Hello World ! ---博弈

/\*

博弈 构造 好题

题意：一排石头，一个数字集合，每次只能取若干个连续的石头（数量必须是给出的数字集合里的)。

思路：看了题解之后觉得跟hdu 3032类似，都可以“把一堆石子分成两堆石子”这个操作。

hdu 3032我是打表找规律的，因为数太大。这道题数据不大，直接SG之。

一个可行操作可以把一个区间分成两个区间（如果操作取的是两端的连续石头，则分出的区间里面有一个为空区间'() ），如长度为5，可行操作为2，则取了之后后继可能为{[1,3]} , {[1], [4,5]} , {[1,2], [5]} , {[3,5]} 四种情况，跟Nim一样，当前状态的sg值等于两个后继状态的sg值的抑或和。

\*/

//hdu 2999

int SG(int u)

{

if(u == 0) return 0;

if(sg[u] != -1) return sg[u];

bool vis[MAXN];

memset(vis, false, sizeof(vis));

for(int i = 0; i < n && S[i] <= u; i++) {

for(int j = 0; j <= u - S[i] && j <= u - S[i] - j; j++) //不加j<=u-S[i]-j剪枝会TLE

vis[ SG(j) ^ SG(u - S[i] - j) ] = true;

}

for(int i = 0; ; i++) if(!vis[i]) return sg[u] = i;

}

int main()

{

int m, x;

while(scanf("%d", &n) != EOF) {

memset(sg, -1, sizeof(sg));

for(int i = 0; i < n; i++) scanf("%d", &S[i]);

scanf("%d", &m);

while(m--) {

scanf("%d", &x);

puts(SG(x) ? "1" : "2");

}

}

return 0;

}

/\*

博弈 棋盘博弈 好题 SG的0/1性质

题意：题意：给你一个n\*m的矩形，0表示空着的，1反之，现在两个人轮流放2\*2的矩形，谁不能放了，谁就输了。

思路：sg

从这道题知道了SG在“如果必胜态的所有后继都是必败态”情况时，SG是具有0/1性质的。也就是说，只要知道状态S的某一邻接状态(S->T)为0，则S的sg肯定为1.

\*/

//hdu 1760

bool check(int i, int j)

{

return ! (a[i][j] || a[i][j-1] || a[i-1][j] || a[i-1][j-1]);

}

void label(int i, int j)

{

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

}

bool SG()

{

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

for(int j = 2; j <= m; j++) if(check(i, j)) {

label(i, j);

if(!SG()) {

label(i, j);

return true;

}

label(i, j);

}

return false;

}

int main()

{

while(scanf("%d%d", &n, &m) != EOF) {

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

for(int j = 1; j <= m; j++) scanf("%1d", &a[i][j]);

puts(SG() ? "Yes" : "No");

}

return 0;

}

/\*

博弈 简单 SG

自己懒得写，以下摘自http://blog.csdn.net/liwen\_7/article/details/7948305

题意：

从当前日期，在他/她转的玩家可以移动到下一个历日或下月的同一天。当在之后的一个月中没有在同一天，播放器只能移动到下一个的日历日期。例如，从1924年12月19日，你可以移动到1924年12月20日，下一个日期，或一月19日，1925年，在同一天在下个月。然而，2001年1月31日，你可以只移动2001年2月1日，因为2001年2月31日是无效的。一个球员赢得比赛时，他/她到底到达的日期2001年11月4日。如果一个玩家移动到日期2001年11月4号之后，他/她输了比赛。

思路:

(2001 , 11 , 4)是个必败点，能到(2001, 11 , 4)的是必胜点，由时间从后向前推。

最后若输入的sg[] = 0即为必败点，输出 NO

\*/

//hdu 1079

bool leap\_year(int yy)

{

if(yy % 400 == 0) return true;

else if(yy % 100 == 0) return false;

else return yy % 4 == 0;

}

int calc\_by\_month(int yy, int mm)

{

if(mm == 2) {

return leap\_year(yy)? 29 : 28;

} else if(mm == 1 || mm == 3 || mm == 5 || mm == 7 || mm == 8 || mm == 10 || mm == 12) {

return 31;

} else return 30;

}

bool check(int &yy, int &mm, int &dd)

{

if(dd > calc\_by\_month(yy, mm)) dd = 1, mm++;

if(mm > 12) mm = 1, yy++;

return true;

}

int SG(int yy, int mm, int dd)

