第一场多校题解

ICPC Team, Beijing University of Posts and Telecommunications

July 22nd, 2019

1 Blank

定义 $dp[i][j][k][t]$ 代表填完前 t 个位置后，$\{0,1,2,3\}$ 这 4 个数字最后一次出现的位置，排序后为 $i,j,k,t(i<j<k<t)$ 的方案数目，则按照第 $t+1$ 位的数字的四种选择，可以得到四种转移。

对于限制可以按照限制区间的右端点分类，求出 $dp[i][j][k][t]$ 后，找到所有以 t 为区间右端点的限制条件，如果当前状态不满足所有限制条件则不合法，不再向后转移。

总时间复杂度 $O(n^4)$。滚动一维，空间复杂度 $O(n^3)$

UPD 一开始的模数为 10^9+7，后来统一改成了 998244353，但最后在 polygon 上生成数据的时候标程放错了，就跑出了模 10^9+7 的结果。对被这道题卡了的大佬们表示歉意！

2 Operation

暴力的做法可以用数据结构维护区间线性基，但肯定过不了。

贪心地维护序列的前缀线性基 (上三角形态)，对于每个线性基，将出现位置靠右的数字尽可能地放在高位，也就是说在插入新数字的时候，要同时记录对应位置上数字的出现位置，并且在找到可以插入的位置的时候，如果新数字比位置上原来的数字更靠右，就将该位置上原来的数字向低位推。

在求最大值的时候，从高位向低位遍历，如果该位上的数字出现在询问中区间左端点的右侧且可以使答案变大，就异或到答案里。

对于线性基的每一位，与它异或过的线性基更高位置上的数字肯定都出现在它右侧 (否则它就会被插入在那个位置了)，因此做法的正确性显然。

3 Milk

由于只能从上向下走，因此可以将行离散化，然后从上向下做背包。

记第 i 行的牛奶数为 c_i，则对于第 i 行，求出在行内向左/右走喝 $1,2,...,c_i$ 包牛奶并且回到/不回到行中点的最短时间，然后合并背包求出在第 i 行内喝 $1,2,...,c_i$ 包牛奶并且回到/不回到行中点的最短时间，然后和在前 $i-1$ 行喝牛奶并回到中点的背包合并，求出在前 i 行内喝 $1,2,...,k$ 包牛奶并且回到/不回到行中点的最短时间，并用不回到中点的背包更新答案。每一行处理的复杂度为 $O(kc_i)$，因此总复杂度为 $O(k^2)$。

注意对第一行要特殊处理。

4 Vacation

把第 i 辆车追上第 $i+1$ 辆车当作一个事件，显然只有 n 个事件，且第 i 辆车追上第 $i+1$ 辆车只可能会对第 $i-1$ 辆车追上第 i 辆车的时间产生影响，且时间一定是变小，因此可以维护车之间的距离和速度来计算事件发生时间，用堆来找出最早发生的事件，不停处理直到 0 车通过停车线。复杂度为 $O(n\log n)$。

上述做法比较麻烦，可以直接二分最终时间，然后从第一辆车开始递推求出每辆车的最终位置。复杂度为 $O(n\log C)$，也可以过。

UPD 发现有很多大佬写了 $O(n)$ 的做法，大概是这样：最终通过停止线的时候，一定是一个车后面堵着剩余所有的车，那么影响时间的就只有最前面这辆车，所以对于每一辆车，假设是它是和 0 车堵在一起的最靠前的一辆车，那么可以计算出一个值，所有的车的计算值的最大值就是答案。

5 Path

题意为给定一个有向图，删掉一条边的代价为边权，要求以尽可能少的总代价删掉一些边，使得最短路增加。

先求出最短路，然后保留所有满足 $d_{e_y}-d_{e_x}=e_w$ 的边，对于新的图求 1 到 n 的最小割即为答案。

6 Typewriter

对于 i 从小到大处理，维护使得 $s[j:i]\in s[1:j-1]$ 的最小的 j($s[l:r]$ 表示子串 $s_ls_{l+1}...s_r$)，那么记 $f[i]$ 为输出前 i 个字符的最小代价，则 $f[i]=\min\{f[i-1]+p,f[j-1]+q\}$。

