多校 8 题解

August 14, 2018

1 Character Encoding

关键词 组合计数；容斥原理

1.1 题意

求 $\sum_{i=1}^{m} x_i = k\ (0 \le x_i < n)$ 的整数解的组数。

1.2 题解

当 x_i 无上界限制时，答案就是 $\binom{k+m-1}{m-1}$。考虑选择 p 个变量违反上界限制，记为 $F(p)$，这样的方案数等于 $\binom{m}{p}$ 乘以 m 个变量和为 $k-pn$ 的情况的解数，即

$$F(p) = \binom{m}{p}\binom{k-pn+m-1}{m-1}$$

最后令 $G(p) = F(p) - G(p+1)$，根据容斥原理，$G(0)$ 即为最终答案。

2 Pizza Hub

关键词 计算几何；分类讨论

2.1 题意

给定三角形和矩形的宽度，求矩形的最小高度，使得矩形能装下三角形。

2.2 题解

注意到总存在一个最优解，使得矩形的一个顶点和三角形的一个顶点重合。然后只要枚举所有可能的放置方法即可。

实现的时候需要特别注意边界条件的判断。

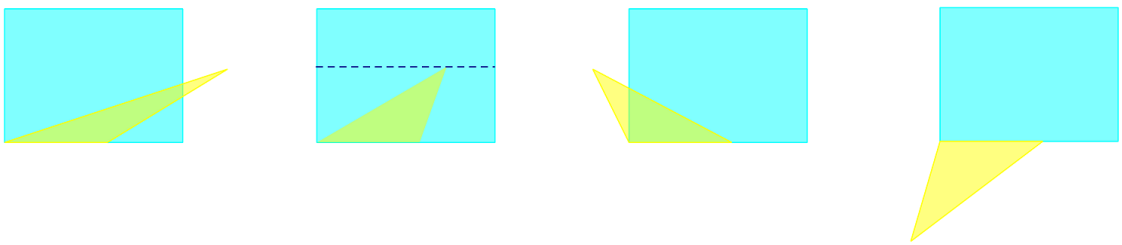

Figure 1: 第一类情况

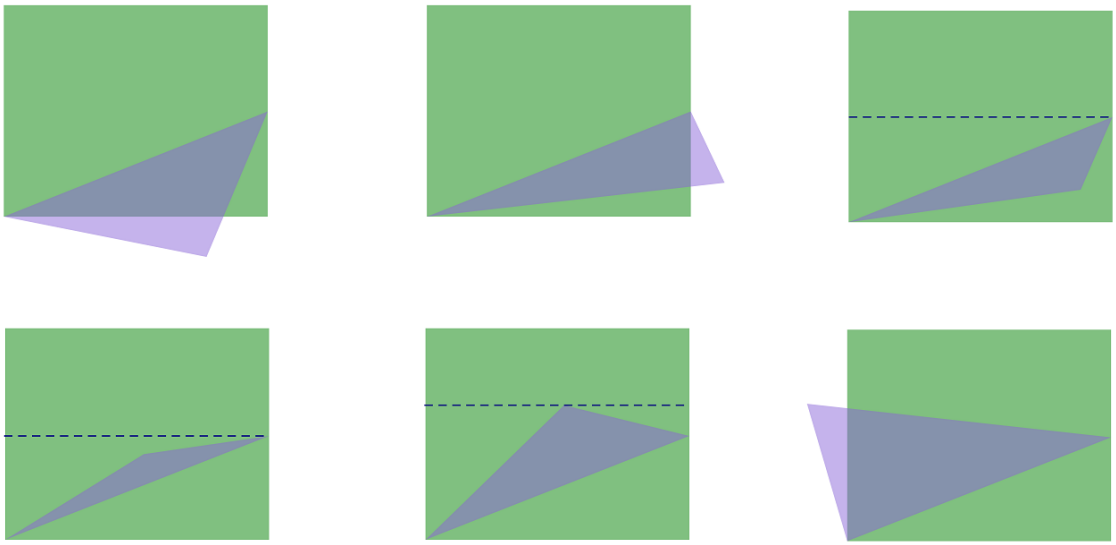

Figure 2: 第二类情况

3 City Development

关键词 线性代数；矩阵快速幂

3.1 题意

给定 $n_0 = n, n_1, n_2, \cdots, n_{k-1}, n_k = 1$，其中 n_i 为 n_{i+1} 的倍数，以及 $d_0(1), d_0(2), \cdots, d_0(n)$，以及以下转移方程

