SG函数

定义

SG函数用于解决公平组合游戏（ICG）的一种方法

ICG:

1、两位玩家

2、两位玩家轮流操作，在一个有限集合内任选一个操作，改变游戏当前的局面

3、一个局面的合法操作，只取决于游戏局面本身且固定存在，与玩家次序和其他任何因素无关

4、无法操作者，即操作集合为空，输掉游戏，另一方获胜

$SG函数的定义$:

对于一个ICG 可以将状态描述成一个图，边即为操作方式，那么对于 u

$$SG(u) = MEX(SG(son(u)))$$

$SG(x) = 0$ 的状态 x 称为必败态，$SG(x) \neq 0$ 的状态 x 被成为必胜态

SG定理

> $SG(x + y) = SG(x) \bigoplus SG(y)$

可以解决多个 IGC 组合起来的问题，例如最经典的Nim游戏:

> n 堆数，Alice 和 Bob 每次可以任意选取一堆取任意数，Alice先手，无法操作者输，问谁会赢

对于任意一堆数 x，
$SG(x) = MEX(SG(x-1), SG(X-2), SG(X-3), \dots\dots, SG(1), SG(0)) = x$

所以整个游戏的 $SG = SG(a_1) \bigoplus SG(a_2) \bigoplus SG(a_3) \bigoplus \dots\dots \bigoplus SG(a_n)$

即 $SG = a_1 \bigoplus a_2 \bigoplus a_3 \bigoplus a_4 \bigoplus \dots\dots \bigoplus a_n$

$SG = 0$ 即为先手必败，其余必胜。

经典模型

> 一堆大小为 n 的石子，每次取不超过 b 个，问Alice和Bob谁会赢

$SG(x) = MEX(j \in [1, b] SG(x - j))$

$SG(x) = n \mod (b + 1)$

> 每次取 $[l, r]$ 个，问Alice和Bob谁会赢

$SG(x) = MEX(j \in [l, r] SG(x - j))$

> 大小为 n 的石子，每次操作可以将其分成两堆，也可以拿掉任意数量的石子，问谁会赢

操作1：分成 i 和 $n - i$ 两堆，$SG(n) = SG(i) \bigoplus SG(n - i)$

操作2：拿掉 i 个石子，$SG(n) = SG(n - i)$

```
int sg[100];
```

```
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
    int n;
    cin >> n;
    set<int> s;
    for(int j = 0; j < n; j++) {
        s.insert(j);
    }
    for(int j = 1; j < n; j++) {
        s.insert(sg[j] ^ sg[n - j]);
    }
    while(s.count(sg[n])) sg[n]++;
    cout << sg[n] << endl;
    return 0;
}
```

打表找规律后，可以发现如下规律

```
int SG(int x)
{
    if(x % 4 == 1 || x % 4 == 2 || x == 0)
        return x;
    if(x % 4 == 3)
        return x + 1;
    return x - 1;
}
```

> 有一个$1 \times n$的长条，每次可以选择两个相邻的格子打✖，谁不能操作谁输，问谁会赢

$SG(i)$ 表示 $1 \times i$ 的长条的 SG 函数值

每次打叉后，一个长条分裂为长度为 j 和 $n-2-j$ 的两个长条

$SG(i) = SG(j) \bigoplus SG(i-j-2), j \in [0, i-2]$

```
    int n = 100;
    for(int i = 1; i <= n; i++) {
        set<int> s;
        for (int j = 0; j <= i - 2; j++) {
            s.insert(sg[j] ^ sg[i - 2 - j]);
        }
        while (s.count(sg[i])) sg[i]++;
        cout << sg[i] << endl;
    }
```

> $1 \times n$ 的长条，两人轮流打叉，谁无法操作谁输

一旦在某个位置打叉，则下一个人无法在左右两个位置继续打叉

则 $SG(i) = SG(j) \bigoplus SG(i-j-5)$

```
    int n = 100;
    for(int i = 1; i <= n; i++) {
        set<int> s;
        for(int j = 0; j <= i - 5; j++) {
            s.insert(sg[j] ^ sg[i - j - 5]);
        }
        if(i >= 3) s.insert(sg[i - 3]);
        if(i >= 4) s.insert(sg[i - 4]);
        while(s.count(sg[i])) sg[i]++;
        cout << i << " " << sg[i] << endl;
    }
```

阶梯Nim

> n 堆石子，两个人轮流取，每次可以在第 i 堆石子里面选取若干石头放到 $i-1$ 堆里面。谁不能操作算输（最终全部移到 0 上）

结论：奇数堆 SG 函数异或和

证明：将奇数位置当作一个 Nim 游戏，假设当前先手必胜，那么先手只要按照奇数位置的 Nim 游戏区进行就可以，后手若想赢则必须打破这个局面，只能将偶数位置的石子挪到奇数位置上，但先手仍能将后手移动的石子从奇数位置再挪到偶数位置上，对当前的局面不会造成任何影响。先手必败也是同理。

> n个格子，一些格子有石子，每次只能把一个石子朝左边移动一格，不能越过石子，不能操作者输。

把空格当作石子数，进行阶梯 Nim 游戏即可。

d 阶 Nim

> n 堆石子，两人轮流取，每次可以取 d 堆，谁不能操作谁输，问先手谁赢

结论：当且仅当在每一个不同的二进制位上，$x_1, x_2 \ldots x_k$ 中在该位上1的个数是 $N+1$ 的倍数时，后手方有必胜策略，否则先手必胜。

Anti-SG 博弈

> 有两个顶尖聪明的人在玩游戏，游戏规则是这样的：
>
> 有 n 堆石子，两个人可以从任意一堆石子中拿任意多个石子（不能不拿），拿走最后一堆石子的人失败，谁会赢？

SJ 定理 : 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 函数的值大于 1

2、 游戏的 SG 函数为 0，且游戏中没有单一游戏的 SG 函数大于 1