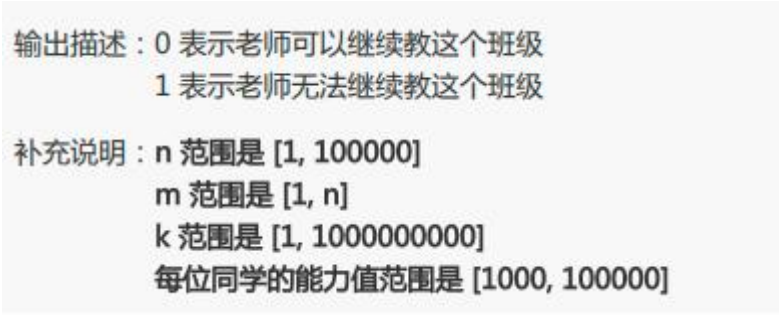
排队游戏

题目描述：新来的老师给班里的同学排一个队。每个学生有一个能力值。一些学生是刺头，不会听老师的话，自己选位置；非刺头同学在剩下的位置按照能力值从小到大排。 对于非刺头同学，如果发现他前面有能力值比自己高的同学，他不满程度就增加，增加的数量等于前面能力值比他大的同学的个数。刺头不会产生不满。 如果整个班级累计的不满程度超过 k，那么老师就没有办法教这个班级了。 输入描述：输入有三行：第一行为 n，m，k，空格隔开，分别表示班级总人数，刺头人数，最大不满程度 k。 第二行为刺头所在位置（从 0 开始，即排队数组的下标，比如 1 代表队伍中第 2 个同学是刺头），位置的数组也是排序的。 第三行有 n 个数，空格隔开， 表示老师排好的队中每个人的能力值，其中非刺头同学一定按照能力值从小到大排好序的。




```
import java.util.*;

// 注意类名必须为 Main, 不要有任何 package xxx 信息
public class Main {
    static long ans = 0;
    static int[] ability;
    static int[] idx;
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        // 注意 hasNext 和 hasNextLine 的区别
        while (in.hasNextInt()) { // 注意 while 处理多个 case
            int n = in.nextInt();
```

```
            int m = in.nextInt();
            int k = in.nextInt();
            idx = new int[n];
            ability = new int[n];
            for (int i = 0; i < m; i++) {
                int id= in.nextInt();
                idx[id] = 1;
            }
            PriorityQueue<Integer> pq = new PriorityQueue<>();
            for (int i = 0; i < n; i++) {
                ability[i] = in.nextInt();
            }
            for (int i = 0; i < n; i++) {
                if (idx[i] == 1) {
                    pq.offer(ability[i]);
                } else {
                    while (!pq.isEmpty() && pq.peek() < ability[i]) pq.poll();
                    ans += pq.size();
                }
            }
            System.out.println(ans <= k ? 0 : 1);
        }
}

// private static void merge(int l, int r) {
//         if (l == r) return;
//         int mid = (l + r) / 2;
//         merge(l, mid);
//         merge(mid + 1, r);
//         int[][] nums = new int[r - l + 1][];
//         int p1 = l, p2 = mid + 1, i = 0;
//         while (p1 <= mid && p2 <= r) {
//             if (ability[p1][1] <= ability[p2][1]) {
//                 nums[i++] = ability[p1++];
//             } else {
//                 if (idx[ability[p2][0]] == 0 && idx[ability[p1][0]] == 1) {
//                     ans += r - p1 + 1;
//                 }
//                 nums[i++] = ability[p2++];
//             }
//         }
//         while (p1 <= mid) nums[i++] = ability[p1++];
//         while (p2 <= r) nums[i++] = ability[p2++];
//         for (int[] n : nums) {
```

```
        //                  ability[l++] = n;
        //          }
        // }
}
```