牛客暑期ACM多校训练营

第 8 场-FizzyDavid

牛客网
NOWCODER

难度预测

本场比赛共11题。赛前命题人将题目分成三类：

Easy --- C E G
Medium --- B H I J
Hard --- A D E K

IC - Touring Cities

关键词： 结论 构造 ~~(插头dp)~~

给定一个n行m列的棋盘，将格子视为点，相邻的格子对应的点间连一条边。再此基础上新加k条边，形成一个n*m个点的无向图。问经过每个点至少一次的环路长度最短是多少。(n,m>1)

IC - Touring Cities

关键词： 结论 构造

由于答案最多为n*m+1，可以将题目转换为判断是否存在哈密尔顿回路。

如果n和m有一个是偶数，那么必定存在哈密尔顿回路。

如果n和m都是奇数：
将棋盘黑白染色，规定左上角的格子是黑色。那么存在哈密尔顿回路的条件是当且仅当新增的边连接了两个不同的黑色格子。

IC - Touring Cities

关键词： 结论 构造

考虑证明：
若没有连接两个不同黑格子的边，那么回路上一个黑格子后必然接一个白格子。那么回路的黑格子数目一定小于等于白格子数目。但是棋盘上的黑格子数目比白格子多一。矛盾。

若存在一个连接两个不同黑格子的边，那么一定存在一条合法的方案。
两个思路：
1）按照起点终点位置分类讨论构造
2）归纳证明
若有一维大于5，一定可以将其减2
归纳到n,m<=5的情况，此时可以自行验证

IC - Touring Cities

关键词： 结论 构造

考虑证明：
若没有连接两个不同黑格子的边，那么回路上一个黑格子后必然接一个白格子。那么回路的黑格子数目一定小于等于白格子数目。但是棋盘上的黑格子数目比白格子多一。矛盾。

若存在一个连接两个不同黑格子的边，那么一定存在一条合法的方案。
两个思路：
1）按照起点终点位置分类讨论构造
2）归纳证明
若有一维大于5，一定可以将其减2
归纳到n,m<=5的情况，此时可以自行验证

E - Counting Paths

关键词： 树形dp 计数

本题有两种解法。

标算作法：
题目要统计的是对于所有连通块，有多少个路径集合满足条件。
考虑统计对于所有路经集合，有多少个连通块满足条件。
如果先将路径集合的路径染成红色，那么满足条件的连通块就是未被染色的连通块。
未被染色的联通块数量等于=未被染色的点数-两个端点都未被染色的边数
（这是因为每个连通块都是一棵树，一棵树的点数比边数恰好多1，那么每个连通块都会使总点数比总边数多1）
那么我们只要统计对于每个点，不含这个点的路径集合数量，以及对于每条边，不含它的两个端点的路径集合数量就行了。最终答案就是前者的和减后者的和。

IC - Counting Paths

关键词： 树形dp 计数

本题有两种解法。

另一种解法：
一个连通块的贡献只要知道它的所有叶子节点以及连通块的根就行了。（这里的根指有根树中连通块深度最浅的节点）
对于每个点 计算出它的子树中不包含它但是包含所有儿子的路径集合数量，记为cnt(x)。那么连通块的贡献就是所有叶子的cnt的积，再乘上根的贡献。
根的贡献就是除了子树外包含它的父亲的路径集合数量。
贡献可以用容斥算。

只要树形dp维护根为某个节点的所有连通块的叶子贡献的积的和就可以算答案了。

两种解法时间复杂度都可以做到O(n)（如果线性预处理2^(n*(n-1)/2)）

G - Counting regions

关键词： 计数

题意：给你一个正n边形，将n个顶点两两连边，问内部有多少个区域。n是奇数。

欧拉公式：F=E-V+2

内部交点个数：C(n,4)

