Greatest Common Divisor

Textbook Reading (Algorithms: Foundations and Design Strategies)

- Section 1.2, pp. 14-16

Greatest Common Divisor (gcd)

The greatest common divisor of two integers a and b, denoted by $\mathrm{gcd}(a, b)$ is the largest integer that divides both.

Computing gcd using prime factorization

The prime factorization of a number *n* is the unique product of prime powers that equals *n*.

For example,

$$3000 = 2^3 \times 3 \times 5^3$$

$$7700 = 2^2 \times 5^2 \times 7 \times 11$$

The gcd(a,b) is obtained the product of the of the smallest power of each prime from the prime factorizations of a and b, where the smallest power is 0 if the prime does not occur in the factorization.

$$\gcd(3000,7700) = 2^2 \times 5^2 = 100$$

- It turns out that prime factorization for large integers, i.e., hundreds of digits, is a “hard” problem and it is not known how to solve in real time.
- However, the greatest common divisor can be computed efficiently using an algorithm that dates all the way back to Euclid who lived c. 325 – c. 270 BC in Alexandria, Egypt.

Recurrence Relation for gcd

The key idea is to use the recursion relation

$\gcd(a, b) = \gcd(b, r)$, where $r = a \bmod b$.

The initial condition is $\gcd(a, 0) = a$.

The concept of 0 had not be invented in Euclid's time, so the initial condition he used was more cumbersome.

Example gcd(3000,7700)

$$3000 = 0 \times 7700 + 3000$$
$$7700 = 2 \times 3000 + 1700$$
$$3000 = 1 \times 1700 + 1300$$
$$1700 = 1 \times 1300 + 400$$
$$1300 = 3 \times 400 + 100$$
$$400 = 4 \times 100 + 0$$

$$\gcd(3000,7700) = \gcd(7700,3000) = \gcd(3000,1700) =$$
$$\gcd(1700,1300) = \gcd(1300,400) = \gcd(400,100) =$$
$$\gcd(100,0) = 100$$

PSN. Using Euclid’s algorithm compute gcd(585,1035)

Recursive version of Euclid GCD

function EuclidGCD(a,b)
 Input: a, b (nonnegative integers)
 Output: gcd(a,b)
  if (b == 0)
   return a
  else
   r = a mod b
   return EuclidGCD(b,r)
  endif
 end EuclidGCD

Correctness

PSN. Prove the recurrence relation

$$\gcd(a, b) = \gcd(b, r) \text{ for } b > 0.$$

Hint: Use the technique $p = q$ iff

$$\forall x \in N, x|p \leftrightarrow x|q$$

where N is the nonzero natural numbers and | stands for divides.

Nonrecursive version

```
function EuclidGCD(a,b)
Input: a, b (nonnegative integers)
Output: gcd(a,b)
    while b ≠ 0 do
        Remainder = a mod b
        a = b
        b = Remainder
    endwhile
    return(a)
end EuclidGCD
```

Most iterations performed

Note that if $a < b$, then a and b get swapped after the first iteration, so no need to make this test.

Proposition. Euclid's algorithm in computing gcd(a,b) where $a \geq b$ makes at most $2n$ iterations where n is the number of binary (base 2) digits of a.

Proof. After one iteration a is replaced with b and b with r and after another iteration b is replaced with r. Thus, after two iterations a is replaced with r. But, the remainder r in binary has at least one fewer digits than a. Thus, after every other iteration a is reduced by at least one binary digit. It follows that the number of iterations performed in computing gcd(a,b) for any a and b where $a \geq b$ is at most twice the number of binary digits of a.

For what input does Euclid's algorithm take the most time?

Answer: $a = \text{fib}(n)$, $b = \text{fib}(n+1)$, where $\text{fib}(n)$ is the n^{th} Fibonacci number.

Fibonacci numbers: 0 1 1 2 3 5 8 13 21 34 55 …

gcd(34,55) = gcd(55,34) = gcd(34,21) = gcd(21,13) = gcd(13,8) = gcd(8,5) = gcd(5,3) = gcd(3,2) = gcd(2,1) = gcd(1,1) = gcd(1,0) = 1

If $a < b$ then a and b get swapped in the first iteration. So assume that $a \geq b$. Let a_{n-i} be the value of a after i iterations, i.e., i recursive calls. Then,

$$a_n = a,\ a_{n-1} = b,\ a_{n-2} = r.$$

Now $a = bq + r$, *so we have*

$$a_n = qa_{n-1} + a_{n-2}.$$

Euclid's gcd algorithm will be slowest when quotient $q = 1$, i.e., remainder is as large as possible. This yields,

$$a_n = a_{n-1} + a_{n-2},$$

which is the recurrence relation for Fibonacci.

