SG函数和SG定理的运用

SG函数和SG定理常用于解决博弈论的相关问题。**其中SG函数的求解主要根据MEX运算**。什么是MEX运算？MEX（ minimal excludant ） 字面上意思是最小除外的那个数。 MEX是对一个集合的运算。指得是对于一个集合 s={a1,a2,,,,,an) 集合中未出现的最小非负整数。 举个例子：比如 s={1,2,3} 则MEX(s)=0, 如果s={0,1,2,4},则MEX(s)=3.
SG定理：博弈论中游戏和的SG值等于各个游戏的SG值的Nim和。Nim和也就是我们所说的按位异或运算下面上例题。

1.Fibonacci again and again

题目传送门：HDU1848 详细解释见下面代码。

```
xxxxxxxxxx
#include<bits/stdc++.h>
using namespace std;
const int N=1e3+5;
#define mst(a) memset(a,0,sizeof s)
int f[16],sg[N],s[N],m,n,p; //f[i]储存改变状态的所有方式  sg[i]储存每个状态的sg值  ,s[i]保存后继状态所有sg值
void solve(){
    for(int i=1;i<=N;i++) //遍历
    {
        mst(s);//每次都要对保存后继状态的集合s清空
        for(int j=1;j<=16&&f[j]<=i;j++)//对所有满足条件的后继状态的sg值放入到集合s中
            s[sg[i-f[j]]]=1;
        for(int k=0;;k++)
            if(!s[k])   //进行mex运算找到最小未出现的非负整数 即为sg值
            {
                sg[i]=k;
                break;
            }
    }
}
int main(){
    f[1]=1,f[2]=2;
    for(int i=3;i<=16;i++)//fibnacci递推
        f[i]=f[i-1]+f[i-2];
    solve();
    while(cin>>m>>n>>p){
        if(!m&&!n&&!p) break;
        puts((sg[m]^sg[n]^sg[p])?"Fibo":"Nacci");//Nim和运算 若非0先手胜，反之后手胜。
    }
    return 0;
}
```

2.巴什博弈的SG解法

如果对于不知道巴什博弈结论的人可以用SG函数直接求解 不过时间复杂度变成了O（nm）数据太大就只能记结论了。

```
xxxxxxxxxx
#include<bits/stdc++.h>
using namespace std;
const int N=1e3+5;
int sg[N],s[N];
#define mst(a) memset(a,0,sizeof a)
void solve(int n,int m){   //时间复杂度O(nm)
    for(int i=1;i<=n;i++)
```

```
    {
        mst(s);
        for(int j=1;j<=i&&j<=m;j++)
            s[sg[i-j]]=1;
        for(int k=0;;k++)
            if(!s[k])
            {
                sg[i]=k;
                break;
            }
    }
}
int main(){
    int t,n,m;
    cin>>t;
    while(t--){
        cin>>n>>m;
        solve(n,m);
        puts(sg[n]?"first":"second");
    }
    return 0;
}
```

3.尼姆博弈的SG解法

```
xxxxxxxxxx
#include<bits/stdc++.h>
using namespace std;
const int N=1e4+5;
int sg[N],s[N],n,a[N];
void sovle(){
    for(int i=1;i<N;i++)
    {
        memset(s,0,sizeof s);
        for(int j=1;j<=i;j++)
             s[sg[i-j]]=1;
        for(int k=0;;k++)
            if(!s[k])
            {
                sg[i]=k;
                break;
            }
    }
}
int main(){
    int t;
    cin>>t;
    while(t--){
        cin>>n;
        int ans=0;
        for(int i=1;i<=n;i++)
            cin>>a[i],ans^=a[i];
        puts(ans?"Yes":"No");
    }
```

```
    return 0;
}
```

4.Roy&October之取石子II

题目传送门：P4860 由题目数据我们可知数据很大，显然用SG是不可能的。但是如果我们没找到规律， 就可以用SG函数进行打表，通过打表发现的规律的可能性更大。下面给个打表代码。

```
xxxxxxxxxx
#include<bits/stdc++.h>
using namespace std;
const int N=5e4+5,M=1e5+5;
int p[M],cnt=1,a[N];
void ss(int n){   //欧拉筛
    for(int i=2;i<=n;i++)
        a[i]=1;
    for(int i=2;i<=n;i++)
    {
        if(a[i]) p[cnt++]=i;
        for(int j=1;j<cnt&&p[j]*i<=n;j++)
        {
            a[i*p[j]]=0;//最小质因数筛合数。
            if(i%p[j]==0) break;
        }
     }
}
int sg[N],s[N];
void solve(){ //求sg函数
    p[0]=1;
    for(int i=1;i<N;i++)
    {
        memset(s,0,sizeof s);
        for(int j=0;p[j]<=i&&j<cnt;j++)
            s[sg[i-p[j]]]=1;
        for(int k=0;;k++)
            if(!s[k])
            {
                sg[i]=k;
                break;
            }
    }
}
int main(){
    ios::sync_with_stdio(false),cin.tie(0);
    ss(N);
    solve();
        for(int i=1;i<=100;i++)
        if(sg[i]) printf("%d:1\n",i);
        else printf("%d:0\n",i);
    return 0;
}
```

```
1:1
2:1
3:1
4:0
5:1
6:1
7:1
8:0
9:1
10:1
11:1
12:0
13:1
14:1
15:1
16:0
17:1
18:1
19:1
20:0
21:1
22:1
23:1
24:0
25:1
26:1
27:1
28:0
29:1
30:1
```


从表中我们知道凡是4的倍数都是后手胜。因此我们可以猜想只要n是4的倍数后手胜，否则先手胜。下面上正解代码

事实上，我们自己找规律也能找出来，比如n=1，2，3显然都是先手胜，n=4时，显然先手只能拿1或2或3个，所以n=4为必败态，n=5,6,7时先手都可以通过拿若干个棋子使对手面临n=4这个必败态，所以n=5,6,7为必胜态。 n=8时，先手只能拿1，2，3，5，7 显然后手胜，所以n=8为必败态，依次类推，规律显然。

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int t,n;
    cin>>t;
    while(t--){
        cin>>n;
        puts(n%4?"October wins!":"Roy wins!");
    }
    return 0;
```

}