Mathematics for CS/AI — Week 1

Introducing Logic and Proofs

Notes by Sarfraz Raza A mathematical foundation for thinking like a Computer Scientist and an AI Engineer.

"The goal is not merely to calculate. The goal is to learn how to reason."

Logic • Implication • Integers • Proofs • Primes • Algorithms

1 Foundations: Why Mathematics and Propositions

Computer Science is often introduced through programming:

$$\text{problem} \longrightarrow \text{code} \longrightarrow \text{output}.$$

But beneath every serious program lies a deeper question:

Why should the program be correct?

To answer that question, we need the language of mathematics.
Mathematics gives us a way to:

- express statements precisely,
- combine conditions logically,
- reason about consequences,
- prove that an argument is correct,
- discover patterns that apply to infinitely many cases, and
- turn mathematical observations into efficient algorithms.

Key Idea

The central theme of this course is:

$$\boxed{\begin{array}{c}\text{Mathematical Thinking} \longrightarrow \text{Logical Reasoning} \longrightarrow \text{Proof} \\ \longrightarrow \text{Algorithms} \longrightarrow \text{Computer Science \& AI}\end{array}}$$

A programmer asks, "Does this work?"
A Computer Scientist learns to ask, "**Why does this always work?**"

1.1 Propositions: The Building Blocks of Logic

A **proposition** is a declarative statement that has a definite truth value: it is either **True** or **False**.
For example,

$$P_1: \quad 1+1=3$$

is a false proposition, while

$$P_2: \quad 1+1=2$$

is a true proposition.

Definition

A proposition is a statement for which exactly one of the two truth values, True or False, can be assigned.

What Is Not a Proposition?

A sentence is not a proposition when it does not have a definite truth value without further meaning or context.

- "C++ is a great language." is subjective. The word *great* has no fixed mathematical criterion, so different people may judge it differently.
- "Today's weather is nice." is vague and context-dependent. We have not fixed the place, the meaning of *nice*, or even the date intended by *today*.
- "Is the server running?" is a question, not a declarative statement.
- $x+1=4$ is an open sentence: its truth depends on the unspecified value of x.

These can become propositions after the missing criteria are supplied. For example, "At noon in a specified city, the temperature is above 20°C" has a definite truth value.

Key Idea

Being false does not stop a statement from being a proposition. Thus $1+1=3$ is a proposition because it is definitely false. By contrast, a subjective or underspecified sentence may fail to be a proposition because no single truth value has yet been determined.

We commonly represent truth values using:

$$\text{True} = 1, \qquad \text{False} = 0.$$

Why Propositions Matter

Propositions are the basic "sentences" from which larger logical statements are constructed.
For example, suppose:

$$P: \quad \text{The server is running}$$

and

$$Q: \quad \text{The database is available.}$$

We can combine them to express statements such as:

$$P \wedge Q$$

or

$$P \vee Q.$$

This is exactly the kind of reasoning that appears in programming conditions, digital circuits, algorithms, databases, and AI systems.

2 Logical Connectors and Evaluation

The three fundamental logical connectors introduced in this lecture are:

$$\boxed{\vee \quad \wedge \quad \neg}$$

They correspond to:

Symbol	Name	Meaning
$\vee$	OR	At least one is true
$\wedge$	AND	Both/all conditions are true
$\neg$	NOT	Reverses the truth value

2.1 OR: $P \vee Q$

The statement

$$P \vee Q$$

is true whenever at least one of P and Q is true.

P	Q	$P \vee Q$
0	0	0
0	1	1
1	0	1
1	1	1

Connection to Computer Science

In programming, this corresponds conceptually to:

```
if (e1 OR e2 OR e3)
```

The overall condition is true when at least one condition is true.

Short-Circuiting

Programming languages often use **short-circuit evaluation**.
If we have:

$$e_1 \vee e_2 \vee e_3$$

and e_1 is already true, there is no logical need to evaluate e_2 or e_3.
This is both a logical idea and an important programming concept.

2.2 AND: $P \wedge Q$

The statement

$$P \wedge Q$$

is true only when both propositions are true.

P	Q	$P \wedge Q$
0	0	0
0	1	0
1	0	0
1	1	1

For several conditions,

$$P_1 \wedge P_2 \wedge P_3 \wedge \cdots$$

the entire statement is true only when **all** conditions are true.

