**博弈**

目录

[一，基础博弈 2](#_Toc527032720)

[1. 巴士博弈 2](#_Toc527032721)

[2. 威佐夫博弈 2](#_Toc527032722)

[3. 斐波那契博弈 2](#_Toc527032723)

[4. nim博弈 2](#_Toc527032724)

[二，高阶博弈 2](#_Toc527032725)

[1. 抛硬币问题 2](#_Toc527032726)

[2. K倍动态减法游戏 3](#_Toc527032727)

[3. 阶梯博弈 5](#_Toc527032728)

[4. \*NIM积 6](#_Toc527032729)

[5. 树上删边游戏 7](#_Toc527032730)

# 一，基础博弈

## 1. 巴士博弈

1堆石头 一次可取1-m个 结论:n%(m+1)=0时先手必败

## 2. 威佐夫博弈

有两堆若干个物品，两个人轮流从某一堆或同时从两堆取同样多的物品，规定每次最少取1个，最后取光者胜。

时先手必败。

## 3. 斐波那契博弈

1堆石子有n个，两个轮流取。先取者第一次可以取任意多个，但不能全部取完。以后每次取的石子数不能超过上次取子数的2倍，取完者胜。

结论:当n为斐波那契数时，先手必败。

## 4. nim博弈

有n堆若干个物品，两个轮流从某一堆取任意多的物品，规定每次至少取一个，多者不限，最后取光者得胜。异或和为0时先手必败。

# 二，高阶博弈

## 1. 抛硬币问题

一般的翻硬币的游戏规则是这样的:

N枚硬币排成一排，有的正面朝上，有的反面朝上。我们从左开始对硬币按1到N编号。

第一，游戏者根据某些约束翻硬币，但他所翻动的硬币中，最右边那个硬币必须是从正面翻到方面。例如只能翻3个硬币的情况，那么第三个硬币必须从正面翻到反面。如果局面是正正反，那么就不能翻硬币了，因为第三个是反的。

第二，谁不能翻谁输。

有这样的结论:局面的SG值为局面中每个正面朝上的妻子单一存在时的SG值异或和。即一个有k个硬币朝上，朝上硬币位置分布在的翻硬币游戏中，SG值等于k个独立的开始时只有一个硬币朝上的翻硬币游戏的SG值异或和。比如THHTTH这个游戏中，2号，3号，6号位是朝上的，它等价于TH,TTH,TTTTTH三个游戏和，即SG[THHTTH]=SG[TH]^SG[TTH]^SG[TTTTTH]。

**约束条件一:每次只能翻一个硬币。**

显然对于任意一个正面的硬币,SG=1所以奇数个正面硬币先手必胜，偶数个正面硬币，先手必败。

**约束条件二: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，初始编号为1，与NIM游戏一样。

**约束条件三:每次必须连续反转k个硬币**

设k=3，枚举一下sg值可以发现sg值的规律为001001001….那么推广一下就是k-1个0加1个1作为一个循环节。

**约束条件4: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，即翻动第x个硬币后，必须选择x-1,x-2,x-3中的其中一个硬币进行翻动，除非x是小于等于3的。(Subtraction Games)**

当N==1时，硬币为:正，先手必赢，所以sg[1]=1.

当N==2时，硬币为反正，因为先手可以翻成反反或正反，可能性为2，所以sg[2]=2

当N==3时，硬币为反反正，先手操作后可以为反反反，正反反，反正反。所以sg[3]=3

位置x:0 1 2 3 4 5 6 7 8 9 10 11 12 13 14

Sg[x]:0 1 2 3 0 1 2 3 0 1 2 3 0 1 2

**约束条件5:每次必须翻动两个硬币，而且这两个硬币的距离在可行集S={1,2,3}中，硬币序号从0开始。(Twins游戏)**

当N==1时，硬币为:正 先手必输 所以sg[0]=0

当N==2时，硬币为反正 先手必赢，所以sg[1]=1

当N==3时，硬币为反反正，先手必应，sg[2]=2

当N==4时，sg[3]=3

当N==5时，sg[4]=0

位置x:0 1 2 3 4 5 6 7 8 9 10 11 12 13 14

Sg[x]:0 1 2 3 0 1 2 3 0 1 2 3 0 1 2

**约束条件6:每次可以翻动一个，二个或三个硬币(Mock Turtles游戏)**

**初始编号从0开始。**

当N==1时硬币为正，先手必胜，所以sg[0]=1

当N==2时硬币为反正，先手操作后可能为反反或正反，方案数为2，所以sg[1]=2.

