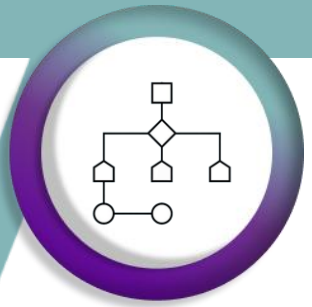
ALGORITHM

Project number : 2023-2-IT03-KA220-YOU-000179130

Co-funded by
the European Union

Advanced Lesson Plan



- Define an algorithm in a technical sense and explain it simply.
- Understand algorithm efficiency: know what “time complexity” and “space complexity” mean.
- Identify common types of algorithms: sorting, searching, recursive, iterative, greedy, etc.
- Analyze an algorithm step by step to check correctness and efficiency.
- Write simple algorithms in pseudocode or flowcharts.

Co-funded by
the European Union

Video

DEAF YOUNG CODE

Co-funded by
the European Union

What is an Algorithm?

DEAF YOUNG CODE

Technical Definition:

An algorithm is a finite sequence of well-defined instructions to solve a problem.

Co-funded by the European Union

Key Characteristics

DEAF YOUNG CODE

1. Input: The data you start with.
2. Output: The result or solution.
3. Finiteness: Must finish after a limited number of steps.
4. Definiteness: Each step is clearly defined.
5. Effectiveness: Each step can be performed in a reasonable amount of time.

Algorithms basics

DEAF YOUNG CODE

How Algorithms Work

- **Define the problem clearly** → What do we want to solve?
- **Design a step-by-step procedure** → What are the instructions to reach a solution?
- **Test with examples** → Check if the algorithm works with sample inputs.
- **Analyze efficiency** → How fast is it? How much memory does it use?
- **Refine and optimize** → Can it be faster or use less memory?

Algorithm Diagram (Flowchart)

DEAF YOUNG CODE

A flowchart is a visual way to show an algorithm step by step.
It helps you see the logic, not just read it.

Element	Symbol	Function	Example
Start / End	Oval 	Shows where the algorithm begins or ends	“Start” / “End”
Process / Step	Rectangle 	An action or instruction	“Add two numbers”
Decision / Condition	Diamond 	A question that changes the flow (Yes/No)	“Is a > b?”
Input / Output	Parallelogram 	Data that goes in or comes out	“Input numbers” / “Output result”
Arrow	→ 	Shows the direction of the flow	
Loop / Connector	Curved arrow or label 	A step that repeats until a condition is met	“Repeat until sorted”

Algorithm Diagram (Flowchart)

DEAF YOUNG CODE

How to Read a Flowchart

- Start → Find the “Start” oval.
- Input → Look at what information enters the system (for example: numbers, text).
- Process → Follow each rectangle. These are actions or steps.
- Decision → When you reach a diamond, it asks a Yes/No question.
  - If “Yes,” follow one arrow.
  - If “No,” follow the other.
- Loop → When an arrow goes back to an earlier step, it means “repeat.”
- Output → At the end, see the result or answer.
- End → The process stops.

