CQU-ACM-ICPC

Sundes@foxmail.com

2019/9/2

目录

博弈论

基础博弈

Bash Game

一堆石子共 n 个，两个玩家轮流拿，每次至少 1 个最多 m 个，问先手是否必胜.
如果 n = (m+1)*k 则先手必败，否则先手必胜

Nim Game

给你 n 堆石子，每次可以从任意一堆拿出除 0 外任意个数的石子，问先手是否必胜.
其实可以看作是组合游戏的特殊情况.mex(n)=n

所以只要所有堆石子数异或和为 0 则先手必败.

```
//#include "bits/stdc++.h"
#include "iostream"
#include "cstdio"
using namespace std;
typedef long long ll;

int n,res;

int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);

    while(cin>>n)
    {
        res=0;
        for(int i=1,u;i<=n;i++)
        {
            cin>>u;
            res^=u;
        }

        if(res==0)
            cout<<"No"<<endl;
```

```
        else
            cout<<"Yes"<<endl;
    }

    return 0;
}
```

Poj2975 问 nim 游戏开局有多少种取法可以获胜.
假设 a1^a2^a3...=S 那么 将任意一项变为 S^ai 异或和就变为 S^ai^(S^ai) ==0
但是需要判断 S^ai 的值要小于 ai (因为你不能添加石头)

```
//#include "bits/stdc++.h"
#include "iostream"
#include "cstdio"
using namespace std;
typedef long long ll;

int n,a[105000];

int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);

    while(cin>>n && n)
    {
        ll res=0;
        for(int i=1;i<=n;i++)
        {
            cin>>a[i];
            res^=a[i];
        }

        if(res==0)
            cout<<0<<endl;
        else
        {
            int cnt=0;
            for(int i=1;i<=n;i++)
                if((a[i]^res)<a[i])
                    cnt++;
            cout<<cnt<<endl;
        }
```

```
    }

    return 0;
}
```

Auti-nim Game

Nim 游戏的变种，只是失败的认定方式改变一下，谁拿到最后一个石子则输.
可以使用 SJ 定理.可以看这篇论文:2019IOI 集训队论文-贾志豪

先手胜当且仅当
（1）所有堆石子数都为 1 且游戏的 SG 值为 0 ，（2）存在某堆石子数大于 1 且游戏的 SG 值不为 0
证明：
（1）若所有堆石子数都为 1 且 SG 值为 0，则共有偶数堆石子，故先手胜。
（2）
i)只有一堆石子数大于 1 时，我们总可以对该堆石子操作，使操作后石子堆数为奇数且所有堆得石子数均为 1
ii)有超过一堆石子数大于 1 时，先手将 SG 值变为 0 即可，且总还存在某堆石子数大于 1
(来源 hzw 博客)

POJ-3480 给 n 堆石，轮流选择一堆拿至少一个，拿到最后一个的败．模板题

```
#include "iostream"
#include "cstdio"
#include "cstring"
using namespace std;

int t,n,ret,cnt;

int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);

    cin>>t;
    while(t--)
    {
        cnt=ret=0;

        cin>>n;
        for(int i=1,u;i<=n;i++)
        {
```

```
            cin>>u;
            ret^=u;
            if(u==1)cnt++;
        }

        if((cnt==n && ret==0) || (cnt!=n && ret!=0))
            cout<<"John"<<endl;
        else cout<<"Brother"<<endl;
    }

    return 0;
}
```

Fib Game

一堆或多堆石子，每次仅能取出斐波那契数列中元素个石子，先取完失败.
根据 sg 函数 sg 定理，随便搞搞就好了.