{

if(yy > 2001) return 1;

if(yy == 2001 && mm > 11) return 1;

if(yy >= 2001 && mm >= 11 && dd > 4) return 1;

if(yy == 2001 && mm == 11 && dd == 4) return 0;

if(sg[yy][mm][dd] != -1) return sg[yy][mm][dd];

bool vis[5]; // 其实只要2就行了

memset(vis, false, sizeof(vis));

int ty = yy, tm = mm, td = dd;

dd++;

if(check(yy, mm, dd)) vis[SG(yy, mm, dd)] = true;

yy = ty, mm = tm, dd = td;

mm++;

if(mm == 13) mm = 1, yy++; //这部分没考虑到WA了好多次

if(dd <= calc\_by\_month(yy, mm)) {

if(check(yy, mm, dd)) vis[SG(yy, mm, dd)] = true;

}

for(int i = 0; ; i++) {

// if(i >= 4) printf("on!\n");

if(!vis[i]) return sg[ty][tm][td] = i;

}

}

int main()

{

int cases, yy, mm, dd;

scanf("%d", &cases);

memset(sg, -1, sizeof(sg));

while(cases--) {

scanf("%d%d%d", &yy, &mm, &dd);

if(yy > 2001 || (yy == 2001 && mm > 11) || (yy == 2001 && mm == 11 && dd >= 4)) {puts("NO"); continue; } //特判

check(yy, mm, dd);

puts(SG(yy, mm, dd) ? "YES" : "NO");

}

return 0;

}

/\*

博弈 推规律 欧几里得游戏 Euclid's Game

题意：给出a,b，每一步只能用a，b中较大的数减去a，b中较小的数的倍数，最终有一个数为0的取胜。

思路：对于两个数a, b(a>=b)，总会出现这样一个局面(a%b, b)。而如果a/b >= 2的话，则先手有可以选择谁面对(a%b, b)局势的优势。

显然，第一次有这个优势的人胜，即第一面对a/b >= 2局面者胜。

因为没有给出数据范围，所以不知道怎么sg，然后就看了题解。

\*/

//hdu 1525

int main()

{

int a, b;

while(scanf("%d%d", &a, &b), a&&b) {

bool win = 0;

while(true) {

if(a < b) swap(a, b);

if(a % b == 0) break;

if(a / b >= 2) break;

a %= b;

win = !win;

}

puts(!win ? "Stan wins" : "Ollie wins");

}

return 0;

}

/\*

博弈 打表 对称

题意：给一个N，游戏初始有数字num = 1，让游戏者不断乘2-9，谁最先让num >= N谁就赢。

思路：打个sg表，看有规律就直接水了。

还不理解其性质/证明...

\*/

//hdu 1517

bool check(long long n)

{

for(int i = 0;i < top; i++) if(n > limit[i] && n <= limit[i]\*2) return true;

return false;

}

int main()

{

long long n;

limit[0] = 9;

for(int i = 1; i < 9; i++) limit[i] = limit[i-1] \* 2 \* 9;

while(scanf("%I64d", &n) != EOF) {

puts(!check(n) ? "Stan wins." : "Ollie wins.");

}

return 0;

}

/\*

打表程序

int SG(int u)

{

if(u <= 1) return 0;

if(sg[u] != -1) return sg[u];

bool vis[MAXN];

memset(vis, false, sizeof(vis));

for(int i = 2; i <= 9; i++) vis[SG( ((u-1)/i+1) )] = true;

for(int i = 0; ; i++) if(!vis[i]) return sg[u] = i;

}

int main()

{

memset(sg, -1, sizeof(sg));

for(int i = 1; i < MAXN; i++) {

//printf("%3d:%d\n", i, SG(i));

if(SG(i) == 0 && SG(i-1) ) printf("# %d\n", i);

else if(SG(i) != 0 && SG(i-1) == 0) printf("$ %d\n\n", i);

}

return 0;

}

\*/

/\*

博弈 Bash博弈扩展 启发题

题意：一堆石子，一次至少取p个，最多取q个，取走最后一颗的输。

思路：bash博弈由(1-m)变到(p-q).

