Codeforces Round #698 (Div. 1)

比赛链接

A. Nezzar and Board

题意

给定 n 个不同的 x_i，每次可以选取一对 x,y，得到 $2x-y$，(注意该操作不会删去原有的 x,y)

问是否能经过有限次操作得到 k。

题解

不妨设 $b=\min(x_i)$，记 $a_i=x_i-b,k'=k-b$，于是有 $2x_i-x_j=2a_i-2b-a_j+b=(2a_i-a_j)-b$。

如果用 a_i 替代 x_i，k' 替代 k，则所有结果相当于都平移了 b，对答案不影响。于是不妨设序列为 $0,a_2,a_3\cdots a_n$。

首先对于 $0,a$，有 $2a-0=2a,2*(2a)-a=3a,2*(3a)-(2a)=4a$，于是可以得到所有 a 的倍数。

然后假设 $n=t$ 时 g_t 一定可以得到，于是能得到 k 当且仅当 $g_t=\text{gcd}(a_2,a_3\cdots a_t)\mid k$ 。

$n=2$ 时相当于 $0,a$ 的情况，显然成立。

$n=t+1$ 时，根据裴蜀定理，知存在 x,y 满足

$$ x*g_t-y*a_{t+1}=\text{gcd}(g_t,a_{t+1})=g_{t+1} $$

根据归纳假设，可以得到 g_t，如果 $x=2x'$，则先得到 $x'*g_t$ 和 $y*a_{t+1}$，于是 $g_{t+1}=2x'g_t-y*a_{t+1}$。

同理 y 为偶数时也能得到 g_{t+1}，下证一定存在 x,y 满足至少有一个偶数。

假设得到的一组 x,y 为全为奇数，则记 $c=\frac {g_t}{g_{t+1}},d=\frac {a_{t+1}}{g_{t+1}}$，于是有

$$ 1=x*c-y*d=(x+d)*c-(y+c)*d $$

因为 $(c,d)=1$，所以 c,d 中至少一个奇数，于是 $x+d,y+c$ 中至少一个偶数，证毕。

D. Nezzar and Hidden Permutations

题意

给定两个 $1\sim n$ 的排列 p,q 和 m 个限制。限制 (l_i,r_i) 要求 $(p_{l_i}-q_{l_i}\ge 0)=(p_{r_i}-q_{r_i}\ge 0)$。

要求在满足所有限制的前提下使得满足 $p_i\neq q_i$ 的 i 最多，输出此时的排列 p,q。

题解

对原题进行转化，等价于对点 $1\sim n$ 每个点有两个权值 p_i,q_i。

同时每个限制条件对 l_i,r_i 连一条边，于是只要有连边的点之间满足之前的限制条件即可。

对点 i 与该点有连边的点要么 $\max(p_j,q_j)\le \min(p_i,q_i)$ 要么 $\min(p_j,q_j)\ge \max(p_i,q_i)$。

于是 $\deg(i)\le \min(p_i,q_i)+n-1-\max(p_i,q_i)\le n-1$ 取等号条件为 $p_i=q_i$。

于是对所有度数为 $n-1$ 的点，依次将 $1,2\cdots k_0$ 分配给这些点，这些点对其他点的限制被消除，直接将这些点删去。

下面证明余下点均可满足 $p_i\neq q_i$。

考虑该图的补图，易知只要使得补图中所有没有连边的点满足之前的限制条件即可。

同时由于删去点后原图中不存在度数为 $n-1$ 于是补图不存在孤立点。

首先考虑菊花图的情况，令中心点为 (k_0+1,k_1) 其他点依次为 $(k_0+2,k_0+1),(k_0+3,k_0+2)\cdots (k_1,k_1-1)$。

由于菊花图的限制仅存在于两两非中心点之间，是该菊花图满足限制且每个点 $p\neq q$。

将原图划分为若干个菊花图，且每个菊花图至少有两个点，将 (k_0+1,k_1) 分配给第一个菊花图 (k_1+1,k_2) 分配给第二个菊花图…

于是每个菊花图内部满足限制，每个菊花图之间也满足限制。

接下来问题转化为怎么将补图划分为若干个菊花图，且每个菊花图至少有两个点。

一个经典算法为将补图先划分为若干连通块，每个连通块任意得到一个生成树。

接下来对每个生成树，取该树的任意叶子，找到与该叶子结点相邻的结点 u 作为菊花图的中心。

对所有与 u 相邻的结点，如果该点是叶子结点，则将该结点加入菊花图。否则删去该结点与 u 的连边。

最后将得到的菊花图从点集中删去，直到点集中没有剩余结点，则算法结束。时间复杂度 $O((n+m)\log n)$。

```
typedef set<int>::iterator iter;
const int MAXN=5e5+5;
struct Node{
```

```
    int idx,deg;
    bool operator < (const Node &b)const{
        if(deg!=b.deg)
        return deg<b.deg;
        else
        return idx<b.idx;
    }
};
struct star{
    int root;
    vector<int> son;
};
int deg[MAXN],idx1[MAXN],idx2[MAXN];
set<int> g0[MAXN],g[MAXN],node0;
set<Node> node;
vector<star> stars;
void dfs(int u){
    node0.erase(u);
    int pos=0;
    while(true){
        iter it=node0.upper_bound(pos);
        if(it==node0.end())
        break;
        pos=*it;
        if(g0[u].find(pos)==g0[u].end()){
            g[u].insert(pos);
            g[pos].insert(u);
            dfs(pos);
        }
    }
}
int main()
{
    int T=read_int();
    while(T--){
        int n=read_int(),m=read_int();
        _rep(i,1,n){
            g0[i].clear();
            g[i].clear();
            node0.insert(i);
        }
        stars.clear();
        while(m--){
            int u=read_int(),v=read_int();
            g0[u].insert(v);
            g0[v].insert(u);
        }
        while(!node0.empty())
        dfs(*node0.begin());
        int pos1=1,pos2=1;
        _rep(i,1,n){
```

```
            deg[i]=g[i].size();
            if(!deg[i]){
                idx1[i]=pos1++;
                idx2[i]=pos2++;
            }
            else
            node.insert(Node{i,deg[i]});
        }
        while(!node.empty()){
            int leaf=node.begin()->idx,u=*g[leaf].begin();
            node.erase(Node{u,deg[u]});
            vector<int> son;
            for(iter it=g[u].begin();it!=g[u].end();++it){
                int v=*it;
                if(deg[v]==1){
                    son.push_back(v);
                    node.erase(Node{v,deg[v]});
                }
                else{
                    g[v].erase(u);
                    node.erase(Node{v,deg[v]--});
                    node.insert(Node{v,deg[v]});
                }
            }
            stars.push_back(star{u,son});
        }
        _for(i,0,stars.size()){
            idx1[stars[i].root]=pos1++;
            _for(j,0,stars[i].son.size()){
                idx1[stars[i].son[j]]=pos1++;
                idx2[stars[i].son[j]]=pos2++;
            }
            idx2[stars[i].root]=pos2++;
        }
        _rep(i,1,n)
        space(idx1[i]);
        puts("");
        _rep(i,1,n)
        space(idx2[i]);
        puts("");
    }
    return 0;
}
```