**Binary search in Java**

**Binary search** is a fast algorithm for finding the position of an element in a **sorted array**. For an array of size n*n*, the running time of the algorithm is O(\log n)*O*(log*n*) in the worst case.

The algorithm begins by comparing the middle element of the array with a target value. If the target value matches the middle element, its position in the array is returned. If the target value is less than or greater than the middle element, the search continues in the left or right subarray, respectively, eliminating the other subarray from consideration. It repeats until the value is found or a new search interval is empty.

## The iterative implementation

The iterative implementation uses a loop for iterating over the passed array. If the considered interval is empty the loop stops and the method returns -1 that indicates the element is not found.

public static int binarySearch(int[] array, int elem, int left, int right) {  
    while (left <= right) {  
        int mid = left + (right - left) / 2; // the index of the middle element  
              
        if (elem == array[mid]) {  
            return mid; // the element is found, return its index  
        } else if (elem < array[mid]) {  
            right = mid - 1; // go to the left subarray  
        } else {  
            left = mid + 1;  // go the the right subarray  
        }  
    }  
    return -1; // the element is not found  
}

The method takes an array of int's, a target element and two boundaries of the subarray where we search the element. The last two parameters are not mandatory but they are useful if we also want to be able to search not in the entire array.

## The recursive implementation

The recursive implementation makes a recursive call instead of using a loop. It doesn't throw the StackOverflowError because it makes not many recursive calls even for large arrays.

public static int binarySearch(int[] array, int elem, int left, int right) {  
    if (left > right) {  
        return -1; // search interval is empty, the element is not found  
    }  
          
    int mid = left + (right - left) / 2; // the index of the middle element  
          
    if (elem == array[mid]) {  
        return mid; // the element is found, return its index  
    } else if (elem < array[mid]) {  
        return binarySearch(array, elem, left, mid - 1); // go to the left subarray  
    } else {  
        return binarySearch(array, elem, mid + 1, right); // go the the right subarray      
    }  
}

Make sure that the method returns the same results as the previous one.

In fact, iterative and recursive implementations are equivalent. Use any of them for educational purposes. But remember, the binary search is implemented in the Java standard library, see [java.util.Arrays.binarySearch(...)](https://docs.oracle.com/javase/8/docs/api/java/util/Arrays.html" \t "_blank) for details. It works for different data types including integer numbers, characters, strings and so on.