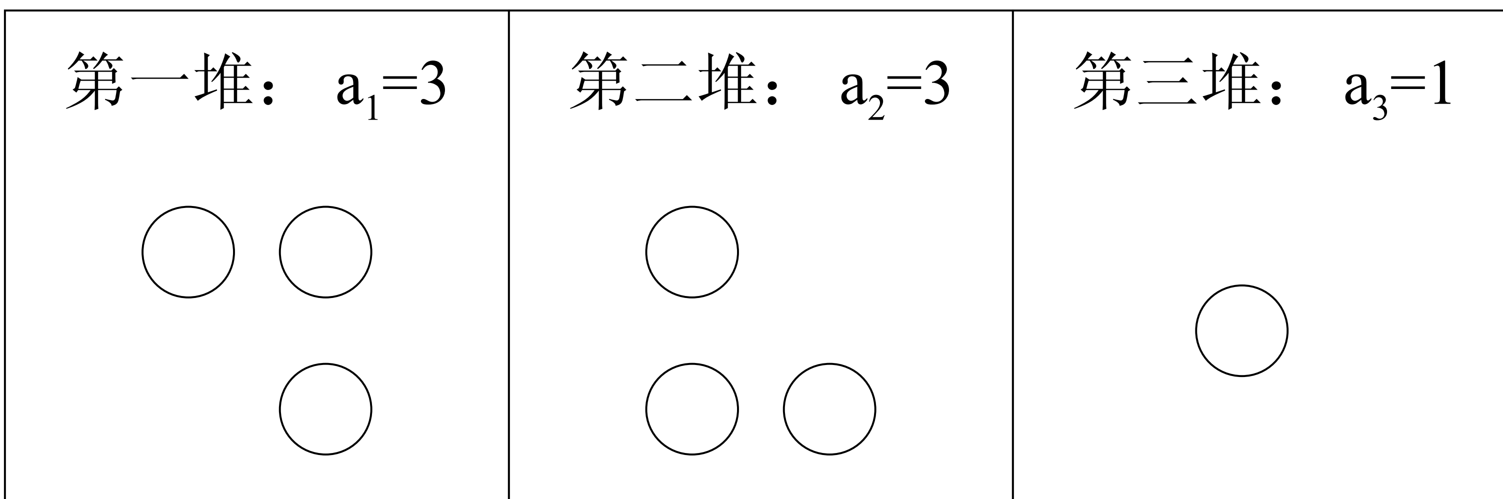
游戏策略

朱全民

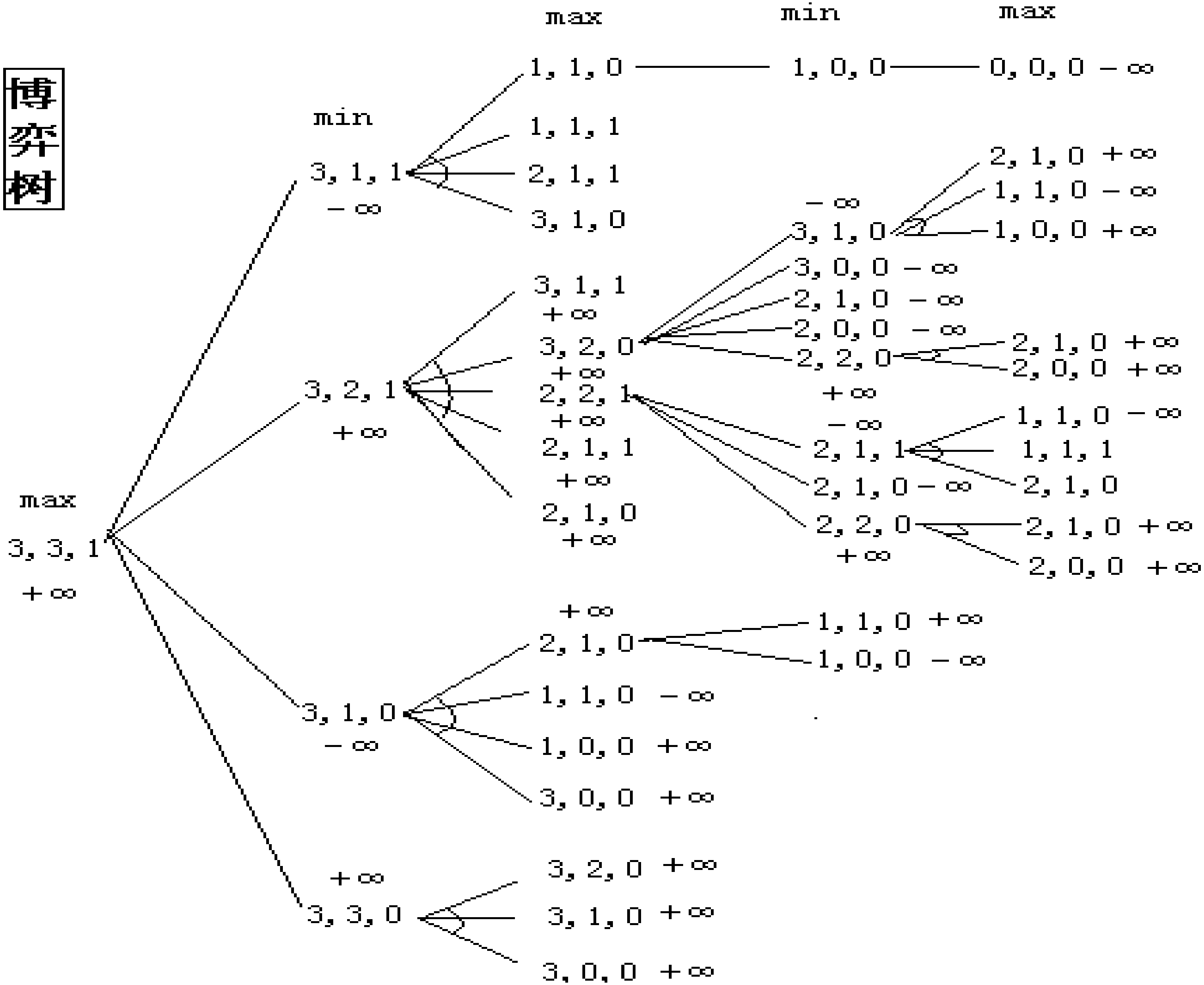
Nim 问题

- 取石子问题

  有 N 堆石子，其中第 i 堆有 P_i 颗石子，每次从某一堆里选出若干石子去掉（但不能不去石子），两人轮流取石，谁不能继续取谁就输了。
- 什么情况下先手必胜，什么情况下后手必胜？



