信息学中的数论（Number theory）

陈剑秋

calois@163.com

三元楼225室

整数的表示法

我们在日常生活中采用十进制表示整数，用一些数字表示10的方幂来表示整数。例如一个整数是83526，它表示的是：

$$8\cdot10^4+3\cdot10^3+5\cdot10^2+2\cdot10^1+6\cdot10^0$$

十进制是计数制的一个例子，每个数字的位置决定了它所代表的数值。例如上面的8，它表示的是$8\cdot10^4$。

随着计算机的发展，十以外的进位制变得越来越重要，尤其以2、8、16为基底的进位制。

定理1.1：令b是正整数，b>1，则每个正整数n都可以唯一地写为如下形式：

$$n=a_kb^k+a_{k-1}b^{k-1}+ \dots +a_1b^1+a_0b^0$$

其中k为非负整数，a_j为整数，$0 \le a_j \le b-1$ $(j=0,1,\dots,k)$，且首项系数$a_k \neq 0$。

若 b=2，则$n=a_k2^k+a_{k-1}2^{k-1}+ \dots +a_12^1+a_02^0$，其中每个$a_j$或者为0或者为1，就是二进制表示。B被称为展开式的**基**（base）或**根**（radix）。基为10的是十进制（decimal），基为2的是二进制（binary），基为8的是八进制（octal），基为16的是十六进制（hexadecimal）。

二进制数转化为十进制数

按权展开求和

$101011.1101_{(2)}=\underline{\ 43.8125\ }_{(10)}$

整数部分

$101011_{(2)}=1\times2^5+0\times2^4+1\times2^3+0\times2^2+1\times2^1+1\times2^0=43_{(10)}$

小数部分

$0.1101_{(2)}=1\times2^{-1}+1\times2^{-2}+0\times2^{-3}+1\times2^{-4}=0.8125_{(10)}$

十进制数转化为二进制数

整数部分：除二取余法

除数	被除数	余数	
2	52	0	低位
2	26	0	
2	13	1	
2	6	0	
2	3	1	
2	1	1	高位
	0		

$$52_{(10)}=110100_{(2)}$$

十进制数转化为二进制数

小数部分：乘二取整法

$0.65 \times 2 = 1.3$	1	高位
$0.3 \times 2 = 0.6$	0	
$0.6 \times 2 = 1.2$	1	
$0.2 \times 2 = 0.4$	0	低位

$0.65_{(10)} = 0.1010_{(2)}$ 取近似值

十进制整数转化为二进制数的代码

转换的方法就是如前所示，除以2取余数，然后把余数反过来取就可以了。

我们可以把每次得到的余数存到一个数组a[100]里，不妨从a[1]开始存放，最后存完了，你可能没存满100个位置，那些你没存的地方还是0，要把这些0先去掉，也就是要从第一个1开始输出。

```
int i,n,a[100]={0};
scanf("%d",&n);
i=1;
while (n!=0)
{    a[i]=n%2;
     n=n/2;
     i++;
     //如果上面的2都改成10, 程序做什么？
}
for(i--;i!=0;i--)
   printf("%d",a[i]);
```

二进制整数转化为十制数的代码

秦九韶算法

计算下式：

$$F(x)=a_nx^n+a_{n-1}x^{n-1}+a_{n-2}x^{n-2}+\ \dots\ +a_1x+a_0$$

如果计算每项的值再相加需要(n(n+1))/2次乘法和n次加法，而秦九韶算法只需要n次乘法和n次加法，大大简化了运算过程。

$$F(x)=a_nx^n+a_{n-1}x^{n-1}+\ \dots\ +a_1x+a_0$$
$$= (a_nx^{n-1}+a_{n-1}x^{n-2}+\ \dots\ + a_2x +a_1)x+a_0$$
$$= ((a_nx^{n-2}+a_{n-1}x^{n-3}+\ \dots\ + a_3x +a_2)x+a_1)x+a_0$$
$$\dots$$
$$=((\ \dots\ (a_nx+a_{n-1})x+a_{n-2})x+\ \dots\ +a_1)x+a_0$$

