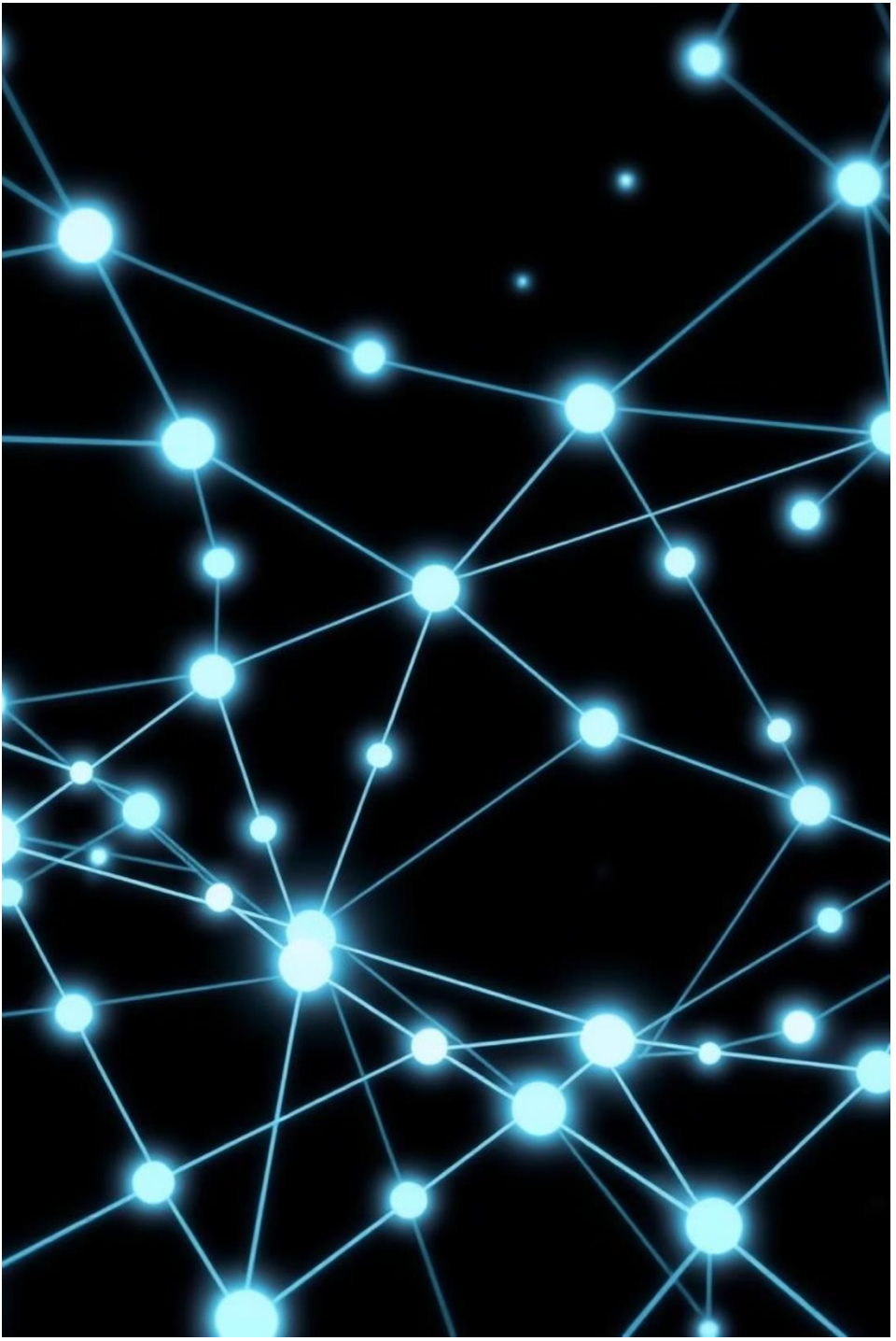


Data Structures

Lecture 1: Introduction to DS

Instructor:
Md Samsuddoha
Assistant Professor
Dept of CSE, BU

Contents

- Data & Information
- Concept of Data Structures (DS)
- Examples of DS
- Types of DT & DS
- Common DS
- Importance & Applications of DS
- Concept of Algorithms
- Homework (Lab)

