LeetCode 1413. 逐步求和得到正数的最小值

1413. 逐步求和得到正数的最小值

难度 简单 3 收藏 分享 切换为英文 关注 反馈

给你一个整数数组 `nums` 。你可以选定任意的 **正数** startValue 作为初始值。

你需要从左到右遍历 `nums` 数组，并将 startValue 依次累加上 `nums` 数组中的值。

请你在确保累加和始终大于等于 1 的前提下，选出一个最小的 **正数** 作为 startValue 。

示例 1:

```
输入：nums = [-3,2,-3,4,2]
输出：5
解释：如果你选择 startValue = 4，在第三次累加时，和小于 1 。
                累加求和
                startValue = 4 | startValue = 5 | nums
                  (4 -3 ) = 1  | (5 -3 ) = 2    |  -3
                  (1 +2 ) = 3  | (2 +2 ) = 4    |   2
                  (3 -3 ) = 0  | (4 -3 ) = 1    |  -3
                  (0 +4 ) = 4  | (1 +4 ) = 5    |   4
                  (4 +2 ) = 6  | (5 +2 ) = 7    |   2
```

示例 2:

```
输入：nums = [1,2]
输出：1
解释：最小的 startValue 需要是正数。
```

示例 3:

```
输入：nums = [1,-2,-3]
输出：5
```

```cpp
class Solution {
public:
    int minStartValue(vector<int>& nums) {
        int s = 0, m = 1;
        for (int x : nums) {
            s -= x;
            m = max(m, s + 1);
        }

        return m;
    }
};
```

1414. 和为 K 的最少斐波那契数字数目

1414. 和为 K 的最少斐波那契数字数目

难度 中等 3 收藏 分享 切换为英文 关注 反馈

给你数字 `k` ，请你返回和为 `k` 的斐波那契数字的最少数目，其中，每个斐波那契数字都可以被使用多次。

斐波那契数字定义为：

- $F_1 = 1$
- $F_2 = 1$
- $F_n = F_{n-1} + F_{n-2}$ ， 其中 $n > 2$ 。

数据保证对于给定的 `k` ，一定能找到可行解。

示例 1：

输入： k = 7
输出： 2
解释： 斐波那契数字为：1，1，2，3，5，8，13，……
对于 k = 7 ，我们可以得到 2 + 5 = 7 。

示例 2：

输入： k = 10
输出： 2
解释： 对于 k = 10 ，我们可以得到 2 + 8 = 10 。

示例 3：

输入： k = 19
输出： 3
解释： 对于 k = 19 ，我们可以得到 1 + 5 + 13 = 19 。

```
class Solution {
public:
    int findMinFibonacciNumbers(int k) {
        int x = 1, y = 1;
        while (x + y <= k) {
            int t = x + y;
            x = y; y = t;
        }

        int ans = 0;
        while (k > 0) {
            if (k >= y) {
                k -= y;
                ans++;
            }
            int t = y - x;
            y = x; x = t;
        }

        return ans;
    }
};
```

1415. 长度为 n 的开心字符串中字典序第 k 小的字符串

1415. 长度为 n 的开心字符串中字典序第 k 小的字符串

难度 中等 7 收藏 分享 切换为英文 关注 反馈

一个「开心字符串」定义为：

- 仅包含小写字母 `['a', 'b', 'c']`.
- 对所有在 `1` 到 `s.length - 1` 之间的 `i` ，满足 `s[i] != s[i + 1]` （字符串的下标从 1 开始）。

比方说，字符串 **"abc"**，**"ac"**，**"b"** 和 **"abcbabcbcb"** 都是开心字符串，但是 **"aa"**，**"baa"** 和 **"ababbc"** 都不是开心字符串。

给你两个整数 `n` 和 `k` ，你需要将长度为 `n` 的所有开心字符串按字典序排序。

请你返回排序后的第 k 个开心字符串，如果长度为 `n` 的开心字符串少于 `k` 个，那么请你返回 **空字符串** 。

示例 1：

```
输入：n = 1, k = 3
输出："c"
解释：列表 ["a", "b", "c"] 包含了所有长度为 1 的开心字符串。按照字典序排序后第三个字符串为 "c" 。
```

示例 2：

```
输入：n = 1, k = 4
输出：""
解释：长度为 1 的开心字符串只有 3 个。
```

示例 3：

```
输入：n = 3, k = 9
输出："cab"
解释：长度为 3 的开心字符串总共有 12 个 ["aba", "abc", "aca", "acb", "bab", "bac", "bca", "bcb", "cab", "cac", "cba", "cbc"] 。第 9 个字符串为 "cab"
```

```cpp
class Solution {
public:
    bool solve(int x, string &cur, int &k) {
        if (x == 0) {
            k--;
            if (k == 0) return true;
            return false;
        }

        for (char c = 'a' ; c <= 'c'; c++)
            if (cur.size() == 0 || cur.back() != c) {
                cur += c;
                if (solve(x - 1, cur, k))
                    return true;
                cur.pop_back();
            }
```

```
        return false;
    }

    string getHappyString(int n, int k) {
        string cur;
        if (solve(n, cur, k))
            return cur;
        return "";
    }
};
```

LeetCode 1416. 恢复数组

1416. 恢复数组

难度 困难 14 收藏 分享 切换为英文 关注 反馈

某个程序本来应该输出一个整数数组。但是这个程序忘记输出空格了以致输出了一个数字字符串，我们所知道的信息只有：数组中所有整数都在 `[1, k]` 之间，且数组中的数字都没有前导 0 。

给你字符串 `s` 和整数 `k` 。可能会有多种不同的数组恢复结果。

按照上述程序，请你返回所有可能输出字符串 `s` 的数组方案数。

由于数组方案数可能会很大，请你返回它对 `10^9 + 7` **取余** 后的结果。

示例 1：

输入： `s = "1000", k = 10000`
输出： `1`
解释： 唯一一种可能的数组方案是 `[1000]`

示例 2：

输入： `s = "1000", k = 10`
输出： `0`
解释： 不存在任何数组方案满足所有整数都 `>= 1` 且 `<= 10` 同时输出结果为 `s` 。

示例 3：

输入： `s = "1317", k = 2000`
输出： `8`
解释： 可行的数组方案为 `[1317]`，`[131,7]`，`[13,17]`，`[1,317]`，`[13,1,7]`，`[1,31,7]`，`[1,3,17]`，`[1,3,1,7]`

示例 4：

输入： `s = "2020", k = 30`
输出： `1`
解释： 唯一可能的数组方案是 `[20,20]` 。 `[2020]` 不是可行的数组方案，原因是 `2020 > 30` 。 `[2,020]` 也不是可行的数组方案，因为 `020` 含有前导 `0` 。

```
class Solution {
public:
```

```
    #define LL long long
    int numberOfArrays(string s, int k) {
        const int mod = 1000000007;
        int n = s.size();
        vector<int> f(n + 1, 0);
        f[0] = 1;

        for (int i = 1; i <= n; i++) {
            LL cur = 0, p = 1;
            for (int j = i - 1; j >= 0 && p <= k; j--, p *= 10) {
                cur += (s[j] - '0') * p;
                if (cur <= k && s[j] != '0')
                    f[i] = (f[i] + f[j]) % mod;
            }
        }

        return f[n];
    }
};
```