

# 博弈论和 SG 函数

## 必胜点和必败点

- $P$  点：必败点，换言之，就是谁处于此位置，则在双方操作正确的情况下必败。
- $N$  点：必胜点，处于此情况下，双方操作均正确的情况下必胜。

必胜点和必败点的性质：

- 所有终结点是必败点  $P$
- 从任何必胜点  $N$  操作，至少有一种方式可以进入必败点  $P$
- 无论如何操作，必败点  $P$  都只能进入必胜点  $N$

## NIM 游戏

两个人玩这个游戏，他们轮流操作。

有若干堆石子，每堆石子的数量都是有限的。

一次合法的移动是“选择一堆石子并拿走若干颗（不能不拿）”。

如果轮到某个人时所有的石子堆都已经被拿空了，则判负。

如果双方都按照最优策略，谁必胜？

## Bouton's Theorem

对于一个 nim 游戏的局面  $(a_1, a_2, \dots, a_n)$  它是  $P$  点当且仅当  $a_1 \oplus a_2 \oplus \dots \oplus a_n = 0$  【证明】

1. 终结点只有一种，就是  $(0, 0, \dots, 0)$  显然符合异或和为  $0$ ，为  $P$  点。
2. 对于  $(a_1, a_2, \dots, a_n)$  且  $a_1 \oplus a_2 \oplus \dots \oplus a_n = 0$  经一次移动后必然到达  $(b_1, b_2, \dots, b_n)$  其中  $b_1 \oplus b_2 \oplus \dots \oplus b_n \neq 0$  从而到达  $N$  点。
3. 对于  $(a_1, a_2, \dots, a_n)$  且  $a_1 \oplus a_2 \oplus \dots \oplus a_n \neq 0$  必存在移动方法可以到达  $P$  点。

我们设  $a_1 \oplus a_2 \oplus \dots \oplus a_n = k$  那么设  $k$  的二进制表示下最高位的  $1$  为第  $p$  位。

那么  $(a_1, a_2, \dots, a_n)$  中必定存在至少一个  $a_i$  使得  $a_i$  二进制表示下第  $p$  位为  $1$ 。

从而，将第  $i$  堆石头取  $a_i - a_i \oplus k$  个石头即可保证一定到达  $P$  点。

首先，由于  $a_i \oplus k$  第  $p$  位为  $0$ ，所以  $a_i \oplus k < a_i$  从而  $a_i - a_i \oplus k > 0$  符合游戏规则。

并且，取  $a_i - a_i \oplus k$  个石头后，第  $i$  堆石头变为  $a_i \oplus k$  对于新局面  $(a_1, a_2, \dots, a_i \oplus k, \dots, a_n)$   $a_1 \oplus a_2 \oplus \dots \oplus (a_i \oplus k) \oplus \dots \oplus a_n = k \oplus k = 0$  从而一定为  $P$  点。

## 有向图移动游戏

有向图移动游戏可以看作所有 *Impartial Combinatorial Games* 的抽象模型。

**NIM** 游戏就是 *Impartial Combinatorial Games* 其中的一种。

也就是说，所有 **ICG** 游戏都可以看成：

给定一个 **DAG** 及一个点上的一个棋子，两名选手交替将棋子沿边移动，无法移动判负。

我们把 **NIM** 游戏的每一个状态看成一个点，把这个状态和其可以转移到的下一个状态连边。那么 **NIM** 游戏也被抽象成了一个有向图移动游戏！

### SG 函数

定义  $\text{mex}(S)=k$  为最小的不属于集合  $S$  的非负整数。

$\text{SG}$  函数的定义：对于任意一个状态  $x$  都定义一个  $\text{SG}$  函数。

我们先给出定义式，再具体说明意义。

对于任意状态  $x$  设  $x$  的后继状态集合为  $S$  则： $\text{sg}(x)=\text{mex}(S)$  如果一个状态为终结点，则  $S=\emptyset$  从而  $\text{sg}(x)=0$

### 有向图移动游戏

事实上，如果把所有 **ICG** 游戏抽象成有向图移动游戏，那么  $\text{sg}$  函数就是  $\text{sg}(x)=\text{mex}\{\text{sg}(y)\mid x\rightarrow y\}$  我们有这样的结论  $\begin{cases} \text{sg}(x)=0\Leftarrow x\text{ 是 }P \\ \text{sg}(x)\neq 0\Leftarrow x\text{ 是 }N \end{cases}$  对于这个结论的正确性显然：

