WEB3CLUBS FOUNDATION LIMITED

Course Instructor: DR. Cyprian Omukhwaya Sakwa
PHONE: +254723584205 Email: cypriansakwa@gmail.com

Foundational Mathematics for Web3 Builders

Implemented in RUST

Lecture 47

August 27, 2024

Subgroups

A subgroup is a subset of a group that itself forms a group under the same operation defined on the original group.

Definition 19

A nonempty subset H of a group G is a subgroup of G if it forms a group under the same operation as that of G.

- The notation $H \leq G$ is used to indicate that H is a subgroup of G.
- If H is a proper subgroup of G, that is, if $H \neq G$ and $H \neq \{e\}$, then we write $H < G$.
- To check whether a nonempty subset H is a subgroup of a group G, it suffices to check that for all $h, k \in H$ we have $h \cdot k \in H$ and that for all $h \in H$ there exists $h^{-1} \in H$. Also check that the identity element of G is in G.

Subgroups are classified as; $\{0\}, \{1\}$

a) Trivial Subgroup: The subgroup that contains only the identity element. $\mathbb{Z}_6 \Rightarrow \{0\}, \mathbb{Z}_6, \ldots$

b) Proper Subgroup: A subgroup that is strictly smaller than the total group yet has more than just the identity element.

c) Improper Subgroup (or Whole Group): The subgroup that is the entire group itself.

That is, the subset $H = \{e\}$ consisting of the identity alone is always a subgroup for any group G. It is often called the trivial subgroup. Every group G is itself a subgroup of G. Any other subgroups of G other than G and $\{e\}$ are referred to as proper subgroups and subgroups other than $H = \{e\}$ are referred to as nontrivial subgroups.

24.1 Examples of Subgroups

a) The set $\mathbb{Z}$ is a group of all integers under addition. The set of even numbers denoted by $(2\mathbb{Z}, +)$ is also a subgroup under addition.

- In fact, for any elements $x, y \in 2\mathbb{Z}$, we have $(x + y) \in 2\mathbb{Z}$, (that is, the sum of two even numbers is even) and therefore closure is satisfied. And for any element $x \in 2\mathbb{Z}$, its inverse $-x \in 2\mathbb{Z}$ exists.

- The identity element $0 \in 2\mathbb{Z}$ exists. We do not need to show associativity in subgroups.

- If 2 is replaced by any other number say $3, 4, ..$, we would still have a subgroup of $\mathbb{Z}$ under addition. That is, every subgroup of $\mathbb{Z}$ has the form $n\mathbb{Z}$ for $n \in \mathbb{Z}$. For example, $9\mathbb{Z} = \{\cdots, -36, -27, -18, -9, 0, 9, 18, 27, 36, \cdots\}$ is a subgroup of $\mathbb{Z}$.

- Note that the set of all odd numbers is not a subgroup of $(\mathbb{Z}, +)$ since closure is not satisfied, that is, the sum of two odd numbers is not odd. Also, the additive identity element 0 is not odd.

$\frac{a}{b}$

b) The numbers $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, and $\mathbb{C}$ form groups under addition. The list $(2\mathbb{Z}, +) \subseteq (\mathbb{Z}, +) \subseteq (\mathbb{Q}, +) \subseteq (\mathbb{R}, +) \subseteq (\mathbb{C}, +)$ is a subgroup of every listed groups that contain it. That is,

$b \neq 0$

$$(2\mathbb{Z}, +) < (\mathbb{Z}, +) < (\mathbb{Q}, +) < (\mathbb{R}, +) < (\mathbb{C}, +).$$

c) For the group $(\mathbb{Z}_6, +)$, the subset $H = \{0, 2, 4\}$ forms a proper subgroup under addition. Clearly, the identity element is 0 and closure is satisfied for example $2 + 2 = 4 \in H$, $2 + 4 = 0 \in H$, $4 + 4 = 2 \in H$. Each element in H has an inverse in H for instance 2 and 4 are inverses of each other while 0 is its own inverse.

