CSE 311

Foundations of Computing I

* All slides are a combined effort between previous instructors of the course

Large Non-negative Integer Operations MOD M

Exponentiation?

$a \equiv 0 \pmod m$

$(a \cdot b \cdot c + d) \bmod m$

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						

a^b

$a^b \bmod m$

$a^{32} = (a^2)^{16} = ((a^2)^2)^8 \ldots$

a has b-bits

$a^5 = a \cdot a^4 = a \cdot (a^2)^2$

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

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

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

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

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

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;
    }
```

$a^n = a^{2k} \rightarrow (a^2)^k$

$a^{2k+1} \rightarrow a \cdot (a^2)^k$

Large Non-negative Integer Operations MOD M

Division?

$\left(\frac{1}{5}\right) \cdot 5 = \frac{5}{5} = 1$

Let's get existential.
What, really, IS division?

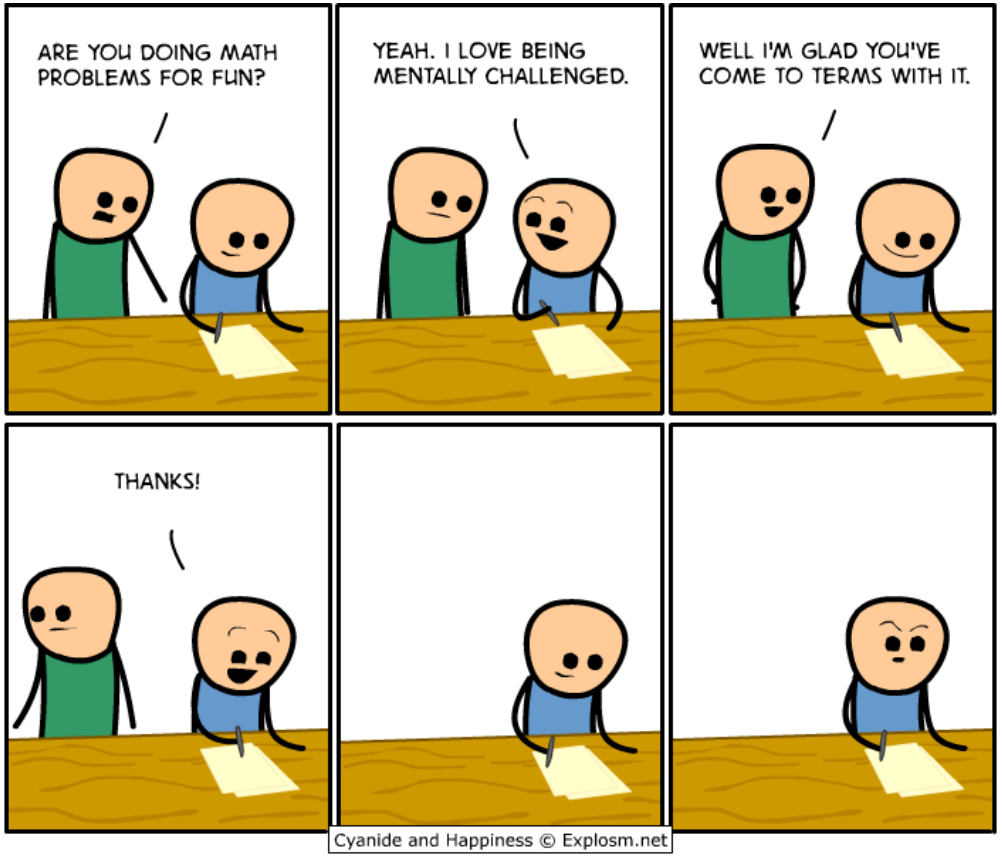
$5 \cdot \frac{1}{2} = \frac{5}{2} =$ "5/2"

$2x = 5$

$2x \equiv 1 \pmod 2$

CSE 311: Foundations of Computing

Lecture 13: Modular Inverses



Greatest Common Divisor

GCD(a, b):

Largest integer d such that $d \mid a$ and $d \mid b$

1 $100 = 2^2 \cdot 5^2$ $125 = 5^3 \cdot 2^0$

- GCD(100, 125) = $2^0 \cdot 5^2 = 25$
- GCD(17, 49) =
- GCD(11, 66) =
- GCD(13, 0) =
- GCD(180, 252) =

GCD and Factoring

$a = 2^3 \cdot 3 \cdot 5^2 \cdot 7 \cdot 11 = 46{,}200$

$b = 2 \cdot 3^2 \cdot 5^3 \cdot 7 \cdot 13 = 204{,}750$

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

Factoring is expensive!