2.3 NOT: $\neg P$

The negation of P is written as:

$$\neg P.$$

It reverses the truth value:

P	$\neg P$
0	1
1	0

2.4 When the Order of Conditions Matters

Logical expressions may look like pure mathematics, but in a computer program the **order of evaluation can matter**.
Consider the idea of checking whether a denominator is non-zero before division.
An unsafe ordering is conceptually:

```
if (N / d == 0 AND d != 0)
```

because the division may be attempted before the program checks that $d \neq 0$.
A safer ordering is:

```
if (d != 0 AND N / d == 0)
```

Now the first condition protects the second operation.

Important

Remember: Order matters.
Mathematical equivalence and program execution are related, but a computer must actually evaluate expressions. A logically unnecessary operation can still be dangerous if it is evaluated before the condition that protects it.

This idea will become increasingly important when we study algorithms, program correctness, and complexity.

3 Implication, Contrapositive, and Negation

One of the most important ideas in mathematical reasoning is the **implication**:

$$\boxed{P \to Q}$$

It is read as:

$$\textbf{If } P\textbf{, then } Q\textbf{.}$$

For example:

$$P: \quad \text{It is raining}$$

$$Q: \quad \text{It is cloudy}$$

Then:

$$P \to Q$$

means:

"If it is raining, then it is cloudy."

3.1 Truth Table for Implication

P	Q	$P \to Q$
0	0	1
0	1	1
1	0	0
1	1	1

The most important row is:

$$P = 1, \qquad Q = 0.$$

This is the **only case in which an implication is false**.

Key Idea

An implication

$$P \to Q$$

is violated only when P happens but Q does not happen.

3.2 The Four Cases

Suppose:

$$P = \text{"It is raining"}$$

and

$$Q = \text{"It is cloudy."}$$

1. If it is raining, then the implication requires it to be cloudy.
2. If it is not raining, we cannot conclude that it is cloudy.
3. If it is cloudy, we cannot conclude that it is raining.
4. If it is not cloudy, then the implication tells us that it cannot be raining.

The last case gives us the contrapositive.

3.3 Contrapositive

For an implication

$$P \to Q,$$

the **contrapositive** is:

$$\boxed{\neg Q \to \neg P}$$

and these two statements are logically equivalent:

$$\boxed{P \to Q \iff \neg Q \to \neg P}$$

For example:

If it is raining, then it is cloudy.

is equivalent to:

If it is not cloudy, then it is not raining.

Key Idea

When a direct proof is difficult, the contrapositive often gives us a much cleaner route:

$$\boxed{P \to Q \quad \text{can be proved through} \quad \neg Q \to \neg P.}$$

Contraposition with Several Known Premises

Usually a conclusion does not depend on only one premise. Suppose the known facts are

$$p_1, p_2, \ldots, p_n$$

and, together with an original premise P, they imply the desired conclusion Q:

$$\boxed{(p_1 \wedge p_2 \wedge \cdots \wedge p_n \wedge P) \to Q.}$$

Negating the conclusion gives the contrapositive

$$\neg Q \to \neg(p_1 \wedge p_2 \wedge \cdots \wedge p_n \wedge P).$$

By De Morgan's law this is

$$\boxed{\neg Q \to (\neg p_1 \vee \neg p_2 \vee \cdots \vee \neg p_n \vee \neg P).}$$

Thus, if Q is false, at least one fact used to derive it must be false: possibly one of $p_1, \ldots, p_n$, or the original premise P. It does *not* follow that every premise is false; the conclusion is that the premises cannot all remain true together with $\neg Q$.

Connection to Computer Science

This explains the close relationship between contraposition and contradiction. In a contradiction proof we assume all known premises and also assume $\neg Q$. If reasoning then forces $\neg p_i$ while p_i is already known, or forces $\neg P$ while P is assumed, we obtain a contradiction. In this useful sense, proof by contradiction is the broader framework: ordinary contraposition is its especially focused form for a single implication.

3.4 De Morgan's Laws

Negating a logical expression requires care.
The two fundamental De Morgan laws are:

$$\boxed{\neg(P \wedge Q) = \neg P \vee \neg Q}$$

and

$$\boxed{\neg(P \vee Q) = \neg P \wedge \neg Q}$$

In words:

- NOT of AND becomes OR of NOTs.
- NOT of OR becomes AND of NOTs.