$$d_{t+1} = \sum_{i=1}^{n} \rho_{\mathrm{LCA}(x,i)} d_t(x)$$

其中 $\mathrm{LCA}(a,b)$ 定义为使 $\lceil a/n_i \rceil = \lceil b/n_i \rceil$ 的最大的 i。求转移 T 次后的结果。

3.2 题解

注意到要求的转移是线性变换，故答案总是可以写成矩阵快速幂的形式：$d_T = M^T d_0$。但是，直接转移由于转移矩阵的维数太高，无法使用矩阵快速幂进行加速。

由于变换是线性变换，我们可以对 d_0 的每一维的贡献，即每个城市对其他城市的最终贡献分别计算。注意到一共最多只有 $\log n$ 种不同的 n_i，从而最多只有 $\log n$ 种不同的 LCA 取值可能，从而我们可以用另一个矩阵 M' 表示转移，其中 M'_{ij} 表示与当前考虑城市 LCA 为 j 的一个城市，对与当前城市 LCA 为 i 的一个城市的贡献率。由对称性可知，M' 对于每个当前考虑的城市都是一样的，具体地可以写成这样：

$$M' = \begin{bmatrix} \rho_0 n_0 & \rho_1(n_1 - n_0) & \rho_2(n_2 - n_1) & \cdots \\ \rho_1 n_0 & \rho_0 n_0 + \rho_1(n_1 - 2n_0) & \rho_2(n_2 - n_1) & \cdots \\ \rho_2 n_0 & \rho_2(n_1 - n_2) & \rho_0 n_0 + \rho_1(n_1 - n_0) + \rho_2(n_2 - 2n_1) & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

这样，只需计算一次 M' 的快速幂，然后对于每个城市，将它带给其他城市的贡献计算出来（可用差分数组维护），最后合并即可。总时间复杂度为 $O(n \log n + \log^3 n \log T)$。

4 Parentheses Matrix

关键词 构造；分类讨论

4.1 题意

构造一个 $h \times w$ 的括号矩阵，使得匹配的行数、列数之和最大。

4.2 题解

当 h 和 w 中有一个是奇数时，构造的方法是显然的。下面仅讨论 h 和 w 都是偶数的情况。不妨设 $h \leq w$。

首先，第一行、最后一列中最多只有一个能够匹配，第一列、最后一行也只有一个能够匹配（考虑右上角和左下角的括号选取方法），故答案不会超过 $w+h-2$。

当 $h=2$ 时，每一列可以构造一对匹配，这样答案已经到达上界。

当 $h=4$ 时，可以如下构造：

```
((((((
)))(((
((()))
))))))
```

这样答案是 $w+h-3$。若存在更优的答案，则必有一个边界能够匹配，不妨设是第一列。这样，就已经有除第一行以外的两行（右括号开头的行）不匹配了，而第一行和最后一列中至少有一个不匹配，因此最优解不会超过 $w+h-3$。

当 $h \geq 6$ 时，可以如下构造：

```
((((((((
()()()()
(()()())
()()()()
(()()())
))))))))
```