Hdu1848 三堆石子

```
#include "bits/stdc++.h"
using namespace std;
typedef long long ll;

int vis[1050],mex[1050],fib[100];
int n,m,p;

int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);

    fib[0]=fib[1]=1;
    for(int i=2;i<=15;i++)
        fib[i]=fib[i-1]+fib[i-2];

    mex[0]=0;
    for(int i=1;i<=1000;i++)
    {
        memset(vis,0,sizeof(vis));
        for(int j=1;j<=15 && fib[j]<=i;j++)
```

```
                vis[mex[i-fib[j]]]=1;

            for(int j=0;j<=1000;j++)
                if(!vis[j])
                {
                    mex[i]=j;
                    break;
                }
        }

        while(cin>>n>>m>>p)
        {
            if(n==0 && m==0 && p==0)return 0;
            if((mex[n]^mex[m]^mex[p])==0)
                cout<<"Nacci"<<endl;
            else
                cout<<"Fibo"<<endl;
        }

        return 0;
    }
```

Wythoff Game

Poj 1067 给两堆石子，分别有 a、b 颗石子，两人轮流游戏，A 先手，可以从一堆拿任意非 0 颗，或者从两堆都拿任意非 0 颗，拿走最后一颗获胜，问能否先手必赢。

黄金分割比值：*0.6180339887 4989484820 458683436565*

```
int Wythoff(ll a,ll b)//威佐夫博弈
{
    if(a<b)swap(a,b);
    ll c=(ll)((a-b)*(sqrt(5.0)+1)/2);
    if(b==c)return 1;//如果拿完最后一颗获胜，则返回 1 表示先手必胜
    return 0;
}
```

高精度的威佐夫博弈：

```cpp
const ll mod=1e9;

int Wythoff(ll a,ll b)//高精度威佐夫
{
    static ll ratio[3]={618033988,749894848,204586834};
    if(a<b)swap(a,b);
    ll tmp=a-b;
    ll loword=tmp%mod,hiword=tmp/mod;

    ll sum=loword*ratio[2];
    sum=hiword*ratio[2]+loword*ratio[1]+sum/mod;
    sum=hiword*ratio[1]+loword*ratio[0]+sum/mod;
    sum=tmp+hiword*ratio[0]+sum/mod;
    if(sum==b)
        return 1;
    return 0;
}
```

Imitate Game

对称游戏

Poj2484 n 个棋子围成一圈，两人轮流从中取走一或两个棋子，不过取两个时必须是连续的棋子. 棋子取走之后留下空位，相隔空位的棋子不连续.

当 n>3 时先手必败，无论 A 第一次怎么选择，B 都可以选择一个对称的位置拿走一个或者两个，将妻子变成两对称的部分.之后只要一直模仿 A 的选择，一定能保证自己拿走最后一个.

```cpp
//#include "bits/stdc++.h"
#include "iostream"
#include "cmath"
#include "cstdio"
using namespace std;
typedef long long ll;
ll n;
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);

    while(cin>>n&&n)
    {
        if(n==1 || n==2)
```

```
        {
            cout<<"Alice"<<endl;
            continue;
        }
        cout<<"Bob"<<endl;
    }
    return 0;
}
```

Euclid Game

Poj 2348 两堆石子，两人轮流操作，每次只能从较大的那一堆中拿去较小那一堆石子的倍数个.

不妨设 a>b

1. 若 a-b<b 那么只有这一种拿的可能 拿完后状态变为(a-b,b)，若(a-b,b)必败则(a,b)必胜，而若(a-b,b)必胜，则(a,b)必败
2. 若 a-b>b 此时假设可以从 a 中拿走 x 个 b，若(a-xb,b)必败，我们只需要拿走 x 个 b 即可，否则若(a-xb,b)必胜，我们只需要拿走(x-1)个 b，这样就可以把必败态丢给对方.

   综上，若 a-b>b 或 a%b==0 则先手必胜，否则胜负与 (a-b,b)相反.

```
//#include "bits/stdc++.h"
#include "iostream"
#include "cstdio"
using namespace std;
typedef long long ll;

ll a,b,res;

int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);

    while(cin>>a>>b && a+b)
    {
        res=1;
        if(b>a)swap(a,b);
        while(b!=0)
```

```
        {
            if(a%b==0 || a-b>b)break;
            a=a-b;
            if(b>a)swap(a,b);
            res^=1;
        }
        if(res)cout<<"Stan wins\n";
        else cout<<"Ollie wins\n";
    }
    return 0;
}
```

