目录

0x00 公平组合游戏ICG

若一个游戏满足：

- 由两名玩家交替行动
- 在游戏进程的任意时刻，可以执行的合法行动与轮到哪名玩家无关
- 游戏中的同一个状态不可能多次抵达，游戏以玩家无法行动为结束，且游戏一定会在有限步后以非平局结束

则称该游戏为一个**公平组合游戏**。

例如 Nim 博弈属于公平组合游戏，而普通的棋类游戏，比如围棋，就不是公平组合游戏。因为围棋交战双方分别只能落黑子和白子，胜负判定也比较复杂，不满足条件2和条件3。

0x01 有向图游戏（博弈图）

给定一个有向无环图，图中有一个**唯一的起点**，在起点上放有一枚棋子。两名玩家交替地把这枚棋子沿**有向边进行移动**，每次可以移动一步，无法移动者判负。该游戏被称为有向图游戏。

任何一个公平组合游戏都可以转化为有向图游戏。具体方法是，把每个局面看成图中的一个节点，并且从每个局面向沿着合法行动能够到达的下一个局面连有向边。 转化为有向图游戏，也称绘制了它的博弈状态图（简称博弈图或游戏图）。

这样，对于组合游戏中的每一次对弈（每一局游戏），我们都可以将其抽象成游戏图中的一条从某一顶点到出度为 0 的点的路径。

组合游戏向图的转化，并不单单只是为了寻找一种对应关系，它可以帮助我们淡化游戏的实际背景，强化游戏的数学模型，更加突出游戏 的数学本质。

0x02 先手必胜和先手必败

- **先手必胜状态：** 先手行动以后，可以让剩余的状态变成**必败状态** 留给对手（下一步是对手（后手）的局面）。
- **先手必败状态：** 不管怎么操作，都达不到必败状态，换句话说，如果无论怎么行动都只能达到一个**先手必胜状态** 留给对手，那么对手（后手）**必胜**，先手必败。

简化一下就是：

- 先手必胜状态：可以走到某一个必败状态
- 先手必败状态：走不到任何一个必败状态

因为我们当前走到的状态是送给对手的状态hhh

通常先手是 `Alice` ，后手是`Bob` 。

定理：

定理 2.1： 没有后继状态的状态是必败状态。

定理 2.2： 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 2.3： 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，则通过这三个定理，我们可以在绘出博弈图的情况下用 $O(N+M)$ 的时间（其中 N 为状态种数，M 为边数）得出每个状态是必胜状态还是必败状态。

0x03 必胜点和必败点

必败点(P点) 前一个(previous player)选手将取胜的点称为必败点

必胜点(N点) 下一个(next player)选手将取胜的点称为必胜点

(1) 所有终结点是必败点（P点）

(2) 从任何必胜点（N点）操作，至少有一种方法可以进入必败点（P点）

(3)无论如何操作， 从必败点（P点）都只能进入必胜点（N点）

0x04 有向图的核

给定一张DAG图 $< V, E >$，如果 V 的一个点集 S 满足：

- S 是独立集（集合内的点互不相通）
- 集合 $V - S$ 中的点都可以通过**一步**到达集合 S 中的点（ $V - S$ 指 S 在 V 中的补集）

则称 S 是图 V 的一个**核**。

结论： **核内节点**对应 SG 组合游戏的**必败态**

因为 Alice 当前的棋子在 S 中，由于 S 是独立集，也就意味着 Alice 只能将棋子从 S 移动到 $V - S$

而 Bob又可以通过一步移动将棋子从 $V - S$ 移动到了 S，这样 Alice 就好像是被支配了一样，被迫把棋子移动到了没有出度的必败节点，Alice 必败，Bob必胜！

0x10 几个经典组合游戏

0x11 尼姆游戏 Nim Game

Problem

给定 N 堆物品，第 i 堆物品有 A_i 个。两名玩家轮流行动，每次可以任选一堆，取走任意多个物品，可把一堆取光，但不能不取。取走最后一件物品者获胜。两人都采取最优策略，问先手是否必胜。

我们把这种游戏称为 Nim 博弈。把游戏过程中面临的状态称为**局面**。整局游戏第一个行动的称为**先手**，第二个行动的称为**后手**。若在某一局面下无论采取何种行动，都会输掉游戏，则称**该局面必败**。

所谓采取最优策略是指，若在某一局面下存在某种行动，使得行动后对手面临必败局面，则优先采取该行动。同时，这样的局面被称为必胜。我们讨论的博弈问题一般都只考虑理想情况，即两人均无失误，都采取最优策略行动时游戏的结果。

Nim 博弈不存在平局，只有先手必胜和先手必败两种情况。

Solution

通过绘制博弈图，可以在 $\mathcal{O}(\Pi_{i=1}^{n}\ a_i)$ 的时间内求出该局是否先手必赢 but，这个时间复杂度太高了

结论：定义 Nim 和为 $a_1 \oplus a_2 \oplus \ldots \oplus a_n$。

当且仅当 Nim 和为 0 时，该状态为先手必败状态 否则该状态为先手必胜状态

> $\oplus$ 表示异或，相同为0不同为1

考虑证明：

我们只需要证明 $a_1 \oplus a_2 \oplus \ldots \oplus a_n \neq 0$ 是先手必胜状态，$a_1 \oplus a_2 \oplus \ldots \oplus a_n = 0$ 是先手必败状态即可。

我们可以从先手必败和先手必胜状态的定义出发，也就是说我们只需要证明下面的三个定理即可（博弈论结论证明都是朝着这个方向证明的）：

定理1： 没有后继状态的状态为必败状态

定理2： 对于 $a_1 \oplus a_2 \oplus \ldots \oplus a_n \neq 0$ 的局面，一定存在某种移动使得 $a_1 \oplus a_2 \oplus \ldots \oplus a_n = 0$ （必胜状态一定能到达一个必败状态）

