Lecture 12: Primes, GCD

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.
Basic Applications of mod

• Hashing
• Pseudo random number generation
Hashing

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

- \( \text{hash}(x) = x \mod p \) 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
## 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></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>
Exponentiation

- Compute $78365^{81453}$

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

- Output is small
  - need to keep intermediate results small
Repeated Squaring – small and fast

Since \( a \mod m \equiv a \pmod{m} \) 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;
        }
        /* Note that exponent is definitely divisible by 2 here. */
        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;
}

\[ b^e \mod m = (b^2)^{e/2} \mod m, \text{ when } e \text{ is even} \]
\[ b^e \mod m = (b*(b^{e-1} \mod m) \mod m)) \mod m \]
78365^{81453} \mod M
= ((78365 \mod M) * (78365^{81452} \mod M)) \mod M
= (78365 * ((78365^2 \mod M)^{81452/2} \mod M)) \mod M
= (78365 * ((78852)^{40726} \mod M)) \mod M
= (78365 * ((78852^2 \mod M)^{20363} \mod M)) \mod M
= (78365 * (86632^{20363} \mod M)) \mod M
= (78365 * ((86632 \mod M)^* (86632^{20362} \mod M)) \mod M
= ...
= 45235
Fast Exponentiation Algorithm

Another way:

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

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

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

The fast exponentiation algorithm computes \( a^m \mod m \) using \( O(\log n) \) multiplications \( \mod m \)
An integer \( p \) greater than 1 is called \textit{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 \textit{composite}.
Every positive integer greater than 1 has a unique prime factorization

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

If \( n \) is composite, it has a (non-trivial) factor, \( f \), where \( f \leq \sqrt{n} \).

Let \( n \) be an arbitrary composite number. Suppose, for contradiction, that all of the factors of \( n \) are greater than \( \sqrt{n} \). Then, since \( n \) is composite, there are two factors, \( a \) and \( b \), such that \( 1 < a, b < n \) such that \( n = ab \).

Note that \( a, b > \sqrt{n} \) by assumption. So, \( n = ab > \sqrt{n}\sqrt{n} = n \), which is a contradiction. It follows that the original claim is true.
Euclid’s Theorem

There are an infinite number of primes.

Proof by contradiction:

Suppose for contradiction that there are $n$ primes for some natural number $n$. Call them $p_1 < p_2 < \ldots < p_n$. Consider $P = p_1 p_2 \ldots p_n$, and define $Q = P + 1$.

Case 1 ($Q$ is prime). Then, we’re done, because $Q$ is larger than any of the primes; so, it is a new prime.

Case 2 ($Q$ is composite). Then, there must be some prime $p | q$. Note that since $P$ divides every possible prime, $p | P$ as well. It follows that $p | (Q - P) \rightarrow p | ((P + 1) - P) \rightarrow p | 1$. This is impossible, because $p$ must be at least two.

Since both cases lead to a contradiction, the original claim is true.
Famous Algorithmic Problems

• **Primality Testing**
  – Given an integer $n$, determine if $n$ is prime

• **Factoring**
  – Given an integer $n$, determine the prime factorization of $n$
Factoring

Factor the following 232 digit number [RSA768]:

123018668453011775513049495838496272077
2853569595334792197322452151726400050726
365751874520219978646938995647494277406
384592519255732630345373154826850791702
612214291346167042921431160222124047927
4737794080665351419597459856902143413
Greatest Common Divisor

GCD(a, b):

Largest integer $d$ such that $d | a$ and $d | b$

- GCD(100, 125) =
- GCD(17, 49) =
- GCD(11, 66) =
- GCD(13, 0) =
- GCD(180, 252) =
GCD and Factoring

\[ a = 2^3 \cdot 3 \cdot 5^2 \cdot 7 \cdot 11 = 46,200 \]
\[ b = 2 \cdot 3^2 \cdot 5^3 \cdot 7 \cdot 13 = 204,750 \]

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

Factoring is expensive!
Can we compute \( \text{GCD}(a,b) \) without factoring?
Useful GCD Fact

If $a$ and $b$ are positive integers, then
\[ \gcd(a, b) = \gcd(b, \ a \mod b) \]

Proof:
Consider an arbitrary divisor, $d$, such that $d | a$ and $d | b$. Note that
$a = (a \div b)b + (a \mod b)$. By definition of $d | a$, we have $(a \div b)b + (a \mod b) = kd$.
Since $d | b$, we also have $b = jd$. Re-arranging, we see
\[ (a \mod b) = kd - (a \div b)b = d(k - (a \div b)j). \]
So, $d | (a \mod b)$.

Now, consider an arbitrary divisor, $d$, such that $d | b$ and $d | (a \mod b)$. It follows that $(a \mod b) = kd$. Adding $(a \div b)b$ to both sides gives
$a = (a \mod b) + (a \div b)b = kd + (a \div b)b = kd + (a \div b)jd = d(k + (a \div b)j)$.
So, $d | a$.

Since all the divisors of $a$ and $b$ are the same as the divisors of $b$ and $a \mod b$, it follows that the greatest divisor of each pair is the same as well.
Euclid’s Algorithm

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

\[ \text{gcd}(660,126) = \text{gcd}(126, 660 \text{ mod } 126) = \text{gcd}(126, 30) \]
\[ = \text{gcd}(30, 126 \text{ mod } 30) = \text{gcd}(30, 6) \]
\[ = \text{gcd}(6, 30 \text{ mod } 6) = \text{gcd}(6, 0) \]
\[ = 6 \]
Euclid’s Algorithm

GCD(x, y) = GCD(y, x mod y)

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

Example: GCD(660, 126)