

Aimai Trip

题目描述

有 n 只猫猫在玩捉迷藏。猫猫们的编号分别为 $1, 2, \cdots, n$ 。

屋里有 k 个房间，每个房间只能容纳一只猫猫。

每个房间有一定的属性，可以用 x_i, y_i 来表示，意为：

- 这个房间要么空着，要么容纳第 x_i 只猫猫，要么容纳第 y_i 只猫猫。

现在云浅可以买下前 m 个房间给猫猫们使用。云浅想知道， m 的值至少是多少，才能容纳所有猫猫。保证买下所有房间后一定可以容纳所有猫猫。

为了更好地帮助小可爱，你还需要输出一个方案。如果有多个可行的方案，输出任意一个即可。

输入格式

第一行两个正整数 n, k 。

第二行 k 个正整数 $x_1, \cdots, x_k$ ，表示每个房间的第一种属性。

第三行 k 个正整数 $y_1, \cdots, y_k$ ，表示每个房间的第二种属性。

输出格式

先输出一行一个正整数 m 表示答案。

第二行输出 m 个正整数，以空格隔开。其中，若第 i 个数为 0 则表示第 i 个房间空着不用，第 i 个数为 1 则表示分给第 x_i 只猫猫，第 i 个数为 2 则表示分给第 y_i 只猫猫。

数据范围

对于 100% 的数据， $1 \leq n \leq 10^5, 1 \leq k \leq 2 \times 10^5, 1 \leq x_i, y_i \leq n, x_i \neq y_i$ 。

测试点编号	n	k
1 ~ 6	≤ 20	≤ 20
7 ~ 10	≤ 1000	≤ 1000
11 ~ 14	≤ 1000	$\leq 10^5$
15 ~ 20	$\leq 10^5$	$\leq 2 \times 10^5$

题解

首先二分答案，相当于只需要判定一下是否所有小可爱都能住进至少一个房间。

考虑对每个房间连边 (x_i, y_i) ，那么，所有小可爱都能住进至少一个房间，当且仅当：

- 图中的每个连通块的边数均不小于点数。

证明：若要求所有小可爱都能住进至少一个房间，显然边数必然不小于点数。

如果边数不小于点数，那么这个连通块中至少存在一个环。考虑令在环上的边所代表的房间依次给第 x_i 只小可爱，然后从这个环向外延伸出去若干条链，链上的边从环内指向环外即可。

构造方案也可以使用类似的方法。

于是就做完了，时间复杂度 $O((n + k) \log k)$ 。

参考代码

```
#include<bits/stdc++.h>

#define int long long

using namespace std;

inline int read(){
    int x=0,f=1;char c=getchar();
    for(;;(c<'0' || c>'9');c=getchar()){if(c=='-')f=-1;}
    for(;;(c>='0'&&c<='9');c=getchar())x=x*10+(c&15);
    return x*f;
}

const int MN=2e5+5;
int n,k;
int fr[MN],to[MN];

struct Edge{
    int to,id;
    Edge(int T,int I):to(T),id(I){}
    Edge(){}
};

vector<Edge>G[MN];
vector<int>S;
bool vis[MN];

void getnodes(int u){
    vis[u]=1;S.push_back(u);
    for(auto e:G[u]){
        int v=e.to,i=e.id;
        if(!vis[v])getnodes(v);
    }
}

vector<int>C;
int fa[MN],pre[MN],res[MN];
bool fi=0;
void findcycle(int u){
    vis[u]=1;
    for(auto e:G[u]){
        int v=e.to,i=e.id;
        if(i==pre[u])continue;
        if(!vis[v]){fa[v]=u,pre[v]=i,findcycle(v);continue;}
        if(fi)continue;
        int x=u;
        while(x!=v){
            C.push_back(x),res[pre[x]]=(fr[pre[x]]==x?2:1);
            x=fa[x];
        }
        res[i]=(fr[i]==u?1:2);fi=1;C.push_back(v);
    }
}

bool chk(int m){
    for(int i=1;i<=n;i++)G[i].clear();
    memset(vis,0,sizeof(vis)),memset(pre,0,sizeof(pre)),memset(fa,0,sizeof(fa));

    for(int i=1;i<=m;i++){
        int u=fr[i],v=to[i];
        G[u].push_back(Edge(v,i)),G[v].push_back(Edge(u,i));
    }

    for(int i=1;i<=n;i++){
        if(vis[i])continue;
        fi=0,C.clear();findcycle(i);
        if(!fi)return 0;
    }
    return 1;
}

void dfs(int u){
    vis[u]=1;
    for(auto e:G[u]){
        int v=e.to,i=e.id;
        if(vis[v] || res[i]!=0)continue;
        res[i]=(fr[i]==v?1:2);dfs(v);
    }
}
```

```

80 }
81
82 void getans(int m){
83     for(int i=1;i<=n;i++)G[i].clear();
84     memset(vis,0,sizeof(vis)),memset(pre,0,sizeof(pre));
85     memset(fa,0,sizeof(fa)),memset(res,0,sizeof(res));
86
87     for(int i=1;i<=m;i++){
88         int u=fr[i],v=to[i];
89         G[u].push_back(Edge(v,i)),G[v].push_back(Edge(u,i));
90     }
91
92     for(int i=1;i<=n;i++){
93         if(vis[i])continue;
94         S.clear();getnodes(i);fi=0;
95         for(int p:S)vis[p]=0;
96         C.clear();findcycle(i);
97         for(int p:S)vis[p]=0;
98         for(int p:C)vis[p]=1;
99         for(int p:C)dfs(p);
100        for(int p:S)vis[p]=1;
101    }
102
103    for(int i=1;i<=m;i++)cout<<res[i]<<" ";puts("");
104 }
105
106 signed main(void){
107
108     freopen("cats.in","r",stdin);
109     freopen("cats.out","w",stdout);
110
111     n=read(),k=read();
112     for(int i=1;i<=k;i++)fr[i]=read();
113     for(int i=1;i<=k;i++)to[i]=read();
114     int L=1,R=k,ans=L;
115     while(L<=R){
116         int mid=(L+R)>>1;
117         if(chk(mid))R=mid-1,ans=mid;
118         else L=mid+1;
119     }
120     cout<<ans<<endl;
121     getans(ans);
122
123     return 0;
124 }

```