求多项式的值时，首先计算最内层括号内一次多项式的值

$S_0=a_n$

$S_1=S_0x+a_{n-1}$ 由内向外逐层计算一次多项式的值

$S_2=S_1x+a_{n-2}$

…

$S_n=S_{n-1}x+a_0$

如果依次输入一个整数二进制的各位数字（从高位到低位，以-1结束），如何把这个数的十进制求出来？

```
int s=0,i;
scanf("%d",&i);
while (i!=-1)
{
    s=s*2+i;
    scanf("%d",&i);
}
printf("%d\n",s);
```

结合上面的内容，如何写一个有点技术含量的将一个数逆序输出的程序，例如：输入1234567，输出7654321。

```
long long int reverse=0,n;
scanf("%lld",&n);
while (n!=0)
{
    reverse=reverse*10+n%10;
    n=n/10;
}
printf("%lld\n”,reverse);
```

➢ 砝码称重问题

现有一个天平，需要配n个砝码，使得能称出1,2,3, … 尽可能多的连续重量，这n个砝码应该如何设计？

归纳下简单情况：

第i砝码重量　　能称出的连续总重量

$W_1=1$	$S_1=1$
$W_2=3$	$S_2=4$
$W_3=9$	$S_3=13$
$W_4=27$	$S_4=40$

…

假设现在有i-1个砝码，能称出1到S_{i-1}的物品，再添加一个新砝码的重量W_i，为了能称出更多的连续重量，应该满足：

$W_i-S_{i-1}=S_{i-1}+1$，即$W_i=2S_{i-1}+1$

$W_i=2(S_{i-1})+1=2(W_1+W_2+\ \dots\ +W_{i-1})+1 \qquad (1)$

$W_{i-1}=2(W_1+W_2+\ \dots\ +W_{i-2})+1 \qquad (2)$

(1)-(2)得： $W_i-W_{i-1}=2W_{i-1}$

$\therefore\ W_i=3W_{i-1}$

$\because\ W_1=1$

$\therefore$这n个砝码重量依次是$3^0,3^1,3^2,\ \dots\ ,3^{n-1}$。

能连续称出的重量是1至$(3^n-1)/2$。

上述只是给出了一种可能的解决方案，但是并没有严格证明这种方案就是问题的解。有一些逻辑上的漏洞，比如：

疑问一：

$W_1=1$ $S_1=1$

$W_2=3$ $S_2=4$

$W_3=9$ $S_3=13$

重量5是否必须是通过第3个砝码称出来，会不会通过后面砝码4和砝码5称出来？

疑问二：

会不会最后砝码组合能称出的重量是这样的：1,2,3, … ,m,m+j,m+j+1, … 其中（j＞1）

下面就严格证明：这n个砝码重量依次是 $3^0,3^1,3^2,\ldots,3^{n-1}$时，才能最多称出连续重量，从1至$(3^n-1)/2$。

要证明上述命题就是证两点：

1、**能**

即1至$(3^n-1)/2$的每个重量都能用 $3^0,3^1,3^2,\ldots,3^{n-1}$这组砝码称出来。

2、**最多**

n个砝码不可能称从1开始连续的超过$(3^n-1)/2$的重量（最多只能称到$(3^n-1)/2$）。

- 先来证明第1点：1至$(3^n-1)/2$的每个重量都能用$3^0,3^1,3^2,\ldots,3^{n-1}$这组砝码称出来。

问题描述：

现有重量为 $3^0,3^1,3^2,\ldots,3^n,\ldots$ 的砝码各一个，给定一个重量为正整数m的物品，怎么把这个物品的重量称出来？？

样例输入：70

样例输出：70=81-9-3+1

表示物品和3、9放左面，81和1放右面。

很明显，现在的问题是**如何将m表示成3的不同幂之和（差）**。

我们马上可以联想到三进制，但是又有不同的地方：三进制表示中每项前面的系数是0、1、2，现在问题前面的系数是-1、0、1。

关键是系数是2的怎么办？

$2 \times 3^k = (3-1) \times 3^k = 3^{k+1} - 3^k$

这样我们把m转换为三进制表示$(A_kA_{k-1}\ldots A_1A_0)_3$，从右向左依次处理，如果$A_i=0$或1则不变；如果$A_i=2$，则将$A_i$置为-1，进位符C记为1；如果$A_i=3$，则将$A_i$置为0，进位符C记为1。下次处理$A_{i+1}$，先让$A_{i+1}=A_{i+1}+C$，得到新的$A_{i+1}$后用前面的方法继续处理；