答案是 $w+h-4$。同理可证明不存在更优的方法。

5 Magic Square

关键词 模拟

5.1 题意

给一个 3×3 的数阵，每次旋转一个 2×2 的子阵，求最终结果。

5.2 题解

直接模拟即可。

6 Boolean 3-Array

关键词 Burnside 引理；整数划分；线性基

6.1 题意

求在面对换、面翻转操作下本质不同的大小为 $m \times n \times p$ 的三维 01 数组的个数。

6.2 题解

本题改编自Project Euler 626。虽然维数变高了，但方法是类似的。

由于数据范围不大，可以打表预处理答案。

令所有 $m\times n\times p$ 的三维 01 数组的集合为 M，并令 M^g 表示置换操作 g 作用在 M 上的不动点的集合。考虑使用 Burnside 引理，其中置换群可以写成三个维度上的对称群 (S_m,S_n,S_p) 和翻转操作构成的群 (Z_2^m,Z_2^n,Z_2^p) 的直积：$G\cong S_m\times S_n\times S_p\times Z_2^m\times Z_2^n\times Z_2^p$，$|G|=2^{mnp}m!n!p!$。

$$|M/G|=\frac{1}{|G|}\sum_{g\in G}|M^g|$$

本题的难点是，如何对某个置换 g，求出其不动点的数量 $|M^g|$。令 $g=\sigma_x\sigma_y\sigma_z f_x f_y f_z$，并对所有轮换模式进行分类

$$\begin{aligned}\sum_{g\in G}|M^g|&=\sum_{\sigma_x\in S_m}\sum_{\sigma_y\in S_n}\sum_{\sigma_z\in S_p}\sum_{f_x\in Z_2^m}\sum_{f_y\in Z_2^n}\sum_{f_z\in Z_2^p}|M^{\sigma_x\sigma_y\sigma_z f_x f_y f_z}|\\&=\sum_{\mathbf{c}}\sum_{\mathbf{d}}\sum_{\mathbf{e}}\#(\mathbf{c})\#(\mathbf{d})\#(\mathbf{e})\sum_{f_x\in Z_2^m}\sum_{f_y\in Z_2^n}\sum_{f_z\in Z_2^p}|M^{\mathbf{cde}f_x f_y f_z}|\end{aligned}$$

其中 $\mathbf{c},\mathbf{d},\mathbf{e}$ 分别表示 S_m,S_n,S_p 中的轮换模式，$\#(\mathbf{c})$ 表示属于轮换模式 $\mathbf{c}$ 的置换的数量，这可以由 Cauchy 公式求得：

$$\#(\mathbf{c})=\frac{m!}{\prod_l L_l!l^{L_l}}$$

其中 $L_l=|\{c_i\in\mathbf{c}:|c_i|=l\}|$，即长度为 l 的轮换个数。

由于 $\sum_{c_i\in\mathbf{c}}|c_i|=m$，$\sum_{d_i\in\mathbf{d}}|d_i|=n$，$\sum_{e_i\in\mathbf{e}}|e_i|=p$，我们可以枚举所有 m,n,p 的整数划分以得到所有可能的轮换模式。

注意到 $\mathbf{cde}$ 将整个数组划分成了 $|\mathbf{c}|\times|\mathbf{d}|\times|\mathbf{e}|$ 块。设其中一块的大小为 $|c_i|\times|d_j|\times|e_k|$，则这一块在 $c_i d_j e_k$ 的作用下共形成 $\frac{|c_i||d_j||e_k|}{\mathrm{lcm}(|c_i|,|d_j|,|e_k|)}$ 个循环（每个循环长度为 $\mathrm{lcm}(|c_i|,|d_j|,|e_k|)$），每个循环可以独立地填上 0,1 两者之一，共有 $2^{\frac{|c_i||d_j||e_k|}{\mathrm{lcm}(|c_i|,|d_j|,|e_k|)}}$ 种填法，故上式可以进步写为

$$\sum_{g\in G}|M^g|=\sum_{\mathbf{c}}\sum_{\mathbf{d}}\sum_{\mathbf{e}}\#(\mathbf{c})\#(\mathbf{d})\#(\mathbf{e})\#_{\mathbf{cde}}(f_x,f_y,f_z)\prod_{c_i\in\mathbf{c}}\prod_{d_j\in\mathbf{d}}\prod_{e_k\in\mathbf{e}}2^{\frac{|c_i||d_j||e_k|}{\mathrm{lcm}(|c_i|,|d_j|,|e_k|)}}$$