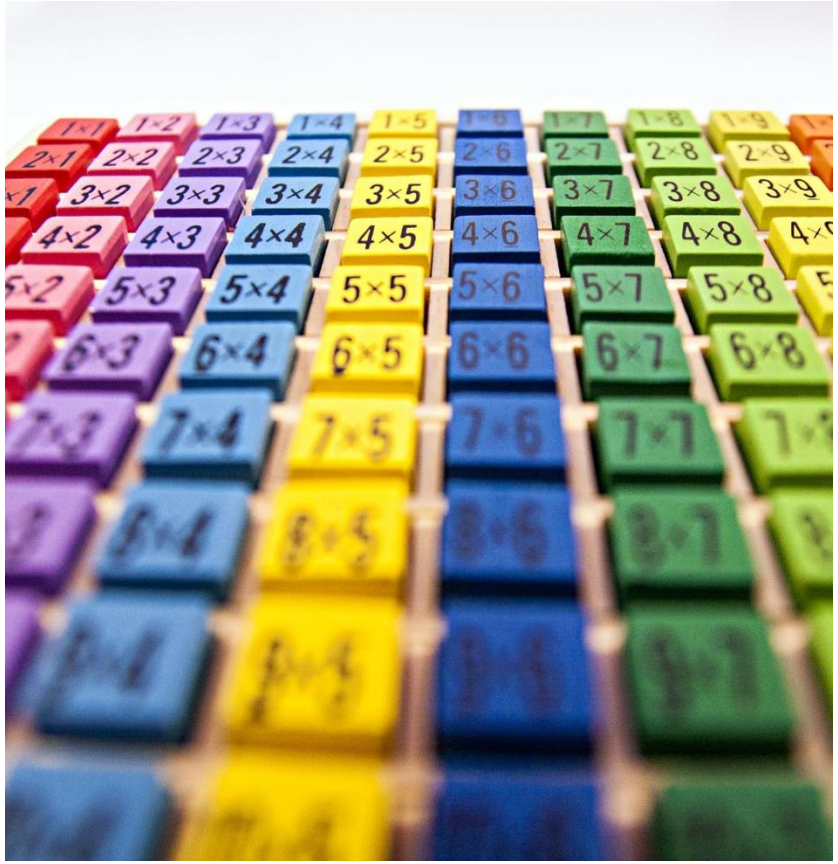
Data

- **Data is a raw, unprocessed collection of facts, figures, or symbols.**
- Data refers to raw, unprocessed facts and figures that lack inherent meaning or context on their own. It's the most basic element in the information hierarchy.
- **Examples**
  - Temperature readings from a sensor (e.g., "25.7", "78.3")
  - Individual survey responses (e.g., "Yes", "No", "5")
  - Levo I shbanglade
- ***Data does not carry meaning until it is processed.***

Information

- Information is **processed, organized, or structured data** that has meaning and is useful for decision-making.
- Meaningful data → Information

- **Examples**
  - Average temperature for the week: 32.2°C
  - Class attendance report: 85% of students attended regularly
  - I Love Bangladesh
- **Information = Data + Context + Meaning**



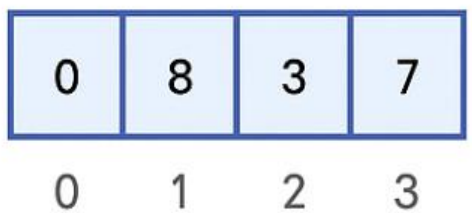
Data Structures

- A Data Structure is a **specialized format/ way/ technique** for **organizing, storing, and managing data** so that it can be accessed and modified efficiently.
- It defines:
  - The **organization** of data (arranged/ stored in memory)
  - The **relationship** between data elements (logical connection: chain, tree)
  - The **operations** that can be performed on the data (Insertion, deletion, updating, searching, traversal)

