# Number Theory

Dr Basel Halak



## Learning Outcomes

**At the end of this lecture you should be able to:**

1. Classify computational problems according to their complexity.
2. Perform computation using modular arithmetic.
3. Discuss a number of intractable problems in modular arithmetic.

## Learning Outcomes

**At the end of this lecture you should be able to:**

1. Classify computational problems according to their complexity.
2. Perform computation using modular arithmetic.
3. Discuss a number of intractable problems in modular arithmetic.

## Computational Complexity Theory

- **Computational complexity** theory is a branch of the theory of computation in mathematics that focuses on classifying computational problems according to their inherent difficulty.

- **A computational problem** is said to be a task that is in principle amenable to being solved by a computer in other words, the problem may be solved by automatic application of mathematical steps, such as an algorithm.

# Computational Complexity Theory

- **The Big O notation** is used to classify algorithms by how they respond to changes in input size in terms of their processing time or working space requirements
- **Big O notation is useful when analyzing algorithms for complexity** (e.g. this notation can be used indicate the relationship between the size of the input and the number of steps needed to execute an algorithm on a space constrained machine).

- **Example:**

$T(n) = n^2 + n + 2$

We can state:

$T(n) \in O(n^2)$

and say that the algorithm has *order of* $n^2$ time complexity

# Learning Outcomes

**At the end of this lecture you should be able to:**

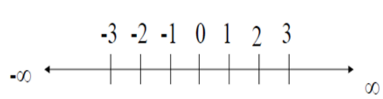
1. Classify computational problems according to their complexity.
2. Perform computation using modular arithmetic.
3. Discuss a number of intractable problems in modular arithmetic.

# QUIZ

| Computation Problem | Complexity |
|---|---|
| Adding two N-digit numbers | |
| Multiplying two *n*-digit numbers | |
| Cracking a n-letter Transposition Cipher by Brute Force Search | |
| Cracking a n-letter shift by Brute Force Search | |

# Modular Arithmetic Basics

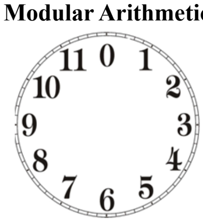
**Normal Arithmetic**




**Important number systems**
- **Z**, the set of all integers 0,±1 ± 2, . . .;

- **Q**, the set of all rational numbers a/b (a, b ∈ Z, b ≠ 0);
- **R**, the set of all real numbers
- **C**, the set of all complex numbers a + bj (a, b R).

**Modular Arithmetic**




**$Z_{12=}$ {0,1,2,3,4,5,6,7,8,9,10,11}**
**a = b (mod n)**
**a = b + k * n**
**where k is ANY integer**

**For example:**

**13 = 1 (mod 12) 13 = 1 + 1* 12**
**26 = 2 (mod 12) 26= 2 + 2*12**

## The mod congruence

- **Definition**: Let $a1$, $a2$ be integers and $b$ be a positive integer. We say that $a1$ is congruent to $a2$ modulo $b$ (denoted by $a1 \equiv a2 \pmod{b}$) if ($a2 - a2$) *is a multiple of b*.
  Equivalently: $a1 \bmod b = a2 \bmod b$.

- The (mod) congruence is useful for manipulating expressions involving the mod function. It lets us view modular arithmetic relative a fixed base, as creating a number system inside of which all the calculations can be carried out.

## Example 1

Calculate the mode congruence of 113 in $Z_{24}$

## Example 1

Calculate the mode congruence of 113 in $Z_{24}$

1. 113 **mod** 24:

$$\begin{array}{r} 4 \\ 24\overline{)113} \\ 96 \\ \hline 17 \end{array}$$

This means 113= 17 in $Z_{24}$

Tip: using your calculator divide 113 by 24 , then the multiply the fractional part of the answer by 24

## Multiplication and Addition in Zn

Example: Compute 100 + 30 in $Z_{24}$

We do the addition as normal 100 + 30 = 130 , then using a calculator we find 130/24= 5.4166667.