很明显上述操作能在有限步能完成，而且最后的A_i=-1、0、1。

现在还要证明一点：不会由于进位导致$A_{n+1}=1$(就是动用第n+1个砝码)

由于m最大是$(3^n-1)/2$，用三进制表示就是11…1（n个1）。

比m小的任意整数的三进制表示从左向右必须前面若干个连续1，然后出现第一个0，那么按照前面的操作，最多导致这个0变成1，不会对再左面的数产生影响，也就不会改变A_{n+1}。

- 接下来证明第2点：

  n个砝码不可能称从1开始连续的超过$(3^n-1)/2$的重量（最多只能称到$(3^n-1)/2$）。

反证法：

假设n个砝码能从1连续称到x($x>(3^n-1)/2$)。

那么每种称法把砝码都左右交换下，也能对应的称出-x到-1（从数学角度可以认为物品重量为负）。再加两边都不放砝码这种情况（这个情况左后交换是一种情况）。则n个砝码能称出的总情况至少是：

$2x+1>2((3^n-1)/2)+1=3^n$。 (1)

而每个砝码有放左边、不放、放右边三种情况，根据乘法原理总共有3^n种情况。 (2)

(1)和(2)矛盾了，说明假设不成立。

上述用到了一种重要的计数思想：算两次。

算两次：通过两种方法来计算某个值，两者相等，建立方程，把其中你想知道的值求出来。（感兴趣的可以参阅中国科学技术大学出版社出版的单墫《算两次》一书）

上面我通过两种途径来计算总的情况数：

- ✓ 总共能称出多少个重量，每个重量对应一种砝码放置情况。
- ✓ 从组合学角度共有多少种砝码放置情况。

■素数（Prime number）

素数是大于1的正整数，并且除了1和本身不能被其它正整数整除。大于1的不是素数的正整数称为**合数**。

良序性质（The Well-Ordering Property）：每个非空的正整数集合都有一个最小元。

定理2.1：每个大于1的整数都有一个素因子。

证明：反证法，假设大于1的整数没有素因子，那么大于1且没有素因子的正整数构成的集合非空，由良序性知道集合存在一个最小的整数n。由于n能被n整除且n没有素因子，那么n不是素数（否则n就有素因子n）。于是n可以写成n=ab，其中1<a<n，1<b<n。因为a<n，(a不属于集合) 所以a一定有素因子（没素因子里n是最小的）。a的因子也是n的因子，那么n就有素因子，这和假设矛盾。

定理2.2：存在无穷多个素数。

该定理是数论中关键性定理之一，它的证明方法有很多种。下面给出的方法是欧几里得（Euclid）在他的《几何原本（Euclid's Elements）》一书中给出的。这个简单而又优美的证明的方法被认为相当的完美，被收录在《数学天书中的证明（Proofs from THE BOOK）》中。

http://item.jd.com/11890076.html

证明：假设只有有限多个素数为$p_1,p_2,\cdots,p_n$，其中n是正整数（假设上面列出了所有的素数）。考虑整数Q_n，由这些素数的乘积加1得到，即：$Q_n=p_1p_2\ldots p_n+1$

由定理2知道，Q_n至少有一个素因子，设为q。我们将证明q不是上述素数中的任何一个，从而得到矛盾。

如果$q=p_j$，$1\le j\le n$，由于$Q_n-p_1p_2\ldots p_n=1$，且q可以整除上面等式左端两项，那么q也能整除1，这是不可能的（1不能被任何素数整除），所以q不是p_j的任何一个，和假设矛盾。

"q不属于{pn}"与"q为素数"矛盾

将素数和合数加以区分是至关重要的，称之为**素性检验**。 最基本的素性检验是**试除法**，一个整数n是素数当且仅当它不能被任何一个小于$\sqrt{n}$ 的素数整除。

