LeetCode 1417. 重新格式化字符串

1417. 重新格式化字符串

难度 简单 2 收藏 分享 切换为英文 关注 反馈

给你一个混合了数字和字母的字符串 `s`，其中的字母均为小写英文字母。

请你将该字符串重新格式化，使得任意两个相邻字符的类型都不同。也就是说，字母后面应该跟着数字，而数字后面应该跟着字母。

请你返回 **重新格式化后** 的字符串；如果无法按要求重新格式化，则返回一个 **空字符串** 。

示例 1：

```
输入：s = "a0b1c2"
输出："0a1b2c"
解释："0a1b2c" 中任意两个相邻字符的类型都不同。 "a0b1c2", "0a1b2c", "0c2a1b" 也是满足题目要求的答案。
```

示例 2：

```
输入：s = "leetcode"
输出：""
解释："leetcode" 中只有字母，所以无法满足重新格式化的条件。
```

示例 3：

```
输入：s = "1229857369"
输出：""
解释："1229857369" 中只有数字，所以无法满足重新格式化的条件。
```

示例 4：

```
输入：s = "covid2019"
输出："c2o0v1i9d"
```

```cpp
class Solution {
public:
    string reformat(string s) {
        vector<char> d, c;
        for (char x : s)
            if (isdigit(x))
                d.push_back(x);
            else
                c.push_back(x);

        string ans;
        if (d.size() == c.size() - 1) {
            for (int i = 0; i < d.size(); i++) {
                ans += c[i];
```

```
                ans += d[i];
            }
            ans += c.back();
        } else if (c.size() == d.size() - 1) {
            for (int i = 0; i < c.size(); i++) {
                ans += d[i];
                ans += c[i];
            }
            ans += d.back();
        } else if (d.size() == c.size()) {
            for (int i = 0; i < c.size(); i++) {
                ans += d[i];
                ans += c[i];
            }
        }

        return ans;
    }
};
```

LeetCode 1418. 点菜展示表

1418. 点菜展示表

难度 中等 13 收藏 分享 切换为英文 关注 反馈

给你一个数组 `orders`，表示客户在餐厅中完成的订单，确切地说， `orders[i]=[customerName_i,tableNumber_i,foodItem_i]` ，其中 `customerName_i` 是客户的姓名，`tableNumber_i` 是客户所在餐桌的桌号，而 `foodItem_i` 是客户点的餐品名称。

请你返回该餐厅的 **点菜展示表** 。在这张表中，表中第一行为标题，其第一列为餐桌桌号 “Table” ，后面每一列都是按字母顺序排列的餐品名称。接下来每一行中的项则表示每张餐桌订购的相应餐品数量，第一列应当填对应的桌号，后面依次填写下单的餐品数量。

注意：客户姓名不是点菜展示表的一部分。此外，表中的数据行应该按餐桌桌号升序排列。

示例 1：

```
输入：orders = [["David","3","Ceviche"],["Corina","10","Beef Burrito"],
["David","3","Fried Chicken"],["Carla","5","Water"],["Carla","5","Ceviche"],
["Rous","3","Ceviche"]]
输出：[["Table","Beef Burrito","Ceviche","Fried Chicken","Water"],
["3","0","2","1","0"],["5","0","1","0","1"],["10","1","0","0","0"]]
解释：
点菜展示表如下所示：
Table,Beef Burrito,Ceviche,Fried Chicken,Water
3    ,0           ,2      ,1            ,0
5    ,0           ,1      ,0            ,1
10   ,1           ,0      ,0            ,0
对于餐桌 3：David 点了 "Ceviche" 和 "Fried Chicken"，而 Rous 点了 "Ceviche"
而餐桌 5：Carla 点了 "Water" 和 "Ceviche"
餐桌 10：Corina 点了 "Beef Burrito"
```

