Chapter 11 Number-Theoretic Algorithms

Jason Pearson and Sam Demorest

April 5, 2015

Overview

Composite and Prime Numbers

Composite Numbers have a divisor other than itself and one.

For example $4|20$ means that $20 = 5 * 4$

The divisors of 12 are 1,2,3,4,6 and 12

Prime numbers have no divisors but 1 and itself

First 10 Primes

2, 3, 5, 7, 11, 13, 17, 19, 23, 29

Greatest Common Divisor

If h|m and h|n then h is called a common divisor
A common divisor is a number that is a factor of both numbers
The greatest common divisor is the largest factor for both numbers
This is denoted gcd(n,m)
For example gcd(12,15) $= 3$

For any two integers n and m where m $\neq 0$ the quotient is given by
$q = \lfloor n/m \rfloor$
The remainder r of dividing n by m is given by
$r = n - qm$

Greatest Common Divisor (cont)

Let n and m be integers, not both 0 and let
$d = \min \{ in + jm \text{ such that } i,j \in Z \text{ and } in + jm > 0\}$
That is, d is the smallest positive linear combination of n and m
For example we know $\gcd(12, 8) = 4$,
the smallest linear combination is
$4 = 3(12) + (-4)8$

Now suppose we have $n \geq 0$ and $m > 0$ and $r = n \bmod(m)$ then
$\gcd(n, m) = \gcd(m, r)$
so

$$
\begin{aligned}
\gcd(64, 24) &= \gcd(24, 16) \\
&= \gcd(16, 8) \\
&= \gcd(8, 0) \\
&= 8
\end{aligned}
$$

Least Common Multiple

For n and m where they are both nonzero, the least common multiple is denoted lcm(n,m)
For example lcm(6,9) $= 18$ because $6|18$ and $9\ |18$
The lcm(n,m) is a product of primes that are common to m and n, where the power of each prime in the product is the larger of its orders in n and m
So $12 = 2^2 3^1$ and $45 = 3^2 5^1$
so lcm(12,45) $= 2^2 3^2 5^1 = 180$

Prime Factorization

Two integers are relatively prime because the gcd of them is 1
For example $\gcd(12, 25) = 1$ so they are relatively prime
If h and m are relatively prime and h divides nm, then h divides m.
That is $\gcd(h,m) = 1$ and $h|nm$ implies $h|n$

Prime Factorization (cont)

Every integer $X > 1$ can be written as a unique product of primes
That is $X = p_1^{k_1} * p_2^{k_2} * \ldots * p_n^{k_n}$
Where $p_1 < p_2 < \ldots p_n$ and this representation of n is unique
Example being $22{,}275 = 3^4 * 5^2 * 11$

To solve gcd(3,185,325, 7,276,500) we know
$3{,}185{,}325 = 3^4 5^2 11^2 13^1$
$7{,}276{,}500 = 2^2 3^3 5^3 7^2 11^1$
We then take the common divisors and take the lower power to create the gcd
so $\gcd(3{,}185{,}325,\ 7{,}276{,}500) = 3^3 5^2 11^1 = 7{,}425$

Euclid's Algorithm

Euclid's Algorithm gives us a straight forward way to find the gcd of two numbers

```
int gcd(int n, int m)
{
    if(m == 0)
        return n;

    else
        return gcd(m, n mod m);
}
```

Extension to Euclid's Algorithm

```
void Euclid (int n, int m, int gcd, int i, int j){
    if (m == 0) {
        gcd = n; i = 1; j = 0;
    }
    else {
        int iprime, jprime, gcdprime;
        Euclid (m, n mod m, gcdprime, iprime, jprime);
        gcd = gcdprime;
        i = jprime;
        j = iprime -⌊n/m⌋ jprime ;
    }
}
```

Time Complexity

Basic Operation: one bit manipulation in the computation of the remainder.
Input Size: The number of bits it takes to encode the input
s = $\lceil lgn \rceil + 1$
t = $\lceil lgm \rceil + 1$
We will analyze when $1 \leq m < n$. If m = n then there will be no recursive calls
If m > n, the first recursive call will be gcd(m,n) instead to keep the first element larger
It has been shown that the worst case calls $Wcalls(s, t) \in \theta(t)$

