顺藤摸瓜 *Pollard's Rho 及其它*

万精油

上期趣味数学专栏的题目是囚犯开盒子问题。为方便解答，我们把上期题目再列一遍。

上期题目：监狱里有 $2k$ 个犯人。监狱长把所有犯人找来对他们说："你们的名字完全随机地放在这 $2k$ 个盒子里，每盒一个。明天你们轮流到这里来，每个人打开一个盒子，看看是不是自己，不是再开下一个，最多可以开 k 个，看到自己的名字就算通过。如果所有人都通过，就释放你们。现在你们可以讨论一个策略，完了之后不准再有任何形式的交流"。

注 1：每个人看到自己名字的概率是 1/2。如果没有策略，释放的概率是 $(1/2)^{(2k)}$，当 $k=5$ 时，成功率已经低于千分之一，k 更大时就几乎成为不可能事件。能不能设计一个策略，使得全体被释放的概率有一个与 k 无关的正下界？

注 2：$2k$ 个盒子从左到右一字排开。每次被打开后马上关上，不能挪动。不能做任何记号。事先商量好对策后犯人之间不能有任何交流。

这由于没有信息传递，初看起来，似乎没有任何办法提高成功率。因为每个人成功的概率是 1/2，大家的选择又相互独立，独立事件的总成功率就是每个成功率的乘积。这个乘积以指数的速度逼近于零。如果我们有办法让大家的 1/2 尽可能的集中（也就是说不独立），那么，我们就可以避开这 $2k$ 个 1/2 的乘积了。下面我们来介绍一种能让大家的 1/2 成功率集中的办法。

题目说盒子从左到右一字排开，所以可以给它们编号。从左到右为 1 到 $2k$。把犯人也编号，1 到 $2k$。每人都记住所有人的号。开盒时先开自己号码对应的盒，如果不对，开盒里的名字对应的盒，这样继续下去直到看到自己的名字。

因为盒子和名字都有号码，盒子和名字的关系可以看成一个置换。比如：(2, 5, 6, 3, 1, 4) 就是一个置换，它把 1 换到 2，2 换到 5，3 换到 6，等等。每个位置对应的数字就是那个位置所换的数。因为数字有限，总有换回来的时候。比如上面这个置换把 5 换到 1，那么(1, 2, 5)就形成了一个子循环。我们可以把上面这个置换分解成两个子置换：(2, 5, 6, 3, 1, 4)=(2, 5, 1)(6, 4, 3)。事实上，如果不是一个大循环，所有置换都可以如此分解成独立的子循环。

囚犯们如果采用这个跟踪名字的策略，每人通过的充分必要条件是自己所在循环的长度不大于 k。所有人都通过的充分必要条件是这个置换没有长度大于 k 的子循环。我们来算一下这件事的概率。

设 $m>k$。假如 $2k$ 个元素的置换有一个长度等于 m 的循环（显然，这样的循环只能有一个）。先算一算这样的循环有多少个。先从 $2k$ 个元素里选 m 个，一共有 $C(2k, m)$ 种选法。一个循环没有始终，如果我们从任意一个元素（比如最小的那个）开始，另外 $m-1$ 可以有 $(m-1)!$ 种不同的顺序，所以，这 m 个元素可以有 $(m-1)!$ 种不同的循环。剩下的 $2k-m$ 个可以有 $(2k-m)!$ 种顺序，所以，$2k$ 个元素的置换有一个长度等于 m 的循环的总数是 $C(2k, m)*(m-1)!*(2k-m)!$，除以总排列数 $(2k)!$ 就是它发生的概率。 $C(2k, m)*(m-$

1)!*(2k − m)!/(2k)! 化简后等于 1/m。M 可以是 k + 1 到 2k 的任意数，把上面的概率从 k+1 到 2k 加起来，得 1/(k + 1) + ... + 1/2k, 这个和的上界为 ln 2。也就是说一个 2k 个元素的置换有一个长度大于 k 的子循环的概率小于 ln 2。因此所有人都通过的概率大于 1 − ln 2 = 0.30685...。这就是采用这种方法后集体通过的概率。这个数与 k 无关，比 1/2 的 2k 次方要大很多。

下面我们来看一个具体例子。

假设有 8 个犯人。从左到右的盒子里的名字（用号码表示）的顺序是 3 7 8 5 4 2 6 1 这个置换群的分解是 (1 3 8), (2 7 6), (5 4)。没有周期大于 4 的子循环，所以每个人都会成功。每个人都先去开对应于自己号码的盒子，然后跟踪盒子里的号码，直到看到自己的号码 (打开 4 个盒子以内)。每个人看到的名字顺序列出来就是