定理2.3：如果n是一个合数，那么n一定有一个不超过$\sqrt{n}$素因子。

证明：既然n是合数，那么n可以写成n=ab，不妨设1<a≤b<n。我们一定有a≤$\sqrt{n}$，否则b≥a>$\sqrt{n}$，那么ab>$\sqrt{n}\cdot\sqrt{n}$=n。根据定理2.1，a至少有一个素因子，也是n的因子，显然这个素因子小于等于$\sqrt{n}$。

素性检验的代码：

```
bool is_prime(int n)    //检查n是否素数
{
    for(int i=2;i*i<=n;i++)
        if (n % i==0) return false;
    return n!=1; 不要写麻烦了
    //除了1没有因子就是素数

}
```

求n所有因子的代码：

```
vector<int> division(int n)
{  vector<int> res; //n所有因子保存在res里
   for(int i=1;i*i<=n;i++)
     if (n % i==0)
     {  res.push_back(i);
        if (i!=n/i) res.push_back(n/i);
        //对于非完全平方数，插入因子i时，同时
 插入因子n/i    }
    return res;
}
```

埃拉托色尼斯（Eratosthenes）筛法

根据定理4，可以找到所有小于等于n的素数，该方法是由古希腊数学家埃拉托色尼斯提出的，所以叫**埃拉托色尼斯筛法**。下图说明找出小于等于100的所有素数。首先小于100的合数一定有一个小于$\sqrt{100}$=10的素因子，只能是2,3,5,7。先删除所有能被2整除的数（红色），再删除所有能被3整除的数（桔黄色），再删除所有能被5整除的数（蓝色），最后删除所有能被7整除的数（绿色），剩下都是素数（除了1）。

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

埃氏筛法的代码：

```
int prime[MAX_N];   //prime依次存放得到的素数
bool is_prime[MAXN+1];  //is_prime[i]为true 表示i是素数
int solve(int n)   //求n以内所有的素数，返回素数个数
{ int p=0;       //p记录当前找到的素数的个数
   for(int i=0;i<=n;i++) is_prime[i]=true;
   is_prime[0]=is_prime[1]=false;
   for(int i=2;i<n;i++)
     if (is_prime[i])  //每次扫第一个是素数，再删掉倍数
     { prime[p++]=i;  //把找到的素数存到prime数组中
        for(int j=2*i;j<=n;j+=i) is_prime[j]=false; }
    return p;  }
```

□算数基本定理：

任何一个大于1的自然数 n，都可以**唯一**分解成有限个质数的乘积n=$p_1{}^{a_1}p_2{}^{a_2}\ldots p_s{}^{a_s}$，这里$P_1 < P_2 < P_3 < \ldots < P_s$均为质数，其中指数$a_i$是正整数。这样的分解称为n的**标准分解式**。

算数基本定理也称**质数幂分解**（prime-power factorization）定理。

➢NIM游戏一

现有三堆石头（每堆的数字随机），两个人轮流取石头，规则：

1、轮到你取，一定要取

2、可以选任意一堆取，最少取1个，最多把这堆都取完

3、不能跨堆取

谁取到最后一个石头就赢了。

分析：

本问题的必胜策略是：每次给对方剩下的三个数，转化为二进制，然后相加（并不是真正的二进制相加，就是统计每列有几个1），如果每次加的结果都是偶数，那么你就是必胜的。结论对方取完后，给你的数相加必有奇数，你一定可以再调整为都是偶数，一直进行下去，直到你获胜。

例如：开始是3,5,7

$$\begin{array}{r} 11 \\ 101 \\ +\ 111 \\ \hline 223 \end{array} \qquad \begin{array}{r} 10 \\ 101 \\ +\ 111 \\ \hline 222 \end{array} \qquad \begin{array}{r} 11 \\ 100 \\ +\ 111 \\ \hline 222 \end{array} \qquad \begin{array}{r} 11 \\ 101 \\ +\ 110 \\ \hline 222 \end{array}$$