Can we compute GCD(a,b) without factoring?

Useful GCD Fact

If a and b are positive integers, then

$$\gcd(a,b) = \gcd(b,\ a \bmod b)$$

Let $a, b \in \mathbb{Z}^+$

$F_{a,b} \Rightarrow F_{b,\ a \bmod b}$

$F_{b,\ a \bmod b} \Rightarrow F_{a,b}$

By division theorem,

$a = b \cdot (a \operatorname{div} b) + a \bmod b$

$a \bmod b = a - b \cdot (a \operatorname{div} b)$

(1) $F_{a,b} \subseteq F_{b,\ a \bmod b}$

Suppose $x \in F_{a,b}$ is arbitrary.

So, $x \mid a$ and $x \mid b$.

So, $a = kx$ and $b = \ell x$

$kx - \ell x (a \operatorname{div} b) = a - b(a \operatorname{div} b) = (\quad)x$

$= a \bmod b$

So, $b = (\quad)x$

So, $x \mid b$ and $x \mid a \bmod b$.

So, $x \in F_{b,\ a \bmod b}$.

Useful GCD Fact

If a and b are positive integers, then

$$\gcd(a,b) = \gcd(b, a \bmod b)$$

Proof:

By definition of mod, $a = qb + (a \bmod b)$ for some integer $q = a \operatorname{div} b$.

Let $d = \gcd(a,b)$. Then $d \mid a$ and $d \mid b$ so $a = kd$ and $b = jd$ for some integers k and j.
Therefore $(a \bmod b) = a - qb = kd - qjd = d(k - qj)$.
So, $d \mid (a \bmod b)$ and since $d \mid b$ we must have $d \leq \gcd(b, a \bmod b)$.

Now, let $e = \gcd(b, a \bmod b)$. Then $e \mid b$ and $e \mid (a \bmod b)$. It follows that $b = me$ and $(a \bmod b) = ne$ for some integers m and n. Therefore

$$a = qb + (a \bmod b) = qme + ne = e(qm+n)$$

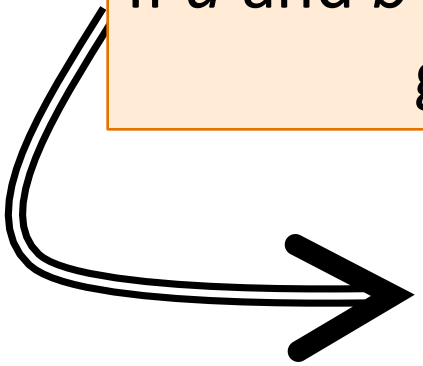
So, $e \mid a$ and since $e \mid b$ we must have $e \leq \gcd(a, b)$.

Therefore $\gcd(a, b) = \gcd(b, a \bmod b)$.

Euclid's Algorithm

If *a* and *b* are positive integers, then

gcd*(a,b)* = gcd(*b, a* mod *b)*



GCD Algorithm

