CSE 321 Discrete Structures

Winter 2008

Lecture 8

Number Theory: Modular Arithmetic

Announcements

- Readings
  - Today:
    - 3.4 (5th Edition: 2.4)
  - Monday and Wednesday:
    - 3.5, 3.6, 3.7 (5th Edition: 2.5, 2.6)

Number Theory (and applications to computing)

- Branch of Mathematics with direct relevance to computing
- Many significant applications
  - Cryptography
  - Hashing
  - Security
- Important tool set

Modular Arithmetic

- Arithmetic over a finite domain
- In computing, almost all computations are over a finite domain

What are the values computed?

```
public void Test1() {
        byte x = 250;
        byte y = 20;
        byte z = (byte) (x + y);
        Console.WriteLine(z);
}
```

```
public void Test2() {
        sbyte x = 120;
        sbyte y = 20;
        sbyte z = (sbyte) (x + y);
        Console.WriteLine(z);
}
```

Arithmetic mod 7

- $a +_7 b = (a + b) \bmod 7$
- $a \times_7 b = (a \times b) \bmod 7$

+	0	1	2	3	4	5	6
0							
1							
2							
3							
4							
5							
6							

x	0	1	2	3	4	5	6
0							
1							
2							
3							
4							
5							
6							

Group Theory

- A group G=(S, •) is a set S with a binary operator • that is "well behaved":
  - Closed under •
  - Associative: a • (b • c) = (a • b) • c
  - Has an identity
  - Each element has an inverse
- A group is commutative if the • operator also satisfies a• b = b • a

Groups, mod 7

- {0,1,2,3,4,5,6} is a group under $+_7$
- {1,2,3,4,5,6} is a group under $\times_7$

Multiplicative Inverses

- Euclid's theorem: if x and y are relatively prime, then there exists integers s, t, such that:

$$sx + ty = 1$$

- Prove $a \in \{1, 2, 3, 4, 5, 6\}$ has a multiplicative inverse under $\times_7$

Generalizations

- $(\{0,\ldots, n\text{-}1\}, +_n)$ forms a group for all positive integers n
- $(\{1,\ldots, n\text{-}1\}, \times_n)$ is a group if and only if n is prime

Basic applications

- Hashing: store keys in a large domain 0…M-1 in a much smaller domain 0…n-1

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

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

Data Permutations

- Caesar cipher, a = 1, b = 2, . . .
  - HELLO WORLD
- Shift cipher
  - $f(x) = (x + k) \bmod n$
  - $f^{-1}(x) = (x - k) \bmod n$
- Affine cipher
  - $f(x) = (ax + b) \bmod n$
  - $f^{-1}(x) = (a^{-1}(x-b)) \bmod n$

a	b	c	d	e	f	g
1	2	3	4	5	6	7
5	6	7	1	2	3	4
5	3	1	6	4	2	7

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						

Fermat's Little Theorem

- If p is prime, $0 < a \leq p-1$, $a^{p-1} \equiv 1 \pmod p$
- Group theory
  - Index of x, smallest $i > 0$ such that $x^i = 1$
  - The index of x divides the order of the group

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;
}
```