Summer Research Fellowship Programme, 2025

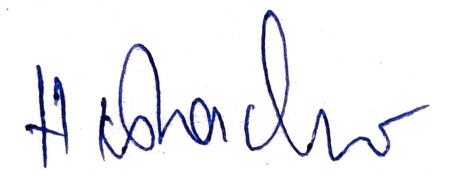
Abstract Algebra

Final Report

Student:
Shivam Raj
Department of Mathematics
Thakur Prasad College, Madhepura(Bihar)

Guide:
Krishna Hanumanthu
Department of Mathematics
CMI, Chennai

Internship Duration: 10 May 2025 – 10 July 2025



Krishna Hanumanthu (Guide)

Abstract

This report explains the main topics we learned during our two-month internship on Group Theory, a central topic in Abstract Algebra. It includes important topics such as groups, subgroups, cyclic groups, permutation groups, homomorphisms, external direct products, normal subgroups, and factor groups. The report also includes several important theorems and examples to help to explain the topics more clearly. We studied these using the book *Contemporary Abstract Algebra* by Joseph A. Gallian and solved many exercises from it. Some of those exercises are also explained in this report.

Along with reading and solving exercises, we also focused on how computers can be used in the field of Group theory. Computers do many things faster and easier, so using them in group theory can help us explore more advanced topics in the future.

As part of this, I wrote several programs related to group theory using Python. Some were suggested by my guide, Prof. Krishna Hanumanthu, and some were based on exercises from the book. The program I wrote includes:

- Finding the inverse, order, and Cayley table of the $U(n)$ and $\mathbb{Z}_n$ groups.
- Finding the order of the group $GL(2, \mathbb{Z}_p)$, and also the inverse and order of its elements.
- Counting the number of elements in each cycle type in the S_n group.
- Finding the order and the Cayley table of the $\mathbb{Z}_m \oplus \mathbb{Z}_n$ group.

The code for these programs can be found at the following link:

https://github.com/shivam-raj12/Internship-Final-Report

Contents

1 Groups **4**

1.1 Exercises Solved 5

1.2 Computational Work 6

2 Subgroups **8**

2.1 Exercises Solved 9

2.2 Computational Work 10

3 Cyclic Groups **12**

3.1 Exercises Solved 13

3.2 Computational Work 13

4 Permutation Groups **14**

4.1 Exercises Solved 14

4.2 Computational Work 15

5 Homomorphisms of Groups **16**

5.1 Homomorphisms 16

5.2 Isomorphisms 17

5.3 Exercises Solved 17

5.4 Computational Work 18

6 Cosets and Lagrange's Theorem **19**

6.1 Exercises Solved 20

7 External Direct Product **21**

7.1 Exercises Solved 21

7.2 Computational Work 22

8 Normal Subgroups and Factor Groups **23**

8.1 Normal Subgroups 23

8.2 Factor Groups 24

8.3 Exercises Solved 24

1 Groups

Definition 1.1. *A **Group** is a non-empty set G together with binary operation $*$ that satisfies the following properties:*

1. ***Closure:*** $\forall\ a, b \in G$ *such that* $a * b \in G$.
2. ***Associative:*** $\forall\ a, b, c \in G$ *such that* $a * (b * c) = (a * b) * c$.
3. ***Identity element:*** $\exists$ *a unique element* $e \in G$ *such that* $a * e = e * a = a$.
4. ***Inverse element:*** $\forall\ a \in G,\ \exists$ *a unique element* $a^{-1} \in G$ *such that* $a * a^{-1} = a^{-1} * a = e$.

Definition 1.2. *An **Abelian Group** is a group which also satisfies the Distributive law property, that is,* $\forall a, b \in G$ *such that* $a * b = b * a$.

Example 1.1. $(\mathbb{Z}, +), (\mathbb{Q}, +), (\mathbb{R}, +), (\mathbb{C}, +)$ *are all examples of the Abelian Group.*

Similarly, $(\mathbb{Z}^*, \cdot), (\mathbb{Q}^*, \cdot), (\mathbb{R}^*, \cdot), (\mathbb{C}^*, \cdot)$ *are also examples of the Abelian Group.*

Theorem 1.1. *There is only a unique identity element in any group.*

Proof. If possible, assume that there exist two identity elements e_1 and e_2 in a group.

By the definition of identity:

- Since e_1 is an identity, it must hold that:

$$e_1 * e_2 = e_2 \quad \text{(because } e_1 \text{ acts as identity on } e_2\text{)}$$

- Since e_2 is an identity, it must hold that:

$$e_1 * e_2 = e_1 \quad \text{(because } e_2 \text{ acts as identity on } e_1\text{)}$$

Now we compare:

$$e_1 * e_2 = e_2 \tag{1}$$
$$e_1 * e_2 = e_1 \tag{2}$$

From eq^n (1) and (2), we get:

$$e_1 = e_2$$

Thus, the identity element in any group is unique. □

Theorem 1.2. $\forall a \in G, \exists$ *a unique* $b \in G$ *such that* $a * b = b * a = e$.

Proof. If possible, assume that there exist two inverses b and c such that

$$a * b = b * a = e \tag{3}$$
$$a * c = c * a = e \tag{4}$$

We will show $b = c$

Now,

$$\begin{aligned} b &= b * e \\ &= b * (a * c) \quad [\text{from } eq^n \ 4\ ] \\ &= (b * a) * c \quad [\text{Associative law}] \\ &= e * c \qquad [\text{from } eq^n \ 3] \\ &= c \end{aligned}$$

.

Thus, inverse of an element is unique. □

1.1 Exercises Solved

Some of the exercises which I solved from the *Contemporary Abstract Algebra* by Joseph A. Gallian book are:

◆ **Exercise 1.** Prove that in a group, $(a^{-1})^{-1} = a$ for all a.