当N==3时硬币为反反正，先手必胜，操作后可能为，反反反，反正反，正反正，正正反，方案数为4所以sg[2]=4

位置x:0 1 2 3 4 5 6 7 8 9 10 11 12 13 14

Sg[x]:1 2 4 7 8 11 13 14 16 19 21 22 25 26 28

看上去sg为2x或2x+1,我们称一个非负整数为odious,当且仅当该数的二进制形式的1出现次数是奇数，否则成为evil。所以1,2,4,7是odious因为他们的二进制为1,10,100,111而0,3,5,6是evil，因为他们的二进制形式是0,11,101,110.而上面那个表中，貌似sg值都是odious数，所以当2x为odious时，sg值为2x，否则为2x+1.

**约束条件7:每次可以连续翻动任意个硬币，至少翻一个。(Ruler游戏)**

初始编号从1开始。

那么这个游戏的SG函数是g(n)=mex{0,g(n-1,g(n-1)^g(n-2)^g(n-3),…,g(n-1)^…^g(1))}

根据SG函数可以得到SG值表如下。

位置x: 1 2 3 4 5 6 7 8 9 10 11 12 13 14

Sg[x]: 1 2 1 4 1 2 1 8 1 2 1 4 1 2

所以sg值为x的因数当中2的能达到的最大次幂。比如14=2\*7，最大1次幂，即2；16=2\*2\*2\*2，最大4次幂，即16.

这个游戏成为尺子游戏是因为SG函数很像尺子上的刻度。

## 2. K倍动态减法游戏

//hdu2486

//规则和斐波那契博弈类似，但不同的是2倍变成了k倍

/\*

这就是K倍动态减法游戏，可以参考曹钦翔从“k倍动态减法游戏”出发探究一类组合游戏问题的论文。

首先k=1的时候，必败态是2^i,因为我们把数二进制分解后，拿掉最后一个1，那么会导致对方永远也取不完，我们可以拿到最后一个1.

k=2的时候，必败态是斐波那契数列，因为任何一个整数n都可以写成两项斐波那契数的和，所以我们拿掉1，对方永远取不完高两位的数。

k的时候我们必须构造数列，将n写成数列中一些项的和，使得这些被取到的项的相邻两个倍数差距>k 那么每次去掉最后一个1 还是符合上面的条件。设这个数列已经被构造了i 项，第 i 项为a[ i ]，前 i 项可以完美对1..b[ i ] 编码使得每个编码的任意两项倍数>K 那么有

a[ i+1 ] = b[ i ] + 1;这是显然的 因为b[ i ] + 1没法构造出来，只能新建一项表示

然后计算b[ i+1] 既然要使用 a[ i+1 ] 那么下一项最多只能是某个 a[ t ] 使得 a[ t ] \* K < a[ i+1 ] 于是

b[ i ] = b[ t ] + a[ i+1 ]

然后判断n是否在这个数列里面

如果在，那么先手必败。否则不停的减掉数列a中的项构造出n的分解，最后一位就是了

\*/

#include<bits/stdc++.h>

using namespace std;

const int maxn=2e6+5;

long long A[maxn],B[maxn];

int main()

{

int T;

scanf("%d",&T);

int cas=1;

while(T--)

{

long long n,k;

scanf("%lld %lld",&n,&k);

int i=0,j=0;

A[0]=1,B[0]=1;

while(A[i]<n)

{

i++;

