Data Structures And Algorithms

ICT2113

Lecture 1

Level II – Semester 1 - 2020

Faculty of Technology, University of Ruhuna

WHAT IS DATA?

- Data
  - A collection of facts from which conclusion may be drawn
  - e.g. Data: Temperature 38°C; Conclusion: It is hot.
- Types of data
  - Textual: For example, your name (Amal)
  - Numeric: For example, your ID (TG/2017/0001)
  - Audio: For example, your voice
  - Video: For example, your voice and picture
  - (…)

DEFINITIONS:

- **ALGORITHM:**
  - Step-by-step procedure which can be applied to data to achieve some goal.

- **PROGRAM:**
  - Implements an algorithm.

- **DATA STRUCTURE:**
  - The manner in which data is represented in the computer to facilitate its access and manipulation by an algorithm.

    Or

    - **Organization** of data needed to solve the problem

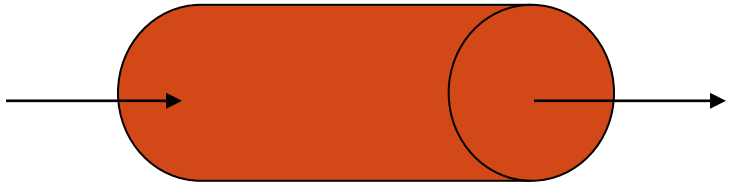
DATA STRUCTURE

- A particular way of storing and organizing data in a computer so that it can be used efficiently and effectively.
- Data structure is the logical or mathematical model of a particular organization of data.
- A group of data elements grouped together under one name.
  - For example, an array of integers