把它调整成每列和都是偶数，只要和从左往右扫，扫到第一个奇数停下，对应这列上哪个是1，哪个就能调整，本例中三个都是1，说明三个数都能调整，调整很简单，

方式

就是把这数擦掉，然后一位位匹配，保证每列是偶数就行。所以本例给对方（2,5,7）或（3,4,7）或（3,5,6），你都是必胜的。

接下来证明：**你给对方的和都是偶数，对方取完后返回你的和中必有奇数。**

很简单，如果他返回你的也都是偶数，不妨设他取了第一个数字，那么第一个数字在取前和取后的奇偶性没有发生变化，二进制的特点，知道该位的奇偶性，就知道该位的值，说明这数在取前和取后每位的数字都没变，所以对方一定是取了0个石头，这是违反游戏规则的。他要返回你的和也都是偶数，还有种方法都是两堆里同时取，这也是违反规则的。

对方如果要赢，应该返回你（0,0,0），加起来都是偶数，而刚才证明了，对方返回你的必有奇数，说明他永远无法返回你（0,0,0），所以他永远赢不了。很明显，这游戏有限步必能结束，必有人赢，对方永远赢不了，当然永远是你赢啦。你只要每次再按上面的方法调整成都是偶数就行了。这里还要证明下：上面的调整方法不会导致新的数比原来的数大（这样变成加石头，不是取石头了），理由就是下面这个公式：

$$2^n=2^{n-1}+2^{n-2}+\ \ldots\ +2^1+2^0+1$$

NIM游戏二（Wythoff's Game）

现有两堆石头，两个人轮流取石头，规则如下：

（1）轮到取一定要取

（2）可以选任一堆取至少1个，最多该堆全部取完

（3）或者在两堆同取相同数量个石头。

谁取到最后一个石头谁赢。

分析：

本题必胜策略是留给对方如下形式的两数：

P	Q
1	2
3	5
4	7
6	10
8	13
9	15

…

上述序列表是这样生成的：开始是(1,2), Q和P的差是1，很明显给对方(1,2)，对方无论怎么取，下次都是你取光所有的数。接下来，P是自然数中在前面还没有出现过的最小的数，每行Q和P的差依次是2,3,4,5, ...。

这样的话，如果你给对方表中的一对数，对方无论怎么取都不会是表中的另一对数，差不同而他取完后，你又能立即调整为表中的一对数。只要你给他表中的数，他必定一次无法取完，所以无法赢，总有人赢，那一定是你赢。

下面先证明：**如果你给对方表中的一对数，对方无论怎么取都不会是表中的另一对数**。

由于P和Q两个序列是单调递增的，对方取完后，最多是前面的一对数一样。如果对方取的是Q，得到Q′，那么(P,Q′)在前面肯定没出现过，因为P是前面没出现过的最小的自然数。如果对方取的是P，得到P′，那么(P′,Q)在前面肯定没出现过，因为Q是已出现数中最大的。如果对方同时取P和Q，差没变，前面的数对差都比你小。

陈剑秋

接下来证明：**给对方表中的一对数，对方无论怎么取，你又能一步调整为表中前面的一对数。**

情况1：如果对方取的是P，得到P′，那么P′是前面已经出现过的（P是前面没出现过最小的），我可以把Q调整成表中前面出现过P′那对数中另一个（Q最大，肯定可行）。

情况2：如果对方两堆同时取，等同情况1。

情况3：如果对方取的是Q，得到Q′。分两种情况：**情况3a**：新数对的差是d2比原数对的差d1小（这里的差是绝对值），这样的话，根据表的生成知道，我可以在左右

各取相同的数，使得转化为表前差为d2的一对数。**情况3b**：新数对的差是d2比原数对的差d1大，这里也有两种情况：一种是取了P，例如：(8,13)取为(7,13)，这就是前面的情况1。另一种是取了Q，得到Q′，Q′比P小，前面肯定出现过，这里还分两种情况，一种是Q′在前面数对中右侧，例如：(8,13)取为(8,2)，这时你只要调整左边的数就行，也是情况1；另一种是Q′在前面数对中左侧，例如(8,13)取为(8,1)，即：原数对(P,Q)通过对方和自己两次调整后变成(P′,Q′)，**关键要证明P>Q′。**

