Data Structure
GTU #3130702

Unit 1: Introduction to data structure

Prof. Ankit Koyani
CSE Department
SLTIET, Rajkot

Mahatma Gandhi Charitable Trust Managed

Shri Labhubhai Trivedi Institute of Engineering & Technology

CREATING CREATORS - SLTIET

Topics to be covered

- Data and Information
- Data structure
- It’s Classification
- Real word Application
- Primitive/Non-primitive
- Linear/Non-linear
- Operations on Data structure
- Time complexity & Space Complexity

Let’s check visibility level:

Data:

- Data is collection of information.
- Data is a raw and unorganized fact that is required to be processed to make it meaningful.
- It can be considered as facts and statistics collected together for reference or analysis.
- Data doesn’t rely on Information.
- Bits and Bytes are the measuring unit of data.
- Example of data is student test score.

Guess what is information?

Information:

- Information is nothing but processed data.
- Information is defined as classified or organized data that has some meaningful value for the user.
- Processed data must meet the following criteria for it to be of any significant use in decision-making:
- Accuracy: The information must be accurate.
- Completeness: The information must be complete.
- Timeliness: The information must be available when it’s needed.
- Example of information is average score of class that is derived from given data.

Data structure

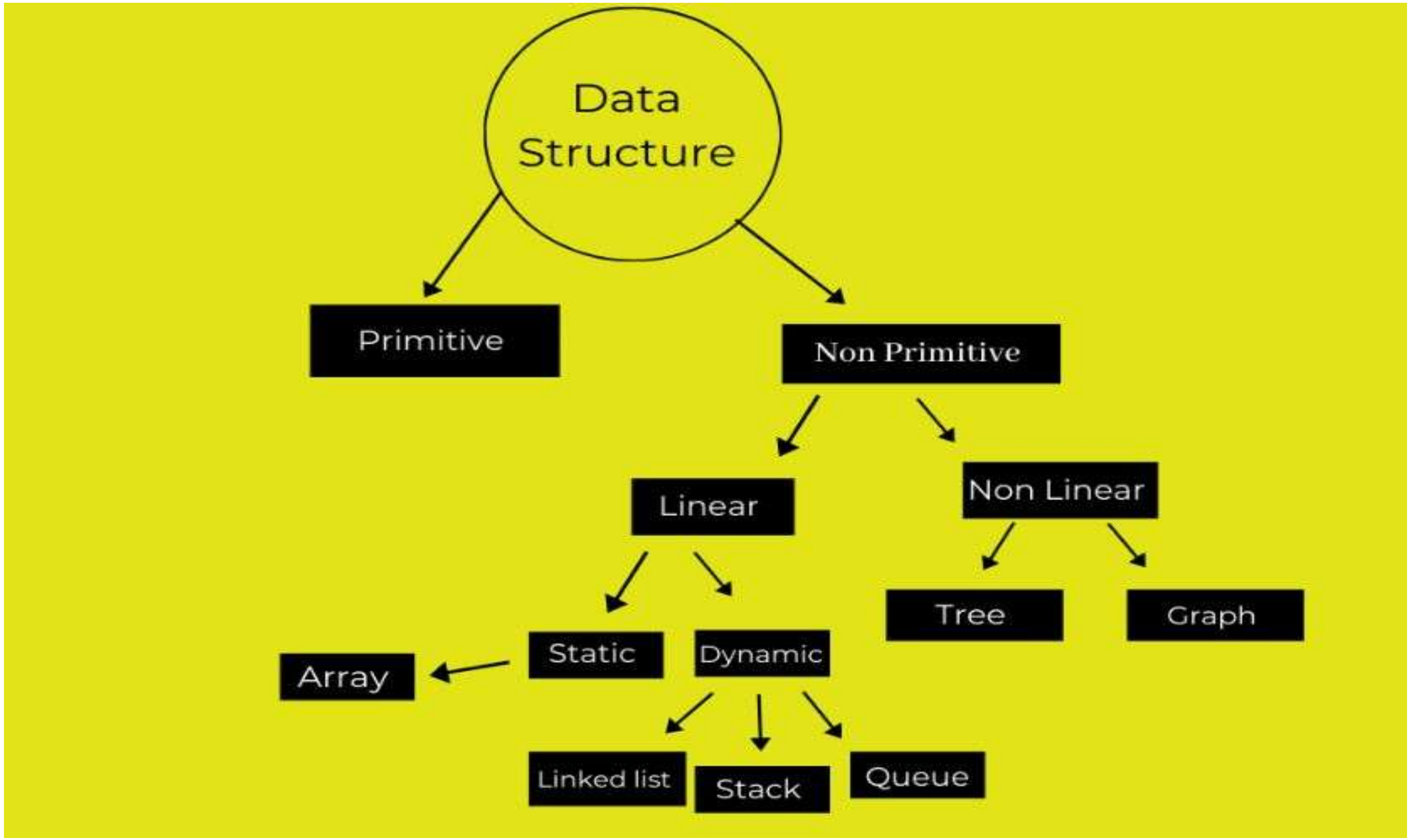
- Data is information, and algorithms are rules and instructions that turn the data into something useful to programming.

