid

        # Search the left half
        elif array[mid] > x:
            return binarySearch(array, x, low, mid-1)

        # Search the right half
        else:
            return binarySearch(array, x, mid + 1, high)

    else:
        return -1


array = [3, 4, 5, 6, 7, 8, 9]
x = 4
```

Binary Search Complexity

Time Complexities

Binary Search Applications

- In libraries of Java, .Net, C++ STL
- While debugging, the binary search is used to pinpoint the place where the error happens.

Next Tutorial:
Greedy Algorithm (/dsa/greedy-algorithm)

Previous Tutorial:
Linear Search (/dsa/linear-search)

Related Tutorials

DS & Algorithms

Linear Search

(/dsa/linear-search)

DS & Algorithms

Quicksort Algorithm

(/dsa/quick-sort)

DS & Algorithms

Insertion Sort Algorithm

(/dsa/insertion-sort)

DS & Algorithms

Hashing

(/dsa/hashing)