```
gcd(a, 0) = a
gcd(a, b) = gcd(b, a mod b)
```

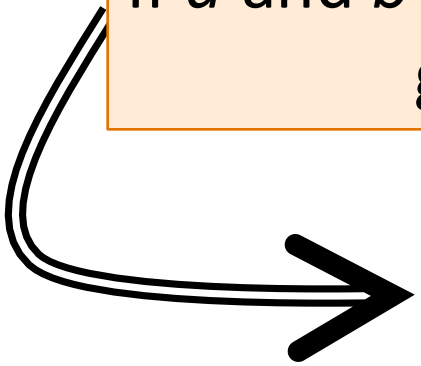
gcd(126, 660) = gcd (660, 126)

= gcd(126, 660 mod 126)

Euclid's Algorithm

If *a* and *b* are positive integers, then

gcd*(a,b)* = gcd(*b, a* mod *b)*



GCD Algorithm

```
gcd(a, 0) = a
gcd(a, b) = gcd(b, a mod b)
```

```
gcd(126, 600) = gcd(600, 600 mod 126)
              = gcd(600, 126)
              = gcd(126, 660 mod 126)
              = gcd(126, 30)
              = gcd(30, 126 mod 60)
              = gcd(30, 6)
              = gcd(6, 30 mod 6)
              = gcd(6, 0)
              = 6
```

Euclid's Algorithm

GCD Algorithm

gcd(a, 0) = a

gcd(a, b) = gcd(b, a mod b)

```
gcd(a, b) {
    if (b == 0) {
        return a;
    }
    else {
        return gcd(b, a mod b);
    }
}
```

Finding x & y

Bézout's Theorem

If *a* and *b* are positive integers, then there exist integers $\mathbf{x}_{(a,b)}$ and $\mathbf{y}_{(a,b)}$ such that

$$\gcd(a, b) = a\mathbf{x}_{(a,b)} + b\mathbf{y}_{(a,b)}$$

GCD Algorithm

```
gcd(a, 0) = a
gcd(a, b) = gcd(b, a mod b)
```

$5x \equiv 1 \pmod{m}$

$\Leftrightarrow 5x + km = 1$

Finding x & y

Bézout's Theorem

If *a* and *b* are positive integers, then there exist integers $x_{(a,b)}$ and $y_{(a,b)}$ such that gcd(a, b) = $ax_{(a,b)} + by_{(a,b)}$

GCD Algorithm

```
gcd(a, 0) = a
gcd(a, b) = gcd(b, a mod b)
```

Case 1: gcd(a, 0) = a

$\gcd(a, 0) = a*X_{a,0} + 0*Y_{a,0}$

$= a \cdot 1 + 0 \cdot 0$

Finding x & y

Bézout's Theorem

If *a* and *b* are positive integers, then there exist integers $x_{(a,b)}$ and $y_{(a,b)}$ such that gcd(a, b) = $ax_{(a,b)} + by_{(a,b)}$

GCD Algorithm

```
gcd(a, 0) = a*1 + 0*0
gcd(a, b) = gcd(b, a mod b)
```

We've figured out the answer for the "base case".

Case 2: `gcd(a, b) = gcd(b, a mod b)`

`gcd(a, b) = a*`$X_{a,b}$ `+ b*`$Y_{a,b}$

Finding x & y

Bézout's Theorem

If *a* and *b* are positive integers, then there exist integers $x_{(a,b)}$ and $y_{(a,b)}$ such that $\gcd(a, b) = ax_{(a,b)} + by_{(a,b)}$

GCD Algorithm

```
gcd(a, 0) = a*1 + 0*0
gcd(a, b) = gcd(b, a mod b)
```

Case 2: gcd(a, b) = gcd(b, a mod b)

$$\gcd(a, b) = aX_{a,b} + bY_{a,b}$$
$$= \gcd(b, a \bmod b) = ?????????$$

We're stuck. We need to find $X_{a,b}$ and $Y_{a,b}$.

We're looking for an equation with a*x + b*y. The "a mod b" doesn't belong.

$$\gcd(b, a \bmod b) = bX_{b,a \bmod b} + (a \bmod b)Y_{b,a \bmod b}$$

Division Theorem

$$a = b(a \operatorname{div} b) + (a \bmod b)$$

Finding x & y

Bézout's Theorem

If *a* and *b* are positive integers, then there exist integers $x_{(a,b)}$ and $y_{(a,b)}$ such that $\gcd(a, b) = ax_{(a,b)} + by_{(a,b)}$

GCD Algorithm

```
gcd(a, 0) = a*1 + 0*0
gcd(a, b) = gcd(b, a mod b)
```

Case 2: gcd(a, b) = gcd(b, a mod b)

$$\gcd(a, b) = aX_{a,b} + bY_{a,b}$$
$$= \gcd(b, a \bmod b) = ?????????$$

We're stuck. We need to find $X_{a,b}$ and $Y_{a,b}$.
We're looking for an equation with a*x + b*y. The "a mod b" doesn't belong.

$$\gcd(b, a \bmod b) = bX_{b,a \bmod b} + (a \bmod b)Y_{b,a \bmod b}$$
$$= bX_{b,a \bmod b} + (a - b(a \operatorname{div} b))Y_{b,a \bmod b}$$

Division Theorem

$$a = b(a \operatorname{div} b) + (a \bmod b)$$
$$(a \bmod b) = a - b(a \operatorname{div} b)$$

Finding x & y

Bézout's Theorem

If *a* and *b* are positive integers, then there exist integers $x_{(a,b)}$ and $y_{(a,b)}$ such that $\gcd(a, b) = ax_{(a,b)} + by_{(a,b)}$

GCD Algorithm