用 SAM 维护 $s[1:j-1]$，若 $s[1:j-1]$ 中包含 $s[j:i+1]$，即加入第 $i+1$ 个字符仍然能复制，就不需要做任何处理。否则，重复地将第 j 个字符加入后缀自动机并 $j=j+1$，相应维护 $s[j:i+1]$ 在后缀自动机上新的匹配位置，直到 $s[j,i+1]\in s[1,j-1]$。

UPD 出题人的的程序跑了 600+ms，验题人的程序跑了 900+ms，也并没有特地优化时间……开这个时限本意就是卡掉非线性的做法，但是目测误伤了很多写线性做法的大佬，非常抱歉！

7 Meteor

题意为求分子分母均小于等于某个值 n 的第 k 大的最简分数。

首先考虑二分一个值 k，求出小于这个值的分子分母 $\le n$ 的最简分数个数，即为：

$$\begin{aligned}&\sum_{i=1}^{n}\sum_{j=1}^{\lfloor ki\rfloor}[(i,j)=1]\\=&\sum_{i=1}^{n}\sum_{j=1}^{\lfloor ki\rfloor}\sum_{d|i,d|j}\mu(d)\\=&\sum_{d}\mu(d)\sum_{i=1}^{\lfloor \frac{n}{d}\rfloor}\lfloor ki\rfloor\end{aligned}$$

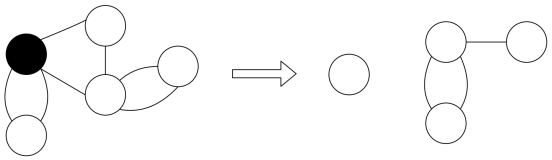
二分的时候写分数二分，二分 $O(\log n)$ 次就可以收敛到仅包含一个符合条件的分数的区间。最后在 Stern-Brocot Tree 上遍历，求出 $\ge$ 左端点 l 且满足条件的最小的符合条件的分数即为答案。

在分数二分之后，k 变成 $\frac{a}{b}$ 的形式，因此后半和式就可以用类欧几里得来算。

二分的部分复杂度为 $O(\sqrt{n}\log n\log C)$(C 代表二分出的分子分母的值)，然后在树上遍历求答案的复杂度为 $O(n)$(在所求答案为 $\frac{1}{n}$ 时可以取到上界)。

8 Desert

对于一个有根仙人掌，去掉根之后的每个连通块都会是许多有根仙人掌串起来的一个“序列”，且互为对称的两个序列被认为相同。例如：



从这个角度考虑，令 a_n 表示 n 个点的有根仙人掌数目，b_n 表示 n 个点的“仙人掌序列 (互为对称的两个序列视为不同)”的个数，c_n 表示共有 n 个点并由奇数个仙人掌组成的“回文仙人掌序列”的个数，d_n 表示共有 n 个点的“仙人掌序列 (互为对称的仙人掌序列视

为相同)”的个数，则 b,c,d 的求法容易得到：

$$\begin{cases} b_n = a_n + \sum\limits_{i=1}^{n-1} a_i b_{n-i} \\ c_n = a_n + \sum\limits_{i=1}^{\lfloor n/2 \rfloor} a_i c_{n-2i} \\ d_n = \dfrac{1}{2}\left(b_n + c_n + [2|n] b_{n/2}\right) \end{cases}$$

而按照前面的分析，a_n 就是由 $n-1$ 个点组成若干个“仙人掌序列 (互为对称的仙人掌序列视为相同)”的方案数。因此

$$\sum_n a_n x^n = x \prod_{i=1}^{\infty} \prod_{j=1}^{d_i} \sum_{k=0}^{\infty} x^{ik}$$

枚举 (i,j) 表示枚举一种序列，k 表示这种序列的个数。等式右边的乘积展开之后每项即为若干个 x^{ik} 的乘积，表示某种 i 个点的仙人掌序列选了 k 个。

