欧拉路径和De Bruijn序列

文 / 顾森

图论是离散数学和算法领域中的一个重要分支，是描述自然现象和人类活动的一个非常有力的模型。给定一些顶点，再告诉你哪些顶点之间有连线，这就构成了一个最基本的图。图论在运筹学中地位很重要，交通道路设计、货物运输路径、管道铺设、活动安排等问题都可以直接转化为图论问题。在一些极其抽象的组合构造类问题中，图论也发挥着巨大的作用。

在Sean Eron Anderson的Bit Twiddling Hacks网页里（http://graphics.stanford.edu/~seander/bithacks.html），有一段非常诡异的代码：

```
unsigned int v;  // find the number of trailing zeros in v
int r;           // result goes here
static const int MultiplyDeBruijnBitPosition[32] =
  {
    0, 1, 28, 2, 29, 14, 24, 3, 30, 22, 20, 15,
    25, 17, 4, 8, 31, 27, 13, 23, 21, 19, 16,
    7, 26, 12, 18, 6, 11, 5, 10, 9
  };
r = MultiplyDeBruijnBitPosition[
  ((uint32_t)((v & -v) * 0x077CB531U)) >> 27];
```

这段代码可以非常快速地给出一个数的二进制表达中末尾有多少个0。比如，67 678 080的二进制表达是100 00001000 10101111 10000000，因此这个程序给出的结果就是7。熟悉位运算的朋友们可以认出，v & -v的作用就是取出末尾连续的0以及右起第一个1。当v = 67 678 080时，v & -v就等于128，即二进制的10000000。怪就怪在，这个0x077CB531是怎么回事？不妨把这个常量写成32位二进制数，可以得到：

```
00000111011111001011010100110001
```

这个01串有一个非常难能可贵的性质：如果把它看作是循环的，它正好包含了全部32种可能的5位01串，既无重复，又无遗漏！

```
[00000]11101111100101110101001100001 → 00000
0[00001]1101111100101110101001100001 → 00001
00[00011]101111100101110101001100001 → 00011
..................................... ......
00]00011101111100101110101001100[001 → 00100
000]00111011111001011010100110000[01 → 01000
0000]01110111111001011010100110000[1 → 10000
```

天哪！这个01串是怎么构造出来的？这还得从18世纪的德国开始说起。

欧拉路径

德国东普鲁士的哥尼斯堡城坐落于普列戈利亚河的两侧，河流中另有两块很大的岛。整个哥尼斯堡城就这样被分成了四块，它们被七座桥连在一起。据说，每逢周日，人们都会在城里散步，并尝试着寻找一条散步的路线，能够既无重复又无遗漏地经过每座桥恰好一次。从哪块区域出发可以由自己决定，最后也不必回到出发时的区域，但每次过桥时必须完全经过这座桥，不允许在走到桥的中间时折返回来。这就是著名的“哥尼斯堡七桥问题”：究竟是否存在这样一条满足要求的路线呢？

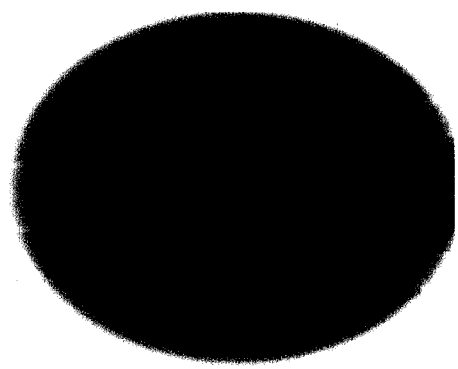


图1 哥尼斯堡的七座桥

瑞士大数学家欧拉开创性地想出了一种分析哥尼斯堡七桥问题的方法。他发现，实际上哥尼斯堡四块区域的形状和位置并不重要，七座桥的长短也不会对问题造成实质性的影响，重要的仅仅是每座桥连接的都是哪里和哪里。于是，欧拉把每个

区域都用一个点来表示，把每座桥都用一条连线来表示，整个地图就被抽象成了图2。这种大胆的抽象奠定了两个数学新分支的基础——图论和拓扑学。我们现在的任务就变成了一个简单的图论问题——判断图2是否能一笔画完。

