数论部分第一节：素数与素性测试

一个数是素数（也叫质数），当且仅当它的约数只有两个——1和它本身。规定这两个约数不能相同，因此1不是素数。对素数的研究属于数论范畴，你可以看到许多数学家没事就想出一些符合某种性质的素数并称它为某某某素数。整个数论几乎就围绕着整除和素数之类的词转过去转过来。对于写代码的人来说，素数比想像中的更重要，Google一下BigPrime或者big_prime你总会发现大堆大堆用到了素数常量的程序代码。平时没事时可以记一些素数下来以备急用。我会选一些好记的素数，比如4567, 124567, 3214567, 23456789, 55566677, 12345678949876543211, 11111111111111111111 (23个1)。我的手机号前10位是个素数。我的网站域名的ASCII码连起来(77 97 116 114 105 120 54 55 46 99 111 109)也是个素数。还有，我的某个MM的八位生日也是一个素数。每次写Hash函数之类的东西需要一个BigPrime常量时我就取她的生日，希望她能给我带来好运。偶尔我叫她素MM，没人知道是啥意思，她自己也不知道。

素数有很多神奇的性质。我写5个在下面供大家欣赏。

1. 素数的个数无限多（不存在最大的素数）

证明：反证法，假设存在最大的素数P，那么我们可以构造一个新的数2 * 3 * 5 * 7 * … * P + 1（所有的素数乘起来加1）。显然这个数不能被任一素数整除（所有素数除它都余1），这说明我们找到了一个更大的素数。

2. 存在任意长的一段连续数，其中的所有数都是合数（相邻素数之间的间隔任意大）

证明：当0<a<=n时，n!+a能被a整除。长度为n-1的数列n!+2, n!+3, n!+4, …, n!+n中，所有的数都是合数。这个结论对所有大于1的整数n都成立，而n可以取到任意大。

3. 所有大于2的素数都可以唯一地表示成两个平方数之差。

证明：大于2的素数都是奇数。假设这个数是2n+1。由于(n+1)^2=n^2+2n+1，(n+1)^2和n^2就是我们要找的两个平方数。下面证明这个方案是唯一的。如果素数p能表示成a^2-b^2，则p=a^2-b^2=(a+b)(a-b)。由于p是素数，那么只可能a+b=p且a-b=1，这给出了a和b的唯一解。

4. 当n为大于2的整数时，2^n+1和2^n-1两个数中，如果其中一个数是素数，那么另一个数一定是合数。

证明：2^n不能被3整除。如果它被3除余1，那么2^n-1就能被3整除；如果被3除余2，那么2^n+1就能被3整除。总之，2^n+1和2^n-1中至少有一个是合数。

5. 如果p是素数，a是小于p的正整数，那么a^(p-1) mod p = 1。

这个证明就有点麻烦了。

首先我们证明这样一个结论：如果p是一个素数的话，那么对任意一个小于p的正整数a，a, 2a, 3a, …, (p-1)a除以p的余数正好是一个1到p-1的排列。例如，5是素数，3, 6, 9, 12除以5的余数分别为3, 1, 4, 2，正好就是1到4这四个数。

反证法，假如结论不成立的话，那么就是说有两个小于p的正整数m和n使得na和ma除以p的余数相同。不妨假设n>m，则p可以整除a(n-m)。但p是素数，那么a和n-m中至少有一个含有因子p。这显然是不可能的，因为a和n-m都比p小。

用**同余式**表述，我们证明了：

(p-1)! ≡ a * 2a * 3a * … * (p-1)a (mod p)

也即：

(p-1)! ≡ (p-1)! * a^(p-1) (mod p)

两边同时除以(p-1)!，就得到了我们的最终结论：

1 ≡ a^(p-1) (mod p)

可惜最后这个定理最初不是我证明的。这是大数学家Fermat证明的，叫做Fermat小定理(Fermat's Little Theorem)。Euler对这个定理进行了推广，叫做Euler定理。Euler一生的定理太多了，为了和其它的“Euler定理”区别开来，有些地方叫做Fermat小定理的Euler推广。Euler定理中需要用一个函数f(m)，它表示小于m的正整数中有多少个数和m互素（两个数只有公约数1称为互素）。为了方便，我们通常用记号φ(m)来表示这个函数（称作Euler函数）。Euler指出，如果a和m互素，那么a^φ(m) ≡ 1 (mod m)。可以看到，当m为素数时，φ(m)就等于m-1（所有小于m的正整数都与m互素），因此它是Fermat小定理的推广。定理的证明和Fermat小定理几乎相同，只是要考虑的式子变成了所有与m互素的数的乘积：m_1 * m_2 … m_φ(m) ≡ (a * m_1)(a * m_2) … (a * m_φ(m)) (mod m)。我为什么要顺便说一下Euler定理呢？因为下面一句话可以增加我网站的PV：这个定理出现在了**The Hundred Greatest Theorems**里。

