Binary Search

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.
```

How Binary Search Works?

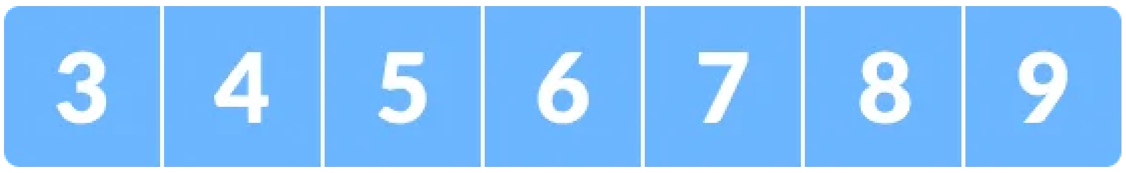
Binary Search Algorithm can be implemented in two ways which are discussed below.

1. Iterative Method
2. Recursive Method

The recursive method follows the divide and conquer approach.

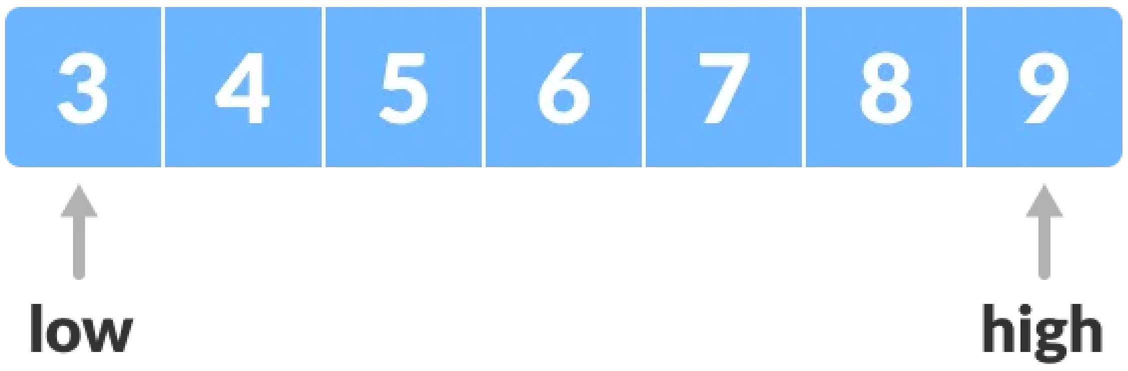
The general steps for both methods are discussed below.

1. The array in which searching is to be performed is:

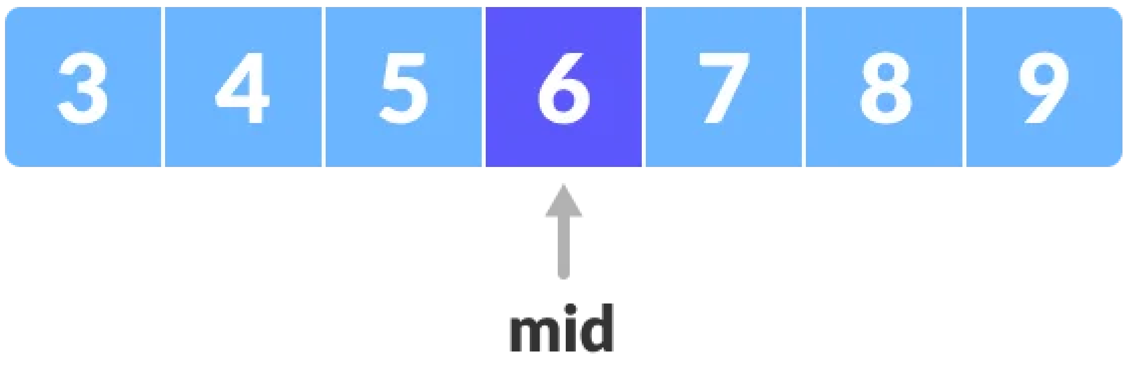


