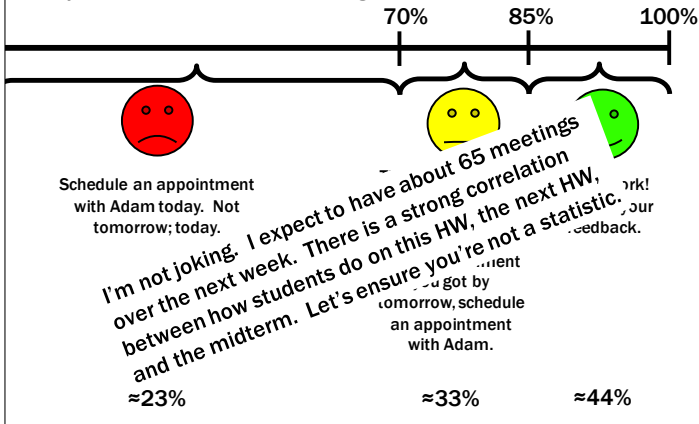




### HW 3 De-Brief

If your HW 3 score is in this range...



### HW 3 De-Brief

Okay, I got it. How do I schedule an appointment?

- Go to <http://meeting.countablethoughts.com>
- If I don't respond by Monday, then it probably didn't go through; so, e-mail me.

### HW 3 De-Brief

#### "How I Oops 311"

- Never read the feedback, or

### HW 3 De-Brief

#### "How I Oops 311"

- Never read the feedback, or
- Read the feedback but don't take it seriously, or

### HW 3 De-Brief

#### "How I Oops 311"

- Never read the feedback, or
- Read the feedback but don't take it seriously, or
- Read the feedback but convince yourself that "you get it now", or

### HW 3 De-Brief

#### "How I Oops 311"

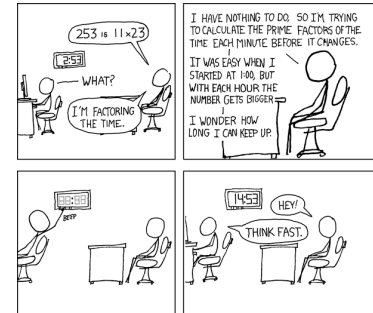
- Never read the feedback, or
- Read the feedback but don't take it seriously, or
- Read the feedback but convince yourself that "you get it now", or
- Read the feedback, talk to a TA, but don't apply what you've learned to future HWs, or...

## HW 3 De-Brief

How smart you are and your grade are not the same thing.

## CSE 311: Foundations of Computing

### Lecture 12: Primes, GCD



### Example

Let  $n$  be an integer.  
Prove that  $n^2 \equiv 0 \pmod{4}$  or  $n^2 \equiv 1 \pmod{4}$

Case 1 ( $n$  is even):

Let's start by looking a small example:

$$\begin{aligned} 0^2 &= 0 \equiv 0 \pmod{4} \\ 1^2 &= 1 \equiv 1 \pmod{4} \\ 2^2 &= 4 \equiv 0 \pmod{4} \\ 3^2 &= 9 \equiv 1 \pmod{4} \\ 4^2 &= 16 \equiv 0 \pmod{4} \end{aligned}$$

It looks like

$$\begin{aligned} n \equiv 0 \pmod{2} &\rightarrow n^2 \equiv 0 \pmod{4}, \text{ and} \\ n \equiv 1 \pmod{2} &\rightarrow n^2 \equiv 1 \pmod{4}. \end{aligned}$$

Case 2 ( $n$  is odd):

Suppose  $n \equiv 1 \pmod{2}$ .  
So,  $2 \mid n-1$ . So,  $n-1 = 2k$ .  
 $n^2 = (2k+1)^2 = 4k^2 + 4k + 1 = 4k^2 + 4k + 1$

### Example

Let  $n$  be an integer.  
Prove that  $n^2 \equiv 0 \pmod{4}$  or  $n^2 \equiv 1 \pmod{4}$

Case 1 ( $n$  is even):

Let's start by looking a small example:

$$\begin{aligned} \text{Suppose } n &\equiv 0 \pmod{2}. \\ \text{Then, } n &= 2k \text{ for some } k. \\ \text{So, } n^2 &= (2k)^2 = 4k^2. \text{ So, by} \\ \text{definition of congruence,} & \\ n^2 &\equiv 0 \pmod{4}. \end{aligned}$$

$$\begin{aligned} 0^2 &= 0 \equiv 0 \pmod{4} \\ 1^2 &= 1 \equiv 1 \pmod{4} \\ 2^2 &= 4 \equiv 0 \pmod{4} \\ 3^2 &= 9 \equiv 1 \pmod{4} \\ 4^2 &= 16 \equiv 0 \pmod{4} \end{aligned}$$

It looks like

$$\begin{aligned} n \equiv 0 \pmod{2} &\rightarrow n^2 \equiv 0 \pmod{4}, \text{ and} \\ n \equiv 1 \pmod{2} &\rightarrow n^2 \equiv 1 \pmod{4}. \end{aligned}$$

Case 2 ( $n$  is odd):

Suppose  $n \equiv 1 \pmod{2}$ .

