Unit 1: Problem Solving

Python Programming — DI02032011

Milav Dabgar

Lecturer, Dept of ICT
GP Palanpur

April 2, 2026

Unit 1 — Table of Contents

Syllabus Coverage

Weightage: ≈10% (10 marks) | Hours: 5

1.1 Introduction to Problem Solving

Steps, Algorithm & Flowchart

What is Problem Solving?

Definition

Problem solving is the process of identifying a problem, developing a plan to address it, and implementing that plan to arrive at a correct solution.

Steps for Problem Solving

1. **Understand** the problem clearly
2. **Identify** inputs, outputs and constraints
3. **Design** the solution (algorithm/flowchart)
4. **Code** the solution in a programming language
5. **Test & Debug** the program

Importance of Algorithm & Flowchart

Why use Algorithms?

- Clear step-by-step solution
- Language-independent
- Easy to convert to code
- Helps detect logical errors early
- Reusable and shareable

Why use Flowcharts?

- Visual representation of logic
- Easy to understand and follow
- Helps in program planning
- Useful for documentation
- Simplifies complex logic

GTU Tip

Both importance and differences between algorithm and flowchart appear frequently in 3-mark and 4-mark questions.

1.2 Algorithm

Definition, Characteristics & Examples

Algorithm — Definition & Characteristics

Definition (GTU Su'23, W'25 — 3M)

An **algorithm** is a finite, ordered sequence of well-defined instructions for solving a problem or performing a task.

Characteristics of a Good Algorithm

Input	Zero or more inputs
Output	At least one output
Definiteness	Each step is clear and unambiguous
Finiteness	Must terminate after finite steps
Effectiveness	Each step is basic and feasible

Algorithm Examples — Simple Programs

Algorithm: Find Area of Circle (W'25 — 3M)

Step 1: Start
Step 2: Read radius r
Step 3: Calculate area $= \pi \times r^2$
Step 4: Print area
Step 5: Stop

Algorithm: Sum of Two Numbers

Step 1: Start
Step 2: Read two numbers A and B
Step 3: Calculate Sum $= A + B$
Step 4: Print Sum
Step 5: Stop

Algorithm: Maximum of Two Numbers (W'23-4M)

Algorithm: Find Maximum of Two Numbers

Step 1: Start
Step 2: Read two numbers A and B
Step 3: If $A > B$ then
Step 3a: Print "A is greater"
Else
Step 3b: If $A < B$ then Print "B is greater"
Else Print "Both are equal"
Step 4: Stop

Algorithm: Maximum of Three Numbers

Step 1: Start
Step 2: Read A, B, C
Step 3: If $A \geq B$ and $A \geq C$: Print "A is max"
Step 4: Else if $B \geq C$: Print "B is max"
Step 5: Else: Print "C is max"
Step 6: Stop

Algorithm: Simple Interest (Su'23, W'25 — 4M)

Algorithm: Find Simple Interest

Step 1: Start
Step 2: Read Principal P, Rate R, Time T
Step 3: Calculate SI $= \dfrac{P \times R \times T}{100}$
Step 4: Print SI
Step 5: Stop

Algorithm: Positive or Negative Number (W'23 — 3M)

Step 1: Start
Step 2: Read number N
Step 3: If $N > 0$: Print "Positive"
Step 4: Else if $N < 0$: Print "Negative"
Step 5: Else: Print "Zero"
Step 6: Stop

Algorithm: Print Odd Numbers 1 to 20 (W'24-7M)

Algorithm: Print Odd Numbers Between 1 and 20

Step 1: Start
Step 2: Set $i = 1$
Step 3: While $i \leq 20$:
Step 3a: If $i \bmod 2 \neq 0$: Print i
Step 3b: Set $i = i + 1$
Step 4: Stop

Advantages of Algorithm

- Easy to understand — language-independent
- Can be directly converted to any programming language
- Helps to detect and fix logical errors before coding

Algorithm: Sum of First N Even Numbers (Su'25-4M)

Algorithm: Find Sum of Two & Max of Two Numbers

(1) Sum of two numbers:
Step 1: Start Step 2: Read A, B Step 3: Sum $= A + B$ Step 4: Print Sum
Step 5: Stop

(2) Maximum of two numbers:
Step 1: Start Step 2: Read A, B
Step 3: If $A > B$: Print "A is max" Else: Print "B is max"
Step 4: Stop

Algorithm: Even/Odd Check

Step 1: Start Step 2: Read N
Step 3: If $N \bmod 2 = 0$: Print "Even" Else: Print "Odd"
Step 4: Stop

1.3 Flowchart

Symbols, Rules & Examples

Flowchart — Definition & Symbols (W'24, Su'25 — 3M)

Definition

A **flowchart** is a diagrammatic representation of an algorithm using standard symbols to show the flow of control in a program.