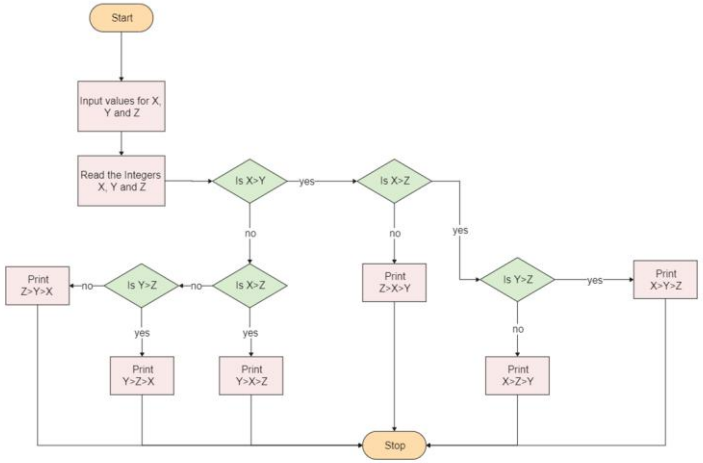
Examples: Sorting Numbers

DEAF YOUNG CODE

Problem: Sort the numbers [4, 2, 7] from smallest to largest

1. Flowchart (Step-by-Step Visual)

```
Start
  |
  v
Input numbers: 4, 2, 7
  |
  v
Compare first two numbers:
Is 4 > 2? Yes → Swap
  |
  v
Numbers now: 2, 4, 7
  |
  v
Compare next numbers: 4 and 7
Is 4 > 7? No → Keep order
  |
  v
Output sorted numbers: 2, 4, 7
  |
  v
End
```



Tip for Deaf Learners:

- Use arrows and boxes for each step.
- Highlight the swap action with a different color.
- Show input → process → output clearly.

Co-funded by the European Union

Examples: Sorting Numbers

DEAF YOUNG CODE

Problem: Sort the numbers [4, 2, 7] from smallest to largest

2. Pseudocode (Simple Text Version)

```
Algorithm SortThreeNumbers:
  Input: a, b, c
  If a > b then
      Swap a and b
  If b > c then
      Swap b and c
  If a > b then
      Swap a and b
  Output: a, b, c
End Algorithm
```

Notes:

- **Pseudocode is like English instructions, not programming language.**
- **Step-by-step clarity is key.**

Co-funded by the European Union

Examples: Sorting Numbers

DEAF YOUNG CODE

Problem: Sort the numbers [4, 2, 7] from smallest to largest

3. Diagram (Visual Overview)

```
[Input: 4,2,7]
      |
      v
[Compare 4 and 2]
      |
      v
[Swap if needed] ---> [Numbers now: 2,4,7]
      |
      v
[Compare 4 and 7]
      |
      v
[Swap if needed] ---> [Numbers now: 2,4,7]
      |
      v
[Output: 2,4,7]
```

Co-funded by the European Union

Key Points

- Flowchart = detailed, step-by-step map of the algorithm.
  - Includes decisions, loops, inputs, outputs.
  - Great for writing pseudocode or coding later.
- Diagram = big-picture visual, simple overview.
  - Shows main actions and flow without all steps.
  - Helps understand the logic quickly, especially for visual learners.

DEAF YOUNG CODE

Co-funded by the European Union

Algorithm Theory Important

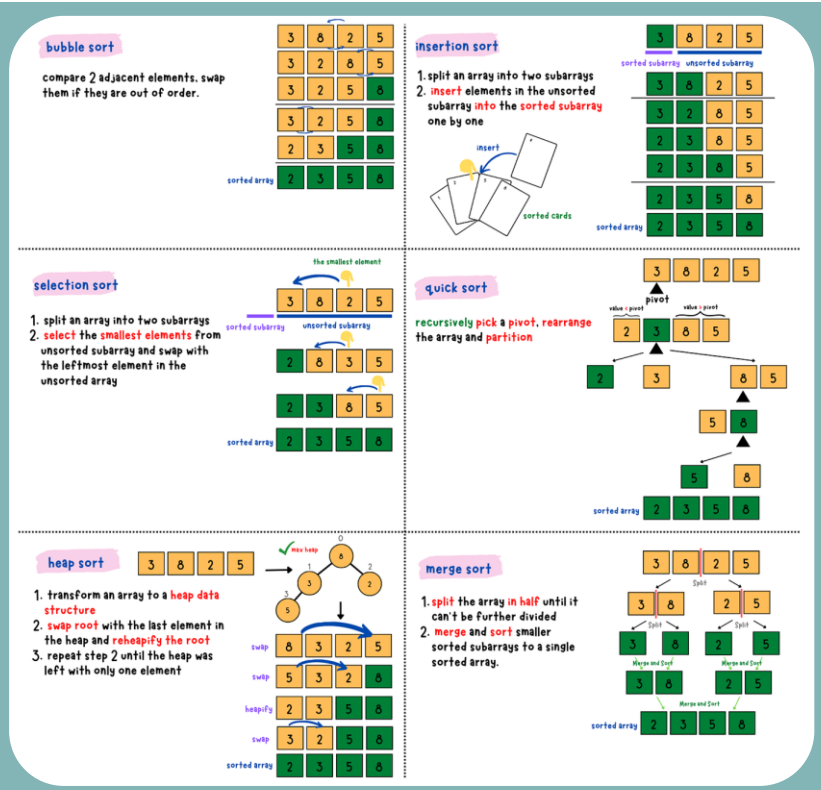
- **Time Complexity (Big O notation): Measures how fast an algorithm runs as input size grows.**
- **Space Complexity: Measures how much memory an algorithm uses.**
- **Common Algorithm Types:**
  - Sorting: Bubble sort, merge sort, quicksort.
  - Searching: Linear search, binary search.
  - Recursion: Algorithm calls itself (e.g., factorial).
  - Greedy Algorithms: Make the best choice at each step (e.g., shortest path).
  - Dynamic Programming: Break problems into smaller subproblems.
- **Data Structures: Algorithms often need lists, arrays, trees, or graphs.**

