(Extended) Euclidean Algorithm and Fermat’s little theorem

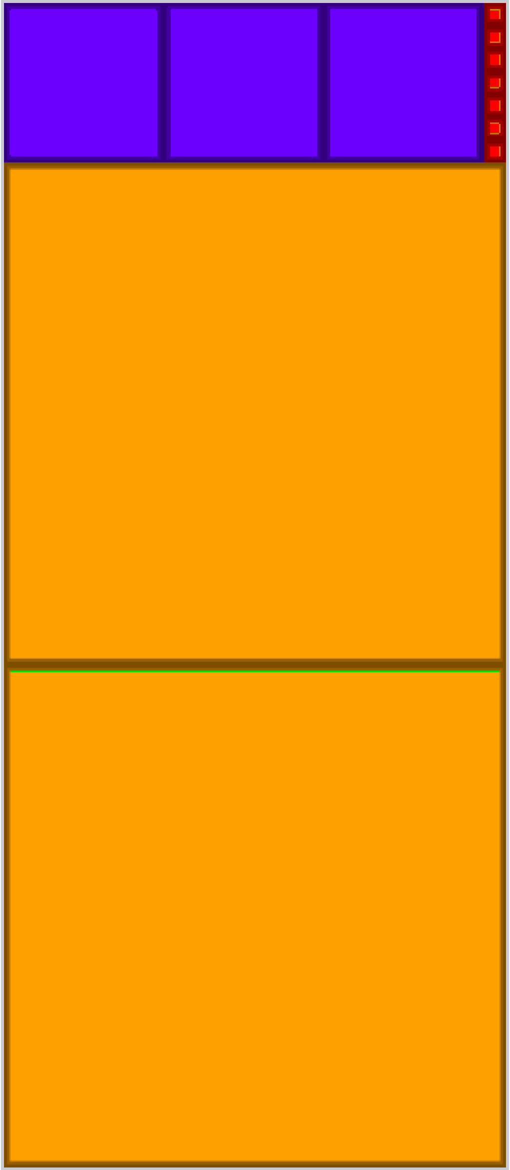
CSE 468 Fall 2025

jedimaestro@asu.edu

For gcd (greatest common divisor)

- https://en.wikipedia.org/wiki/Euclidean_algorithm

Subtraction-based animation of the Euclidean algorithm. The initial rectangle has dimensions *a* = 1071 and *b* = 462. Squares of size 462×462 are placed within it leaving a 462×147 rectangle. This rectangle is tiled with 147×147 squares until a 21×147 rectangle is left, which in turn is tiled with 21×21 squares, leaving no uncovered area. The smallest square size, 21, is the GCD of 1071 and 462.



```
function gcd(a, b)
    if b = 0
        return a
    else
        return gcd(b, a mod b)
```

```
function gcd(a, b)
    while a ≠ b
        if a > b
            a := a − b
        else
            b := b − a
    return a
```

https://stackoverflow.com/questions/11175131/code-for-greatest-common-divisor-in-python

Source code from the `inspect` module in Python 2.7:

```
>>> print inspect.getsource(gcd)
def gcd(a, b):
    """Calculate the Greatest Common Divisor of a and b.

    Unless b==0, the result will have the same sign as b (so
    b is divided by it, the result comes out positive).
    """
    while b:
        a, b = b, a%b
    return a
```

Extended Euclidean Algorithm

- https://en.wikipedia.org/wiki/Extended_Euclidean_algorithm
- https://crypto.stackexchange.com/questions/5889/calculating-rsa-private-exponent-when-given-public-exponent-and-the-modulus-fact

The following table shows how the extended Euclidean algorithm proceeds with input 240 and 46. The greatest common divisor is the last non zero entry, 2 in the column "remainder". The computation stops at row 6, because the remainder in it is 0. Bézout coefficients appear in the last two entries of the second-to-last row. In fact, it is easy to verify that −9 × 240 + 47 × 46 = 2. Finally the last two entries 23 and −120 of the last row are, up to the sign, the quotients of the input 46 and 240 by the greatest common divisor 2.

index i	quotient q_{i-1}	Remainder r_i	s_i	t_i
0		240	1	0
1		46	0	1
2	240 ÷ 46 = 5	240 − 5 × 46 = 10	1 − 5 × 0 = 1	0 − 5 × 1 = −5
3	46 ÷ 10 = 4	46 − 4 × 10 = 6	0 − 4 × 1 = −4	1 − 4 × −5 = 21
4	10 ÷ 6 = 1	10 − 1 × 6 = 4	1 − 1 × −4 = 5	−5 − 1 × 21 = −26
5	6 ÷ 4 = 1	6 − 1 × 4 = 2	−4 − 1 × 5 = −9	21 − 1 × −26 = 47
6	4 ÷ 2 = 2	4 − 2 × 2 = 0	5 − 2 × −9 = 23	−26 − 2 × 47 = −120

```
function extended_gcd(a, b)
    (old_r, r) := (a, b)
    (old_s, s) := (1, 0)
    (old_t, t) := (0, 1)

    while r ≠ 0 do
        quotient := old_r div r
        (old_r, r) := (r, old_r − quotient × r)
        (old_s, s) := (s, old_s − quotient × s)
        (old_t, t) := (t, old_t − quotient × t)

    output "Bézout coefficients:", (old_s, old_t)
    output "greatest common divisor:", old_r
    output "quotients by the gcd:", (t, s)
```

Multiplicative inverses for finite fields...

