博弈论入门

山东省实验中学
宁华

问题引入

- 取石子游戏

- 有一堆共 100 个石子，甲、乙两人轮流取，每人每次最多取 2 个石子，不能不取。若轮到某人时无石子可取，则判其输。若甲先取，他有无必胜把握？若有，他该怎么取？若没有，请说明理由。

取石子游戏

- 取石子游戏是一个古老的博弈游戏，**发源于中国**，它是**组合数学领域**的一个经典问题。

- 它有许多不同的玩法，基本上是两个玩家，玩的形式是轮流抓石子，胜利的标准是抓走了最后的石子。

- 玩家设定： 先取石子的是玩家A，后取石子的是玩家B。

博弈问题

- 所讨论的博弈问题满足以下条件：

- 1、玩家只有两个人，轮流做出操作
- 2、游戏的状态集有限，保证游戏在有限步后结束，这样必然会产生不能操作者，其输
- 3、对任何一种局面，合法的操作集合只决定于局面本身，而与轮到哪位选手、以前的任何操作等因素无关

- 一般称满足以上条件的游戏为 ICG（Impartial Combinatorial Games, 公平的组合游戏），比如刚才的取石子游戏。作为一个对比，我们平时玩的象棋就不属于ICG，它不满足第三条，因为红方只能移动红子，黑方只能移动黑子，合法的操作集合取决于轮到哪名选手操作。

策梅洛定理（Zermelo's theorem）

- 若一个游戏满足如下条件：

- 双人、回合制；
- 信息完全公开（perfect information）；
- 无随机因素（deterministic）；
- 必然在有限步内结束；
- 没有平局；

- 则游戏中的任何一个状态，要么先手有必胜策略，要么后手有必胜策略

ICG的两个状态

- ICG具有两个状态，我们称为必胜态和必败态：

- 1. 没有平局，每个游戏局面要么是必胜态，要么是必败态；
- 2. 一个状态是必败态，当且仅当它的所有后继状态都是必胜态；
- 3. 一个状态是必胜态，当且仅当它的后继状态存在至少一个必败态。

- 必胜策略的核心本质是：理清必胜态和必败态，并构造必胜态与必败态之间的状态转移。

平衡状态（奇异局势）的概念

- 平衡状态，就是刚才提到的必败态，也称作奇异局势。当面对这个局势时，先手会失败。

- 任意非平衡态经过一次操作可以变为平衡态
- 任意平衡态经过一次操作可以变为非平衡态

- 每个玩家都会努力使自己抓完石子之后的局势为平衡态，将这个奇异局势留给对方。因此，先手（玩家A）能够在初始为非平衡态的游戏中取胜，后手（玩家B）能够在初始为平衡态的游戏中取胜。

取石子游戏的三种经典玩法

- 一、巴什博奕(Bash Game)，有 1 堆共 n 个石子，两个人轮流取，规定每次最多取 m 个，不能不取。最后取光者得胜。

- 二、尼姆博奕(Nimm Game)，有 k 堆各若干个石子，两个人轮流取，规定每次可以任选某一堆取其中任意多个石子，不能不取。最后取光者得胜。

- 三、威佐夫博奕(Wythoff Game)，有 2 堆各若干个石子，两个人轮流取，规定每次可以任选某一堆取其中任意多个石子，或同时从两堆中取同样多的石子，不能不取。最后取光者得胜。（POJ1067）

一、巴什博奕(Bash Game)

- 有 1 堆共 n 个石子，两个人轮流取，规定每次最多取 m 个，不能不取。最后取光者得胜。

分析

- 1、当 n <= m 的时候，显然先手获胜，因为一次就能取完。
- 2、当 n = m+1 的时候，由于先手最多取走m个，无论其取走多少个，剩下的后手均可以一次取完，显然后手胜。
- 3、根据以上分析，我们可以将 n 写成 n = (m+1) * r + s 的形式。对于先手玩家，可以取走s个，给对方造成剩下 (m+1) 整数倍的情形。此时无论对手取走多少个，假设对手取走 k 个， 我们一定可以做到取走 (m+1-k) 个，此时剩下 (m+1) * (r-1) 个，那么留给对方又是 (m+1) 的整数倍，如此就可以保证 先手 取胜。