Time Complexity (cont)

For each recursive call we compute one remainder which in the worst case number of bit manipulations is bounded from above by $c\lceil(1 + lgq)lgm\rceil$ where q is the quotient of dividing n by m and c is a constant

For r > 0 the worst case number of bit manipulations is $c\lceil(1 + lgn)lgm - lgm \times lgr\rceil$

Now with worst case number of bit manipulations considered we know $q = (n - r)/m$ and $1 \leq r < m$

$$
\begin{aligned}
1 + lgq &= 1 + lg((n - r)/m) \\
&\leq 1 + lg((n - r)/r) \\
&\leq 1 + lg(n - r) \\
&\leq 1 + lgn - lgr
\end{aligned}
$$

Time Complexity (cont)

This last equality combined with the worst case for number of bit manipulations and recursive calls results is bounded from above by $= c\lceil lgn \times lgm + lgm + lgr + lg(mmodr) + \ldots \rceil$ Since $n > m > r > mmodr > \ldots$ where the dots denote the remaining terms.
We conclude $W(s, t) \in O(st)$

Why Use the Other Algorithm?

This other algorithm will give us integers i and j as well

So, gcd = in + jm

For Example Euclid(42, 30, gcd, i, j) outputs

gcd = 6, i = -2 and j = 3

6 = -2(42) + 3(30)

Proof Extended Algorithm

Induction Base: In the last recursive call m = 0, which means gcd(n, m) = n
Since the values of i and j are assigned values 1 and 0 respectively we have
$in + jm = 1n + 0m = n = gcd\,(n, m)$

Induction Hypothesis: Assume in the kth recursive call the values determined for i and j are such that
gcd(n,m) = in + mj
Then the values returned by that call for i' and j' are values such that
gcd(m, n mod m) = i'm + j'n mod m

Proof Extended Algorithm (cont)

Induction Step: We have for the (k - 1)st call that

$$
\begin{aligned}
in + mj &= j'n + (i' - \lfloor n/m \rfloor\, j')m \\
&= i'm + j'(n - \lfloor n/m \rfloor m) \\
&= i'm + j'n \bmod m \\
&= \gcd(m, n \bmod m) \\
&= \gcd(n,m)
\end{aligned}
$$

The second to last equality is due to the induction hypothesis

Group Theory

A closed binary operation * on a set S is a rule for combining two elements of S to yield another element of S.
This operation must be associative
Must have an identity element for each element in S
For each element in S there must exist an inverse for that element
For example with integers $\in$ Z with addition constitute a group.
The identity element is 0 and the inverse of a is -a
A group is said to be finite if S contains a finite number of elements
A group is said to be commutative (or abelian) if for all a, b $\in$ S
$a * b = b * a$

Congruency Modulo n

Let m and k be integers and n be a positive integer. If n|(m - k) we say m is congruent to k modulo n, and this is written by
$m \equiv k \mathrm{mod} n$
For Example
Since 5|(33 - 18), $33 \equiv 18 \mathrm{mod} 5$

The integers 2, 5, 9 are pairwise prime and

$$184 \equiv 4 \mathrm{mod} 2$$
$$184 \equiv 4 \mathrm{mod} 5$$
$$184 \equiv 4 \mathrm{mod} 9$$

Since $2 * 5 * 9 = 90$ this implies $184 \equiv 4 \mathrm{mod} 90$
Congruency modulo n is an equivalence relation on the set of all integers.