1 → 3 8 1
2 → 7 6 2
3 → 8 1 3
4 → 5 4
5 → 4 5
6 → 2 7 6
7 → 6 2 7
8 → 1 3 8

每个人都在开四个盒子以前就找到了自己的名字（号码），集体成功。

但是，如果从左到右的盒子里的名字（用号码表示）的顺序是 3 4 8 6 7 2 1 5 这个置换群的分解是 (1 3 8 5 7), (2 4 6)。有一个子循环周期大于 4。所以，有 5 个人开完四个盒子后都找不到自己的名字，集体失败。

根据我们前面的计算，在完全随机的情况下，一个置换群有周期大于其半数的概率小于 ln 2，所以，我们这个方法的成功率大于 1 − ln 2。一个与犯人个数无关的常数。

这个方法能够有这么一个与个数无关的成功率的原因是，每个人都遵循同样的原则，顺着一条路走下去。如果有人不成功，就有很多人都不成功。这样就把出错的机会聚集在一起，集体成功的机会自然就增大了。

在很多时候，在看似杂乱无章的地方找东西，最重要的就是要先找到这种可以顺着摸的藤。没有藤，造也要造一条藤出来。整数分解的一个很有名的方法，Pollard's Rho，用的就是这种方法。这个方法很有意思，我们就来简单介绍一下。

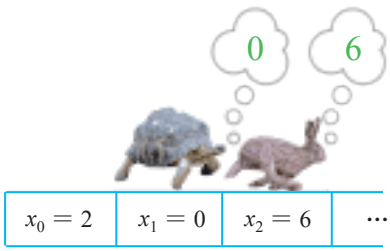
整数分解是一个很古老的问题，对小一点的数，最常见而且最有效的整数分解法是两千多年前古希腊数学家发明的筛法。这个“小一点的数”在不同情况下有不同的意义。如果用手算，5、6 位数就不小了，用计算机算，7、8 位数也算小。不过，不管用什么算，当位数超过 10 位以上时，这种算法的弱点就暴露出来了。因为它必须要从 2 开始，走遍被分解数平方根以内的所有素数，运算量上升的很快。

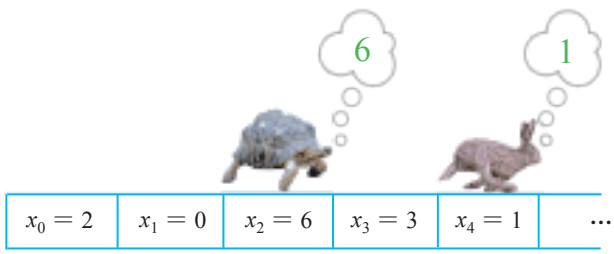
Pollard's Rho 不需要用一个个素数来筛，而是采用顺藤摸瓜的方法。这个藤的建立基于一个被称之为龟兔算法的寻找周期的方法。

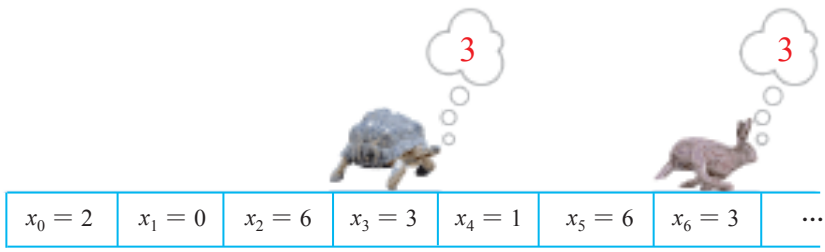
假设有一个从有限集合到有限集合的映射，那么，一直重复这个映射（英文叫 iterated map），总会有一个周期（因为集合有限）。怎样找到这个周期的长度呢？最直观的办法是一直重复这个映射，记下它所经过的所有点，每次新映射，查看新映射的点是否在过去的记录中（也就是说要把过去经过的点都检查一遍），如果有，那么这两个重复点之间的映

射步数就是这个周期的长度。但是，这个方法有两个缺点，一个是需要保持一个数据向量（周期越长，向量长度越大），因为事先不知道这个周期有多长，还必须预备一个大向量。更致命的一个缺点是每次需要查看的数据越来越长。如果周期是 M，总共需查看 $M*(M+1)/2$ 次。龟兔算法不需要查这么多次。除了步数以外，只需保持两个数字，每一步只需检查这两个数是否相等。具体方法是，假设映射是 f，任选一个初始值，然后开始跟踪从这个点出发的两条轨迹。x, y 从同一点出发，x 每次映射一次，$x = f(x)$，y 映射两次，$y = f(f(y))$；因为 f 有周期。进入周期后 x 和 y 就会开始在周期里转圈，y 总会追上 x。当 $x = y$ 时，我们就知道 x 与 y 的步数差就是 f 的周期长度。为什么把它叫龟兔算法呢？x 就是乌龟，每次走一步，y 就是兔子，每次走两步。当兔子甩下乌龟一圈时，兔子与乌龟之间的步数就是周期长度。有人或许会问，如果乌龟不动，兔子走完一圈不是也可以碰到它吗？问题是，并不是所有初始点都在循环圈内，要走一段才会进入周期，所以乌龟和兔子都得动。下面的图会就是一个最好的说明。

这个图显示了龟兔算法应用到序列：2, 0, 6, 3, 1, 6, 3, 1, 6, 3, 1, ...