结论

- 1、当 n <=m 时，先手必胜。
- 2、当 n % (m+1) = 0时，后手必胜。
- 3、当 n % (m+1) != 0时，先手必胜。

- 其中上述的情况1和3可以合并，故：

- 1、当 n % (m+1) = 0时，后手必胜。
- 2、当 n % (m+1) != 0时，先手必胜。

用专业术语再分析一遍

- 最后一个奇异局势是n=(0)。
- 一种奇异局势是，n=(m+1)，无论当前玩家取走多少个，对方都能够一次取走剩余所有的物品取胜。

- 奇异局势的判定：
- 一般的奇异局势是n=(m+1)*i，其中i为自然数，即n%(m+1)=0，面对这种情况无论当前玩家怎么取，对方总可以将其恢复为n%(m+1)=0，一直到n=(m+1)局势。

- 玩家的策略：
- 就是把当前面对的非奇异局势变为奇异局势留给对方。如果当前的石子个数为(m+1)*i+s，那么就将s个石子取走，使其达到奇异局势。

再引入两个概念

- 什么是 P、N 点?
- 就是之前介绍的必败态、必胜态。
- 1、P-position，称为 P 点，面对 P 时先手必败。P 表示 Previous player 必胜，即当前选手的必败点。
- 2、N-position，称为 N 点，面对 N 时先手必胜。N 表示 Next player 即当前选手的必胜点。

PN点的属性

- 1、所有终结点均为必败点(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，必败点(P点)都只能进入必胜点(N点)。

- 我们研究必胜点和必败点的目的是为了对问题进行简化，有助于我们的分析。**通常我们分析必胜点和必败点都是以终结点进行逆序分析。**

画PN图步骤

- 1、将所有终结位置标记为必败点(P点);
- 2、将所有一步能进入必败点(P点)的位置标记为必胜点(N点);
- 3、如果从某个点开始的所有一步操作都只能进入必胜点(N点)，则将该位置标记为必败点(P点);
- 4、如果在步骤3中未能找到新的必败点(P点)，算法终止，否则返回步骤2.

#1681. 取石子游戏

分析

- 当 n = 0 时，显然为必败点，因为此时你已经无法进行操作了
- 当 n = 1 时，因为你一次就可以拿完所有石子，故此时为必胜点
- 当 n = 2 时，也是一次就可以拿完，故此时为必胜点
- 当 n = 3 时，要么剩一个要么剩两个，无论怎么取对方都将面对必胜点，故这一点为必败点。
- 以此类推，最后你就可以得到：

n	0	1	2	3	4	5	6	……
position	P	N	N	P	N	N	P	……

- 然后分析找到其规律：？

#1682. 石子游戏

- 本题是巴什博奕的变形。
- 我们采用画P/N图的方法来解决。

P/N 图

- 一.所有终结点都是必败点P
- 二.只要有一步能走到必败点P的就是N点
- 三.通过一步操作只能到N点的就是P点

- 下面就是 P/N 图
- 最左下角一定是终结点所以是P

P	N	P	N	P	N	P
N	N	N	N	N	N	N
P	N	P	N	P	N	P
N	N	N	N	N	N	N
P	N	P	N	P	N	P
N	N	N	N	N	N	N
P	N	P	N	P	N	P

- 发现规律：
- ？

P	N	P	N	P	N	P
N	N	N	N	N	N	N
P	N	P	N	P	N	P
N	N	N	N	N	N	N
P	N	P	N	P	N	P
N	N	N	N	N	N	N
P	N	P	N	P	N	P

SG函数：Sprague-Grundy函数

- SG函数为计算博弈状态的函数，当SG[X] = 0时，说明先手必败。
- 为了求解SG函数，首先定义mex(minimal excludant)运算，这是施加于一个集合的运算，表示最小的不属于这个集合的非负整数。例如mex{0,1,2,4}=3、mex{2,3,5}=0、 mex{}=0。
- 对于任意状态 x， 定义 SG(x) = mex(S),其中 S 是 x 后继状态的SG函数值的集合。如 x 有三个后继状态分别为 a,b,c，那么SG(x) = mex{SG(a),SG(b),SG(c)}。
- 这样 集合S 的终态必然是空集，所以SG函数的终态为 SG(x) = 0, 当且仅当 x 为必败点P时。