一条线段会被一个交点分成两段，所以x条直线的交点会多分出来x条线段，利用V可以算出E。

B - Filling pools

关键词： 动态规划 FFT

题意：给定一个n*n的棋盘，问你有多少种每一行每一列染黑一个格子的方法，使得最后棋盘所有格子都变黑。每一时刻一个格子相邻两个中至少有两个是黑的就会变黑。

发现一个合法的方案要么n=1，要么存在n=n1+n2，使得左上角是一个n1*n1的合法方案且右下角是n2*n2的合法方案，或左下角是一个n1*n1的合法方案且右上角是n2*n2的合法方案，将他们称为A型划分和B型划分。

每个方案要么属于A型划分，要么属于B型划分。并且A型划分与B型划分一一对应。

设F(n)为n*n的原问题的方案数，G(n)为n*n的A型（或B型）划分的方案数，那么有 $G(n) = \frac{F(n)+[n==1]}{2}$

B - Filling pools

关键词： 动态规划 FFT

对于一个方案可能有多种划分方法n=n1+n2，我们将n1最小的那个称为最小化分方法。这时，n1*n1的方案需要满足，n1=1或n1*n1的划分类型和n*n的划分类型不一样。

我们枚举最小的n1，可以得到递推式：$G(n) = \sum_{i=1}^{n-1} G(i)F(n-i) \ (n > 1)$

由于这是卷积形式，可以用分治FFT优化。时间复杂度O(n log^2 n)

也可以用生成函数推出一个多项式开方的形式。O(n log n)

H - Playing games

关键词： FWT

令 $x = \oplus_{i=1}^{n} a_i$，题目可转化为删去最少的数使得删去的数的 $\oplus$ 为 x。我们令 $dp_{i,x}$ 表示删去 i 个数 $\oplus$ 能否为 x，发现答案上届是 $O(log\ x)$ 的，所以状态数是 $O(x\ log\ x)$ 的，转移使用 fwt 优化即可。

时间复杂度O(x log^2 x)。可以通过本题。

其实也可以做到O(x log x)。但是由于两个log作法常数较小，并且如果要卡的话输入会成为瓶颈，于是良心出题人就没有卡。

II - Permuting cows

关键词： 贪心 数据结构 分类讨论

如果所有数字都相同，则答案为 $1, 2, 3, \ldots, n$。

否则我们另 $b_{i,j}$ 为 $a_i \oplus a_j$ 的二进制下最高位。我们找到最大的 $b_{x,y}$，并将所有数分成两个集合 $S_1 = \{a_i \mid a_i \otimes b_{x,y} > 0\}$, $S_2 = \{a_i \mid a_i \otimes b_{x,y} = 0\}$。易发现答案 $ans = Min_{x \in S_1, y \in S_2}\{x \oplus y\}$。我们将所有 $a_i \oplus a_j = ans$ 的连边，可以得到一个二分图，两边点集分别为 S_1 和 S_2 中的 i，之后贪心选取，每次选完使得仍然存在一个路径 P 满足 P 包含所有未选择的点，并且对于 P 中相邻的两个点 (u, v) 满足要么 u, v 有边相连，要么 u, v 在二分图一侧，使用数据结构维护即可。

时间复杂度O(n log n)。注意常数。

J – Calculating sums

关键词： 贪心 数据结构 分类讨论

先将 N 变为偶数，使得答案不变。另 $F(x) = \frac{(x+1)^{N+2}-1}{(x+1)^2-1}$，则所求即为 $\sum_{i=0}^{M} [i = 0 \bmod 2] [x^i]F(x)$。

接下来考虑如何做除法，另 $a_i = [x^i](x+1)^{N+2}$, $b_i = [x^i]F(x)$ 则有 $b_0 = \frac{a_1}{2}$, $b_i = \frac{a_i - b_{i-1}}{2}$ $(i > 0)$。另 $K = s\,2^t$, $s = 1 \bmod 2$，如果分别求出 ans 对 s, 2^t 取模的值就可以通过 CRT 合并来解决。