Solution: Let $b = (a^{-1})^{-1}$.

We want to show that $b = a$.

By the definition of an inverse:

$$a^{-1} \cdot b = e.$$

But we also know that:

$$a \cdot a^{-1} = e.$$

So, a is the inverse of a^{-1}, which means:

$$a = (a^{-1})^{-1}.$$

Therefore,

$$b = a = (a^{-1})^{-1}.$$

Hence,

$$a = (a^{-1})^{-1}.$$

◆ **Exercise 2.** Prove that a group G is Abelian if and only if $(ab)^{-1} = a^{-1}b^{-1}$ for all a and b in G.

Solution: We have to prove two things:

1. G is abelian $\implies (ab)^{-1} = a^{-1}b^{-1}$
2. $(ab)^{-1} = a^{-1}b^{-1} \implies G$ is Abelian group.

Proof (i) We know that

$$(ab)^{-1} = b^{-1}a^{-1}$$

and since G is Abelian, we also have

$$b^{-1}a^{-1} = a^{-1}b^{-1}$$

Combining, both the equation we have:

$$(ab)^{-1} = a^{-1}b^{-1}$$

(ii) We have

$$\begin{aligned} (ab)^{-1} &= a^{-1}b^{-1} \\ \Rightarrow b^{-1}a^{-1} &= a^{-1}b^{-1} \\ \Rightarrow (b^{-1}a^{-1})^{-1} &= (a^{-1}b^{-1})^{-1} \quad \text{[Taking inverse both side]} \\ \Rightarrow ab &= ba \end{aligned}$$

Therefore, G is abelian.

Hence, we have G is abelian $\iff (ab)^{-1} = a^{-1}b^{-1} \ \forall \ a, b \in G$.

1.2 Computational Work

Some of the computer exercises which I developed to help us in the study of groups are:

1. This program prints the order of the $U(n) = \{a \in \mathbb{Z}_n \mid gcd(a, n) = 1\}$ group.

   How it works

   This program uses a mathematical formula to calculate the order of the $U(n)$ group.
   For example, let's say the prime factorization of $n = p_1^{a_1} \cdot p_2^{a_2} \cdots p_k^{a_k}$.
   We have,

$$| \, U(n) \, | = n \times \left(1 - \frac{1}{p_1}\right) \times \left(1 - \frac{1}{p_2}\right) \times \cdots \times \left(1 - \frac{1}{p_k}\right)$$

The program takes an integer n as input and calculates the prime divisors of n (in this case it is $p_1, p_2, \cdots, p_k$). Then it multiplies n by $\left(1 - \frac{1}{p_x}\right)$ where $x \in \{p_1, p_2, \cdots, p_k\}$

The code for this program is available here[1].

2. This program prints the elements of the $U(n)$ group.

How it works

The program takes an integer n as input and iterates through all integers a from 1 to $(n-1)$. For each a, it calculates the greatest common divisor of a and n using the built-in GCD function. If $gcd(a, n) = 1$, the number is added to the list of elements of $U(n)$.

The code for this program is available here[2].

3. This program prints the Cayley table for the $U(n)$ group.

How it works

This program uses the external Python library `tabulate` to print the Cayley table in a well-formatted tabular structure.

The program takes an integer n as input and stores all the elements of the $U(n)$ group from the function defined in Point 2.

Now, the program uses two nested `for` loops to iterate through all elements of the group. The outer loop selects each element a from the group elements list, and the inner loop iterates over each element b, computing the group operation $a * b$ for every pair. The result of each operation is stored in a structured format, which is later printed as a table using the `tabulate` library.

The code for this program is available here[3].

4. This program prints the Cayley table for the $\mathbb{Z}_n = \{a \in \mathbb{Z} \mid 0 \leq a < n\}$ group.

[1] https://github.com/shivam-raj12/Internship-Final-Report/blob/master/U_n_Group/U(n)_Size.py

[2] https://github.com/shivam-raj12/Internship-Final-Report/blob/master/U_n_Group/U_n_Elements.py

[3] https://github.com/shivam-raj12/Internship-Final-Report/blob/master/U_n_Group/Cayley_Table.py

How it works

This program uses the same working principle as the Cayley table program for the $U(n)$ group (Point 3). The only difference is that this program uses $\mathbb{Z}_n$ group elements instead of $U(n)$ group.

The code for this program is available here[4].

2 Subgroups

Definition 2.1. *A subset H of a group G is called a **subgroup** of G if H is itself a group under the same operation as G.*

Definition 2.2. *The **order of an element** $a \in G$ is the smallest positive integer n such that $a^n = e$, where e is the identity element of the group G.*

Example 2.1. $(\mathbb{Z}, +) \leq (\mathbb{Q}, +) \leq (\mathbb{R}, +) \leq (\mathbb{C}, +)$ *are all examples of a subgroup.*

Theorem 2.1. *Let G be a group and H be a non-empty subset of G. Then H is a subgroup of G if and only if for all $a, b \in H, ab^{-1} \in H$.*

Proof. ($\Rightarrow$) Assume $H \leq G$, i.e., H is a subgroup of G. Let $a, b \in H$. Since H is a subgroup, it is closed under taking inverses and the group operation. Thus:

$$b^{-1} \in H \quad \text{and} \quad ab^{-1} \in H$$

because both a and b^{-1} are in H, and H is closed under multiplication. Hence, $ab^{-1} \in H$.

($\Leftarrow$) Assume H is a non-empty subset of G, and for all $a, b \in H$, we have $ab^{-1} \in H$. We want to show that H is a subgroup of G.

Since H is non-empty, there exists some $h \in H$.

- **Identity:** Let $a = b \in H$. Then

  $$ab^{-1} = aa^{-1} = e \in H$$

  so the identity element e is in H.