博弈树


max
min
max
min
max
3, 3, 1
+∞
3, 1, 1
−∞
1, 1, 0
1, 0, 0
0, 0, 0 −∞
1, 1, 1
2, 1, 1
3, 1, 0
3, 2, 1
+∞
3, 1, 1
+∞
3, 2, 0
+∞
2, 2, 1
+∞
2, 1, 1
+∞
2, 1, 0
+∞
−∞
3, 1, 0
2, 1, 0 +∞
1, 1, 0 −∞
1, 0, 0 +∞
3, 0, 0 −∞
2, 1, 0 −∞
2, 0, 0 −∞
2, 2, 0
+∞
2, 1, 0 +∞
2, 0, 0 +∞
−∞
2, 1, 1
1, 1, 0 −∞
1, 1, 1
2, 1, 0
2, 1, 0 −∞
2, 2, 0
+∞
2, 1, 0 +∞
2, 0, 0 +∞
3, 1, 0
−∞
+∞
2, 1, 0
1, 1, 0 +∞
1, 0, 0 −∞
1, 1, 0 −∞
1, 0, 0 +∞
3, 0, 0 +∞
+∞
3, 3, 0
3, 2, 0 +∞
3, 1, 0 +∞
3, 0, 0 +∞

分析

- 从上述博弈树可以看出 3,3,1 是必胜点，那么我们可以这么想，如果某个点是必胜点，则取完棋子后，必须使得对方落在必败点。
- 若只有一堆石子，先收走必胜
- 若有 m 堆石子，每堆只有一颗石子， m 堆为奇数时，先手必胜。
- 若有 m 堆石子，每堆有 k 颗石子， m 堆为奇数时，先手必胜。

  第 1 次，先手取 k 棵，轮到对手走时，若对手取 k 棵，则先手也取 k 棵，若对手取 x<k 棵，则先手也取另外一堆的 x 棵，因为剩下的是偶数堆，总能将剩下的堆变成若干个两两相等的堆。只要始终保持这种取法，先手总能取到最后的石子。

一般情况？

- 假设某个初始局面为先手必胜，那么先手每走一步都必须使得对手落在必败节点。
- 因此，对于每一个局面，要么为胜局面，要么为负局面，如果我们将胜局面非 0 表示，那么负局面就可以用 0 表示。
- 因此，对于某一个局面，若为非 0 局面，它的任务就是要寻找某一种取法，使得局面变为 0 局面。那么他的对手无论怎么取，都会使得局面又变成 0 局面。
- 有什么规律呢？

结论

定理：

如果一个局面先手必胜，就称之为 N 局面，反之称之为 P 局面。对于一个局面，令 $S=P_1$ XOR P_2 XOR P_3 XOR … XOR P_n。若 S=0 则为 P 局面，否则为 N 局面。

证明：

- 当 $P_1=P_2=....=P_n=0$ 时， S=0，满足终状态是 P 局面。
- 若 S=0，即 P_1 XOR … XOR $P_n=0$，若取堆 i 中的石子， $P_i>P_i'$， S ->S'， $P_i>P_i'$，则 P_i XOR P_i'<>0。所以 S' XOR P_i XOR P_i'=S=0，即 S'=P_i XOR P_i'<>0。满足 P 局面的所有子局面都是 N 局面。
- 若 S<>0，设 S 的二进制位是 $A_1…A_n$，考虑第一位是 1 的。在 P 中取出该位同是 1 的，不妨设为 P_1。可知 P_1 XOR S<P1，令 $P_1'=P_1$ XOR S。可知 P_1' XOR P_2 XOR … XOR $P_n=0$。即 N 局面存在至少一个子局面是 P 局面。

示例

P1, P2, P3, P4 =33, 9, 30, 11

$$
\begin{array}{r}
100001\\
001001\\
011110\\
\oplus\ 001011\\
\hline
111101
\end{array}
$$

A1, A2, A3, A4, A5, A6 = 111101

取P1 100001

变化为 011101 =29

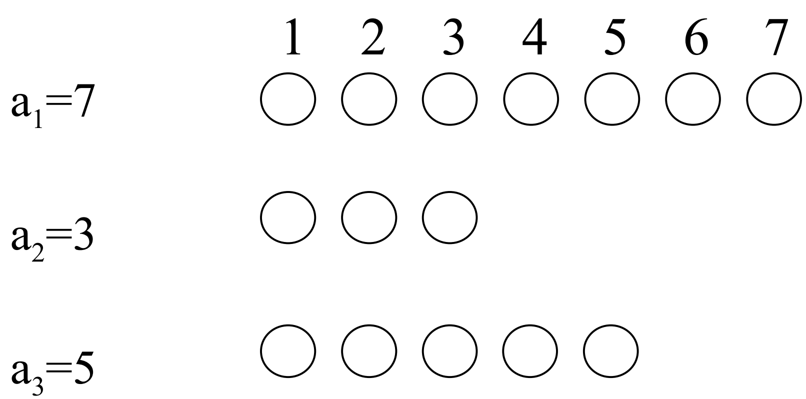
P1, P2, P3, P4 =29, 9, 30, 11

$$
\begin{array}{r}
011101\\
001001\\
011110\\
\oplus\ 001011\\
\hline
000000
\end{array}
$$

Nim 问题的扩展

- 取石子问题

  有 N 堆石子，其中第 i 堆有 P_i 颗石子，每次去掉某一堆里最多 m 棵石子（ m>0 ），两人轮流取石，谁不能继续取谁就输了。
- 什么情况下先手必胜，什么情况下后手必胜？
- 示例 m=2