- 由于 $gcd(s, 2) = 1$，所以在模 s 意义下存在 2 的逆元 2^{-1}，所以 $b_0 = a_1\,2^{-1}$, $b_i = (a_i - b_{i-1})\,2^{-1}$ $(i > 0)$。
- 可以发现 $b_i = \sum_{j \geq i+2} (-2)^{j-(i+2)}\,a_j$，当 $j - (i+2) \geq t$ 时 $(-2)^{j-(i+2)}\,a_j = 0 \bmod 2^t$，所以只用计算 $j < t + i + 2$ 即可。

总时间复杂度 $O(M\ log\ K)$。

A - Connecting segments

关键词： 计算几何 数据结构

题意：给定N个向量，每个向量有一个种类，问你每个种类选最多一个，并且总共选至多K个向量的和的模长最长是多少。(N<=1000)

如果我们能在平面上找到所有答案的候选点，那么我们只需要关心这些点组成的凸包就好了。

那么对于凸包上的点，一定存在一个单位向量V，使得V与它的点积是所有候选点中最大的。

那么我们可以考虑枚举所有单位向量V，并找出与V点积最大的方案来更新答案。我们将V成为方向向量。

A - Connecting segments

关键词： 计算几何 数据结构

先考虑没有类别限制的情况。由于点积有性质：$(A+B)\cdot V = A\cdot V + B\cdot V$

记a_i为第i个向量，当前方向向量为V，记第i个向量与方向的点积为$v_i = V\cdot a_i$。那么若要选择k个向量，由于我们要使它们的和与方向向量点积最大，也就是让他们与方向向量点积的和最大，所以一定是选v_i最大的k个向量。

我们考虑类似扫描线的方法扫描单位向量。由于选的向量之和他们点积的相对大小关系有关，所以我们可以只要考虑点积相对大小关系改变的时候，也就是说可以只考虑一些关键的单位向量，而不是所有单位向量。

不难发现两个向量相对位置改变的只有两个关键单位向量，所以关键单位向量的个数O(n^2)级别的。

A - Connecting segments

关键词： 计算几何 数据结构

如果我们用一个序列从大到小维护点积的相对大小关系，那么扫描线到每个关键向量时只会交换序列中相邻的两个元素。

考虑如何更新答案，记选k个向量的答案为ANS[k]。一次暴力的更新是我们每次用序列中前k个向量来更新ANS[k]。我们记录序列前k个向量的和sum[k]，交换序列中相邻两个元素时只要更新那两个位置的sum和ANS就好了。

这样复杂度就是O(n^2 log n)了。

A - Connecting segments

关键词： 计算几何 数据结构

考虑如何加上组别的限制，在序列中每个组别只有第一个能记录答案，那么每次交换相邻两个元素时，若一个组别内的第一个元素发生了变化，那我们就暴力整个更新一遍sum和ANS。

考虑发生变化的次数只有O(n)次，所以这部分复杂度是O(n^2)的，不会对总复杂度产生影响。

最终复杂度O(n^2 log n)。

需要注意一些细节的实现。

ID – Compressing data

关键词： 字符串

题意：给你一个字符串，问你如何使用最小代价将其划分成几个循环串。每段的代价为B+A*循环节长度 。

aabcabc
aabcabcbcabc
aabcabcbcabzbcabz
无法使用贪心，需要考虑所有循环串的转移。

记dp[x]为只考虑前x个字符的最小代价。我们需要快速转移这个dp。

先考虑转移循环次数为1的子串。
维护前缀min即可O(1)转移。

$$dp_x = min(dp_i + A(x-i) + B) \ \ (0 \le i < x)$$
$$dp_x = min(dp_i - Ai) + Ax + B \ \ (0 \le i < x)$$

ID – Compressing data

关键词： 字符串

考虑转移循环次数至少为2的，枚举它的循环节为L。
定义一个S[1…n]有一个period p(1<=p<=n), 当且仅当对于所有i(1<=i<=n-p)满足S[i]=S[i+p]。

我们找出所有所有满足有一个period为L且长度大于等于2L的极长子串，它们可以用一些不相交的区间表示。我们对每个区间分别处理。设当前处理区间为[l,r]。
找出这些区间可以设L,2L,3L…kL为关键点。由于长度大于等于2L的区间一定覆盖至少两个关键点，所以我们对于相邻的两个关键点向前向后求lcp即可找到包含这两个关键点的极长区间。