对于  $\text{sg}(x)=0$  的结点，显然根据定义  $x$  的后继中一定不存在  $\text{sg}(y)=0$  的结点  $y$

同时，对于  $\text{sg}(x)\neq 0$  的结点，根据定义，一定存在一个  $x$  的后继  $y$  使  $\text{sg}(y)=0$

### 取石子游戏

两个人取石子，每个人可以取  $1,3,4$  个石子。

共有  $n$  个石子，求是先手必胜还是后手必胜。

$$\text{sg}(0)=0,\text{sg}(i)=\text{mex}\{\text{sg}(i-1),\text{sg}(i-3),\text{sg}(i-4)\}$$

### SG 定理

假设一个游戏可以分成若干个子游戏，这些子游戏的  $\text{sg}$  函数值为  $s_1,s_2,\dots,s_k$  则：整个游戏

的  $sg$  函数为  $sg(All)=s_1 \oplus s_2 \oplus \dots \oplus s_k$  我们设  $sg(x)=a$  那么也就是说  $x$  的后继结点  $y$  能取遍  $1, 2, \dots, a-1$

那么我们选取后继，事实上可以看成“取石子”的过程。

这样想的话，就可以利用 **Bouton's Theorem** 的证明来理解  $SG$  定理了。

## 例题

### HDU 1848 Fibonacci again and again

#### 题意

- 这是一个二人游戏，两人轮流走；
- 一共有 3 堆石子，数量分别是  $m, n, p$  个；
- 每走一步可以选择任意一堆石子，然后取走  $f$  个；
- $f$  只能是斐波那契数列中的元素；
- 最先取光所有石子的人为胜者；
- 假设双方都使用最优策略，请判断先手的人会赢还是后手的人会赢；
- $0 \leq n, m, p \leq 1000$

#### 题解

分成三个子游戏，分别求出每个子游戏的  $SG$  函数，异或得总游戏  $SG$  函数即可。

是一个  $SG$  函数和  $SG$  定理的简单应用。

#### 代码

```
#include<bits/stdc++.h>
using namespace std;
int f[1005],p;
bool s[1005];
int sg[1005];
void SG(){
    for(int i=1;i<=1000;i++){
        memset(s,0,sizeof(s));
        for(int j=1;j<=p;j++){
            if(f[j]>i) break;
            s[sg[i-f[j]]]=1;
        }
        for(int j=0;j<=1000;j++){
            if(!s[j]){
                sg[i]=j;
                break;
            }
        }
    }
}
int main(){
```

```
f[1]=1,f[2]=2;
for(int i=3;;i++){
    f[i]=f[i-1]+f[i-2];
    if(f[i]>1000){
        p=i-1;
        break;
    }
}
SG();
int m,n,p;
while(scanf("%d %d %d",&m,&n,&p)){
    if(!m&&!n&&!p) break;
    int ret=0;
    ret=sg[m]^sg[n]^sg[p];
    if(ret) puts("Fibo");
    else puts("Nacci");
}
return 0;
}
```

## HackerRank Bob and Ben

### 题意

给出一片森林，每棵树有两个参数，结点数  $n$  和特殊参数  $k$ 。其中  $k$  意义为：第  $i$  个结点的父亲为第  $\max(1, \lfloor \frac{i}{k} \rfloor)$  个结点。两人进行游戏，每次可以删除一棵树（该树必须存在非叶子）或树中的一个叶子。其中，叶子定义为度数为  $1$  的点。无法操作的人输，询问先手是否必胜。

### 题解

考虑一棵大小为  $n$  的树。

当  $n=1$  时， $sg(1)=1$

当  $n=2$  时， $sg(2)=0$

当  $n \geq 3$  时，一定存在非叶子结点  $sg(n)=\text{mex}\{sg(n-1), 0\}$

归纳知  $n \geq 3$  时， $sg(2k)=2, sg(2k+1)=1$

利用  $SG$  定理合并即可。

（事实上，我们发现此题中  $k$  并没有作用）

### 代码

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int t;
```

```

scanf("%d",&t);
while(t--){
    int m;
    scanf("%d",&m);
    int ret=0;
    for(int i=1;i<=m;i++){
        int n,k;
        scanf("%d %d",&n,&k);
        if(n==1) ret^=1;
        else if(n==2) ret^=0;
        else if(n&1) ret^=1;
        else ret^=2;
    }
    if(ret) puts("BOB");
    else puts("BEN");
}
return 0;
}

```

## HDU 6892 Lunch

### 题意

有  $n$  堆石头，每堆有  $m_i$  个石头，两个人轮流进行操作，如果有谁不能操作了，则判负。操作为：选择其中一个堆，将这个堆分为  $t(t \neq 1)$  堆，且每堆的石头数量相同。

### 题解

通过打表找规律发现结论  $sg(x) = x$  的奇质因子个数  $+ [x \% 2 == 0]$

### 打表代码