谈到Fermat小定理，数学历史上有很多误解。很长一段时间里，人们都认为Fermat小定理的逆命题是正确的，并且有人亲自验证了a=2, p<300的所有情况。国外甚至流传着一种说法，认为中国在孔子时代就证明了这样的定理：如果n整除2^(n-1)-1，则n就是素数。后来某个英国学者进行考证后才发现那是因为他们翻译中国古文时出了错。1819年有人发现了Fermat小定理逆命题的第一个反例：虽然2的340次方除以341余1，但341=11*31。后来，人们又发现了561, 645, 1105等数都表明a=2时Fermat小定理的逆命题不成立。虽然这样的数不多，但不能忽视它们的存在。于是，人们把所有能整除2^(n-1)-1的合数n叫做伪素数(pseudoprime)，意思就是告诉人们这个素数是假的。

不满足2^(n-1) mod n = 1的n一定不是素数；如果满足的话则多半是素数。这样，一个比试除法效率更高的素性判断方法出现了：制作一张伪素数表，记录某个范围内的所有伪素数，那么所有满足2^(n-1) mod n = 1且不在伪素数表中的n就是素数。之所以这种方法更快，是因为我们可以使用二分法快速计算2^(n-1) mod n 的值，这在计算机的帮助下变得非常容易；在计算机中也可以用二分查找有序数列、Hash表开散列、构建Trie树等方法使得查找伪素数表效率更高。

有人自然会关心这样一个问题：伪素数的个数到底有多少？换句话说，如果我只计算2^(n-1) mod n的值，事先不准备伪素数表，那么素性判断出错的概率有多少？研究这个问题是很有价值的，毕竟我们是OIer，不可能背一个长度上千的常量数组带上考场。统计表明，在前10亿个自然数中共有50847534个素数，而满足2^(n-1) mod n = 1的合数n有5597个。这样算下来，算法出错的可能性约为0.00011。这个概率太高了，如果想免去建立伪素数表的工作，我们需要改进素性判断的算法。

最简单的想法就是，我们刚才只考虑了a=2的情况。对于式子a^(n-1) mod n，取不同的a可能导致不同的结果。一个合数可能在a=2时通过了测试，但a=3时的计算结果却排除了素数的可能。于是，人们扩展了伪素数的定义，称满足a^(n-1) mod n = 1的合数n叫做以a为底的伪素数(pseudoprime to base a)。前10亿个自然数中同时以2和3为底的伪素数只有1272个，这个数目不到刚才的1/4。这告诉我们如果同时验证a=2和a=3两种情况，算法出错的概率降到了0.000025。容易想到，选择用来测试的a越多，算法越准确。通常我们的做法是，随机选择若干个小于待测数的正整数作为底数a进行若干次测试，只要有一次没有通过测试就立即把
这个数扔回合数的世界。这就是Fermat素性测试。

人们自然会想，如果考虑了所有小于n的底数a，出错的概率是否就可以降到0呢？没想到的是，居然就有这样的合数，它可以通过所有a的测试（这个说法不准确，详见我在地核楼层的回复）。Carmichael第一个发现这样极端的伪素数，他把它们称作Carmichael数。你一定会以为这样的数一定很大。错。第一个Carmichael数小得惊人，仅仅是一个三位数，561。前10亿个自然数中Carmichael数也有600个之多。Carmichael数的存在说明，我们还需要继续加强素性判断的算法。

Miller和Rabin两个人的工作让Fermat素性测试迈出了革命性的一步，建立了传说中的Miller-Rabin素性测试算法。新的测试基于下面的定理：如果p是素数，x是小于p的正整数，且x^2 mod p = 1，那么要么x=1，要么x=p-1。这是显然的，因为x^2 mod p = 1相当于p能整除x^2-1，也即p能整除(x+1)(x-1)。由于p是素数，那么只可能是x-1能被p整除(此时x=1)或x+1能被p整除(此时x=p-1)。

我们下面来演示一下上面的定理如何应用在Fermat素性测试上。前面说过341可以通过以2为底的Fermat测试，因为2^340 mod 341=1。如果341真是素数的话，那么2^170 mod 341只可能是1或340；当算得2^170 mod 341确实等于1时，我们可以继续查看2^85除以341的结果。我们发现，2^85 mod 341=32，这一结果摘掉了341头上的素数皇冠，面具后面真实的嘴脸显现了出来，想假扮素数和我的素MM交往的企图暴露了出来。