Then we multiply the fractional part(0.4166667) by 24, so the answer is 10

Check that 130 = 5*24 +10

Exercise: Compute $392 \times 514$ in $Z_{1024}$

# Modular Inversion

- Over the rationales, inverse of 2 is ½ . What about $\mathbf{Z_n}$ ?
- **Definition:** The **inverse** of x in $\mathbf{Z_n}$ is an element y in $\mathbf{Z_n}$ such as x.y =1, y is denoted $x^{-1}$ .

# Modular Inversion

- **Example:** What is the multiplicative inverse of x=3 in $\mathbf{Z_7}$

**Multiplication modulo7**

| * | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 2 | 0 | 2 | 4 | 6 | 1 | 3 | 5 |
| 3 | 0 | 3 | 6 | 2 | 5 | 1 | 4 |
| 4 | 0 | 4 | 1 | 5 | 2 | 6 | 3 |
| 5 | 0 | 5 | 3 | 1 | 6 | 4 | 2 |
| 6 | 0 | 6 | 5 | 4 | 3 | 2 | 1 |

## Division in Zn

Example : Compute 2/3 in $Z_7$

## Division in Zn

Example : Compute 2/3 in $Z_7$

$2/3 = 2 * 1/3 = 2* 5 = 10 = 3$ in $Z_7$

# Division in Zn

**Do all elements have an inverse in $Z_n$ ?**

- Lemma: for all integers x,y there exist integers a,b such that

$$a \cdot x + b \cdot y = gcd(x, y)$$

- **Theorem: $x$ in $Z_n$ has an inverse if and only if $gcd(x, N) = 1$**

# Euclid's Algorithm

- **This algorithm is based on the simple observation that:**

$$\gcd(r_0, r_1) = \gcd(r_0 - r_1, r_1)$$

where $r_0$ , $r_1$ are positive integer and $r_0 > r_1$

- Example

gcd(77, 44 )= 11

gcd((77-44),44 )=?

**The above observation allows us to obtain the following lemma**

$$\gcd(r_0, r_1) = \gcd(r_1, r_0 \bmod r_1)$$

# Euclid's Algorithm

- We can apply this process iteratively to find gcd for large numbers:

$\gcd(r_0, r_1) = \gcd(r_0 - r_1, r_1) = \gcd(r_0 - 2r_1, r_1) = \text{-------} = \gcd(r_0 - mr_1, r_1)$

as long as $r_0 - mr_1 > 0$

# Euclid's Algorithm

- We can apply this process iteratively to find gcd for large numbers:

$\gcd(r_0, r_1) = \gcd(r_0 - r_1, r_1) = \gcd(r_0 - 2r_1, r_1) = \text{-------} = \gcd(r_0 - mr_1, r_1)$

as long as $r_0 - mr_1 > 0$

- To reduce the number of steps we use the maximum value of m, in this case:

$$r_0 - mr_1 = r_0 \bmod r_1$$

- Hence

$$\gcd(r_0, r_1) = \gcd(r_0 \bmod r_1, r_1)$$

# Euclid’s Algorithm

- We can apply this process iteratively to find gcd for large numbers:

$\gcd(r_0, r_1) = \gcd(r_0 - r_1, r_1) = \gcd(r_0 - 2r_1, r_1) = \text{-------} = \gcd(r_0 - mr_1, r_1)$

as long as $r_0 - mr_1 > 0$

- To reduce the number of steps we use the maximum value of m, in this case:

$$r_0 - mr_1 = r_0 \bmod r_1$$

- Hence

$$\gcd(r_0, r_1) = \gcd(r_0 \bmod r_1, r_1)$$

- Since $r_0 \bmod r_1 < r_1$, we need to swap them:

$$\gcd(r_0, r_1) = \gcd(r_1, r_0 \bmod r_1)$$


# Euclid’s Algorithm

- We can apply this process iteratively to find gcd for large numbers:

$\gcd(r_0, r_1) = \gcd(r_0 - r_1, r_1) = \gcd(r_0 - 2r_1, r_1) = \text{-------} = \gcd(r_0 - mr_1, r_1)$

as long as $r_0 - mr_1 > 0$

- To reduce the number of steps we use the maximum value of m, in this case:

$$r_0 - mr_1 = r_0 \bmod r_1$$

- Hence

$$\gcd(r_0, r_1) = \gcd(r_0 \bmod r_1, r_1)$$

- Since $r_0 \bmod r_1 < r_1$, we need to swap them:

$$\gcd(r_0, r_1) = \gcd(r_1, r_0 \bmod r_1)$$

## Euclid's Algorithm

- **Example: Calculate gcd (60,22)**

60 = 2 x 22 + 16 gcd(60, 22)=gcd (22, 60 mod 22) =

## Euclid's Algorithm

- **Example:** Calculate: gcd (60,22)

60 = 2 x 22 + 16 gcd(60, 22)=gcd (22, 60 mod 22) = gcd(22, 16)

22 = 1 x 16 + 6 gcd(22, 16)=gcd (16, 22 mod 16) =

# Euclid's Algorithm

- **Example:** Calculate: gcd (60,22)

60 = 2 x 22 + 16 gcd(60, 22)=gcd (22, 60 mod 22) = gcd(22, 16)
22 = 1 x 16 + 6 gcd(22, 16)=gcd (16, 22 mod 16) = gcd(16, 6)
16 = 2 x 6 + 4 gcd(16, 6)=gcd (6, 16 mod 6) =

# Euclid's Algorithm

- **Example: Calculate: gcd (60,22)**

60 = 2 x 22 + 16 **gcd(60, 22)=**gcd (22, 60 mod 22) = gcd(22, 16)
22 = 1 x 16 + 6 gcd(22, 16)=gcd (16, 22 mod 16) = gcd(16, 6)
16 = 2 x 6 + 4 gcd(16, 6)=gcd (6, 16 mod 6) = gcd(6, 4)
6 = 1 x 4 + 2 gcd(6,4)=gcd (4, 6 mod 4) = gcd(4, 2)
4 = 2 x 2 + 0 gcd(4, 2)=gcd (2, 4 mod 2) = **gcd(2, 0)**

## Euclid's Algorithm

- **Example: Calculate: gcd (60,22)**

| | |
|---|---|
| 60 = 2 x 22 + 16 | gcd(60, 22)=gcd (22, 60 mod 22) = gcd(22, 16) |
| 22 = 1 x 16 + 6 | gcd(22, 16)=gcd (16, 22 mod 16) = gcd(16, 6) |
| 16 = 2 x 6 + 4 | gcd(16, 6)=gcd (6, 16 mod 6) = gcd(6, 4) |
| 6 = 1 x 4 + 2 | gcd(6,4)=gcd (4, 6 mod 4) = gcd(4, 2) |
| 4 = 2 x 2 + 0 | gcd(4, 2)=gcd (2, 4 mod 2) = gcd(2, 0) |

Therefore:

gcd(60, 22) =2

## Modular Inversion using Extended Euclidean Algorithm

**How to find an inverse of an element in $\mathbf{Z_n}$**

1. Check if x has an inverse in $\mathbf{Z_n}$ (gcd(x, N) must be equal to 1
2. **Find a, b such that:** $a \cdot x + b \cdot N = 1$ (this can be done using the Extended Euclidean algorithm).
3. a is the inverse of x in $\mathbf{Z_n}$

## Finding an a multiplicative inverse using the Extended Euclidean Algorithm

- **Example: Find the multiplicative inverse of 8 mod 11, using the Euclidean Algorithm.**

Solution. We'll organize our work carefully. We'll do the Euclidean Algorithm in the left column. It will verify that gcd(8, 11) = 1. Then we'll solve for the remainders in the right column, before back solving:

$\mathbf{11} = \mathbf{8}(1) + \mathbf{3}$ | $\mathbf{3} = 11 - 8(1)$

## Finding an a multiplicative inverse using the Extended Euclidean Algorithm

- **Example: Find the multiplicative inverse of 8 mod 11, using the Euclidean Algorithm.**

Solution. We'll organize our work carefully. We'll do the Euclidean Algorithm in the left column. It will verify that gcd(8, 11) = 1. Then we'll solve for the remainders in the right column, before back solving:

$\mathbf{11} = \mathbf{8}(1) + \mathbf{3}$ | $\mathbf{3} = 11 - 8(1)$

$\mathbf{8} = \mathbf{3}(2) + \mathbf{2}$ | $\mathbf{2} = 8 - 3(2)$

## Finding an a multiplicative inverse using the Extended Euclidean Algorithm

- **Example: Find the multiplicative inverse of 8 mod 11, using the Euclidean Algorithm.**

Solution. We'll organize our work carefully. We'll do the Euclidean Algorithm in the left column. It will verify that gcd(8, 11) = 1. Then we'll solve for the remainders in the right column, before back solving:

$\mathbf{11} = \mathbf{8}(1) + \mathbf{3}$ | $\mathbf{3} = 11 - 8(1)$

$\mathbf{8} = \mathbf{3}(2) + \mathbf{2}$ | $\mathbf{2} = 8 - 3(2)$

$\mathbf{3} = \mathbf{2}(1) + \mathbf{1}$ | $\mathbf{1} = 3 - 2(1)$

$\mathbf{2} = \mathbf{1}(2)$

## Finding an a multiplicative inverse using the Extended Euclidean Algorithm

- **Example: Find the multiplicative inverse of 8 mod 11, using the Euclidean Algorithm.**

Solution. We'll organize our work carefully. We'll do the Euclidean Algorithm in the left column. It will verify that gcd(8, 11) = 1. Then we'll solve for the remainders in the right column, before back solving:

$\mathbf{11} = \mathbf{8}(1) + \mathbf{3}$ | $\mathbf{3} = 11 - 8(1)$

$\mathbf{8} = \mathbf{3}(2) + \mathbf{2}$ | $\mathbf{2} = 8 - 3(2)$

$\mathbf{3} = \mathbf{2}(1) + \mathbf{1}$ | $\mathbf{1} = 3 - 2(1)$

$\mathbf{2} = \mathbf{1}(2)$

Now reverse the process using the equations on the right.

$\mathbf{1} = 3 - \mathbf{2}(1)$

## Finding an a multiplicative inverse using the Extended Euclidean Algorithm

- **Example: Find the multiplicative inverse of 8 mod 11, using the Euclidean Algorithm.**

Solution. We'll organize our work carefully. We'll do the Euclidean Algorithm in the left column. It will verify that $\gcd(8, 11) = 1$. Then we'll solve for the remainders in the right column, before back solving:

$\mathbf{11} = \mathbf{8}(1) + \mathbf{3}$ | $\mathbf{3} = 11 - 8(1)$

$\mathbf{8} = \mathbf{3}(2) + \mathbf{2}$ | $\mathbf{2} = 8 - 3(2)$

$\mathbf{3} = \mathbf{2}(1) + \mathbf{1}$ | $\mathbf{1} = \mathbf{3} - 2(1)$

$\mathbf{2} = \mathbf{1}(2)$

Now reverse the process using the equations on the right.

$\mathbf{1} = \mathbf{3} - \mathbf{2}(1)$

$1 = \mathbf{3} - (8 - 3(2))(1) = \mathbf{3} - (8 - (3(2)) = \mathbf{3}(3) - 8$

## Finding an a multiplicative inverse using the Extended Euclidean Algorithm

- **Example: Find the multiplicative inverse of 8 mod 11, using the Euclidean Algorithm.**

Solution. We'll organize our work carefully. We'll do the Euclidean Algorithm in the left column. It will verify that $\gcd(8, 11) = 1$. Then we'll solve for the remainders in the right column, before back solving:

$\mathbf{11} = \mathbf{8}(1) + \mathbf{3}$ | $\mathbf{3} = 11 - 8(1)$

$\mathbf{8} = \mathbf{3}(2) + \mathbf{2}$ | $\mathbf{2} = 8 - 3(2)$

$\mathbf{3} = \mathbf{2}(1) + \mathbf{1}$ | $\mathbf{1} = 3 - 2(1)$

$\mathbf{2} = \mathbf{1}(2)$

Now reverse the process using the equations on the right.