$\sqrt{-1} = i$

$-i = i^2$

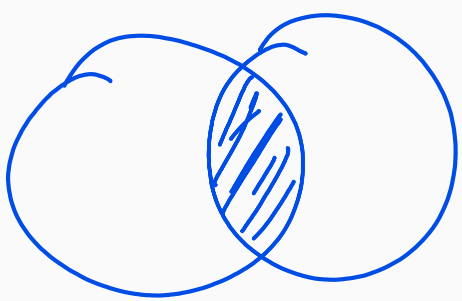
d) For the group $(\mathbb{C}^*, \times)$ the a set of complex numbers given by $H = \{i, -i, 1, -1\}$ forms a subgroup. See example 56. This is a proper subgroup.

e) For $n \times n$ matrices, each of the additive groups in the list

$$M_n(\mathbb{C}) \supseteq M_n(\mathbb{R}) \supseteq M_n(\mathbb{Q}) \supseteq M_n(\mathbb{Z})$$

is a subgroup of every listed groups that contain it.

f) The set $GL(n, \mathbb{R})$ of invertible $n \times n$ matrices is a group under multiplication. Let $SL(n, \mathbb{R})$ denote the set of invertible $n \times n$ matrices whose determinant is 1. Then $SL(n, \mathbb{R})$ is a proper subgroup of $GL(n, \mathbb{R})$.



24.2 Intersection of Subgroups

- The intersection of two or more subgroups of G is also a subgroup of G.
- If H_1 and H_2 are subgroups of G, then $H_1 \cap H_2$ is a subgroup of G.

24.3 Order of a Subgroup

The order of a subgroup H, denoted $|H|$ is the number of elements in H.

Theorem 20 (Lagrange's Theorem)

If H is a subgroup of a finite group G, then the order of H divides the order of G.

24.4 Subgroups of an additive group $\mathbb{Z}_n$

To comprehend the subgroups of the additive group $\mathbb{Z}_n$, one must first recognize that these subgroups are the cyclic subgroups generated by the divisors of n. Each divisor d of n has a corresponding cyclic subgroup.

Example 96

Find the proper subgroups of $(\mathbb{Z}_4, +)$

$\mathbb{Z}_4 = \{0, 1, 2, 3\}$

Solution

The divisors of 4 are $\{1, 2, 4\}$

Here are the subgroups generated by these divisors.

First, 1 generates, $1, 1+1 = 2, 1+1+1 = 3, 1+1+1+1 = 0$ and so $\langle 1 \rangle = \{0, 1, 2, 3\}$ Improper

2 generates, $2, 2+2 = 0$ and so $\langle 2 \rangle = \{0, 2\}$

4 generates, $4 = 0$ and so $\langle 4 \rangle = \{0\}$. $\Rightarrow$ Trivial

Solution (conti...)

Thus, $(\mathbb{Z}_4, +)$ has three subgroups; $\{0\}$, $\{0, 2\}$, $\{0, 1, 2, 3\}$.

The only proper subgroup here is $\{0, 2\}$.

Trivial Subgroup is $\{0\}$.

Improper Subgroup is $\{0, 1, 2, 3\}$ the entire group itself.

Example 97

Find the subgroups of $(\mathbb{Z}_{12}, +)$

Solution

The divisors of 12 are $\{1, 2, 3, 4, 6, 12\}$. The subgroups generated by each of these divisors are the following;

$\langle 1 \rangle = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$

$\langle 2 \rangle = \{0, 2, 4, 6, 8, 10\}$ pr

$\langle 3 \rangle = \{0, 3, 6, 9\}$ pr ✓

$\langle 4 \rangle = \{0, 4, 8\}$ ✓ proper

$\langle 6 \rangle = \{0, 6\}$ ✓ proper

Solution (conti...)

$\langle 12 \rangle = \{0\}$ ✓ Trivial

This Rust program generates and prints all subgroups of the additive group $\mathbb{Z}_n$. The subgroups of $\mathbb{Z}_n$ are cyclic subgroups generated by the divisors of n. For each divisor of n, there is a corresponding cyclic subgroup.

```rust
use std::collections::HashSet;

// Function to compute the divisors of n
fn divisors(n: u64) -> Vec<u64> {
        let mut divs = Vec::new();
        for i in 1..=n {
                if n % i == 0 {
                        divs.push(i);
                }
        }
        divs
}

// Function to generate a subgroup of Z_n given a generator
fn generate_subgroup(n: u64, generator: u64) -> HashSet<u64> {
        let mut subgroup = HashSet::new();
        let mut current = 0;
        while !subgroup.contains(&current) {
                subgroup.insert(current);
                current = (current + generator) % n;
        }
        subgroup
}