这就是Miller-Rabin素性测试的方法。不断地提取指数n-1中的因子2，把n-1表示成d*2^r（其中d是一个奇数）。那么我们需要计算的东

西就变成了a的d*2^r次方除以n的余数。于是，a^(d * 2^(r-1))要么等于1，要么等于n-1。如果a^(d * 2^(r-1))等于1，定理继续适用于a^(d * 2^(r-2))，这样不断开方开下去，直到对于某个i满足a^i 指数中的2用完了得到的a^d mod n=1或n-1。这样，Fermat小定理加强为如下形式：

尽可能提取因子2，把n-1表示成d*2^r，如果n是一个素数，那么或者a^d mod n=1，或者存在某个i使得a^(d*2^i) mod n=n-1（0<=i<r）（注意i可以等于0，这就把a^d mod n=n-1的情况统一到后面去了）

Miller-Rabin素性测试同样是不确定算法，我们把可以通过以a为底的Miller-Rabin测试的合数称作以a为底的强伪素数(strong pseudoprime)。第一个以2为底的强伪素数为2047。第一个以2和3为底的强伪素数则大到1 373 653。

Miller-Rabin算法的代码也非常简单：计算d和r的值（可以用位运算加速），然后二分计算a^d mod n的值，最后把它平方r次。程序的代码比想像中的更简单，我写一份放在下边。虽然我已经转C了，但我相信还有很多人看不懂C语言。我再写一次Pascal吧。函数IsPrime返回对于特定的底数a，n是否是能通过测试。如果函数返回False，那说明n不是素数；如果函数返回True，那么n极有可能是素数。**注意这个代码的数据范围限制在longint，你很可能需要把它们改成int64或高精度计算。**

```
function pow( a, d, n:longint ):longint;
begin
   if d=0 then exit(1)
   else if d=1 then exit(a)
   else if d and 1=0 then exit( pow( a*a mod n, d div 2, n) mod n)
   else exit( (pow( a*a mod n, d div 2, n) * a) mod n);
end;
```

```
function IsPrime( a,n:longint ):boolean;
var
   d,t:longint;
begin
   if n=2 then exit(true);
   if (n=1) or (n and 1=0) then exit(false);
   d:=n-1;
   while d and 1=0 do d:=d shr 1;
   t:=pow( a, d, n );
   while ( d<>n-1 ) and ( t<>1 ) and ( t<>n-1 ) do
   begin
      t:=(t * t)mod n;
      d:=d shl 1;
   end;
   exit( (t=n-1) or (d and 1=1) );
end;
```

对于大数的素性判断，目前Miller-Rabin算法应用最广泛。一般底数仍然是随机选取，但当待测数不太大时，选择测试底数就有一些技巧了。比如，如果被测数小于4 759 123 141，那么只需要测试三个底数2, 7和61就足够了。当然，你测试的越多，正确的范围肯定也越大。如果你每次都用前7个素数(2, 3, 5, 7, 11, 13和17)进行测试，所有不超过341 550 071 728 320的数都是正确的。如果选用2, 3, 7, 61和24251作为底数，那么10^16内唯一的强伪素数为46 856 248 255 981。这样的一些结论使得Miller-Rabin算法在OI中非常实用。通常认为，Miller-Rabin素性测试的正确率可以令人接受，随机选取k个底数进行测试算法的失误率大概为4^(-k)。

Miller-Rabin算法是一个RP算法。RP是时间复杂度的一种，主要针对判定性问题。一个算法是RP算法表明它可以在多项式的时间里完成，对于答案为否定的情形能够准确做出判断，但同时它也有可能把对的判成错的（错误概率不能超过1/2）。RP算法是基于随机化的，因此多次运行该算法可以降低错误率。还有其它的素性测试算法也是概率型的，比如Solovay-Strassen算法。另外一些素性测试算法则需要预先知道一些辅助信息（比如n-1的质因子），或者需要待测数满足一些条件（比如待测数必须是2^n-1的形式）。前几年AKS算法轰动世界，它是第一个多项式的、确定的、无需其它条件的素性判断算法。当时一篇论文发表出来，题目就叫PRIMES is in P，然后整个世界都疯了，我们班有几个MM那天还来了初潮。算法主要基于下面的事实：n是一个素数当且仅当(x-a)^n≡(x^n-a) (mod n)。注意这个x是多项式中的未知数，等式两边各是一个多项式。举个例子来说，当a=1时命题等价于如下结论：当n是素数时，杨辉三角的第n+1行除两头的1以外其它的数都能被n整除。

Matrix67原创