其中 $\#_{\mathbf{cde}}(f_x,f_y,f_z)$ 表示在轮换模式 $\mathbf{cde}$ 下允许的翻转操作的数量。我们用变量 x_i,y_j,z_k 表示 $A[i][\cdot][\cdot]$，$A[\cdot][j][\cdot]$，$A[\cdot][\cdot][k]$ 翻转与否。在大小为 $|c_i|\times|d_j|\times|e_k|$ 子块中，每个循环经历的翻转操作的数量应当为偶数个，这样才能保证一个元素经过该循环后仍然等于原来的值：

$$\sum_{l\in c_i}x_l\frac{\mathrm{lcm}(|c_i|,|d_j|,|e_k|)}{|c_i|}+\sum_{l\in d_j}y_l\frac{\mathrm{lcm}(|c_i|,|d_j|,|e_k|)}{|d_j|}+\sum_{l\in e_k}z_l\frac{\mathrm{lcm}(|c_i|,|d_j|,|e_k|)}{|e_k|}\equiv 0\pmod 2$$

这样我们可以构造出关于 x_i,y_j,d_k 的模 2 意义下的线性齐次方程组，每个方程表示一个子块对翻转操作的约束：

$$\begin{bmatrix}\mathbf{M}_x & \mathbf{M}_y & \mathbf{M}_z\end{bmatrix}\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\\ \mathbf{z}\end{bmatrix}\equiv\mathbf{0}\pmod 2$$

其中对于 $c_i d_j e_k$ 这一块所代表的方程，系数为

$$\begin{aligned}(\mathbf{M}_x)_{c_i d_j e_k,l}&=[l\in c_i]\frac{\mathrm{lcm}(|c_i|,|d_j|,|e_k|)}{|c_i|}\\(\mathbf{M}_y)_{c_i d_j e_k,l}&=[l\in d_j]\frac{\mathrm{lcm}(|c_i|,|d_j|,|e_k|)}{|d_j|}\\(\mathbf{M}_z)_{c_i d_j e_k,l}&=[l\in e_k]\frac{\mathrm{lcm}(|c_i|,|d_j|,|e_k|)}{|e_k|}\end{aligned}$$

因此，求出 $\mathbf{M} = \begin{bmatrix}\mathbf{M}_x & \mathbf{M}_y & \mathbf{M}_z\end{bmatrix}$ 的秩 $\mathrm{rank}(\mathbf{M})$，允许的翻转操作的数量就是

$$\#_{\mathbf{cde}}(f_x, f_y, f_z) = 2^{mnp-\mathrm{rank}(\mathbf{M})}$$

这用线性基很容易求出。

因此，总的做法是：通过整数划分枚举出所有可能的轮换模式，然后对于每种轮换模式可以用线性基求出允许的翻转操作的数量，最后用 Burnside 引理求出答案即可。

7 Card Game

关键词 图论；树；基环树；dfs；树上 dp

7.1 题意

给定 n 张卡片，每张卡片正反面各有一个数。问至少要翻转多少张卡片，才能使正面向上的数互不相同，并求方案数。

7.2 题解

首先建图：每个数字为一个节点，每张卡片反面数字向正面数字连一条有向边。问题转化为：至少要反转多少条边的方向，才能使得每个点的入度不会超过 1。我们对每个弱连通分量分别处理。易知，当底图是树或基环树时，才可能有解。对于基环树，先把环找出来，然后将环上的边的方向统一一下；非环边的方向则是唯一确定的，从环上的点向外做一遍 dfs 即可。对于树，可以正反两次 dfs 处理出每个点作为根时所需要的反向次数，并统计出最小值以及方案数。最后将答案合并即可。

8 K-Similar Strings

关键词 分类讨论

8.1 题意

定义非空字符串上的 k-相似关系：

- 每个非空字符串和自己是 k-相似的；
- 对于两个长度和不超过 k 的非空串 A, B，如果 $A \circ B$ 和 $B \circ A$ k-相似，则 A 和 B k-相似；
- 如果 S, T 是 k-相似的，那么 $P \circ S \circ Q$ 和 $P \circ T \circ Q$ k-相似；
- 如果 S 和 U, U 和 T 相似，那么 S 和 T k-相似。

