CSE 311 Foundations of Computing I

Lecture 12

Modular Arithmetic and Applications

Autumn 2012

Announcements

- Reading assignments
  - Today and Friday:
    - 4.1-4.3 7th Edition
    - 3.5, 3.6 6th Edition
    - 2.5, 2.6 up to p. 191 5th Edition

Divisibility

Integers a, b, with a ≠ 0, we say that a *divides* b is there is an integer k such that b = ak. The notation a | b denotes a divides b.

Let *a* be an integer and *d* a positive integer. Then there are *unique* integers *q* and *r*, with 0 ≤ *r* < *d*, such that *a* = *dq* + *r*.

q = *a* **div** *d* *r* = *a* **mod** *d*

Modular Arithmetic

Let a and b be integers, and m be a positive integer. We say a *is congruent to b modulo m* if m divides a – b. We use the notation a ≡ b (mod m) to indicate that a is congruent to b modulo m.

Modular arithmetic

Let a and b be integers, and let m be a positive integer. Then a ≡ b (mod m) if and only if a mod m = b mod m.

Modular arithmetic

Let m be a positive integer. If a ≡ b (mod m) and c ≡ d (mod m), then

- a + c ≡ b + d (mod m) and
- ac ≡ bd (mod m)

Example

Let n be an integer, prove that $n^2 \equiv 0 \pmod 4$ or $n^2 \equiv 1 \pmod 4$

n-bit Unsigned Integer Representation

- Represent integer x as sum of powers of 2:
  If $x = \sum_{i=0}^{n-1} b_i 2^i$ where each $b_i \in \{0,1\}$
  then representation is $b_{n-1}...b_2\, b_1\, b_0$

  99 = 64 + 32 + 2 + 1
  18 = 16 + 2

- For n = 8:
  99: 0110 0011
  18: 0001 0010

Signed integer representation

n-bit signed integers
Suppose $-2^{n-1} < x < 2^{n-1}$
First bit as the sign, n-1 bits for the value

99 = 64 + 32 + 2 + 1
18 = 16 + 2

For n = 8:
99: 0110 0011
-18: 1001 0010

Any problems with this representation?

Two's complement representation

n bit signed integers, first bit will still be the sign bit
Suppose $0 \le x < 2^{n-1}$, x is represented by the binary representation of x
Suppose $0 < x \le 2^{n-1}$, -x is represented by the binary representation of 2^n-x

Key property: Two's complement representation of any number y is equivalent to y mod 2^n so arithmetic works mod 2^n

99 = 64 + 32 + 2 + 1
18 = 16 + 2

For n = 8:
99: 0110 0011
-18: 1110 1110

Signed vs Two's complement

-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7
1111	1110	1101	1100	1011	1010	1001	0000	0001	0010	0011	0100	0101	0110	0111

Signed

-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7
1000	1001	1010	1011	1100	1101	1110	1111	0000	0001	0010	0011	0100	0101	0110	0111

Two's complement

Two's complement representation

- Suppose $0 < x \le 2^{n-1}$, -x is represented by the binary representation of 2^n-x
- To compute this: Flip the bits of x then add 1:
  - All 1's string is 2^n-1 so
    - Flip the bits of x $\equiv$ replace x by 2^n-1-x

Basic applications of mod

- Hashing
- Pseudo random number generation
- Simple cipher

Hashing

- Map values from a large domain, 0…M-1 in a much smaller domain, 0…n-1
- Index lookup
- Test for equality
- Hash(x) = x mod p
- Often want the hash function to depend on all of the bits of the data
  – Collision management

Pseudo Random number generation

- Linear Congruential method

$$x_{n+1} = (a\, x_n + c) \bmod m$$

Simple cipher

- Caesar cipher, A = 1, B = 2, . . .
  – HELLO WORLD
- Shift cipher
  – f(p) = (p + k) mod 26
  – $f^{-1}(p) = (p - k) \bmod 26$
- f(p) = (ap + b) mod 26

Modular Exponentiation

x	1	2	3	4	5	6
1	1	2	3	4	5	6
2	2	4	6	1	3	5
3	3	6	2	5	1	4
4	4	1	5	2	6	3
5	5	3	1	6	4	2
6	6	5	4	3	2	1

a	a^1	a^2	a^3	a^4	a^5	a^6
1						
2						
3						
4						
5						
6						

Exponentiation

- Compute 78365^{81453}
- Compute 78365^{81453} mod 104729

Fast exponentiation

```
int FastExp(int x, int n){
    long v = (long) x;
    int exp = 1;
    for (int i = 1; i <= n; i++){
        v = (v * v) % modulus;
        exp = exp + exp;
        Console.WriteLine("i : " + i + ", exp : " + exp + ", v : " + v );
    }
    return (int)v;
}
```


Program Trace

```
i : 1,   exp : 2,       v : 82915
i : 2,   exp : 4,       v : 95592
i : 3,   exp : 8,       v : 70252
i : 4,   exp : 16,      v : 26992
i : 5,   exp : 32,      v : 74970
i : 6,   exp : 64,      v : 71358
i : 7,   exp : 128,     v : 20594
i : 8,   exp : 256,     v : 10143
i : 9,   exp : 512,     v : 61355
i : 10,  exp : 1024,    v : 68404
i : 11,  exp : 2048,    v :  4207
i : 12,  exp : 4096,    v : 75698
i : 13,  exp : 8192,    v : 56154
i : 14,  exp : 16384,   v : 83314
i : 15,  exp : 32768,   v : 99519
i : 16,  exp : 65536,   v : 29057
```


Fast exponentiation algorithm

- What if the exponent is not a power of two?

$81453 = 2^{16} + 2^{13} + 2^{12} + 2^{11} + 2^{10} + 2^{9} + 2^{5} + 2^{3} + 2^{2} + 2^{0}$

The fast exponentiation algorithm computes a^n mod p in time O(log n)

Primality

An integer *p* greater than 1 is called *prime* if the only positive factors of *p* are 1 and *p*.

A positive integer that is greater than 1 and is not prime is called composite.

Fundamental Theorem of Arithmetic

Every positive integer greater than 1 has a unique prime factorization

48 = 2 • 2 • 2 • 2 • 3
591 = 3 • 197
45,523 = 45,523
321,950 = 2 • 5 • 5 • 47 • 137
1,234,567,890 = 2 • 3 • 3 • 5 • 3,607 • 3,803

Factorization

- If n is composite, it has a factor of size at most sqrt(n)

Euclid's theorem

- There are an infinite number of primes.
- Proof by contradiction:

  Suppose there are a finite number of primes: p_1, p_2, . . ., p_n