杂题

ZOJ-3057 三堆石头，两人轮流，每次可以从一堆中拿任意非 0 个，或者从任意两堆中同时拿任意非 0 个. N<=300

只需要考虑必胜态和必败态. 三堆同时为 0 时是必败态，能转移到必败态的状态全为必胜态. 因此 DP 即可.

```
//#include "bits/stdc++.h"
#include "iostream"
#include "cmath"
#include "cstdio"
using namespace std;
typedef long long ll;

bool f[301][301][301];
int a,b,c;

int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);

    f[0][0][0]=0;

    for(int i=0;i<=300;i++)
        for(int j=0;j<=300;j++)
            for(int k=0;k<=300;k++)
```

```
                if(!f[i][j][k])
                {
                    for(int d=1;i+d<=300 && j+d<=300;d++)
                        f[i+d][j+d][k]=1;
                    for(int d=1;i+d<=300 && k+d<=300;d++)
                        f[i+d][j][k+d]=1;
                    for(int d=1;j+d<=300 && k+d<=300;d++)
                        f[i][j+d][k+d]=1;

                    for(int d=1;i+d<=300;d++)
                        f[i+d][j][k]=1;
                    for(int d=1;j+d<=300;d++)
                        f[i][j+d][k]=1;
                    for(int d=1;k+d<=300;d++)
                        f[i][j][k+d]=1;
                }

    while(cin>>a>>b>>c)
        cout<<f[a][b][c]<<'\n';
    return 0;
}
```

Hdu-4388 给你 n 堆石子，每次操作可以从任意一堆 a 个中拿走一部分剩下 k 个，使得 k<a 且 a^k<a．然后再加入有 a^k 个石子的新堆，(就是把 a 分成 k 和 a^k 两堆,且 k<a , a^k<a)

首先如果一堆石头数量是2^i 肯定是无法操作的，因为找不到一个数 k 使得 k<2^i且 k^2^i<2^i，题解里说，如果一堆石子个数为 a，那么每次操作本质是从 a 中分离出二进制中的 1.假设 a 有 s 个 1，那么 a 最多操作 s-1 次．所以答案和 所有石子堆中石子数的二进制中 1 的个数减去 1 之和有关..

这样确实可以 A 这道题.但是我还有一个没想清楚的点.假设一堆石子是 10001000 个,拿走一部分使得剩余 00001100 个,10001000^00001100=10000100 满足 k<a,k^a<a 但是这样操作可操作次数不仅仅是 s-1．姑且先放在这．下次想通了再改.

```
//#include "bits/stdc++.h"
#include "iostream"
#include "cstdio"
using namespace std;
typedef long long ll;

ll res,n,cas,t;

int main()
{
```

```
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);

    cin>>t;
    while(t--)
    {
        res=0;
        cin>>n;
        for(int i=1,u;i<=n;i++)
        {
            cin>>u;
            res+= __builtin_popcount(u)-1;
        }

        cout<<"Case "<<++cas<<": ";
        if(!(res&1))cout<<"No\n";
        else cout<<"Yes\n";
    }

    return 0;
}
```

Poj-1740 n 堆石子，每次你可以选择在一堆中取至少一颗石子，之后可以选择（也可以不选择）将剩下的石子放到其他堆里面，不能取输.

观察规律:
n=1 时先手必胜
n=2 时，若两组相同，先手任何操作均可模仿，故必败，若不同可以变成相同,故必胜.
n=3 时，先手选择较多那组，一定可以去掉较多组，并且使最少组数量变为和次小组相同，故回到了情况 2 相同局势，则先手必胜
n=4 时，若两两相同，则可以看作是两组 2 情况的游戏的组合，后者模仿之故必败，若两两不相同，则选择最大那一组，一定能将 4 组推平，故先手必胜.
N=5...

结论:
1.当 n 为奇数的时候，先手必胜。
2.当 n 为偶数的时候，如石子堆两两相等，则先手必败，反之先手必胜。

```
#include "iostream"
#include "algorithm"
#include "cstring"
#include "vector"
using namespace std;
```

```

int n,a[100];

int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);

    while(cin>>n && n)
    {
        for(int i=1;i<=n;i++)
            cin>>a[i];
        if(n%2)cout<<1<<endl;
        else
        {
            sort(a+1,a+n+1);
            bool f=1;
            for(int i=1;i<n;i+=2)
                if(a[i]!=a[i+1])
                {
                    f=0;
                    cout<<1<<endl;
                    break;
                }
            if(f)cout<<0<<endl;
        }
    }
    return 0;
}
```

Poj-2068 有 s 个石子, 2n 个人, 1 3 5..2n-1 是我们的人 2 4 6..2n 是对方的, 每个人有一个自己能拿的石子个数上限. 1 号拿完 2 号拿..以此类推.. 现在从 1 号开始.问有没有必胜策略.

