CBS Department

Introduction to Data Structures & Algorithms

Soma Soleiman Zadeh

Data Structures & Algorithms (CBS 216)

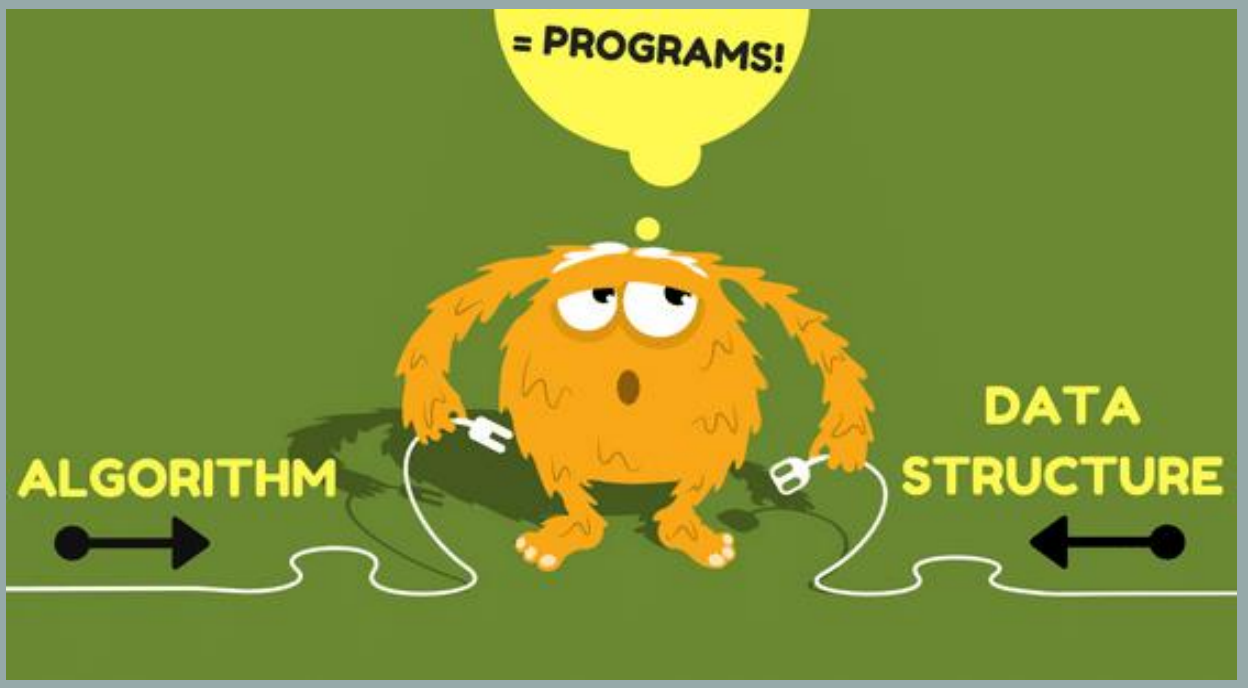
Spring 2025 - 2026

Week 1

February 02-03, 2026

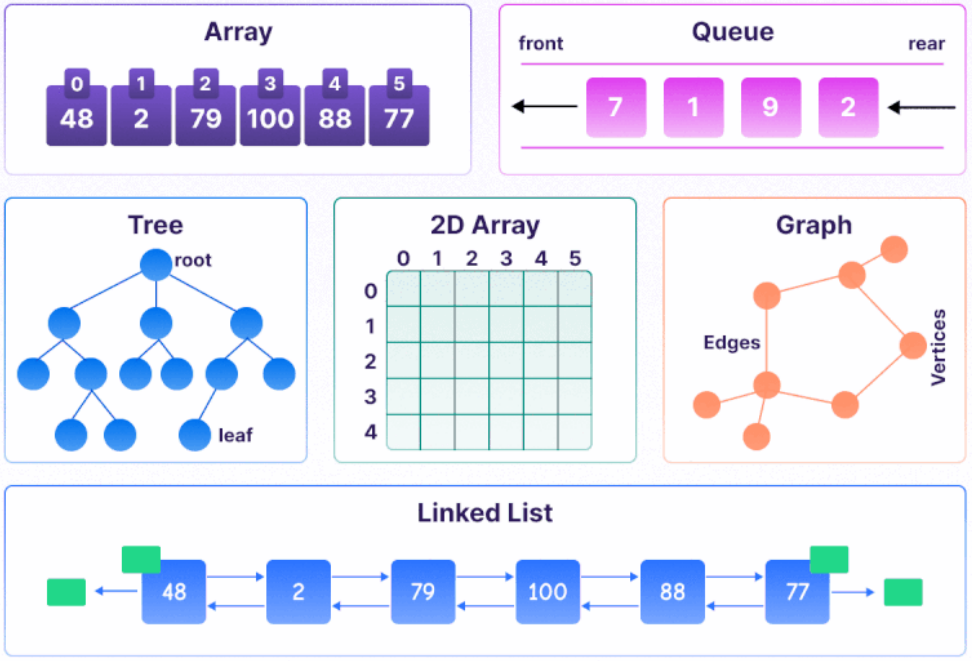
Outline

- Why **Data Structures**?
- Why **Algorithms**?
- Classifications of **Data Structures**
- Operations on **Data Structures**
- Analysis of **Algorithms**



Data Structures

- Programming largely works with **data**.
- Computer programs are all about receiving, modifying, and returning data.
- **Data Structure** is a special format for **storing and organizing data** in a computer so that **it can be used efficiently**.



Classification of Data Structures

- There are two types of data structures in terms of the organization of elements:

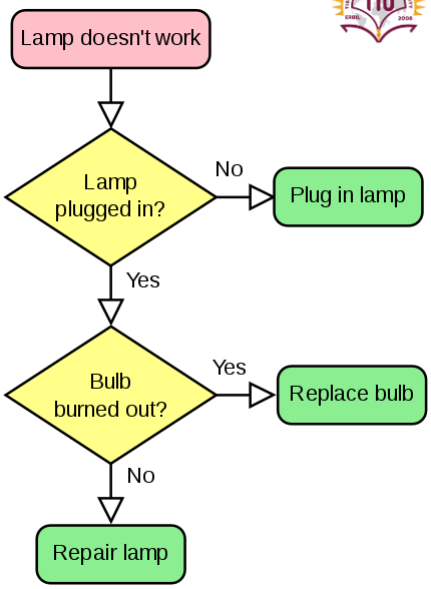
1. **Linear Data Structures:** Elements are accessed in a sequential order, but it is not compulsory to store all elements sequentially.