Applications of gcd

- Lowest Common Multiple
- Fraction in Lowest Form
- Cryptographic algorithms such as RSA

Lowest Common Multiple (lcm)

$\operatorname{lcm}(a, b)$ is the smallest multiple of both a and b.

Proposition. $\operatorname{lcm}(a, b) = \frac{ab}{\gcd(a,b)}$.

Example. $a = 585, b = 1035$

$$a \times b = 585 \times 1035 = 605475$$

$$\gcd(585{,}1035) = 45$$

$$\begin{aligned} \operatorname{lcm}(585{,}1035) &= 23 \times 585 = 13 \times 1035 \\ &= 13455 \\ &= \frac{605475}{45} \end{aligned}$$

Fraction in Simplest Form

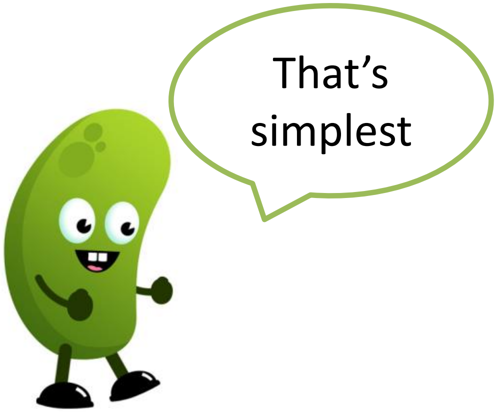
$$\frac{a}{b} \text{ in simplest form is } \frac{a/\gcd(a,b)}{b/\gcd(a,b)}$$

Example. $a = 585, b = 1035$

$$\frac{a}{b} = \frac{585}{1035}$$

$$\frac{a/\gcd(a,b)}{b/\gcd(a,b)} = \frac{585/\gcd(585,1035)}{1035/\gcd(585,1035)}$$

$$= \frac{585/45}{1035/45} = \frac{13}{23}$$



Extended Euclid's algorithm

We now design an extension of Euclid's GCD algorithm that computes integers g, s, t, where $g = \gcd(a,b)$ and

$$g = sa + tb.$$

This algorithm has important applications including use in the design of the **RSA public-key cryptosystem** which is used extensively for encryption and digital signatures on the Internet.

Example of Extended Euclid's algorithms for $a = 6700, b = 3000$

$6700 = 2 \times 3000 + 700 \Rightarrow 700 = 6700 - 2 \times 3000$

$3000 = 4 \times 700 + 200 \Rightarrow 200 = 3000 - 4 \times 700$

$700 = 3 \times 200 + 100 \Rightarrow 100 = 700 - 3 \times 200$

$200 = 2 \times 100 + 0$

$100 = 700 - 3 \times 200$. Substituting from above we have

$100 = 700 - 3 \times (3000 - 4 \times 700)$. Simplifying we have

$100 = 13 \times 700 - 3 \times 3000$. Substituting from above we have

$100 = 13 \times (6700 - 2 \times 3000) - 3 \times 3000$. Simplifying we have

$100 = 13 \times 6700 - 29 \times 3000$

We have solved $g = \gcd(a, b) = sa + tb$ for $a = 6700, b = 3000$ obtaining $g = 100, s = 13, t = -29$.

PSN. Solve $g = \gcd(a, b) = sa + tb$ for $a = 1035, b = 585$

Extended Euclid GCD

Suppose we have $g = s'b + t'r$

By definition $a = bq + r$, where q is the quotient, so that

$$r = a - bq$$

Substituting this value for r into $g = s'b + t'r$ we obtain

$$g = s'b + t'\,(a - bq) = t'a + (s' - t'q)b$$

Assigning $s = t'$ and $t = s' - t'q$, we have the desired result

$$g = sa + tb$$

Extended Euclid GCD Algorithm

```
function ExtEuclidGCD(a,b,g,s,t)
    Input: a, b (nonnegative integers)
    Output: return g = gcd(a,b) and integers s and t such that sa + tb = g
      if (b == 0)    //BOOTSTRAP CONDITION
            g = a
            s = 1
            t = 0
      else
            r = a mod b
            q = a/b
            ExtEuclidGCD(b,r,g,s,t)    //recursive call
            stemp = s
            s = t
            t = stemp – t*q
end ExtEuclidGCD
```

Historically Bad Joke

Humphrey Bogart is sitting in his bar in Casablanca, enjoying the sublime beauty of geometry…

He raises his glass and says,

"Here's looking at Euclid."