- **Inverses:** Let $a \in H$, and use $e \in H$ (shown above). Then

  $$ea^{-1} = a^{-1} \in H$$

  so H contains inverses.

[4] https://github.com/shivam-raj12/Internship-Final-Report/blob/master/Z_n_Group/Cayley_table.py

- **Closure:** Let $a, b \in H$. Then $b^{-1} \in H$ (since H contains inverses), so:
$$ab = a(b^{-1})^{-1} \in H$$
since $a \in H$, $b^{-1} \in H$, and $ab = a(b^{-1})^{-1} \in H$ by assumption.

Therefore, H is a subgroup of G.

□

2.1 Exercises Solved

◆ **Exercise 1.** Let G be a group, and let $a \in G$. Prove that $C(a) = C(a^{-1})$.

Solution: We will show that $C(a) \subseteq C(a^{-1})$ and $C(a^{-1}) \subseteq C(a)$, which implies $C(a) = C(a^{-1})$.

(1) Show that $C(a) \subseteq C(a^{-1})$:

Let $g \in C(a)$. Then $ga = ag$.

We want to show that $g \in C(a^{-1})$, i.e., $ga^{-1} = a^{-1}g$.

Starting from $ga = ag$, multiply both sides on the right by a^{-1}:
$$gaa^{-1} = aga^{-1} \Rightarrow g = aga^{-1}$$
Now multiply both sides on the left by a^{-1}:
$$a^{-1}g = a^{-1}(aga^{-1}) = (a^{-1}a)ga^{-1} = ga^{-1}$$
Thus, $a^{-1}g = ga^{-1}$, so $g \in C(a^{-1})$.

Hence, $C(a) \subseteq C(a^{-1})$.

(2) Show that $C(a^{-1}) \subseteq C(a)$:

Let $g \in C(a^{-1})$. Then $ga^{-1} = a^{-1}g$.

We want to show that $ga = ag$.

Multiply both sides of $ga^{-1} = a^{-1}g$ on the right by a:
$$ga^{-1}a = a^{-1}ga \Rightarrow g = a^{-1}ga$$
Multiply both sides on the left by a:
$$ag = a(a^{-1}ga) = (aa^{-1})ga = ga$$
So $ag = ga$, i.e., $g \in C(a)$. Hence, $C(a^{-1}) \subseteq C(a)$.

Conclusion: Since both inclusions hold, we conclude that
$$C(a) = C(a^{-1}).$$

2.2 Computational Work

1. This program prints the inverse and order of each element of the $U(n)$ group.

   How it works

   This program uses an external library `tabulate` to print the inverse and order of each element in a tabular structure.

   The program takes an integer n as input and stores all the elements of the $U(n)$ group from the function defined in Section 1; Point 2. These elements form a group under multiplication modulo n.

   Next, the program uses two nested `for` loops to find the multiplicative inverse of each element in $U(n)$.

   - The outer loop selects each element a from the list of group elements.
   - The inner loop iterates over each element b in the group and computes $(a \cdot b) \bmod n$.
   - If the result is 1, then b is the multiplicative inverse of a. The inverse b is stored, and the inner loop is ended using break.

   Then the program determines the order of each element in the group.

   - For each element a, another loop iterates over i from 1 to n (order of an element is always less than or equal to order of the group), computing $a^i \bmod n$.
   - When the result is 1, i is the order of a in the group. The order is stored and the loop breaks.

   After processing all elements, the program prints the results in table format using the `tabulate` library, showing each element along with its inverse and order.

   The code for this program is available here[5].

2. This program prints the inverse and order of each element of the $\mathbb{Z}_n$ group.

   How it works

   Working of this program is same as discussed in Point 1.

   The code for this program is available here[6].

3. This program prints the order of the $GL(2, \mathbb{Z}_p)$ group where p is prime.

[5] https://github.com/shivam-raj12/Internship-Final-Report/blob/master/U_n_Group/Inverse_and_Order.py

[6] https://github.com/shivam-raj12/Internship-Final-Report/blob/master/Z_n_Group/Inverse_and_Order.py

How it works

This program takes an integer p as input and calculates the order of the $GL(2, \mathbb{Z}_p)$ group using a mathematical formula.

$$|GL(2, \mathbb{Z}_P)| = \big(p^2 - 1)(p^2 - p\big) \text{ where } p \text{ is prime.}$$

The code for this program is available here[7].

4. This program prints the inverse of an element of $GL(2, \mathbb{Z}_p)$ group.

How it works

This program takes 5 integer value as input a, b, c, d, p where a, b, c, d represents the element of matrix in the below format:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

and p (must be prime) represent the value of $\mathbb{Z}_p$.

Now, the inverse is calculated using the formula:

$$A^{-1} = (detA)^{-1} \times \begin{bmatrix} a & b \\ c & d \end{bmatrix} \bmod p$$

where $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ and $(detA)^{-1}$ is the modular inverse of the determinant in $\mathbb{Z}_p$.

The code for this program is available here[8].

5. This program prints the order of an element of $GL(2, \mathbb{Z}_p)$ group.

How it works

This program takes 5 integer value as input a, b, c, d, p where a, b, c, d represents the element of matrix in the below format:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

and p (must be prime) represent the value of $\mathbb{Z}_p$.

The program uses the fact that the order of an element in a group must divide the order of the group. Therefore, it starts a `for` loop from 1 to $|\mathrm{GL}(2, \mathbb{Z}_p)|$, which is the total number of invertible 2×2 matrices over $\mathbb{Z}_p$. The group order is given by:

$$|\mathrm{GL}(2, \mathbb{Z}_p)| = (p^2 - 1)(p^2 - p).$$

[7] https://github.com/shivam-raj12/Internship-Final-Report/blob/master/Matrix/order_of_group.py

[8] https://github.com/shivam-raj12/Internship-Final-Report/blob/master/Matrix/inverse_of_element.py

In each iteration, the matrix is multiplied by itself (modulo p) until the identity matrix is reached. The smallest such exponent n is the order of the matrix.