```
gcd(a, 0) = a*1 + 0*0
gcd(a, b) = gcd(b, a mod b)
```

Case 2: gcd(a, b) = gcd(b, a mod b)

$$\gcd(a, b) = aX_{a,b} + bY_{a,b}$$
$$= \gcd(b, a \bmod b) = \text{?????????}$$

$$\gcd(b, a \bmod b) = bX_{b,a \bmod b} + (a \bmod b)Y_{b,a \bmod b}$$
$$= bX_{b,a \bmod b} + (a - b(a \operatorname{div} b))Y_{b,a \bmod b}$$

We're still looking for $X_{a,b}$ and $Y_{a,b}$. The equation has x and y terms. Group them…

Finding x & y

Bézout's Theorem

If *a* and *b* are positive integers, then there exist integers $x_{(a,b)}$ and $y_{(a,b)}$ such that $\gcd(a, b) = ax_{(a,b)} + by_{(a,b)}$

GCD Algorithm

```
gcd(a, 0) = a*1 + 0*0
gcd(a, b) = gcd(b, a mod b)
```

Case 2: gcd(a, b) = gcd(b, a mod b)

$$\gcd(a, b) = aX_{a,b} + bY_{a,b}$$
$$= \gcd(b, a \bmod b) = ?????????$$

$$\gcd(b, a \bmod b) = bX_{b,a \bmod b} + (a \bmod b)Y_{b,a \bmod b}$$
$$= bX_{b,a \bmod b} + (a - b(a \operatorname{div} b))Y_{b,a \bmod b}$$

We're still looking for $X_{a,b}$ and $Y_{a,b}$. The equation has x and y terms. Group them...

$$= bX_{b,a \bmod b} + aY_{b,a \bmod b} - b(a \operatorname{div} b)Y_{b,a \bmod b}$$
$$= b(X_{b,a \bmod b} - (a \operatorname{div} b)Y_{b,a \bmod b}) + aY_{b,a \bmod b}$$
$$= aY_{b,a \bmod b} + b(X_{b,a \bmod b} - (a \operatorname{div} b)Y_{b,a \bmod b})$$

$$\gcd(b, a \bmod b) = aY_{b,a \bmod b} + b(X_{b,a \bmod b} - (a \operatorname{div} b)Y_{b,a \bmod b})$$

Finding x & y

Bézout's Theorem

If *a* and *b* are positive integers, then there exist integers $x_{(a,b)}$ and $y_{(a,b)}$ such that gcd(a, b) = $ax_{(a,b)} + by_{(a,b)}$

GCD Algorithm

```
gcd(a, 0) = a*1 + 0*0
gcd(a, b) = gcd(b, a mod b)
```

Case 2: gcd(a, b) = gcd(b, a mod b)

$$\begin{aligned}\gcd(a, b) &= aX_{a,b} + bY_{a,b} \\ &= \gcd(b, a \bmod b) \\ &= aY_{b,a \bmod b} + b(X_{b,a \bmod b} - (a \operatorname{div} b)Y_{b,a \bmod b})\end{aligned}$$

Finding x & y

Bézout's Theorem

If *a* and *b* are positive integers, then there exist integers $x_{(a,b)}$ and $y_{(a,b)}$ such that $\gcd(a, b) = ax_{(a,b)} + by_{(a,b)}$

GCD Algorithm

```
gcd(a, 0) = a
gcd(a, b) = gcd(b, a mod b)
```

Case 2: gcd(a, b) = gcd(b, a mod b)

$$\begin{aligned}\gcd(a, b) &= aX_{a,b} + bY_{a,b} \\ &= \gcd(b, a \bmod b) \\ &= aY_{b,a \bmod b} + b(X_{b,a \bmod b} - (a \operatorname{div} b)Y_{b,a \bmod b})\end{aligned}$$

EGCD Algorithm