Standard Flowchart Symbols

Symbol Shape	Purpose
Oval (Terminator)	Start / Stop of the program
Parallelogram	Input / Output operation
Rectangle (Process)	Processing / Calculation step
Diamond (Decision)	Yes/No decision (if/else)
Arrow (Flow line)	Direction of flow
Circle (Connector)	Connects different parts

Flowchart — Rules & Limitations (Su'23 — 4M)

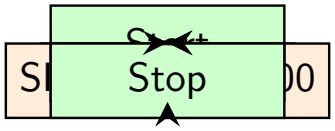
Rules for Drawing a Flowchart

1. Every flowchart must have a **Start** and **Stop** symbol
2. Use **standard symbols** only
3. Flow must go from **top to bottom** or left to right
4. Each symbol has **one entry** and **one exit** point
5. Decision box has **two exits**: Yes and No
6. Use **connectors** to avoid crossing lines

Limitations of Flowchart

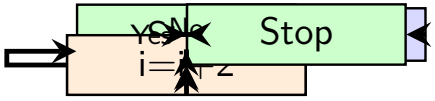
Complex programs $\Rightarrow$ very large and hard to read | Difficult to modify | No standard for detail level

Flowchart: Simple Interest (Su'23, W'25 — 4M)



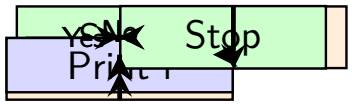
Flowchart: Even or Odd (Su'24 — 4M)

Flowchart: Sum of First 20 Even Numbers (W'24 — 7M)



Flowchart: Positive, Negative or Zero

Flowchart: Print Odd Numbers from 1 to 20 (W'24 OR — 7M)



1.4 Comparison

Flowchart vs Algorithm

Flowchart vs Algorithm (Su'23 OR — 3M)

Comparison Table

Parameter	Algorithm	Flowchart
Form	Written text (steps)	Graphical diagram
Symbols	Plain English / text	Standard shapes
Readability	Easy for programmers	Easy for all
Modification	Easy to edit	Difficult to redraw
Complexity	Can handle complex logic	Gets messy for complex logic
Tool needed	Pen/paper/editor	Drawing tool
Debugging	Harder to visualize	Easier to trace flow

1.5 Pseudocode

Definition & Examples

Pseudocode — Definition (W'23, Su'23 — 3M)

Definition

Pseudocode is an informal, high-level description of an algorithm using structured English-like syntax that is easy to read but not tied to any specific programming language.

Characteristics

- Not a real programming language — no strict syntax
- Uses keywords like `IF`, `ELSE`, `WHILE`, `FOR`, `PRINT`
- Closer to a programming language than plain English
- Easy to convert to actual code
- Bridges the gap between algorithm and code

Pseudocode Examples (W'23 — 3M; Su'23 — 3M)

Check Positive or Negative

```
BEGIN
READ N
IF N > 0 THEN
PRINT "Positive"
ELSE IF N < 0 THEN
PRINT "Negative"
ELSE
PRINT "Zero"
END IF
END
```

Pseudocode: Smallest of Two (Su'23 — 3M)

Find Smallest of Two Numbers

```
BEGIN
READ A, B
IF A < B THEN
PRINT "A is smallest"
ELSE IF B < A THEN
PRINT "B is smallest"
ELSE
PRINT "Both are equal"
END IF
END
```

Pseudocode: Largest of Three (Su'24 — 4M)

Find Largest of Three Numbers X, Y, Z

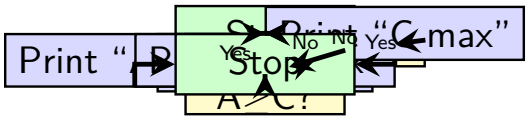
```
BEGIN
READ X, Y, Z
IF X >= Y AND X >= Z THEN
PRINT "X is largest"
ELSE IF Y >= Z THEN
PRINT "Y is largest"
ELSE
PRINT "Z is largest"
END IF
END
```

