Course «Algorithms and Data Structures»

6B06102 – Information Systems
6B06109 – Artificial Intelligence
6B06103 – Computer Engineering
6B06104 – Computer Science·

Lecture 2



Pseudocode and Flowcharts

Textbook: A. Levitin, Sections 1.1–1.2

How do we write an algorithm down?

Natural language

easy to read, but ambiguous

Pseudocode

precise, but not tied to any language

Flowchart

a picture: good for small algorithms

And finally: a program in a real programming language.

Running example: Euclid’s algorithm

The greatest common divisor gcd(m, n) is the largest integer that divides both m and n.

$$\gcd(m, n) = \gcd(n, m \bmod n), \gcd(m, 0) = m$$

$$\gcd(60, 24) = \gcd(24, 12) = \gcd(12, 0) = 12$$

In natural language

1. If n = 0, return m and stop; otherwise go to Step 2.
2. Divide m by n and assign the remainder to r.
3. Assign the value of n to m and the value of r to n. Go to Step 1.

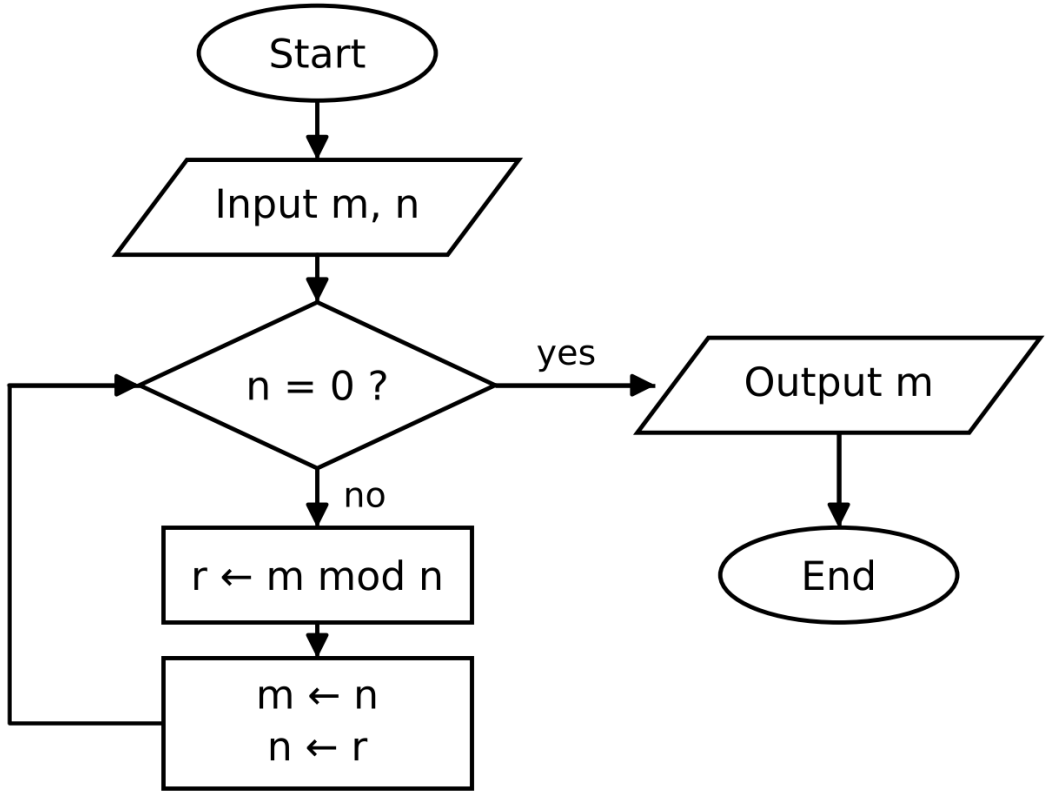
A student writes: “Divide the bigger number by the smaller one and keep doing that until the remainder is zero.” Keep doing **what**? Divide **which** numbers? Words are ambiguous.

In pseudocode

```
ALGORITHM Euclid(m, n)
    //Computes gcd(m, n) by Euclid's algorithm
    //Input: Two nonnegative, not-both-zero integers m and n
    //Output: Greatest common divisor of m and n
    while n ≠ 0 do
        r ← m mod n
        m ← n
        n ← r
    return m
```

Short, precise, and any programmer can turn it into code.

As a flowchart



As a program (Python)

```
def gcd(m, n):
    while n != 0:
        r = m % n
        m = n
        n = r
    return m
```

Almost a line-by-line copy of the pseudocode: pseudocode is the idea, the program is that idea for one machine.

