Solution - Contest 5 by group C

预期难度

C1 = E < D = F1 < C2 < A < B < G < F2

实际难度

C1 < E < D < C2 < A < B < F1 < G < F2

A - Matrix

Last accepted submission
孔畅

如果 $n = p$ 是质数,设$f(x) \in \mathbb{Z}(x)$ 是A的特征多项式,由于$A \neq E$, A必有不为1的特征值λ, 且λ 为$x^{p-1} + \cdots + 1$的根,由于它不可约,并且$\mathbb{Z}(x)$是UFD,所以$x^{p-1} + \cdots + 1 | f(x)$, 因此$k \geq p - 1$.

假设$n = p_1 p_2 \cdots p_k$, 可以容易得到$k \geq \text{least prime factor}(n) - 1$.

构造:考虑递推式$a_i = -a_{i-1} - a_{i-2} - \cdots - a_{i-k}$, 可以得到$a_{i+n} = a_i$, 写成矩阵即可.

B - Longest Path

Last accepted submission
章可循

首先1以及满足$2p > n$的质数p都没有边,不可能在路径上.

其次满足$2p \leq n, 3p > n$的质数只有一条边,必定在路径开始或结尾,因此只能有两个此类质数.

并且这个上界是准确的.

一种构造方法:考虑先把6的倍数排成一列,6,12,18...

然后对于每个满足$p > 5, 3p \leq n$的质数,把$2p, p, 3p$插到这些6的倍数中间.

再稍加调整把剩下可以插入的数都插入到合适的位置.

C1 - Subset of Subset (Easy)

Last accepted submission
黄彦玮

题意

给出树上的子点集，从中选出不相邻的点使得权值和最大。 询问次数较多。

解法

尽量不要对所有询问点遍历边，否则过不了菊花图。

判断询问点的父亲是否也是询问点，建立新图dp即可。

C2 - Subset of Subset (Hard)

Last accepted submission
杨沛霖

题意

给出树上的子点集，从中选出两两距离超过k=1或k=2的点使得权值和最大。 询问次数较多。

解法

k=1时同C1。

k=2时，同样需要缩减dp树的规模，考虑用虚树建立O(m)的dp树，dp时多判断距离即可。

D - Dynamic Tree

Last accepted submission
杨沛霖

这是一道猜结论题，大家玩得开心吗？

首先可以想到的是最后会出现循环----在一条比周围的所有的值都大的链上

经过大量数据的检验，我们发现循环节会在$O(n)$ 内出现

但是，我们无法证明出现循环的理论步数

所以此题采用了**随机数据(并在题面中说明)**， 我们检验过所有数据，都有$O(n)$的循环节。

于是直接暴力用hash的方式找出循环节，即可算出答案

E - Transform

Last accepted submission
祁佳晨

投影映射(Projective Mapping)是将图片投影到一个新的视平面(Viewing Plane)。通用的变换公式为：

$$\begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

(x, y)是原始坐标，对应得到变换后的坐标$(\frac{u'}{w'}, \frac{v'}{w'})$。

那么我们有

$$u_i = \frac{a_{11}x_i + a_{12}y_i + a_{13}}{a_{31}x_i + a_{32}y_i + 1}$$
$$v_i = \frac{a_{21}x_i + a_{22}y_i + a_{23}}{a_{31}x_i + a_{32}y_i + 1}$$

变形得：

$$a_{11}x_i + a_{12}y_i + a_{13} - a_{31}x_iu_i - a_{32}y_iu_i = u_i$$
$$a_{21}x_i + a_{22}y_i + a_{23} - a_{31}x_iv_i - a_{32}y_iv_i = v_i$$

写成矩阵的形式为：

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1u_1 & -y_1u_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1v_1 & -y_1v_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2u_2 & -y_2u_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2v_2 & -y_2v_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3u_3 & -y_3u_3 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3v_3 & -y_3v_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4u_4 & -y_4u_4 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -x_4v_4 & -y_4v_4 \end{bmatrix} \begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{21} \\ a_{22} \\ a_{23} \\ a_{31} \\ a_{32} \end{bmatrix} = \begin{bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \\ u_4 \\ v_4 \end{bmatrix}$$

解该线性方程得到变换矩阵后模拟变换即可。

F1 - Sum of Digits (Easy)

这道题是一道经典的数位dp

题解中认为所有数最高位是第n位，最低位是第1位

我们将$\sum s(a_i) \equiv s(\sum a_i) (\mod d)$这个柿子的右边移到左边，变成$\sum s(a_i) - s(\sum a_i) \equiv 0 (\mod d)$