- Organization → Relationship → Operations

The Organization of Data

How data is arranged and stored



Array

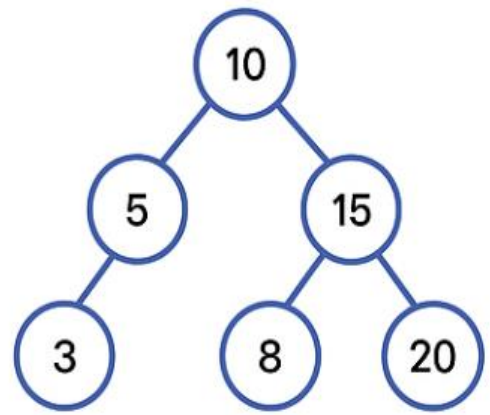
The Relationship Between Data Elements

How data items are logically connected

Linked list

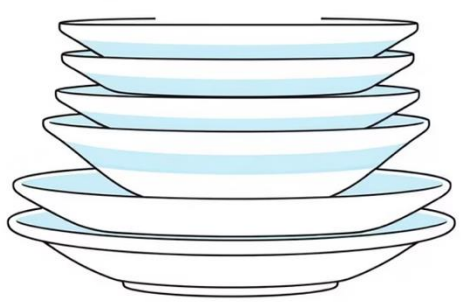
The Operations That Can Be Performed on t Data

Functions or actions that can be applied



Tree

Examples of DS



Stack for Undo/Redo

In text editors, every action is "pushed" onto a stack. "Undo" "pops" the last action off, resembling a stack of plates.

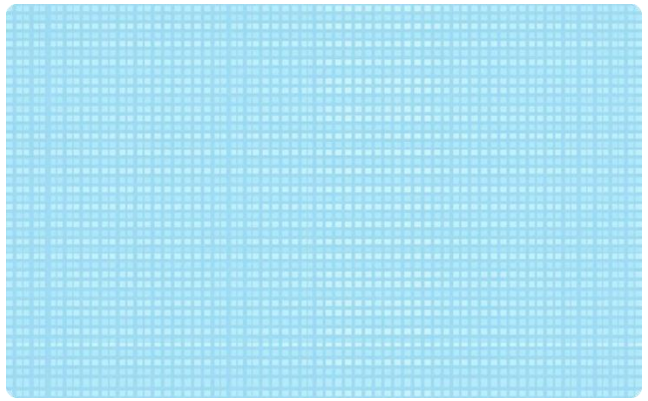


Image Storage (Arrays)

Images are stored as 2D or 3D arrays of pixels. pixels. Each pixel’s color value is an element in element in the array, allowing efficient rendering.

Graphs for Social Networks

Social media connections, road networks, networks, and airline routes are modeled modeled using graphs, where users/locations are nodes and connections connections are edges.

Common DS

Array	A collection of elements of the same data type stored at contiguous memory locations. Elements are accessed using an index. Fixed-size after creation.
Linked List	A sequence of nodes, where each node contains data and a pointer (or link) to the next node in the sequence. Dynamic in size.
Stack	A Last In, First Out (LIFO) data structure where elements are added (pushed) and removed (popped) from the same end, called the "top."
Queue	A First In, First Out (FIFO) data structure where elements are added (enqueued) at one end (rear) and removed (dequeued) from the other end (front).
Tree	A hierarchical data structure consisting of nodes connected by edges. It has a root node and child nodes, forming a parent-child relationship.
Graph	A non-linear data structure consisting of a finite set of vertices (nodes) and a set of edges connecting them, representing relationships.

Data Types

Primitive Data Types

- **Definition:** Basic building blocks directly supported by a programming language.
- **Examples:** int (integers), char (characters), float (floating-point numbers), boolean (true/false).
- They represent fundamental data values and typically have fixed memory sizes.

User-Defined Data Types

- **Definition:** Created by the programmer using combinations of primitive and other user-defined types.
- **Examples:** Classes (in OOP languages), Structs (in C/C++), Enums.
- Enable custom data formats to represent complex real-world entities.

Abstract Data Types (ADT)

- **Definition:** Conceptual models defining a set of data values and the operations that can be performed on them, without specifying how those operations are implemented.
- **Examples:** List, Stack, Queue, Map.
- Focus on "what" an operation does rather than "how" it does it, promoting abstraction.