示例 2：

```
输入：orders = [["James","12","Fried Chicken"],["Ratesh","12","Fried Chicken"],
["Amadeus","12","Fried Chicken"],["Adam","1","Canadian Waffles"],
["Brianna","1","Canadian Waffles"]]
输出：[["Table","Canadian Waffles","Fried Chicken"],["1","2","0"],
["12","0","3"]]
```

```cpp
class Solution {
public:
    vector<vector<string>> displayTable(vector<vector<string>>& orders) {
        set<string> foods;
        for (const auto &v : orders)
            foods.insert(v[2]);

        int cnt = 1;
        unordered_map<string, int> fp;

        for (const auto &s : foods)
            fp[s] = cnt++;


        vector<vector<int>> res;
```

```
        cnt = 0;
        unordered_map<string, int> tp;
        for (const auto &v : orders) {
            if (tp.find(v[1]) == tp.end()) {
                tp[v[1]] = cnt;
                res.push_back(vector<int>(foods.size() + 1));
                res[cnt][0] = stoi(v[1]);
                cnt++;
            }

            res[tp[v[1]]][fp[v[2]]]++;
        }

        sort(res.begin(), res.end());

        vector<vector<string>> ans;

        ans.push_back(vector<string>(1));
        ans[0][0] = "Table";

        for (const auto &v : foods)
            ans[0].push_back(v);

        for (int i = 0; i < res.size(); i++) {
            ans.push_back(vector<string>(foods.size() + 1));
            for (int j = 0; j <= foods.size(); j++)
                ans[i + 1][j] = to_string(res[i][j]);
        }

        return ans;
    }
};
```

1419. 数青蛙

1419. 数青蛙

难度 中等 14 收藏 分享 切换为英文 关注 反馈

给你一个字符串 `croakOfFrogs`，它表示不同青蛙发出的蛙鸣声（字符串 "croak" ）的组合。由于同一时间可以有多只青蛙呱呱作响，所以 `croakOfFrogs` 中会混合多个 “croak” 。请你返回模拟字符串中所有蛙鸣所需不同青蛙的最少数目。

注意：要想发出蛙鸣 "croak"，青蛙必须 **依序** 输出 `‘c’, ’r’, ’o’, ’a’, ’k’` 这 5 个字母。如果没有输出全部五个字母，那么它就不会发出声音。

如果字符串 `croakOfFrogs` 不是由若干有效的 "croak" 字符混合而成，请返回 `-1` 。

示例 1：

```
输入：croakOfFrogs = "croakcroak"
输出：1
解释：一只青蛙 “呱呱” 两次
```

示例 2：

```
输入：croakOfFrogs = "crcoakroak"
输出：2
解释：最少需要两只青蛙，“呱呱” 声用黑体标注
第一只青蛙 "crcoakroak"
第二只青蛙 "crcoakroak"
```

示例 3：

```
输入：croakOfFrogs = "croakcrook"
输出：-1
解释：给出的字符串不是 "croak" 的有效组合。
```

示例 4：

```
输入：croakOfFrogs = "croakcroa"
输出：-1
```

```cpp
class Solution {
public:
    int minNumberOfFrogs(string croakOfFrogs) {
        int n = croakOfFrogs.size();
        vector<int> cnt(4, 0);
        int pending = 0;
        int ans = 0;

        for (char c : croakOfFrogs) {
            if (c == 'c') {
                cnt[0]++;
                pending++;
            } else if (c == 'r') {
                if (cnt[0] == 0)
                    return -1;
                cnt[0]--;
                cnt[1]++;
```

```
            } else if (c == 'o') {
                if (cnt[1] == 0)
                    return -1;
                cnt[1]--;
                cnt[2]++;
            } else if (c == 'a') {
                if (cnt[2] == 0)
                    return -1;
                cnt[2]--;
                cnt[3]++;
            } else if (c == 'k') {
                if (cnt[3] == 0)
                    return -1;
                cnt[3]--;
                pending--;
            }
            ans = max(ans, pending);
        }

        if (pending > 0)
            return -1;

        return ans;
    }
};
```

LeetCode 1420. 生成数组

1420. 生成数组

难度 困难 25 收藏 分享 切换为英文 关注 反馈

给你三个整数 `n`、`m` 和 `k` 。下图描述的算法用于找出正整数数组中最大的元素。

```
maximum_value = -1
maximum_index = -1
search_cost = 0
n = arr.length
for (i = 0; i < n; i++) {
    if (maximum_value < arr[i]) {
        maximum_value = arr[i]
        maximum_index = i
        search_cost = search_cost + 1
    }
}
return maximum_index
```

请你生成一个具有下述属性的数组 `arr` ：

- `arr` 中有 `n` 个整数。
- `1 <= arr[i] <= m` 其中 `(0 <= i < n)` 。
- 将上面提到的算法应用于 `arr` ， `search_cost` 的值等于 `k` 。

返回上述条件下生成数组 `arr` 的 **方法数** ，由于答案可能会很大，所以 **必须** 对 `10^9 + 7` 取余。

示例 1：

输入： `n = 2, m = 3, k = 1`
输出： `6`
解释： 可能的数组分别为 `[1, 1], [2, 1], [2, 2], [3, 1], [3, 2] [3, 3]`