$f[i][0/1][0/1][0/1][0/1][t][d]$ 表示考虑到第i位，第一个数和第二数的顶上下界情况，t表示从第$i-1$位是否进位，d表示上面式子的值

由于范围非常宽松，转移时直接暴力枚举即可（我们原本准备出到100000位所以标程做了很多预处理，但在这道题中不需要）

F2 - Sum of Digits (Hard)

这个hard version当然也是数位DP 题解中认为所有数最高位是第n位，最低位是第1位

考虑如何将F1的数位DP拓展到$k(k>2)$维：

1. 在数位DP中，我们要对每个数记录它是否贴了上下界，这题中我们显然不能开$2^k*2^k=4^k$这么大的状态存每个数贴的上下界情况，考虑如何优化
   - 对于L,R，我们假设它们第$i'+1$到第n位相同，从第i'位开始不同，那么前i'位这k个数一定是同时贴上下界，且从第i'位开始，他们一定不可能同时贴上下界，也就是只有3种情况：贴上界，贴下界，都不贴。于是我们对第$i'+1$位到第n位特殊处理，第i'位开始dp，就可以将上下界的状态缩小到3^k
   - 事实上3^k对于这个数据范围还是太大，而且可以进一步优化。其实没有必要记录每个数上下界的状态，我们只要对于这3个状态，记录每个状态的数有几个即可，用$[a][b]$记录贴下界的有a个数，贴上界的有b个，那么都不贴的就是$k-a-b$个，于是我们就用$k*k$的状态存下了k个数上下界的情况
2. 因为是k个数的和，当然会存在进位，要开个状态$[c]$记录下一位会进c位到这一位
3. 我们将$\sum s(a_i)\equiv s(\sum a_i)(\mod d)$这个柿子的右边移到左边，变成$\sum s(a_i)-s(\sum a_i)\equiv 0(\mod d)$，我们再开一维$[d]$存这个差

于是我们现在的状态是$f[i][a][b][c][d]$，表示当前dp到第i位，有a个数贴下界，b个数贴上界，第$i-1$位会进c位上来，柿子的值是d

然后考虑枚举了i,a,b,c,d后的转移

从第i'位开始dp，转移时枚举ai,bi,ci,di代表从i位转移到$i-1$位时，有ai个数从贴下界到不贴下界，有bi个数从贴上界到不贴上界，$i-2$位进位到$i-1$位进了ci位，第$i-1$位对柿子的贡献是di

我们用一个$h[i][ai][bi][ci][di]$表示第i位满足上述限制的填法的数量，对于每个枚举的i,a,b,c，去dp这个h数组

这个h不好直接dp，注意到影响h的值的只有第i位的上下界，外层的a，b，c，其实是不影响h的值的，所以对于每个i，我们再做一个dp

- 用$g[i][ai][bi][ci][cc][kc]$表示对于第i位，有ai个人从贴下界上来，bi个人从贴上界下来，ci个人原来就不贴上下界，下一位进位了kc个,$ai+bi+ci$个数这一位的和加上kc的进位的和为cc，这$ai+bi+ci$个数这一位的填法数量

有了g数组后h数组就很容易得到，注意处理h数组时转移要乘上组合数

得到了h数组，枚举一下就可以转移f了

为了卡掉复杂度不优的做法时限开到0.5s，标程最慢的点跑了0.12s，应该是不存在卡常这一说法的

G - Magpie Bridge

part1：维护kruskal 重构树

每次询问的联通块是最小生成树上的一个联通块

因为w从小到大，满足kruskal的加边顺序，于是在kruskal重构树中，这是一颗子树

需要支持合并，动态维护倍增表：维护每个点的2^k^的儿子，合并的时候要考虑叶子节点的高度变化

代码实现中，我采用了维护所有叶子的高度，并启发式合并。复杂度$O(nlog^2n)$

part2：查询

把查询的式子拆开：$(xi-x)^2+yi^2=-2xi*x+xi^2+yi^2+x^2$

这是斜率优化的标准形式。维护凸壳。

在part1中我们知道，现在要在一个子树的凸壳上查询最优值。

如何合并凸壳

用二项堆的思想，维护大小为$2^0,2^1,2^2,...,2^k$的凸壳。合并的时候直接合并相同大小（这个可以采用归并，也可以直接暴力插入，因为这不是复杂度瓶颈）

因为每个点只会在$logn$个凸壳中出现，总的大小是$O(nlogn)$。因此合并的总复杂度是$O(nlogn-nlog^2n)$

查询的时候直接在$logn$个凸包中查询。每次二分。总复杂度$O(nlog^2n)$