若n % (p+q) == 0，先手可以先取q个，然后对方取k个的话我再取(q+p-k)个，这样，维持剩余的石子数mod(p+q) == p

其实到了这里可以看出，取石子是以(p+q)为循环节继续下去的。所以找规律时候可以直接考虑p+q >= n >= 0

若n % (p+q) == r, 且r > p，则先手取p~r-1颗都可以，这样，可以同<1>的方法，维持剩余石子数mod(p+q) <=p

若n % (p+q) == r, 且r <= p，则无论先手取多少，后手总是可以利用<1><2>的方法来对付先手。

收获：这种类型的bash取石子，好像都可以先在一个区间内先找出规律，然后再找出他们的循环节，就可以容易地推出规律。如下面的打表方法。

\*/

//hdu 2897

int main()

{

int n, p, q;

while(scanf("%d%d%d", &n, &p, &q) != EOF) {

int win = 1;

if(n % (p + q) == 0) win = 0;

if(n % (p + q) > p) win = 0;

puts(!win ? "WIN" : "LOST");

}

return 0;

}

/\* 某人用sg来打表找出规律，ORZ...

我也想了个找规律的办法，只要打n = 1..(p+q+1)\*2 说不定规律就看出来了。因为目测答案更(p+q)或者(p+q+1)有关，bash博弈升级版嘛。

#include <stdio.h>

#include <string.h>

#define MAXN 65538

int sg[MAXN];

int n, p, q;

int SG(int u)

{

if(u <= p) return 0;

if(sg[u] != -1) return sg[u];

bool vis[MAXN];

memset(vis, false, sizeof(bool) \* (n+1));

for(int i = p; i <= q; i++) vis[SG(u - i)] = true;

for(int i = 0; ; i++) if(!vis[i]) return sg[u] = i;

}

int main()

{

while(scanf("%d%d%d", &n, &p, &q) != EOF) {

memset(sg, -1, sizeof(sg));

puts(SG(n) ? "WIN" : "LOST");

}

return 0;

}

\*/

/\*

博弈 对称性 好题

题意：给你一个由n个硬币组成的环。每次可取1-k个硬币，取最后硬币的胜。

思路：利用对称性解决问题。

若先手不能在第一次翻完，则因为第一次先手把环拆成了链，那么这是只要后首一次翻完或者把其分成相等数量的两段，之后先手不怎操作后手怎么做，做Nim。

\*/

//hdu 3951

int main()

{

int n, k, cases, Cas = 0;

scanf("%d", &cases);

while(cases--) {

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

int win;

if( k >= n ) win = 0;

else if(k == 1) win = !(n % 2);

else win = 1;

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

puts(!win ? "first" : "second");

}

return 0;

}

/\*

博弈 打表找规律

找不出规律，看到有人打表找规律，我也打了一下，发现真的有规律。

下面是cxlove的证明：

..Lasker's Nim游戏：每一轮允许两会中操作之一：①、从一堆石子中取走任意多个，②、将一堆数量不少于2的石子分成都不为空的两堆。

..很明显：sg(0) = 0，sg(1) = 1。

..状态2的后继有：0，1和（1，1），他们的SG值分别为0，1，0，所以sg(2) =2。

..状态3的后继有：0、1、2、（1，2），他们的SG值分别为0、1、2、3，所以sg(3) = 4。

..状态4的后继有：0、1、2、3、（1，3）和（2，2），他们的SG值分别为0，1，2，4，5，0，所以sg(4) = 3.

..由数学归纳法可以得出 sg(4k)=4k-1;sg(4k+1)=4k+1;sg(4k+2)=4k+2;sg(4k+3)=4k+4;

\*/

//hdu 3032

int main()

{

int x, cases, n;

scanf("%d", &cases);

while(cases--) {

scanf("%d", &n);

int ans = 0;

while(n--) {

scanf("%d", &x);

if(x % 4 == 0) ans ^= x-1;

else if(x % 4 == 3) ans ^= x+1;

else ans ^= x;

}

puts(ans ? "Alice" : "Bob");

}

return 0;

}

/\* 打表程序 =====================================================

int SG(int u)

{

if(u == 0) return 0;

if(sg[u] != -1) return sg[u];

bool vis[1000];

memset(vis, false, sizeof(vis));

for(int i = 1; i <= u; i++) vis[SG(u-i)] = true;

for(int i = 1; i <= u/2; i++) vis[SG(i) ^ SG(u-i)] = true;

for(int i = 0; ; i++) if(!vis[i]) return sg[u] = i;

}

int main()

{

memset(sg, -1, sizeof(sg));

for(int i = 0; i < 20; i++) printf("sg[%d] = %d\n", i, SG(i));

return 0;

}

===================================================================\*/

/\*

阶梯博弈扩展

题意：给编号为1-N的盒子里放若干颗石子，若A，B满足:A非空 && (A+B)%3==0 && (A+B)%2==1 则可以从A中取任意石头挪到盒子B去。谁不能挪动谁就输。

