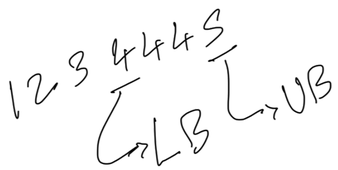
The upper bound algorithm finds the first or the smallest index in a sorted array where the value at that index is greater than the given key i.e. x.
The upper bound is the smallest index, ind, where arr[ind] > x.

But if any such index is not found, the upper bound algorithm returns n i.e. size of the given array. The main difference between the lower and upper bound is in the condition.

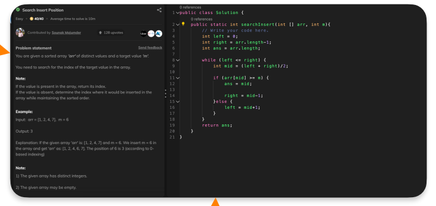
For the lower bound the condition was arr[ind] >= x and here, in the case of the upper bound, it is arr[ind] > x.

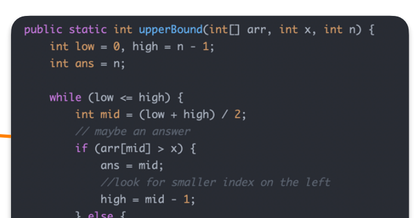
```java
public static int upperBound(int[] arr, int x, int n) {
    int low = 0, high = n - 1;
    int ans = n;

    while (low <= high) {
        int mid = (low + high) / 2;
        // maybe an answer
        if (arr[mid] > x) {
            ans = mid;
            //look for smaller index on the left
            high = mid - 1;
        } else {
            low = mid + 1; // look on the right
        }
    }
    return ans;
}
```



BS on Arrays

Striver





Binary Search

```java
public static int lowerBound(int []arr, int n, int x) {
    int low = 0, high = n - 1;
    int ans = n;

    while (low <= high) {
        int mid = (low + high) / 2;
        // maybe an answer
        if (arr[mid] >= x) {
            ans = mid;
            //look for smaller index on the left
            high = mid - 1;
        } else {
            low = mid + 1; // look on the right
        }
    }
    return ans;
}
```

The lower bound algorithm finds the first or the smallest index in a sorted array where the value at that index is greater than or equal to a given key i.e. x.

```java
static int findFloor(int[] arr, int n, int x) {
    int low = 0, high = n - 1;
    int ans = -1;

    while (low <= high) {
        int mid = (low + high) / 2;
        // maybe an answer
        if (arr[mid] <= x) {
            ans = arr[mid];
            //look for smaller index on the left
            low = mid + 1;
        } else {
            high = mid - 1; // look on the right
        }
    }
    return ans;
}

static int findCeil(int[] arr, int n, int x) {
    int low = 0, high = n - 1;
    int ans = -1;

    while (low <= high) {
        int mid = (low + high) / 2;
        // maybe an answer
        if (arr[mid] >= x) {
            ans = arr[mid];
            //look for smaller index on the left
            high = mid - 1;
        } else {
            low = mid + 1; // look on the right
        }
    }
    return ans;
}
```

Problem Statement: You're given an sorted array arr of n integers and an integer x. Find the floor and ceiling of x in arr[0..n-1].

The floor of x is the largest element in the array which is smaller than or equal to x.
The ceiling of x is the smallest element in the array greater than or equal to x.

Example 1:
Input Format: n = 6, arr[] ={3, 4, 4, 7, 8, 10}, x= 5
Result: 4 7
Explanation: The floor of 5 in the array is 4, and the ceiling of 5 in the array is 7.

Example 2:
Input Format: n = 6, arr[] ={3, 4, 4, 7, 8, 10}, x= 8
Result: 8 8
Explanation: The floor of 8 in the array is 8, and the ceiling of 8 in the array is also 8.