A[i]=B[i-1]+1;

while(A[j+1]\*k<A[i])j++;

if(A[j]\*k<A[i])B[i]=A[i]+B[j];

else B[i]=A[i];

}

printf("Case %d: ",cas++);

if(A[i]==n)puts("lose");

else

{

int ans=-1;

for(int j=i;j>=0&&n;j--)

{

if(n>=A[j])

{

ans=j;

n-=A[j];

}

}

printf("%d\n",A[ans]);

}

}

}

## 3. 阶梯博弈

//poj-1704

/\*从左到右有一排石子，给出石子所在的位置。规定每个石子只能向左移动，且不能跨过前面的石子。最左边的石子最多只能移动到1位置。每次选择一个石子按规则向左移动，问先手是否能赢。

我们把棋子按位置升序排列后，从后往前把他们两两绑定成一对。如果总个数是奇数，就把最前面一个和边界（位置为0）绑定。

在同一对棋子中，如果对手移动前一个，你总能对后一个移动相同的步数，所以一对棋子的前一个和前一对棋子的后一个之间有多少个空位置对最终的结果是没有影响的。

于是我们只需要考虑同一对的两个棋子之间有多少空位。

这样一来就成了N堆取石子游戏了.

\*/

#include<iostream>

#include<algorithm>

#include<cstring>

#include<stdio.h>

using namespace std;

const int maxn=1005;

int a[maxn];

int main()

{

int T;

scanf("%d",&T);

while(T--)

{

int n;

scanf("%d",&n);

for(int i=1;i<=n;i++)

scanf("%d",&a[i]);

sort(a+1,a+1+n);

int begin=2;

if(n%2)begin=1;

int ans=0;

a[0]=0;

for(int i=begin;i<=n;i+=2)

{

ans^=(a[i]-a[i-1]-1);

}

if(ans==0)puts("Bob will win");

else puts("Georgia will win");

}

return 0;

}

## 4. \*NIM积

/\*

一个二维矩阵上，有若干个亮着的灯泡

每次选择一个矩阵（右上角的灯泡必须是亮的），改变四个角灯泡的状态,不能操作的人输

Hdu3404

\*/

#include<iostream>

#include<cstdio>

#include<cstring>

#define N 2000000

using namespace std;

int m[2][2]={0,0,0,1};

int Nim\_Mult\_Power(int x,int y){

if(x<2)

return m[x][y];

int a=0;

for(;;a++)

if(x>=(1<<(1<<a))&&x<(1<<(1<<(a+1))))

break;

int m=1<<(1<<a);

int p=x/m,s=y/m,t=y%m;

int d1=Nim\_Mult\_Power(p,s);

int d2=Nim\_Mult\_Power(p,t);

return (m\*(d1^d2))^Nim\_Mult\_Power(m/2,d1);

}

int Nim\_Mult(int x,int y){

if(x<y)

return Nim\_Mult(y,x);

if(x<2)

return m[x][y];

int a=0;

for(;;a++)

if(x>=(1<<(1<<a))&&x<(1<<(1<<(a+1))))

break;

int m=1<<(1<<a);

int p=x/m,q=x%m,s=y/m,t=y%m;

int c1=Nim\_Mult(p,s),c2=Nim\_Mult(p,t)^Nim\_Mult(q,s),c3=Nim\_Mult(q,t);

return (m\*(c1^c2))^c3^Nim\_Mult\_Power(m/2,c1);

}

int main(){

int t,n,x,y;

scanf("%d",&t);

while(t--){

scanf("%d",&n);

int ret=0;

while(n--){

scanf("%d%d",&x,&y);

ret^=Nim\_Mult(x,y);

}

if(ret)

puts("Have a try, lxhgww.");

else

puts("Don't waste your time.");

}

return 0;

}

## 5. 树上删边游戏

给定一颗n个点的有根树，每次可以删掉一个子树，则叶子节点的SG为0，非叶子节点的SG为他所有孩子节点(SG值+1)的异或和。