📘 Class 9 Maths – Ganita Manjari

💻 Chapter 11: The World of Algorithms

These notes are based on **Ganita Manjari Part II, Chapter 11: The World of Algorithms**. The chapter focuses on algorithms, addition, divisors, greatest common divisor (GCD), data structures, Euclid's algorithm, and Aryabhata's division algorithm.

---

1. 💡 What is an Algorithm?

An **algorithm** is a **step-by-step procedure** used to arrive at an answer or solve a problem.

An algorithm should describe the steps as **precisely as possible**.

Example

The familiar method of adding numbers digit by digit is an algorithm.

For example:

473 + 695

We:

1. Write the numbers one below the other.
2. Align the digits from the right.
3. Add the digits from right to left.
4. Carry when necessary.
5. Continue until all digits have been added.

The textbook uses this familiar addition process to introduce the idea of an algorithm.

---

2. ➕ Adding Numbers Digit by Digit

The textbook describes an algorithm for addition using the **Indian base-ten place-value system**.

Steps

Step 1: Write the two numbers one below the other, aligning their digits from right to left.

Step 2: Add the rightmost digits.

- If the sum is less than 10, write the sum directly.
- If the sum is 10 or more, write the units digit and carry 1 to the next place.

Step 3: Move one place to the left.

Step 4: Add the digits along with the carry.

Step 5: Continue until all the digits have been processed.

Example

Add:

473 + 695

Units:

3 + 5 = 8

Tens:

7 + 9 = 16

Write 6 and carry 1.

Hundreds:

4 + 6 + 1 = 11

Therefore:

473 + 695 = 1168

The algorithm works because numbers are grouped into **units, tens, hundreds, thousands, ...**, and a group of ten units is carried into the next place.

---

3. 🔢 Why Do We Need Algorithms?

A simple method may work for small numbers but become inefficient for large numbers.

For example, counting dots to add two numbers would require much more work as the numbers become larger.

A carefully designed algorithm can solve the same problem much more efficiently.

The chapter therefore asks us to think about:

- How many steps an algorithm takes.
- Why an algorithm works.
- Whether the algorithm can be made more efficient.

---

4. 🔍 Greatest Common Divisor (GCD)

The **greatest common divisor (GCD)** of two numbers is the largest number that divides both numbers.

It is also called the **highest common factor (HCF)**.

For example:

Divisors of 12:

1, 2, 3, 4, 6, 12

Divisors of 18:

1, 2, 3, 6, 9, 18

Common divisors:

1, 2, 3, 6

Therefore:

GCD(12, 18) = 6

The chapter explores how to find GCD directly through algorithms rather than simply relying on previously learned methods.

---

5. 📋 Computing the Divisors of a Number

To find the divisors of a number **n**, the textbook gives an algorithm.

Algorithm to Find the Divisors of n

Step 1: Start with an empty list called **list-of-divisors**.

Step 2: Consider each number:

1, 2, 3, ..., n

Step 3: For each number **j**:

- If **j divides n**, add j to the list.
- If j does not divide n, do nothing.

The resulting list contains the divisors of n in increasing order.

---

6. ✏️ Example: Divisors of 18

Check the numbers from 1 to 18.

The numbers that divide 18 exactly are:

1, 2, 3, 6, 9, 18

Therefore:

Divisors(18) = [1, 2, 3, 6, 9, 18]

Because the numbers are checked in increasing order, the list is also arranged in increasing order.

---

7. 🔢 Divisors of 375 and 825

The textbook uses the numbers **375 and 825** while developing the GCD algorithm.

Divisors of 375

[1, 3, 5, 15, 25, 75, 125, 375]

Divisors of 825

[1, 3, 5, 11, 15, 25, 33, 55, 75, 165, 275, 825]

Common divisors:

[1, 3, 5, 15, 25, 75]

The largest common divisor is:

75

Therefore:

GCD(375, 825) = 75

---

8. 🔗 Finding GCD Using Lists

Once we have the divisors of two numbers:

1. Find the list of divisors of the first number.
2. Find the list of divisors of the second number.
3. Compare the two lists.
4. Create a list of common divisors.
5. The largest common divisor is the GCD.

For example:

GCD(375, 825) = 75

This is the chapter's **first algorithm for GCD**.

---

9. 📦 Data Structures

While executing an algorithm, we often need to keep track of intermediate quantities.

For example:

- `divisors-of-m`
- `divisors-of-n`
- `common-divisors`

The textbook uses **lists** to store these collections of numbers.

A **data structure** is a way of organising information so that it can be used efficiently by an algorithm.

A list arranged in increasing order is an example of a **data structure**.

---

10. 📋 Working with Lists

Suppose **List 1** and **List 2** contain numbers arranged in increasing order.

An algorithm can be designed to find:

- Elements in List 1 that are not in List 2.
- Elements in List 2 that are not in List 1.

The important idea is that algorithms can operate not only on individual numbers but also on **collections of information**.

---

11. 🔄 Euclid's Algorithm for GCD

The first GCD algorithm using complete divisor lists can involve a lot of work for large numbers.

The chapter therefore develops a more efficient method based on **repeated reduction**.

The key idea is:

If **m > n**, then:

GCD(m, n) = GCD(n, m − n)

This allows a GCD problem to be reduced to a smaller one.

---

12. 🧮 Euclid's Algorithm

The textbook gives the following form:

Algorithm for GCD(m, n)

Step 1: If m < n, reverse the numbers and compute GCD(n, m).

Step 2: If n = 0, report the answer as m.