Then,  $n = 2k + 1$  for some  $k$ .

So,  $n^2 = (2k + 1)^2 = 4k^2 + 4k + 1 = 4(k^2 + k) + 1$ . So, by definition of congruence,  $n^2 \equiv 1 \pmod{4}$ .

## 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

$$48 = 2 \cdot 2 \cdot 2 \cdot 2 \cdot 3 = 2^4 \cdot 3$$

$$591 = 3 \cdot 197$$

$$45,523 = 45,523$$

$$321,950 = 2 \cdot 5 \cdot 5 \cdot 47 \cdot 137$$

$$1,234,567,890 = 2 \cdot 3 \cdot 3 \cdot 5 \cdot 3,607 \cdot 3,803$$

$$n = p_1^{a_1} \cdot p_2^{a_2} \cdot \dots \cdot p_m^{a_m}$$

## Euclid's Theorem

There are an infinite number of primes.

Proof by contradiction:

Suppose that there are only a finite number of

primes:  $p_1, p_2, \dots, p_n$

$$N = p_1 \cdot p_2 \cdot \dots \cdot p_n + 1$$

$\exists$  if  $N$  is prime ✓

composite  $\rightarrow$   $N$  is composite,  $p_i \mid N$   
 $p_i \mid N \rightarrow p_i \mid p_1 \cdot p_2 \cdot \dots \cdot p_n + 1$   
 $p_i \mid N \rightarrow p_i \mid 1$  (impossible)

$$\frac{N-1}{p} = \frac{p_1 \cdot p_2 \cdot \dots \cdot p_n}{p} \in \mathbb{Z}$$

## 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  $p_1 < p_2 < \dots < p_n$ . Consider  $P = p_1 p_2 \dots p_n$ , 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  $p$  such that  $p \mid Q$ . Note that since  $P$  divides every possible prime,  $p \mid P$  as well. It follows that  $p \mid (Q - P) \rightarrow p \mid ((P + 1) - P) \rightarrow p \mid 1$ . 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  
 285356959533479219732245215172640050726  
 365751874520219978646938995647494277406  
 384592519255732630345373154826850791702  
 612214291346167042921431160222124047927  
 4737794080665351419597459856902143413

12301866845301177551304949583849627207728535695953347  
 92197322452151726400507263657518745202199786469389956  
 47494277406384592519255732630345373154826850791702612  
 21429134616704292143116022212404792747377940806653514  
 19597459856902143413

=

334780716989568987860441698482126908177047949837  
 137685689124313889828837938780022876147116525317  
 43087737814467999489

×

367460436667995904282446337996279526322791581643  
 430876426760322838157396665112792333734171433968  
 10270092798736308917

## Factoring

Okay. Fun...I guess...

It turns out this is hard.  
 Why should I care?

## Factoring

---

**We can use (“abuse”)  
the fact that factoring  
is difficult for security.**

## Large Numbers

---

**But...**

```
Test.java:3: error: integer number too large:
12345678901234567890
    long x = 12345678901234567890;
            ^
1 error
```

**Thanks Java...**

```
367460436667995904282446337996279526322791581643
430876426760322838157396665112792333734171433968
10270092798736308917
```

**How do we store large numbers?**

## Large Non-negative Integer Representation

---

**String**

“12345”

$$\sum_{i=0}^n d_i 10^i$$

**Sum**

$$5 \cdot 10^0 + 4 \cdot 10^1 + 3 \cdot 10^2 + 2 \cdot 10^3 + 1 \cdot 10^4$$

**List**

[5, 4, 3, 2, 1]

## Large Non-negative Integer Representation

---

**String**

“11000000111001”

$$\sum_{i=0}^n b_i 2^i$$

**Sum**

$$1 \cdot 2^0 + 0 \cdot 2^1 + 0 \cdot 2^2 + 1 \cdot 2^3 + 1 \cdot 2^4 + 1 \cdot 2^5 + 0 \cdot 2^6 + 0 \cdot 2^7 + 0 \cdot 2^8 + 0 \cdot 2^9 + 0 \cdot 2^{10} + 0 \cdot 2^{11} + 1 \cdot 2^{12} + 1 \cdot 2^{13}$$

**List**

[1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1]

## Large Non-negative Integer Operations MOD M

---

**Just like before, we will consider these operations mod another number. This turns out to be “more general” and more useful.**

So, instead of  $(a + b)$ , we'll look at  $(a + b) \bmod m$ ...