由于每个人能拿的个数是不一样的. 显然是个不公平博弈. 因此只能考虑搜索状态来解决. 由于 n<=20/ s<=8000 这里我们尝试记忆化搜索构建博弈树,直接搜索得到结果.

```
#include "iostream"
#include "algorithm"
#include "cstring"
#include "vector"
using namespace std;

int dp[25][10000],n,a[25],s;
```

```

int dfs(int u,int res)
{
    if(u==n+1)u=1;
    if(dp[u][res]!=-1)return dp[u][res];
    if(res==1)return 0;
    if(res<=a[u])return 1;

    for(int i=1;i<=min(res,a[u]);i++)
        if(!dfs(u+1,res-i))
            return dp[u][res]=1;
    return dp[u][res]=0;
}

int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);

    while(cin>>n && n)
    {
        memset(dp,-1,sizeof(dp));
        cin>>s;
        n<<=1;
        for(int i=1;i<=n;i++)
            cin>>a[i];
        if(dfs(1,s))
            cout<<1<<endl;
        else
            cout<<0<<endl;
    }

    return 0;
}
```

POJ-1704 阶梯博弈 给你一个 1xN 的棋盘，上面有一些棋子，每次可以选择一个棋子向左走至少一步，但是不能跨过其他棋子，谁先无法行动败.

考虑从右往左，将棋子两两看成一个取石子游戏，若为奇数个，则第一个棋子与原点 0 视为一个游戏，所有棋子即组成经典的 nim 游戏，为什么能当作取石子游戏？首先假设对手移动一对棋子中的后一个，间隔可以视作石子，移动则可以视为取走一定数量的石子，而若对方移动前一个棋子，则我们一定可以移动后一个棋子相同长度，则将原本的局面又还给了对方，

那么必然存在一个情况无法再移动前棋子，所以必然可以看作取石子游戏，同时这也说明，一对的前棋子与上一对的后棋子之间的间隔其实对游戏并没有影响.

```
#include "iostream"
#include "algorithm"
#include "cstring"
#include "vector"
using namespace std;

int a[1050],t,n;

int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);

    cin>>t;
    while(t--)
    {
        int ret=0;
        cin>>n;
        for(int i=1;i<=n;i++)
            cin>>a[i];
        sort(a+1,a+n+1);
        for(int i=n;i>=1;i-=2)
            ret^=(a[i]-a[i-1]-1);
        if(ret)cout<<"Georgia will win"<<endl;
        else cout<<"Bob will win"<<endl;
    }
    return 0;
}
```

POJ-2505 乘数博弈 一开始 p=1 给出 n，两个人轮流可以选择使 p 乘[2,9]中的一个数字，如果乘完后大于等于 n 则获胜.

分析一下

n= 0~9　　先手必胜
n=10~18　后手必胜
n=19~162 先手必胜..

可以观察发现分界线.

```cpp
#include "iostream"
#include "cstdio"
#include "vector"
using namespace std;
typedef long long ll;

double n;

int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);

    while(cin>>n)
    {
        while(n>18)n/=18;
        if(n>9)cout<<"Ollie wins."<<endl;
        else cout<<"Stan wins."<<endl;
    }

    return 0;
}
```

SG 定理

决策空间为空则败的多个游戏组合一起的 sg 值为各个游戏单独 sg 值的 xor 和；先手必败当且仅当 sg 值为 0.否则均为先手必胜.

SJ 定理

Anti-SG 游戏规定，决策集合为空的游戏者胜，Anti-SG 其他规则与 SG 游戏相同.

对于任意一个 Anti-SG 游戏，如果我们规定当局面中所有的单一游戏的 SG 值为 0 时，游戏结束，则先手必胜当且仅当：

(1) 游戏的 SG 函数不为 0，且游戏中某个单一游戏的 SG 函数大于 1

(2) 游戏的 SG 函数为 0，且游戏中没有单一游戏的 SG 函数值大于 1.

