```
        }
    }

    public HashMap<Integer, Integer> getReals(int[] arr,
                HashMap<Integer, Integer> cands) {
        HashMap<Integer, Integer> reals = new HashMap<Integer, Integer>();
        for (int i = 0; i != arr.length; i++) {
                int curNum = arr[i];
                if (cands.containsKey(curNum)) {
                        if (reals.containsKey(curNum)) {
                                reals.put(curNum, reals.get(curNum) + 1);
                        } else {
                                reals.put(curNum, 1);
                        }
                }
        }
        return reals;
    }
```

【扩展】

这种一次删掉 K 个不同的数的思想在面试中通常会变形之后反复出现。例如，下面这道面试真题：有一场投票，投票有效的条件是必须有一个候选人得票数超过半数，但是验票人员不能看到每张选票上选了谁，只能把任意两张选票放到一台机器上看这两张选票是否一样，若一样，则机器给出 true 的提醒，不一样则给出 false 的提醒。如果你作为验票的人员，怎么判断这场投票是有效的？

这道题目就是原问题的变形，但是“不能看到每张选票上选了谁”的这个限制实际上把用哈希表来解题的可能性完全堵死了。但本文的方法却可以满足题目的要求，因为我们实现的方法只需要当前数和候选数做比较，而不需要知道每个数的值。

在行列都排好序的矩阵中找数

【题目】

给定一个有 $N \times M$ 的整型矩阵 matrix 和一个整数 K，matrix 的每一行和每一列都是排好序的。实现一个函数，判断 K 是否在 matrix 中。

例如：

0　1　2　5

2　3　4　7

4　4　4　8

5　7　7　9

如果 K 为 7，返回 true；如果 K 为 6，返回 false。

【要求】

时间复杂度为 $O(N+M)$，额外空间复杂度为 $O(1)$。

【难度】

士 ★☆☆☆

【解答】

符合要求的解法比较巧妙且易于理解。

可以用以下步骤解决：

1. 从矩阵最右上角的数开始寻找(row=0，col=M-1)。
2. 比较当前数 matrix[row][col]与 K 的关系：

- 如果与 K 相等，说明已找到，直接返回 true。
- 如果比 K 大，因为矩阵每一列都已排好序，所以在当前数所在的列中，处于当前数下方的数都会比 K 大，则没有必要继续在第 col 列上寻找，令 col=col-1，重复步骤 2。
- 如果比 K 小，因为矩阵每一行都已排好序，所以在当前数所在的行中，处于当前数左方的数都会比 K 小，则没有必要继续在第 row 行上寻找，令 row=row+1，重复步骤 2。

3. 如果找到越界都没有发现与 K 相等的数，则返回 false。

或者，也可以用以下步骤：

1. 从矩阵最左下角的数开始寻找（row=N-1，col=0）。
2. 比较当前数 matrix[row][col]与 K 的关系：

- 如果与 K 相等，说明已找到，直接返回 true。
- 如果比 K 大，因为矩阵每一行都已排好序，所以在当前数所在的行中，处于当前数右方的数都会比 K 大，则没有必要继续在第 row 行上寻找，令 row=row-1，重复步骤 2。
- 如果比 K 小，因为矩阵每一列都已排好序，所以在当前数所在的列中，处于当前

数上方的数都会比 K 小，则没有必要继续在第 col 列上寻找，令 col=col+1，重复步骤 2。

3．如果找到越界都没有发现与 K 相等的数，则返回 false。

具体请参看如下代码中的 isContains 方法：

```
public boolean isContains(int[][] matrix, int K) {
        int row = 0;
        int col = matrix[0].length - 1;
        while (row < matrix.length && col > -1) {
                if (matrix[row][col] == K) {
                        return true;
                } else if (matrix[row][col] > K) {
                        col--;
                } else {
                        row++;
                }
        }
        return false;
}
```

最长的可整合子数组的长度

【题目】

先给出可整合数组的定义。如果一个数组在排序之后，每相邻两个数差的绝对值都为 1，则该数组为可整合数组。例如，[5,3,4,6,2]排序之后为[2,3,4,5,6]，符合每相邻两个数差的绝对值都为 1，所以这个数组为可整合数组。

给定一个整型数组 arr，请返回其中最大可整合子数组的长度。例如，[5,5,3,2,6,4,3]的最大可整合子数组为[5,3,2,6,4]，所以返回 5。

【难度】

尉 ★★☆☆

【解答】

时间复杂度高但容易理解的做法。对 arr 中的每一个子数组 arr[i..j](0<=i<=j<=N-1)，都验证一下是否符合可整合数组的定义，也就是把 arr[i..j]排序一下，看是否依次递增且每次递增 1。然后在所有符合可整合数组定义的子数组中，记录最大的那个长度，返回即可。