$1 = 3 - 2(1)$

$1 = 3 - (8 - 3(2))(1) = 3 - (8 - (3(2)) = 3(3) - 8$

$1 = (11 - 8(1))(3) - 8 = 11(3) - 8(4) = 11(3) + 8(-4)$

## Finding an a multiplicative inverse using the Extended Euclidean Algorithm

- **Example: Find the multiplicative inverse of 8 mod 11, using the Euclidean Algorithm.**

Solution. We'll organize our work carefully. We'll do the Euclidean Algorithm in the left column. It will verify that gcd(8, 11) = 1. Then we'll solve for the remainders in the right column, before back solving:

$\mathbf{11} = \mathbf{8}(1) + \mathbf{3}$ | $\mathbf{3} = 11 - 8(1)$

$\mathbf{8} = \mathbf{3}(2) + \mathbf{2}$ | $\mathbf{2} = 8 - 3(2)$

$\mathbf{3} = \mathbf{2}(1) + \mathbf{1}$ | $\mathbf{1} = 3 - 2(1)$

$\mathbf{2} = \mathbf{1}(2)$

Now reverse the process using the equations on the right.

$1 = 3 - 2(1)$

$1 = 3 - (8 - 3(2))(1) = 3 - (8 - (3(2)) = 3(3) - 8$

$1 = (11 - 8(1))(3) - 8 = 11(3) - 8(4) = 11(3) + 8(-4)$

Therefore $1 \equiv 8(-4) \bmod 11$

This can be written as

$1 \equiv 8(7) \bmod 11$ **Hence 7 is the inverse of 8 mod 11**

## Division in Zn

- Example: Compute 10/8 in $Z_{11}$

## Division in Zn

- **Example: Compute 10/8 in $\mathbf{Z}_{11}$**

10/8 = 10 *1/8 = 10* 7= 70 = 4 in $\mathbf{Z}_{11}$

## Modular Inversion using Fermat's Little theorem

- **Fermat Little Theorem:** Let p be a prime

$$\forall x \in (Z_p)^* : \quad x^{p-1} = 1 \text{ in } Z_p$$

Where $(Z_N)^*$ = (set of invertible elements in $Z_N$)

- Example: p=5. $3^4 = 81 = 1$ in $Z_5$

## Modular Inversion using Fermat's Little theorem

- **Quiz**

How to use Fermat' s theorem to find a modular inverse in $Z_p$

Theorem**:** Let p be a prime

$$\forall x \in (Z_p)^* : \quad x^{p-1} = 1 \text{ in } Z_p$$

- **Solution:**

$\forall x \in (Z_p)^*$ $x \in (Z_p)^* \Rightarrow x \cdot x^{p-2} = 1 \Rightarrow x^{-1} = x^{p-2}$ in $Z_p$

another way to compute inverses, but less efficient than Euclid

time = $O(n^3)$

## Modular Inversion using Fermat's Little theorem

- **Example**: Compute 10/8 in $Z_{11}$

## Modular Inversion using Fermat's Little theorem

- **Example**: Compute 10/8 in $Z_{11}$

11 is a prime number so we can use Fermat little theorem to compute inverses

$10/8 = 10 * 1/8 = 10 * 8^{-1}$

$8^{-1} = 8^{11-2} = 8^9 = 134217728 = 7$ in $Z_{11}$ using Fermat little theorem

Therefore $10/8 = 10*7 = 70 = 4$ in $Z_{11}$

## Computing roots in Zp

- **Definition**: Let p be a prime and $c \in Z_p$ . Let $x \in Z_p$ s.t. $x^e = c$ in $Z_p$ . **x is called an e'th root of c .**

- **Examples**:

$7^{1/3} = 6$ in $Z_{11}$

$3^{1/2} = 5$ in $Z_{11}$

$1^{1/3} = 1$ in $Z_{11}$

$2^{1/2} = ?$

When does $\mathbf{c^{1/e}}$ **in** $\mathbf{Z_p}$ exist? Can we compute it efficiently?