```
egcd(a, 0) = a*1 + 0*0
```

$$\text{egcd}(a, b) = a*Y_{b,a \bmod b} + b*(X_{b,a \bmod b} - (a \operatorname{div} b)Y_{b,a \bmod b})$$

Finding x & y

GCD Algorithm

$\gcd(a, 0) = a$

$\gcd(a, b) = \gcd(b, a \bmod b)$

EGCD Algorithm

$\text{egcd}(a, 0) = a*1 + 0*0$

$\text{egcd}(a, b) = a*Y_{b,a \bmod b} + b*(X_{b,a \bmod b} - (a \text{ div } b)Y_{b,a \bmod b})$

EGCD Algorithm

$\text{egcd}(a, 0) = (a, 1, 0)$

$\text{egcd}(a, b) = (\gcd(b, a \bmod b), Y_{b,a \bmod b}, X_{b,a \bmod b} - (a \text{ div } b)*Y_{b,a \bmod b})$

Computing x & y By Hand

EGCD Algorithm

egcd(a, 0) = (a, 1, 0)

egcd(a, b) = (gcd(b, a mod b), $Y_{b,a \bmod b}$, $X_{b,a \bmod b} - (a \operatorname{div} b) * Y_{b,a \bmod b}$)

Find x and y such that gcd(35, 27) = 35*x + 27*y

a	b	a mod b	a div b	gcd(a, b)	=	a	*	x	+	b	*	y
35	27	8	1		=	35	*		+	27	*	
					=		*		+		*	
					=		*		+		*	
					=		*		+		*	
					=		*		+		*	
					=		*		+		*	

We fill in this table row by row. First, we fill in all the left side. Then, we fill in the right side on our way back up.

Computing x & y By Hand

EGCD Algorithm

egcd(a, 0) = (a, 1, 0)

egcd(a, b) = (gcd(b, a mod b), $Y_{b,a \bmod b}$, $X_{b,a \bmod b} - (a \operatorname{div} b) * Y_{b,a \bmod b}$)

Find x and y such that gcd(35, 27) = 35*x + 27*y

a	b	a mod b	a div b	gcd(a, b)	=	a	*	x	+	b	*	y
35	27	8	1		=	35	*		+	27	*	
27	8				=	27	*		+	8	*	
					=		*		+		*	
					=		*		+		*	
					=		*		+		*	
					=		*		+		*	

Since gcd(a,b) = gcd(b, a mod b), we copy down b and a mod b.

Computing x & y By Hand

EGCD Algorithm

egcd(a, 0) = (a, 1, 0)

egcd(a, b) = (gcd(b, a mod b), $Y_{b,a \bmod b}$, $X_{b,a \bmod b}$ – (a div b)*$Y_{b,a \bmod b}$)

Find x and y such that gcd(35, 27) = 35*x + 27*y

a	b	a mod b	a div b	gcd(a, b)	=	a	*	x	+	b	*	y
35	27	8	1		=	35	*		+	27	*	
27	8	3	3		=	27	*		+	8	*	
8	3	2	2		=	8	*		+	3	*	
3	2	1	1		=	3	*		+	2	*	
2	1	0	2		=	2	*		+	1	*	
1	0				=	1	*		+	0	*	

We got to gcd(a, 0) = a. So, fill in gcd(a, b) all the way up.

Computing x & y By Hand

EGCD Algorithm

egcd(a, 0) = (a, 1, 0)

egcd(a, b) = (gcd(b, a mod b), $Y_{b,a \bmod b}$, $X_{b,a \bmod b}$ – (a div b)*$Y_{b,a \bmod b}$)

Find x and y such that gcd(35, 27) = 35*x + 27*y

a	b	a mod b	a div b	gcd(a, b)	=	a	*	x	+	b	*	y
35	27	8	1	1	=	35	*		+	27	*	
27	8	3	3	1	=	27	*		+	8	*	
8	3	2	2	1	=	8	*		+	3	*	
3	2	1	1	1	=	3	*		+	2	*	
2	1	0	2	1	=	2	*		+	1	*	
1	0			1	=	1	*		+	0	*	

Now, we back-fill X and Y using the equation for egcd:

- $X_n = Y_{n+1}$
- $Y_n = X_{n+1} - (a \text{ div } b) * Y_{n+1}$