定理3： 对于 $a_1 \oplus a_2 \oplus \ldots \oplus a_n = 0$ 的局面，一定不存在某种移动使得 $a_1 \oplus a_2 \oplus \ldots \oplus a_n = 0$ （必败状态一定不能到达任何必败状态）

定理1证明： 没有后继状态的状态（必败点）只有一个，即全 0 局面 此时 $a_1 \oplus a_2 \oplus \ldots \oplus a_n = 0$。定理1得证 □

定理2证明： 假设 $a_1 \oplus a_2 \oplus \ldots \oplus a_n = k \neq 0$ ，若进行一次操作，取走第 i 堆石子，使得 $a_i \Rightarrow a_i'$ 时，上式异或和为0，则显然 $a_i' = a_i \oplus k$ （$k \oplus k = 0$）。

然后我们证明一定存在这种操作即可。

设 k 在二进制下最高位的 1 在第 x 位，则一定有**奇数个** a_i 的二进制下在 x 位为 1。

因此 $a_i \oplus k = a_i' < a_i$ ，所以我们就可以从 a_i 这堆中拿走 $a_i - (a_i \oplus k) = a_i'$ 个石子（$a_i > a_i'$，所以有足够的石子去拿），则 a_i 就变成了 $a_i - (a_i - a_i \oplus k) = a_i \oplus k = a_i'$

则剩下的 Nim 和为 $a_1 \oplus a_2 \oplus \cdots \oplus a_i \oplus k \oplus a_{i+1} \oplus \cdots \oplus a_n = k \oplus k = 0$，定理2得证 □

定理3证明： 当前状态为 $a_1 \oplus a_2 \oplus \ldots \oplus a_n = 0$ ，我们仅需证明不管怎么拿，Nim 和都不会为 0 。

我们使用反证法：

假设我们从 a_i 拿走若干石子变为 a_i' ，使得Nim和变为 0 。

则说明存在：

$$a_1 \oplus a_2 \oplus \cdots \oplus a_i' \oplus a_{i+1} \oplus \cdots \oplus a_n = 0$$

但我们知道当前状态 Nim和为 0，即：

$$a_1 \oplus a_2 \oplus \cdots \oplus a_i \oplus a_{i+1} \oplus \cdots \oplus a_n = 0$$

两式异或起来得出 $a_i = a_i'$ ，很明显这是在原地转圈，不是合法的移动，不存在这种走法，定理3得证 □

定理 1 ~ 3 得证，故 $a_1 \oplus a_2 \oplus \ldots \oplus a_n \neq 0$ 是先手必胜状态，$a_1 \oplus a_2 \oplus \ldots \oplus a_n = 0$ 是先手必败状态， Nim 游戏解题结论得证。

Code

```cpp
#include <cstdio>
using namespace std;
int n, x, ans;
int main()
{
    scanf("%d", &n);
    for(int i = 1; i ≤ n; ++ i) {
        scanf("%d", &x);
        ans ^= x;
    }
    if(ans == 0) puts("No");
    else puts("Yes");
    return 0;
}
```

0x12 巴什博弈 Bash Game

Problem

有 1 堆石子，总个数是 n ，两名玩家轮流在石子堆中拿石子，每次至少取 1 个，至多取 m 个。取走最后一个石子的玩家为胜者。判定先手和后手谁胜。

Solution

结论

若 $(m+1) \mid n$ （整除）则先手必败 否则先手必胜

考虑证明：

情况一： 当 $n \leq m$ 时，显然先手获胜

情况二： 当 $n = m + 1$ 时，先手最多可取走 m 个，无论其取走多少个，剩下的后手总能一次取完。

情况三： 若 $(m+1) | n$ ，假设先手拿走了 x 个，那么后手一定可以拿走 $(m+1) - x$ 个，这样无论怎么拿剩下的石头个数都将是 $m+1$ 的倍数。那么最后一次取的时候石头个数必定还剩下 $m+1$ 个，即情况二。

否则的话，先手可以取走模 $m+1$ 余数个数的石头 此时转换为 $(m+1) | n$ 的局面 送给后手，这样后手变成了先手，也就就是后手必败。

Code

可以直接模拟

```cpp
int main() {
    scanf("%d%d", &n, &m);
    if (n % (m + 1))
        puts("first win");
    else
        puts("second win");

    return 0;
}
```

也可以转换成SG 游戏使用 SG 函数 （SG函数见下节）

```cpp
bool vis[N];
int sg[N];
void SG(int n, int m) {
    for (int i = 0; i ≤ n; ++i) {
        memset(vis, 0, sizeof(vis));
        for (int j = max(i - m, 0), j < i; ++j)
            vis[sg[j]] = 1;
        for (int j = 0; j ≤ n; ++j)
            if (!vis[j]) {
                sg[i] = j;
                break;
            }
    }
}
int main() {
    scanf("%d%d", &n, &m);
    SG(n, m);
    puts(sg[n] ? "first win" : "second win");
    return 0;
}
```