If no such n is found within the group order, the program concludes that the matrix is not in $\mathrm{GL}(2, \mathbb{Z}_p)$, i.e., it is not invertible modulo p.

The code for this program is available here[9].

3 Cyclic Groups

Definition 3.1. *A group G is called **cyclic** if $\exists$ an element $g \in G$ such that*

$$G = \{g^n \mid n \in \mathbb{N}\}.$$

The element g is called a generator of the group, and the group is said to be generated by g.

Example 3.1. *$(\mathbb{Z}, +)$ is a cyclic group. Both 1 and -1 are generators.*

Theorem 3.1. *Prove that $(\mathbb{Q}, +)$ is not a cyclic group.*

Proof. Suppose for contradiction that there is an element q that generates $(\mathbb{Q}, +)$. Then

$$\mathbb{Q} = \langle q \rangle = \{nq \mid n \in \mathbb{Z}\}$$

Let $q = \dfrac{a}{b}$ in lowest terms, with $a, b \in \mathbb{Z}$, $b > 0$, and $\gcd(a, b) = 1$.

Then every element in $\langle q \rangle$ has the form $\dfrac{na}{b}$, and hence all elements have denominator dividing b.

However, the rational number $\dfrac{1}{b+1} \in \mathbb{Q}$ cannot be written as $\dfrac{na}{b}$, since its denominator $b+1$ does not divide b.

Thus, $\dfrac{1}{b+1} \notin \langle q \rangle$, contradicting the assumption that $\langle q \rangle = \mathbb{Q}$.

Therefore, $(\mathbb{Q}, +)$ is not a cyclic group.

□

[9] https://github.com/shivam-raj12/Internship-Final-Report/blob/master/Matrix/order_of_element.py

3.1 Exercises Solved

◆ **Exercise 1.** Prove that a group of order 3 must be cyclic.

Solution: By Lagrange's Theorem, the order of any element $g \in G$ must divide the order of the group. Therefore, the possible orders of elements in G are the divisors of 3, i.e., 1 or 3.

Let e denote the identity element in G. Clearly, e has order 1.

Now consider any element $g \in G$ such that $g \neq e$. Since $g \in G$ and $g \neq e$, and $|G| = 3$, there are only two non-identity elements in G.

The order of g must divide 3, and it cannot be 1 (since $g \neq e$), so the order of g must be 3.

Therefore, the subgroup generated by g is:

$$\langle g \rangle = \{e, g, g^2\},$$

which contains all three elements of G. So,

$$\langle g \rangle = G,$$

which shows that G is generated by a single element.

Hence, G is cyclic.

3.2 Computational Work

1. This program prints the cyclic subgroups generated by each element of the $U(n)$ group.

   How it works

   This program uses an external library `tabulate` to print the subgroups generated by each element in a well-structured table format.

   This program takes an integer value n as input for the $U(n)$ group and obtains a list of all the elements of the $U(n)$ group.

   Now, it starts iterating for x through all the elements of the $U(n)$ group. Inside this loop, the program starts another loop i from 0 to $|U(n)|$ and calculates $x^i \bmod n$ and stores its value in a structured way.

   When it completes the iteration for all elements in $U(n)$, the program prints all the generated cyclic subgroups from each element using the library `tabulate`.

   The code for this program is available here[10].

[10] https://github.com/shivam-raj12/Internship-Final-Report/blob/master/U_n_Group/Cyclic_Subgroups.py

2. This program prints the cyclic subgroups generated by each element of the $\mathbb{Z}_n$ group.

How it works

The working principle of this program is same as the program for the cyclic subgroups generated by each element of the $U(n)$ group. (Point 1)

The code for this program is available here[11].

4 Permutation Groups

Definition 4.1. *A **permutation group** on a set S is a group consisting of bijective functions (called permutations) from S to itself, with the group operation being composition of functions.*

Example 4.1. *Let $S = \{1, 2, 3\}$. The set of all permutations of S forms a group under composition, called the symmetric group on 3 elements, denoted by S_3.*

The group S_3 has 6 elements:

$$S_3 = \{e, (1\ 2), (2\ 3), (1\ 3), (1\ 2\ 3), (1\ 3\ 2)\}$$

Definition 4.2. *The **alternating group** A_n is the group of all even permutations of n elements, i.e., permutations that can be written as a product of an even number of transpositions.*

It is a subgroup of the symmetric group S_n with order $\frac{n!}{2}$.

Example 4.2. $A_3 = \{e, (1\ 2\ 3), (1\ 3\ 2)\}$

Definition 4.3. *Let G be a group of permutations of a set S. For each i in S,*

$$\text{stab}_G(i) = \{\phi \in G \mid \phi(i) = i\}.$$

*We call $\text{stab}_G(i)$ the **stabilizer of i in G**.*

4.1 Exercises Solved

◆ **Exercise 1.** If α is even, prove that α^{-1} is even. If α is odd, prove that α^{-1} is odd.

Solution: Let $\alpha \in S_n$ be a permutation.

[11] https://github.com/shivam-raj12/Internship-Final-Report/blob/master/Z_n_Group/Cyclic_subgroup.py

A permutation is **even** if it can be expressed as a product of an even number of transpositions, and **odd** otherwise. The sign function $\text{sgn} : S_n \to \{-1, 1\}$ satisfies:

$$\text{sgn}(\alpha\beta) = \text{sgn}(\alpha)\,\text{sgn}(\beta)$$

Since $\alpha\alpha^{-1} = e$, we have:

$$\text{sgn}(\alpha\alpha^{-1}) = \text{sgn}(\alpha)\,\text{sgn}(\alpha^{-1}) = \text{sgn}(e) = 1$$

$$\Rightarrow \text{sgn}(\alpha^{-1}) = \text{sgn}(\alpha)$$

Conclusion: The inverse of an even permutation is even and the inverse of an odd permutation is odd.