These laws appear everywhere in Computer Science: conditions, search, digital logic, database queries, software testing, and program verification.

4 Integers and Direct Proof

The set of integers is denoted by:

$$\boxed{\mathbb{Z}}$$

and consists of:

$$\mathbb{Z} = \{\ldots, -4, -3, -2, -1, 0, 1, 2, 3, 4, \ldots\}.$$

Integers include:

- negative integers,
- zero, and
- positive integers.

4.1 Even Integers

An integer e is even if it can be written as:

$$\boxed{e = 2k}$$

for some:

$$k \in \mathbb{Z}.$$

4.2 Odd Integers

An integer o is odd if it can be written as:

$$\boxed{o = 2k + 1}$$

for some:

$$k \in \mathbb{Z}.$$

These two representations are extremely powerful because they allow us to prove statements about *all* even and odd integers at once.

4.3 Example: Even + Even = Even

Proposition 4.1. *The sum of two even integers is even.*

Proof

Let

$$N_1 = 2k_1, \qquad N_2 = 2k_2,$$

where

$$k_1, k_2 \in \mathbb{Z}.$$

Then

$$\begin{aligned} N_1 + N_2 &= 2k_1 + 2k_2 \\ &= 2(k_1 + k_2). \end{aligned}$$

Let

$$k_3 = k_1 + k_2.$$

Since integers are closed under addition,

$$k_3 \in \mathbb{Z}.$$

Therefore,

$$N_1 + N_2 = 2k_3,$$

which is precisely the definition of an even integer.
Hence,

$$\boxed{\text{Even} + \text{Even} = \text{Even}.}$$

Takeaway

Notice what the proof did *not* do.
It did not test examples such as

$$2+4=6, \qquad 8+10=18.$$

Instead, it represented an arbitrary even number as $2k$ and proved the result for every possible choice of integers k_1 and k_2.

4.4 Example: Odd + Odd = Even

Proposition 4.2. *The sum of two odd integers is even.*

Proof

Let

$$N_1 = 2k_1 + 1$$

and

$$N_2 = 2k_2 + 1,$$

where

$$k_1, k_2 \in \mathbb{Z}.$$

Then

$$\begin{aligned} N_1 + N_2 &= (2k_1 + 1) + (2k_2 + 1) \\ &= 2k_1 + 2k_2 + 2 \\ &= 2(k_1 + k_2 + 1). \end{aligned}$$

Let

$$k_3 = k_1 + k_2 + 1.$$

Since $k_3 \in \mathbb{Z}$,

$$N_1 + N_2 = 2k_3.$$

Therefore,

$$\boxed{N_1 + N_2 \text{ is even.}}$$

Connection to Computer Science

This is an early example of the difference between **testing** and **proving**.
A computer can test thousands or millions of examples.
A mathematical proof establishes a statement for an entire infinite family.

5 Rational, Irrational, and Contradiction

A rational number is a number that can be written as:

$$\boxed{\frac{p}{q}}$$

where:

$$p, q \in \mathbb{Z}, \qquad q \neq 0.$$

The set of rational numbers is denoted by:

$$\mathbb{Q}.$$

Examples include:

$$\frac{1}{2}, \qquad -\frac{7}{3}, \qquad 5 = \frac{5}{1}.$$

An irrational number cannot be represented as a ratio of two integers.
Examples include:

$$\sqrt{2}, \qquad \sqrt{3}, \qquad \pi.$$

5.1 Example: Rational + Irrational = Irrational

Proposition 5.1. *The sum of a rational number and an irrational number is irrational.*

Proof

Let

$$R = \frac{p}{q}$$

be rational, and let I be irrational.
Suppose, for contradiction, that

$$R + I$$

is rational.
Then

$$I = (R + I) - R.$$

The difference of two rational numbers is rational. Therefore I would have to be rational.
But this contradicts the assumption that I is irrational.
Therefore,

$$\boxed{\text{Rational} + \text{Irrational} = \text{Irrational}.}$$

Key Idea

This is an example of **proof by contradiction**:

Assume the opposite $\longrightarrow$ derive a contradiction
$\longrightarrow$ reject the assumption.