关键点个数为n/1+n/2+n/3…=O(n ln n)
用后缀数组及st表预处理O(n log n)，O(1)回答lcp。这部分复杂度为O(n log n)。

ID – Compressing data

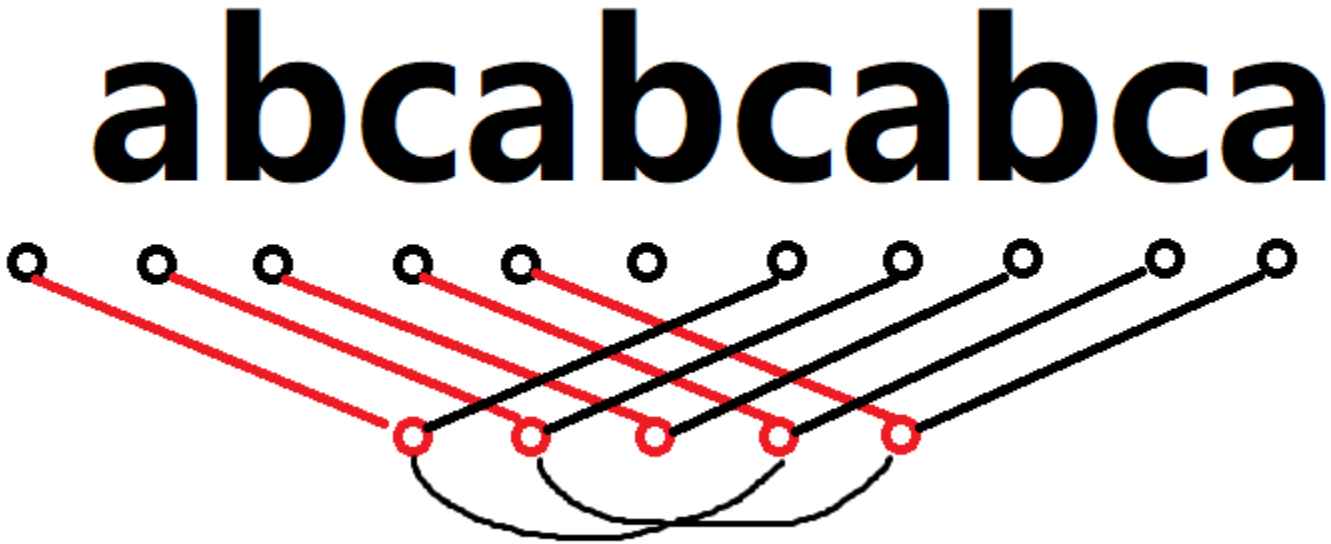
关键词： 字符串

这个区间内所有长度是L的倍数的子串都是一个循环串，并且它们的代价都是一样的。

我们考虑新建一些状态来辅助转移，记为f。
dp(x)->f(x+L) 代价为A*L+B (l-1<=x<=r-2L)
f(x)->f(x+L) 代价为0 (l+L-1<=x<=r-2L)
f(x)->dp(x+L) 代价为0 (l+L-1<=x<=r-L)
注意每个区间新建的f是不共享的。我们新建了r-l+1-2L+1个新状态。
设s= r-l+1-2L+1，那么我们新建了O(s)个状态与O(s)条边。

ID – Compressing data

关键词： 字符串



ID – Compressing data

关键词： 字符串

发现当字符串为aaa....a时，s的总和到达O(n^2)。
考虑优化，当一个极长区间的最小period不为L时，我们便不新建状态。这是因为如果确定好一个划分，那么每段里循环节长度一定是越短越好。

考虑计算此时s的总和。我们发现对于一个极长区间[l,r]，它的s相当于区间内最小循环节长度恰好为一半的字符串数量。那么s的总和相当于字符串中满足循环节长度恰好为一半的子串数量。