思路：因为实现知道是阶梯博弈，所以就尽力往阶梯博弈上靠。

经典阶梯博弈是分奇数阶梯/偶数阶梯的，就想把这道题的阶梯也分个奇偶。然后就打表出所有合法(A,B)，根据(A,B)表找规律，发现确实能一个数确实只属于奇数阶梯或只属于偶数阶梯。继续下去就没分析出奇偶阶梯的分布规律了，直接暴力打表出odd\_jump[i]表第i阶梯是否为奇阶梯。

后面就是经典阶梯博弈了。

\*/

//hdu 3389

int SG(int u)

{

if(u == 0) return 0;

if(sg[u] != -1) return sg[u];

bool vis[105];

memset(vis, false, sizeof(vis));

for(int i = 1; i <= u; i++) vis[SG(u-i)] = true;

for(int i = 0; ; i++) if(!vis[i]) return sg[u] = i;

}

void init()

{

for(int i = 1; i <= 10002; i++) {

int j = i-1;

for( ; j > 0; j--) if((i+j) % 2 == 1 && (i+j) % 3 == 0) {

odd\_jump[i] = !odd\_jump[j];

break;

}

}

}

int main()

{

memset(sg, -1, sizeof(sg));

init();

int cases, Cas = 0, x, n;

scanf("%d", &cases);

while(cases--) {

scanf("%d", &n);

int ans = 0;

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

scanf("%d", &x);

if(odd\_jump[i]) {

ans ^= SG(x);

}

}

printf("Case %d: %s\n", ++Cas, ans ? "Alice" : "Bob");

}

return 0;

}

/\*

博弈论 简单题

题意：bash博弈

小记：如果只剩1~m个，则先手赢；如果剩下m+1个，则后手赢；

对于2\*(m+1) > n > m+1，我们总有办法把数量变成m+1；同理k\*(m+1) > n > (k-1)\*(m+1).

所以只要先手初始时在k\*(m+1)位置，那他就囧了。

\*/

#include <stdio.h>

int main()

{

int cases, n, m;

scanf("%d", &cases);

while(cases--) {

scanf("%d%d", &n, &m);

if(n % (1 + m) == 0) printf("second\n");

else printf("first\n");

}

return 0;

}

/\*

博弈DP 记忆化博弈

题意：题目：有两个队，每个队有n个人，每个人每次有数量限制，取最后一块的输。

思路：dp[i][j]表示第i个人取，还有j块石头 。

当j为0的时候，没有石头，这时候是胜，为1。

后继中有必败态的为必胜态。

\*/

//hdu 2068

int SG(int idx, int remain)

{

if(remain == 0) return 1;

if(sg[idx][remain] != -1) return sg[idx][remain];

for(int j = 1; j <= a[idx] && remain >= j; j++)

if(!SG((idx+1)%(2\*n), remain - j)) return sg[idx][remain] = 1;

return sg[idx][remain] = 0;

}

int main()

{

int s;

while(scanf("%d", &n) != EOF && n) {

scanf("%d", &s);

for(int i = 0; i < 2\*n; i++) {

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

}

memset(sg, -1, sizeof(sg));

printf("%d\n", SG(0, s));

}

return 0;

}

/\*

博弈 翻硬币博弈

题意：每次只能翻1-3个硬币，最右边必须是head-up

思路：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出现的次数是奇数，否则称作evil。而上面那个表中，貌似sg值都是odious数。所以当2x为odious时，sg值是2x，当2x是evil时，sg值是2x+1.

一开始我的sg是nim的sg，跪跪跪...上述结论一点也不看懂...

http://blog.csdn.net/acm\_cxlove/article/details/7854181 题解

http://blog.csdn.net/acm\_cxlove/article/details/7854534 翻硬币小结

\*/

/\*

经典翻硬币游戏结论：局面的sg值为局面种每个正面朝上的棋子单一存在是的sg值的异或值。

\*/

#include <stdio.h>

#include <algorithm>

using namespace std;

int a[1000];

int calc(int x)

{

int sum = 0;

while(x) {

x &= (x-1);

x >>= 1;

}

return sum;

}

int SG(int x)

{

int res = calc(x\*2);

if(res & 1) return 2\*x;

else return 2\*x+1;

}

int main()

