

# 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 涨回原rating

#### 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。

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

**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  $j$  ( $0 \leq j \leq L_i$ ) 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  $O(n \cdot \sum L_i)$  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  $O((n + \log \sum L_i) \cdot \log \sum L_i)$

**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：**

[SWERC 2018 J](#)

标签：

hash碰撞，随机算法

题意：

给定一个随机数生成器，以及 $4$ 个种子 $S_i$ 知道算法可以求得其运行 $C_i$ 次会生成数 $x_i$ (这个需要一步一步求)，我们截取 $x_i$ 二进制前 $N$ 位，计算 $w = x_1 \oplus x_2 \oplus x_3 \oplus x_4$ 题目要求选取合适的 $C_i$ 使 $w = 0$

题解：

下述 $x_i$ 默认只保留二进制前 $N$ 位

先考虑两个数情况，对于 $x_1$ 与 $x_2$ 我们实际是要寻找 $x_1 = x_2$ 情况，对于给定种子 $S_i$ 我们可以根据随机序列定义 $\text{Hash}: h(S_i, x_1) = c_k$ 即给定种子下生成值与次数的关系

那么 $x_1 = x_2$ 可以理解为对 $x_1$ 的 $\text{Hash}$ 在 $\{x_2\}$ 中寻找一次原像碰撞

将设我们对 $x_1$ 进行哈希了 $D$ 次，我们可以利用\*生日攻击\*知道 $D = 2^{\frac{N}{2}}$ 最优（或者利用期望耗费为 $D + \frac{2^N}{D}$ 分析）

于是对于原题我们得到一种想法：对 $\{x_1\}$ 构建大小为 $2^{\frac{N}{2}}$ 的哈希表 $\{x_3, x_4\}$ 随便生成，遍历 $\{x_2\}$ 寻找碰撞 $x_1 = x_2 \oplus x_3 \oplus x_4$ 复杂度为 $O(2^{\frac{N}{2}})$ 会TLE

现在考虑利用 $x_3, x_4$ 可以考虑将 $(x_1, x_2), (x_3, x_4)$ 看作两个整体，要求 $x_1 \oplus x_2 \oplus x_3 \oplus x_4$ 分别将前 $M$ 比特位碰撞为0，再在对于两个整体在后 $N-M$ 位中寻找哈希，复杂度为 $O(4 * 2^{\frac{M}{2}})$

$M$ 作为参数进行选取，理论 $\frac{N}{2}$ 即可，但实际可能没有碰撞，调参后选取 $\frac{2N}{3} + 4$ 通过

## 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。

From:  
<https://wiki.cvbbacm.com/> - CVBB ACM Team

Permanent link:  
[https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:alchemist:weekly\\_digest\\_12&rev=1598606732](https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:alchemist:weekly_digest_12&rev=1598606732)

Last update: 2020/08/28 17:25