可以证明一个字符串中满足循环节长度恰好为一半的子串数量最多O(n log n)个。然而这个数量更像是线性的，实际上出题人还没有找到这个子串数量大于等于字符串长度的字符串。目前出题人找到的最大的就是aaa...a。它有n-1个这样的子串。
Bonus：大家可以尝试证明个数是线性的，或找到一个具有大于线性个数的满足要求的子串的字符串。

ID – Compressing data

关键词： 字符串

对于所有左端点相同的子串，我们考虑那些满足条件的子串，并且列出他们的循环节长度$l_i, l_2 \dots l_k$，并且将其从小到大排好序。
一个字符串若存在两个period p_1, p_2，那么必然存在一个长度为$gcd(p_1, p_2)$的period $(p_1, p_2 \leq |S|/2)$
而对于一个前缀$s[1 \dots len]$，如果存在一个长度为p的period，那么$l_i (l_i \leq len)$中不会出现大于p的倍数，不然就不满足条件了。
我们从后往前考虑l_i及$s[1 \dots 2l_i]$的period，首先对于$s[1 \dots 2l_k]$肯定存在一个长度为l_k的period，设我们知道的$s[1 \dots 2l_i]$的period长度为p，我们先将p设为l_k
当我们考虑到$i (i < k)$时，有下面几种情况：
1. $2l_i < p$ 将p设为l_i
2. $l_i \neq p$ 将p设为$gcd(p, l_i)$ 此时由于l_i不可能是p的倍数，所以$2gcd(p, l_i) \leq p$
3. $l_i = p$

注意到情况1、2中p都减少了至少一半，而当$p = 1$时则不可能有新的l_i了，情况1、2只会出现$O(\log n)$次。
那么不同的p只有$O(\log n)$个，所以第三种情况只会出现$O(\log n)$次。而每个$l_i (i < k)$都对应着某一种情况，所以k也只有$O(\log n)$了。

ID – Compressing data

关键词： 字符串

建出所有额外状态以及转移边后，这个图是一个拓扑图。（所有边都是从左边指向右边）可以O(|V|+|E|)从左到右依次转移。

最终复杂度O(n log n)

E – Protecting lawn

关键词： 模拟 数据结构

题意：有一个长度为L的草坪，你需要维护四种操作：放置坚果墙、放置地刺、出现一只僵尸、询问僵尸的位置。所有在地刺上的僵尸会受到一个单位的持续伤害，坚果墙会阻止僵尸前进，并在总共被吃了dur秒后消失。正在吃坚果墙的僵尸也会受到下一格的地刺的伤害。

注意我们为了避免浮点数运算，坚果墙只有被吃和没被吃两种状态，有多只僵尸同时吃一个坚果墙也不会加速坚果墙的消失。

注意我们为了简化问题，保证一个位置只会被种一次植物。

不强制在线。

IE – Protecting lawn

关键词： 模拟 数据结构

一只僵尸要么在走路，要么停下来了。那我们可以用一个动作序列来表示一只僵尸所有时刻的位置。（即维护从什么时候开始走了，从什么时候停下来了）

如果两只僵尸位置重合了，那么他们直到死之前都会在同一个位置，即他们重合后共用一个动作序列。那么我们只要知道共用的位置序列和一只僵尸的死亡时间，就可以找出一只僵尸在某个时刻的位置。

这就启发我们对一个位置上的所有僵尸同时维护。我们可以用可并堆维护在同一个位置的僵尸的所有血量，可以支持整个堆加减、弹出最小血量的僵尸、合并两堆。

E – Protecting lawn

关键词： 模拟 数据结构

考虑如何算伤害。我们需要计算某个僵尸从某个时刻开始向左走t秒受到的伤害，这其实是一个二维点查询。可以用主席树在线一个log回答询问。一个僵尸停止不动收到的伤害只要知道当前位置地刺的出现时间就可以维护。