DS Types

Linear Data Structures

- Elements are arranged sequentially, forming a linear relationship. Each element has a predecessor and a successor, except for the first and last.
- Arrays
- Linked Lists
- Stacks
- Queues

Non-Linear Data Structures

- Elements are not arranged sequentially; instead, they are connected in a hierarchical or networked manner, allowing for more complex relationships.
- Trees
- Graphs

Static vs. Dynamic

- **Static:** Fixed size, memory allocated at compile time (e.g., traditional Arrays).
- **Dynamic:** Size can change at runtime, memory allocated/ deallocated as needed (e.g., Linked Lists, dynamic Arrays).

Importance of DS

- **Efficiency:** Optimized data structures lead to faster data processing and reduced resource consumption.
- **Scalability:** Well-chosen data structures can handle increasing amounts of data without significant performance degradation.
- **Code Organization:** They provide a clear and logical way to organize complex data, making code more readable and maintainable.
- **Problem Solving:** Many real-world problems can be solved efficiently by mapping them to appropriate data structures (e.g., graphs for social networks, hash tables for dictionaries).

Real life applications

- **Operating Systems** → Process scheduling (Queue), memory management (Heap)
- **Databases** → Indexing (B-Tree, Hash Table), query optimization
- **Web Browsers** → Back/Forward navigation (Stack)
- **Networking** → Routing algorithms (Graph), packet queues
- **Search Engines** → Auto-complete (Trie), ranking results (Heap)
- **Social Networks** → Friend suggestions (Graph traversal)
- **E-commerce** → Product catalog (Hash Map, Tree), recommendation engines
- **Game Development** → Scene graphs, pathfinding (Graph, Tree)

Algorithms

- An **algorithm** is a step-by-step, well-defined set of instructions to solve a problem or perform a task.
- An efficient algorithm can help us to find the solution we are looking for, and to transform a slow program into a faster one.
- Examples
  - Finding the fastest route in a GPS navigation system
  - Navigating an airplane or a car (cruise control)
  - Finding what users search for (search engine)
  - Sorting, for example sorting movies by rating

Algorithm: Linear Search in an Array

1. Start
2. Input array A[] and element X to search
3. For i = 0 to length(A) - 1:
If A[i] == X:
Output "Found at position i"
Stop
4. If not found, Output "Not found"
5. End

DSA: Key points

- Data structures **store and organize data**.
- Algorithms **process and manipulate that data**.
- The choice of data structure often determines which algorithm can be used efficiently.
- In programming, Data Structures decide how data is stored; Algorithms decide how data is processed. Both together determine ***performance, reliability, and usability***.
- *Data Structure is like a **container**; Algorithm is the **recipe** to process what's inside.*

Table 1: Student List

Name	Roll	Bangla	English	Physics	Chemistry	Math
Amina Rahman	101	85	78	88	82	90
Rafiq Hasan	102	72	69	80	75	85
Tahmid Islam	103	90	88	92	85	95
Nabila Akter	104	78	82	75	80	84
Saif Hossain	105	65	70	68	72	75
Farzana Jahan	106	88	85	90	88	92
Mahin Chowdhury	107	81	77	84	80	86
Shorna Khatun	108	75	80	78	76	82
Arif Khan	109	69	65	70	72	74
Jannat Ferdous	110	92	90	95	93	97

DS vs DBMS

- **Data Structures (DS)**
  - A way of organizing and storing **data in memory (Primary Memory)** for efficient access and modification during program execution.
  - Examples: Array, Linked List, Stack, Queue, Tree, Graph, Hash Table.
  - Focuses on **in-memory data organization** for algorithm efficiency.
  - Stored in **primary memory (RAM)**, volatile (lost when program ends).
- **Database Management System (DBMS)**
  - A software system that allows users to store, manage, retrieve, and manipulate **data in databases (Secondary Memory)**.
  - Examples: MySQL, PostgreSQL, Oracle DB, MongoDB.
  - Focuses on **persistent storage** of large datasets.
  - Stored in **secondary storage (HDD/SSD)**, non-volatile (data persists).

Homework 1 (On table 1)

- **Write a C program to find the GPA of all students.**
- Find the maximum marks in each subject.
- Find the average marks in each subject.
- Sort the student list according to their highest GPA/ highest marks.

References

- Books
  - **Introduction to Algorithms** – Cormen
  - **Data Structures** by Seymour Lipschutz
  - **Data Structures using C** by E. Balagurusamy

Thank You