CSE 311 Foundations of Computing I

Lecture 11

Modular Exponentiation and Primes

Autumn 2011

Announcements

- Reading assignments
  - Today and Monday:
    - 4.3 — 7^{th} Edition
    - 3.5, 3.6 — 6^{th} Edition
    - 2.5, 2.6 up to p. 191 — 5^{th} Edition
  - Wednesday
    - Start on induction
- Homework 4
  - Available now (posted Wednesday night)

Highlights from last lecture

- Introduction of modular arithmetic

What is the difference between r = a mod d and r ≡ a (mod d) ?

- Fumbling with the projector and whiteboard (morning lecture)

Division Theorem

Let a be an integer and d a positive integer. Then there are *unique* integers q and r, with $0 \leq r < d$, 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.

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.

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)

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.

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

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 $78365^{81453} \bmod 104729$

Fast exponentiation

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

Program Trace

i : 1, m : 2, v : 82915
i : 2, m : 4, v : 95592
i : 3, m : 8, v : 70252
i : 4, m : 16, v : 26992
i : 5, m : 32, v : 74970
i : 6, m : 64, v : 71358
i : 7, m : 128, v : 20594
i : 8, m : 256, v : 10143
i : 9, m : 512, v : 61355
i : 10, m : 1024, v : 68404
i : 11, m : 2048, v : 4207
i : 12, m : 4096, v : 75698
i : 13, m : 8192, v : 56154
i : 14, m : 16384, v : 83314
i : 15, m : 32768, v : 99519
i : 16, m : 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: $p_1, p_2, \ldots p_n$

Distribution of Primes

2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89
97 101 103 107 109 113 127 131 137 139 149 151 157 163 167 173
179 181 191 193 197 199 211 223 227 229 233 239 241 251 257 263
269 271 277 281 283 293 307 311 313 317 331 337 347 349 353 359

- If you pick a random number n in the range [x, 2x], what is the chance that n is prime?

Famous Algorithmic Problems

- Primality Testing:
  - Given an integer n, determine if n is prime
- Factoring
  - Given an integer n, determine the prime factorization of n

Primality Testing

- Is the following 200 digit number prime:

4099240841609602817976123253258752540290928509908622013340
39205254095520835286062154399159482608757188937978247351
18621138192569490840098061133066650255608065609253901288801
3020354418848781879442190033