Equivalence Class Modulo n Containing m

The set of all integers congruent to m modulo n is called the equivalence class modulo n containing m

For example the equivalence class modulo 5 containing 13 is $\{\ldots, -7, -2, 3, 8, 13, 18, 23, 28, 33, \ldots\}$

Equivalence classes modulo n containing m are represented by $[m]_n$

So for our previous example we would represent it by $[3]_5$

The set of all equivalence classes modulo n is denoted $\mathbf{Z}_n$

$\mathbf{Z}_n = \{[0]_n, [1]_n, ..., [n-1]_n\}$

Example of Addition using $\mathbf{Z}_5 = \{[0]_5, [1]_5, [2]_5, [3]_5, [4]_5\}$

$[2]_5 + [4]_5 = [6]_5 = [1]_5$

For every positive integer n, $(\mathbf{Z}_n, +)$ is a finite commutative group

Every element has an additive inverse so we know the identity element is $[0]_n$

Equivalence Class Modulo n Containing m (cont)

Using $\mathbf{Z}_5 = \{[0]_5, [1]_5, [2]_5, [3]_5, [4]_5\}$
For multiplication $[2]_5 * [4]_5 = [8]_5 = [3]_5$
This isn't always the case though because not every element in $(\mathbf{Z}_n,)$ has a multiplicative inverse
For example we consider $\mathbf{Z}_9$
Suppose $[6]_9$ has a multiplicative inverse $[k]_9$. Then
$[6]_9[k]_9 = [6k]_9 = [1]_9$
Which means there exists an integer i such that
1 = 6k + 9i which implies gcd(6,9) = 1 which is not the case

Equivalence Class Modulo n Containing m (cont)

This will work if we only include the relatively prime numbers for example

$z_9^* = \left\{[1]_1, [2]_9, [4]_9, [5]_q, [7]_9, [8]_9\right\}$

Using $(z_9^*, \times)$ we have the following multiplicative inverses

$[1]_9 * [1]_9 = [1]_9$

$[2]_9 * [5]_9 = [10]_9 = [1]_9$

$[4]_9 * [7]_9 = [28]_9 = [1]_9$

$[8]_9 * [8]_9 = [64]_9 = [1]_9$

The number of elements in z_n^* is given by Euler's totient function

$\varphi(n) = n \prod_{p:p|n} \left(1 - \frac{1}{p}\right)$ For example

$\varphi(60) = 60 \prod_{p:p|60} \left(1 - \frac{1}{p}\right) = 60 \left(1 - \frac{1}{2}\right)\left(1 - \frac{1}{3}\right)\left(1 - \frac{1}{5}\right) = 16$

If the number is prime the totient function is simply $\varphi(p) = p - 1$

SubGroups

If G = (S, $\times$) is a group, S' $\subseteq$ S, and G' = (S', $\times$) is a group then G' is said to be a **subgroup** of G. It is a **proper subgroup** if S' $\neq$ S
For E, the set of even integers and Z the set of integers.
(E, $+$) is a proper subgroup of (Z, $+$)
|S| denotes the number of elements in S it has been shown |S'| | |S|

Suppose we have a finite group G = (S, $\times$) and a $\in$ S.
$\langle a \rangle = \{a^k$ such that k is a positive integer $\}$
Clearly $\langle a \rangle$ is closed under $\times$. So, $(\langle a \rangle, \times)$ is a subgroup of G.
This new group is called the subgroup generated by a.
If the subgroup generated by a is G we call a a generator of G
For example $(\mathbf{Z}_6, +)$. We have

$$\begin{aligned}\langle [2]_6 \rangle &= \{[2]_6, [2]_6 + [2]_6, [2]_6 + [2]_6 + [2]_6, \ldots\} \\ &= \{[2]_6, [4]_6, [0]_6, [2]_6, \ldots\}\end{aligned}$$

SubGroups (cont)

When generating a subgroup we can stop once we reach the identity element

ord(a) is the least positive integer t such that $a^t = e$ where e is the identity element

Consider the group $(\mathbf{Z}_6, +)$. We have

$\langle [3]_6 \rangle = \{[3]_6, [3]_6 + [3]_6\} = \{[3]_6, [0]_6\}$

and

$\langle [2]_6 \rangle = \{[2]_6, [2]_6 + [2]_6, [2]_6 + [2]_6 + [2]_6\} = \{[2]_6, [4]_6, [0]_6\}$

Clearly

$\langle [1]_6 \rangle = \mathbf{Z}_6$

SubGroups (cont)