给定 k, A, B，判断 A, B 是否 k-相似。

8.2 题解

这题又是一道分类讨论题，虽然最后的算法非常简单，但中间思考的过程还是比较困难的。

首先可以看出 k-相似关系是等价关系。另外可以看出 A, B 是 k-相似的一个必要条件是 A, B 的第一个和最后一个字符分别相等。这些结论可以很容易地用归纳法证明。

注意到只有第二条规则涉及到 k。记直接由第二条规则得到的相似关系的集合为 S_k，然后我们只考虑另外 3 条规则。第 3 条规则是说将任一子串替换成另一相似子串，得到的串是相似的。将 S_k 中的相似关系视为替换规则，结合第四条规则，k-相似可以理解为，两个串是 k-相似的当且仅当其中一个可由若干次根据 S_k 中的替换规则进行子串替换得到另一个。这样要解决的问题就是研究清楚 S_k 中到底有哪些规则。我们根据 k 的取值，结合 k-相似的必要条件分类讨论。

当 $k = 1$ 时，无法用第二条规则生成任何替换规则，故直接判断两个串是否相等就行了。

当 $k = 2$ 时，唯一可以由第二条规则生成的替换规则是 $\mathsf{a} \to \mathsf{a}$ 型的，但这种规则没有任何作用，故同样直接判断两个串是否相等即可。

当 $k=3$ 时，除 S_2 中的替换规则外，可以生成新的规则有 $\mathtt{a}\to\mathtt{aa}$ 和 $\mathtt{aa}\to\mathtt{a}$，所以将两个串中连续重复的字母只保留一个，然后判断是否相等即可。

当 $k=4$ 时，除 S_3 中的替换规则外，可能可以生成的规则有 $\mathtt{a}\to\mathtt{aba}$，$\mathtt{aa}\to\mathtt{aa}$ 和 $\mathtt{aba}\to\mathtt{a}$ 三种，其中只有第一种和最后一种有用。这两种规则事实上都是可以用 S_3 中的规则生成的：$\mathtt{aaba}\to\mathtt{aba}\to\mathtt{abaa}$。判断两个串是否是 4-相似，首先还是将两个串中连续重复的字符只保留一个，然后对每个串进行如下处理：

> 维护一个栈，若当前字符和栈中倒数第二个字符相等，则弹出栈顶元素；否则将当前字符压入栈中。最终栈中的内容即为处理结束后的内容。

如果处理后的结果相同，那么这两个串即为 4-相似的。此外可以证明，任意两个 4-相似的串经过上述处理的结果一定是一样的；这只需要对每个 S_4 中的替换规则，证明应用该规则后不改变处理的结果即可，这可以手动验证或者写个程序枚举所有情况。

当 $k=5$ 时，除 S_4 中的替换规则外，我们讨论这两个可能生成的规则：$\mathtt{ab}\to\mathtt{acb}$，$\mathtt{acb}\to\mathtt{ab}$。这两个规则都是可以用 S_4 中的规则生成的：$\mathtt{abacb}\to\mathtt{acb}\to\mathtt{acbab}$。反复利用这两个规则可以把字符串化简到只剩首尾两个字符，故判断首尾两个字符是否分别相等即可。也就是说，此时 k-相似的必要条件已经成为了充要条件。

当 $k>5$ 时，不会有更多有用的替换规则，因此判断方法同 $k=5$ 的情况。

9 Make ZYB Happy

关键词 SAM

9.1 题意

给定若干个字符串，每个字符串有一个快乐值。随机选取一个长度不超过 q 的串，问是给定字符串中父串的快乐值的乘积的期望。

9.2 题解

题目就是要我们要把每个串的本质不同子串找出来，把他们的快乐值都乘上 $h[i]$，然后再把每种长度 i 的串的快乐值都统计到 $ans[i]$ 里面。

那么问题就变成了有多个串，需要对每个串的本质不同子串进行更新，用一个广义 SAM 即可。

广义 SAM 的一种操作是：把每个串都从 SAM 的根节点开始，完整插入到 SAM 中即可。

本质不同子串操作是：考虑串 i，把串 i 所在的所有 SAM 节点都加 $a[i]$ 即可，可以设置一个访问标记，每个点更新 parent 链，每个节点只被更新一次。

