DATA STRUCTURES AND ALGORITHMS

- LECTURE 1-

(2ND YEAR OF STUDY)

Skopje, 2023/24

Contents

1.1. Introduction

- **Data Structure and Algorithms (DSA)** is defined as a combination of two *separate* yet *interrelated* topics – Data Structure, and Algorithms.
- DSA is one of the most important skills that every computer science student must have.
- It is often seen that people with good knowledge of these technologies are better programmers than others and thus, crack the interviews of almost every tech giant.

1.1. Introduction

- Data structures and Algorithms are dependent on each other. And it is also clear that data structure stores the unstructured data in an organized form, whereas algorithms are the set of instructions that a computer follows to solve a particular task.
- *Data structures* are the *building blocks* of *Algorithms*, and *Algorithms* are the *platforms* upon which *Data Structures* are applied and tested.

1.1. Introduction

- A **Data Structure** is a storage that is used to store and organize data. It is a way of arranging data on a computer so that it can be accessed and updated efficiently.
- A data structure is not *only* used for organizing the data. It is also used for ***processing***, ***retrieving***, and ***storing*** data. There are different basic and advanced types of data structures that are used in almost every program or software system that has been developed.
- So, we must have *good knowledge* about data structures.

1.1. Introduction

- How **Data Structure** varies from **Data Type:**

Data Type	Data Structure
The data type is the form of a variable to which a value can be assigned. It defines that the particular variable will assign the values of the given data type only.	Data structure is a collection of different kinds of data. That entire data can be represented using an object and can be used throughout the program.
It can hold value but not data. Therefore, it is dataless.	It can hold multiple types of data within a single object.
The implementation of a data type is known as abstract implementation.	Data structure implementation is known as concrete implementation.

1.1. Introduction

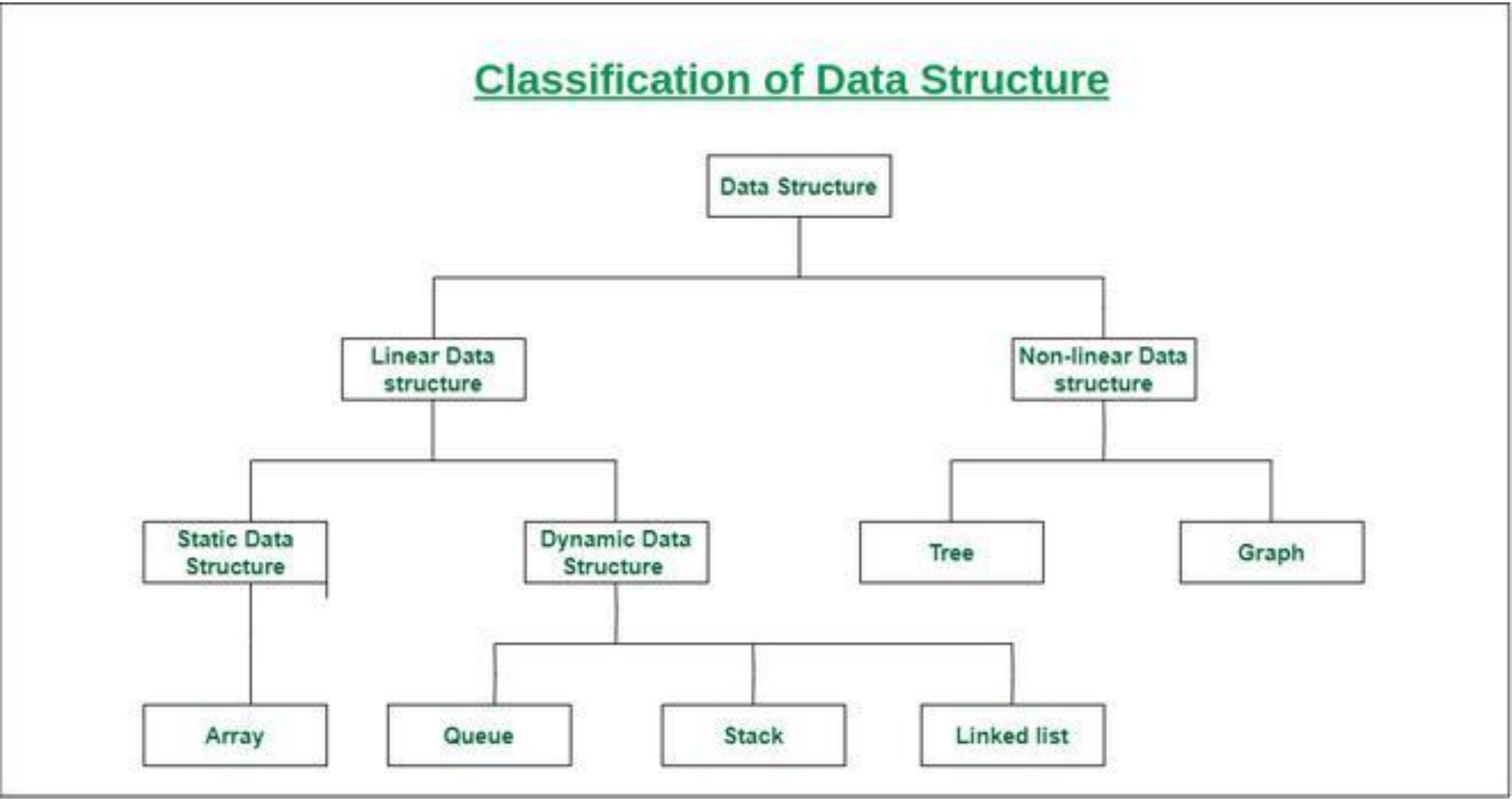
- How **Data Structure** varies from **Data Type:**

There is no time complexity in the case of data types.	In data structure objects, time complexity plays an important role.
In the case of data types, the value of data is not stored because it only represents the type of data that can be stored.	While in the case of data structures, the data and its value acquire the space in the computer's main memory. Also, a data structure can hold different kinds and types of data within one single object.
Data type examples are int, float, double, etc.	Data structure examples are stack, queue, tree, etc.