## Computing roots in Zp (gcd( e , p-1 ) = 1)

- **Case 1: if gcd( e , p-1 ) = 1**
  - **Theorem: if gcd(e, p-1)=1, d is the inverse of e in $Z_{p-1}$ ( d = $e^{-1}$ in $Z_{p-1}$ ) than $c^{1/e} = c^d$ in $Z_p$**
  - **To compute the eth root of C in in this case:**

1. Find the modular inverse of e in $Z_{p-1}$ (let us call it d)
2. Compute $c^{1/e} = c^d$ in $Z_p$

- **Case 2 : if gcd( e , p-1 ) ≠ 1**
  In this case the problem of finding an inverse is harder (e.g. computing the square root (e=2) modular odd prime, in this gcd( 2, p-1) ≠ 1)

## Computing roots in Zp (gcd( e , p-1 ) = 1)

**Example** : Compute $\mathbf{7}^{1/11}$ in $Z_{17}$

## Computing roots in Zp (gcd( e , p-1 ) = 1)

**Example** : Compute $\mathbf{7^{1/11}}$ in $Z_{17}$

- **First we check if gcd(e, p-1) = 1**
- **gcd(e, p-1) = gcd(11, 16) therefore :**

## Computing roots in Zp (gcd( e , p-1 ) = 1)

**Example** : Compute $\mathbf{7^{1/11}}$ in $Z_{17}$

- **First we check if gcd(e, p-1) = 1**
- **gcd(p-1) = gcd(11, 16) therefore :**

| | |
|---|---|
| 16 = 11 + 5 | \| 5 = 16 − 11 (a) |
| 11= 2*5 + 1 | \| 1= 11 −2* 5 (b) |
| 5 = 5(1) | |

## Computing roots in Zp (gcd( e , p-1 ) = 1)

**Example** : Compute $\mathbf{7^{1/11}}$ in $Z_{17}$

- **First we check if gcd(e, p-1) = 1**
- **gcd(e, p-1) = gcd(11, 16) therefore :**

16 = 11 + 5 | 5 = 16 − 11 (a)

11= 2*5 + 1 | 1= 11 −2* 5 (b)

5 = 5(1)

Therefore gcd(11, 16) = gcd(1,0) =1 so the previous theorem applies

Now reverse the process using the equations on the right in order to compute the inverse of 11 in $Z_{16}$


## Computing roots in Zp (gcd( e , p-1 ) = 1)

**Example** : Compute $\mathbf{7^{1/11}}$ in $Z_{17}$

- **First we check if gcd(e, p-1) = 1**
- **gcd(e, p-1) = gcd(11, 16) therefore :**

16 = 11 + 5 | 5 = 16 − 11 (a)

11= 2*5 + 1 | 1= 11 −2* 5 (b)

5 = 5(1)

Therefore gcd(11, 16) = gcd(1,0) =1 so the previous theorem applies

Now reverse the process using the equations on the right in order to compute the inverse of 11 in $Z_{16}$


1=11- 2*5

1= 11- 2(16-11)

1= 3(11) -2(16)

Therefore 1 ≡ 3*11 mod 16

Hence 3 is the multiplicative inverse of 11 mod 16

$\mathbf{7^{1/11} = 7^3 = 3}$ in $Z_{17}$

# Computing roots in Zp (gcd( e , p-1 ) ≠1)

- **Special case: Computing the square root in** Zp
  - **Quadratic residue(Q.R)**: An element x in $Z_p$ is said to be a quadratic residue (Q.R.) if it has a square root in $Z_p$
  - **Example**: in $Z_{11}$: $\sqrt{1}=\{1,10\}$, $\sqrt{4}=\{2,9\}$, $\sqrt{9}=\{3,8\}$, $\sqrt{5}=\{4,7\}$, $\sqrt{3}=\{5,6\}$

So in this case: {1,4,9,5,3,0} are quadratic residues

# Computing the square root modular prime

- **How can we tell which elements are Q.R**
  - **Euler's theorem:** Let p be an odd prime, if x in $(Z_p)^*$ is a Q.R. then $x^{(p-1)/2} = 1$ in $Z_p$
  - Example:

    in $Z_{11}$: $1^5$, $2^5$, $3^5$, $4^5$, $5^5$, $6^5$, $7^5$, $8^5$, $9^5$, $10^5$