对这个等式进行处理：

$$\begin{aligned} & \sum_n a_n x^n = x \prod_{i=1}^{\infty} \prod_{j=1}^{d_i} \sum_{k=0}^{\infty} x^{ik} \\ \Leftrightarrow & \sum_n a_n x^{n-1} = \prod_{i=1}^{\infty} \frac{1}{(1-x^i)^{d_i}} \\ \Leftrightarrow & \ln \sum_n a_n x^{n-1} = \sum_{i=1}^{\infty} \ln \frac{1}{(1-x^i)^{d_i}} = \sum_{i=1}^{\infty} d_i \sum_{j=1}^{\infty} \frac{x^{ij}}{j} \\ \Leftrightarrow & \frac{\mathrm{d}}{\mathrm{d}x} \ln \sum_n a_n x^{n-1} = \sum_{i=1}^{\infty} i d_i \sum_{j=1}^{\infty} x^{ij-1} \\ \Leftrightarrow & \sum_{n>1} (n-1) a_n x^{n-2} = \left(\sum_n a_n x^{n-1}\right) \left(\sum_{i=1}^{\infty} i d_i \sum_{j=1}^{\infty} x^{ij-1}\right) \\ \Leftrightarrow & \sum_{n>1} (n-1) a_n x^{n} = \left(\sum_n a_n x^{n}\right) \left(\sum_{i=1}^{\infty} i d_i \sum_{j=1}^{\infty} x^{ij}\right) \\ \Leftrightarrow & \sum_{n>1} (n-1) a_n x^{n} = \left(\sum_n a_n x^{n}\right) \left(\sum_{i=1}^{\infty} x^i \sum_{j|i} j d_j\right) \end{aligned}$$

令左右两边对应项系数相等即可以得到 a_n 的递推式：

$$a_n = [n=1] + [n>1] \frac{1}{n-1} \sum_{i=1}^{n-1} \left(\sum_{j|i} j d_j\right) a_{n-i}$$

分治 FFT 求出 a,b,c,d 即可。

9 String

一位位地构造答案字符串，每次贪心地加能加入的最小的字符 (判断能否加入只要判断加入之后原字符串剩下的后缀中的每种字符的数目能否足够满足条件)。

10 Kingdom

记前序序列为 $a[]$，中序序列为 $b[]$。首先考虑到，如果一个数既没有在 $a[]$ 中出现也没有在 $b[]$ 中出现，那么说明当所有出现过的数确定之后，它放在任何一个位置上都可以。

再来考虑出现过的数，显然如果树的形态确定了，那么这些数的位置一定确定了。定义 $f[x][l][r]$ 为 $a[x]$ 做根，对应的子树为 $b[]$ 中的 $[l,r]$ 区间，有多少不同的树的形态。如果一个数仅在 $a[]$ 或 $b[]$ 中出现过，那么它可以和 0 对应。否则，需要把它在两个序列中的位置对应。在区间 DP 划分区间的时候考虑这些数出现的合法性再转移即可。

最后，假如有 k 个数既没有出现在 $a[]$ 中也没有出现在 $b[]$ 中，那么答案即为 $f[1][1][n]\cdot k!$

另外，对于每个状态 (x,l,r)，可以推得这种状态中有多少个对位置可以随意确定，那么可以在转移的过程中乘一个组合数，相当于把空位可以填的数字分给两个子区间，这样可以直接求出最终的答案。

由于非法状态比较多，写记忆化搜索跑的比较快。

11 Function

$$\sum_{i=1}^{n}\gcd(\lfloor\sqrt[3]{i}\rfloor,i)$$
$$=\sum_{a=1}^{\lfloor\sqrt[3]{n}\rfloor}\sum_{i=1}^{n}\gcd(a,i)[\lfloor\sqrt[3]{i}\rfloor=a]$$

又有 $\lfloor\sqrt[3]{i}\rfloor=a\Leftrightarrow a\le\sqrt[3]{i}<a+1\Leftrightarrow a^3\le i\le(a+1)^3-1$ 因此

$$\sum_{i=1}^{n}\gcd(a,i)[\lfloor\sqrt[3]{i}\rfloor=a]$$
$$=\sum_{i=a^3}^{\min\{n,(a+1)^3-1\}}\gcd(a,i)$$