1.1. Introduction

- Data structure is a way of storing and organizing data efficiently such that the required operations on them can be performed be efficient with respect to time as well as memory.
- Simply, Data Structure are used to reduce complexity (mostly the time complexity) of the code.

1.2. Classification



1.2. Classification

- **Linear data structure:**

  - Data structure in which data elements are arranged sequentially or linearly, where each element is attached to its previous and next adjacent elements, is called a ***linear data structure*.**

- Examples of *linear data structures* are:

✓ *array,*

✓ *stack,*

✓ *queue,*

✓ *linked list.*

1.2. Classification

- **Static data structure:** Static data structure has a *fixed memory size*. It is easier to access the elements in a static data structure. An example of this data structure is an ***array***.
- **Dynamic data structure:** In dynamic data structure, the *size is not fixed*. It can be randomly updated during the runtime which may be considered efficient concerning the memory (space) complexity of the code. Examples of this data structure are ***queue***, ***stack***, ***linked list***.

1.2. Classification

- ***Static data structures***, such as arrays, have a fixed size and are allocated at **compile-time**.
- This means that their memory size cannot be changed during program execution.
- Index-based access to elements is fast and efficient since the address of the element is known.

1.2. Classification

- ***Dynamic data structures***, on the other hand, have a variable size and are allocated at **run-time**.
- This means that their memory size can be changed during program execution. Memory can be dynamically allocated or de-allocated during program execution.
- Due to this dynamic nature, accessing elements based on index may be slower as it may require memory allocation and de-allocation.

1.2. Classification

- **Non-linear data structure:**

  - Data structures where data elements are not placed sequentially or linearly are called ***non-linear data structures***.

- In a non-linear data structure, we can't traverse all the elements in a single run only.
- Examples of *non-linear data structures* are:

✓ *trees,*

✓ *graphs.*

1.2. Classification

Aspect	Static Data Structure	Dynamic Data Structure
Memory allocation	Memory is allocated at compile-time	Memory is allocated at run-time
Size	Size is fixed and cannot be modified	Size can be modified during runtime
Memory utilization	Memory utilization may be inefficient	Memory utilization is efficient as memory can be reused
Access	Access time is faster as it is fixed	Access time may be slower due to indexing and pointer usage
Examples	Arrays, Stacks, Queues, Trees (with fixed size)	Lists, Trees (with variable size), Hash tables

1.2. Classification

- **Advantage of Static data structure:**

- Fast access time
- Predictable memory usage
- Ease of implementation and optimization
- Efficient memory management
- Simplified code
- Reduced overhead

1.2. Classification

- **Advantage of Dynamic data structure:**
  - Flexibility
  - Reduced memory waste
  - Improved performance for some operations
  - Simplified code
  - Scalability

1.2. Classification

- **Need Of Data structure:** The structure of the data and the synthesis of the algorithm are relative to each other.
- Here is a list of the needs for data structure.

  - Data structure modification is easy.
  - It requires less time.
  - Save storage memory space.
  - Data representation is easy.
  - Easy access to the large database.

1.3. Operations

- Data Structure is the way of storing data in computer's memory so that it can be used easily and efficiently.
- There are different data-structures used for the storage of data. It can also be defined as a mathematical or logical model of a particular organization of data items.
- The representation of particular data structure in the main memory of a computer is called as ***storage structure***.

1.3. Operations

- **Operations on different Data Structure:** There are different types of operations that can be performed for the manipulation of data in every data structure.
- Some operations are illustrated below:

➢ **Traversing:** Traversing a Data Structure means to visit the element stored in it. It visits data in a systematic manner. This can be done with any type of DS.

1.3. Operations

- **Insertion:** It is the operation which we apply on all the data-structures. Insertion means to add an element in the given data structure.
- **Deletion:** It is the operation which we apply on all the data-structures. Deletion means to delete an element in the given data structure.
- **Create:** It reserves memory for program elements by declaring them. The creation of data structure can be done during: Compile-time, or Run-time.

1.3. Operations

- **Selection:** It selects specific data from present data. You can select any specific data by giving condition in loop.
- **Update:** It updates the data in the data structure. You can also update any specific data by giving some condition in loop, like select approach.
- **Merge:** Merging data of two different orders in a specific order may ascend or descend. We use merge sort to merge sort data.

1.3. Operations

- **Split Data:** Dividing data into different sub-parts to make the process complete in less time.
- **Sort:** Sorting data in a particular order (ascending or descending). We can take the help of many sorting algorithms to sort data in less time. There are many algorithms present like: insertion sort, selection sort, bubble sort, merge sort, quick sort, etc.