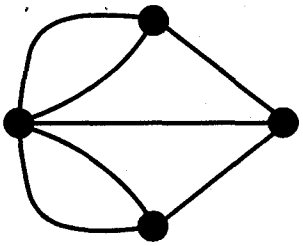


图2 抽象后的七桥问题

在1736年的一篇论文中，欧拉证明了这个图是不能一笔画完成的。假设这个图是可以一笔画完成的，那么除了一笔画线路的起点和终点以外，其他的所有点都必然是“有进必有出”，“进去过多少次就出来过多少次”。如果我们把一个点引出的线条数叫做这个点的“度”的话，那么你会发现，除了一笔画线路的起点和终点允许奇数的度，其余所有点的度都必须是偶数。但图2中4个点的度数都是奇数，因而它是不能一笔画完成的。

我们把遍历图中每条连线恰好一次的路径叫做“欧拉路径”。上面的推理实际上刻画了一个图里存在欧拉路径的必要条件：度数为奇数的顶点最多只能有两个。如果额外地要求这条路径最后必须回到起点处，则图里完全不能有任何度数为奇数的顶点出现。我们接下来关心的问题就是，这些条件同时也是欧拉路径存在的充分条件吗？换句话说，如果一个图真的满足这些条件，那么欧拉路径就一定存在吗？1873年，Carl Hierholzer的一篇论文对此给出了一个肯定的回答：只要整个图是连通的，并且满足前面关于度数的条件，那么欧拉路径不但是一定存在的，而且我们还能高效地生成一条合法的路径。

Hierholzer的欧拉路径生成方法非常简单。让我们先来看一下图中没有奇数度顶点，所有顶点度数均为偶数的情况。首先，我们可以从中选择任意一个点作为出发点，往任意一个方向走一步（此时出发点的度数就变成奇数了），并且不断地走下去，直至回到出发点为止。在这个过程中，我们绝对不会“走死”，因为除了现在的出发点以外，每个顶点的度数都是偶数，进去了是一定能出来的。但是，当我们走回出发点后，我们可能还没有走遍所有的路。没关系。现在，我们擦掉所有已走过的路。容易看出，对于每一个顶点来说，如果它的度数减少了，那一定是成双成对地减少。因此，擦掉走过的路后，剩余的图仍然满足所有顶点度数均为偶数的性质。我们在走过的路上找出一个点，使得它上面还连着一些没有走过的、仍然留在剩余图中的道路。从这个点出发，像刚才那样，在剩余的图中走出一条回路，然后把这条新的回路插入到刚才的路线里，并从剩余的图中擦掉。不断从剩余的图中寻找回路，并且并入已生成的回路里。考虑到整个图是连通的，因此最终我们将会得到遍历所有道路恰好一次的路径。

这个方法很容易扩展到有两个奇数度顶点的情况，只需要一开始从其中一个奇数度顶点出发，最后必然会到达另外一个奇数度顶点。然后，像刚才那样，不断在这条路径上插入一个个回路，直到这条路径包含了所有的道路。

即使在有重边（同一对顶点间有多条连线）、有自环（从某个顶点出发连一条线到自身）的情况下，上面的这些推理同样适用。欧拉路径的问题还可以扩展到有向图上：假如我们规定每条连线都是“单行道”，那么怎样的图才会有遍历所有连线的路径呢？对于任意一个顶点，我们把指向该顶点的线条数叫做它的“入度”，把从该顶点向外发出的线条数叫做它的“出度”。容易想到，为了保证欧拉路径的存在，我们必须要让每个顶点的入度都等于出度；如果允许路线的起点和终点不重合，我们还可以允许有一个顶点的出度等于入度加1，有另一个顶点的入度等于出度加1。事实上，这些条件不但是欧拉路径存在的必要条件，也是欧拉路径存在的充分条件，推理方法仍然如上，这里就不再赘述了。

De Bruijn序列

很多保险箱的密码都是四位数，这足以给人带来安全感——由于从0000到9999共有10000种情况，要想试遍所有的密码，按40000次数字键似乎是必需的。但有些保险箱的数字键盘上并没

有输入键。只要连续输入的四个数字恰好和预先设定的密码相同，保险箱都会打开。例如，当你尝试输入1234和1235两个密码时，2341、3412、4123也被试过了。聪明的小偷可以利用这一特性，设计出一种数字输入顺序，大大减少最坏情况下需要的总按键次数。你认为，试遍所有的密码最少需要按多少个键呢？