- Find $d = e^{-1}$ for a finite field mod p:
  - $sp + te = gcd(p, e)$
  - $sp + e^{-1}e = 1$
    - $t = d = e^{-1}$, can throw away s
- Easier way (you'll do both on HW and exam): Fermat's little theorem...

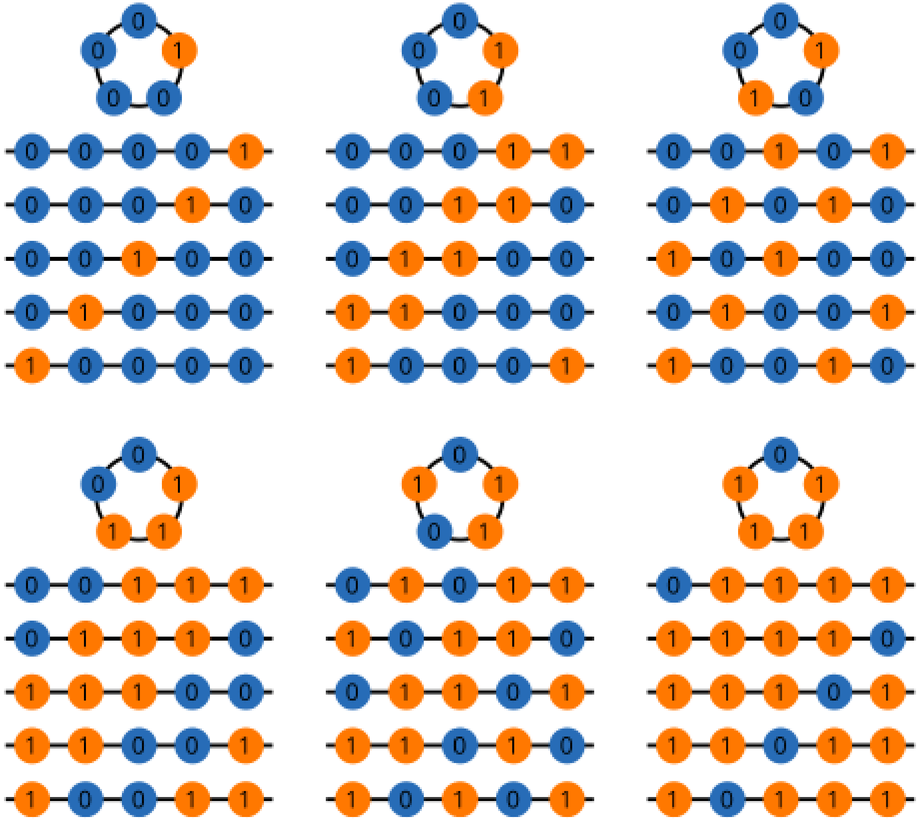
https://en.wikipedia.org/wiki/Finite_field_arithmetic#Multiplicative_inverse

- Since the nonzero elements of GF(p^n) form a finite group with respect to multiplication, $a^{p^n-1} = 1$ (for $a \neq 0$), thus the inverse of a is a^{p^n-2}. This algorithm is a generalization of the modular multiplicative inverse based on Fermat's little theorem.

We only care about n=1 for the HW and exam.

$$a^p \bmod p = a \pmod{p}$$
$$a^{p-1} \bmod p = 1 \pmod{p}$$
$$a^{p-2} \bmod p = a^{-1} \pmod{p}$$

https://mathlesstraveled.wordpress.com/2017/12/12/fermats-little-theorem-proof-by-necklaces/



We already know there are $a^p - a$ strands with at least two colors; since we can put them in groups of p, one for each necklace of at least two colors, $a^p - a$ must be evenly divisible by p. QED!

Finite fields *mod p*

- Inverse is just e^{p-2}
- **So why study the Extended Euclidean algorithm?** Because we can't do signatures with Diffie-Hellman, since Fermat's little theorem is an easy way to find multiplicative inverses.
- Same is true of any finite field, so RSA uses ring theory:
  - $n = pq$ where p and q are prime
  - $\varphi(n) = (p - 1)(q - 1)$ is Euler's totient function, which counts the numbers less than n that are co-prime to n

Your goal is to find d such that $ed \equiv 1 \pmod{\varphi(n)}$.

Recall the EED calculates x and y such that $ax + by = \gcd(a, b)$. Now let $a = e$, $b = \varphi(n)$, and thus $\gcd(e, \varphi(n)) = 1$ by definition (they need to be coprime for the inverse to exist). Then you have:

$$ex + \varphi(n)y = 1$$

Take this modulo $\varphi(n)$, and you get:

$$ex \equiv 1 \pmod{\varphi(n)}$$

And it's easy to see that in this case, $x = d$. The value of y does not actually matter, since it will get eliminated modulo $\varphi(n)$ regardless of its value. The EED will give you that value, but you can safely discard it.