Pseudocode vs Algorithm vs Flowchart

Quick Comparison

Aspect	Algorithm	Flowchart	Pseudocode
Form	Numbered steps	Graphic diagram	Structured English
Language	Natural/English	Symbols	Semi-code
Coding ease	Good	Moderate	Very good
Modification	Easy	Hard	Easy
Audience	Programmer	Anyone	Programmer

Flowchart: Maximum of Three Numbers



1.6 GTU PYQ Practice

All exam questions from Su'23 – W'25

3-Mark PYQ Questions

Try These (3 marks each)

1. Define flowchart and list any four symbols used. *[W'24, Su'25]*
2. Define algorithm. What are the advantages of Algorithm? *[Su'23]*
3. List the importance of flowchart and algorithm. *[Su'24]*
4. Define Algorithm. Write algorithm to find area of a circle. *[W'25]*
5. Write the difference between flowchart and algorithm. *[Su'23 OR]*
6. Write pseudocode to check if a number is positive or negative. *[W'23]*
7. Define pseudocode. Write pseudocode to find smallest of two numbers. *[Su'23]*

4-Mark PYQ Questions

Try These (4 marks each)

1. State rules for problem solving using flowchart. Design flowchart for SI. *[Su'23, W'25]*
2. Define Algorithm. Design algorithm for max of three numbers. *[W'23]*
3. Draw flowchart to find whether entered number is even or odd. *[Su'24]*
4. Prepare algorithms: (1) max of two numbers (2) sum of two numbers. *[Su'25]*
5. Define pseudocode. Write pseudocode for largest of three X, Y, Z. *[Su'24]*

7-Mark PYQ Questions

Try These (7 marks each)

1. Design flowchart to calculate sum of first 20 even natural numbers. *[W'24]*
2. Create algorithm to print odd numbers between 1 to 20. *[W'24 OR]*

Exam Pattern Reminder

Q.1(a) = 3M **Q.1(b)** = 4M **Q.1(c) / Q.2(c)** = 7M each with an OR option.
Unit 1 questions appear mainly in **Q.1(a)**, **Q.1(b)**, **Q.1(c)**.

Worked: Algorithm + Flowchart — Area of Circle (W'25)

Algorithm

Step 1: Start
Step 2: Read radius r
Step 3: Set $\pi = 3.14159$
Step 4: Area $= \pi \times r^2$
Step 5: Print Area
Step 6: Stop

Key Notes

The algorithm has:
✓ Definite start/stop
✓ Clear input/output
✓ Finite steps
✓ One main calculation
✓ No branching needed

Worked: Algorithm — Odd Numbers 1 to 20 (W'24 OR)

Create algorithm to print odd numbers between 1 to 20

Step 1: Start
Step 2: Initialize $i = 1$
Step 3: If $i > 20$ go to Step 7
Step 4: If $i \bmod 2 \neq 0$ then Print i
Step 5: Set $i = i + 1$
Step 6: Go to Step 3
Step 7: Stop

Output: 1, 3, 5, 7, 9, 11, 13, 15, 17, 19

Practice: Design Tips for Flowcharts

How to Draw a Correct Flowchart in Exam

1. Start with the **Oval** labelled "Start"
2. Use **Parallelogram** for every Read/Print
3. Use **Rectangle** for every calculation
4. Use **Diamond** for every condition (if/loop check)
5. For loops: draw the **back-arrow** from last step to decision
6. End with **Oval** labelled "Stop"
7. Label all **Yes/No** exits from each diamond

Common Mistakes to Avoid

Missing Start/Stop | Unlabelled decision exits | Parallelogram used for calculations | Missing flow arrows

Algorithm Characteristics — Quick Recall

5 Essential Properties (Knuth)

Finiteness	Must terminate after finite number of steps
Definiteness	Each step must be precisely defined
Input	Zero or more inputs from outside
Output	At least one result/output produced
Effectiveness	All steps must be feasible/basic

GTU 3M Answer Template

"An **algorithm** is a finite sequence of well-defined instructions to solve a problem. Its advantages are: (1) language-independent, (2) easy to understand, (3) easy to convert to code, (4) helps detect errors early."