6 Prime Numbers and Efficient Testing

6.1 Euclid: There Are Infinitely Many Primes

Prime numbers are positive integers greater than 1 whose only positive divisors are 1 and themselves.
Composite numbers are integers greater than 1 that are not prime. Every composite number can be written as a product of primes (and, apart from the order of the factors, that prime factorisation is unique).
The first few primes are:

$$2, 3, 5, 7, 11, 13, 17, \ldots$$

A natural question is:

Does the list of primes eventually end?

Euclid showed that the answer is no.

Theorem 6.1. *There are infinitely many prime numbers.*

Key Idea

This theorem is stated as a claim with a conclusion and no explicit hypothesis:

There are infinitely many primes.

That does not mean the proof uses no prior knowledge. Its background premises are established mathematical facts, including:

- a prime has exactly two positive *divisors*, 1 and itself;
- every integer greater than 1 is prime or has a prime divisor; and
- equivalently, every composite integer is a product of primes.

These background facts are the $p_1, p_2, \ldots$ available to the proof. (The word is *divisors*, not multiples: a prime has infinitely many multiples.)

Proof

Suppose, for contradiction, that there are only finitely many primes.
List them as:

$$p_1, p_2, \ldots, p_a.$$

Now construct:

$$T = \prod_{i=1}^{a} p_i$$

and define:

$$U = T + 1.$$

Thus,

$$U = p_1 p_2 \cdots p_a + 1.$$

Consider any prime p_i in our supposed complete list.
Because p_i divides T, dividing $U = T + 1$ by p_i leaves remainder 1.
Therefore U is not divisible by any of

$$p_1, p_2, \ldots, p_a.$$

But every integer greater than 1 is either prime or has a prime factor.
Therefore U is either:

- itself a new prime, or
- divisible by a prime that is not in our list.

Either possibility contradicts the assumption that $p_1, \ldots, p_a$ were all the primes.
Hence:

There are infinitely many primes.

Side Note: Primes as Elementary Numbers

Prime numbers play the role of elementary building blocks. The analogy is with chemistry: atoms of different elements combine to form compounds, and the compound is made from those atoms rather than from some additional kind of matter. Similarly, primes combine by multiplication to form every positive integer greater than 1; composite numbers contain no new multiplicative building blocks beyond primes. This is why primes are often described as the "atoms" or elementary numbers of arithmetic. The analogy is structural, not literal: integers have unique prime factorisation, whereas chemical structure involves further physical rules.

Takeaway

The proof does not need us to find the next prime.
It proves something much stronger:

No finite list can contain every prime.

This is the power of mathematical proof.

6.2 Prime Testing and the Square-Root Insight

Suppose we want to determine whether a positive integer N is prime.
A naive approach might try every possible divisor:

$$2, 3, 4, \ldots, N - 1.$$

But mathematics gives us a much better idea.
Suppose N is composite. Then:

$$N = a \cdot b$$

for integers $a, b > 1$.
At least one of a and b must satisfy:

$$\boxed{a \leq \sqrt{N} \quad \text{or} \quad b \leq \sqrt{N}.}$$

Why?
Assume both were greater than $\sqrt{N}$:

$$a > \sqrt{N}, \qquad b > \sqrt{N}.$$

Then:

$$ab > \sqrt{N}\sqrt{N} = N.$$

But $ab = N$, which is a contradiction.
Therefore, whenever N is composite, it has a factor no larger than $\sqrt{N}$.

Key Idea

To test whether N is prime, it is sufficient to test possible divisors up to:

$$\boxed{\sqrt{N}}.$$

Prime Testing Algorithm
Conceptually:

```
isPrime(N):
    for d = 2 to floor(sqrt(N)):
        if N % d == 0:
            return false
    return true
```

The important lesson is not merely the loop.
The important lesson is the mathematical reasoning that allowed us to **remove unnecessary work**.
A Scale Example
Consider:

$$N = 10^{20}.$$

Checking all the way up to 10^{20} would be enormous.
But:

$$\sqrt{10^{20}} = 10^{10}.$$

The search range is still large, but it is dramatically smaller.
For comparison:

$$10^{20} = 100,000,000,000,000,000,000$$

whereas:

$$10^{10} = 10,000,000,000.$$

Connection to Computer Science

This is an early lesson in algorithmic efficiency:

Mathematical insight can reduce computation by enormous amounts.

Later, when we study time complexity, we will learn how to describe such improvements formally using Big-O notation.