Computing x & y By Hand

EGCD Algorithm

egcd(a, 0) = (a, 1, 0)

egcd(a, b) = (gcd(b, a mod b), $Y_{b,a \bmod b}$, $X_{b,a \bmod b}$ − (a div b)*$Y_{b,a \bmod b}$)

Find x and y such that gcd(35, 27) = 35*x + 27*y

a	b	a mod b	a div b	gcd(a, b)	=	a	*	x	+	b	*	y
35	27	8	1	1	=	35	*		+	27	*	
27	8	3	3	1	=	27	*		+	8	*	
8	3	2	2	1	=	8	*		+	3	*	
3	2	1	1	1	=	3	*		+	2	*	
2	1	0	2	1	=	2	*		+	1	*	
1	0			1	=	1	*	**1**	+	0	*	**0**

Computing x & y By Hand

EGCD Algorithm

egcd(a, 0) = (a, 1, 0)

egcd(a, b) = (gcd(b, a mod b), $Y_{b,a \bmod b}$, $X_{b,a \bmod b}$ – (a div b)*$Y_{b,a \bmod b}$)

Find x and y such that gcd(35, 27) = 35*x + 27*y

a	b	a mod b	a div b	gcd(a, b)	=	a	*	x	+	b	*	y
35	27	8	1	1	=	35	*		+	27	*	
27	8	3	3	1	=	27	*		+	8	*	
8	3	2	2	1	=	8	*		+	3	*	
3	2	1	1	1	=	3	*		+	2	*	
2	1	0	2	1	=	2	*	**0**	+	1	*	**1**
1	0			1	=	1	*	**1**	+	0	*	**0**

Computing x & y By Hand

EGCD Algorithm

egcd(a, 0) = (a, 1, 0)

egcd(a, b) = (gcd(b, a mod b), $Y_{b,a \bmod b}$, $X_{b,a \bmod b}$ − (a div b)*$Y_{b,a \bmod b}$)

Find x and y such that gcd(35, 27) = 35*x + 27*y

a	b	a mod b	a div b	gcd(a, b)	=	a	*	x	+	b	*	y
35	27	8	1	1	=	35	*	**-10**	+	27	*	**13**
27	8	3	3	1	=	27	*	**3**	+	8	*	**-10**
8	3	2	2	1	=	8	*	**-1**	+	3	*	**3**
3	2	1	1	1	=	3	*	**1**	+	2	*	**-1**
2	1	0	2	1	=	2	*	**0**	+	1	*	**1**
1	0			1	=	1	*	**1**	+	0	*	**0**

So, 1 = gcd(35, 27) = 35*(-10) + 27*13

Large Non-negative Integer Operations MOD M

Division?

FINALLY! We're back to division mod m.

In normal arithmetic, if I multiply x * (1/x), I get back x.

In MODULAR arithmetic, if I multiply x * ?, I get back x.

"1/x" is the unique number that, when multiplied by x gives 1.

"1/x" is a solution, N, to the equation $xN \equiv 1 \pmod{m}$.

$$\begin{aligned} xN \equiv 1 \pmod{m} &\leftrightarrow m \mid (xN - 1) \\ &\leftrightarrow xN - 1 = km \\ &\leftrightarrow xN + (-k)m = 1 \end{aligned}$$

We know how to do this now! It's just EGCD!

Large Non-negative Integer Operations MOD M

Division?

FINALLY! We're back to division mod m.

In normal arithmetic, if I multiply x * (1/x), I get back x.

In MODULAR arithmetic, if I multiply x * ?, I get back x.

"1/x" is the unique number that, when multiplied by x gives 1.

Example

Solve: $7x \equiv 1 \pmod{26}$

Example

Solve: $7x \equiv 1 \pmod{26}$

gcd(26, 7) = gcd(7, 5) = gcd(5, 2) = gcd(2, 1) = 1

26 = 7*3 + 5	5 = 26 – 7*3
7 = 5*1 + 2	2 = 7 – 5*1
5 = 2*2 + 1	1 = 5 – 2*2

1 = 5 – (7 – 5*1)*2
= (–7)*2 + 5*3
= (–7)*2 + (26 – 7*3)*3
= 7*(-11) + 26*3

So, $x = 15 + 26k$ for $k \in \mathrm{N}$.