Worked: Two Algorithms (Su'25 — 4M)

(1) Algorithm: Max of Two Numbers

Step 1: Start Step 2: Read A, B
Step 3: IF A $>$ B: PRINT "A is max" ELSE: PRINT "B is max"
Step 4: Stop

(2) Algorithm: Sum of Two Numbers

Step 1: Start Step 2: Read A, B
Step 3: Sum $=$ A $+$ B
Step 4: PRINT Sum
Step 5: Stop

Importance of Flowchart & Algorithm (Su'24 — 3M)

Importance of Flowchart

- Provides a **visual overview** of the solution before coding
- Easier to **communicate** the logic to others
- Helps in **identifying bottlenecks** and logical errors
- Useful for **documentation** and maintenance
- Makes **debugging** easier

Importance of Algorithm

- **Language-independent** blueprint of the solution
- Can be **directly converted** to any programming language
- Promotes **step-by-step logical thinking**
- Helps in **estimating time complexity**
- Useful for **team collaboration**

Algorithm: Factorial of a Number (Su'24 OR, W'24 OR — 7M)

Algorithm: Factorial of N

Step 1: Start
Step 2: Read number N
Step 3: Set $fact \leftarrow 1$, $i \leftarrow 1$
Step 4: Repeat while $i \leq N$:
Step 4a: $fact \leftarrow fact \times i$
Step 4b: $i \leftarrow i + 1$
Step 5: Print $fact$
Step 6: Stop

Note: Factorial Formula

$N! = N \times (N-1) \times (N-2) \times \cdots \times 2 \times 1$

$0! = 1$ (by definition) $\quad 5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$

Algorithm: Fibonacci Series (W'25 OR, Su'25 — 7M)

Algorithm: Print First N Fibonacci Numbers

Step 1: Start
Step 2: Read N
Step 3: Set $a \leftarrow 0$, $b \leftarrow 1$, $count \leftarrow 1$
Step 4: Print a
Step 5: Repeat while $count \leq N - 1$:
Step 5a: Print b
Step 5b: $temp \leftarrow a + b$
Step 5c: $a \leftarrow b$, $b \leftarrow temp$
Step 5d: $count \leftarrow count + 1$
Step 6: Stop

Fibonacci Sequence

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ... Each term = sum of previous two terms.

Algorithm: GCD of Two Numbers (Su'23 OR, W'23 — 7M)

Algorithm: GCD using Euclidean Method

Step 1: Start
Step 2: Read A and B
Step 3: Repeat while $B \neq 0$:
Step 3a: $temp \leftarrow B$
Step 3b: $B \leftarrow A \bmod B$
Step 3c: $A \leftarrow temp$
Step 4: Print A (GCD)
Step 5: Stop

Example

GCD(12, 8): $12 \bmod 8 = 4 \rightarrow$ GCD(8,4): $8 \bmod 4 = 0 \rightarrow$ GCD $= 4$

Algorithm: Swap Two Numbers (W'23 OR, Su'24 — 3M)

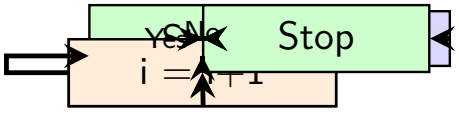
Using Temporary Variable

Step 1: Start
Step 2: Read A, B
Step 3: $temp \leftarrow A$
Step 4: $A \leftarrow B$
Step 5: $B \leftarrow temp$
Step 6: Print A, B
Step 7: Stop

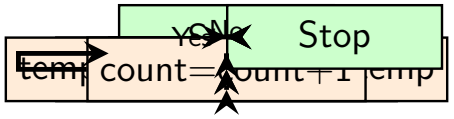
Without Temp Variable

Step 1: Start
Step 2: Read A, B
Step 3: $A \leftarrow A + B$
Step 4: $B \leftarrow A - B$
Step 5: $A \leftarrow A - B$
Step 6: Print A, B
Step 7: Stop

Flowchart: Factorial of N (Su'24 OR, W'24 OR — 7M)



Flowchart: Fibonacci Series (W'25 OR — 7M)



Pseudocode: Factorial (Su'24 OR, W'24 — 7M)