- Put another way, remember these two simple equations:
- **1) Related data + Permissible operations on the data = Data Structures**
- **2) Data structures + Algorithms = Programs**
- **https://www.youtube.com/watch?v=d_XvFOkQz5k**

Data structure

- A data structure is a specialized format for organizing, processing, retrieving and storing data.
- There are several basic and advanced types of data structures, all designed to arrange data to suit a specific purpose.
- Every application, piece of software, or programs foundation consists of two components: algorithms and data.

Data structure classification



Primitive Data structure

- Primitive data structures, also known as basic data structures, are the building blocks of any program.
- They are used to represent simple data types such as numbers, characters, and Boolean values and are built into the programming language itself.
- **Integer:** Used to store whole numbers, such as 1, 2, 3, etc.
- **Floating-point:** a storage unit for decimal numbers like 1.5, 2.3, etc. sometimes produce unexpected results.

Primitive Data structure

- **Character:** Used to store a single character, such as 'a', 'b', 'c', etc. Characters are represented using a specific encoding, such as ASCII or Unicode, which assigns each character a unique numerical value.
- **Boolean:** Used to store true/false values, such as true or false. Boolean variables are often used to represent logical conditions in programs, such as whether a statement is true or false.

Time and Space complexity

- Time complexity is defined in terms of how many times it takes to run a given algorithm, based on the length of the input.
- Time complexity is a type of computational complexity that describes the time required to execute an algorithm.
- The time complexity of an algorithm is the amount of time it takes for each statement to complete
- The **amount of time** required to execute an algorithm is called time complexity.

Time and Space complexity

- When an algorithm is run on a computer, it necessitates a certain amount of memory space.
- The amount of memory used by a program to execute it is represented by its space complexity.
- Because a program requires memory to store input data and temporal values while running, the space complexity is auxiliary and input space.
- The **amount of space** required to execute an algorithm is called space complexity.

Operations on Data structure

- Data structure operations are the methods used to manipulate the data in a data structure. The most common data structure operations are:
- **Traversal**
  Traversal operations are used to visit each node in a data structure in a specific order. This technique is typically employed for printing, searching, displaying, and reading the data stored in a data structure.
- **Insertion**
  Insertion operations add new data elements to a data structure. You can do this at the data structure's beginning, middle, or end.

Operations on Data structure

- **Deletion**
  Deletion operations remove data elements from a data structure. These operations are typically performed on nodes that are no longer needed.
- **Search**
  Search operations are used to find a specific data element in a data structure. These operations typically employ a compare function to determine if two data elements are equal.
- **Sort**
  Sort operations are used to arrange the data elements in a data structure in a specific order. This can be done using various sorting algorithms, such as insertion sort, bubble sort, merge sort, and quick sort.

Operations on Data structure

- **Merge**
  Merge operations are used to combine two data structures into one. This operation is typically used when two data structures need to be combined into a single structure.
- **Copy**
  Copy operations are used to create a duplicate of a data structure. This can be done by copying each element in the original data structure to the new one.

Worst, Average and Best Case Analysis

Best case (Omega Notation)

- It defines the **best case** of an algorithm's time complexity,
- the Omega notation defines whether the set of functions will grow faster or at the same rate as the expression.
- Furthermore, it explains the minimum amount of time an algorithm requires to consider all input values.
- So time complexity in the best case would be $\Omega(1)$

Worst, Average and Best Case Analysis

Average case (Theta Notation)

- It defines the **average case** of an algorithm’s time complexity,
- the Theta notation defines when the set of functions lies in both **O(expression)** and **Omega(expression)**, then Theta notation is used.
- This is how we define a time complexity average case for an algorithm.

Worst, Average and Best Case Analysis

Worst case (Big-O Notation)

- We define an algorithm’s **worst-case** time complexity by using the Big-O notation,
- which determines the set of functions grows slower than or at the same rate as the expression.
- Furthermore, it explains the maximum amount of time an algorithm requires to consider all input values.

Real-word applications of DS:

- Ticket booking by array
- Google map by graph
- Browsing by Stack
- Token counter by queue
- Music player by linked list

ANY Questions?

thank you

CREATING CREATORS - SLTIET