实例：取石子游戏

- 游戏规则：
- 有 1 堆共 n 个石子，两个人轮流取，每人每次只能取 1 或 3 或 4 个石子，无石子可取者为输。

- 输入不同的 n 值，问先手是否有必胜的策略。

SG函数分析

- f = {1,3,4} (表示每次取有3种方案：取1个，或取3个，或取4个)；
- SG[0] = 0； (显然没有石子可取时必败)
- 当石子 x = 1 时, 可以取走 1 个石子, SG[1] = mex(SG[1-1]) = mex(SG[0]) = mex(0) = 1;
- 当石子 x = 2 时, 可以取走 1 个石子, SG[2] = mex(SG[2-1]) = mex(SG[1]) = mex(1) = 0;
- 当石子 x = 3 时, 可以取走 1 或 3 个石子, SG[3] = mex(SG[3-1],SG[3-3]) = mex(0,0) = 1;
- 当石子 x = 4 时, 可以取走 1 或 3 或 4 个石子, SG[4] = mex(SG[4-1],SG[4-3],SG[4-4]) = mex(SG[3], SG[1], SG[0]) = mex(1,1,0) = 2;
- 以此类推…

- 我们可以打出SG函数的表，根据表来判断是先手必胜还是先手必败。

x	0	1	2	3	4	5	6	7	8	……
sg[x]	0	1	0	1	2	3	2	0	1	……

- 对任意的 x ， 若 sg[x]==0 则先手必败，否则先手必胜。

计算SG函数的方法：

- 1、使用 数组 f 将 可改变当前状态 的方式记录下来。
- 2、然后我们使用 另一个数组mex 将当前状态x 的后继状态标记。
- 3、最后模拟mex运算，也就是我们在标记值中 搜索 未被标记值的最小值，将其赋值给 SG(x)。
- 4、我们不断的重复 2 - 3 的步骤，就完成了 计算 1~n 的函数值。

```
//f[1..m]:可改变当前状态的方式，m为方式的种类，f[]要在getSG之前先预处理
//SG[]:0~n的SG函数值
//mex[]:为x后继状态的集合
int f[M],SG[N];
bool mex[N];
void getSG(int n)
{
        memset(SG,0,sizeof(SG));
        //因为SG[0]始终等于0，所以i从1开始
        for(int i = 1; i <= n; i++)
        {
                //每一次都要将上一状态 的 后继集合 重置
                memset(mex,0,sizeof(mex));
                for(int j = 0; f[j] <= i ; j++)
                        mex[SG[i-f[j]]] = 1; //将后继状态的SG函数值进行标记
                for(int j = 0; ; j++) //查询当前后继状态SG值中最小的非零值
                        if(!mex[j])
                        {
                                SG[i] = j;
                                break;
                        }
        }
}
```

课堂练习：

- 自行用计算SG函数的方式编写程序：hdu 1847

二、尼姆博弈(Nimm Game)

- 有 k 堆各若干个石子，两个人轮流取，规定每次可以任选某一堆取其中任意多个石子，不能不取。最后取光者得胜。

分析

- 由于石子的总数量在严格减小，此博弈是有限的；
- 且玩家可以知晓对手的行动，双方均具有完全信息；
- 且博弈中不含运气成分；

- 那么由策梅洛定理可知，先手方或后手方有必胜策略。

例：k=2

- 我们从最简单的两堆石子的情形开始分析，并使用二元组（n，m）来表示当前局势，其中 n, m 分别表示这两堆石子当前的数量。
- 我们将说明：当 n=m 时，(n, n) 是奇异局势，即先手必败 。
- （1）显然，当 n = m = 0 时，(0, 0) 是奇异局势。
- （2）当 n = m > 0 时：
- ①当n=m=1时，先手方仅能够将其中一堆完全取走，故后手方只需要将另一堆的1个石子取走即可获胜。即（1，1）是奇异局势。
- ②当n=m=i>1时，无论先手方选择哪一堆，取走多少数量的石子，后手方只需要选择另一堆，并取走相同数量的石子，就可以将石子堆的状态从（i，i）转换到（j，j），其中j<i。如此操作下去，由于两堆石子的数量保持相等且严格递减，故后手方总能将状态转换为（1，1）或（0，0）。由上面的讨论与游戏规则知，后手方必胜。
- 综上，（n，m），当 n = m 时是奇异局势。

- 当 n != m 时，先手方只需要从较多的一堆石子中拿取适量的石子，使得两堆石子数量一致，就将情形转换为了上述n=m的情况，且自身处于后手方。由上面的讨论知，先手方必胜。即（n，m），当 n != m 时是必胜态。

- 结论：两堆石子初始数目相同，先手必败。否则，先手必胜。

- 多堆呢?

