1. 문제 요약

1. 이진 탐색 구현
2. Upper Bound
3. Lower Bound

2. 접근 아이디어

1. 값이 정렬되어있는 상황
2. 값끼리 대소 비교가 가능하면
3. 값을 찾을 때마다 필요 없는 부분을 반절씩 걷어낼 수 있다.

3. 시간 복잡도

O(logN)

4. 구현 코드

```
import java.util.Arrays;

public class BinarySearch {
    private static int[] arr;
    public static void main(String[] args) {
        arr = new int[] {1, 5, 5, 5, 4, 2, 8, 0};

        int result = upperBound(5);

        System.out.println(result);
    }

    private static int binarySearch(int target) {
        // 정렬
        Arrays.sort(arr);

        // 중앙값 설정
        int mid;

        int l = 0;
        int r = arr.length - 1;

        while (l < r) {
            mid = (l + r) / 2;

            if (arr[mid] == target) {
                return mid;
            }

            else if (arr[mid] < target) {
                l = mid + 1;
            }
```

```
            else if (arr[mid] > target) {
                r = mid - 1;
            }
        }

        return -1 * l - 1;
    }

    private static int upperBound(int target) {
        // 정렬
        Arrays.sort(arr);

        // 중앙값 설정
        int mid;

        int l = 0;
        int r = arr.length;

        while (l < r) {
            mid = (l + r) / 2;

            if (arr[mid] <= target) {
                l = mid + 1;
            }
            else if (arr[mid] > target) {
                r = mid;
            }
        }

        return l;
    }

    private static int lowerBound(int target) {
        // 정렬
        Arrays.sort(arr);

        // 중앙값 설정
        int mid;

        int l = 0;
        int r = arr.length;

        while (l < r) {
            mid = (l + r) / 2;

            if (arr[mid] >= target) {
                r = mid;
            }
            else if (arr[mid] < target) {
                l = mid + 1;
            }
        }

        return l;
```

```
        }
    }
```

5. Upper Bound와 Lower Bound