∵ 原数对(P,Q)的差大于新数对(P′,Q′)的差（因为原数对排在后面），即：

$$Q-P > Q'-P' \qquad (1)$$

∵ 对方将原数对(P,Q)中的Q调整为P′后，差比原数对大，所以：

$$P-P' > Q-P \qquad (2)$$

由（1）和（2）联立，得：

$$P-P' > Q-P > Q'-P'$$

∴ $P-P' > Q'-P'$

∴ $P > Q'$ 保证我能通过调整P得到前面的数对

上面是构造法，逐行来生成序列表，对于P序列{1,3,4,6,8,9,11, … }，Q序列{2,5,7,10,13,15,18, …, }，你能看出它们的通项公式吗？即，指定任意一个n，求第n行上P和Q的值？

答案：

$$P_n=[\frac{\sqrt{5}+1}{2}\cdot n]$$

$$Q_n=[\frac{\sqrt{5}+3}{2}\cdot n]$$

贝蒂定理（Betti theorem）

设x是任何一个正的无理数，y是它的倒数，那么两个序列{[n(1+x)]，[n(1+y)]}合在一起恰好不重复地构成自然数集，记号[x]表示不超过x的最大整数。

形式二：（第20届普特南数学竞赛试题）

设a、b是正无理数且 1/a+1/b =1。记{ P=[na]，Q={ [nb] }，n 为任意的正整数，则P与Q是Z+的一个划分，即P∩Q为空集且P∪Q为正整数集合Z+。

先来证明上面两种形式等价:

设a=1+x,b=1+y, 则:

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

$$=\frac{1}{1+x}+\frac{1}{1+\frac{1}{x}}$$

$$=\frac{1}{1+x}+\frac{x}{1+x}=1$$

所以，上面两种形式等价。

现在证形式二：

因为a、b为正且1/a+1/b=1，则a,b>1

所以对于不同的整数n，[na]各不相同，类似对b有相同的结果。

因此任一个整数至多在集合P或Q中出现一次。

像这种“恰好构成”，“完美覆盖”等描述，关键是证明两点：无重无漏，即

P和Q没有重复的元素

P和Q没有漏掉的元素

现证明**P和Q没有重复的元素**，即$P\cap Q$为空集；

(反证法)假设k为$P\cap Q$的一个整数

则存在正整数m、n使得$[ma]=[nb]=k$

即$k < ma, nb < k+1$

等价地改写不等式为

$$m/(k+1) < 1/a < m/k$$

$$n/(k+1) < 1/b < n/k$$

相加起来得 $(m+n)/(k+1) < 1 < (m+n)/k$

即 $k < m+n < k+1$

这与m,n为整数有矛盾，所以$P\cap Q$为空集。

接下来证明**P和Q没有漏掉的元素**

(反证法)假设有一个元素k不在P和Q中

则存在正整数m、n使得

$$[ma] < k < [(m+1)a]$$

$$[nb] < k < [(n+1)b]$$

由此得 $ma < k \le [(m+1)a]-1 < (m+1)a - 1$

类似地有 $nb < k \le [(n+1)b]-1 < (n+1)b - 1$

等价地改写为 $m/k < 1/a < (m+1)/(k+1)$

$$n/k < 1/b < (n+1)/(k+1)$$

两式加起来， 得

$$(m+n)/k < 1 < (m+n+2)/(k+1)$$

即 $m+n < k < k+1 < m+n+2$

这与m, n, k皆为正整数矛盾。

再回想Wythoff's Game的构造法，满足P和Q恰好构成自然数序列，设$P_n=[n(1+x)]$,那么$Q_n=[n(1+x+1)]$（因为$Q_n-P_n=n$）。

根据刚才的贝蒂定理知道：x和(x+1)互为倒数，即$x=\frac{1}{x+1}$，∴$x^2+x-1=0$

解得：$x=\frac{-1\pm\sqrt{5}}{2}$，负根舍去。

$$\therefore P_n=[\frac{\sqrt{5}+1}{2}\cdot n]$$

$$Q_n=[\frac{\sqrt{5}+3}{2}\cdot n]$$

NIM游戏三（Fabonacci-NIM Game）

现有一堆石头，两个人轮流取石头，规则如下：

（1）轮到取一定要取

（2）至少取一个，最多取上一步对方取的石头数的两倍（如果是第一步，无任何限制）

谁取到最后一个石头谁赢。

分析：

本题的必胜策略是：把石头数n写成**不连续**的**斐波那契数之和**。斐波那契数列是：1,1,2,3,5,8,13,21,34,55,89, …, 每项是前两项之和。例如：$n=100=89+8+3$。取最后一个斐波那契数，对方取后一定不能取完，采用同样的策略继续，直到你胜利。