0x13 威佐夫博弈 Wythoff Game

Problem A 取石子游戏（POJ 1063）

有两堆石子，石子数可以不同。两人轮流取石子，每次可以在一堆中取，或者从两堆中取走相同个数的石子，数量不限，取走最后一个石头的人获胜。判定先手是否必胜。

Solution

一共只有两堆石子，我们可以把问题放到到二维坐标系上，设 x, y 分别对应两堆的石子的数量。

模拟发现显然 $(0, 0)$ 先手必败，我们将先手必败节点称为 **奇异节点**。

可以发现奇异节点上下左右四个结点，以及右上和右下的两个结点都不是奇异节点。

也就意味着如果 Alice 不在奇异节点上，那么 Alice 可以通过一步操作到达奇异节点，把奇异结点留给 Bob ，这样 Bob 必败。

我们可以发现 $(1, 2), (3, 5)$ 等等也都是奇异节点。

经过奇异节点的 3 条直线上的点，都能通过一步到达奇异节点。

发现这就建立起了一个有向图的核的模型。

考虑引入 **Beatty定理**

如果两个无理数 a, b 满足：

$$\frac{1}{a} + \frac{1}{b} = 1$$

那么对于两个集合 A, B：

$$A = \{\lfloor na \rfloor\}, B = \{\lfloor nb \rfloor\}, n \in Z$$

有下面两个结论：

$A \cap B = \varnothing, A \cup B = N^+$ （结论证明）

打表 发现 $B_i - A_i = i$

可得 $b = a + 1$ ，带入 $\frac{1}{a} + \frac{1}{b} = 1$ 解得 $a = \frac{\sqrt{5}+1}{2}, b = \frac{3-\sqrt{5}}{2}$

最终得出威佐夫博弈结论：假设两堆石子为 (a, b)（其中$a < b$ ）

那么先手必败，当且仅当 $(b - a) \times \frac{(\sqrt{5}+1)}{2} = a$

其中的 $\frac{(\sqrt{5}+1)}{2}$ 实际就是黄金分割数 1.618，细思极恐，细思极恐...

Code

```
int main() {
    scanf("%d%d", &n, &m);
    if (a > b)
        swap(a, b);
    int ans = (b - a) * ((1.0 + sqrt(5.0)) / 2.0);
    if (ans == a)
        puts("0");
    else
        puts("1");
    return 0;
}
```

Problem A Game of Taking Stones（2017 ACM ICPC dalian C）

给你两个石堆的石头数量 a, b，两个人轮流拿，两人轮流从任意一堆取至少一个或者从两堆取同样多的物品。问你先手获胜还是后手胜。

$a, b \leq 10^{100}$

威佐夫博弈模板题，但是数据开到了 10^{100}，普通的C++高精会炸，所以直接用Java就行了（而且Java比C++高精好写多了 ~ ）

我们直接用 Java 二分求出 $\sqrt{5}$，设 $b > a$ 计算 $\frac{(b-a)\times(\sqrt{5}+1)}{2}$ 即可。

Code

```java
import java.math.*;
import java.util.*;

public class Main {
    public static void main(String[] args) {
        BigDecimal one = BigDecimal.valueOf(1);
        BigDecimal two = BigDecimal.valueOf(2), five = BigDecimal.valueOf(5);
        BigDecimal t = one.add(sqrt(five, 500)).divide(two);
        Scanner sc = new Scanner(System.in);

        while (sc.hasNext()) {
            BigDecimal a, b, tmp = null;
            a = sc.nextBigDecimal();
            b = sc.nextBigDecimal();

            if (a.compareTo(b) > 0) {
                tmp = a;
                a = b;
                b = tmp;
            }

            if (b.subtract(a).multiply(t).setScale(0, BigDecimal.ROUND_DOWN).equals(a)) {
                System.out.println(0);
            } else
                System.out.println(1);

        }

        sc.close();
    }

    private static BigDecimal sqrt(BigDecimal x, int n) {
        BigDecimal l = BigDecimal.ZERO, r = x, mid;
        BigDecimal two = BigDecimal.valueOf(2);

        for (int i = 0; i ≤ n; i++) {
            mid = l.add(r).divide(two);

            if (mid.pow(2).compareTo(x) ≤ 0)
                l = mid;
            else
                r = mid;
        }

        return l;
    }
}
```

0x14 斐波那契博弈 Fibonacci Game

有一堆个数为 $n(n \geq 2)$的石子，游戏双方轮流取石子，规则如下：

- 先手不能在第一次把所有的石子取完，至少取 1 颗；
- 之后每次可以取的石子数至少为 1，至多为对手刚取的石子数的 2 倍。

约定取走最后一个石子的人为赢家，求必败态。

结论：

先手必败，当且仅当石子数为斐波那契数

先证明必要性，斐波那契数一定先手必败，可以用数学归纳法，大致思路就是一定能拆成两堆斐波那契数，不论先手怎样取，后手总能取到最后一颗

然后证明充分性，由定理：**任何正整数可以表示为若干个不连续的Fibonacci数之和**，那么就回到了斐波那契数列里

具体证明见：斐波那契博弈（Fibonacci Nim）

```cpp
#include <cstdio>
#include <map>
#include <iostream>
#include <algorithm>
#include <cstring>

using namespace std;
const int N = 50007;
int f[N], x;
map<int,bool>mp;
int main()
{
    fib[1] = 1;
    fib[2] = 1;
    for(int i = 3;i ≤ 50; ++ i) f[i] = f[i-1] + f[i-2], mp[f[i]] = 1;
    while(scanf("%d", &x) && x ≠ 0)
        puts(mp[x] == 1 ? "Second win" : "First win");
    return 0;
}
```

0x20 SG函数

0x21 前置知识： Mex 运算

设 S 表示一个非负整数集合。定义 $mex(S)$ 为求出**不属于**集合S的**最小非负整数**的运算，即：

$mex(S) = min\{x\}$，x 属于自然数，且 x 不属于 S 。

0x22 SG函数

SG函数是对游戏图中每一个节点的评估函数。

规定游戏终点的 SG 函数值定为 0，即 $SG(\text{终点}) = 0$。

在有向图游戏中（任何一个博弈都可以转换为一个有向图游戏），对于每个节点 x （局面），设从 x 出发共有 k 条有向边（合法的操作），分别到达节点 $y_1, y_2, \ldots, y_k$（下一个局面），定义 $SG(x)$ 为 x 的**后继节点（注意只是一层的后继结点）** $y_1, y_2, \ldots, y_k$ 的 SG 函数值构成的集合再执行 $mex(S)$ 运算的结果，即：

