博弈论

博弈图

定理1：没有后继状态的状态是必败状态

定理2：一个状态是必胜状态当且仅当存在至少一个必败状态为它的后继状态

定理3：一个状态是必败状态当且仅当它的所有后继状态均为必胜状态

如果博弈图是一个有向无环图，则我们在绘制出博弈图的情况下用 $O(N+M)$ 的时间得出每个状态是必胜状态还是必败状态。其中 N 为状态种数 M 为边数。

对于 nim 游戏，当异或和为 0 时，状态为必败状态，否则为必胜状态。显然对于异或和为 0 的局面，一定不存在某一种移动让异或和仍然为 0。而对于异或和不为 0 的局面，我们总有某种移动让异或和变为 0。如果能证明这个，我们就能证明最终结论。

比如现在所有石子异或和为 t 我们显然要对 t 做二进制分解，取出最高位，所以原来的石子中，一定有奇数堆石子二进制这一位为 1，我们任选一堆，取这一堆，让最高位为 0，之后对于原来的 t 哪一位为 1，就让取的这一堆剩余的二进制和原来的二进制相反就可以了，可以知道这个数一定存在，且一定比原来少（因为刚才的最高位被取没了，且更高的位没动，所以一定变少）。所以小定理得证，大定理也就得证。其实设取的这一堆原来是 a_{i} 则 $a_{i}'=a_{i}$ ^ t

有向图游戏与 SG 函数

在一个有向无环图中，只有一个起点，上面有一个棋子，两个玩家可以沿着有向边，轮流移动棋子，不能移动的玩家判负。

定义 mex 函数的值为不属于集合 S 中的最小非负整数。

对于状态 x 和它的所有 k 个后继状态 $y_{1},y_{2},...,y_{k}$ 定义 SG 函数：

$SG(x)=mex\{SG(y_{1}),SG(y_{2}),...,SG(y_{k})\}$

SG 值为 0 是必败态。

对于由 n 个有向图游戏组成的游戏，设起点分别为 $s_{1},s_{2},...,s_{n}$ 则有定理，当且仅当 $SG(s_{1})$ ^ $SG(s_{2})$ ^ $...$ ^ $SG(s_{n})≠0$ 时，先手必胜，同时这是组合游戏状态 x 的 SG 值。

此证明省去。

对于 nim 游戏，一个堆取走任意不为 0 的数量，都会变成后继状态，所以 $SG(x)=x$ 即可满足题意，也就验证了上面的结论。

$Anti-SG$ 游戏

$Anti-SG$ 游戏规定，决策集合为空的游戏者赢。

$Anti-SG$ 游戏其他规则与 SG 游戏相同。

$Anti-Nim$ 游戏

现在我们把取完石子定为失败状态。

显然当每堆只有一个石子时，石子为偶数堆也就是异或和为 0 时，先手必胜，奇数堆时先手必败。

当有一堆石子不为 1 时，显然先手可以控制将这堆石子取剩下 1 或者 0，来控制 1 的堆数，根据上面的推论，先手必胜。

当至少有两堆石子数大于 1 时，还是考虑异或和，举了两个特例，提出异或和为 0 时，先手必败，不为 0 时，先手必胜的猜想。不为 0 的情况先手必胜和为 0 先手必败可以看成等价，所以只证明后者即可。

注意到当剩余大于 1 的石子堆数为 1 时，异或和不可能为 0。当后手面对异或和为 0 的情况，只要他操作之后大于 1 的石子堆数大于 1，只需要将异或和还原为 0 即可。步数有限，后手早晚要面对到大于 1 的石子堆数等于 2，且不得不将其中一堆石子变为 1 的情况，这时根据情况 2，先手必胜，证毕。

SJ 定理

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（对应 $Anti-Nim$ 游戏中的石子数大于 1 的堆数只有一堆的情况，和有至少两堆石子数大于 1，且石子异或和不为 0）

2.游戏的 SG 函数为 0 且游戏中没有单一游戏的 SG 函数大于 1（所有石子堆均为 1 个石子，这时候偶数堆必胜）

$Every-SG$ 游戏

$Every-SG$ 游戏规定，每轮操作要对所有单一游戏进行操作，其他规则和普通 SG 游戏相同。

显然有一个贪心，对于先手而言，先手希望自己必胜的小游戏进行的轮数尽量多，必败的小游戏尽量短。也就是说对于 SG 值为 0 的点，我们需要知道最快几步能让游戏终止，对于 SG 不为 0 的点，我们需要知道最慢几步游戏终止，用 $step$ 函数来表示这个值。

$step(u)=0$ □如果 u 是终止状态。

$step(u)=max\{step(v)\}+1$ □ $sg(u)≠0 ∧ v$ 为 u 的后继 $ ∧ sg(v)=0$

$step(u)=min\{step(v)\}+1$ □ $sg(u)=0 ∧ v$ 为 u 的后继

我们有定理：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。证明显然。

$Multi-SG$ 游戏

$Multi-SG$ 游戏指在符合拓扑原则的前提下，一个单一游戏的后继可以为多个单一游戏，其他规则和 SG 游戏相同。

比如 $Multi-Nim$ 游戏，定义为：有 n 堆石子，两个人可以从任意一堆石子中多拿任意多个石子（不能不拿）或把一堆数量不少于 2 石子分为两堆不为空的石子，没发拿的人失败。