结论

- 将 $P_1P_2P_3 \ldots P_n$ 对 m+1 求余得到 $P_1'P_2'P_3' \ldots P_n'$，然后符合定理一的结果，记 $S=P_1'$ XOR P_2' XOR P_3' XOR … XOR P_n'。若 S=0 则为 P 局面，否则为 N 局面。

证明：

- 将 $P_1P_2P_3 \ldots P_n$ 分解成为两部分 $P_1'P_2'P_3' \ldots P_n'$ 和 $R_1R_2R_3 \ldots R_n$，其中 $R_1R_2R_3 \ldots R_n$ 都是 m+1 的倍数。
- 若对 $P_1'P_2'P_3' \ldots P_n'$ 部分取子，则按 NIM 方法走步，若对 $R_1R_2R_3 \ldots R_n$ 部分取子，则后手取 k 颗，先手方取 m-k+1 颗，先手始终保持不对 $R_1R_2R_3 \ldots R_n$ 部分先取子。
- 若初始局面为胜局面，$P_1'P_2'P_3' \ldots P_n'$ 部分 NIM 方法取子必胜，由于 $R_1R_2R_3 \ldots R_n$ 都为 m+1 的倍数，因此，按 m+1 互补的取法，先手一定能取到最后 K<=m 颗石子。

Nim_k 问题

- 取石子问题

  有 N 堆石子，其中第 i 堆有 P_i 颗石子，每次可以从最多 K 堆中选出若干石子去掉（但不能不去石子），两人轮流取石，谁不能继续取谁就输了。
- 什么情况下先手必胜，什么情况下后手必胜？

结论

- K=1，为 Nim 问题。
- 对于 K>1 的情况，我们令把 P_1~P_n 这 n 个数，转成二进制，然后每位分别相加，每位最后结果 mod (K+1) 即可。如果每一位结果都是 0，则为 P 局面，否则是 N 局面
- 示例

  $P_1P_2P_3P_4$=3,5,10,15 ， K=2

3			1	1
5		1	0	1
10	1	0	1	0
15	1	1	1	1
	2	2	0	0

  所以这是 N 局面。
- 证明与 NIM 证明类似。下面我们看看具体的取法。

Nim_k 问题的取石子方法

- 设 $P_1P_2P_3\ldots P_n$ 为 n 堆石子数目。 P_i 已标记的石子堆， D0[i] 和 D1[i] 分别表示所有已标记的石子堆中第 i 位为 0 和 1 的总数。

1. 找出加法结果非 0 的最高位，设为 W 。
2. 找出一个二进制第 W 位为 1 、而且未标记的石子堆 Pi ，将 Pi 标记，并把它的第 W 位由 1 改为 0 。对于 Pi 的第 1 到 (W-1) 位，逐个判断，第 j 位如果为 0 则 Inc(D0[j]) ，否则 Inc(D1[j]) 。
3. 若更新后的 S 某一位非 0 （即 S[i]≠0 ）， 且 S[i]+D0[i]>K ，或 S[i]-D1[i]<1 ，可以通过修改以前已标记的石子堆将 S[i] 修正为 0 。
4. 如果加法结果已经全部为 0 ，则确认所有已经做的更改，并结束；否则转到 1 。

P1, P2, P3, P4 =33, 9, 30, 11 k=2

```
   100001
   001001
   011110
⊕  001011
---------
   110120
```

```
   000010
   001001
   011110
⊕  001011
---------
   010102
```

A1, A2, A3, A4, A5, A6 =	110120
取P1	100001
标记P1	000001
D0:	11110
D1:	00001
按规则修改P1	000010

A1, A2, A3, A4, A5, A6 =	010102
取P3	100001
标记P3	001110
D0:	1112
D1:	1111
按规则修改P3	001011

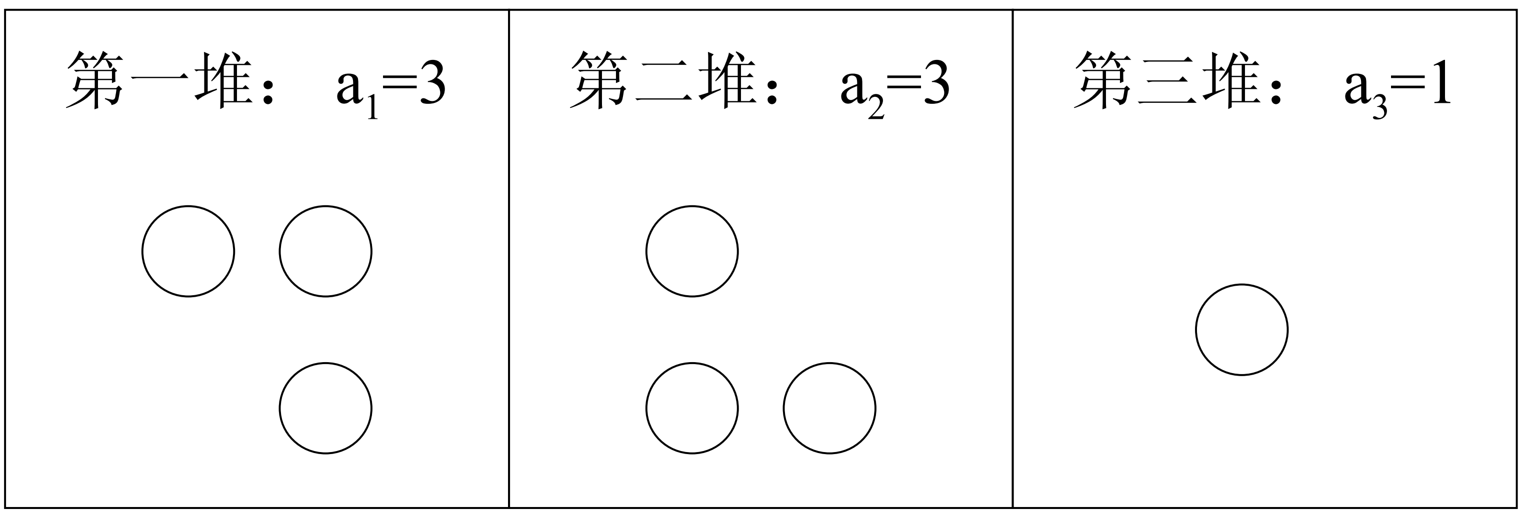
P1, P2, P3, P4 = 2 , 9, 11, 11 k=2

```
   000010
   001001
   001011
⊕  001011
---------
   000000
```