尼姆和

- 为了将两堆的简单情形推广到多堆物品的一般情形，我们需要引入物品堆的尼姆和这一概念。

- 定义：物品堆的尼姆和是由所有物品堆中物品的数量进行异或运算得到的。设有k个物品堆，分别有 n1,n2,...,nk 个物品，那么物品堆的尼姆和为：
  Nim_Sum = n1 ⊕ n2 ⊕ ... ⊕ nk

- 利用此概念我们会发现：多堆物品情形中，所有物品堆的尼姆和为0这一条件，与两堆物品情形中 n=m 这一条件具有完全相同的地位。

Bouton定理

- 状态（x1,x2.....xn）为必败状态当且仅当他们的尼姆和为0. 即：
- x1 XOR x2 XOR ..... XOR xn = 0

- （从二进制位的角度上说，必败状态每一个bit位上1的个数都是偶数。）
- 当前状态为 0 时，进行的某个操作总能使Nim sum变为非0（因为改变任何一个数字（即任何一堆石子），它的二进制形式会有一位或多位的变化，此时原本各位上都为0 的Nim sum就会有所变动（某些位会变成1）），即所有后继都是必胜状态了；当前状态为非0 时，必定会有某个操作能使Nim sum变为0（只需在Nim sum二进制上那些为1的位上面着手即可），即一定有一个后继状态为必败态。

- 在尼姆和为0时，无论如何拿取物品，拿取之后物品堆的尼姆和一定不为0；
- 在尼姆和不为0时，总存在一种拿取物品的方式，使得拿取之后物品堆的尼姆和为0。

例：k=3

- 对于一个奇异局势(A,B,C)，A XOR B XOR C = 0
- 前置知识：对数字a,b 有：a XOR b XOR (a XOR b) = 0
- 对于一个非奇异局势(A,B,C), 我们只需要将C转化为(A XOR B)即可, 即从C这一堆拿走 C-(A XOR B) 个物品（前提：C >= A XOR B）。

例：k=3

- 假设3堆物品数量分别为3，4，5。物品堆的尼姆和可以如下计算：
- 将数量转换为二进制
- $3 = (011)_2$
- $4 = (100)_2$
- $5 = (101)_2$
- 进行异或运算
- 2^0这一位是 $1 \oplus 0 \oplus 1 = 0$
- 2^1这一位是 $1 \oplus 0 \oplus 0 = 1$
- 2^2这一位是 $0 \oplus 1 \oplus 1 = 0$
- 最终结果为$(010)_2$
- 尼姆和不为0，说明初始状态是必胜态，必胜策略可以这样构造：
- 从第1堆中拿走若干物品，使得第1堆物品剩余 $4 \oplus 5 = 1$ 个，即从第1堆中拿走 $3 - 1 = 2$ 个。

玩家的策略

- 就是把面对的非奇异局势变为奇异局势留给对方。也就是从某一堆取出若干石子之后，使得每一个bit位上1的个数都变为偶数，这样的取法一般不只有一种。可以将其中一堆的石子数变为其他堆石子数的位异或运算的值（如果这个值比原来的石子数小的话）。

小知识

- A XOR B XOR B = A，可推广到N个变量，A1 XOR A2 XOR A3 XOR …… XOR AN XOR AN XOR AN-1 XOR …… XOR A3 XOR A2 = A1。

Sprague-Grundy定理（SG定理）

- 一个游戏的SG值等于所有子游戏SG值的异或和。

- 这样就可以将每一个子游戏分而治之，从而简化了问题。而Bouton定理就是Sprague-Grundy定理在Nim游戏中的直接应用，因为单堆的Nim游戏 SG函数满足 SG(x) = x。

HDU 1848 – Fibonacci again and again

- 1、 这是一个二人游戏;
- 2、 一共有3堆石子，数量分别是m, n, p个；
- 3、 两人轮流走;
- 4、 每走一步可以选择任意一堆石子，然后取走f个；
- 5、 f只能是菲波那契数列中的元素（即每次只能取1，2，3，5，8…等数量）；
- 6、 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。