$SG(x) = mex(\{SG(y1), SG(y2), \ldots, SG(yk)\})$

如 **图20.1** 所示：

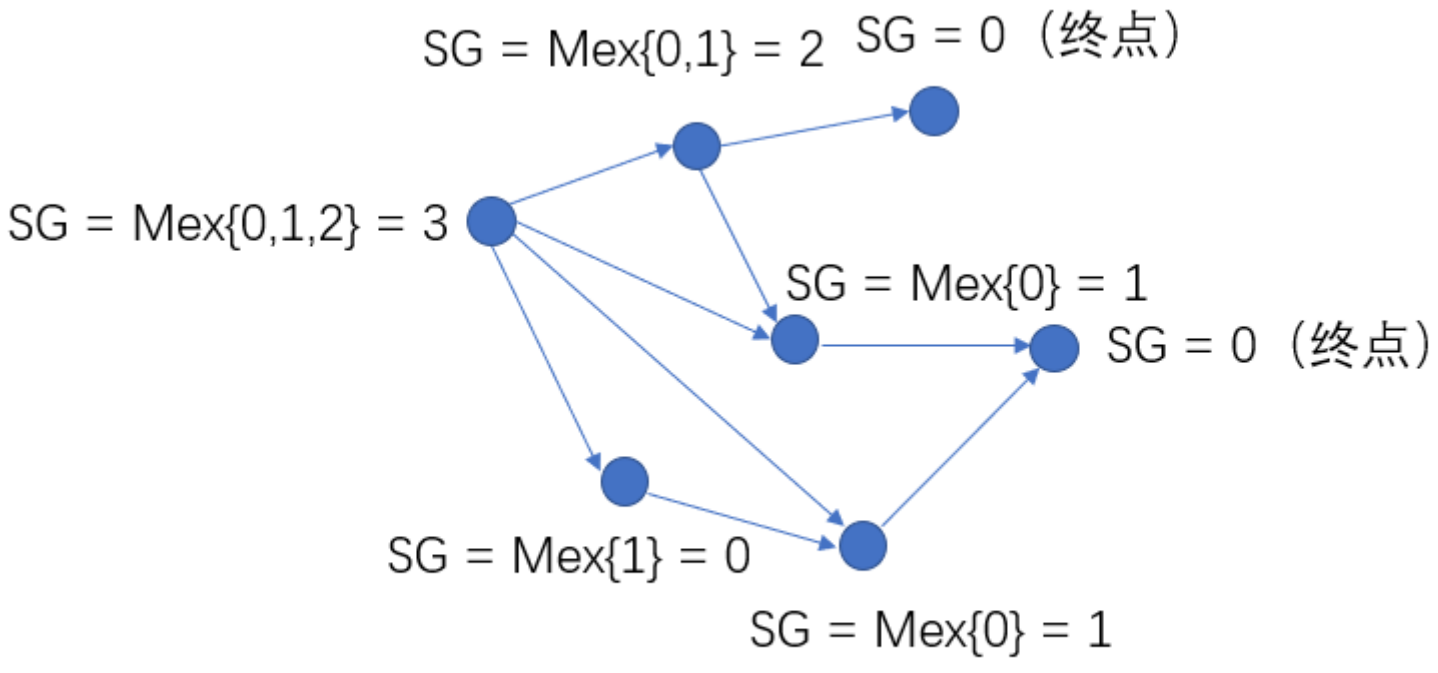

图20.1

特别地，整个有向图游戏 G 的 SG 函数值被定义为有向图游戏起点 s 的 SG 函数值，即 $SG(G) = SG(s)$ 。

我们发现若 $SG(x) = 0$ 则为必败状态，若 $SG(x) \neq 0$，则为必胜状态。（若非零说明这个点直接指向了 0，也就意味着可以到达必败状态，是必胜状态）

0x23 SG 定理

两个SG函数的性质：

（1）对于任意的局面，如果它的 SG 值为 0 ，那么它的任何一个后继局面的SG值不为 0 。

（2）对于任意的局面，如果它的 SG 值不为 0 ，那么它一定有一个后继局面的 SG 值为 0 。

SG（Sprague − Grundy）定理 ：所有一般胜利下的公平组合游戏都能转化成尼姆数表达的尼姆堆博弈，一个博弈的 **尼姆值** 定义为这个博弈的等价尼姆数，即：对于当前游戏 X，它可以拆分成若干个子游戏 $x_1, x_2, \ldots, x_n$ 那么 $SG(X) = SG(x_1) \oplus SG(x_2) \oplus \ldots \oplus SG(x_n)$。

对于由 n 个有向图游戏组成的组合游戏 设它们的起点分别为 $s_1, s_2, \ldots, s_n$（好多个起点，有好多个博弈图，也就是有好多个 图20.1 ），则当且仅当 $SG(s_1) \oplus SG(s_2) \oplus \ldots \oplus SG(s_n) \neq 0$ 时，这个游戏**为先手必胜** 。

也就意味着，我们将原本需要考虑博弈图的所有点，复杂度较高，但是我们通过SG函数，变成了只需要考虑起点即可。

证明方法类似尼姆游戏，略。

0x24 转换为 Nim 游戏

事实上，每一个简单SG-组合游戏都可以完全等效成一堆数目为 K 的石子（Nim 游戏），其中 K 为该简单游戏的 SG 函数值。这样的等效是充要的。

定义游戏的和： 考虑任意多个同时进行的SG-组合游戏，这些SG-组合游戏的和是这样一个SG-组合游戏，在它进行的过程中，游戏者可以任意挑选其中的一个 **单一游戏** 进行决策，最终，没有办法进行决策的人输。