SG Game

POJ2960 给你 n 堆石子，每次只能取给定的 k 个数个石子，问能先手的情况.

先搜出单堆石子的 mex 值，根据 SG 定理，组合游戏的结果取决于各堆的 xor-sum，若为 0 则先手必败，否则先手必胜.

```
//#include "bits/stdc++.h"
#include "iostream"
#include "cmath"
#include "cstdio"
#include "vector"
#include "queue"
#include "algorithm"
#include "cstring"
using namespace std;
typedef long long ll;

vector<int>pb;

int mex[10500],vis[10500];
int k,m;

int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);

    while(cin>>k && k)
    {
        pb.clear();
        for(int i=1,u;i<=k;i++)
        {
            cin>>u;
            pb.push_back(u);
        }

        sort(pb.begin(),pb.end());

        mex[0]=0;
        for(int i=1;i<=10000;i++)
        {
            memset(vis,0,sizeof(vis));
            for(int j=0;j<k && pb[j]<=i;j++)
                vis[mex[i-pb[j]]]=1;
            for(int j=0;j<=10000;j++)
                if(!vis[j])
```

```
                {
                    mex[i]=j;
                    break;
                }
        }

        cin>>m;
        for(int i=1,u,v,res;i<=m;i++)
        {
            cin>>u;
            res=0;
            for(int j=1;j<=u;j++)
            {
                cin>>v;
                res^=mex[v];
            }

            if(res==0)
                cout<<'L';
            else
                cout<<'W';
        }
        cout<<'\n';
    }

    return 0;
}
```

POJ-2425 给你一棵有向树，很多节点上有棋子，一个节点上可能有多个棋子，棋子每次可以沿着边任意移动，谁不能移动谁输.

棋子的移动可以看作取石子过程，多个棋子其实就是标准的 SG 组合游戏. 记忆化搜索一下 sg 函数值即可.

```
#include "iostream"
#include "cstdio"
#include "vector"
using namespace std;
typedef long long ll;

int sg[1050],vis[1050][1050],n,t;
vector<int>g[1050];

int get_sg(int u)
{
```

```
    if(sg[u]!=-1)return sg[u];

    for(int i=0;i<(int)g[u].size();i++)
    {
        int v=g[u][i];
        vis[u][get_sg(v)]=1;
    }

    for(int i=0;i<=1000;i++)
        if(!vis[u][i])
            return sg[u]=i;
}

int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);

    while(cin>>n && n)
    {
        for(int i=0;i<n;i++)
        {
            g[i].clear();
            sg[i]=-1;
            for(int j=0;j<=1000;j++)
                vis[i][j]=0;
        }

        for(int i=0,u,v;i<n;i++)
        {
            cin>>u;
            for(int j=1;j<=u;j++)
            {
                cin>>v;
                g[i].push_back(v);
            }
        }

        while(cin>>t && t)
        {
            int ret=0;
            for(int i=1,u;i<=t;i++)
            {
```

```
                    cin>>u;
                    ret^=get_sg(u);
                }

                if(ret)cout<<"WIN"<<endl;
                else cout<<"LOSE"<<endl;
            }
        }
        return 0;
}
```

Multi-SG Game

Multi-SG 游戏规定，在符合拓扑原则下，一个单一游戏的后继可能是多个单一游戏.其他规则与 SG 游戏一致，与 SG 游戏一样, Multi-SG 游戏同样可以使用 SG 函数对局面进行判断.

Every-SG Game

定义：Every-SG 游戏规定，对于还没有结束的单一游戏，游戏者必须对该游戏进行一步决策，其他规则与 SG 游戏相同.

在通过拓扑关系计算某一个状态点的 SG 函数时（事实上，我们只需要计算该状态点的 SG 值是否为 0），对于 SG 值为 0 的点，我们需要知道最快几步能将游戏带入终止状态，对于 SG 值不为 0 的点，我们需要知道最慢几步游戏会被带入终止状态，我们用 step 函数来表示这个值。

$$\mathrm{step}(v)=\begin{cases} 0 & v\text{为终止状态} \\ \max(\mathrm{step}(u))+1 & SG(v)>0 \wedge u\text{为}v\text{的后继状态} \wedge SG(u)=0 \\ \min(\mathrm{step}(u))+1 & SG(v)=0 \wedge u\text{为}v\text{的后继状态} \end{cases}$$

定理：对于 Every-SG 游戏先手必胜当且仅当单一游戏中最大的 step 为奇数.

