Fall 2015
Lecture 12: Primes, GCD, applications

I have nothing to do, so I'm trying to calculate the prime factors of the time each minute before it changes.

It was easy when I started at 1:00, but with each hour the number gets bigger.

I wonder how long I can keep up.

Hey! Think fast.

253 is 11 \times 23

WHAT?

I'm factoring the time.

I'm sleep.
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 $b_{n-1} \cdots b_2 b_1 b_0$

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

- For $n = 8$:

  99: 0110 0011
  18: 0001 0010

  \[ \n = 8 \]

  \[ \begin{array}{cccccccc}
  128 & 64 & 32 & 16 & 8 & 4 & 2 & 1 \\
  0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
  \end{array} \]
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 $n = 8$:

\[ 99 + (-18) = -117 = 11110101 \]

99: 0110 0011
-18: 1001 0010

Any problems with this representation? Yes
two's complement representation

n-bit signed integers, first bit will still be the sign bit

Suppose $0 \leq x < 2^{n-1}$,
- $x$ is represented by the binary representation of $x$

Suppose $0 \leq x \leq 2^{n-1}$,
- $-x$ 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

$99 \mod 2^8 = \frac{99}{2^8} = 81$

$-18 \mod 2^8 = \frac{-18}{2^8} = 81$

$238 = 128 + 64 + 32 + 8 + 4 + 2$

$256 - 18 = 238$
# sign-magnitude vs. two’s complement

<table>
<thead>
<tr>
<th>-7</th>
<th>-6</th>
<th>-5</th>
<th>-4</th>
<th>-3</th>
<th>-2</th>
<th>-1</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1111</td>
<td>1110</td>
<td>1101</td>
<td>1100</td>
<td>1011</td>
<td>1010</td>
<td>1001</td>
<td>0000</td>
<td>0001</td>
<td>0010</td>
<td>0011</td>
<td>0100</td>
<td>0101</td>
<td>0110</td>
<td>0111</td>
</tr>
</tbody>
</table>

**Sign-Magnitude**

<table>
<thead>
<tr>
<th>-8</th>
<th>-7</th>
<th>-6</th>
<th>-5</th>
<th>-4</th>
<th>-3</th>
<th>-2</th>
<th>-1</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>1001</td>
<td>1010</td>
<td>1011</td>
<td>1100</td>
<td>1101</td>
<td>1110</td>
<td>1111</td>
<td>0000</td>
<td>0001</td>
<td>0010</td>
<td>0011</td>
<td>0100</td>
<td>0101</td>
<td>0110</td>
<td>0111</td>
</tr>
</tbody>
</table>

**Two’s complement**
For $0 < x \leq 2^{n-1}$, $-x$ 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 $2^n - 1 - x$.

For example, consider $x = 18$:

- $x = 00010010$
- $2^{5} - 18 = 11110010$
- $-18 = 11101110$
- $2^{n-1} - x = 1\overline{X}_{n-2} \cdots \overline{X}_1 \overline{X}_0$. 
Theorem: A positive integer \( n \) is divisible by 3 if and only if the sum of its decimal digits is divisible by 3.

\[
\begin{align*}
n &= 138 \\
1 + 3 + 8 &= 12 \\
3 | 12 \\
138 &= 46 \cdot 3 \\
3 \cdot 138 &= 394 \quad \text{(corrected from 3142)}
\end{align*}
\]

\[
\begin{align*}
n &= d_k d_{k-1} \ldots \ d_0 \\
3 | n &\iff 3 | d_k \ldots \ d_0 \\
n &= d_k (10)^k + d_{k-1} (10)^{k-1} + \ldots + d_0 \cdot 1 \\
&\equiv d_k 1^k + d_{k-1} 1^{k-1} + \ldots + d_0 \cdot 1 \mod 3 \\
&\equiv d_k + d_{k-1} + \ldots + d_0 \mod 3 \\
n \mod 3 &= d_k + \ldots + d_0 \mod 3
\end{align*}
\]
basic applications of mod

- Hashing
- Pseudo random number generation
- Simple cipher
Scenario:
Map a small number of data values from a large domain \(\{0, 1, ..., M - 1\}\) into a small set of locations \(\{0, 1, ..., n - 1\}\) so one can quickly check if some value is present.
**Scenario:**

Map a small number of data values from a large domain \(\{0, 1, \ldots, M - 1\}\) into a small set of locations \(\{0, 1, \ldots, n - 1\}\) so one can quickly check if some value is present.