6.3 Example: Factor Pairs and the Square-Root Boundary

Consider the factor pairs of:

$$100.$$

They include:

$$\begin{aligned} 100 &= 2 \times 50 \\ 100 &= 3 \times 33 \\ 100 &= 4 \times 25 \\ 100 &= 5 \times 20 \\ 100 &= 6 \times 16 \\ 100 &= 7 \times 14 \\ 100 &= 8 \times 12 \\ 100 &= 9 \times 11 \\ 100 &= 10 \times 10. \end{aligned}$$

But:

$$\sqrt{100} = 10.$$

Once we reach 10, the factor pairs simply reverse:

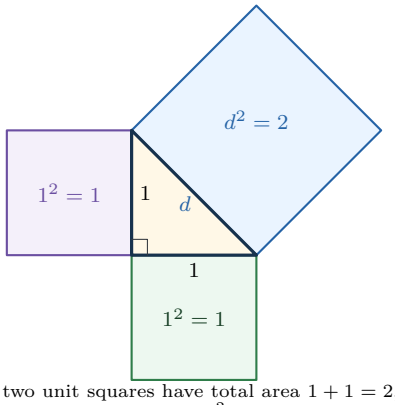
$$11 \times 9, \quad 12 \times 8, \quad \ldots, \quad 50 \times 2.$$

So checking beyond $\sqrt{100}$ repeats information we have already obtained.

Takeaway

The square-root rule is not a magic programming trick.
It is a direct consequence of the structure of factor pairs.

7 The Pythagorean Theorem and the Appearance of $\sqrt{2}$



Pythagorean area diagram. The two unit squares have total area $1 + 1 = 2$, so the square on the hypotenuse d has area $d^2 = 2$.

The diagram begins with a geometric object: construct a right triangle whose perpendicular sides each have length one, and call its hypotenuse d. At this stage, d is already a perfectly definite length. We have not yet called it $\sqrt{2}$, nor assumed that it can be written as a fraction.

7.1 Historical Note: The Greek Geometric Argument

Euclid's *Elements*, Book I, Proposition 47, states the theorem in geometric language:

Key Idea

In right-angled triangles, the square on the side subtending the right angle is equal to the squares on the sides containing the right angle.

Apply this statement to the right triangle in the diagram. The squares on its two perpendicular legs are each one unit square. Therefore the square erected on the hypotenuse d has the same area as those two unit squares together. The labels in the picture show the geometric conclusion directly:

$$\boxed{d^2 = 1 + 1 = 2.}$$

This way of speaking starts with the *magnitude* d and then studies the square constructed on it. Thus "the square of the diagonal" makes geometric sense before a numerical name for the diagonal is known.
The Greeks would distinguish between a magnitude and a number measuring that magnitude. The diagonal is determined by the construction, but it is *incommensurable* with the side: there is no smaller common length that measures both an exact whole number of times. In modern language, their ratio cannot be expressed as a ratio of integers.

Greek-Style Reconstruction of the Incommensurability Argument

Suppose the side and diagonal were commensurable. Choose a common measuring unit so that the side and diagonal are represented by whole numbers having no common factor.
Because the square on the diagonal equals two squares on the side, the square number belonging to the diagonal is even. Hence the diagonal's whole number must be even. Removing its factor of two from the same relation then shows that the side's whole number must also be even.
Both numbers therefore have the common factor two, contradicting the choice that they had no common factor. Hence the diagonal and side are incommensurable.

7.2 The Same Reasoning in Modern Algebra

Algebra compresses the geometric statement into the equality

$$\boxed{a^2 + b^2 = c^2}.$$

For the right triangle in the diagram, let its already constructed hypotenuse have length d. Since the legs have length one,

$$d^2 = 1^2 + 1^2 = 2.$$

The positive magnitude whose square is 2 is then named

$$\boxed{d = \sqrt{2}}.$$

The order of thought matters:

$$\boxed{\text{construct } d \;\longrightarrow\; \text{determine } d^2 = 2 \;\longrightarrow\; \text{name } d = \sqrt{2}.}$$

Important

It would be misleading to say that the diagonal itself "cannot be determined." Geometry determines it exactly. What cannot be done is to express its length as a ratio of integers. The symbol $\sqrt{2}$ is an exact name for that irrational magnitude, not a decimal approximation.
The picture establishes $d^2 = 2$ through the Pythagorean theorem. It does not, by itself, prove that d is irrational; for that we must prove that no ratio of integers can have square equal to 2.

