博弈论(二)

前置芝士——SG函数

可以看我上一篇博客：https://blog.csdn.net/SC_Linno/article/details/121181361

首先给出一种ICG博弈游戏模型，给定一个有向无环图和一个起始顶点上的一枚棋子，两名选手交替的将这枚棋子沿有向边进行移动，无法移动者判负。

将ICG问题进行转换：任何一个ICG都可以通过把每个局面看成一个顶点，对每个局面和它的子局面连一条有向边来抽象成这个“有向图游戏”。

于是我们可以通过将ICG问题转换为上述这个游戏，再通过寻找这个游戏的一遍解法来解决ICG问题。

首先定义mex(minimal excludant)运算，这是施加于一个集合的运算，表示最小的不属于这个集合的非负整数。例如mex{0,1,2,4}=3,mex{2,3,5}=0,mex{}=0;

对于一个给定的有向无环图，定义关于图的每个顶点的SG函数如下：

sg(x)=mex{sg(y)|y是x的后继}

步骤

一、找出必败态（SG值为0）

二、找到当前所有状态的前驱节点

三、根据定义计算节点SG值

重复上述步骤，直到整棵树建立完成

SG定理

游戏的和的SG函数值是它的所有子游戏的SG函数值的异或。

因此，当我们面对n个不同的游戏组成的游戏时，只需求出每个游戏的SG函数值把这些SG值全部看成Nim的石子堆，然后依照找Nim的必胜策略的方法来找这个游戏的必胜策略。

翻硬币问题

问题描述

N枚硬币排成一排，有的正面朝上，有的反面朝上。游戏者轮流根据某种约束翻硬币（每次只能翻一或两枚，或者只能翻连续的几枚），谁不能翻谁输。

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。对于任意一个硬币的SG值为2^k（k为硬币编号）

树上删边

Green Hachenbush（树上公平删边游戏）

双方轮流在一棵树删删边，不再与根节点相连的部分会被移除。游戏中存在多棵树，最后无法删边的玩家失败。

如果这个游戏中所有的树都是没有分支的"竹子"，那么就变成了普通的Nim游戏，此时SG[x]=x。当我们知道了克朗原理，我们就可以将所有分支一根竹子，就转化成了普通的Nim游戏了。

克朗原理（Colon Principle）

对于树上的某一个点，ta的分支可以转化成以这个点为根的一根竹子，这个竹子的长度就是**ta各个分支的边的数量的异或和。**

图上删边

费森原理

（环上的点可以融合，且不改变图的SG值。）

一般来说，我们可以把一个带有奇数边的环等价成一个端点和一条边，而偶数边的环等价于一个点。

Christmas Game

```
#include<iostream>
#include<cstring>
#include<vector>
using namespace std;
```

```cpp
const int N=1007;
const int mod=1e9+7;

vector<int>G[N];
inline void addedge(int u,int v){
    G[u].push_back(v);
    G[v].push_back(u);
}

int n,m,u,v,k,vis[N],SG[N];
int stk[N],top;

void dfs(int x,int fa){
    stk[++top]=x;
    vis[x]=1;
    SG[x]=0;
    bool flag=0;
    for(int i=0;i<G[x].size();i++){
        int to=G[x][i];
        if(to==fa&&!flag){flag=1;continue;} //第一次连向父节点
        if(vis[to]==1){
            int cnt=1,y=x;
            while(y!=to) cnt++,vis[y]=-1,y=stk[--top];
            if(cnt&1) SG[y]^=1; //奇环
        }else if(!vis[to]){
            dfs(to,x);
            if(~vis[to]) SG[x]^=(SG[to]+1); //不在环上的才能更新
        }
    }
    if(~vis[x]) --top;
}

signed main(){
    while(cin>>n){
        int ans=0;
        for(int p=1;p<=n;p++){
            cin>>m>>k;
            for(int i=1;i<=m;i++) G[i].clear(),vis[i]=0;
            for(int i=1;i<=k;i++){
                cin>>u>>v;
                addedge(u,v);
            }
            dfs(1,0);
            ans^=SG[1];
        }
        if(ans) puts("Sally");
        else puts("Harry");

    }
    return 0;
}
```