- Input
- 输入数据包含多个测试用例，每个测试用例占一行，包含3个整数m,n,p（1<=m,n,p<=1000）。
- m=n=p=0则表示输入结束。

- Output
- 如果先手的人能赢，请输出“Fibo”，否则请输出“Nacci”，每个实例的输出占一行。

- Sample Input
- 1 1 1
- 1 4 1
- 0 0 0
- Sample Output
- Fibo
- Nacci

分析

- 根据SG定理，我们可以将其转化成三个子游戏，最后只需要将这三个子游戏的SG函数xor起来即可判断胜负关系。

主函数

```
int main()
{
	init();
	int m,n,p;
	while(scanf("%d%d%d",&m,&n,&p), m)
		//根据SG定理求解答案
		if(SG[m]^SG[n]^SG[p]) puts("Fibo");
		else puts("Nacci");

	return 0;
}
```

getSG()

```
#define N 1005
int Fib[25],SG[N];;
bool vis[N];
void getSG()//预处理出每一个状态的SG函数
{
        Fib[1]=Fib[2]=1;
        for(int i=3;i<=20;i++)
                Fib[i]=Fib[i-1]+Fib[i-2];
        for(int i=0;i<=1001;i++)
        {
                memset(vis,0,sizeof(vis));
                for(int j=1;Fib[j]<=i;j++)
                        vis[SG[i-Fib[j]]]=1;
                for(int j=0;;j++)
                        if(!vis[j])
                        {
                                SG[i]=j;
                                break;
                        }
        }
}
```

课堂练习

- Light OJ 1199： Partitioning Game

- T组数据
- 有n(1<=n<=100)堆石子，每一堆分别有ai(1<=ai<=10000)个石子
- 一次操作可以使一堆石子变成两堆数目不相等的石子
- 最后不能操作的算输，问先手胜还是后手胜

- Sample Input
- 3
- 1
- 4
- 3
- 1 2 3
- 1
- 7

- Sample Output
- Case 1: Bob
- Case 2: Alice
- Case 3: Bob

主函数

```
int main()
{
	getSG();
	int T;
	cin>>T;
	for(int t=1;t<=T;t++)
	{
		int n,x,NimSum=0;
		cin>>n;
		for(int i=1; i<=n; i++)
		{
			cin>>x;
			NimSum^=sg[x];
		}
		if(NimSum)
			printf("Case %d: Alice\n",t);
		else
			printf("Case %d: Bob\n",t);
	}
	return 0;
}
```

getSG()

```
#define N 10005
int sg[N];
bool vis[10005];
void getSG()
{
        memset(sg,0,sizeof(sg));
        for(int i=1; i<=10000; i++)
        {
                memset(vis,0,sizeof(vis));
                for(int j=1; j+j<i; j++)
                        vis[sg[j]^sg[i-j]]=1;
                for(int j=0;;j++)
                        if(!vis[j])
                        {
                                sg[i]=j;
                                break;
                        }
        }
}
```

POJ2311 Cutting Game

- 两个人在玩游戏。游戏规则如下：准备一张分成 W × H 的格子的长方形纸张，参与游戏的两个人轮流沿着格子的边界线切割纸张，水平或者垂直的奖纸张切割成两个部分。切割了n次之后就得到了n+1张纸，每次都选择切得的某一张纸再次进行切割。首先切出只有一个格子的纸张（1 x 1的各自组成的纸张）的一方获胜。
- Input
- 输入包含多组测试样例。每组测试样例包含两个整数W和H(2 <= W, H <= 200)，分别表示纸张的长和宽。
- Output
- 对于每个测试样例输出一行：如果先手必胜输出"WIN"，否则输出 "LOSE"。
- Sample Input
- 2 2
- 3 2
- 4 2
- Sample Output
- LOSE
- LOSE
- WIN

主函数

```
int main()
{
	getSG();
	int w,h;
	while(scanf("%d%d", &w, &h)!=EOF)
	{
		if(sg[w][h]) printf("WIN\n");
		else printf("LOSE\n");
	}
	return 0;
}
```

getSG()

```
void getSG()
{
    for(int i=2; i<=200; i++)
    {
        for(int j=2; j<=200; j++)
        {
            memset(vis, 0, sizeof(vis));
            //当出现w==1或h==1就输了就没得转移了
            for(int k=2; i-k>=2; k++)//枚举可以转移到的转态
                vis[sg[k][j]^sg[i-k][j]]=1;
            for(int k=2; j-k>=2; k++)//枚举可以转移到的转态
                vis[sg[i][k]^sg[i][j-k]]=1;
            sg[i][j]=0;
            for(int k=0;;k++)
            if(vis[k]==0)//计算对应的sg数
            {
                sg[i][j]=k;
                break;
            }
        }
    }
}
```

例：lightoj 1296 Again Stone Game

- 有 n 堆石子，每人每次的操作是：
- 任选一堆石子，从中至少取1个，至多取不超过该堆石子个数的一半。(特殊：若一堆仅有1个石子取不了)
- 数据范围：每堆石子个数 [1, 1e9].