MisèreNim 问题

- 取石子问题

  有 N 堆石子，每次从某一堆里选出若干石子去掉（但不能不去石子），两人轮流取石，谁不能继续取谁就赢了。
- 什么情况下先手必胜，什么情况下后手必胜？



结论

1. 所有石子堆的数目都为 1：显然，若有偶数堆石子堆，则必胜，否则必败。
2. 如果恰好只有一堆石子数目大于 1 。我们可以把这堆石子取完或者取得只剩下 1，使得只剩下奇数堆数目为 1 的石子留给对方，由 1)，必胜。
3. 如果有至少 2 堆石子的数目大于 1 。

   考虑异或值：若异或值不为 0，则按照 Nim 走法取石。这样，当对手某次取完石子后，肯定会出现情况 2，必胜。

   证明：按照 Nim 走法则取完石子后，必定会给对手留下异或值为 0 的局面。因此不可能给对手留下情况 2 的局面（容易证明，情况 2 局面的异或值肯定不为 0），而对手一次最多将一堆石子数大于 1 的石子堆处理掉。因此情况 2 的情况肯定会出现。

   相反，若异或值为 0，则无论如何走都会给对方留下情况 2 或情况 3 的情况，必败。

SG 函数简介

- 如果我们把游戏中的某一个局面看作一个顶点，把局面之间的转换用边来表示，那么很多游戏都可以转化成**图游戏**模型。
- **图游戏模型** 给定有向无环图 G=(V,E) 和一个起始点，双方轮流行动。每个人每次可以从当前点出发沿着一条有向边走到另外一个点。谁无法走了谁就输。
- 一些图游戏可以通过 Sprague-Grundy 函数来判定先手的胜负情况（简称 SG 函数）。
- **SG 函数** 一个图 G=(V,E) 的 SG 函数 g，是定义在 v 上的一个非负整数函数：

  $g(x)=\min\{n>=0 \mid n\neq g(y) \text{ for } <x,y>\in E\}$

  如果 x 的出度为 0，那么 g(x)=0。（边界条件）

SG 函数的内涵

- g(x) 就是 x 的后继点的 SG 值中没有出现过的最小值。
- 这样定义有什么好处呢？我们把一个**图的当前状态值**定义为游戏者处在的这个点的 SG 值。
- 如果游戏者处在一个点 x， g(x)≠0 。那么 0, 1, …, g(x)-1 这些数必然都出现在 x 的后继节点的 SG 值中，而游戏者可以走到这些点中的任意一个。也就是说：游戏者可以通过一步走棋把**图的当前状态值**任意的减小（当然必须保证状态值始终 >=0 ）。
- 如果游戏者处在一个点 x， g(x)=0 。那么**游戏者无论如何移动，下一个点的 SG 值都不等于 0** 。

SG 函数性质

- 对于一个图游戏，如果**图的当前状态**等于 0，那么先手必败，否则必胜。
- 证明：
- 如果当前点 SG=0，先手无论怎么走，都会到达一个 SG<>0 的点；接着后手就能设法到达一个 SG=0 的点。也就是说后手总是能移动，而先手总是处在 SG=0 的点。游戏不能无限的进行下去，一旦先手到达一个出度等于 0 的点，游戏结束，先手败。
- 如果当前点 SG≠0，先手可以走到一个 SG=0 的点，这样后手面对一个必败状态，所以先手必胜。

SG 函数在多图游戏中的应用

- **多图游戏** 有多个图，每个图都有一个当前节点。两个游戏者轮流行动。每个人每次可以把**某一个**图中的当前节点沿着该点连出的有向边移动到另一个点。无法移动的那个人输。