Euler proved for any integer n > 1 for all $[m]_n \in \mathbf{Z}_n$
$(|m|_n)^{\varphi(n)} = |1|_n$
Consider the group $(\mathbf{Z}_{20}, \times)$ We have that
$\varphi\left(20\right) = 20 \prod_{p:p|20} \left(1 - \frac{1}{p}\right) = 20 \left(1 - \frac{1}{5}\right) \left(1 - \frac{1}{2}\right) = 8$
and $([3]_{20})^8 = [6561]_{20} = [1]_{20}$

Also Fermat has shown that if p is prime then for all $[m]_p \in \mathbf{Z}_p$
$\left([m]_p\right)^{p-1} = [1]_p$
For example group $(\mathbf{Z}_7, \times)$. We have that
$([2]_7)^{7-1} = [64]_7 = [1]_7$

Pre Solving Modular Linear Equations

The modular equation
$[m]_n x = [k]_n$
for X, where X is an equivalence class modulo n, and m, n $>$ 0.
$\langle [6] \rangle_8 = \{[0]_8, [6]_8, [4]_8, [2]_8\}$
the equation
$[6]_8 x = [k]_8$
has a solution if and only if $[k]_8$ is $[0]_8, [6]_8, [4]_8, or [2]_8$ For example, solutions to
$[6]_8 x = [4]_8$
are x $=$ $[2]_8$ and x $=$ $[6]_8$

Pre Solving Modular Linear Equations (cont)

Consider the group $(\mathbf{Z}_n, +)$ For any $[m]_n \in \mathbf{Z}_n$ we have that $\langle [m]_n \rangle = \langle [d]_n \rangle = \{[0]_n, [d]_n, [2d]_n, \ldots, [(nd-1)\, d]_n\}$ where d = gcd(n, m). This means $ord([m]_n) = |\langle [m]_n \rangle| = \frac{n}{d}$

The equation $[m]_n\, x = [k]_d$
has a solution if and only if d | k where d = gcd(n,m).
Furthermore if the equation has a solution it has d solutions.
There is only a solution for every equivalence class $[k]_n$ if and only if gcd(n,m) = 1

Pre Solving Modular Linear Equations Examples

Using the group $(\mathbf{Z}_8, +)$. Since gcd(8,5) $= 1$
So, $[5]_8 x = [k]_8$ has exactly one solution when solving for any k that is a member of $\langle [5] \rangle_8$. When k $= 3$ we know that x $= [7]_8$
Using the same group we use 6 instead so gcd(8,6) $= 2$
So, $[6]_8 x = [k]_8$ has exactly two solutions when solving for any k that is a member of $\langle [6] \rangle_8$. When k $= 4$ we know that x $= [6]_8$ and x $= [2]_8$

Solving Modular Linear Equations

Let d = gcd(n,m) and let i and j be integers such that d = in + jm
Suppose further d | k Then the equation $[m]_n x = [k]_n$ has solution
$x = \left[\frac{jk}{d}\right]_n$ For example, consider $[6]_8 x = [4]_8$ we have gcd(8,6) = 2
2 = (1) 8 + (-1) 6 and 2 | 4 so it must have the solution
$x = \left[\frac{-1(4)}{2}\right]_8 = [-2]_8 = [6]_8$ This is only one solution though to solve the other we use the equation
$\left[j + \frac{wn}{d}\right]_n$ for $w = 0, 1, \ldots, d-1$
So for the other solution we have
$\left[6 + \frac{1(8)}{2}\right]_8 = [10]_8 = [2]_8$

Psuedocode For Solving Modular Linear Equations

void solvelinear (int n, int m, int k)
{
index l;
int i, j, d;
Euclid(n,m,d,i,j);
if (d|k) {
for(w = 0; w $\leq$ d - 1; w++) {
cout << $\left[\frac{jk}{d} + \frac{wn}{d}\right]_n$;
}
}
}

Time Complexity Analysis

The input size in our linear solver is the number of bits it takes to encode input

s $= \lceil lgn \rceil + 1$

t $= \lceil lgm \rceil + 1$

u $= \lceil lgk \rceil + 1$

The time complexity for Euclid's Algorithm is O(st), plus the time complexity for the for-w loop.

Since d can be as large as m or n, this time complexity is worst-case exponential in terms of input size.