分析

- 数据范围：每堆石子个数 [1, 1e9].

- 计算每一个 sg 函数存储下来肯定不行，所以打表找 sg 结果的规律。

打表

```
#include<bits/stdc++.h>
using namespace std;
const int N=105;
int sg[N];
bool vis[N];
void getSG(int n)
{
	//memset(sg,0,sizeof sg);
	sg[0]=sg[1]=0;
	for(int i=2;i<=n;i++)
	{
		memset(vis,0,sizeof vis);
		int m=i>>1;
		for(int j=1;j<=m;j++)
			vis[sg[i-j]]=1;

		for(int j=0;;j++)
			if(!vis[j]) {sg[i]=j; break;}
	}
}
int main()
{
	int n=15;
	getSG(n);
	for(int i=0;i<=n;i++)
		printf("%3d",i);
	printf("\n");
	for(int i=0;i<=n;i++)
		printf("%3d",sg[i]);
}
```

- 打表如下：

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	1	0	2	1	3	0	4	2	5	1	6	3	7	0

- 发现规律是：

参考代码

```
#include <bits/stdc++.h>
using namespace std;
const int N=1e3+10;
int calc(int n)
{
	if(n&1) return calc(n/2);
	return n/2;
}
int main()
{
	int T;
	cin>>T;
	for(int t=1;t<=T;t++)
	{
		int n;
		scanf("%d",&n);
		int NimSum=0, x;
		for(int i=1;i<=n;i++)
		{
			scanf("%d",&x);
			NimSum^=calc(x);
		}

		if(NimSum)printf("Case %d: Alice\n",t);
		else printf("Case %d: Bob\n",t);
	}
	return 0;
}
```

- 求解SG函数还有一个深搜版本，具体实现和循环差不多。
- DFS一般只在打表解决不了的情况下用，首选打表预处理。

- 打表解决不了的情况?
- MLE
- TLE

例：HDU 1536 S-Nim

- 多组数据
- 每组数据先给出 k 个数的集合 s，再给 m 组数据，每组 l 堆石子，每堆石子数为hi，每次可以取的个数只能是集合 s 中的数。问先手胜还是输？

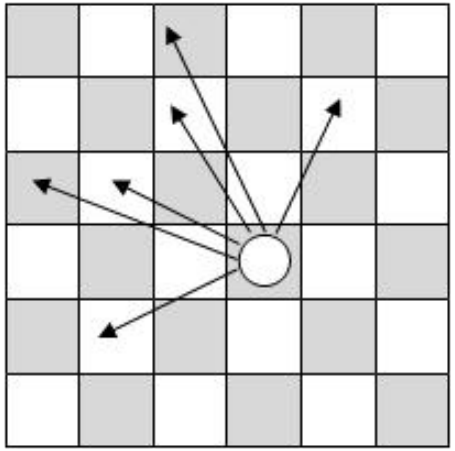
- $0 < k \leq 100$
- $0 < si \leq 10000$
- $0 < m \leq 100$
- $0 < l \leq 100$
- $0 \leq hi \leq 10000$

- Sample Input
- 2 2 5
- 3
- 2 5 12
- 3 2 4 7
- 4 2 3 7 12
- 5 1 2 3 4 5
- 3
- 2 5 12
- 3 2 4 7
- 4 2 3 7 12
- 0

- Sample Output
- LWW
- WWL

LightOJ 1315 Game of Hyper Knights

- 题意：
- T组数据($T \le 200$)
- 棋盘左上角的格子坐标为(0,0)，有n个骑士($1 \le n \le 1000$)，给定n个骑士的坐标(xi,yi)，($0 \le x, y < 500$)。每人每次任选一个骑士移动一步。骑士不能出上边界和左边界。最后不能移动骑士的算输，问先手胜还是后手胜。