DEAF YOUNG CODE

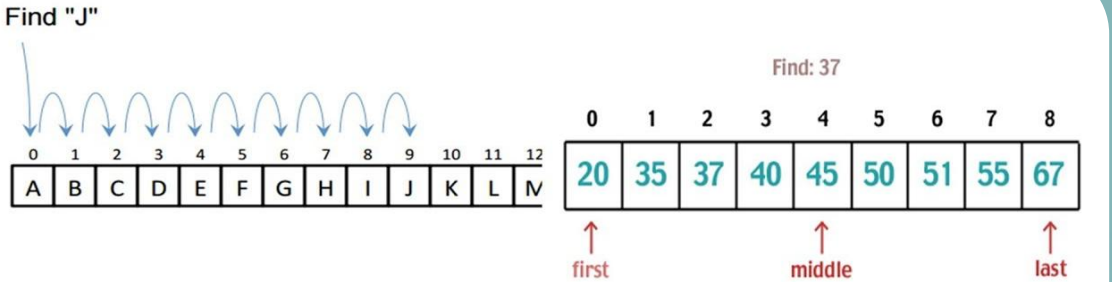
Co-funded by the European Union

Comparison-Based Sorting

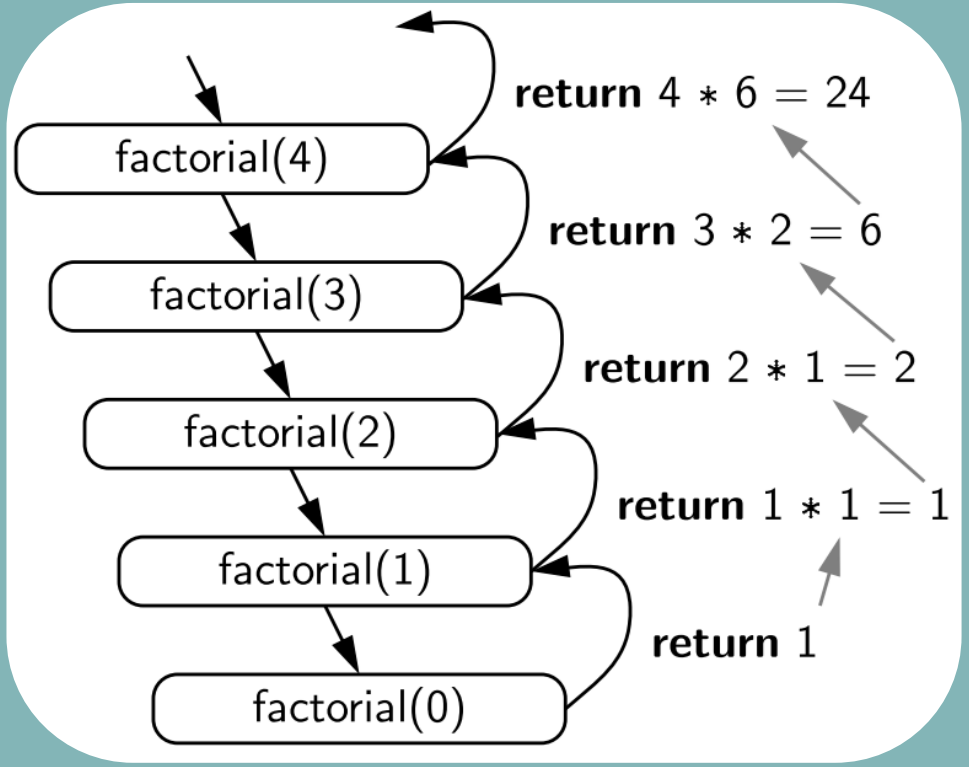
DEAF YOUNG CODE



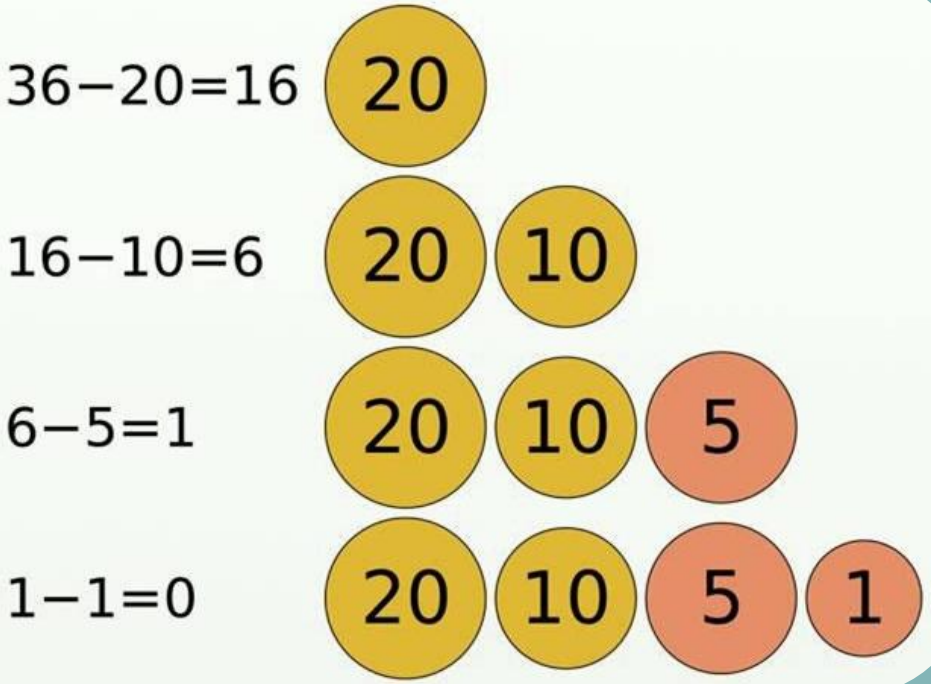
Comparison-Based Sorting



Comparison-Based Sorting

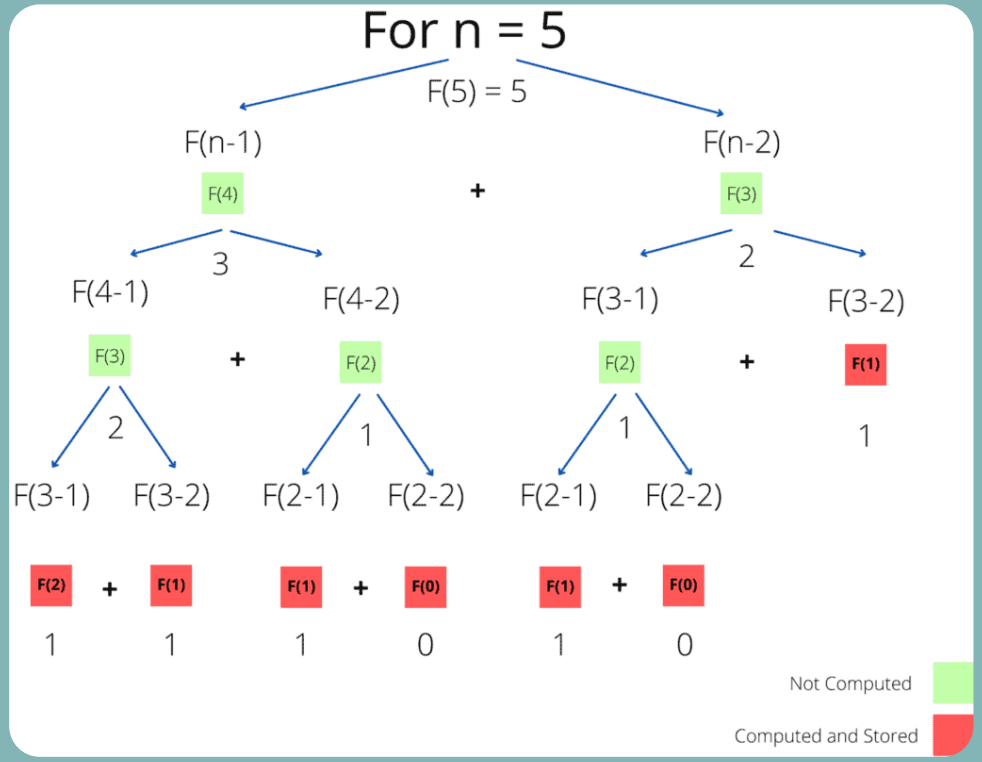


Comparison-Based Sorting



DEAF YOUNG CODE

Comparison-Based Sorting



Interactive Exercise Idea (Hands-On Learning)

DEAF YOUNG CODE

Objective

- Help students practice algorithm design in a fun and visual way.
- Reinforce understanding of steps, decisions, loops, and efficiency.

Interactive Exercise Idea (Hands-On Learning)

DEAF YOUNG CODE

Step 1: Choose a Simple Task

- Examples of everyday tasks:
- Making tea
- Sorting books by height
- Morning routine (wake up → brush teeth → eat breakfast → get dressed)
- Packing a backpack
- Ordering fruits by size

Interactive Exercise Idea (Hands-On Learning)

DEAF YOUNG CODE

Step 2: Draw a Flowchart