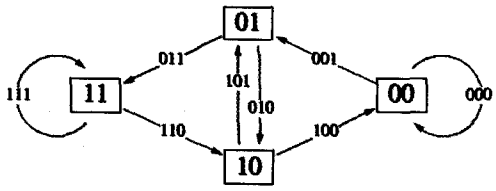


图3　图上有00、01、10、11四个点，如果两个点满足xy和yz的关系（x、y、z有可能代表相同的数字），就画一条从xy到yz的路，这条路就记作xyz

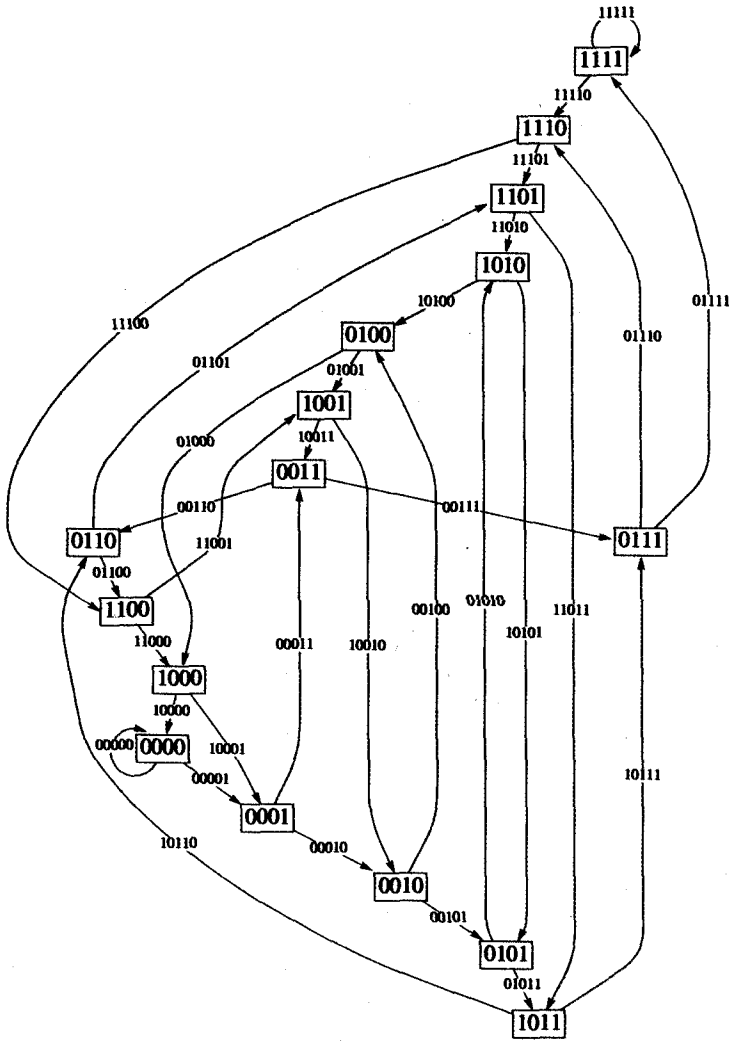


图4　用于生成B(2, 5)的图，图中的每一条欧拉路径都对应了一个B(2, 5)的具体方案

如果一个数字序列包含了10000个不同的四位数，这个数字序列至少得有10003位长。令人吃惊的是，真的就存在这么一个10003位的数字序列，它既无重复又无遗漏地包含了所有可能的四位数。更神的是，满足要求的数字序列不止一种，而寻找数字序列的任务竟完全等价于一笔画问题！

为了把事情解释清楚，不妨让我们先来看一个更为简单的问题：假如密码是一个只由数字0和1构成的三位数（这有8种可能的情况），如何构造一个10位数字串，使它正好包含所有可能的三位数？现在，让我们在图上画四个点，分别标记为00、01、10、11，它们表示数字串中相邻两个数字可能形成的4种情况。如果某个点上的数的后面一位，恰好等于另一个点上的数的第一位，就从前面那个点出发，画一个到后面那个点的箭头，表示从前面那个点可以走到后面这个点。举例来说，00的后一位数正好是01的前一位数，则我们画一个从00到01的箭头，意即从00可以走到01。注意，有些点之间是可以相互到达的（例如10和01），有些点甚至有一条到达自己的路（例如00）。

