이진 탐색 (Binary Search)

※ Data가 정렬이 되어 있어야 한다!

Algorithm,, Search

- 일반적 구현

```
    public static int binarySearch(int[] arr, int target) {
        int left = 0;
        int right = arr.length - 1;
        int mid = 0;
        while(left <= right) {
            mid = (left + right) / 2;
            if(arr[mid] < target)
                left = mid + 1;
            else if (arr[mid] > target)
                right = mid - 1;
            else
                return mid;
        }
        return Integer.MIN_VALUE;
    }
```

- 재귀적 구현

```
    public static int binarySearchRecursive(int[] arr, int target, int left,
int right) {
        if(left > right)
            return Integer.MIN_VALUE;
        int mid = (left + right) / 2;
        if(arr[mid] < target)
            return binarySearchRecursive(arr, target, left + 1, right);
        else if (arr[mid] > target)
            return binarySearchRecursive(arr, target, left, right - 1);
        else
            return mid;
    }
```

Interpolation Search (보간 탐색)

중간부터 찾지 말고, 실제 값이 있는 위치부터 찾기!

- 일반적 구현

```
    public static int interpolationSearch(int[] arr, int target) {
        int left = 0;
        int right = arr.length - 1;
        int mid = 0;
        int count = 0;
        while(left <= right) {
            mid = (int)(((double)target - arr[left]) / (arr[right] -
arr[left]) * (right - left) + left);
            if(arr[mid] < target)
                left = mid + 1;
            else if (arr[mid] > target)
                right = mid - 1;
            else {
                System.out.println(count);
                return mid;
            }
            count++;
        }
        return Integer.MIN_VALUE;
    }
```

- 재귀적 구현

```
    public static int interpolationSearchRecursive(int[] arr, int target,
int left, int right) {
        if(arr[left] > target || arr[right] < target) // 값의 범위를 넘어서는 경
우 있음.
            return Integer.MIN_VALUE;
        int mid = (int)(((double)target - arr[left]) / (arr[right] -
arr[left]) * (right - left) + left);
        if(arr[mid] < target)
            return interpolationSearchRecursive(arr, target, left + 1,
right);
        else if (arr[mid] > target)
            return interpolationSearchRecursive(arr, target, left, right -
1);
        else
            return mid;
    }
```

- A □ X ≤ B 조건을 검색. (반환값이 index+1 이므로)

```
  // 보간탐색을 이용한 근사값 찾기
    public static int interpolationSearch(char[] arr, int left, int right,
int target) {
        int mid = 0;
```

```
        while(left <= right) {
            mid = (int)(((double)target - arr[left]) / (arr[right] -
arr[left]) * (right - left) + left);
            System.out.println("mid ?? " + mid);
            if(arr[mid] < target)
                left = mid + 1;
            else if (arr[mid] > target)
                right = mid - 1;
            /*else {
                return mid;
            }*/
            // index가 1 이상이고, 끝이 아닐 때
            if(mid >= 1 && mid != arr.length - 1) {
                if(arr[mid - 1] < target && target <= arr[mid]) {
                    //return arr.length > mid + 1 ? mid + 1 : arr.length -
1;
                    return mid + 1;
                }
            }
            else if (mid == 0) {
                return 1;
            }
            else
                return Integer.MIN_VALUE;
        }
        return Integer.MIN_VALUE;
    }
```