定理 23.1： 在我们每次只能进行一步操作的情况下，对于任何的游戏的和，我们若将其中的任一单一SG-组合游戏换成数目为它的SG值的一堆石子， 该单一SG-组合游戏的规则变成取石子游戏的规则（可以任意取，甚至 取完），则游戏的和的胜负情况不变。

这个定理告诉我们，在考虑游戏的和时，每一个单一游戏的具体细节是可以被忽略的，我们所关心的只是**SG函数值**。所以我们可以将组成它的所有子游戏全部换成相应数目的一堆石子。这样，所有的游戏的和都等价成一个 Nim 游戏。

0x24 有向图游戏的和

设 $G_1, G_2, \ldots, G_m$ 是 m 个有向图游戏。定义有向图游戏 G ，它的行动规则是任选某个有向图游戏 G_i ，并在 G_i 上行动一步。G 被称为有向图游戏$G_1, G_2, \ldots, G_m$的和。

有向图游戏的和的 SG 函数值等于它包含的各个子游戏 SG 函数值的异或和，即：

$$SG(G) = SG(G_1) \oplus SG(G_2) \oplus \ldots \oplus SG(G_m)$$

定理22.1： 有向图游戏的某个局面必胜，当且仅当该局面对应节点的SG函数值大于0。

定理22.2： 有向图游戏的某个局面必败，当且仅当该局面对应节点的SG函数值等于0。

我们只需要判断一下 $SG(G)$ 即可。

Problem A 集合- Nim游戏 （ AcWing 893）

给定 n 堆石子以及一个由 k 个不同正整数构成的数字集合 S 。

现在有两位玩家轮流操作，每次操作可以从任意一堆石子中拿取石子，每次拿取的石子数量必须包含于集合 S ，最后无法进行操作的人视为失败。

问如果两人都采用最优策略，先手是否必胜。

每一堆输入的**石子数**就是每一堆的**起点**，（这个石子数可不是 sg 函数值）答案就是所有**起点**的 SG 函数值异或和

假设某一堆有 10 个石子，$S = \{2, 5\}$ 那么博弈图就是从 10 开始连边，如 **图20.1** 所示。

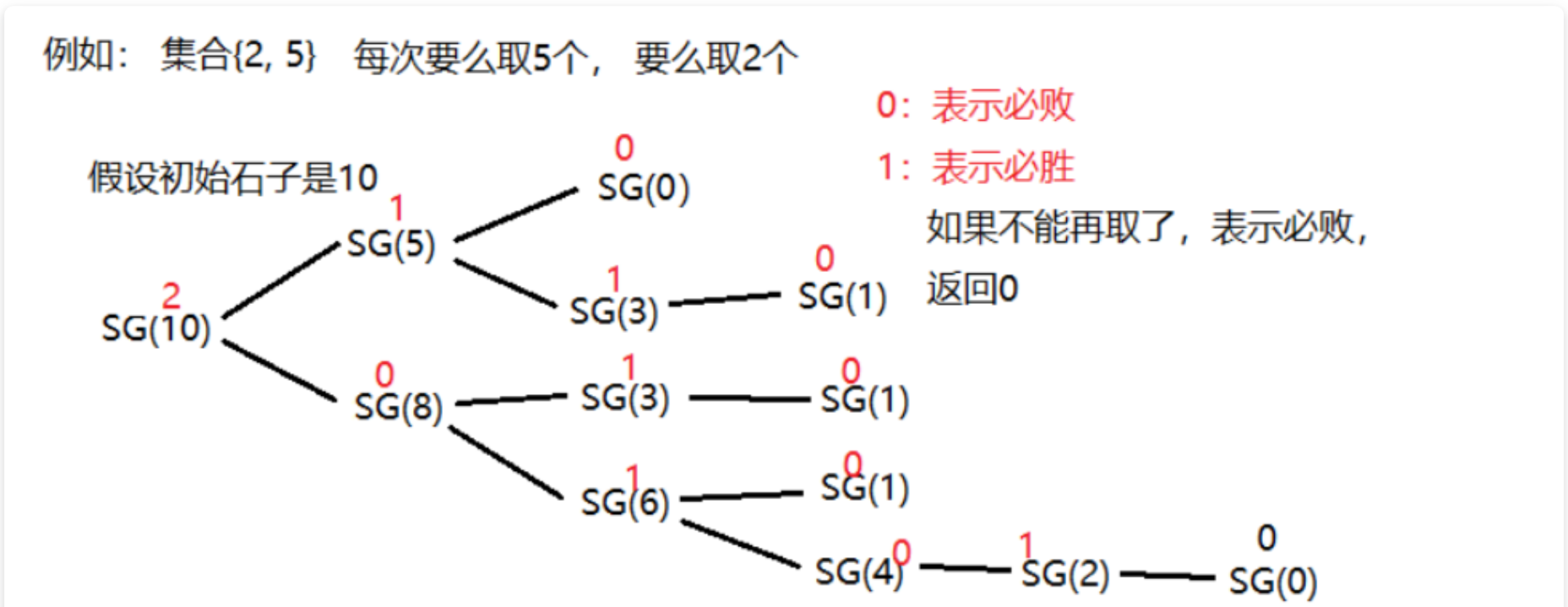

图20.2

本题的连边方式就是从集合 S 中选择一个数往下走（取走这么多数，往下搜 $x-sum$）

我们直接记忆化搜索sg函数即可。

Code

```cpp
#include <cstdio>
#include <iostream>
#include <algorithm>
#include <set>
#include <unordered_set>
#include <cstring>

using namespace std;
const int N = 507, M = 50007;
typedef long long ll;
typedef int itn;

itn n, m;
int a[N];
itn s[N], f[M];

int sg(int x)//记忆化搜索
{
    if(f[x] != -1) return f[x];

    unordered_set<int>S;
    for(int i = 1; i <= m; ++ i) {
        itn sum = s[i];
        if(x >= sum) S.insert(sg(x - sum));
    }
    for(int i = 0; ; ++ i) {
        if(!S.count(i))
        return f[x] = i;
    }
}

int main()
{
    scanf("%d", &m);
    for(int i = 1; i <= m; ++ i) {// m 个起点
        scanf("%d", &s[i]);
    }
    int res = 0;
    scanf("%d", &n);
    memset(f, -1, sizeof f);
    for(int i = 1; i <= n; ++ i) {
        int x;
        scanf("%d", &x);
        res ^= sg(x);
    }

    if(res == 0) puts("No");
    else puts("Yes");
    return 0;
```

```
50 }
```

0x30 SG游戏及其拓展变形

0x 31 Anti-SG 游戏（走完最后一步者输）

我们先从最基本的 Anti-Nim 游戏开始讲起。

0x31.1 Anti-Nim游戏

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**)，拿走最后一个石子的人**失败**。问谁会胜利