◆ **Exercise 2.** Prove that S_n is non-Abelian for all $n \geq 3$.

Solution: Consider $n \geq 3$. Define the following permutations in S_n:

$$\alpha = (1\ 2), \quad \beta = (2\ 3)$$

Compute the compositions:

$$\alpha\beta = (1\ 2)(2\ 3) = (1\ 2\ 3), \quad \beta\alpha = (2\ 3)(1\ 2) = (1\ 3\ 2)$$

Since $\alpha\beta \neq \beta\alpha$, we conclude that composition in S_n is not commutative.

Therefore, S_n is non-abelian for all $n \geq 3$.

4.2 Computational Work

1. This program prints the stabilizer of a point in a permutation group acting on a set S. (See definition 4.3)

   How it works

   This program first takes two inputs:

   - An integer n, which defines the Symmetric group S_n.
   - An element of the set $\{1, 2, 3, \cdots, n\}$ whose stabilizer is to be computed.

   This program uses `SymmetricGroup` from the `sympy` library and its `stabilizer` method to find all permutations that fix that element. The stabilizer elements are generated with `generate_dimino()` and printed in cycle notation.

   The code for this program is available here[12].

2. This program prints the number of elements in each cycle of a symmetric group.

[12] https://github.com/shivam-raj12/Internship-Final-Report/blob/master/Permutation/stabilizer.py

How it works

This program takes a positive integer input representing the size of the set for the symmetric group.

Now, the program calls a function that I defined to generate a list of all possible cycles in the symmetric group S_n. It iterates through each cycle and calls another function that I wrote to determine how many elements are in each cycle. After processing all cycles, the results are displayed in a neatly formatted table using the `tabulate` library.

The code for this program is available here[13].

5 Homomorphisms of Groups

5.1 Homomorphisms

Definition 5.1.1. *Let $(G, *)$ and $(H, \circ)$ be two groups, and let f be a function from G to H. Then f is called a* ***homomorphism*** *of G into H if $\forall\ a, b \in G$,*

$$f(a * b) = f(a) \circ f(b).$$

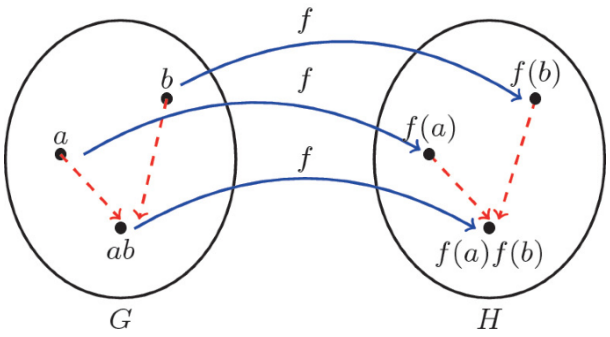


Figure 1: Homomorphism diagram illustrating a group homomorphism $f : G \to H$.

Example 5.1.1. *Let $G = (\mathbb{Z}, +)$ and $H = (\{1, -1\}, \cdot)$.*

Define $f : \mathbb{Z} \to \{1, -1\}$

$$f(n) = \begin{cases} 1 & \text{if } n \text{ is even} \\ -1 & \text{if } n \text{ is odd} \end{cases}$$

Here f is a homomorphism from G to H.

[13] https://github.com/shivam-raj12/Internship-Final-Report/blob/master/Permutation/cycle_elements.py

Definition 5.1.2. *Let $\phi : A \to B$ be a homomorphism between two groups A and B.*

*The **image of** ϕ denoted by* $\operatorname{Im}\phi$ *is defined as:*

$$\operatorname{Im}\phi = \{\phi(a) \mid a \in A\} \subseteq B$$

Definition 5.1.3. *Let $\phi : A \to B$ be a homomorphism between two groups A and B.*

*The **kernel of** ϕ denoted by* $\ker\phi$ *is defined as:*

$$\operatorname{Im}\phi = \{x \in G \mid \phi(x) = e_B, \text{ the identity element of } B\}.$$

Definition 5.1.4. *Let G and G_1 be two groups and $f : G \to G_1$ be a homomorphism of groups.*

*(i) f is called a **monomorphism**, if f is an injective function (one-one).*

*(ii) f is called an **epimorphism**, if f is a surjective function (onto).*

5.2 Isomorphisms

Definition 5.2.1. *A homomorphism f from a group G to a group G_1 is called an **isomorphism** if the mapping $f : G \to G_1$ is a bijective mapping.*

Example 5.2.1. *Let $G = (\mathbb{R}, +)$ and $H = (\mathbb{R}^+, \cdot)$.*

Define $f : \mathbb{R} \to \mathbb{R}^+$ by $f(n) = e^n \ \forall\ n \in \mathbb{R}$.

Here f is an isomorphism from G to H.

Definition 5.2.2. *An **automorphism** of G is a group isomorphism from G to itself.*

Example 5.2.2. *Let $G = (\mathbb{C}, +)$.*

Define $f : \mathbb{G} \to \mathbb{G}$ by $f(a + bi) = a - bi$.

Here f is an automorphism from G to G.

5.3 Exercises Solved

◆ **Exercise 1.** Show that there does not exist any isomorphism from the group $(\mathbb{R}, +)$ to the group $(\mathbb{R}^*, \cdot)$.

Solution: Observe that 0 is the identity element of the group $(\mathbb{R}, +)$. Hence. no nonzero element of $\mathbb{R}$ is of finite order.

Now for the group $(\mathbb{R}^*, \cdot)$, 1 is the identity element and -1 is an element of order 2.