先证明一个引理

齐肯多夫(Zeckendorf)定理：任何正整数都可以表示成若干个不连续的斐波那契数（不包含第2个斐波那契数$F_2 = 1$）之和。

更精确的描述如下：

对于每一个正整数n，它可以唯一地表示为

$$n = F_{k_1} + F_{k_2} + \dots + F_{k_r}$$

其中F_i是斐波那契数，$F_1 = 1$，$F_2 = 1$，$F_3 = 2$，… 例如：$100 = F_{12} + F_6 + F_4$

$$k_i \geq k_{i+1} + 2 \qquad i = 1,2,3, \dots, r-1$$

数学归纳法证明：

n=1时，结论显然成立。

假设n<m时结论都成立，现在考虑n=m的情形。

取F_{k_1}为小于等于m的斐波那契数列中的最大值，则对m-F_{k_1}< m-1，由归纳假设，m-F_{k_1}有唯一的斐波那契数表示：$\widehat{F_{k_1}}+\widehat{F_{k_2}}+\ldots+\widehat{F_{k_s}}$

其中$\widehat{F_{k_i}}$是不连续的斐波那契数，而且F_{k_1}和$\widehat{F_{k_1}}$也不连续，否则能得到一个更大的斐波那契数，这和F_{k_1}最大矛盾。所以，命题得证。

否则m>Fk1+Fk2=Fk3
与假设矛盾

下面来证明前面的结论。

若$n = F_{k_1}+F_{k_2}+ \ldots +F_{k_r}$，则定义

$$\begin{cases} \mu(n)=F_{k_r} & n>0 \\ \mu(n)=\infty & n=0 \end{cases}$$

即μ(n)是齐肯多夫表示的最后一项。

（1）若n>0，则μ[n-μ(n)]>2μ(n)

∵ $\mu[n-\mu(n)]=F_{k_{r-1}} \geq F_{k_r+2} > 2F_{k_r}$，且$k_r \geq 2$

这式子意味着，你每次取最后一个斐波那契数，对方无法泡制，因为超出了2倍。例如：n=100=89+8+3，你取3后，对方无法取8。

（2）若$0<m<F_k$，则$\mu(m)\le 2(F_k-m)$

令$\mu(m)=F_j$

$\because m\le F_{k-1}+F_{k-3}+\ldots+F_{j+(k-1-j)\,mod\,2}$（根据齐肯多夫表示最大情况）

$=-F_{j-1+(k-1-j)\,mod\,2}+F_k$（斐波那契数性质）

$\le -\frac{1}{2}F_j+F_k$（斐波那契数另一性质）

$\therefore\ \frac{1}{2}F_j\le F_k-m$

如果m是对方在F_k里取剩下的数，则F_k-m就是对方取的数，上式说明你再按必胜策略取的数不超过上一步对方取数的2倍，是可行的。

（3）若0<m<μ(n)，则μ(n-μ(n)+m)≤2[μ(n)-m]

这就是上面的（2），不过这里用(μ(n)-m)替代（2）里的 F_k -m，μ(n-μ(n)+m)= μ(m)。m表示对方在F_{k_r}里取剩下的石头数，(μ(n)-m)就是对方取的石头数，（n-（μ(n)-m))就是对方取后剩下的总石头数。

这不等式说明你要取的石头数不超过上一步对方取的两数2倍。

证明完毕。

只要开始时石头数不是一个斐波那契数（齐肯多夫表示至少有两项），先手每次取最后一个斐波那契数就必胜，如果开始就是一个斐波那契数，那么后手就必胜。

能否先手取掉一个数后，剩下还是一个斐波那契数给后手？

不可能！

例如开始是F_k，先手要剩下一个斐波那契数给后手，那么至少要取F_{k-2}（这样才留给后手F_{k-1}），而$(F_{k-2})/(F_k)\geq 1/3$的，先手取超过F_{k-2}的数，后手都能一步取完。