看上去好像颠覆了SG游戏的规则，连胜利的条件都反了这怎么玩?

先给出结论

先手必胜当且仅当:

(1) $\forall$ **所有堆的石子数都为** 1 **且游戏的SG值为** 0 **。**

(2) $\exists$ **有些堆的石子数大于** 1 **且游戏的SG值不为** 0 **。**

考虑证明:

游戏大概可以被分为 3 种情况

- **每堆只有一个石子**

每一堆石子的 SG值 显然是这堆石子的个数，每一堆石子就是一个起点，故:

当总异或值（**游戏和**的SG值）为 0 时（有偶数堆），先手必胜
当总异或值（**游戏和**的SG值）不为 0 时（有奇数堆），先手必败

- **只有一堆石子数大于1，先手必胜**

我们发现先手可以对数量大于 1 的那堆石子下手，如果除去这堆异或值不为 0 ，那先手就可以把这堆拿完，使得留给对手的总异或值不为 0，后手必败（对于后手的局面先手必败，此时后手就是先手）。

- **存在至少两堆石子数大于** 1

当异或和为 0 时，先手必败
当异或和不为 0 时，先手必胜

当异或和为 0 时，由于至少有两堆石子的数目大于 1 ，则在先手决策完之后，必定至少有一堆的石子数大于 1，且SG值不为 0 ，这时对于此时的先手（比赛开始前的后手）到达了只有一堆石子数大于 1 的情况，先手必胜。 也就意味着此时，无论先手如何决策，都只会将游戏**带入**到先手必胜局，所以先手必败。

当异或和不为 0 时，由于还有至少两堆石子的数目大于 1，则先手通过一次操作将 SG值 变为 0 即可，局面就变成了上面那种先手必败的局面送给后手，也就意味着先手必胜。

综上所述，定理得证 □

0x31.2 Anti-SG游戏

上一小节的关于 Anti - Nim 游戏的结论推导只对 Anti - Nim 这一简单游戏成立。因为我们在证明 SG 函数性质时，用到了这样一个性质：**SG值为0的局面不一定为终止局面** 。

也就是说 Anti - Nim 游戏的结论并不适合所有的 SG 游戏。

因此对于我们再来研究一下普通的 Anti - SG 游戏。

定义Anti - SG游戏：决策集合为空的游戏者获胜，也可以理解将所有集合变为空的游戏者即为失败。

其余规则与普通的 SG 游戏相同。

为了解决这一问题，**定义 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：

- **游戏的 SG 函数值不为 0 且游戏中某个单一游戏的 SG 函数值大于 1。**
- **游戏的SG函数值为 0 且游戏中没有任意一个单一游戏的SG函数值大于 1 。**

考虑证明：

> 以下证明来自 **《组合游戏略述——浅谈SG游戏的若干拓展及变形》贾志豪IOI2009国家集训队论文**

我们只需要证明：

（1） 所有的终止局面为先手必胜局。（这一点显然，证明中略去）

（2） 游戏中的任何一个先手必败局一定只能够转移到先手必胜局；

（3） 游戏中的任何一个先手必胜局一定能够转移到至少一个先手必败局。

情况一：局面的SG函数为0且游戏中某个单一游戏的SG函数大于 1。

∵当前局面的SG函数值为0

又∵SG函数性质（1）

∴它所能转移到的任何一个局面的SG函数值不为0 ①

∵当前局面的SG函数值为0且游戏中某个单一游戏的SG函数大于 1。

∴当前局面中必定至少有2个单一游戏的SG函数大于1。

又∵每次至多只能更改一个单一游戏的SG值

∴它所能转移到的任何一个局面都至少有一个单一游戏的SG值大于 1。 ②

由①②得，情况一所能转移到的任何一个局面都为先手必胜局。

情况二：局面的SG函数不为0且游戏中没有单一游戏的SG函数大 于1。

显然，当前局面一定有奇数个游戏的SG函数值为1，其余游戏的SG 函数值为0。

（1） 将某个单一游戏的SG值更改为大于1的数。

∵转移前没有单一游戏的SG值大与1，转移将某个单一游戏的SG 值更改为大于1的数。

∴转移后的局面一定有且只有一个单一游戏的SG值大于1。 ③

∴后继局面的SG值一定不为0。 ④

由③④得，后继局面一定为先手必胜局。

（2） 将某个单一游戏的SG值更改为0或1。

∵转移是将某个SG值为0的单一游戏改成SG值为1的单一游戏， 或将某个SG值为1的单一游戏改成SG值为0的单一游戏。

∴转移后的局面一定有偶数个SG值为1的单一局面且不含有SG值 大于1的局面。

∴后继局面一定为先手必胜局。

情况三：局面的SG函数不为0且游戏中某个单一游戏的SG函数大 于1。

（1）局面中只有1个单一游戏的SG值大于1。 我们选择更改SG值最大的单一游戏，我们可以选择将其更改成0或 1来保证转移后的局面有且只有奇数个SG值为1的单一游戏。