结论：

设这些图的当前状态值分别是 a1, a2, …, ak ，如果： a1 ⊕ a2 ⊕ … ⊕ ak = 0 ，那么先手必败，否则必胜。

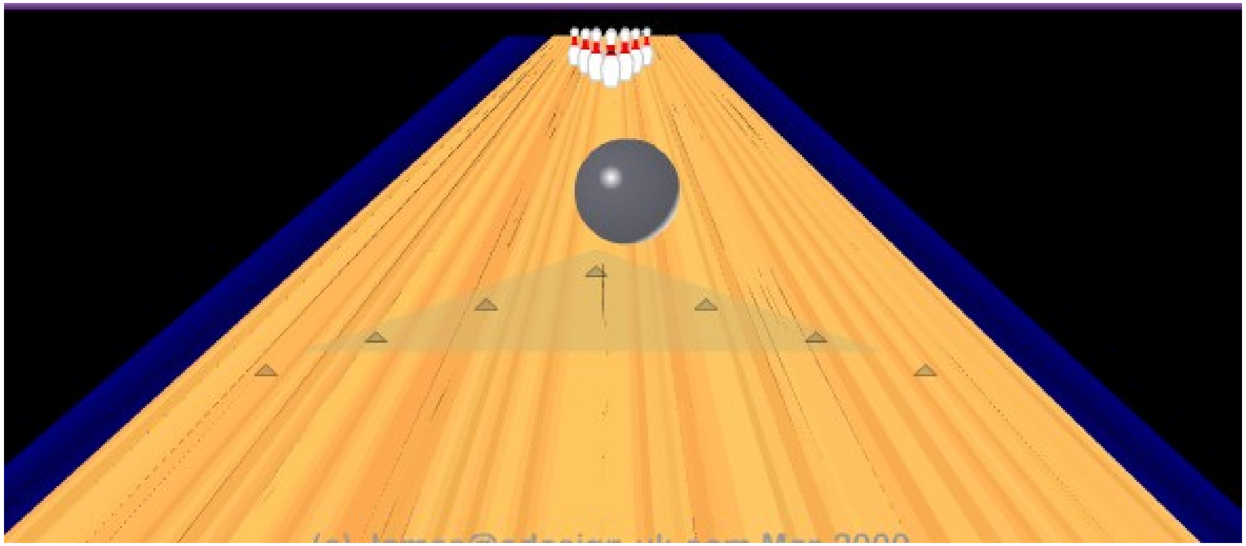
证明：

请回忆 SG 函数的性质：如果当前某图的状态值 >0 ，那么游戏者可以通过一步走棋把**图的当前状态值**任意的减小。因此一个状态值为 x 的图等价于 Nim Game 中规模为 x 的一堆石子！

- 如果 a1 ⊕ a2 ⊕ … ⊕ ak = 0 ，先手无论怎么走都会令 a1’ ⊕ a2’ ⊕ … ⊕ ak’ ≠ 0 。（ ai’ 是 ai 变化后的值）
- 如果 a1 ⊕ a2 ⊕ … ⊕ ak ≠ 0 ，那么就完全等价于 Nim Game ，先手可以按照 Nim Game 的走法行动，使得 a1’ ⊕ a2’ ⊕ … ⊕ ak’ = 0 。
- 证毕。
- **SG 的妙处就是把多图游戏转化成了 Nim Game 。**

保龄球问题

- 在一行中有 n 个木瓶，你和你的朋友轮流用保龄球去打这些木瓶，由于你们都是高手，每一次都可以准确的击倒一个或相邻的两个木瓶，谁击倒最后一个剩余的木瓶谁将获得胜利。如果由你先打，请你分析，你应该采取什么策略来确保赢得胜利。



分析

为了更方便的看清楚问题的本质，我们用另一种方式来描述这个游戏。

- 最开始有一堆石子（ n 个），每一次你可以进行以下四种操作中的一种：
- 从中取出一颗石子
- 从中取出两颗石子
- 从一堆中取出一颗石子，并且将这一堆中余下的石子任意分成两堆（每堆至少一颗）
- 从一堆中取出两颗石子，并且将这一堆中余下的石子任意分成两堆（每堆至少一颗）

这四种操作，实际上就依次对应于原来游戏中的以下四种击倒法：

- 击倒一段连续的木瓶中最靠边的一个
- 击倒一段连续的木瓶中最靠边的连续两个
- 击倒一段连续的木瓶中不靠边的一个
- 击倒一段连续的木瓶中不靠边的连续两个

求解

- 把局面看作顶点，游戏规则看作边，这是一个典型的图游戏。
- 如果当前局面被分成了 M 堆，（A1， A2， …， Am），则 $SG(A1,A2,\ldots,Am)=SG(A1)\oplus SG(A2)\oplus\ldots\oplus SG(Am)$

显然， SG(0)=0

- 剩余一个时，只能取到 0 个，而 SG(0)=0， 所以 SG(1)=1 。
- 剩余两个时，可以取到 0 或 1， 其中 SG(0)=0， SG(1)=1， 所以 SG(2)=2
- 剩余三个时，我们可以把局面变成 1 或 2 或两堆均为 1 。
- 其中 SG(1)=1， SG(2)=2， SG(1， 1)=SG(1) ⊕ SG(1)=0， 所以 SG(3)=3
- SG(4) 可分解为 {SG(2), SG(3),SG(2)⊕SG(1), SG(1)⊕SG(1)} ={2,3,3,0}, 所以 SG(4)=1
- 对于任意一种局面 P， 的 SG 值为 SG(P)， 为了赢得胜利，我们只需将他的局面变成 Q, 使得 SG(Q)=0 即可。

优化

- 对于每一个 N 值，我们为了求出他的 SG 值，时间复杂度为 O(N)，如果只要求你求出你第一步应该如何行动，那么这种普通的方法需要 $O(N^2)$ 的复杂度，显然不能令我们满意。
- 这个问题看似已经解决，但我们可以进行优化。
- 事实上，我们通过观察较小的数的 SG 值，可以发现：

  0 ~11 的 SG 值为： 0 1 2 3 1 4 3 2 1 4 2 6

  12~23 的 SG 值为： 4 1 2 7 1 4 3 2 1 4 6 7

  24~35 的 SG 值为： 4 1 2 8 5 4 7 2 1 8 6 7

  36~47 的 SG 值为： 4 1 2 3 1 4 7 2 1 8 2 7

  48~59 的 SG 值为： 4 1 2 8 1 4 7 2 1 4 2 7

  60~71 的 SG 值为： 4 1 2 8 1 4 7 2 1 8 6 7

  72~83 的 SG 值为： 4 1 2 8 1 4 7 2 1 8 2 7

  并且从 72 开始， SG 值以 12 为循环节，不断的重复出现，这样我们求出所有 SG 值的复杂度就降到了常数，这样判断第一步的如何选择的复杂度就降为了 O(N) 。

Strips(poi2000)

- Stripes is a two player game. Necessary requisites are a board and rectangular stripes in three colours: red, green and blue. All the red stripes have dimensions c x 1, green - z x 1, and blue - n x 1, where c, z and n are integers. Players have at their disposal an unlimited pool of stripes of each colour.
- A game board is a rectangle of dimensions p x 1 and consists of p fields of size 1 x 1. Players make their moves by turns. Move consists of laying a stripe of any colour on the board. There are the following rules in force:
- A stripe cannot stick out of the board,
- The covering (even partially) the earlier laid stripes is forbidden.
- The ends of a stripe have to adhere to the edges of the fields on the board. The player, who is not able to perform his move in accordance to the game rules first, loses.
- The first player is this one, who makes the first move in the game. It is said, that the first player has a winning strategy, if independently of the moves of the second player he can always win.
- Task
- Write a program, which:
- reads sizes of stripes and of at least one board from the text file PAS.IN for each board determines, whether the first player has a winning strategy, writes the results to the text file PAS.OUT.