Pseudocode conventions (Levitin)

Construct	How we write it	Example
Header	ALGORITHM Name(parameters)	ALGORITHM Euclid(m, n)
Comment	// text	//Input: an array
Assignment	←	maxval ← A[0]
Condition	if … else	if A[i] > maxval
Counting loop	for i ← a to b do	for i ← 1 to n – 1 do
Conditional loop	while condition do	while n ≠ 0 do
Result	return value	return m

Assignment is an arrow, not “=”

x ← x + 1

an **action**: put a new value into x

x = y

a **question**: are they equal?

In Python and C: = for assignment and == for comparison. Mixing them up is one of the most common beginner mistakes.

Three more rules

- **Indentation shows structure.** No begin/end, no braces.
- **Every algorithm starts with //Input and //Output.** They say which inputs are legitimate.
- **One line = a few simple operations.** “sort A” or “find the maximum” is a whole algorithm, not one step.

Example: the largest element

```
ALGORITHM MaxElement(A[0..n − 1])
    //Determines the value of the largest element
    maxval ← A[0]
    for i ← 1 to n − 1 do
        if A[i] > maxval
            maxval ← A[i]
    return maxval
```

Remember the first number as “the biggest so far”; replace it whenever you see a bigger one.

Tracing by hand: A = 3, 8, 2, 9, 5

Step	i	A[i]	A[i] > maxval?	maxval
start	–	–	–	3
1	1	8	yes	8
2	2	2	no	8
3	3	9	yes	9
4	4	5	no	9

C(n) = n – 1 comparisons

Example: sequential search

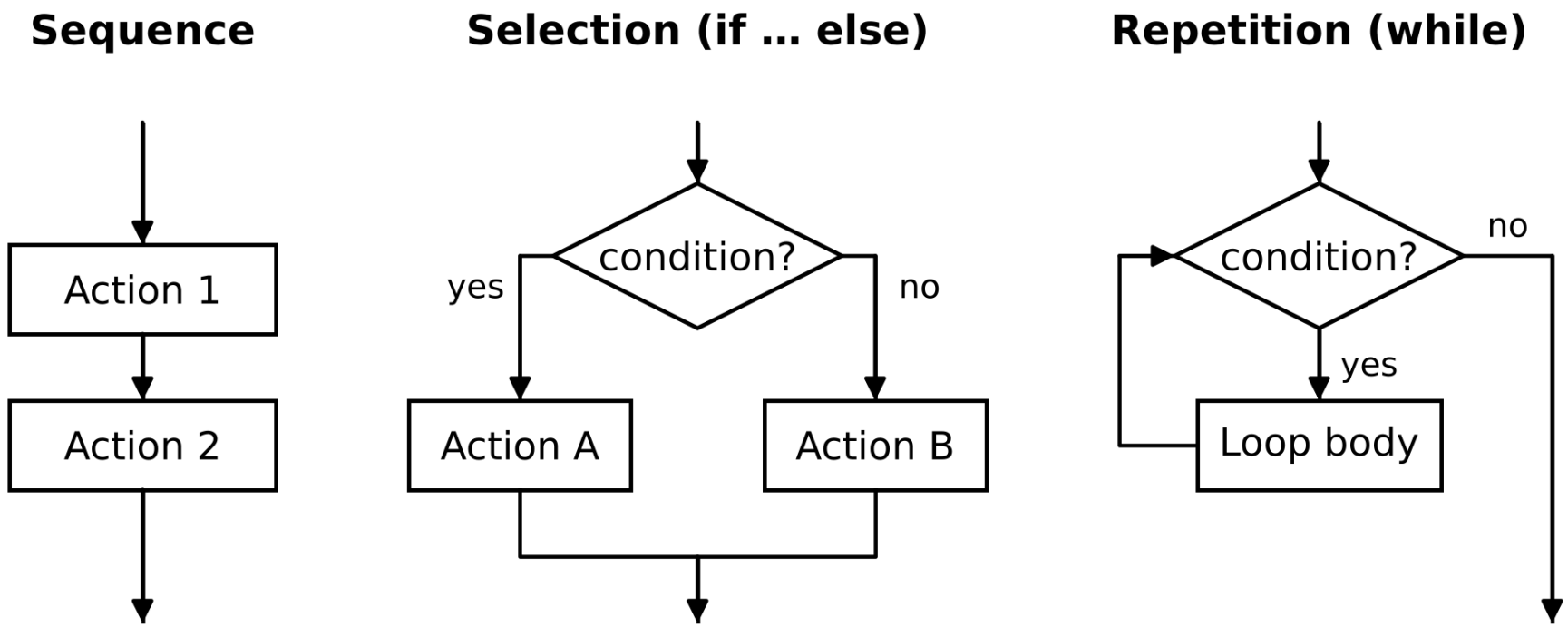
```
ALGORITHM SequentialSearch(A[0..n − 1], K)
    i ← 0
    while i < n and A[i] ≠ K do
        i ← i + 1
    if i < n return i
    else return −1
```

“and” stops as soon as the first part is false. When i = n, the element A[n] is never checked. Swap the two conditions and the algorithm breaks.

