## **Day1**

### **T1 卡牌游戏**

据说有O(n)O(n)做法

先二分答案，然后枚举最小值是来自A还是来自B

这样我们需要反转一段前缀以及一段后缀，处理个B的前后缀最大/小值就能判断

从小往大枚举可以双指针做到O(n)O(n)check

时间复杂度O(nlogAi)O(nlogAi)

### **T2 矩阵游戏**

一道类似的题[link](https://www.codechef.com/DEC20A/problems/DGMATRIX" \t "https://www.cnblogs.com/deaf/p/_blank)

一道类似的题[link](https://www.luogu.com.cn/problem/P3208" \t "https://www.cnblogs.com/deaf/p/_blank)

传统的做法是枚举矩阵的a11a11然后设出每行和每列的第一个数差分约束建图，或者是搜索

但是这道题好像做不了，原因是值域太大

我们考虑先确定一个不考虑值域限制的矩阵然后调整它

那么你的操作就是交替给每一行/列执行++--的操作

于是你会发现你的每一个格子也是一个形如L≤ai±bj≤RL≤ai±bj≤R的形式

考虑反转所有奇数的行和偶数的列

那么每个格子都可以变成L≤ai−bj≤RL≤ai−bj≤R的形式

直接差分约束即可

juruo不会写spfa导致T成傻子，学了dalao的写法双端队列才过

认为n，mn，m同阶，时间复杂度O(n3)O(n3)

### **T3 图函数**

不难想到考虑每个点的贡献

一个点uu能对v≥uv≥u产生贡献当且仅当

uu和编号比它大的点构成的导出子图中uu和vv在同一个环上

于是就有一个每个点每张图都跑一次tarjan的优秀做法

时间复杂度O(nm2)O(nm2)

这个做法看起来没法优化

我们考虑这其实只要求一个环而不需要知道SCC

于是我们考虑直接对每个点uu考虑他能和vv成环当且仅当

图中存在一条u−>vu−>v的路径和v−>uv−>u的路径只经过编号≥u≥u的点

这好像是废话

但是这样就可以做了

考虑这条路径的贡献其实就是路径上编号最小的边，编号在这之前的图都可以满足条件

不难发现只需要对每个点dijkstra跑个最短路就可以计算贡献

但是这样看起来是O(nmlogm)O(nmlogm)的，貌似过不了

考虑一个更easy的想法

我们只需每次加入一个边的时候扩展uu所在的连通块

这样每个点/边只需被加入一次

时间复杂度O(nm)O(nm)

注意卡常

## **Day2**

### **T1 宝石**

树上跳等差数列一类的东西不难想到倍增

预处理出f/gf/g两个倍增数组，表示每个点向上第一个点权是这个点点权的前驱/后继的点

于是对于一个询问s−>ts−>t

s−>lcas−>lca的这段可以考虑用gg直接O(logn)O(logn)处理

lca−>tlca−>t的这段好像不好搞

只能先二分答案，然后用ff从tt跳上去判断是否可行

这个是O(log2n)O(log2n)的

我们发现不管是预处理倍增数组还是处理询问都需要访问一个点祖先上最近的某一种颜色的位置

不难想到把询问离线，开一个桶在树上扫3遍即可

时间复杂度O(nlogn+qlog2n)O(nlogn+qlog2n)

### **T2 滚榜**

不难想到O(n!∗n)O(n!∗n)的做法

即枚举所有排列

那么可以扫一遍出bi−bi−1bi−bi−1的最小值，做个前缀和判断是否≤m≤m即可

考虑优化成状压dp

不难想到设状态f[S][i][j][k]f[S][i][j][k]表示当前选了SS中的点，最后一个是ii，当前最后一个bibi是jj，当前所有bb的总和是kk的方案总数

发现时间复杂度O(2nn2m2)O(2nn2m2)甚至跑不过阶乘算法

不难想到jj和kk可以合在一起，考虑每一个差分即bi−bi−1bi−bi−1的贡献是独立的

于是状态简化为f[S][i][j]f[S][i][j]表示当前选了SS中的点，最后一个是ii，当前所有差分的贡献总和是jj的方案总数

时间复杂度O(2nn2m)O(2nn2m)

用计算器一算5e8，还有很多空状态，用刷表dp跑得飞快

### **T3 支配**

不难得到支配树的O(nm)O(nm)建法

即暴力算出每一个点支配哪些点

于是我们就得到了一个每次询问重构支配树的O(nmq)O(nmq)算法

考虑每次直接在支配树上搞

我们发现加边只会让点的支配集减小

并且若一个点的支配集减小，那么它的子树的支配集也会减小

于是不能难发现一个点，若它祖先的支配集都没被改变，那么它的支配集里一定删去了支配树上它的父亲

那么考虑加入x−>yx−>y会影响哪些点uu，使它的支配集里删去了它的父亲，然后再dfs一遍即可算出答案

首先，yy需要能够不经过uu支配树上的父亲而到达uu

这个可以枚举每个点，在反图中删去它的父亲来预处理

其次，xx和uu的lca不能是uu的父亲

这一点只需判断fa[u]fa[u]是否支配xx即可

于是我们就在O(nm)O(nm)预处理，单次O(n)O(n)处理询问的复杂度内解决了这个问题

总时间复杂度O(n(m+q))O(n(m+q))