So, if there exists an isomorphism $f : \mathbb{R} \to \mathbb{R}^*$, there should exist $a \in \mathbb{R}$ such that $o(a) = 2$. But this is not the case.

Hence, there is no ismomorphism from the group $(\mathbb{R}, +)$ to the group $(\mathbb{R}^*, \cdot)$.

◆ **Exercise 2.** Let G, H and K be three groups. If $G \simeq H$ and $H \simeq K$, then prove that $G \simeq K$.

Solution: Since $G \simeq H$, there exists an isomorphism $f : G \to H$, and since $H \simeq K$, there exists an isomorphism $g : H \to K$.

Define a map $\varphi : G \to K$ by $\varphi = g \circ f$, i.e.,

$$\varphi(x) = g(f(x)) \quad \text{for all } x \in G.$$

We will show that φ is an isomorphism from G to K.

- **Homomorphism:** Since both f and g are homomorphisms, for all $x, y \in G$,
$$\varphi(xy) = g(f(xy)) = g(f(x)f(y)) = g(f(x))g(f(y)) = \varphi(x)\varphi(y).$$
Hence, φ is a homomorphism.
- **Injective:** Since f and g are injective, suppose $\varphi(x_1) = \varphi(x_2)$. Then:
$$g(f(x_1)) = g(f(x_2)) \Rightarrow f(x_1) = f(x_2) \Rightarrow x_1 = x_2.$$
So φ is injective.
- **Surjective:** Let $z \in K$. Since g is surjective, there exists $y \in H$ such that $g(y) = z$. Since f is surjective, there exists $x \in G$ such that $f(x) = y$. Then:
$$\varphi(x) = g(f(x)) = g(y) = z.$$
Hence, φ is surjective.

Since φ is a bijective homomorphism, it is an isomorphism. Therefore, $G \simeq K$.

5.4 Computational Work

1. This program computes the order of $\mathrm{Aut}(D_n)$.

   How it works

   This program takes an integer n as input for the size of D_n group.

   Now, it calculates the order of $\mathrm{Aut}(D_n)$ using a mathematical formula:

   $$|\mathrm{Aut}(D_n)| = n \cdot \phi(n)$$

   where $\phi(n)$ is Euler's totient function (the number of integers between 1 and n that are coprime to n).

   The code for this program is available here[14].

[14] https://github.com/shivam-raj12/Internship-Final-Report/blob/master/isomorphism.py

6 Cosets and Lagrange's Theorem

Definition 6.1. *Let H be a subgroup of a group G. If $a \in G$, the subset*

$$aH = \{ah \mid h \in H\}$$

is called a ***left coset*** *of H in G. Similarly,*

$$Ha = \{ha \mid h \in H\}$$

is called a ***right coset*** *of H in G.*

Example 6.1. *Let $G = (\mathbb{Z}, +)$ and $H = 5\mathbb{Z} = \{5n \mid n \in \mathbb{Z}\}$.*

The left cosets of $H = 5\mathbb{Z}$ in $\mathbb{Z}$ are

$$0 + 5\mathbb{Z}, 1 + 5\mathbb{Z}, 2 + 5\mathbb{Z}, 3 + 5\mathbb{Z}, 4 + 5\mathbb{Z}.$$

Theorem 6.1. *Let H be a subgroup of a group G. If $a \in G$, then $|aH| = |H| = |Ha|$.*

Proof. To show that $|H| = |aH|$, we show that there exists a bijective mapping of H onto aH.

Consider the function $f : H \to aH$ given by $f(h) = ah$.

- **Injective:** Suppose $f(h_1) = f(h_2)$. Then $ah_1 = ah_2 \Rightarrow h_1 = h_2$, since left multiplication by a is a bijection in a group.
- **Surjective:** For any element $ah \in aH$, there exists $h \in H$ such that $f(h) = ah$.

Hence, f is a bijection and, therefore, $|aH| = |H|$.

Similarly, define $g : H \to Ha$ by $g(h) = ha$. A similar argument shows that g is also a bijection, so $|Ha| = |H|$.

Thus,

$$|aH| = |H| = |Ha|.$$

□

Theorem 6.2 (Lagrange's Theorem)**.** *If G is a finite group and H is a subgroup of G, then $|H|$ divides $|G|$. Moreover,*

$$|G| = [G : H]|H|$$

Proof. Since G is a finite group, $[G : H]$ is finite.

Let $[G : H] = r$.

This implies that there exist r distinct left cosets $a_1H, a_2H, \cdots, a_rH$ of H in G.

Since distinct left cosets are mutually disjoint, we have

$$\begin{aligned} |G| &= \left|\bigcup_{i=1}^{r} a_iH\right| = |a_1H| + |a_2H| + \cdots + |a_rH| \\ &= \underbrace{|H| + |H| + \cdots + |H|}_{r \text{ times}} \quad [\text{by Theorem 6.1, } |a_iH| = |H|] \\ &= r|H| = [G : H]|H|. \end{aligned}$$

Hence the theorem. □

Theorem 6.3 (Fermat's Little Theorem)**.** *Let p be a prime integer and a be an integer such that p does not divide a. Then $a^{p-1} \equiv 1(\mathrm{mod}p)$.*

Proof. For the prime integer p, $U_p = \{[a] \in \mathbb{Z}_p \mid gcd(a, p) = 1\}$. Then $U_p = \mathbb{Z}_p \setminus [0]$ is a group of order $p - 1$.

Let a be an integer such that p does not divide a. Then $[a]$ is a nonzero element of $\mathbb{Z}_p$ and so $[a] \in U_p$.

Thus $[a]^{p-1} = [1]$, i.e., $[a^{p-1}] = [1]$.

Hence $a^{p-1} \equiv 1(\mathrm{mod}p)$. □

6.1 Exercises Solved

◆ **Exercise 1.** Every group of prime order is cyclic.