Step 3: Otherwise, reduce the problem to:

GCD(n, m − n)

Repeat the process until the second number becomes 0.

Then the first number is the GCD.

---

13. ✏️ Example: GCD(375, 825)

Start with:

GCD(375, 825)

Since 375 < 825:

GCD(825, 375)

Now:

825 − 375 = 450

So:

GCD(375, 450)

Then:

GCD(450, 375)

450 − 375 = 75

So:

GCD(375, 75)

375 − 75 = 300

So:

GCD(75, 300)

Then:

GCD(300, 75)

300 − 75 = 225

So:

GCD(75, 225)

Then:

GCD(225, 75)

225 − 75 = 150

Then:

GCD(75, 150)

Then:

GCD(150, 75)

Then:

GCD(75, 75)

Then:

GCD(75, 0)

Since the second number is 0:

GCD = 75

---

14. ⚠️ Limitation of Repeated Subtraction

Although Euclid's subtraction algorithm works, it can require many steps.

For example:

GCD(99, 2)

would require repeatedly subtracting 2.

For a number of the form:

2k + 1

the number of reductions can be roughly proportional to **k**.

Thus, the number of steps can depend on the numerical value rather than simply on the number of digits.

---

15. ➗ Aryabhata's Division Algorithm

The chapter introduces a more efficient version using **division with remainder**.

The idea is to replace repeated subtraction by a single division step.

Instead of:

GCD(m, n) → GCD(n, m − n)

we use:

GCD(m, n) → GCD(n, m mod n)

where **m mod n** means the **remainder when m is divided by n**.

Examples

17 mod 5 = 2

because:

17 = 5 × 3 + 2

33 mod 7 = 5

because:

33 = 7 × 4 + 5

---

16. 🧱 Improved GCD Algorithm

Algorithm

Step 1: If m < n, reverse the numbers.

Step 2: If n = 0, report m as the answer.

Step 3: Otherwise, reduce the problem to:

GCD(n, m mod n)

Continue until the second number becomes 0.

---

17. ✏️ Example: GCD(99, 2) Using Remainders

Start with:

GCD(99, 2)

99 mod 2 = 1

Therefore:

GCD(99, 2) = GCD(2, 1)

Now:

2 mod 1 = 0

Therefore:

GCD(2, 1) = GCD(1, 0)

Since the second number is 0:

GCD = 1

Only **two reduction steps** are needed.

---

18. 🇮🇳 Aryabhata and the Division Method

The chapter connects the division-with-remainder approach with the **long division method used in Indian mathematics**.

It discusses **Aryabhata's Aryabhatiya (499 CE)** and the use of Indian place-value numerals and repeated division by remainder for finding GCD.

Examples given include:

- GCD(99, 2)
- GCD(825, 375)
- GCD(60, 16)

The division-based approach is much more efficient than repeatedly subtracting the smaller number.

---

19. 📜 A Pinch of History – The Word "Algorithm"

The chapter also discusses the history of the word **algorithm**.

The Persian mathematician **Al-Khwarizmi (c. 780–850 CE)** studied Indian mathematics and wrote about Indian place-value numerals and arithmetic procedures.

His name became associated with these computational methods. The term developed through **algorismus** and later became **algorithm**.

The word eventually came to mean a **well-defined, step-by-step systematic procedure for solving a class of problems**.

---

20. 🔑 Important Terms

Algorithm

A step-by-step procedure for solving a problem or obtaining an answer.

Execute an Algorithm

To carry out the steps of an algorithm one by one.

Divisor

A number that divides another number exactly.

GCD

The greatest common divisor of two numbers.

List

An ordered collection of values used to store information.

Data Structure

A way of organising information so that an algorithm can use it effectively.

Remainder

The amount left over after division.

m mod n

The remainder when **m** is divided by **n**.

---

21. 📌 Important Algorithms to Remember

Algorithm 1: Addition

Align → Add from right to left → Carry → Continue

Algorithm 2: Divisors of n

Start empty list → Check 1 to n → Add every divisor

Algorithm 3: GCD Using Divisor Lists

Find divisors of both numbers → Find common divisors → Take the largest

Algorithm 4: Euclid's Subtraction Algorithm

GCD(m,n) → GCD(n,m−n)

until the second number becomes 0.

Algorithm 5: Division-with-Remainder Algorithm

GCD(m,n) → GCD(n,m mod n)

until the second number becomes 0.

---

22. ⚡ Quick Revision

- An **algorithm** is a precise, step-by-step procedure.
- Addition by place value is an example of an algorithm.
- Algorithms should be precise enough to be executed step by step.
- A **divisor** divides a number exactly.
- The **GCD** is the largest common divisor.
- Divisors can be stored in an ordered **list**.
- A list is an example of a **data structure**.
- GCD can first be found by comparing lists of divisors.
- **Euclid's algorithm** uses repeated subtraction.
- The improved version uses **division with remainder**.
- **m mod n** means the remainder when m is divided by n.
- The division-with-remainder method is much more efficient for GCD.
- The chapter connects the history of algorithms with Indian mathematics and Aryabhata.

---

23. 📚 Chapter Summary

The World of Algorithms introduces algorithms as precise, step-by-step procedures for solving problems. The chapter begins with the familiar algorithm for addition and then develops algorithms for finding divisors and the greatest common divisor.

The chapter introduces **lists as data structures** for organising intermediate information. It then develops **Euclid's algorithm** using repeated subtraction and improves it using **division with remainder**.

The chapter also connects the development of algorithms with the history of Indian mathematics, including **Aryabhata's division method** and the historical origin of the word **algorithm**.