Week 12

比赛简记

Max.D.

专题

暂无

比赛

一场cf edu, 好惨好惨

题目

暂无

Hardict

专题

无

比赛

cf div2

题目

暂无

MountVoom

专题

无

比赛

cf edu, 算错了复杂度浪费了半个小时。

题目

无

个人总结

陈铭煊 Max.D.

补题+学习

龙鹏宇 Hardict

补题+整理板子

肖思炀 MountVoom

补题

本周推荐

陈铭煊 Max.D.

来源：

Codeforces Round 662(Div. 2) E2 Twilight and Ancient Scroll (harder version)

标签：

字符串处理，动态规划，哈希，双指针

题意：

给出n个非空字符串，每个长L_i现在让你在每个字符串中选择删除一个字符，或者是不删除，使得得到的字符串组按顺序字典序不减，求符合要求的操作的个数，答案对1000000007取模。

$1\le n\le 10^5,\sum L_i\le 10^6$

题解：

我们容易想到设置dp状态为$dp[i][j]$表示到了第i个字符串删除第$j(0\le j \le L_i)$个字符，满足前i个不减的种类数（其中$j=L_i$代表不删除）。这个时候我们会发现即使用一些预处理来快速比较相邻两个删除后的字符串，由于更新时要枚举位置，所以效率不会比$O(n*\sum L_i)$更优。

这时候有个很精彩的技巧，我们如果能够预处理出每个字符串删除之后得到的字符串的一个字典序数组，那么dp更新时等价于做一个区间加操作——对于前状态的字符串，用双指针找到第一个比其大的后状态，接着后面的所有状态都可以加上这个前状态的值。

怎么预处理呢？通过双指针求出一个nxt数组$nxt[i]$代表第i个字符之后第一个出现不同的位置。那么对于两个删除点p,q我们就可以通过对nxt和其大小的讨论，对状态字符串进行排序。

```
    for (int i = 0; i < res.size(); i++)
        res[i] = i;
    sort(res.begin(), res.end(), [&](int x, int y) {
        bool f = true;
        if (x > y)
            swap(x, y), f = false;
        if (nxt[x] <= y) {
            f ^= s[nxt[x]] > s[x];
        } else
            f = !f;
        return f;
    });
```

剩下还有一个问题，怎么快速比较前后两个状态字符串大小呢？这里用哈希求出公共前缀更方便，当然用后缀数组也可以。

因此总的效率为$O((n+\log\sum L_i)*\log \sum L_i)$

评论：

离散的动态规划我们可以将状态排序使之紧凑，很好的思路。

龙鹏宇 Hardict

来源：

HDU 6061

标签：

多项式卷积,NTT

题意：

先给定一个$f(x)=\sum_{i=0}^{n}c_{i}x^{i}$,然后求解$g(x)=f(x-\sum_{i}a_{i})=\sum_{i=0}^{n}b_{i}x^{i}$

输出b_{i}

题解：

$A=-sum{a[i]}$并取模变成求解$f(x+A)$少去正负计算

然后按$f(x+A)$每一项进行一个展开(会成一个三角表)

目标多项式系数$b_j=\sum_{i \geq j} c_i A^{i-j} C_i^j$

化解后有$j!A^jb_j=\sum_{i=j}^{n} \frac{c_{i}i!A^i}{(i-j)!}$

整体上$g(x)=\sum_{i=0}^{n}\frac{x^{i}}{A^{i}i!}\sum_{j=0}^{n-i}\frac{c_{i+j}(i+j)!A^{i+j}}{j!}$

这里令$k=n-(i+j),g(x)=\sum_{i=0}^{n}\frac{x^{i}}{A^{i}i!}\sum_{j+k=n-i}\frac{c_{n-k}(n-k)!A^{n-k}}{j!}$
$=\sum_{k=0}^{n}\frac{x^{k}}{A^{k}k!}\sum_{n+k=i+j}\frac{c_{i}(i)!A^{i}}{(n-j)!}$

将$\frac{1}{j!}$当作一个翻转(两者都可),求$\sum_{i}c_{i}i!A^{i}x^{i}$与$\sum_{i}\frac{1}{(n-i)!}x^{i}$的卷积,然后取结果$n+i$项系数即可

肖思炀 MountVoom

来源：

D. Berserk And Fireball

标签：

枚举，贪心

题意：

给定一个两个排列，长度分别为n和m

有两种操作，花费x的代价消除恰好k个连续的人(操作1)或者花费y的代价消除相邻的两个人中力量较小的人(操作2)，消除以后两边会合并。

问最小花费多少代价可以使得第一个排列变成第二个排列，有无解情况。

题解：

首先第二个排列一定要满足是第一个排列的子序列。

然后考虑如何消除子序列相邻两个位置间的人，设要留下的数字是a和b那么这之间如果有比a和b都大的数字，那么我们至少要进行一次操作1。

分两种情况，如果$y * k \leq x$那么肯定尽量一个一个消除更优，如果都比a或者b小那么直接一个一个消除即可。否则我们考虑最大的数，可以用它先把其它的数消除到只剩k个然后一波带走，如果人数本身不够k个那么显然无解。

如果$y * k > x$也就是说尽量一段一段的消除，从前往后贪心即可，遇到比a或者b小的数就先留着，遇到比a和b都大的数就以它为起点删除长度为k的一段，如果长度不够看去拿开始留着的数凑就行了，凑不够就无解。最后剩下的数先尽量整段消除再一个一个消除即可。

评论：

还是分类讨论没有处理好WA了一发。