- \(\text{hash}(x) = x \mod p\) for \(p\) a prime close to \(n\)
  - or \(\text{hash}(x) = (ax + b) \mod p\)

- Depends on all of the bits of the data
  - helps avoid collisions due to similar values
  - need to manage them if they occur
pseudo-random number generation

Linear Congruential method:

\[ x_{n+1} = (a \cdot x_n + c) \mod m \]

Choose random \( x_0 \), \( a \), \( c \), \( m \) and produce a long sequence of \( x_n \)'s

Adv: Fast

Dis: Far from random

[good for some applications, really bad for many others]
simple ciphers

- **Caesar cipher**, $A = 1$, $B = 2$, \ldots
  - $HELLO$ $WORLD$
- **Shift cipher**
  - $f(p) = (p + k) \mod 26$
  - $f^{-1}(p) = (p - k) \mod 26$
- **More general**
  - $f(p) = (ap + b) \mod 26$
**modular exponentiation mod 7**

<table>
<thead>
<tr>
<th>x</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>a</th>
<th>a¹</th>
<th>a²</th>
<th>a³</th>
<th>a⁴</th>
<th>a⁵</th>
<th>a⁶</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
modular exponentiation mod 7

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>1</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>6</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>a¹</th>
<th>a²</th>
<th>a³</th>
<th>a⁴</th>
<th>a⁵</th>
<th>a⁶</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2</td>
<td>6</td>
<td>4</td>
<td>5</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
modular exponentiation mod 7

<table>
<thead>
<tr>
<th>x</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>1</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>6</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>a</th>
<th>a^1</th>
<th>a^2</th>
<th>a^3</th>
<th>a^4</th>
<th>a^5</th>
<th>a^6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2</td>
<td>6</td>
<td>4</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>1</td>
<td>6</td>
<td>1</td>
<td>6</td>
<td>1</td>
</tr>
</tbody>
</table>
Compute $78365^{81453}$

Compute $78365^{81453} \mod 104729$

Output is small
   need to keep intermediate results small

\[ a \equiv (a \mod m)^2 \mod m \]
\[ a^3 \equiv (a \mod m)^3 \mod m \]
\[ n_1 = a \mod m \]
\[ n_2 = a \cdot n_1 \mod m \]
\[ n_3 = a \cdot n_2 \mod m \]
\[ \vdots \]
\[ n_b = a \cdot n_{b-1} \mod m \]
\[ n_b = a^b \mod m \]
repeated squaring – small and fast

Since \( a \mod m \equiv a \mod m \) for any \( a \)

we have \( a^2 \mod m = (a \mod m)^2 \mod m \)
and \( a^4 \mod m = (a^2 \mod m)^2 \mod m \)
and \( a^8 \mod m = (a^4 \mod m)^2 \mod m \)
and \( a^{16} \mod m = (a^8 \mod m)^2 \mod m \)
and \( a^{32} \mod m = (a^{16} \mod m)^2 \mod m \)

Can compute \( a^k \mod m \) for \( k = 2^i \) in only \( i \) steps
public static long FastModExp(long base, long exponent, long modulus) {
    long result = 1;
    base = base % modulus;

    while (exponent > 0) {
        if ((exponent % 2) == 1) {
            result = (result * base) % modulus;
            exponent -= 1;
        }
        exponent /= 2;
        base = (base * base) % modulus;
        /* The last iteration of the loop will always be exponent = 1 */
        /* so, result will always be correct. */
    }
    return result;
}

be mod m = (b2)e/2 mod m, when e is even
be mod m = (b*(be−1 mod m) mod m)) mod m
Let $M = 104729$

$78365^{81453} \mod M$

$= ((78365 \mod M) \times (78365^{81452} \mod M)) \mod M$

$= (78365 \times ((78365^2 \mod M)^{81452/2} \mod M)) \mod M$

$= (78365 \times ((78852^{40726} \mod M)) \mod M$

$= (78365 \times ((78852^2 \mod M)^{20363} \mod M)) \mod M$

$= (78365 \times (86632^{20363} \mod M)) \mod M$

$= (78365 \times ((86632 \mod M) \times (86632^{20362} \mod M)) \mod M$

$= ...$

$= 45235$
Another way:

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

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

$$a^{81453} \mod m = (\ldots(((a^{2^{16}} \mod m \cdot a^{2^{13}} \mod m) \mod m \cdot a^{2^{12}} \mod m) \mod m \cdot a^{2^{11}} \mod m) \mod m \cdot a^{2^{10}} \mod m) \mod m \cdot a^{2^{9}} \mod m) \mod m \cdot a^{2^{5}} \mod m) \mod m \cdot a^{2^{3}} \mod m) \mod m \cdot a^{2^{2}} \mod m) \mod m \cdot a^{2^{0}} \mod m) \mod m$$

The fast exponentiation algorithm computes $a^n \mod m$ using $O(\log n)$ multiplications $\mod m$
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*. 
Every positive integer greater than 1 has a unique prime factorization

\begin{align*}
48 &= 2 \cdot 2 \cdot 2 \cdot 2 \cdot 3 \\
591 &= 3 \cdot 197 \\
45,523 &= 45,523 \\
321,950 &= 2 \cdot 5 \cdot 5 \cdot 47 \cdot 137 \\
1,234,567,890 &= 2 \cdot 3 \cdot 3 \cdot 5 \cdot 3,607 \cdot 3,803
\end{align*}
If $n$ is composite, it has a factor of size at most $\sqrt{n}$. 
There are an infinite number of primes.

Proof by contradiction:

Suppose that there are only a finite number of primes:

$p_1, p_2, \ldots, p_n$
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
Factor the following 232 digit number [RSA768]:

1230186684530117755130494958384962720772
8535695953347921973224521517264005072636
5751874520219978646938995647494277406384
5925192557326303453731548268507917026122
1429134616704292143116022212404792747377
94080665351419597459856902143413
123018668453011775513049495838496272077285356959533479
219732245215172640050726365751874520219978646938995647
494277406384592519255732630345373154826850791702612214
291346167042921431160222124047927473779408066535141959
7459856902143413

334780716989568987860441698482126908177047949837
13768568912431388982883793878131777117
43087737814467999489

3674604366679959042824463379866943643
43087642676032283815739666511792968
10270092798736308917
GCD(a, b):

Largest integer \( d \) such that \( d \mid a \) and \( d \mid b \)

- \( \text{GCD}(100, 125) = \)
- \( \text{GCD}(17, 49) = \)
- \( \text{GCD}(11, 66) = \)
- \( \text{GCD}(13, 0) = \)
- \( \text{GCD}(180, 252) = \)
gcd and factoring

\[ a = 2^3 \cdot 3 \cdot 5^2 \cdot 7 \cdot 11 = 46,200 \]

\[ b = 2 \cdot 3^2 \cdot 5^3 \cdot 7 \cdot 13 = 204,750 \]

\[ \text{GCD}(a, b) = 2^{\text{min}(3,1)} \cdot 3^{\text{min}(1,2)} \cdot 5^{\text{min}(2,3)} \cdot 7^{\text{min}(1,1)} \cdot 11^{\text{min}(1,0)} \cdot 13^{\text{min}(0,1)} \]

Factoring is expensive!
Can we compute \( \text{GCD}(a, b) \) without factoring?
If $a$ and $b$ are positive integers, then
$$\text{gcd}(a, b) = \text{gcd}(b, a \mod b)$$

Proof:
By definition $a = (a \div b) \cdot b + (a \mod b)$
If $d \mid a$ and $d \mid b$ then $d \mid (a \mod b)$.
If $d \mid b$ and $d \mid (a \mod b)$ then $d \mid a$. 

useful GCD fact
euclid’s algorithm

Repeatedly use the GCD fact to reduce numbers until you get $\text{GCD}(x, 0) = x$.

$\text{GCD}(660, 126)$
Euclid's Algorithm

\[ \text{GCD}(x, y) = \text{GCD}(y, x \mod y) \]

```c
int GCD(int a, int b){ /* a >= b, b > 0 */
    int tmp;
    while (b > 0) {
        tmp = a % b;
        a = b;
        b = tmp;
    }
    return a;
}
```

Example: \( \text{GCD}(660, 126) \)