则通过这种方式转以后的局面为先手必败局。⑤

∵局面中有至少两个单一游戏的SG值大于1

又∵每次最多只能更改一个单一游戏的SG值

∴后继局面中至少有一个游戏的SG值大于1 ⑥

由⑤⑥得，后继局面为先手必败局。

情况四：局面的SG函数为0且游戏中没有单一游戏的SG函数大于 1。

当局面中所有单一游戏的SG值为0时，游戏结束，先手必胜。

否则，局面有且仅有偶数个SG值为1的单一游戏，其余游戏的SG 值为0。

我们只需将其中的某一个SG值为1的单一游戏的SG值变为0，游戏 中即可出现奇数个SG值为1的单一游戏，到达先手必败局。

综上，证明完毕！

实际上，聪明的读者可能会发现，我们在SJ定理中给出的附加条件 “规定当局面中所有的单一游戏的SG值为0时，游戏结束”过于严格， 完全可以替换成“当局面中所有的单一游戏的SG值为0时，存在一个单 一游戏它的SG函数能通过一次操作变为1”。

笔者为什么要将限制条件设制成这样？

因为笔者发现这样可以出题，我们可以将题目模型设成这样：游戏 中存在一个按钮，游戏双方都可以触动按钮，当其中一个人触动按钮时，
触动按钮的人每次必须移动对方上次移动的棋子。如果触动按钮的人能 保证他能够使得对方无路可走，那么他同样获胜！

Problem B 小约翰的游戏（luogu P4279 [SHOI2008]）

小约翰经常和他的哥哥玩一个非常有趣的游戏：桌子上有n堆石子，小约翰和他的哥哥轮流取石子，每个人取的时候，可以随意选择一堆石子，在这堆石子中取走任意多的石子，但不能一粒石子也不取，我们规定取到最后一粒石子的人算输。小约翰相当固执，他坚持认为先取的人有很大的优势，所以他总是先取石子，而他的哥哥就聪明多了，他从来没有在游戏中犯过错误。小约翰一怒之前请你来做他的参谋。自然，你应该先写一个程序，预测一下谁将获得游戏的胜利。

Solution

Anti - Nim 模板题，直接用结论就好。

Code

```cpp
#include <cstdio>
#include <iostream>
#include <algorithm>
#include <cstring>
#include <cmath>

using namespace std;
typedef long long ll;
typedef int itn;
const int N = 2e5 + 7;

int n, m, t, k;
int a[N];


```

```cpp
int main()
{
    scanf("%d", &t);
    while(t -- ) {
        scanf("%d", &n);
        int sg = 0;
        bool flag = 0;
        for(int i = 1; i ≤ n; ++ i) {
            scanf("%d", &a[i]);
            sg ^= a[i];
            if(a[i] > 1)
                flag = 1;
        }
        if((flag == 0 && sg == 0) || (flag == 1 && sg ≠ 0)) {
            puts("John");
        }
        else puts("Brother");
    }
    return 0;
}
```

0x33 Multi-SG游戏（可以将一堆石子分成多堆）

0x33.1 Multi - Nim 游戏

我们还是先从最简单的 Multi-Nim 游戏出发

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）或者可以把一堆数量不少于 2 石子堆分为两堆不为空的石子堆，没法拿的人失败。问谁会胜利。

Multi - Nim 游戏一共有两种操作，其中操作一很明显就是普通的 Nim 游戏。操作二实际上就是把一个单一游戏分为两个单一游戏，根据 SG 定理，我们知道两个游戏的异或和就是这个单一游戏拆分前的SG函数值，作为一个后继状态。

Example 33.1.1： $SG(3)$ 的后继状态有 $\{(0),(1),(2),(1,2)\}$ 也就是这堆有 3 个石子的石子堆，可以拿走或者分开等四种情况，他们的SG值分别为 $\{0,mex\{0\}=1,mex\{0,1\}=2,mex\{0,1,2\}=3\}$ ，因此 $SG(3)=mex\{0,1,2,3\}=4$

Multi - Nim 游戏的性质:

$$SG\left(x\right)=\begin{cases}x-1\left(x\mod 4=0\right)\\x\left(x\mod 4=1\text{ or }2\right)\\x+1\left(x\mod 4=3\right)\end{cases}\tag{1}$$

0x33.2 Multi - SG游戏

因此我们定义 Multi - SG 游戏:

- Multi - SG 游戏规定，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 。
- Multi-SG其他规则与SG游戏相同。

可以理解为每次操作能将一个当前的单一游戏分为多个单一游戏，也就是将当前这个堆石子分为多堆石子的特殊游戏。

对于一个状态来说，不同的划分方法会产生多个不同的后继，而在一个后继中可能含有多个独立的游戏

一个后继状态的SG值即为后继状态中所有独立游戏的异或和

该状态的 SG 函数值即为后继状态的 SG 函数值中未出现过的最小值

注意区分，可以再看一遍 **Example 33.1.1** 加深理解。

- **竞赛例题选讲**

Problem A Nim or not Nim?（HDU - 3032）

题意：给定n堆石子，两人轮流操作，每次选一堆石子，取任意石子或则将石子分成两个更小的堆(非0)，取得最后一个石子的为胜。

Solution

Mul - Nim 游戏的模板题。
我们打表找规律可以得到上面给出的性质：

$$SG\left(x\right)=\begin{cases}x-1\left(x\mod 4=0\right)\\x\left(x\mod 4=1\text{ or }2\right)\\x+1\left(x\mod 4=3\right)\end{cases}\tag{2}$$