我们可以用一个平衡树（std::set）按时刻维护几种事件，来模拟发生过程：
1）一个操作
2）僵尸撞上坚果墙（或走到位置0）
3）僵尸撞上正在啃坚果墙的僵尸
4）坚果墙被啃完了或僵尸啃坚果墙的时候死了

不难发现这些操作的个数都是O(Q)级别的。

E – Protecting lawn

关键词： 模拟 数据结构

可以再用一个平衡树（std::set）维护僵尸和坚果的相对位置，来模拟事件和更新将要发生的事件。

需要注意一些细节的实现。

总时间复杂度O(Q log Q)
不保证O(Q log^2 Q)能通过

K – Decoding graphs

关键词： 动态规划 容斥

首先考虑没有条件 1，即不考虑 $b_u \neq b_v$ 的做法。另 $F(\{a_1, a_2, \ldots, a_n\}, x)$ 为 $0 \leq b_i \leq a_i\ (1 \leq i \leq n), b_1 \oplus b_2 \oplus \ldots \oplus b_n = x$ 的 b 的个数。另 a_i 为 $a_1, a_2, \ldots, a_n$ 中的最大值，另 t 为不超过 a_i 的最大的 2 的次幂。若不超过 x 的最大的 2 的次幂大于 t 则答案为 0。否则考虑当 $b_i < t$ 时，即 $t \otimes (\oplus_{j \neq i} b_j) = t \otimes x$ 时，对于满足此限制的任意一种 $b_j (j \neq i)$，都存在一个 b_i 使得 $\oplus_{j=1}^{n} b_j = x$ 并且 $0 \leq b_i \leq a_i$，此答案可以通过 $O(n)$ 的 dp 求出。而当 $b_i \geq t$ 时，递归求解 $F(\{a_1, a_2, \ldots, a_{i-1}, a_i \oplus t, a_{i+1}, \ldots, a_n\}, x \oplus t)$ 即可。

K – Decoding graphs

关键词： 动态规划 容斥

现在考虑有 $b_u \neq b_v$ 的限制的情况，考虑容斥。我们枚举一种划分 $\{\{v_{1,1}, v_{1,2}, \ldots, v_{1,s_1}\}, \{v_{2,1}, v_{2,2}, \ldots, v_{2,s_2}\}, \ldots, \{v_{k,1}, v_{k,2}, \ldots, v_{k,s_k}\}\}$，并要求对于每个集合中的 v，b_v 都相等，对于不同集合中的 v，b_v 不做约束。我们考虑计算容斥系数。可以通过归纳法证明划分 $\{\{v_{1,1}, v_{1,2}, \ldots, v_{1,s_1}\}, \{v_{2,1}, v_{2,2}, \ldots, v_{2,s_2}\}, \ldots, \{v_{k,1}, v_{k,2}, \ldots, v_{k,s_k}\}\}$ 的容斥系数为其每个集合的容斥系数的积，即 $coef_{\{\{v_{1,1}, v_{1,2}, \ldots, v_{1,s_1}\}, \{v_{2,1}, v_{2,2}, \ldots, v_{2,s_2}\}, \ldots, \{v_{k,1}, v_{k,2}, \ldots, v_{k,s_k}\}\}} = \Pi_{i=1}^{k} coef_{\{v_{i,1}, v_{i,2}, \ldots, v_{i,s_i}\}}$。另 $f_{\{v_1, v_2, \ldots, v_k\}} = [\{v_1, v_2, \ldots, v_k\}$ 是独立集 $]$。则所求的容斥系数 $coef_{\{v_1, v_2, \ldots, v_k\}}$ 应该满足 $f_S = \sum_{\cup_{i=1}^{k} S_i = S, \forall (i,j)(i \neq j), S_i \cap S_j = \emptyset} \frac{\Pi_{i=1}^{k} coef_{S_i}}{k!}$。另 $F(s)$ 为 f 的集合幂级数，$COEF(s)$ 为 $coef$ 的集合幂级数，则有 $COEF = ln(F)$，直接集合幂级数求 ln 即可。

最终复杂度 $O(Bell(n) + n^2 2^n)$

Thanks