For the modern algebraic proof of irrationality, suppose in lowest terms that

$$d = \frac{m}{n}, \qquad m, n \in \mathbb{Z}, \quad n \neq 0.$$

Then

$$m^2 = 2n^2.$$

Thus m^2 is even, so m is even; write $m = 2k$. Substitution gives $n^2 = 2k^2$, so n is also even. This contradicts the assumption that m/n was in lowest terms. Therefore $d = \sqrt{2}$ is irrational. The parity fact used here is proved in the next section.

Takeaway

The Greek geometric language and modern algebra express the same structure:

$$\text{square on the diagonal} = \text{two unit squares} \qquad \Longleftrightarrow \qquad d^2 = 2.$$

Algebra did not create the geometric fact; it gave us a compact symbolic language in which to state and manipulate it.

8 Proof Methods and Reusable Patterns

8.1 Even Squares and Odd Squares

An important property of integers is:

$$\boxed{a^2 \text{ is even} \to a \text{ is even.}}$$

Its contrapositive is:

$$\boxed{a \text{ is odd} \to a^2 \text{ is odd.}}$$

These are equivalent statements.

Proof That Odd Squared Is Odd

Suppose a is odd.
Then, by definition, there exists an integer k such that:

$$a = 2k + 1.$$

Square both sides:

$$\begin{aligned} a^2 &= (2k+1)^2 \\ &= 4k^2 + 4k + 1 \\ &= 2(2k^2 + 2k) + 1. \end{aligned}$$

The expression has the form:

$$2m + 1,$$

which is the definition of an odd integer.
Therefore:

$$\boxed{a^2 \text{ is odd.}}$$

Hence, by contrapositive:

$$\boxed{a^2 \text{ even} \to a \text{ even.}}$$

8.2 A Proof Pattern Worth Remembering

The previous examples reveal a reusable mathematical workflow.

1. **Define the mathematical object.**
   Example:
   $$a = 2k + 1$$
   means a is odd.
2. **Translate the statement into algebra.**
3. **Manipulate the algebra carefully.**
4. **Return to the definition.**
   If we obtain:
   $$a = 2m,$$
   then we know a is even.
5. **Conclude the desired statement.**

Key Idea

A strong proof often follows the pattern:

$$\boxed{\text{Definition} \to \text{Algebra} \to \text{Structure} \to \text{Conclusion}}$$

8.3 Proof by Contradiction: The Core Pattern

Proof by contradiction begins by assuming that the statement we want to prove is false.
Then we reason until the assumption produces an impossibility.
The structure is:

$$\boxed{\begin{array}{c} \text{Want to prove } P \\ \Downarrow \\ \text{Assume } \neg P \\ \Downarrow \\ \text{Reason logically} \\ \Downarrow \\ \text{Contradiction} \\ \Downarrow \\ P \text{ must be true} \end{array}}$$

For a conclusion Q derived from several premises, the same pattern is

$$p_1, \ldots, p_n, P, \neg Q \longrightarrow \text{an impossibility.}$$

The impossibility may be a direct falsehood such as $0 = 1$, or it may be both p_i and $\neg p_i$ (or both P and $\neg P$). Therefore the assumptions cannot all be true. Since the established premises are retained, the temporary assumption $\neg Q$ is rejected, and Q follows.
We saw this pattern in:

- Rational + irrational = irrational.
- Euclid's proof of infinitely many primes.

8.4 Direct Proof vs. Contrapositive vs. Contradiction

These proof strategies are related, but they are not identical.

Method	Basic Idea
Direct proof	Start from the assumptions and derive the desired conclusion.
Contrapositive	To prove $P \to Q$, prove the equivalent statement $\neg Q \to \neg P$. With many premises, $\neg Q$ shows that at least one premise must fail.
Contradiction	Assume the desired conclusion is false while retaining the known premises, then derive an impossibility. This provides the broader framework in which contraposition can be understood.

Takeaway

Good mathematicians do not merely know how to prove things.
They learn to choose the proof strategy that makes the structure of the problem easiest to see.

8.5 Mathematics as Algorithmic Thinking

Look back at the major examples from this lecture.

Prime Testing

Instead of checking every possible divisor up to N, recognize that only divisors up to $\sqrt{N}$ matter.