   - **Linked Lists**, **Stacks**, and **Queues**
2. **Non-linear Data Structures:** Elements of this data structure are stored/accessed in a non-linear order.
   - **Trees** and **Graphs**

Operations on Data Structures

- There are four basic operations on most data structures:
  - **Read:** Reading a value from a particular index within the data structure.
  - **Search:** Searching refers to looking for a particular value within a data structure.
  - **Insert:** Insertion refers to adding another value to our data structure.
  - **Delete:** Deletion refers to removing a value from our data structure.

Algorithms

- An **algorithm** is a step-by-step set of clear instructions **to solve a given problem** in a finite amount of time.
- Two criteria for evaluating algorithms:
  1. **Correctness**
  2. **Efficiency**

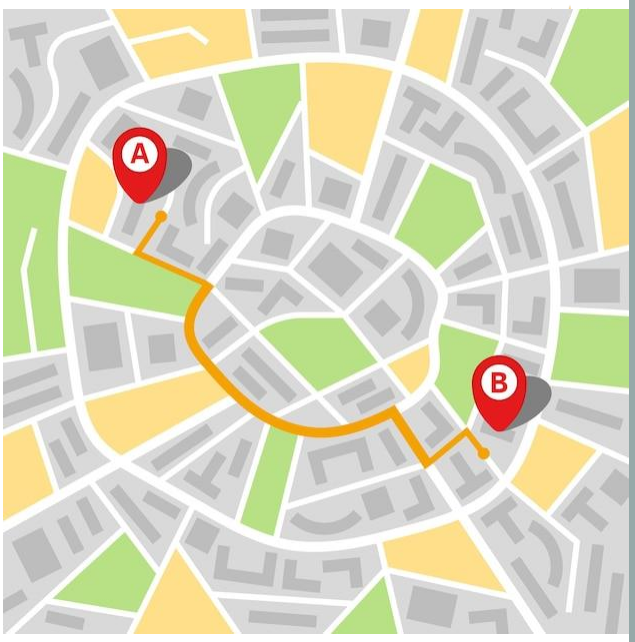


Origin of the Word "Algorithm"