Pseudocode: Compute Factorial of N

BEGIN
INPUT N
SET $fact \leftarrow 1$
SET $i \leftarrow 1$
WHILE $i \leq N$ **DO**
$fact \leftarrow fact \times i$
$i \leftarrow i + 1$
ENDWHILE
PRINT $fact$
END

Trace: N=4

i	fact	i ≤ N?
1	1	Yes
2	2	Yes
3	6	Yes
4	24	Yes
5	—	No ⇒ Print 24

Pseudocode: Fibonacci Series (W'25, Su'25 — 7M)

Pseudocode: Print N Fibonacci Numbers

BEGIN
INPUT N
SET $a \leftarrow 0$, $b \leftarrow 1$, $count \leftarrow 1$
PRINT a
WHILE $count \leq N - 1$ **DO**
PRINT b
SET $temp \leftarrow a + b$
SET $a \leftarrow b$
SET $b \leftarrow temp$
$count \leftarrow count + 1$
ENDWHILE
END

Algorithm: Check Prime Number (W'24, Su'25 — 4M)

Algorithm: Prime Number Check

Step 1: Start
Step 2: Read N
Step 3: Set $flag \leftarrow 1$, $i \leftarrow 2$
Step 4: Repeat while $i \leq N/2$:
Step 4a: If $N \bmod i = 0$: Set $flag \leftarrow 0$; Break
Step 4b: $i \leftarrow i + 1$
Step 5: If $flag = 1$: Print "N is Prime"
Else: Print "N is Not Prime"
Step 6: Stop

Pseudocode: GCD of Two Numbers (Su'23 OR — 7M)

Pseudocode: GCD using Euclidean Method

BEGIN
INPUT A, B
WHILE $B \neq 0$ **DO**
SET $temp \leftarrow B$
SET $B \leftarrow A \bmod B$
SET $A \leftarrow temp$
ENDWHILE
PRINT A $\quad \{A$ is GCD$\}$
END

Trace: GCD(18, 12)

A	B	temp
18	12	12
12	6	6
6	0	$0 \Rightarrow$ Stop

GCD = 6

Pseudocode: Sum of Digits (W'24 OR — 4M)

Pseudocode

BEGIN
INPUT N
SET $sum \leftarrow 0$
WHILE $N > 0$ **DO**
$digit \leftarrow N \bmod 10$
$sum \leftarrow sum + digit$
$N \leftarrow N \div 10$
ENDWHILE
PRINT sum
END

Trace: N=1234

N	digit	sum
1234	4	4
123	3	7
12	2	9
1	1	10
0	–	Stop

Sum = 10

Types of Errors in Problem Solving (W'23 OR — 3M)

1. Syntax Error

Grammatical mistake in the program; detected by compiler/interpreter.
Example: Missing colon, wrong indentation, misspelled keyword.

2. Runtime Error

Error that occurs during execution; program crashes unexpectedly.
Example: Division by zero, accessing invalid index.

3. Logical Error

Program runs but produces wrong output; hardest to detect.
Example: Using + instead of − in a formula.

Steps in Problem Solving / Software Development

1. **Problem Analysis** — Understand input, output, and constraints.
2. **Algorithm Design** — Write step-by-step logical solution.
3. **Flowchart / Pseudocode** — Visualize or describe the logic.
4. **Coding** — Translate algorithm to a programming language.
5. **Testing & Debugging** — Run with sample inputs; fix errors.
6. **Documentation** — Record how the program works.
7. **Maintenance** — Update as requirements change.

Key Point

Algorithm and Flowchart are done **before** coding. They are **language-independent** tools.

Algorithm: Reverse a Number (Su'23, W'25 OR — 3M)

Algorithm: Reverse Digits of N

Step 1: Start
Step 2: Read N
Step 3: Set $rev \leftarrow 0$
Step 4: Repeat while $N > 0$:
Step 4a: $digit \leftarrow N \bmod 10$
Step 4b: $rev \leftarrow rev \times 10 + digit$
Step 4c: $N \leftarrow N \div 10$
Step 5: Print rev
Step 6: Stop

Trace: N=456

456 $\rightarrow$ digit=6, rev=6 | 45 $\rightarrow$ digit=5, rev=65 | 4 $\rightarrow$ digit=4, rev=654 | 0 $\rightarrow$ Stop.
Result: 654