Input

The first line of the input file PAS.IN consists of three integers c, z and n, 1 <= c, z, n <= 1000, equal to the lengths of stripes, adequately: red, green and blue ones. Numbers in the line are separated by single spaces.

The second line of the file PAS.IN consists of one number m, 1 <= m <= 1000, which is equal to the number of different boards to consider. Lines from the 3-rd to the (m+2)-th consists of one number p, 1 <= p < 1000. Number in the (i+2)-th line is the length of the i-th board.

Output

The output file PAS.OUT should contain m lines. Only one number should be written in the i-th line of the file:

1 - if the first player has a winning strategy on the i-th board

2 - otherwise.

Example

For the input file PAS.IN:

1 5 1

3

1

5

6

the correct result is the output file PAS.OUT

1

1

2

异或乘

- 问题引入
- 在一个 n*n 的方格阵中，每个格子里放一枚硬币。有的正面朝上（ H ），有的反面朝上 (T) 。两个人做游戏，轮流翻硬币。规则是这样的：选一个跟方格阵平行的矩形，将它的四个角上的硬币翻转，并且要求矩形的右下角必须从正面翻到反面（这个限制条件的目的是为了让游戏最终能够顺利结束并且严格分出胜负）。不能操作的人就输了。
- 判断某一种状态是先手必胜还是后手必胜。

基本理论

1. 异或加⊕（即 pascal 的⊕运算，又称二进制不进位加法）

   若 x 可以分解成独立的若干个状态 x1,x2,…,xk，则

   $g(x)=g(x1)\oplus g(x2)\oplus \ldots \oplus g(xk)$

2. 异或乘⊗

   它是一种二元运算，运算法则如下： 它满足交换律、结合律、分配律

   $x\otimes y=y\otimes x$

   $x\otimes (y\otimes z)=(x\otimes y)\otimes z$

   $x\otimes (y\oplus z)=x\otimes y\oplus x\otimes z$

   $0\otimes x=0$

   $1\otimes x=x$

   对于 2^{2^n} 的数 x （ n>=0 ）和一个任意的数 y,

   $x\otimes y=x*y$

   $x\otimes x=3*x/2$

原问题分析

- 对于判断翻硬币游戏的胜负情况，关键就是计算给定状态的 g 函数值。
- 对于每一个翻硬币游戏状态，我们都可以分解成若干个子状态的加和，例如：（T 表示反面朝上， H 表示正面朝上）

  $g(HTTHH)=g(H) \oplus g(TTTH) \oplus g(TTTTH)$
- 我们先来考虑原问题的一维情况：

  n 个硬币排成一行，每次可以取一个正面朝上的硬币，和它左边的任一枚硬币，将它们翻转。两个人轮流操作，不能操作者输。

  由于每一个状态可以分解成只有一个反面朝上的子状态，则我们只要考虑这种情况的 g() 函数。则我们设 H 左边有 n 个 T 的状态

  $$g(\underbrace{T\ldots T}_{n\text{个}}H)=g'(n)$$

  根据 g(x) 的定义，我们可以得出： $g'(n)=n$
- 回到二维情况 :

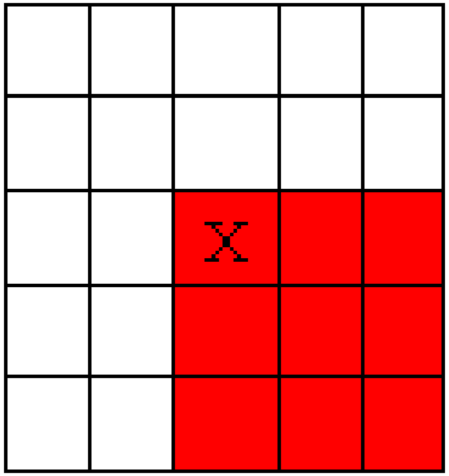
  根据游戏论理论，对于 $g'(x,y)$ 表示在 (x,y) 格中有唯一 H 的状态的 g() 函数值（坐标从 0 开始）

  则 $g'(x,y)=g'(x)\otimes g'(y)$

  这样，我们只需要计算每一个二维状态的正面朝上的所有位置的 $g'(x,y)$ 的值，再用异或操作加和起来，就得到了给定状态的 g 函数值。原问题得到解决

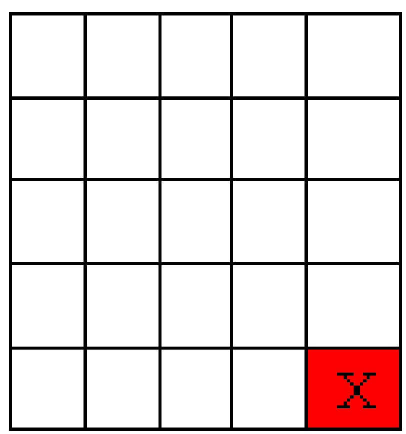
棋盘游戏

- 一个 n*m 的棋盘，两个人轮流走。每次可以选一个格子，把这个格子及其右下方的所有格子全部拿走。
- 比如右图 5*5 的棋盘，某一个人选 X，就可以把红色的格子全部拿走。
- 左上角的格子 (1,1) 是不能选的。
- 如果某个游戏者不能走了就输。
- 问：对于一个 n*m 的棋盘先手有没有必胜策略？