操作一和普通的 Nim 游戏等价。操作二实际上是将一个游戏分为两个游戏，于是相比于最开始的 Nim 游戏，我们增加了若干个操作二带来的后继，可以将这两个游戏的 sg 值异或起来作为一个新的游戏后继，再对所有后继求一个 mex 就知道现在这个点的 sg 值了，这个可以打表解决。

```
int sg[101],vis[305];
int main() {
    sg[0]=0,sg[1]=1;
    for(int i=2;i<=100;i++) {
        memset(vis,0,sizeof(vis));
        for(int j=0;j<i;j++) vis[sg[j]]=1;
        for(int j=1;j<=i/2;j++) vis[sg[j]^sg[i-j]]=1;
        for(int j=0;j<305;j++) {
            if(!vis[j]) {
                sg[i]=j;
                printf("%d %d\n",i,j);
                break;
            }
        }
    }
    return 0;
}
```

对于这道题而言，我们可以找到规律：

$SG(x)=x-1(x\ mod\ 4=0)$

$SG(x)=x(x\ mod\ 4=1||x\ mod\ 4=2)$

$SG(x)=x+1(x\ mod\ 4=3)$

可以背下来，但是没必要 $\times$

翻硬币游戏

一般的游戏规则是这样的 N 枚硬币排成一排，有的正面向上，有的反面向上，从左到右按照 1 到 N 编号。

游戏者按照某些约束翻硬币，但是他翻动的硬币中，最右边的必须是从正面到反面，谁不能翻谁输。

有一个结论，不管限制是怎么样的，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。比如现在有 $DUUDDU$ U 代表证明朝上 D 代表反面朝上。这等价于 $DUDDDD+DDUDDD+DDDDDU$ 三个游戏的和，同时 sg 值有 $sg[DUUDDU]=sg[DUDDDD]^sg[DDUDDD]^sg[DDDDDU]$ 所以重点是如何求单个硬币朝上的 sg 值。

条件一：每次只能翻一个。

显然这种情况下，每个的异或值等同于朝上硬币数的奇偶。

条件二：每次能翻转一个或两个硬币（不用连续）。

每个硬币的 sg 值为它的编号，当前局面的 sg 值是所有正面向上的硬币编号异或和。

类似于 Nim 游戏，我们总能让一个异或和不为零的局面异或和变为零，最终局面的异或值为 0 。

对应于游戏中，比如当前异或和为 $x≠0$ 我们取 x 二进制为 1 的最高位，再在所有 sg 值这一位为 1 的数中任选一个，将其变为 t^x 。

如果 t^x 在 sg 值中没出现，则相当于是翻一上和一下，且上的编号一定大于下，所以满足题意。

如果出现了，就相当于将两个向上的硬币翻到下面。

如果 $t^x=0$ 则只翻这一个硬币，其余不动。

条件三：每次必须连续翻转 k 个硬币

比如 $k=3$ 。

由上，只考虑单枚硬币向上的情况。显然 $sg[1]=sg[2]=0,sg[3]=mex(sg[2]^sg[1])=1$

$N=3t$ 时，先手必赢 $sg[3t]=1$ 其余必输 $sg[3t+1]=sg[3t+2]=0$ 。

因为 $3t$ 翻完之后，会变成 $3t-1$ 和 $3t-2$ 这两种情况的异或和，也就是 $sg[3t]=mex(sg[3t-1]^sg[3t-2])=1$ 对于 $3t+1$ $sg[3t+1]=mex(sg[3t]^sg[3t-1])=mex(1)=0$ 对于 $3t+2$ 同理。

推广到别的值也成立。

条件四：每次翻动一个硬币后，必须翻动其左侧最近三个硬币中的一个，除非这个位置编号小于等于 3（编号小的话可以翻左边任意一个不一定是三个中的一个，也可以不翻）

$N=1$ 时，先手必胜 $sg[1]=1$

$N=2$ 时，先手必胜 $sg[2]=mex(sg[1],sg[0])=2$

$N=3$ 时，先手必胜 $sg[3]=mex(sg[0],sg[1],sg[2])=3$

$N=4$ 时，先手必败 $sg[4]=mex(sg[3],sg[2],sg[1])=0$

继续往下推可以发现是 $1\ 2\ 3\ 0$ 循环的，所以就推出来了。

条件五：每次可以翻动一个、两个或三个硬币

初始编号为 0

显然对于一个硬币的情况先手必胜

$N=1$ 时， $sg[0]=mex(0)=1$

$N=2$ 时， $sg[1]=mex(0,1)=2$

$N=3$ 时， $sg[2]=mex(0,1,2,1^2=3)=4$

$N=4$ 时， $sg[3]=mex(0,1,2,4,1^2=3,1^4=5,2^4=6)=7$

剩下太麻烦了，打个表

```
int sg[101],vis[1005];
int main() {
    sg[0]=1,sg[1]=2;
    for(int i=2; i<=200; i++) {
        memset(vis,0,sizeof(vis));
        vis[0]=1;
        for(int j=0; j<i; j++) {
            vis[sg[j]]=1;
            for(int k=j; k<i; k++) {
                vis[sg[j]^sg[k]]=1;
            }
        }
        for(int j=0; j<1005; j++) {
            if(!vis[j]) {
                sg[i]=j;
                printf("%d %d\n",i,j);
                break;
            }
        }
    }
    for(int i=0; i<=200; i++) {
        if(sg[i]==i*2) {
            printf("%d ",i);
            int j=i,cnt=0;
            while(j) {
                if(j&1) {
                    cnt++;
                }
                j>>=1;
            }
            if(cnt&1) printf("%d",);
            puts("");
```

```
            }

    }
    return 0;

}
```