## Large Non-negative Integer Operations MOD M

---

**Addition**

```
[0, 0, 1, 1]
+ [1, 0, 0, 1]
-----
[1, 0, 1, 0, 1]
```

...and then we mod by m at the end.

**Hey! This looks familiar!**

**Multiplication**

**This was HW, right?** ...and then we mod by m at the end.

## Large Non-negative Integer Operations MOD M

Subtraction?

Let's borrow from circuit design (again).

How does subtraction mod  $2^{32}$  work?

## Two's Complement Representation

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

$$\begin{array}{r}
 [0, 0, 1, 1] \\
 - [1, 0, 0, 1] \\
 \hline
 [0, 1, 1, 0] \\
 + [1] \\
 \hline
 [1, 1, 1, 0]
 \end{array}$$

## Two's Complement Representation

$$\begin{array}{r}
 [0, 0, 1, 1] \quad 0 \cdot 2^0 + 0 \cdot 2^1 + 1 \cdot 2^2 + 1 \cdot 2^3 = 12 \\
 - [1, 0, 0, 1] \quad 1 \cdot 2^0 + 0 \cdot 2^1 + 0 \cdot 2^2 + 1 \cdot 2^3 = 9 \\
 \hline
 \end{array}$$

????????????????

$$\begin{array}{r}
 \sim [1, 0, 0, 1] \\
 [0, 1, 1, 0] \\
 + [1] \\
 \hline
 [1, 1, 1, 0]
 \end{array}$$

*Handwritten:  $8 \bmod 16 = 8 - 16 = -8$*

$$\begin{array}{r}
 [0, 0, 1, 1] \quad 0 \cdot 2^0 + 0 \cdot 2^1 + 1 \cdot 2^2 + 1 \cdot 2^3 = 12 \\
 + [1, 1, 1, 0] \quad 1 \cdot 2^0 + 1 \cdot 2^1 + 1 \cdot 2^2 + 0 \cdot 2^3 = 7 \\
 \hline
 [1, 1, 0, 0, \text{X}] \quad 1 \cdot 2^0 + 1 \cdot 2^1 + 0 \cdot 2^2 + 0 \cdot 2^3 = 3
 \end{array}$$

## Large Non-negative Integer Operations MOD M

Division?

This is weird. We'll come back to it next lecture.

## Large Non-negative Integer Operations MOD M

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 <sup>1</sup>	a <sup>2</sup>	a <sup>3</sup>	a <sup>4</sup>	a <sup>5</sup>	a <sup>6</sup>
1						
2						
3						
4						
5						
6						

*Handwritten:*  $a^{16} = (a^2)^8 = \left( \frac{a^2}{\frac{a^2}{\frac{a^2}{\dots}} \bmod m} \right)^2 \bmod m$

## Exponentiation

- Compute  $78365^{81453}$
- Compute  $78365^{81453} \bmod 104729$
- Output is small
  - need to keep intermediate results small

## Repeated Squaring – small and fast

Since  $a \bmod m \equiv a \pmod{m}$  for any  $a$   
we have  $a^2 \bmod m = (a \bmod m)^2 \bmod m$   
and  $a^4 \bmod m = (a^2 \bmod m)^2 \bmod m$   
and  $a^8 \bmod m = (a^4 \bmod m)^2 \bmod m$   
and  $a^{16} \bmod m = (a^8 \bmod m)^2 \bmod m$   
and  $a^{32} \bmod m = (a^{16} \bmod m)^2 \bmod m$

Can compute  $a^k \bmod m$  for  $k=2^i$  in only  $i$  steps

## Fast Exponentiation Algorithm

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

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

$$\begin{aligned} a^{81453} \bmod m = & \\ & (\dots((((a^{2^{16}} \bmod m \cdot \\ & \quad a^{2^{13}} \bmod m) \bmod m \cdot \\ & \quad a^{2^{12}} \bmod m) \bmod m \cdot \\ & \quad a^{2^{11}} \bmod m) \bmod m \cdot \\ & \quad a^{2^{10}} \bmod m) \bmod m \cdot \\ & \quad a^{2^9} \bmod m) \bmod m \cdot \\ & \quad a^{2^5} \bmod m) \bmod m \cdot \\ & \quad a^{2^3} \bmod m) \bmod m \cdot \\ & \quad a^{2^2} \bmod m) \bmod m \cdot \\ & \quad a^{2^0} \bmod m) \bmod m \end{aligned}$$

The fast exponentiation algorithm computes  $a^n \bmod m$  using  $O(\log n)$  multiplications mod  $m$

## Fast Exponentiation

```
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;
}
```

## P.S.: Pseudo-Random Number Generation

### Linear Congruential method

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

Choose random  $x_0, a, c, m$  and produce a long sequence of  $x_n$ 's

(Lagged Fibonacci Generator, Mersenne Twister...)