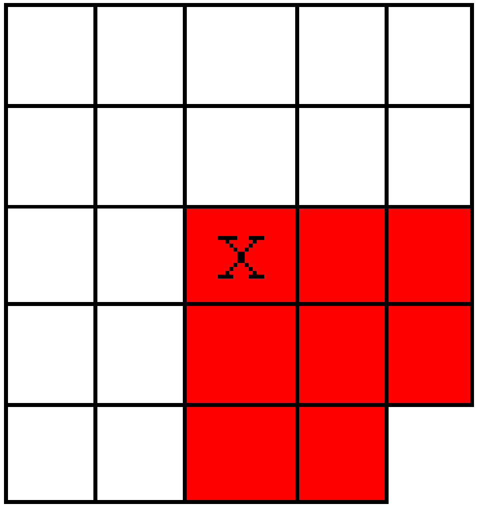
分析

- 首先明确一点：如果把一个状态看作一个点，给可以到达的状态之间连有向边，那么问题就转成了：一个有向图，从指定的点开始，游戏参与双方轮流沿着边走，不能走的输。这是个有向无环图，所以每个状态不是先手必胜就是先手必败。
- 先手可以这么走：（下面这个状态称为 A 状态）



分析

- 如果 A 是一个先手必败状态，那么原棋盘就是一个先手必胜状态。
- 否则如果 A 是一个先手必胜状态，那么必然可以通过拿掉某个格子变成先手必败状态。不妨设这个状态是 B：
- 我们发现，在一开始游戏的时候，先手可以直接达到 B 状态；而 B 是必败状态，所以原棋盘还是必胜状态。
- 也就是说无论如何， n*m 的棋盘都是先手必胜状态。特别之处是，我们无法给出最优策略是什么。



Green game (POI2001)

- "Green game" is a game for two players, say Ann and Billy. Their task is to shift a pawn on the board.†Some fields of the board are green, the rest is white. All of them are numbered by integers from the interval 1...(*a*+*b*). The fields with integers from the interval 1...*a* belong to Ann, the fields with numbers (*a*+1)...(*a*+*b*) to Billy.†
- For each field there is given *a set of successors*, containing the fields one can get to from this field in one move. These sets were chosen in such a way that from the field belonging to Ann one can get in one move to the Billy's fields only, and vice versa. All the fields have non-empty sets of successors, thus one can always make a move.
- At the beginning of the game we put a pawn on the arbitrarily chosen *start field **P***, then players shift the pawn by turns from their field to any successor of this field (we know it belongs to the opponent). The game is started by an owner of the start field **P**. The game is finished when the pawn stays for the second time on the same field, say the field **Q**. If in the sequence of moves from the field **Q** to the field **Q** taken for the second time, the pawn was put at least once on the green field, Ann wins the game, otherwise Billy wins. We say that Ann has *a winning strategy for the given start field **P*** in case when there is such a method, which guarantees that she wins the game beginning from this field, no matter what moves Billy makes.

- Write a program which:
- reads from the text file GRA.IN the description of the board, computes the set of fields for which Ann has a winning strategy, writes the result in the text file GRA.OUT.†
- Input
- In the first line of the text file GRA.IN there are written two positive integers *a*, *b*, separated by a single space, meaning respectively: the number of fields belonging to Ann, the number of fields belonging to Billy. Integers *a*, *b* satisfy the condition: 1 <= *a*+*b* <= 3000. In the following *a*+*b* lines there are descriptions of the fields of the board: first, descriptions of fields belonging to Ann, and then, of ones belonging to Billy. The (*i*+1)-st line, for 1 <= *i* <= *a*+*b*, begins with integers *z*, *k* meaning respectively the colour of the field *i* (0 means white, 1 - green) and the number of successors of this field. Then *k* integers (1 <= *k* < *a*+*b*) are written (still in the same line). They are the numbers denoting the successors of the *i*-th field. The integers in each line are separated by single spaces. The number of green fields on the board is not greater than 100. The total number of successors of all the fields on the board is not grater than 30000.†
- Output
- The first line of the text file GRA.OUT should contain exactly one integer *l*, which indicates the number of fields for which Ann has a winning strategy. The following *l* lines should contain numbers of these fields written in ascending order - each integer should be written in a separate line.

分析

- 在整个的游戏过程当中， Ann 和 Billy 都会设法让自己赢得游戏。首先我们必须明确一点：从某一区域出发，如果 Ann 没有必胜策略，那么 Billy 显然存在着必胜策略，反之亦然。根据这一特点，设置一个布尔函数 y=f(i)，当 f(i)=true 时表示从编号为的区域开始游戏 Ann 存在着赢得游戏的必胜策略，当 f(i)=false 时则表示 Billy 存在着赢得游戏的必胜策略。
- 当士兵处于属于 Ann 的区域的时候，如果这个区域的某一个后继区域能够是 Ann 必胜的话， Ann 肯定会让士兵走向那一个区域，同样当士兵处于属于 Billy 的区域的时候 Billy 也会采用相同的策略。那么，可以得到这样一个递推公式：

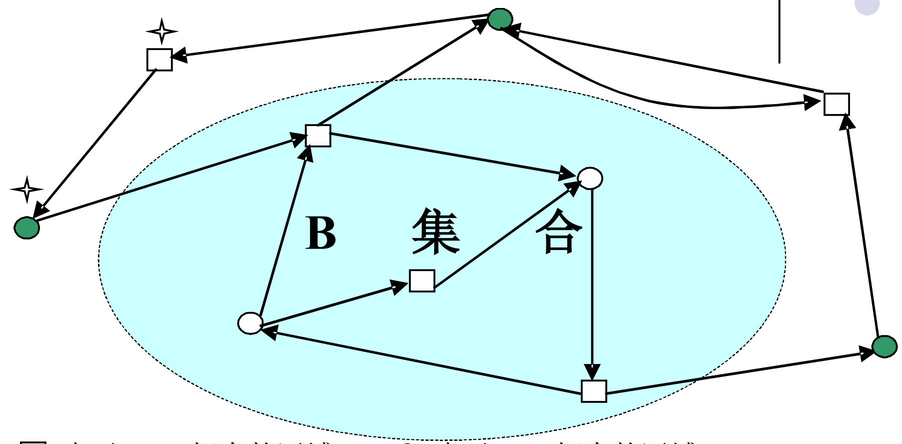
$$f(i)=\begin{cases} f(x_1) \quad and \quad f(x_2) \quad and \quad \ldots \quad f(x_m) & (i>a) \\ f(x_1) \quad or \quad f(x_2) \quad or \quad \ldots \quad f(x_m) & (i\le a) \end{cases}$$