Solution: Let $|G| = p$.

$\because$ $p > 1$, G has an element $a \neq e$. Let H be a subgroup $\langle a \rangle = \{a^n \mid n \in \mathbb{Z}\}$. By Lagrange's theorem $|\langle a \rangle|$ divides p.

Hence $|\langle a \rangle| = 1$ or p. Since $a \neq e, |\langle a \rangle| \neq 1$ and so $|\langle a \rangle| = p$. Now $|\langle a \rangle| \subseteq G$ and $|\langle a \rangle| = |G| = p$.

Hence, $G = \langle a \rangle$. This shows that G is a cyclic group.

7 External Direct Product

Definition 7.1. *Let $(G, *)$ and $(H, \circ)$ be two groups. The* ***external direct product*** *of G and H, denoted by $G \oplus H$, is the group consisting of all ordered pairs (g, h) where $g \in G$ and $h \in H$, with the group operation define component-wise:*

$$(g_1, h_1) \cdot (g_2, h_2) = (g_1 * g_2, h_1 \circ h_2)$$

Example 7.1. *$U(8) = \{1, 3, 5, 7\}$ and $U(10) = \{1, 3, 7, 9\}$*

$$\begin{aligned} \therefore U(8) \oplus U(10) = \{&(1,1), (1,3), (1,7), (1,9), \\ &(3,1), (3,3), (3,7), (3,9), \\ &(5,1), (5,3), (5,7), (5,9), \\ &(7,1), (7,3), (7,7), (7,9)\} \end{aligned}$$

7.1 Exercises Solved

◆ **Exercise 1.** Show that the direct product $\mathbb{Z} \oplus \mathbb{Z}$ is not a cyclic group.

Solution: Suppose $\mathbb{Z} \oplus \mathbb{Z}$ is a cyclic group. Let (n, m) be a generator of $\mathbb{Z} \oplus \mathbb{Z}$.

We have $(1, 0), (0, 1) \in \mathbb{Z} \oplus \mathbb{Z}$. Thus, there are integers r, s such that $(1, 0) = r(n, m)$ and $(0, 1) = s(n, m)$. But then $rn = 1$ and $sn = 0$, together imply that $s = 0$ (as $n \neq 0$, since $rn = 1$), which is a contradiction, since $sm = 1$.

So, it follows that $\mathbb{Z} \oplus \mathbb{Z}$ is not a cyclic group.

◆ **Exercise 2.** Let H and K be two finite groups. Prove that $o(a, b) = \text{lcm}\{o(a), o(b)\}$, where $(a, b) \in H \oplus K$.

Solution: Let $o(a) = m$ and $o(b) = n$, which means:

$$a^m = e_H, \quad b^n = e_K.$$

We define the order of the element (a, b) in $H \oplus K$ as:

$$o(a, b) = \min\left\{k \in \mathbb{N} : (a, b)^k = (e_H, e_K)\right\}.$$

Now observe that:

$$(a, b)^k = (a^k, b^k).$$

So:

$$(a, b)^k = (e_H, e_K) \iff a^k = e_H \text{ and } b^k = e_K.$$

This implies that k must be a common multiple of $o(a)$ and $o(b)$, and hence:

$$o(a, b) = \text{lcm}(o(a), o(b)).$$

7.2 Computational Work

1. This program prints the elements of $U_s(st) = \{x \in U(st) \mid x \equiv 1 (\mathrm{mod} s)\}$, where s and t are relatively prime.

   How it works

   This program takes two integer values s and t as input. The values s and t must be relatively prime.

   This program iterates from 1 to st and checks whether each number is co-prime to st. If it is, then it further checks whether the number is congruent to 1 mod s. If both conditions are satisfied, the element is printed.

   The code for this program is available here[15].

2. This program computes the elements of the subgroup $U(n)^k = \{x^k \mid x \in U(n)\}$.

   How it works

   This program takes integers n and k as input, computes $x^k \mod n$ for all x co-prime to n, and prints the distinct results. It also prints the order of the subgroup.

   The code for this program is available here[16].

3. This program prints the Cayley table of the $\mathbb{Z}_m \oplus \mathbb{Z}_n$ group.

   How it works

   This program takes two positive integers m and n as input for the $\mathbb{Z}_m \oplus \mathbb{Z}_n$ group. Now, this program uses the same working principle as the program used to print the cayley table of the $U(n)$ group. (Section 1; Point 3)

   The code for this program is available here[17].

4. This program prints the order of all elements of the $\mathbb{Z}_m \oplus \mathbb{Z}_n$ group and also prints whether $\mathbb{Z}_m \oplus \mathbb{Z}_n$ is a cyclic group or not.

[15] https://github.com/shivam-raj12/Internship-Final-Report/blob/master/external_product/program1.py

[16] https://github.com/shivam-raj12/Internship-Final-Report/blob/master/external_product/program2.py

[17] https://github.com/shivam-raj12/Internship-Final-Report/blob/master/external_product/cayley_table.py

How it works

This program takes two positive integers m and n as input for the $\mathbb{Z}_m \oplus \mathbb{Z}_n$ group. Now, this program uses the same working principle as the program used to print the order of all elements of the $U(n)$ group. (Section 2; Point 1). It also checks whether the order of an element is equal to the order of the group. If it is, the program prints that the group is cyclic; otherwise, it prints that the group is not cyclic.

The code for this program is available here[18].

8 Normal Subgroups and Factor Groups

8.1 Normal Subgroups

Definition 8.1.1. *Let G be a group. A subgroup H of G is said to be a* ***normal subgroup*** *of G if $aH = Ha \ \forall \ a \in G$.*

Every subgroup of an Abelian group is a normal subgroup.