- The word "Algorithm" comes from the name of a 9th-century mathematician, Muhammad ibn Musa Al-Khwarizmi.
- He made substantial contributions in astronomy and mathematics (especially in algebra and trigonometry).

Choosing Best Path

- If you are going to move from **point A** to **point B**, which path do you choose?
  - A path shorter in distance?
  - A path shorter in time?



Algorithm vs. Program

- **Algorithm:** step-by-step list of instructions for solving a problem.
- **Program:** an algorithm that has been encoded into some programming language.
- **There may be many programs for the same algorithm.**
- When two programs solve the same problem in different ways, is one program better than the other?

Good Data Structure and Good Algorithm

- We are interested in designing **"good" data structures** and **algorithms**.
- How can we classify a data structure and algorithm as **good**?
- We need some criteria to analyze an algorithm.
    - **running time**
    - memory,
    - developer effort,
    - Etc.

Algorithm Analysis

Both the following codes will compute the sum of the first ***n*** integers.

```
def sum_1(n):
    summation = 0
    for i in range(1,n+1):
        summation = summation + i
    return summation
```

```
def sum_2(n):
    summation = 0
    for i in range(1,n+1):
        num = i
        summation = summation + num
    return summation
```

Which program is more readable?

Which program is faster?

Algorithm Analysis - Using Computing Resources

- We analyze algorithms based on **the amount of computing resources** that each algorithm uses.
- One algorithm is better than the other because it **uses fewer computing resources**.
- **Computing resources:**
  - the **amount of memory** an algorithm needs to solve the problem.
  - the **amount of time** an algorithm needs to execute. (**Running Time**)

Algorithm Analysis – Running Time

- How can we measure the running time of an algorithm?
  - Tracking the actual time taken by the program to compute its result.
  - Recording the starting time and ending time of the program.
- How can we measure the running time of an algorithm in **Python**?
  - By importing the **time** module and using a function named **time**.
  - **time** function will return the current system clock time in seconds.

Algorithm Analysis – Running Time

- How can we measure the running time of an algorithm?
  - Tracking the actual time taken by the program to compute its result.
  - Recording the starting time and ending time of the program.
- How can we measure the running time of an algorithm in **Python**?
  - By importing the **time** module and using a function named **time**.
  - **time** function will return the current system clock time in seconds.

Measuring Running Time

- Implement the algorithm in Python
- Run it
- Time it
- Repeat for different input sizes

Is Running Time a Good Measure for Algorithm Analysis?

- **Input size** → The running time might increase if we increase the input size.
- **Hardware or Programming Language** → The running time might change if we run the same code on a different computer or a different programming language.
- **Think of a better measure for Algorithm Analysis!**

Algorithm Analysis – Number of Steps

- When we decide on how "fast" an algorithm is, it is better to measure **how many steps it takes**.

```
def sum_1(n):
    summation = 0
    for i in range(1,n+1):
        summation = summation + i
    return summation
```

1 operation

2 operations per loop

1 operation

Algorithm Analysis – Number of Steps

```
def sum_1(n):
    summation = 0
    for i in range(1,n+1):
        summation = summation + i
    return summation
```

1 operation

2 operations per loop

1 operation

Total number of operations depends on loop length, and loop length depends on value n.

T(n) = 2n + 2

Algorithm Analysis – Number of Steps

- When we decide on how "fast" an algorithm is, it is better to measure **how many steps it takes**.

```
def sum_2(n):
    summation = 0
    for i in range(1,n+1):
        num = i
        summation = summation + num
    return summation
```

1 operation

1 operation per loop

2 operations per loop

1 operation

Thank You!