Game on tree

树上删边

给出一个有 n 个点的树，有一个节点作为树的根，游戏者轮流从树中删除边，删去一条

边后，不与根节点相连的部分将被移走，谁无法移动谁输.

结论：叶子节点的 sg 值为 0；中间节点的 SG 值为它的所有子节点的 sg 值加 1 后的异或和.可以使用数学归纳法证明.参考：自为风月马前卒博客

一棵树的状态可以由根节点的 mex 值表示．组合游戏则是多个树的 mex 值异或和.

而如果这棵树变为仙人掌(某些叶子节点处会挂着一个环，每个节点最多参与一个环，每个环最多只有一个节点与主树相连)

Fusion Principle

Fusion Principle：
我们可以对无向图做如下改动：将图中的任意一个偶环缩成一个新点，任意一个奇环缩成一个新点加一个新边；所有连到原先环上的边全部改为与新点相连。这样的改动不会影响图的 SG 值。

无向图删边

仙人掌删边:
Poj 3710 给你一些树，每棵树只可能在叶子节点有一些环，这些环保证只有一个节点与树相连接，且每个节点最多只参与一个环．游戏开始,Sally 先手，双方可以选择图上的一些边删除，删除边后未与根节点相连的部分都会消失，轮流操作，无法操作的人输.

```
#include "iostream"
#include "cstring"
#include "cstdio"
#include "vector"
using namespace std;

int mp[105][105],vis[105];
int dfn[105],low[105],id;
int s[105],top;
vector<int>g[105];
//int ins[105];

void tarjan(int u,int pre)
{
    //cout<<"#"<<u<<" "<<pre<<endl;
    dfn[u]=low[u]=++id;
    s[top++]=u;

    for(int i=0;i<(int)g[u].size();i++)
    {
        int v=g[u][i];
```

```
        if(v==pre)
        {
            if(mp[u][v]%2==0)vis[u]=1;
            continue;
        }

        if(!dfn[v])
        {
            tarjan(v,u);
            low[u]=min(low[u],low[v]);
        }
        else
            low[u]=min(low[u],dfn[v]);
    }

    if(dfn[u]==low[u])
    {
        int cnt=1;top--;
        while(s[top]!=u)
        {
            vis[s[top--]]=1;
            cnt++;
        }

        if(cnt&1)vis[s[top+1]]=0;
    }
}

int get_sg(int u,int pre)
{
    int ret=0;
    for(int i=0;i<(int)g[u].size();i++)
    {
        int v=g[u][i];
        if(v==pre || vis[v])continue;
        ret^=(get_sg(v,u)+1);
    }
    return ret;
}

int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
```

```
    cout.tie(0);

    int n,m,k;
    while(cin>>n)
    {
        int ret=0;
        while(n--)
        {
            cin>>k>>m;
            memset(vis,0,sizeof(vis));
            memset(mp,0,sizeof(mp));
            memset(dfn,0,sizeof(dfn));
            memset(low,0,sizeof(low));
            for(int i=0;i<=k;i++)g[i].clear();
            //memset(ins,0,sizeof(ins));
            for(int i=1,u,v;i<=m;i++)
            {
                cin>>u>>v;

                mp[u][v]++;
                mp[v][u]++;
                g[u].push_back(v);
                g[v].push_back(u);
            }
            tarjan(1,0);//cout<<"OK"<<endl;
            ret^=get_sg(1,0);
        }
        //cout<<"#"<<ret<<endl;
        cout<<(ret?"Sally":"Harry")<<endl;
    }
    return 0;
}
```

Coins Game

N 枚硬币排一排，编号 1~N，游戏者根据某些约束翻硬币，但所有翻动中最左或最右或中间某一位置必须是从正面翻至反面，谁不能翻谁败.

结论：局面的 SG 值为局面中每个正面朝上的棋子单一存在时的 SG 值的异或和

反 正 正 反 反 正

= 反 正

^ 反 反 正

^ 反 反 反 反 反 正