Example 8.1.1. *Let $G = S_3$ and $H = \{e, (1\ 2\ 3), (1\ 3\ 2)\}$. Then H is a subgroup of S_3. We know $xH = Hx \ \forall x \in H$. Now,*

$$(1\ 2)H = \{(1\ 2), (1\ 2)(1\ 2\ 3), (1\ 2)(1\ 3\ 2)\} = H(1\ 2).$$

Similarly, $(2\ 3)H = H(2\ 3)$ and $(1\ 3)H = H(1\ 3)$.

Hence, $H \trianglelefteq G$

Theorem 8.1.1. *A subgroup H of G is normal in G if and only if $xHx^{-1} \subseteq H \ \forall x \in G$.*

Proof. ($\Rightarrow$) Suppose $H \trianglelefteq G$. By definition,

$$xHx^{-1} = H \quad \forall x \in G.$$

So in particular $xHx^{-1} \subseteq H$.

($\Leftarrow$) Suppose $xHx^{-1} \subseteq H$ for all $x \in G$. Then, for any $x \in G$, we also have $x^{-1}Hx \subseteq H$, because $x^{-1} \in G$.

Now conjugate this inclusion:

$$x(x^{-1}Hx)x^{-1} = H \subseteq xHx^{-1}.$$

[18] https://github.com/shivam-raj12/Internship-Final-Report/blob/master/external_product/order.py

So we have both:

$$xHx^{-1} \subseteq H \quad \text{and} \quad xHx^{-1} \supseteq H,$$

which implies

$$xHx^{-1} = H \quad \forall x \in G.$$

Therefore, $H \trianglelefteq G$. □

8.2 Factor Groups

Definition 8.2.1. *Let G be a group and $N \trianglelefteq G$. The **factor group** G/N is the set of left cosets of N in G:*

$$G/N = \{gN \mid g \in G\}.$$

The group operation is defined by:

$$(gN)(hN) = (gh)N \quad \text{for all } g, h \in G.$$

Example 8.2.1. *Consider the group $4\mathbb{Z}$ of the group $(\mathbb{Z}, +)$.*

$$\therefore \mathbb{Z}/4\mathbb{Z} = \{0 + 4\mathbb{Z}, 1 + 4\mathbb{Z}, 2 + 4\mathbb{Z}, 3 + 4\mathbb{Z}\}.$$

The Cayley table for the factor group $\mathbb{Z}/4\mathbb{Z}$ is given by:

$*$	$0+4\mathbb{Z}$	$1+4\mathbb{Z}$	$2+4\mathbb{Z}$	$3+4\mathbb{Z}$
$0+4\mathbb{Z}$	$0+4\mathbb{Z}$	$1+4\mathbb{Z}$	$2+4\mathbb{Z}$	$3+4\mathbb{Z}$
$1+4\mathbb{Z}$	$1+4\mathbb{Z}$	$2+4\mathbb{Z}$	$3+4\mathbb{Z}$	$0+4\mathbb{Z}$
$2+4\mathbb{Z}$	$2+4\mathbb{Z}$	$3+4\mathbb{Z}$	$0+4\mathbb{Z}$	$1+4\mathbb{Z}$
$3+4\mathbb{Z}$	$3+4\mathbb{Z}$	$0+4\mathbb{Z}$	$1+4\mathbb{Z}$	$2+4\mathbb{Z}$

8.3 Exercises Solved

◆ **Exercise 1.** Prove that A_n is normal in S_n.

Solution: Recall that:

$$A_n = \{\sigma \in S_n \mid \operatorname{sgn}(\sigma) = +1\}$$

is the set of even permutations, and the sign function

$$\operatorname{sgn} : S_n \to \{\pm 1\}$$

is a group homomorphism.

Let $\sigma \in S_n$, $\tau \in A_n$. Then:

$$\operatorname{sgn}(\sigma\tau\sigma^{-1}) = \operatorname{sgn}(\sigma)\ \operatorname{sgn}(\tau)\ \operatorname{sgn}(\sigma^{-1}) = \operatorname{sgn}(\sigma)\,(+1)\ \operatorname{sgn}(\sigma)^{-1} = +1.$$

So $\sigma\tau\sigma^{-1} \in A_n$, which shows that A_n is closed under conjugation in S_n.

$$\therefore \quad A_n \trianglelefteq S_n.$$

◆ **Exercise 2.** Prove that factor group of a cyclic group is cyclic.

Solution: Let $G = \langle g \rangle$ be a cyclic group, and let $N \trianglelefteq G$ be a normal subgroup. Then every element of G is of the form g^k for some $k \in \mathbb{Z}$, and the elements of the factor group G/N are the cosets $g^k N$.

Now consider the coset $gN \in G/N$. We claim that:

$$G/N = \langle gN \rangle.$$

Indeed, for any $k \in \mathbb{Z}$, we have:

$$(gN)^k = g^k N,$$

which means that every element $g^k N \in G/N$ is a power of gN.

Therefore, G/N is generated by gN, and therefore it is cyclic.

◆ **Exercise 3.** If N and M are normal subgroups of G, prove that NM is also a normal subgroup in G.

Solution: Let N and M be normal subgroups of G. Define

$$NM = \{nm \mid n \in N,\, m \in M\}.$$

This set is a subgroup of G because both N and M are subgroups, and one of them is normal (in this case, both are).

To show that NM is normal in G, let $g \in G$ and take any element $nm \in NM$, where $n \in N$ and $m \in M$. Since N and M are normal, we know that

$$gng^{-1} \in N \quad \text{and} \quad gmg^{-1} \in M.$$

Then,

$$g(nm)g^{-1} = (gng^{-1})(gmg^{-1}) \in NM.$$

So, whenever we place an element of NM between an element $g \in G$ and its inverse, the result is still in NM. This means that NM is closed under this operation, and therefore NM is a normal subgroup of G.