    = 1 -1 1 1 1, -1, -1, -1, 1, -1

  - This theory is very useful for computing the order of elliptic curve groups

## Computing the square root modular odd prime

- **Case 1: p = 3 (mod 4)**
  - Theorem: if $c \in (Z_p)^*$ is Q.R. Then $\sqrt{\boldsymbol{c}} = \boldsymbol{c}^{\frac{\boldsymbol{p+1}}{\boldsymbol{4}}}$ in $Z_p$
  - **Proof**: $[c^{\frac{p+1}{4}}]^2 = c^{\frac{p+1}{2}} = \underbrace{c^{\frac{p-1}{2}}}_{1} \cdot C = \underbrace{(c^{p-1})^{\frac{1}{2}}}_{1} \cdot C = C$ in $Z_P$
- **Case 2: p = 1 (mod 4)**

  In this case finding the square root can be done using a randomized algorithm with run time ≈ $O(\log^3 p)$.

## Computing the square root modular odd prime

- **Example for case 1**

**Compute $\sqrt{6}$ *in* $Z_{43}$ given 6 is a QR *in* $Z_{43}$**

P=43 =3 mod(4)

Therefore $\sqrt{\mathbf{6}} = \mathbf{6}^{\frac{\mathbf{43+1}}{\mathbf{4}}} = \mathbf{6}^{\mathbf{11}} = 36$

Check that 36*36= 6 $Z_{43}$

# Groups

Let G be a non-empty set, and let * be a binary operation on G. This means that for every two points a, b ∈ G, a value a * b is defined.

We say that G is a group if it has the following properties:

1. **CLOSURE**: ∀ a, b ∈ G then (a * b) ∈ G .
2. **ASSOCIATIVITY**: ∀ a, b, c ∈ G then (a * b) * c = a *(b * c).
3. **IDENTITY**: there exists e ∈ G such that a * e = a = e * a for all a ∈ G.
4. **INVERTABILITY**: for every a ∈ G there exists ai ∈ G such that a * ai = e = ai * a.

# Groups

- **Example**:

The numbers systems Z,Q,R,C and Zn are groups under addition, with * = +, e = 0 and ai = −a.

## Groups

- **Quiz**

Let N be a positive integer . Prove that $Z_N$ is a group under addition modulo N.

- **Solution:**

Addition modulo N: a, b $7\longrightarrow$ a + b mod N

- Closure: $a, b \in Z_N \Rightarrow a + b \bmod N \in Z_N$
- Associative:

$((a + b \bmod N) + c) \bmod N = (a + (b + c \bmod N)) \bmod N$

- Identity: $a + 0 \equiv 0 + a \equiv a \pmod N$
- Inverse: Inverse of a is $-a \equiv N - a \pmod N$

## Groups

- **Quiz:**

Prove that $Z^*_{12} = \{1, 5, 7, 11\}$ is a group under multiplication modulo 12

# Groups

- **Solution**:

**Closure**: a, b $\in Z^*_{12} \Rightarrow$ ab mod 12 $\in Z^*_{12}$. That is
gcd(a,12) = gcd(b,12) = 1 $\Rightarrow$ gcd(ab mod 12,12) = 1
Check: 5 · 7 mod 12 = 35 mod 12 = 11 $\in Z^*_{12}$
If a, b $\in Z^*_{12}$ , ab mod 12 can never be 3!
**Associative**: ((ab mod 12)c) mod 12 = (a(bc mod 12)) mod 12
Check:
(5 · 7 mod 12) · 11 mod 12 = (35 mod 12) · 11 mod 12
= 11 · 11 mod 12 = 1
5 · (7 · 11 mod 12) mod 12 = 5 · (77 mod 12) mod 12
= 5 · 5 mod 12 = 1
**Identity**: 1 is the identity element because a · 1 ≡ 1 · a ≡ a (mod 12)for all a.
**Inverse**: $\forall a \in Z^*_{12}\ \exists a-1 \in Z^*_{12}$ such that a · a−1 mod 12 = 1.
Check: 5−1 is the x ∈ Z∗ 12 satisfying 5x ≡ 1 (mod 12)

# Group Order

- The **order of a group** G is its size |G|, meaning the number of elements in it.
- Example: the order of ( $Z_{12}$ , +) is 12
- Quiz: What is the order of ($Z^*_{12}$ , *)

# Abelian Groups

- **Definition** A group G is said to be commutative (or abelian) if $(a* b)= (b *a)$ for all $a, b \in G$ (**commutativity**).