龟兔算法能被用来分解整数还依赖于一个准则——生日准则。生日准则说当一群人的人数超过 23 时，这群人中有两个人同生日的概率大于 50%。更一般的情况是，不断随机取两个数 X 与 Y，取到 $1.177\sqrt{p}$ 次以上时，这两个数模 p 出现同余的可能性大于 50%。在生日准则中，$p = 365$，$1.177\sqrt{p} = 22.49$。

准备工作做好了，我们现在回到整数分解。

假设 p 是我们想要分解的数 n 的一个因子，如果我们有一对数 x，y 模 p 同余，$d =$

gcd($x-y$, n) 就是 n 的一个因子。这里 gcd 表示最大公约数。如果 d 不是 1 或 n, 那么 d 就是 n 的一个因子。

Pollard's Rho 采用的方法就是构造某种映射，然后采用龟兔算法模拟随机数对。这些映射都是模 n 以后取值，生日准则告诉我们跟踪映这些数对足够次以后，这些数对出现同余的可能性很大，这样我们就找到了我们想要分解的数的因子。

具体一点，Pollar's Rho 算法可写成

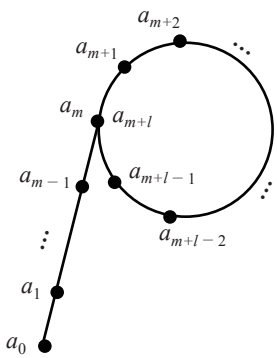
输入 n, 这是我们希望分解的数
输出：n 的因子。如果这个因子不是 1 或 n，成功，否则失败。

1. $x \leftarrow 2, y \leftarrow 2; d \leftarrow 1$
2. While $d = 1$:
1. $x \leftarrow f(x)$
2. $y \leftarrow f(f(y))$
3. $d \leftarrow \mathrm{GCD}(|x-y|, n)$
3. If $d = n$, return failure.
4. Else, return d.

上面算法中，f 是一个映射，模 n 取值（映射到有限集合）。我们现在来看一个具体例子：

假设 $n = 8051$, $f(x) = \mathrm{mod}\ (x^{2+1}, n)$,
给初始值 $x = 2$; $y = 2$;
第一步：$x = 5$; $y = 26$; $\gcd(x-y, 8051) = 1$;
第二步：$x = 26$; $y = 7474$; $\gcd(x-y, 8051) = 1$;
第三步：$x = 677$; $y = 871$; $\gcd(x-y, 8051) = 97$;
三步就找到了 8051 的因子 97。

这个算法发明者是 John Pollar，算法名称后面那个 Rho 是因为这些数对除了开始的尾巴，很快就进入周期，很像希腊字母 ρ。见附图。



需要注意的是，因为这个算法建立在概率基础上，不能保证每次都成功，有时会失败。这时我们就需要变换映射函数 f。第二，如果 n 是素数，这个算法会一直执行下去，所以，执行足够次以后需要加一个停止指令。

Pollar's Rho 最有效的时候是 n 有小因子的时候。如果 n 的因子都很大，则效率不高。

Pollar's Rho 最辉煌的成就是首先成功分解了第八个费尔马数 $2^{2^8}+1$。这个数有 78 位数。筛法是不可能筛出来的。因为要走遍 39 位数以内所有的素数，走到太阳系毁灭时间也不够。

我们上面给的只是最简单的算法，基于同样原理的算法后来又被改造，发展出一些更有效更快的算法，有兴趣的读者可以自己去找来看一看。还有很多与 Pollar's Rho 不同，或者说比它更高级的整数分解算法，需要用到椭圆曲线。专门用来对付大整数分解。

整数分解算法之所以重要，是因为绝大多数网络安全系统都依赖于整数分解的难度。整数分解是目前网络安全系统研究的重要课题之一。

本期趣味题目：

棋盘定位：教授把你叫到他的办公室，给你一个国际象棋棋盘（8×8 方格），棋盘上有些格子里有棋子，有些格子里没有棋子。棋子都是翻过来的，也就是说每个棋子都一样。教授在棋盘上随便指了一格。你现在需要做下面两件事中的一个：

1. 选一个有棋子的格子，把棋子从格子中拿走。

或者

2. 选一个没有棋子的格子，放一个棋子到格子中。

教授再把你的朋友叫进来，把你改变过的棋盘给他看。他的任务是从棋盘中看出教授所指的那个格子。

注意，棋盘的初始状态是随机的，教授所指的格子也是随机的。两个动作你只能选一样来做。你可以事先与你的朋友商量好一套利用棋盘上的棋子位置的信息传递系统，使得你的朋友总可以确定教授所指的格子。