```
Let x = 4 be the element to be searched.
```

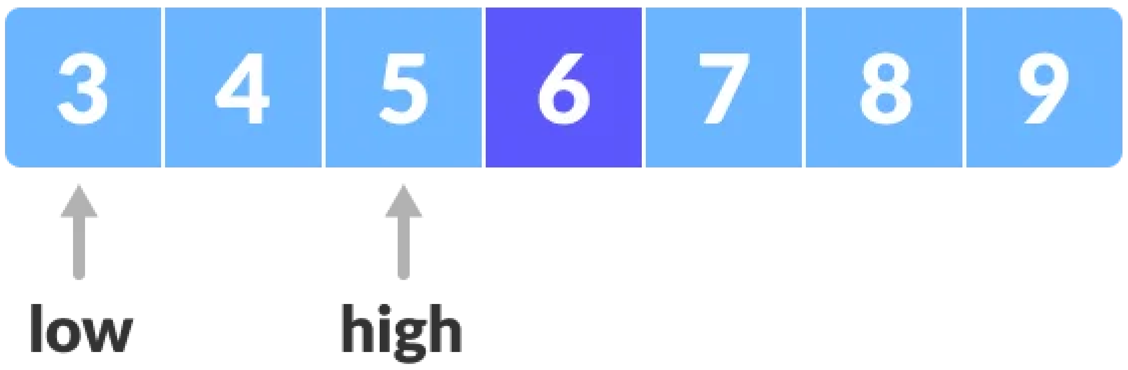
2. Set two pointers low and high at the lowest and the highest positions respectively.



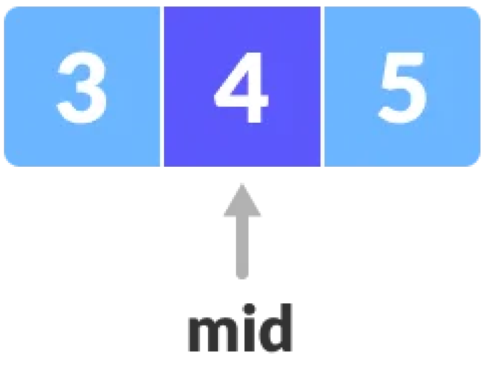
3. Find the middle element mid of the array ie. arr[(low + high)/2] = 6.



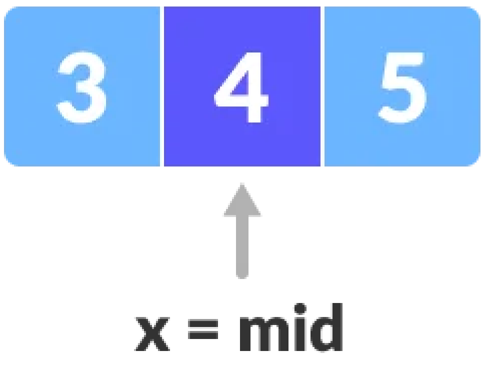
4. If x == mid, then return mid.Else, compare the element to be searched with m.
5. If x > mid, compare x with the middle element of the elements on the right side of mid. This is done by setting low to low = mid + 1.
6. Else, compare x with the middle element of the elements on the left side of mid. This is done by setting high to high = mid - 1.



7. Repeat steps 3 to 6 until low meets high



8. x = 4 is found.



Binary Search Algorithm

Iteration Method

```
do until the pointers low and high meet each other.
    mid = (low + high)/2
    if (x == arr[mid])
        return mid
    else if (x > arr[mid]) // x is on the right side
        low = mid + 1
    else                       // x is on the left side
        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 right side
            return binarySearch(arr, x, low, mid - 1)
```

Java Syntax for Iterative Binary Search

```
// Binary Search in Java

class BinarySearch {
  int binarySearch(int array[], int x, int low, int high) {

    // Repeat until the pointers low and high meet each other
    while (low <= high) {
      int mid = low + (high - low) / 2;

      if (array[mid] == x)
        return mid;

      if (array[mid] < x)
        low = mid + 1;

      else
        high = mid - 1;
    }

    return -1;
  }

  public static void main(String args[]) {
    BinarySearch ob = new BinarySearch();
    int array[] = { 3, 4, 5, 6, 7, 8, 9 };
    int n = array.length;
    int x = 4;
    int result = ob.binarySearch(array, x, 0, n - 1);
    if (result == -1)
      System.out.println("Not found");
    else
      System.out.println("Element found at index " + result);
  }
}
```

Java Syntax for Recursive Binary Search

```
// Binary Search in Java

class BinarySearch {
  int binarySearch(int array[], int x, int low, int high) {

    if (high >= low) {
      int mid = low + (high - low) / 2;

      // If found at mid, then return it
      if (array[mid] == x)
        return mid;

      // Search the left half
      if (array[mid] > x)
        return binarySearch(array, x, low, mid - 1);

      // Search the right half
      return binarySearch(array, x, mid + 1, high);
    }

    return -1;
  }

  public static void main(String args[]) {
    BinarySearch ob = new BinarySearch();
    int array[] = { 3, 4, 5, 6, 7, 8, 9 };
    int n = array.length;
    int x = 4;
    int result = ob.binarySearch(array, x, 0, n - 1);
    if (result == -1)
      System.out.println("Not found");
    else
      System.out.println("Element found at index " + result);
  }
}
```

Linear Search Complexities

Time Complexities

Best case complexity: O(1) Average case complexity: O(log n) Worst case complexity: O(log n)

Space Complexity

The space complexity of the binary search is O(1).

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..