- Sample Input
- 2
- 1
- 1 0
- 2
- 2 5
- 3 5

- Sample Output
- Case 1: Bob
- Case 2: Alice

- 思路：n个骑士相互独立，所以可以应用SG定理。
- 其一，注意到骑士总是往当前对角线的左边走，所以在计算每一格的SG函数时可以按照row+col为定值依次计算。
- 其二，注意到骑士只有六种走法，所以对于每一格的SG函数值SG(x,y)不会超过6。

- 此题打表不好处理，只好DFS。

主函数

```
int main()
{
	memset(sg,-1,sizeof(sg));
	int T;
	cin>>T;
	for(int t=1;t<=T;t++)
	{
		int n,x,y,NimSum=0;
		cin>>n;
		for(int i=1;i<=n;i++)
		{
			cin>>x>>y;
			NimSum^=dfs(x,y);
		}
		printf("Case %d: %s\n",t,NimSum?"Alice":"Bob");
	}
	return 0;
}
```

记忆化dfs计算sg函数

```
int dfs(int x,int y)
{
	int vis[105]={0};
	if(~sg[x][y]) return sg[x][y];
	for(int i=0;i<6;i++)
	{
		int nx=x+dir[i][0], ny=y+dir[i][1];
		if(nx>=0&&ny>=0)
			vis[dfs(nx,ny)]=1;
	}
	for(int j=0;;j++)
		if(!vis[j]) return sg[x][y]=j;
}
```

三、威佐夫博弈(Wythoff Game)

- 有 2 堆各若干个石子，两个人轮流取，规定每次可以任选某一堆取其中任意多个石子，或同时从两堆中取同样多的石子，不能不取。最后取光者得胜。（POJ1067）

- 设这两堆石子分别有(A,B)个，并且A<=B。我们先来看一下先手必败的局势。

- 1、(0,0) 先手必败，很明显他没得取了。

- 2、(1,2) 先手必败。
- 具体分析一下，先手有以下几种取法：
- （1）取第一堆的1个，后手取走第二堆的2个获胜。
- （2）从第一堆第二堆各取1个，后手取走第二堆剩下的1个获胜。
- （3）取第二堆的1个，后手从第一堆第二堆各取1个获胜。
- （4）取第二堆的2个，后手取走第一堆的1个获胜。
- 综上所述，先手必败。

- 3、(3,5) 先手必败。

- 首先可以明确的一点是，先手不能把任意一堆取完，如果取完显然必败。

- （1）先讨论先手只从第一堆中取的情况
- 先手可以从第一堆中取1个，后手从第二堆中取4个，转化为(1,2),先手必败。
- 先手可以从第一堆中取2个，后手从第二堆中取3个，转化为(1,2),先手必败。

- （2）再讨论先手只从第二堆中取的情况
- 先手可以从第二堆中取1个，后手从两堆中各取2个，转化为(1,2)，先手必败。
- 先手可以从第二堆中取2个，后手从两堆中各取3个，转化为(0,0)，先手必败。
- 先手可以从第二堆中取3个，后手从两堆中各取1个，转化为(1,2)，先手必败。
- 先手可以从第二堆中取4个，后手从第一堆中取2个，转化为(1,2)，先手必败。
- （3）最后讨论先手从两堆中同时取的情况
- 先手可以从两堆中各取1个，转化为(2,4)，此时情况较多：
- 后手足够聪明，他从第二堆中取了1个，转化为(2,3)，先手也足够聪明，他为了不直接输掉，从第一堆中取了1个(其他取法直接输掉了)，转化为(1,3)，此时后手从第二堆中取1个，转化为(1,2)，先手必败。
- 先手可以从两堆中各取2个，后手从第二堆中取一个，转化为(1,2)，先手必败。

- 综上所述，先手必败。

- 其他的先手必败局势
- (4,7),(6,10),(8,13),(9,15),(11,18)

- 现在把它们看成一个奇异局势组成的序列
(0,0),(1,2),(3,5),(4,7),(6,10),(8,13),(9,15),(11,18).....

- 我们会发现这个序列的规律，每个奇异局势的差值是自然数列。
- 设序列第k个奇异局势元素为(Ak,Bk)，k为自然数。那么，初始条件k=0时是，A0=B0=0，递推关系为下一个奇异局势的Ak是未在前面出现过的最小自然数，且Bk = Ak + k。

奇异局势的判定

- 上面发现了奇异局势的递推公式，那么给出一个局势，如何判断其是否是奇异局势呢?