Da Biao Code

```cpp
#include <iostream>
#include <cstdio>
#include <cmath>
#include <cstring>
#include <algorithm>

using namespace std;

typedef long long ll;
const int N = 50007, M = 507;
const int INF = 0x3f3f3f3f;


int sg[N];
bool vis[M + 7];
int main(){
    sg[0] = 0, sg[1] = 1;
    for (int i = 2; i < M; ++ i){
        memset(vis, 0, sizeof(vis));
        //操作一，至少取一个
        for (int j = 1; j ≤ i; ++ j)
            vis[sg[i - j]] = 1;
        //操作二，分成两堆，不为空
        for (int j = 1; j < i; ++ j)
                    vis[sg[j] ^ sg[i - j]] = 1;
        int j = 0;
        while (vis[j]) j ++ ;
        sg[i] = j;
    }

    for (int i = 1; i ≤ M; ++ i)
        printf("sg[%d] : %d\n", i, sg[i]);
    return 0;
}
```

AC Code

```cpp
#include <cstdio>
```

```cpp
#include <iostream>
#include <algorithm>
#include <cstring>
#include <cmath>

using namespace std;
typedef long long ll;
typedef int itn;
const int N = 2e6 + 7;

int n, a[N], sg[N];

int main()
{
    int t;
    scanf("%d", &t);
    while(t -- ) {
        int ans = 0;
        scanf("%d", &n);
        for(int i = 1; i ≤ n; ++ i)
            scanf("%d", &a[i]);
        for(int i = 1; i ≤ n; ++ i) {
            if(a[i] % 4 == 0) sg[i] = a[i] - 1;
            else if(a[i] % 4 == 3) sg[i] = a[i] + 1;
            else sg[i] = a[i];
            ans ^= sg[i];
        }
        if(ans == 0) puts("Bob");
        else puts("Alice");
    }
    return 0;
}
```

Problem C A Simple Nim（HDU-5795）

游戏中有 n 堆石子，每次行动可以选择：

- 取走某堆的任意数量的石子（不可不取）。
- 将石子拆分成三堆（三堆都不可为空）。

最后取走为胜，问先手胜还是后手。

Solution

打表找规律：

$$SG\left(x\right)=\begin{cases}x-1\left(x\mod 8=0\right)\\x\left(otherwise\right)\\x+1\left(x\mod 8=7\right)\end{cases}\tag{3}$$

Da Biao Code

将上题代码略作修改即可。

```cpp
//操作二，分成三堆不为空
for (int j = 1; j ≤ i; ++ j)
    for (int k = j; k ≤ i; ++ k)
        if ((j + k) < i)
            vis[sg[k] ^ sg[j] ^ sg[i - j - k]] = 1;
```

AC 代码基本同上题，略...

0x34 Every-SG游戏（每一个可以移动的棋子都要移动）

定义 Every - SG 游戏:

给定一张无向图，上面有一些棋子，两个顶尖聪明的人在做游戏，每人每次必须将所有可以移动的棋子都进行移动，最后不能移动的人输。

Solution

题目中的要求实际是“不论前面输与否，只要最后一个棋子胜利，那么就算胜利”

这样的话，能赢得游戏必须赢

因为两个人都顶尖聪明，因此当一个人知道某一个游戏一定会输的话，它一定会尽力缩短游戏的时间，当它知道某一个游戏一定会赢的话，一定会尽力延长游戏的时间

定义Every-SG游戏

对于还没有结束的单一游戏，游戏者必须对该游戏进行一步决策；
其他规则与普通SG游戏相同
Every-SG游戏与普通SG游戏最大的不同就是它多了一维：时间

对于SG值为0的点，我们需要知道最少需要多少步才能走到结束，
对于SG值不为0的点，我们需要知道最多需要多少步结束

这样我们用step变量来记录这个步数

0x35翻硬币游戏

0x36 无向图删边游戏

0x40 经典组合游戏拓展

0x41 巴什博奕的扩展——k倍动态减法游戏

0x42 尼姆博弈的三种扩展

0x50 寻找必败态解题

0x60 不平等博弈问题

0x70 更多例题

Problem A Euclid's Game（POJ 1063 ）

给定两个正整数 a, b，每次操作，可以将大的数减掉小的数的整数倍。当一个数变为 0 的时候结束。先将其中一个数减为 0 的获胜。Stan先手，问谁能赢。

Solution

把问题转化成我们熟悉的模型，相当于有一排石子堆，必须把前面的石子堆取完了才能取后面的，取最后一个石子的人赢，问谁能赢

- 博弈论知识汇总
- 博弈论合集（博弈）
- 博弈论全家桶
- 博弈论题目总结（一）——简单组合游戏
- 博弈论题目总结（二）——SG组合游戏及变形
- 博弈论进阶之Anti-SG游戏与SJ定理
- 博弈论总结
- 《浅谈如何解决不平等博弈问题》 方展鹏
- 《从“k倍动态减法游戏”出发探究一类组合游戏问题》曹钦翔
- 《组合游戏略述——浅谈SG游戏的若干拓展及变形》贾志豪
- acm中的一些博弈论知识
- ACM-ICPC中博弈论的一些小小总结
- 寻找必败态——一类博弈问题的快速解法
- 运筹学基础教程（第二版）第9章 对策（博弈）论.ppt

- https://blog.csdn.net/ACM_cxlove/article/details/7854526