设 $r=\lfloor\sqrt[3]{n}\rfloor-1$，则带回原式子

$$\begin{aligned}&\sum_{i=1}^{n}\gcd(\lfloor\sqrt[3]{i}\rfloor,i)\\=&\sum_{i=\lfloor\sqrt[3]{n}\rfloor^3}^{n}\gcd(\lfloor\sqrt[3]{n}\rfloor,i)+\sum_{a=1}^{r}\sum_{i=a}^{(a+1)^3-1}\gcd(a,i)\end{aligned}$$

而

$$\begin{aligned}&\sum_{i=1}^{n}\gcd(a,i)\\=&\sum_{d}d\sum_{i=1}^{n}[\gcd(a,i)=d]\\=&\sum_{d}d\sum_{t|\frac{i}{d},t|\frac{a}{d}}\mu(t)\\=&\sum_{T|a}\lfloor\frac{n}{T}\rfloor\sum_{d|T}d\mu(\frac{T}{d})\end{aligned}$$

注意 $\sum_{d|T}d\mu(\frac{T}{d})=\varphi(T)$，则 $O(\sqrt[3]{n})$ 预处理 $\varphi(T)$ 之后，答案式子中的第一个和式可以 $O(\sqrt[6]{n})$ 地计算出来。第二个和式可以化为

$$\begin{aligned}&\sum_{a=1}^{r}\sum_{T|a}(\lfloor\frac{(a+1)^3-1}{T}\rfloor-\lfloor\frac{a^3-1}{T}\rfloor)\varphi(T)\\=&\sum_{T}\varphi(T)\sum_{b=1}^{\lfloor\frac{r}{T}\rfloor}(\lfloor\frac{(bT+1)^3-1}{T}\rfloor-\lfloor\frac{(bT)^3-1}{T}\rfloor)\end{aligned}$$

而

$$\begin{aligned}&\lfloor\frac{(bT+1)^3-1}{T}\rfloor-\lfloor\frac{(bT)^3-1}{T}\rfloor\\=&\lfloor b^3T^2+3b^2T+3b\rfloor-\lfloor b^3T^2-\frac{1}{T}\rfloor\\=&3Tb^2+3b+1\end{aligned}$$

因此

$$\begin{aligned}&\sum_{a=1}^{r}\sum_{i=a}^{(a+1)^3-1}\gcd(a,i)\\=&\sum_{T=1}^{r}\varphi(T)[(3T\sum_{b=1}^{\lfloor\frac{r}{T}\rfloor}b^2)+(3\sum_{b=1}^{\lfloor\frac{r}{T}\rfloor}b)+\lfloor\frac{r}{T}\rfloor]\end{aligned}$$

可以 $O(\sqrt[3]{n})$ 求出。总复杂度 $O(T\sqrt[3]{n})$。

n 比较小的情况可能会出问题，注意要适当特判。也有其它的算法，只要是线性应该都是能过的。把 $\sqrt[3]{n}$ 出到 10^7 是为了卡非线性做法。

12 Sequence

令 $b_i = \sum_{j=i-k\cdot x} a_j (0 \le x, 1 \le j \le i)$，则 $\sum b_i x^i = (\sum a_i x^i)(\sum x^{ik})$，因此操作的顺序没有影响，可以统计各类操作次数，然后批量处理，也就是 $\sum x^{ik}$ 的幂。

由于 n, m 比较大，直接做快速幂或者 $\ln + \exp$ 大概会 T，但是 $(\sum x^{ik})^n = \sum \binom{n-1+i}{i} x^{ik}$，因此可以省掉求幂，对于一种操作只做一次卷积即可。

13 Code

$d = 2$ 时，$f(\boldsymbol{x}) = \text{sign}(\boldsymbol{w^T} \cdot \boldsymbol{x}) = w_0 + w_1 x_1 + w_2 x_2$，$f(\boldsymbol{x}) = 0$ 对应于二维平面上的一条直线，直线一侧的点取值为 1，直线另一侧的取值为 -1。故该问题等价于能否找到一条直线将平面上的两类点分开，等价于判断这两类点分别组成的两个凸包是否相交。