```

#include<iostream>
#include<algorithm>
#include<cstdio>
#include<cstring>
#include<stack>
#include<vector>
#include<unordered_set>
#include<unordered_map>
using namespace std;

typedef long long ll;
const int maxn=4e4+10;
const int maxm=1e4+10;
#define ios ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);
int n,m;
int f[maxm];

int sg(int x){

```

```
if(f[x]!=-1) return f[x];
unordered_set<int> S;
vector<int> q;
for(ll i=2;i<=sqrt(x);++i){
    if(x%i==0){
        q.push_back(i);
        if(i*i!=x) q.push_back(x/i);
    }
}
q.push_back(x);
for(int i=0;i<q.size();++i)
{
    if(q[i]%2==0) S.insert(0);
    else S.insert(sg(x/q[i]));
}
for(int i=0;;++i){
    if(!S.count(i)) return f[x] = i;
}
}

int main(void)
{
    memset(f, -1, sizeof(f));
    f[1]=0;
    for(int i=1;i<=30;i++) cout << i << ' ' << sg(i) << " ";
}
```

## AC 代码

```
#include<bits/stdc++.h>
using namespace std;
const int N=3.2e4+5;
int p[N];
bool b[N];
int SG(int x){
    int ret=0,cnt=0;
    if(x%2==0) ret=1;
    for(int i=1;i<=p[0];i++){
        if(x%p[i]==0){
            while(x%p[i]==0){
                x/=p[i];
                if(p[i]!=2) cnt++;
            }
        }
        if(x==1) break;
    }
    if(x!=1) cnt++;
    return cnt+ret;
}
```

```

}
int main(){
    for(int i=2;i<N;i++){
        if(!b[i]) p[++p[0]]=i;
        for(int j=1;j<=p[0]&& i*p[j]<N;j++){
            b[i*p[j]]=1;
            if(i%p[j]==0) break;
        }
    }
    int t;
    scanf("%d",&t);
    while(t--){
        int n;
        scanf("%d",&n);
        int ret=0;
        for(int i=1;i<=n;i++){
            int x;
            scanf("%d",&x);
            ret^=SG(x);
        }
        if(ret) puts("W");
        else puts("L");
    }
    return 0;
}

```

## 威佐夫博弈

### 简介

两个玩家轮流行动，在两堆石子中选一堆取走任意个，或同时在两堆石子中取走相等的石子数，最后取光所有石子的人获胜。这个游戏的一个等价描述是：一个棋子放在一个大棋盘上，两人轮流移动棋子，向下，向左或向左下移动任意步，胜者是将棋移动至原点的人。

### 最优策略

游戏中的任一状态可由一对整数描述  $(n,m)(n \leq m)$ 。游戏中的状态点分为必败点和必胜点。在必胜点的最优策略是移动至任一可达的必败点。必败点和必胜点的分类由以下三条规则递归给出：

1.  $(0,0)$  是必败点；
2. 可一步走到必败点的点是必胜点；
3. 如果无论怎样走都只能到达必胜点，则该点是必败点。

前几个必败点是： $(0,0), (1,2), (3,5), (4,7), (6,10), (8,13)$ 。

变形：最后一步移动的人为败者

$(0,1), (2,2)$  是必败点  $(n,m) (2 < n \leq m)$  是必败点当且仅当  $(n-2, m-2)$  在正常游戏中是必败点。

## 必败点的判定准则

$\phi = \frac{\sqrt{5}+1}{2}$  第  $k$  个必败点  $(n_k, m_k)$   $n_k = \lfloor k\phi \rfloor, m_k = \lfloor k\phi^2 \rfloor$  若给定  $(n,m)$  判断其是否为必败点，则判断  $\lfloor (m-n)\phi \rfloor = n$  必要时使用  $\lfloor \sqrt{5(m-n)^2} \rfloor = 3n-m$  来判定（二分答案算根号）。

## 多于两堆的情况

玩家可任选一堆石子取走任意数量，当选择多于一堆石子时，则每堆取走的石子数需相同。

例如， $(1,1,3)$  和  $(1,2,3)$  是必胜点，因为它们能到达必败点  $(0,1,2)$ 。 $(1,1,4), (1,3,3)$  是必败点。

From: <https://wiki.cvbbacm.com/> - CVBB ACM Team

Permanent link: [https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal\\_string:lgwza:%E5%8D%9A%E5%BC%88%E8%AE%BA&rev=1626618250](https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:lgwza:%E5%8D%9A%E5%BC%88%E8%AE%BA&rev=1626618250)

Last update: 2021/07/18 22:24