最后来一个前缀和线性处理一下答案即可。

由于 SAM 的构造和本质不同串的更新都是线性的，答案也可以线性求得，回答询问是 $O(1)$ 的，所以整个算法是 $O(|\Sigma|(n+q))$ 的。

10 Taotao Picks Apples

关键词 预处理；动态规划；二分查找；ST 表

10.1 题意

给一个序列，每次贪心选取比前一个数大的数。每次询问修改一个数，求修改后的序列的能选出多少个数。询问不叠加。

10.2 题解

考虑每次修改不叠加，因此我们可以从如何对原序列进行预处理着手。

通过观察可以发现，将原序列从任意位置断开，我们可以通过分别维护左右段的某些信息来拼接得到答案。

对于左段来说：

- 需要知道最大值的位置，以及到达最大值需要几步；
- 使用 ST 维护信息。

对于右段来说：

- 由于前半部分的信息未知，因此我们需要维护从每个位置开始到结尾可以走几步；
- 从后往前做 DP，每次找出右边第一个比自己大的数，答案就是它的 DP 值 +1。

对于每次询问：

- 考虑这个数左边的部分加上这个数之后的答案和最大值；
- 再找到右边第一个大于左半部分最大值的数，答案相加即可。

预处理使用 ST 表，每次查询需要一个二分，总复杂度 $O(n\log n+q\log n)$。

11 Pop the Balloons

关键词 状态压缩动态规划；剪枝

11.1 题意

给定一个气球矩阵，扎掉一个气球后，同行同列的气球都消失。问对于每个 $1\le x\le k$，扎恰好 x 次能够清除所有气球的方案数。

11.2 题解

显然扎掉的气球两两不同行且不同列。只需要求出扎 x 个气球的集合，然后乘以 $x!$ 即可。

由于行数较少，考虑枚举扎掉的行集合，设扎掉的行的 bitmap 为 `mask1`。设 `dp[r][mask2]`为考虑前 r 列，已经扎掉的气球所在行为 `mask2`的方案数。考虑状态 `dp[r][mask2]`的转移：

- 如果第 $i+1$ 列的气球被 `mask1`包含，则 `dp[r+1][mask2] += dp[r][mask2]`；
- 对于第 $i+1$ 列的每个在 `mask1`中，但不在 `mask2`中的气球 w，我们可以将它扎掉，即为 `dp[r+1][mask2|w] += dp[r][mask2]`；

此外，dp 的过程中会产生大量值为 0 的无效状态，这些状态可以直接不转移。

设行的集合为 S，则总复杂度为 $n\times\sum_{U\subset S}\sum_{V\subset U}|V|=O(nm3^m)$，实际运行常数极小且跑不满。

12 From ICPC to ACM

关键词 贪心；模拟

12.1 题意

告诉你工厂每个月的原料价格，客户需求，产能，生产成本，原料和产品的仓储成本，产品的仓库容量限制，求满足客户需求前提下的最小成本。

12.2 题解

由于原料的购买量和仓储量不限，我们可以很容易地处理出当前的最低原料成本：$c'_i=\min_j\{c_j+\sum_{k=j}^{i-1}R_k\}$。然后，我们维护一个产品仓库。每次按照产量将产品生产出来放入仓库中，然后选取综合成本最低的商品卖掉；跨月时，按照仓库容量丢弃成本最高的产品。

容易证明这样贪心是正确的。跨月丢弃成本最高的产品的正确性显然，只需要讨论销售当前成本最低的商品的正确性。设有两个商品，成本分别是 a 和 $a+p$，假设当前先选择了后者，如果前者最终没有被选到，那么当前改选前者显然更优；如果前者最终也被选到了，这种情况当前先选前者，给之后带来的额外代价不会超过 p，而先选前者已经节约了 p 的成本，因此结果一定也不会更差。