- 黄金分割数：数学真奇妙，竟然发现，这个序列跟黄金分割数有关系。黄金分割数是2/(1+sqrt(5))=(sqrt(5)-1)/2，其小数约等于0.61803398875，其倒数是(1+sqrt(5))/2，约等于1.61803398875。

- 这个奇异局势的序列的通项公式可以表示为：
- Ak = [k*(1+sqrt(5.0)/2]
- Bk = Ak + k
- 其中k=0，1，2，...,n ，方括号表示int取整函数。

- 有了这个通项式子，逆向的，对于某一个局势，只需要判断其A是否是黄金分割数的某个k的倍数，然后再确认B是否等于A+k即可。
- 或者：对于一个奇异局势(A,B), A = 下取整[(B-A) * 1.618]，更准确的说，1.618 = (sqrt(5) + 1) / 2.

玩家的策略

- 就是把面对的非奇异局势变为奇异局势留给对方。也就是说，玩家必须知道临近的奇异局势是什么，然后把当前局势变成奇异局势。

常见的几类问题

- 1、给出一个局面（n,m)，判断先手输赢。(检查是否是奇异局势)

- 首先求出差值，差值 * 1.618 == 最小值 的话后手赢，否则先手赢。（注意这里的1.618最好是用上面式子计算出来的，否则精度要求高的题目会错)

- 2、给出局面，判断先手输赢，若先手赢，求出其首步取法。

- 首先讨论在两边同时取的情况，很明显两边同时取的话，不论怎样取他的差值是不会变的，那么我们可以根据差值计算出其中的小的值，然后加上差值就是大的一个值，当然能取的条件是求出的最小的值不能大于其中小的一堆的石子数目。
- 假如在一堆中取的话，可以取任意一堆，那么其差值也是不定的，但是我们可以枚举差值，差值范围是0 --- 大的石子数目，然后根据上面的理论判断满足条件的话就是一种合理的取法。

其他：斐波那契博弈

- hdu 2516
- 基本描述：
- 有一堆个数为n（n>=2）的石子，游戏双方轮流取石子，满足：

- 1、先手不能在第一次把所有的石子取完，至少取1颗；
- 2、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。

- 这个和之前的取石子游戏有一个很大的不同点，就是游戏规则的动态化。之前的规则中，每次可以取的石子的策略集合是基本固定的，但是这次有规则2：一方每次可以取的石子数依赖于对手刚才取的石子数。

- 这个游戏叫做Fibonacci Nim，肯定和Fibonacci数列：1,2,3,5,8,13,21,34,55,89,… 有密切的关系。

- 结论
- 当且仅当n不是Fibonacci数时，先手必胜。换句话说，先手必败构成Fibonacci数列。

- 分析
- 证明需要前置技能，“Zeckendorf定理”（齐肯多夫定理），其表述为：任何正整数可以表示为若干个不连续的Fibonacci数之和。

- 具体证明，自行研究。

- 分解的时候，要取尽量大的Fibonacci数。
- 比如分解85：85在55和89之间，于是可以写成85=55+30，然后继续分解30, 30在21和34之间，所以可以写成30=21+9，
- 依此类推，最后分解成85=55+21+8+1。
- 则我们可以把n写成 $n = f[a_1]+f[a_2]+\ldots\ldots+f[a_p]$。 （$a_1>a_2>\ldots\ldots>a_p$）
- 我们令先手先取完$f[a_p]$，即最小的这一堆。由于各个 f 之间不连续，则$a_{p-1} > a_p + 1$，则有$f[a_{p-1}] > 2*f[a_p]$。即后手只能取$f[a_{p-1}]$这一堆，且不能一次取完。
- 此时后手相当于面临这个子游戏（只有$f[a(_{p-1})]$这一堆石子，且后手先取）的必败态，即先手一定可以取到这一堆的最后一颗石子。
- 同理可知，对于以后的每一堆，先手都可以取到这一堆的最后一颗石子，从而获得游戏的胜利。

更多的练习

- POJ 2234
- HDU 4388
- POJ 2975
- HDU 1367
- POJ 2505
- ZOJ 3057
- POJ 2484
- POJ 2425
- POJ 2960
- POJ 1740
- POJ 1704
- POJ 2068
- POJ 3480
- POJ 2348
- HDU 2645
- POJ 3710
- POJ 3533