试密码的过程，其实就相当于沿着箭头在图3中游走的过程。不妨假设你最开始输入了00。如果下一次你输入了1，那么你就试过了001这个密码，同时你最近输过的两位数就变成了01；如果你下一次还是输入的0，那么你就试过了000这个密码，你最近输过的两个数仍然是00。从图上看，这无非是一个从00点出发走了哪条路的问题：你是选择了沿001这条路走到了01这个点，还是沿着000这条路走回了00这个点。同理，你每按下一个数字，就相当于沿着某条路走到了一个新的点，路上所写的三位数就是你刚才试过的密码。我们的问题就可以简单地概括为，如何既无重复又无遗漏地走完图3中的所有路。也就是说，我们要解决的仅仅是一个欧拉路径的问题！

稍试几下，我们便可以找出一条欧拉路径。其中一条路就是：

```
00 → 00 → 01 → 10 → 01 → 11 → 11 → 10 → 00
```

它给出了一个满足要求的 10 位数字序列：

```
0001011100
```

这个10位数字串就真的包含了全部8个由0和1构成的三位数！事实上，利用上一节的结论，我们可以直接看出这个图一定能一笔画走完的。很显然，在图3中，从任意一点出发，都有两条路可以走；同时，走到这个点也总有两种不同的途径。这说明，图中每个点的出度都等于入度。这就告诉我们，遍历所有路恰好一次的方法是一定存在的。

同样地，对于0到9组成的全部四位数来说，我们可以设置1000个顶点，分别记作000, 001, …, 999。如果某个数的后两位等于另一个数的前两位，就从前者出发，画一个箭头指向后者。容易想到，每个顶点的入度和出度都是10，因此图中存在一条欧拉路径。也就是说，只需要按10003次数字键，便能试遍所有可能的四位数密码了。利用前面讲到的*Hierholzer*算法，我们可以很快给出一个满足要求的10003位数字序列。

其实，由于图中的所有点入度都等于出度，因此图中的任意一条欧拉路径一定是一条终点等于起点的回路。所以，这10003位数的末3位一定等于头3位。或者，我们可以把它看作一个循环的10000位数，末位的下一位就直接跳到了首位。这样一来，10000位数里就可以包含10000个长度为4的子串，它们恰好对应了所有可能的四位数密码。

类似地，假设我们有一个大小为k的字符集，那么我们一定能找出一个字符串，如果把它看作是循环的，则它既无重复又无遗漏地包含所有可能的n位字符串。我们把这样的字符串叫做De Bruijn序列，记作B(k, n)。它是以荷兰数学家Nicolaas Govert de Bruijn的名字命名的，这位涉猎广泛的数学家在2012年刚刚逝世，享年93岁。

本文开头提到的01串，其实就是B(2, 5)。理论上，为了把B(2, 5)写成一个01串，我们可以从任意地方断开这个循环的“字符圈”。出于某种后面将会作出解释的目的，我们有意在“00000”的前面断开，得到00000111011111001011010100110001。把它写成十六进制，就成了本文开头的代码中设定的神秘常量0x077CB531。

0x077CB531在代码中究竟有什么用呢？它的用途其实很简单，就是为32种不同的情况提供了一个唯一索引。比方说，10000000后面有7个0，将10000000乘以0x077CB531，就得到：

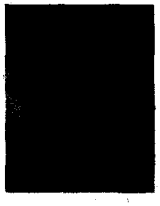
```
00000111011111001011010100110001
→ 10111110010110101001100010000000
```

这相当于是把De Bruijn序列左移了7位。再把这个数右移27位，就相当于提取出了这个数的头5位：

```
10111110010110101001100010000000
→                              10111
```

由于De Bruijn序列的性质，因此当输入数字的末尾0个数不同时，最后得到的这个5位数也不同。而数组MultiplyDeBruijnBitPosition则相当于一个字典的功能。10111转换回十进制是23，于是我们查一查MultiplyDeBruijnBitPosition[23]，程序即返回7。注意到当输入数字末尾的0超过27个时，代码仍然是正确的，因为我们选用的De Bruijn序列是以00000开头的，而对二进制数进行左移时低位正好是用0填充的，因此这就像是把开头的00000循环左移过来了一样，于是De Bruijn序列的性质便得以继续保存。

在科学研究和工业设计中，De Bruijn序列还有很多奇妙的应用，感兴趣的朋友们不妨在网上查阅相关的资料。



顾森

网名Matrix67，数学爱好者。2005年开办数学博客http://www.matrix67.com，至今已积累上千篇文章，有上万人订阅。新书《思考的乐趣》已在图灵公司出版。

责任编辑：卢鹏翔（ludx@csdn.net）