「一本通 6.7 例 3」移棋子游戏

给定一个有 个节点的有向无环图，图中某些节点上有棋子，两名玩家交替移动棋子。

玩家每一步可将任意一颗棋子沿一条有向边移动到另一个点，无法移动者输掉游戏。

对于给定的图和棋子初始位置，双方都会采取最优的行动，询问先手必胜还是先手必败。

因为是一张有向无环图，正好对应有向图游戏和模型，可以利用sg函数异或和求解。我们发现，出度为0的点，不能再移动，是P点，所以其sg函数必是0。其它点的sg函数都可以由其儿子节点推出。所以我们对整张图dfs就可以求得所有点的sg函数(图可能要多次dfs)。然后求对应点sg函数异或和，非0则先手必胜，0则先手必败。

```
#include<bits/stdc++.h>
using namespace std;
const int N=2007;

int n,m,k,sg[N],res=0;
vector<int>G[N];

inline int dfs(int x){ //sg函数
    if(sg[x]!=-1) return sg[x];
    bitset<N>vis;
    int flag=0,ma=-1;
    for(auto to:G[x]){
        flag=1;
        int w=dfs(to);
        ma=max(ma,w);
        vis[w]=1;
    }
    if(!flag) return sg[x]=0;
    for(int i=0;i<=ma+1;++i){
        if(!vis[i]){
            sg[x]=i;
            break;
        }
    }
    return sg[x];
}
signed main(){
    cin>>n>>m>>k;
    memset(sg,-1,sizeof(sg));
    for(int i=1,u,v;i<=m;++i){
        cin>>u>>v;
        G[u].emplace_back(v);
    }
    for(int i=1;i<=n;++i){
        if(sg[i]==-1){
            dfs(i);
        }
    }
    for(int i=1,x;i<=k;++i){
        cin>>x;
        res^=sg[x];
    }
    if(res) cout<<"win\n";
    else cout<<"lose\n";
    return 0;
```

```
}
```

「一本通 6.7 练习 1」取石子游戏

取石子游戏的规则是这样的，每个人每次可以从一堆石子中取出若干个石子，每次取石子的个数有限制，谁不能取石子时就会输掉游戏。小 H 先进行操作，他想问你他是否有必胜策略，如果有，第一步如何取石子。

思路
每次取完石子就到一个新的状态，状态图是一个有向无环图，可以利用sg函数解决，如果必胜的话，最小的第一次取可以直接暴力枚举判断得到。

```
#include<bits/stdc++.h>
using namespace std;
const int N=1e5+7;

int n,m,mx,vis[N],sg[N],a[N],b[N],c[N];

void get_SG(){
    sg[0]=0;
    for(int i=1;i<=mx;++i){
        int ma=-1;
        memset(vis,0,sizeof(vis));
        for(int j=1;j<=m&&(i-b[j])>=0;++j){
            vis[sg[i-b[j]]]=1;
            ma=max(ma,sg[i-b[j]]);
        }
        int j=0;
        while(vis[j]) ++j;
        sg[i]=j;
    }
}

int tmp[15];
bool check(int x,int y){
    for(int i=1;i<=n;++i) tmp[i]=a[i];
    tmp[x]-=y;
    int ans=0;
    for(int i=1;i<=n;++i) ans^=sg[tmp[i]];
    if(ans) return false;
    else return true;
}

signed main(){
    cin>>n;
    for(int i=1;i<=n;++i) cin>>a[i],mx=max(mx,a[i]);
    cin>>m;
    for(int i=1;i<=m;++i) cin>>b[i],c[b[i]]=1;
    get_SG();
    int ans=0;
    for(int i=1;i<=n;++i) ans^=sg[a[i]];
    if(!ans) cout<<"NO\n";
    else{
        cout<<"YES\n";
        bool flag=0;
```

```
        for(int i=1;i<=n;++i){
            for(int j=1;j<=a[i];++j){
                if(check(i,j)&&c[j]){
                    flag=1;
                    cout<<i<<" "<<j<<"\n";
                    break;
                }
            }
            if(flag) break;
        }
    }
    return 0;
}
```

参考资料

https://blog.csdn.net/wu_tongtong/article/details/79311284

https://www.cnblogs.com/maoyiting/p/14169209.html#/cnblog/works/article/14169209

《组合游戏略述——浅谈 SG 游戏的若干拓展及变形》

https://blog.csdn.net/weixin_44491423/article/details/108433347

https://blog.csdn.net/weixin_44491423/article/details/108435475