Even and Odd Proofs

Instead of checking individual numbers, represent every even number as:

$$2k$$

and every odd number as:

$$2k + 1.$$

Euclid

Instead of searching forever for evidence that primes continue, construct a number that forces the existence of another prime.

Connection to Computer Science

These examples share one deep habit:

Look for structure before performing computation.

This habit is one of the foundations of algorithm design.

9 Summary and Revision

9.1 Lecture Summary

In this lecture we introduced the mathematical language required for rigorous Computer Science reasoning.

1. A **proposition** is a statement with a definite truth value.
2. Logical connectors allow propositions to be combined:
$$P \vee Q, \qquad P \wedge Q, \qquad \neg P.$$
3. An implication is written:
$$P \to Q.$$
4. The contrapositive is:
$$\neg Q \to \neg P.$$
With several premises, negating Q implies that at least one premise must fail.
5. De Morgan's laws describe how negation distributes over AND and OR:
$$\neg(P \wedge Q) = \neg P \vee \neg Q,$$
$$\neg(P \vee Q) = \neg P \wedge \neg Q.$$
6. Even integers have the form:
$$2k.$$
7. Odd integers have the form:
$$2k + 1.$$
8. The sum of two even integers is even.
9. The sum of two odd integers is even.
10. A rational plus an irrational number is irrational.
11. Euclid's argument proves that there are infinitely many primes.
12. Primes have exactly two positive divisors, and every composite integer factors into primes.
13. Prime testing only needs to check possible divisors up to:
$$\sqrt{N}.$$
14. Pythagoras' theorem connects geometry with irrational numbers:
$$a^2 + b^2 = c^2.$$
15. If a^2 is even, then a is even.
16. Mathematical proofs establish general truths rather than merely collecting examples.

9.2 The Big Picture

The ideas of this lecture can be viewed as a chain:

Proposition → Logic → Implication
→ Proof → Mathematical Structure → Algorithm

And this chain is fundamental to Computer Science.
A program is not merely a collection of instructions.
A good algorithm is an argument:

Given these inputs and assumptions, these steps will always produce the required result.

Mathematics gives us the language to make that argument precise.

Takeaway

The programmer learns to make a computer do something.

The Computer Scientist learns to explain *why it must work.*

9.3 Quick Revision Sheet

Concept	Key Result
Proposition	A statement that is either True or False.
OR	$P \vee Q$: at least one must be true.
AND	$P \wedge Q$: both must be true.
NOT	$\neg P$: reverses the truth value.
Implication	$P \to Q$: If P, then Q.
Contrapositive	$P \to Q \iff \neg Q \to \neg P$.
Many premises	$\neg Q$ implies at least one premise is false.
Contradiction	Keep the premises, assume $\neg Q$, and derive an impossibility.
De Morgan 1	$\neg(P \wedge Q) = \neg P \vee \neg Q$.
De Morgan 2	$\neg(P \vee Q) = \neg P \wedge \neg Q$.
Even integer	$2k$.
Odd integer	$2k + 1$.
Even + Even	Even.
Odd + Odd	Even.
Rational	p/q, where $p, q \in \mathbb{Z}$ and $q \neq 0$.
Rational + Irrational	Irrational.
Prime testing	Test divisors only up to $\sqrt{N}$.
Pythagoras	$a^2 + b^2 = c^2$.
Odd square	a odd $\to a^2$ odd.
Even square	a^2 even $\to a$ even.
Euclid	There are infinitely many primes.
Prime structure	Every composite integer is a product of primes.

9.4 Questions to Think About

1. Why is the implication $P \to Q$ false only when P is true and Q is false?
2. What is the difference between:
$$P \to Q \qquad \text{and} \qquad Q \to P?$$
3. Why is the contrapositive logically equivalent to the original implication?
4. Can you prove that the sum of an even integer and an odd integer is odd?
5. Can you prove that the product of two odd integers is odd?
6. Why is checking divisors beyond $\sqrt{N}$ unnecessary for basic primality testing?
7. In Euclid's proof, why does adding 1 to the product of all listed primes create a contradiction?
8. Can you give a real programming example where the order of conditions matters?
9. What is the difference between testing a statement with examples and proving it for all possible cases?

Final Reflection

Do not rush to calculate.

First ask:

"What structure is hiding inside the problem?"

That question is at the heart of mathematical thinking, algorithm design, and eventually intelligent systems.