构图

- 为了便于思考，将问题简化一下：把 a+b 个区域看成 a+b 个节点，节点编号与区域编号对应，如果编号为 *i* 的区域是编号为 *j* 的区域的后继区域，就从编号为 *i* 的节点向编号为 *j* 的节点连一条有相弧 i→j，这样便构成了一个由 a+b 个节点构成的有向图。
- 然而在这个有向图中存在着许许多多的环，上面的这一递推式显然存在后效性，是不能够求解的。但是如果某一些区域的 *f* 值能够直接求得的话，其它区域的 *f* 值就有可能求出来。

求 *f* 值

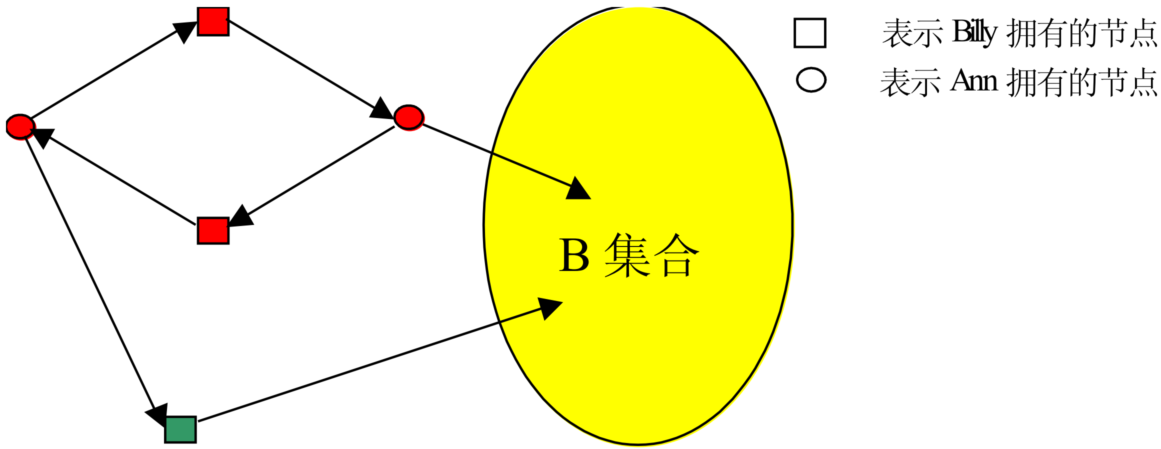
- 定义　B 集合是一个点集，该集合当中不存在绿色节点，所有的节点构成一个强连通分量，若编号为 *i* 的节点属于该集合，当时，该节点的所有后继节点都属于该集合，当时，该节点至少有一个后继节点属于该集合。



□ 表示 Billy 拥有的区域　　○ 表示 Ann 拥有的区域

- B 集合有一个非常重要的性质：一旦士兵走入了某一个 B 集合， **Billy** 总能够使士兵始终处于该集合当中，而 **Ann** 无法使士兵走出该集合。由于 B 集合当中节点个数是有限的，所以士兵经过的路线上必定会出现一个没有绿色节点的环。因此只要士兵走进某一个 B 集合当中，那么 **Billy** 必然会赢得比赛的胜利，故所有 B 集合当中的节点的 *f* 值都为 ***False*** 。
- 接下来，根据上面的递推式能够确定另外一些节点（当且仅当该节点的所有后继节点的 *f* 值已经确定了）的 *f* 值，这些节点的 *f* 值都只可能为 ***False*** 。例如上图那一个例子，图中标有星号的那些节点可以在确定 B 集合之后确定它们的 *f* 值。

进一步分析



- 实际上从上图涂有红色的那些节点出发仍然是 **Billy** 获得比赛的胜利，因此这些节点的 ***f*** 值也应当为 ***False***。可按照上面的步骤 ***f*** 值是不能够确定的，怎样处理这种情况呢？
- 确定 **B** 集合的目的是为了使士兵走入这个点集后 **Ann** 不能让他走不出来，从而使 **Billy** 获得游戏的胜利，如果 **Ann** 可以使士兵走出来，但不过是使士兵走到了一个使 **Billy** 必胜的节点，这样在本质上与士兵仍处于 **B** 集合是一样的。
- 因此可以把所有已经确定了 ***f*** 值的节点删除，那么被涂成红色的节点就成为了一个新的 **B** 集合。

算法框架

while 能够从图中找到 B 集合 do

{

寻找所有的 B 集合并将所有 B 集合当中的节点的 *f* 值置为 *False*；

寻找所有可以确定 *f* 值的节点，并将它们的置值为 *False*；

将已确定 *f* 值的节点从图中删除；

}

将剩余的节点的 *f* 置值为 *True*.

- 下面简单的论证一下该算法的正确性。
- 因为剩余节点当中已经不存在 B 集合，那么 Billy 完全无法控制士兵能够到达的节点的范围，那么 Ann 是一定能够在游戏结束之前使士兵到达一个特定的强连通分量中的某一节点，这个连通分量中任何一个节点的后继节点都处于该连通分量中，因为不存在 B 集合，那么这一连通分量中必然存在绿色节点，因此 Ann 可以使士兵在这一个连通分量中走出一个含有绿色节点的环。

性能分析

- 时间复杂度：该算法中寻找 B 集合和寻找可以确定 *f* 值的节点的过程的时间复杂度为， *m* 表示图中有向弧的条数。因此算法的时间复杂度为， *k* 相当于算法框架当中 while 循环的次数， *k* 的取值与有向图的构型有关， *k* 的上限为 (a+b)/2 。
- 空间需求：$(5(a+b)+8m)Bytes < 300KB$

参考资料

《人工智能导论》
清华大学出版社
林尧瑞 马少平

《博弈论》
中国人民大学出版社

朱·弗登博格 （法）
让·梯若尔

《博弈与信息：博弈论概论（第二版）》
Games and Information:An Introduction to Game Theory
北京大学出版社
[美] 艾里克.拉斯缪森
王晖 白金辉 吴任昊