Flowchart symbols

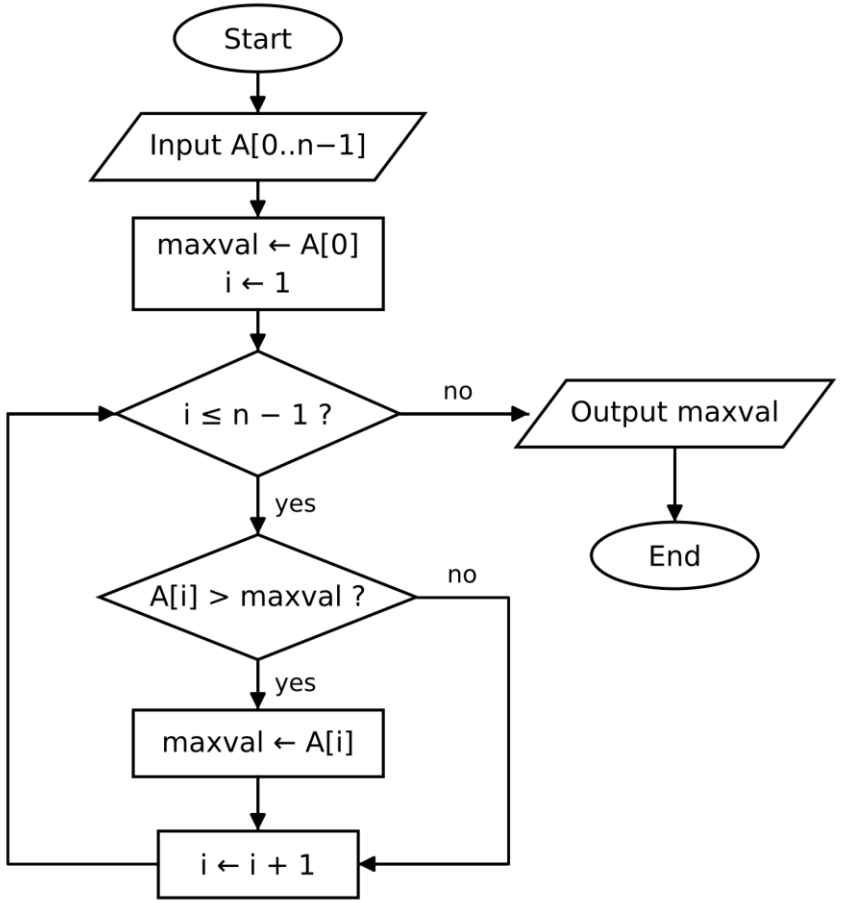
Shape	Meaning	Example
Oval	start or end	Start, End
Parallelogram	input or output	Input m, n
Rectangle	processing, assignment	r ← m mod n
Diamond	decision: two exits, yes and no	n = 0 ?
Arrow	the order of execution	

Three building blocks



Böhm and Jacopini (1966): any algorithm can be built from these three structures.

MaxElement as a flowchart



- Outer diamond with the back arrow: **repetition** (the for loop)
- Inner diamond: **selection** (the if)
- Boxes in a row: **sequence**
- One line of pseudocode, “for i ← 1 to n – 1”, becomes three shapes

Pseudocode or flowchart?

	Pseudocode	**Flowchart**
Strong side	compact, precise, close to code	visual, control flow is easy to see
Weak side	conventions to learn	grows very fast, hard to edit
Good for	any algorithm, and for analysis	short algorithms, teaching

Levitin: flowcharts are inconvenient for all but very simple algorithms. From now on we use pseudocode.

Summary

- Natural language is ambiguous; pseudocode is precise but language-independent.
- Our conventions: ← for assignment, // for comments, indentation for structure, Input and Output in the header.
- Flowcharts use ovals, parallelograms, rectangles, diamonds and arrows.
- Every algorithm is built from sequence, selection and repetition.

Review questions

1. Why is natural language a poor way to specify an algorithm? What does pseudocode add, and what does it leave out compared with a program?
2. Write pseudocode that counts the negative elements of A[0..n – 1]. Draw its flowchart and mark sequence, selection and repetition.
3. Trace Euclid’s algorithm on m = 31415, n = 14142. How many iterations does the loop make?

Reading: Levitin, Sections 1.1–1.2; Cormen et al., Section 2.1.