【哈希表】-恢复数字序列

视频直播讲解：2024/09/07 真题讲解（2024E卷）

题目描述与示例

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。

示例一

输入

```
19801211 5
```

输出

```
8
```

说明

还原出的序列为 `8 9 10 11 12`，故输出 `8`

示例二

输入

```
1 432111111111 4
```

输出

```
1 111
```

说明

还原出的序列为 `111 112 113 114`，故输出 `111`

解题思路

直接从序列 `s` 中进行判断，较难完成。

考虑到数据量不大，可以通过穷举的方式来完成。

对于 `1` 到 `1000` 的数字 `num`，考虑长度为 `n` 的序列

`num num+1 num+2 ... num+n-2 num+n-1`

将该序列中的每一个数字转化为字符串，再统计每一个数字字符出现的个数，用哈希表 `cnt_new` 来记录。

原序列 `s` 中的数字字符，用另外一个哈希表 `cnt` 来记录。即

```
for num in range(1, 1001):
    cnt_new = Counter()
    for i in range(num, num+n):
        for ch in str(i):
            cnt_new[ch] += 1
```

或者更加简洁的写法

```
for num in range(1, 1001):
    cnt_new = Counter("".join(str(i) for i in range(num, num+n)))
```

若发现存在 `cnt_new == cnt` ，说明此时 `num` 是所寻找的序列的第一个数字。

这是一种**相对而言比较暴力的解法**，但也足以通过所有用例。

如果想要进一步优化，可以使用**固定滑窗**的做法。

固定长度为序列长度 `n` 的窗口，对正整数序列 `1 2 3 ... 1000` 进行滑动，构建一个 `win_cnt` 表示窗口中各个数字出现的次数。

左边的数字离开，则在 `win_cnt` 中减去对应的频次；右边的数字加入，则在 `win_cnt` 中加入对应的频次。

感兴趣的同学可以自行完成上述代码。

代码

Python

```
# 题目：【哈希表】2024E-恢复数字序列
# 分值：100
# 作者：许老师-闭着眼睛学数理化
# 算法：哈希表
# 代码看不懂的地方，请直接在群上提问


from collections import Counter

# 属于打乱的字符串s和数字序列长度n
s, n = input().split()
# 将n转换为数字
n = int(n)

```

```python
# 将序列s中的每一个字符进行统计
cnt = Counter(s)

# 序列最大不会超过1000，故遍历1-1000的所有元素
for num in range(1, 1001):
    # 考虑长度为n的序列
    # 起始位置为num，终止位置为num+n-1
    # 将每一个int转为str后再进行字符串合并
    # 合并完再使用Counter进行元素频率统计
    cnt_new = Counter("".join(str(i) for i in range(num, num+n)))
    # 如果cnt_new等于cnt，直接输出num，并且退出循环
    if cnt_new == cnt:
        print(num)
        break
```

Java

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

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String s = scanner.next();
        int n = scanner.nextInt();
        scanner.close();

        Map<Character, Integer> cnt = new HashMap<>();
        for (char c : s.toCharArray()) {
            cnt.put(c, cnt.getOrDefault(c, 0) + 1);
        }

        for (int num = 1; num <= 1000; num++) {
            Map<Character, Integer> cntNew = new HashMap<>();
            for (int i = num; i < num + n; i++) {
                String iStr = Integer.toString(i);
                for (char c : iStr.toCharArray()) {
                    cntNew.put(c, cntNew.getOrDefault(c, 0) + 1);
                }
            }
            if (cntNew.equals(cnt)) {
                System.out.println(num);
                break;
            }
```

```
        }
    }
}

```

C++

```cpp
#include <iostream>
#include <string>
#include <unordered_map>

int main() {
    std::string s;
    int n;
    std::cin >> s >> n;

    std::unordered_map<char, int> cnt;
    for (char c : s) {
        cnt[c]++;
    }

    for (int num = 1; num <= 1000; num++) {
        std::unordered_map<char, int> cntNew;
        for (int i = num; i < num+n; i++){
            std::string iStr = std::to_string(i);
            for (char c : iStr) {
                cntNew[c]++;
            }
        }
        if (cntNew == cnt) {
            std::cout << num << std::endl;
            break;
            }
    }

    return 0;
}

```

时空复杂度

时间复杂度：`O(KM)`。其中`M`为字符串长度最大值为`200`，`K`为序列的最大值为`1000`。

空间复杂度：`O(1)`。哈希表的长度最多为`10`，可以认为是常数时间复杂度。