Worked: Algorithm & Flowchart for Factorial (7M)

Algorithm

Step 1: Start
Step 2: Read N
Step 3: $fact = 1$, $i = 1$
Step 4: While $i \leq N$:
$fact = fact \times i$
$i = i + 1$
Step 5: Print $fact$
Step 6: Stop

Key Points

- Initialize before loop
- Update inside loop
- Print after loop

$4! = 4 \times 3 \times 2 \times 1 = 24$
$5! = 120$, $6! = 720$

Examiner mark scheme:
Algorithm: 4M
Flowchart: 3M

Worked: Algorithm & Pseudocode for GCD (7M)

Algorithm

Step 1: Start
Step 2: Read A, B
Step 3: While $B \neq 0$:
$temp = B$
$B = A \bmod B$
$A = temp$
Step 4: Print A (GCD)
Step 5: Stop

Trace: GCD(24,16)

A=24, B=16
$24 \bmod 16 = 8 \rightarrow$ A=16, B=8
$16 \bmod 8 = 0 \rightarrow$ A=8, B=0
B=0: Stop. GCD = **8**

LCM: $\text{LCM} = \dfrac{A \times B}{\text{GCD}}$

Worked: Algorithm + Flowchart for Fibonacci (7M)

Algorithm

Step 1: Start
Step 2: Read N
Step 3: $a = 0,\ b = 1,\ c = 1$
Step 4: Print $a,\ b$
Step 5: While $c \leq N - 2$:
$next = a + b$
Print $next$
$a = b,\ b = next$
$c = c + 1$
Step 6: Stop

First 8 Terms (N=8)

0, 1, 1, 2, 3, 5, 8, 13

Pattern:

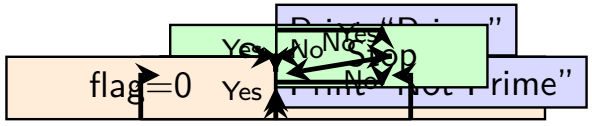
$F_n = F_{n-1} + F_{n-2}$

$F_0 = 0, F_1 = 1$
$F_2 = 1, F_3 = 2$
$F_4 = 3, F_5 = 5$
$F_6 = 8$

Algorithm Quick Reference Card

Problem	Key Steps in Algorithm	GTU
Factorial of N	Init fact=1,i=1; loop: fact*=i,i+=1	7M
Fibonacci N terms	a=0,b=1; print a; loop: print b, update	7M
GCD of A,B	Euclidean: while B≠0: A,B = B, A%B	7M
Swap A,B	temp=A; A=B; B=temp	3M
Largest of 3	if A≥B and A≥C: max=A	4M
Sum of digits	while N>0: sum+=N%10; N//=10	4M
Reverse number	while N>0: rev=rev*10+N%10; N//=10	3M
Count digits	while N>0: count+=1; N//=10	3M
Prime check	flag=1; for i=2 to N/2: if N%i=0: flag=0	4M
SI	SI = P*R*T/100 (direct formula)	4M

Flowchart: Check Prime Number (W'24, Su'25 — 4M)



Common Mistakes & Exam Tips for Unit 1

Common Mistakes

× Forgetting **Start/Stop** in flowchart
× Using wrong symbol (rectangle for decision)
× Not labelling Yes/No on diamond
× Algorithm without step numbers
× Not initializing variables
× Infinite loop (no update in loop)
× Writing code instead of pseudocode

Exam Tips

✓ 7M: Write both algorithm + flowchart
✓ 4M: Algorithm or flowchart (as asked)
✓ 3M: Definition + comparison table
✓ Use proper notation: $\leftarrow$ or $=$
✓ Show variable trace for verification
✓ Neat flowchart with arrows
✓ Number all algorithm steps

Unit 1 — Summary

Key Formulas & Concepts

Algorithm	Finite ordered steps; language-independent
Flowchart	Graphical using standard symbols
Pseudocode	Structured English; between algo and code
Symbols	Oval=Start/Stop, Parallelogram=I/O, Rectangle=Process, Diamond=Decision
Rules	Start/Stop, top-bottom, label Yes/No

Exam Weightage (Unit 1 — 10%)

3-mark × 1–2 | 4-mark × 1 | 7-mark × 0–1