- **Example**: The sets of non-zero elements in Q, R and C are all commutative groups under multiplication

# Groups

- **Definition:** A group **G** is said to be **cyclic** if it has a generator **g**, an element $g \in G$ such that every element $a \in G$ has the form $a= g^i$ (or $ig$ in additive notation) for some integer i.
- **Examples**:
- Z is cyclic, since every element has the form $1+1+\cdots+1$
- However, Q, R and C are not cyclic.

## Euler's generalization of Fermat (1736)

- **Euler's $\phi$ function**

**For an integer N , we define $\phi(N) = |(Z_N)^*|$**

- **Examples:** $\phi(12) = |\{1,5,7,11\}| = 4$

  $\phi(7) = 6$

## Euler's generalization of Fermat

- **How to Compute Euler Totient Function $\varphi(N)$ :**
  - For N= p (p prime) $\varphi(N) = N-1$
  - For N=p.q (p,q prime) $\varphi(N) = (p-1)(q-1)$
- **Examples:**

  $\varphi(37) = 36$

  $\varphi(21) = (3-1)\times(7-1) = 2\times6 = 12$

## Euler's generalization of Fermat

- **Fermat Little Theorem:** Let p be a prime

$$\forall\, x \in (Z_p)^* : \quad x^{p-1} = 1 \text{ in } Z_p$$

- **Euler's Theorem**: a generalisation of Fermat's Theorem

$$\forall\, x \in (Z_N)^* \;:\; x^{\varphi(N)} \bmod N = 1$$

**Example**: $5^{\phi(12)} = 5^4 = 625 = 1$ in $Z_{12}$

**This theorem forms the basis of the RSA cryptosystem**

## Learning Outcomes

**At the end of this lecture you should be able to:**

1. Classify computational problems according to their complexity.
2. Perform computation using modular arithmetic.
3. Discuss a number of intractable problems in modular arithmetic.

## Discrete Logarithm Problem

- **Example**: Compute the $\mathrm{Dlog}_2$ for elements of $Z_{13}$

| | |
|---|---|
| $y = 2^x$: | 1, 2, 3, 4, 5, 6, 7,8,9, 10,11, 12 |
| x=Dlog$_2$(y) : | 0, 1, |

## Discrete Logarithm Problem

- **Example**: Compute the $\mathrm{Dlog}_2$ for elements of $Z_{13}$

| | |
|---|---|
| $y = 2^x$: | 1, 2, 3, 4, 5, 6, 7,8,9, 10,11, 12 |
| x=Dlog$_2$(y) : | 0, 1, 4 |

## Discrete Logarithm Problem

- **Definition** Fix a prime p>2 and g in $(Z_p)^*$ of order q.

  Consider the function: $y \mapsto g^x$ in $Z_p$

  Now, consider the inverse function:

  $$\mathrm{Dlog}_g(g^x) = x \quad \text{where } x \text{ in } \{0, \ldots, q-2\}$$

- Given g, x it is relatively easy to compute y
- Given g, y it is hard to compute x (Discrete log problem)
- Best Known Algorithm is General number field sieve: with a run time of the order: $e^{O(\sqrt[3]{N})}$

## The factoring problem

Gauss (1805): *"The problem of distinguishing prime numbers from composite numbers and of resolving the latter into their prime factors is known to be one of the most important and useful in arithmetic."*

- Best Known Algorithm is number field sieve: with a run time of the order: $e^{O(\sqrt[3]{N})}$
- Current Record: **RSA-768** (232 digits) (two years and 100s of machines)
- Factorizing a 1024-bit integer (309 digits) ?