- Instructions for students:
- Start → Draw the start symbol (oval).
- Input / Output → Show the data or result (parallelogram).
- Actions / Steps → Use rectangles for each action.
- Decision Points → Use diamonds for Yes/No questions.
- Loops → Draw arrows that go back to repeat steps if needed.
- End → Show the final result or completion (oval).
- Tip: Use colors or sticky notes for actions, decisions, and outputs.

Interactive Exercise Idea (Hands-On Learning)

DEAF YOUNG CODE

Step 3: Share and Compare

- Ask each student or group to present their flowchart.

Discuss:

- Did they include all important steps?
- Are there unnecessary steps that can be removed?
- Can the task be done faster or more efficiently?

Interactive Exercise Idea (Hands-On Learning)

DEAF YOUNG CODE

Step 4: Optional Challenge (Efficiency)

- Introduce the idea of algorithm efficiency:
  - Fewer steps = faster algorithm
  - Repeating unnecessary steps = slower algorithm

Interactive Exercise Idea (Hands-On Learning)

DEAF YOUNG CODE

Step 4: Optional Challenge (Efficiency)

- Example: Sorting 3 books
  - **Student A:** Checks all pairs twice → 6 steps
  - **Student B:** Smart swaps only → 3 steps
- Ask: **Which algorithm is better? Why?**
- Show that **even simple everyday tasks** can have “more efficient” algorithms.

Interactive Exercise Idea (Hands-On Learning)

DEAF YOUNG CODE

Step 5: Hands-On Bonus (Kinesthetic Learning)

- Use **real objects**: books, cups, sticky notes.
- Students **act out the steps** physically, following their own flowchart.
- Helps **visual and tactile learners** understand sequence, loops, and decisions.

Thank you for your attention!

DEAF YOUNG CODE

Stay tuned for the next presentation!

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA).
Neither the European Union nor EACEA can be held responsible for them.

Project number : 2023-2-IT03-KA220-YOU-000179130

Co-funded by the European Union