{

int n;

while(scanf("%d", &n) != EOF) {

int ans = 0;

for(int i = 0; i < n; i++) scanf("%d", &a[i]);

sort(a, a+n);

int idx = 0;

for(int i = 0; i < n; i++)

if(i == 0 || a[i] != a[i-1]) a[idx++] = a[i];

n = idx;

for(int i = 0; i < n; i++) ans ^= SG(a[i]);

puts(!ans ? "Yes" : "No");

}

return 0;

}

/\*

博弈 every-sg every-nim SG

题意:题目有N个游戏同时进行，每个游戏有两堆石子，每次从个数多的堆中取走数量小的数量的整数倍的石子。取最后一次的获胜。并且N个游戏同时进行，除非游戏结束，否则必须操作。

思路：用sg搞。网上多为直接根据性质搞，不会，找到了这个用sg搞的。

以下摘自：http://blog.sina.com.cn/s/blog\_51cea4040100h3l9.html

于是我们开一个Step数组。

表示对于先手必胜的单一游戏而言，它最少走好多步胜利。对于先手必败的单一游戏而言，它最多走好多步。

这样，我们只需要看最后所有单一游戏最大的step那组的SG是0还是非0就可以断定是否先手必胜了。

很容易得出：

（u是v的子状态）

step[v] = 0； （v为终止状态）

step[v] = max{step[u]} + 1； （sg[v]>0,sg[u]=0）

step[v] = min{step[u]} + 1； （sg[v]==0）

\*/

int SG(int x, int y)

{

int ans1 = 0, ans2 = 999999999;

if(x > y) swap(x, y);

if(x == 0) return 0;

if(sg[x][y] != -1) return sg[x][y];

for(int i = x; i <= y; i += x) {

if(!SG(x, y-i)) {

ans1 = max(ans1, step[x][y-i] + 1);

sg[x][y] = sg[y][x] = 1;

} else ans2 = min(ans2, step[x][y-i] + 1);

}

if(sg[x][y] == 1) {

step[x][y] = step[y][x] = ans1;

return 1;

} else {

sg[x][y] = sg[y][x] = 0;

step[x][y] = step[y][x] = ans2;

return 0;

}

}

int main()

{

int x, y, n;

memset(sg, -1, sizeof(sg));

while(scanf("%d", &n) != EOF) {

int ans1 = 0, ans2 = 0;

while(n--) {

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

if(SG(x, y)) ans1 = max(ans1, step[x][y]);

else ans2 = max(ans2, step[x][y]);

}

puts(ans1 > ans2 ? "MM" : "GG");

}

return 0;

}

/\*

思路题 博弈论 图论 好题

题意：一个无向带点权带边权图，Alice和Bob轮着选取没有被选过的点，选到的点的权值作为分数加给游戏者。如果两个相邻的点被同一人选取到，则这两点之间的边也作为额外权值赋给游戏者。假设Alice和Bob都使用最优策略，问最后的Alice - Bob分数的结果。

思路：哪来的思路呀...比赛的时候哪想得到这个解法呀...只能说大家怎么都这么聪明呀...算法题也就算了，这可是思路题呀，大家都怎么想到的喂...我真的那么笨吗喂喂喂...=\_=

还是黄牛给我讲了他的思考过程：n <= 10^5，要么搜索，要么生成树；在搜索这边啃了很久啃不出来想生成树，生成树果断想kruskal，因为prim的更新耗时，在n=10^5的数量级是过不了的；想到拆边，拆成正负边权，w/-w；但两个点的关系没联系到一起，没搞好；刷了一会儿游戏=\_=；把边权和点权同一起来，就是把边权加到点权上；用刚才的拆成正负边显然不行；灵光一闪，把边拆成1/2, 1/2就可以了，因为他想到了一个数学规律：把一个数拆成两半，这两半相加为原数，相减为0。

下面就直接贴题解的原话吧：

若没有边权，则对点权从大到小排序即可。。

考虑边，将边权拆成两半加到它所关联的两个点的点权中即可。

因为当两个人分别选择不同的点时，这一权值将互相抵消。

\*/

int main()

{

int u, v;

double wi;

while(scanf("%d%d", &n, &m) != EOF) {

for(int i = 1; i <= n; i++) scanf("%lf", &w[i]);

while(m--) {

scanf("%d%d%lf", &u, &v, &wi);

w[u] += wi/2;

w[v] += wi/2;

}

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

double ans = 0;

for(int i = n; i >= 2; i -= 2) ans += w[i] - w[i-1];

printf("%I64d\n", (long long)ans);

}

return 0;

}