```

```rust
// Function to generate all subgroups of Z_n
fn generate_all_subgroups(n: u64) -> Vec<HashSet<u64>> {
        let mut subgroups = Vec::new();
        for d in divisors(n) {
                let generator = n / d;
                let subgroup = generate_subgroup(n, generator);
                subgroups.push(subgroup);
        }
        subgroups
}

fn main() {
        let n = 24; // Replace with the desired value of n
        let subgroups = generate_all_subgroups(n);
        println!("Subgroups␣of␣Z_{}:", n);
        for subgroup in subgroups {
                let mut subgroup_vec: Vec<u64> = subgroup.into_iter().collect();
                subgroup_vec.sort_unstable();
                println!("{:?}", subgroup_vec);
        }
}
```

Understanding the Rust code

1) use std::collections::HashSet; is necessary because the code relies on HashSet for efficiently managing collections of unique elements, which is essential for generating the subgroups of $\mathbb{Z}_n$ without duplicates and with efficient membership checks.

2) divisors $(n : u64) -> $ Vec $< u64 >$: Computes and returns a vector containing all divisors of the integer n.

```
fn divisors(n: u64) -> Vec<u64> {
        let mut divs = Vec::new();
        for i in 1..=n {
                if n % i == 0 {
                        divs.push(i);
                }
        }
        divs
}
```

- The function initializes an empty vector divs.
- It then iterates over all integers i from 1 to n.

Understanding the Rust code (conti...)

- If n is divisible by i (i.e.,$n\%i == 0$), i is added to the divs vector.
- Finally, the vector divs containing all divisors of n is returned.

3) generate_subgroup($n : u64$, generator: $u64) -> $ HashSet$< u64 >$: Generates and returns the subgroup of $\mathbb{Z}_n$ generated by a given element (generator).

```
fn generate_subgroup(n: u64, generator: u64) -> HashSet<u64> {
        let mut subgroup = HashSet::new();
        let mut current = 0;
        while !subgroup.contains(&current) {
                subgroup.insert(current);
                current = (current + generator) % n;
        }
        subgroup
}
```

- The function initializes an empty HashSet called subgroup to store the elements of the subgroup.

Understanding the Rust code (conti...)

- It then enters a loop where it repeatedly adds the current element to the subgroup and updates the current element to be (current + generator) % n.
- The loop continues until the current element (which is the residue class) has already been seen in the subgroup, meaning that the subgroup has cycled back to its starting point.
- The function returns the subgroup as a HashSet.

4) generate_all_subgroups$(n : u64)-$ $>$ Vec¡HashSet$< u64 >>$: Generates and returns all distinct subgroups of $\mathbb{Z}_n$.

```
fn generate_all_subgroups(n: u64) -> Vec<HashSet<u64>> {
        let mut subgroups = Vec::new();
        for d in divisors(n) {
                let generator = n / d;
                let subgroup = generate_subgroup(n, generator);
                subgroups.push(subgroup);
        }
        subgroups
}
```

Understanding the Rust code (conti...)

- The function initializes an empty vector subgroups to store all subgroups of $\mathbb{Z}_n$.
- It calls the divisors function to get all divisors d of n.
- For each divisor d, it computes the generator as generator $= \frac{n}{d}$.
- The generate_subgroup function is then called with this generator to generate the corresponding subgroup, which is added to the subgroups vector.
- After processing all divisors, the vector subgroups containing all distinct subgroups of $\mathbb{Z}_n$ is returned.

Understanding the Rust code (conti...)

5) main(): The entry point of the program where n is defined and all subgroups of $\mathbb{Z}_n$ are computed and printed.

```
fn main() {
        let n = 24; // Replace with the desired value of n
        let subgroups = generate_all_subgroups(n);
        println!("Subgroups␣of␣Z_{}:", n);
        for subgroup in subgroups {
                let mut subgroup_vec: Vec<u64> = subgroup.into_iter().collect();
                subgroup_vec.sort_unstable();
                println!("{:?}", subgroup_vec);
        }
}
```

- The variable n
- The generate_all_subgroups function is called to compute all subgroups of $\mathbb{Z}_n$.
- The code then prints each subgroup. The elements of each subgroup are first collected into a vector, sorted, and then printed.