多的访问次数。

欧几里得算法

第二个例子是计算最大公因数的欧几里得算法。两个整数的最大公因数(gcd)是同时整除二者的最大整数。于是，$gcd(50,15)=5$。图 2-10 所示的算法计算 $gcd(M,N)$，假设 $M \geqslant N$(如果 $N>M$，则循环的第一次迭代将它们互相交换)。

```
1       public static long gcd( long m, long n )
2       {
3           while( n != 0 )
4           {
5               long rem = m % n;
6               m = n;
7               n = rem;
8           }
9           return m;
10      }
```

图 2-10 欧几里得算法

算法连续计算余数直到余数是 0 为止，最后的非零余数就是最大公因数。因此，如果 $M=1989$ 和 $N=1590$，则余数序列是 399,393,6,3,0。从而，$gcd(1989,1590)=3$。正如例子所表明的，这是一个快速算法。

如前所述，估计算法的整个运行时间依赖于确定余数序列究竟有多长。虽然 $\log N$ 看似像理想中的答案，但是根本看不出余数的值按照常数因子递减的必然性，因为我们看到，例中的余数从 399 仅仅降到 393。事实上，在一次迭代中余数并不按照一个常数因子递减。然而，我们可以证明，在两次迭代以后，余数最多是原始值的一半。这就证明了，迭代次数至多是 $2\log N=O(\log N)$从而得到运行时间。这个证明并不难，因此我们将它放在这里，可从下列定理直接推出它。

定理 2.1 如果 $M>N$，则 $M \bmod N<M/2$。

证明：

存在两种情形。如果 $N \leqslant M/2$，则由于余数小于 N，故定理在这种情形下成立。另一种情形是 $N>M/2$。但是此时 M 仅含有一个 N 从而余数为 $M-N<M/2$，定理得证。 ■

从上面的例子来看，$2\log N$ 大约为 20，而我们仅进行了 7 次运算，因此有人会怀疑这是不是可能的最好的界。事实上，这个常数在最坏的情况下还可以稍微改进成 $1.44\log N$(如 M 和 N 是两个相邻的斐波那契数时就是这种情况)。欧几里得算法在平均情况下的性能需要大量篇幅的高度复杂的数学分析，其迭代的平均次数约为$(12\ln 2\ln N)/\pi^2+1.47$。

幂运算

我们在本节的最后一个例子是处理一个整数的幂(它还是一个整数)。由取幂运算得到的数一般都是相当大的，因此，我们只能在假设有一台机器能够存储这样一些大整数(或有一个编译程序能够模拟它)的情况下进行我们的分析。我们将用乘法的次数作为运行时间的度量。

计算 X^N 的明显的算法是使用 $N-1$ 次乘法自乘。有一种递归算法效果更好。$N \leqslant 1$ 是这种递归的基准情形。否则，若 N 是偶数，我们有 $X^N=X^{N/2}\cdot X^{N/2}$，如果 N 是奇数，则 $X^N=X^{(N-1)/2}\cdot X^{(N-1)/2}\cdot X$。

例如，为了计算 X^{62}，算法将如下进行，它只用到 9 次乘法：

$$X^3=(X^2)X,\ X^7=(X^3)^2X,X^{15}=(X^7)^2X,X^{31}=(X^{15})^2X,X^{62}=(X^{31})^2$$

显然，所需要的乘法次数最多是 $2\log N$，因为把问题分半最多需要两次乘法(如果 N 是奇数)。这