华为OD机试 - 恢复数字序列（Java & JS & Python）

题目描述

对于一个连续正整数组成的序列，可以将其拼接成一个字符串，再将字符串里的部分字符打乱顺序。如序列8 9 10 11 12，拼接成的字符串为89101112，打乱一部分字符后得到90811211，原来的正整数10就被拆成了0和1。

现给定一个按如上规则得到的打乱字符的字符串，请将其还原成连续正整数序列，并输出序列中最小的数字。

输入描述

输入一行，为打乱字符的字符串和正整数序列的长度，两者间用空格分隔，字符串长度不超过200，正整数不超过1000，保证输入可以还原成唯一序列。

输出描述

输出一个数字，为序列中最小的数字。

用例

输入	19801211 5
输出	8
说明	无

题目解析

本题“打乱字符的字符串”的是由“连续正整数序列”组成的，而题目又说：正整数不超过1000

因此，本题可以考虑使用滑动窗口，在1~1000整数范围内滑动，滑窗窗口的长度为输入的“连续正整数序列”的长度。

我们只需要统计“打乱字符的字符串”中各字符的数量，以及滑窗中各字符的数量，如果所有字符对于的数量都相同，那么说明对应滑窗就是一个符合要求的连续正整数数列，我们只需要取滑窗第一个数值即可。

滑窗在运动过程中，可以利用差异比较，即当前滑窗相交于上一个滑窗，失去了一个数字，新增了一个数字，当前滑窗只需要针对失去数字和新增数字的字符做数量修改即可。

```java
import java.util.HashMap;
import java.util.Scanner;

public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    String s = sc.next(); // 打乱字符的字符串
```

```
    int k = sc.nextInt(); // 连续正整数序列的长度

    System.out.println(getResult(s, k));
  }

  public static int getResult(String s, int k) {
    // base：统计打乱字符的字符串中 各字符的数量
    HashMap<Character, Integer> base = new HashMap<>();
    for (int i = 0; i < s.length(); i++) {
      char c = s.charAt(i);
      base.put(c, base.getOrDefault(c, 0) + 1);
    }

    // 初始滑窗（长度k）中各字符的数量
    HashMap<Character, Integer> count = new HashMap<>();
    for (int i = 1; i <= k; i++) {
      countNumChar(i + "", count, true);
    }

    // 比较滑窗各字符数量，和base统计的各字符数量是否一致，若一致，则说明初始滑窗就是一个符合要求的连续整数数列，该数列的最小值为1
    if (cmp(base, count)) return 1;

    // 否则继续尝试后续滑窗，注意题目说正整数不超过1000，因此我们可以尝试连续正整数序列取值范围就是1~1000
    for (int i = 2; i <= 1000 - k + 1; i++) {
      // 相较于上一个滑窗失去的数字
      String remove = i - 1 + "";
      countNumChar(remove, count, false);

      // 相较于上一个滑窗新增的数字
      String add = i + k - 1 + "";
      countNumChar(add, count, true);

      // 比较
      if (cmp(base, count)) return i;
    }

    return -1; // 题目说存在唯一的序列满足条件，因此这里返回语句是走不到的
  }
```

```
47
48    public static void countNumChar(String num, HashMap<Character, Integer> count,
      boolean isAdd) {
49        for (int j = 0; j < num.length(); j++) {
50            char c = num.charAt(j);
51            count.put(c, count.getOrDefault(c, 0) + (isAdd ? 1 : -1));
52        }
53    }
54
55    public static boolean cmp(HashMap<Character, Integer> base, HashMap<Character,
      Integer> count) {
56        for (Character c : base.keySet()) {
57            if (!count.containsKey(c) || count.get(c) - base.get(c) != 0) {
58                return false;
59            }
60        }
61        return true;
62    }
63 }
```