一道Hard类型的题目：【双周赛赛题】

题目描述：

5720. 使字符串有序的最少操作次数

给你一个字符串 `s` （**下标从 0 开始**）。你需要对 `s` 执行以下操作直到它变为一个有序字符串：

1. 找到 **最大下标** `i` ，使得 `1 <= i < s.length` 且 `s[i] < s[i - 1]` 。
2. 找到 **最大下标** `j` ，使得 `i <= j < s.length` 且对于所有在闭区间 `[i, j]` 之间的 `k` 都有 `s[k] < s[i - 1]` 。
3. 交换下标为 `i - 1` 和 `j` 处的两个字符。
4. 将下标 `i` 开始的字符串后缀反转。

请你返回将字符串变成有序的最少操作次数。由于答案可能会很大，请返回它对 $10^9 + 7$ **取余** 的结果。

我的分析：

1. 这道题如果按照题目的要求按部就班的做的话铁定超时
2. 数学思维：

3.这里的运算需要用到阶乘，并且可能用到的还非常的大，求大数阶乘可以用动态规划

4.这里还需要处理的一个问题式大整数的加法，乘法相关的取余问题

大佬代码，可以看懂一点

```
#define LL long long
const LL MOD = 1e9 + 7;
const int MAXN = 3050;
int cnt[30];
LL comb[MAXN][MAXN];



class Solution {
public:
    void init(int n){
        comb[0][0] = 1;
        for (int i = 1; i <= n; i++){
            comb[i][0] = comb[i][i] = 1;
            for (int j = 1; j < i; j++) comb[i][j] = (comb[i - 1][j - 1]
+ comb[i - 1][j]) % MOD;
        }
    }
    LL Comb(int n, int m){
        if (n == 0) return 1LL;
        if (m == 0) return 1LL;
        if (m > n) return 1LL;
        return comb[n][m];
    }
    int makeStringSorted(string S) {
        int n = S.length();
        init(n);
        LL ans = 0;
        memset(cnt, 0, sizeof(cnt));

        for (int i = n - 1; i >= 0; i--){
            int C = S[i] - 'a';
            cnt[C] += 1;

            for (int c = C - 1; c >= 0; c--){
                if (cnt[c] == 0) continue;
                cnt[c] -= 1;
                int left = n - i - 1; LL cur = 1;
                for (int k = 0; k < 26; k++){
                    cur = cur * Comb(left, cnt[k]) % MOD;
                    left -= cnt[k];
                }
                cnt[c] += 1;
                ans = (ans + cur) % MOD;
            }
        }
```

```
        return ans;
    }
};
```