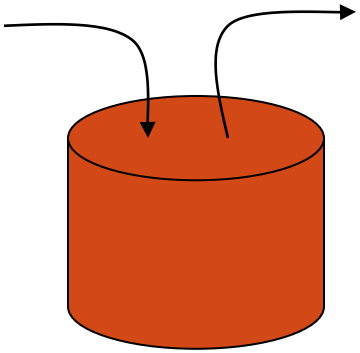
Types of data structures

Array

Linked List

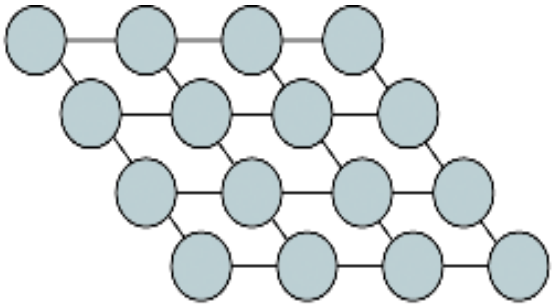


Queue

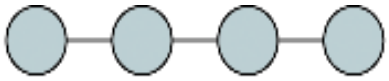


Stack

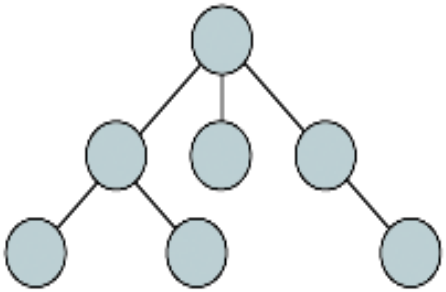
Types of data structures



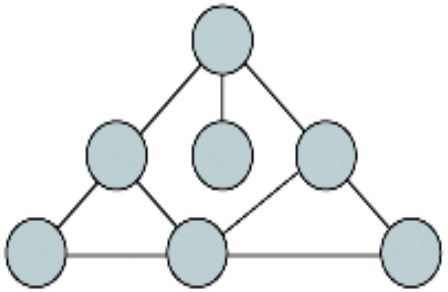
(a) Matrix



(b) Linear list



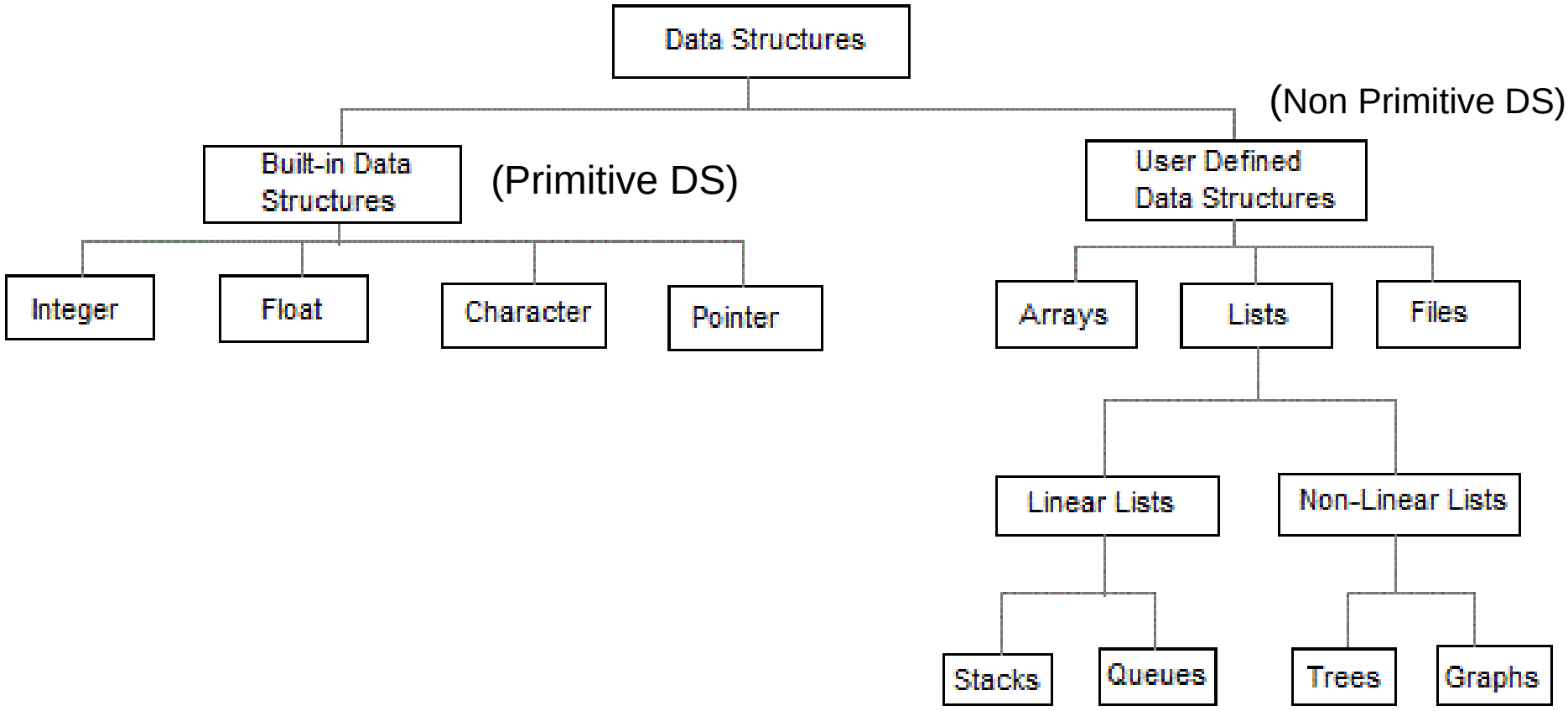
(c) Tree



(d) Graph

There are many, but we named a few. We'll learn these data structures in detail!

Data Structures hierarchy



Basis characteristics of the data structures

Characteristic	Description
Linear	In Linear data structures, the data items are arranged in a linear sequence. Example: **Array**
Non-Linear	In Non-Linear data structures, the data items are not in sequence. Example: **Tree**, **Graph**
Homogeneous	In homogeneous data structures, all the elements are of same type. Example: **Array**
Non-Homogeneous	In Non-Homogeneous data structure, the elements may or may not be of the same type. Example: **Structures**
Static	Static data structures are those whose sizes and structures associated memory locations are fixed, at compile time. Example: **Array**
Dynamic	Dynamic structures are those which expands or shrinks depending upon the program need and its execution. Also, their associated memory locations changes. Example: **Linked List created using pointers**

THE NEED FOR DATA STRUCTURES

- Goal: to **organize data**

- Criteria: to facilitate **efficient**
  - **storage** of data
  - **retrieval** of data
  - **manipulation** of data

- Design Issue:
  - **select and design** appropriate data types (This is the main motivation to learn and understand data structures)

DATA STRUCTURE OPERATIONS

- Traversing
  - Accessing each data element exactly once so that certain items in the data may be processed
- Searching
  - Finding the location of the data element (key) in the structure
- Insertion
  - Adding a new data element to the structure

DATA STRUCTURE OPERATIONS (CONT.)

- Deletion
  - Removing a data element from the structure
- Sorting
  - Arrange the data elements in a logical order (ascending/descending)
- Merging
  - Combining data elements from two or more data structures into one

WHAT ARE DATA STRUCTURES AND ALGORITHMS GOOD FOR?

- Real-world data storage.
  - To keep the details of a set of people.
- Programmer's tools
  - The data structures that are meant for the access of the program itself.
    - Stacks, queues,...
- Real-world modelling
  - To model real world situations.
    - graphs, queues,…

OVERALL PICTURE

Data Structure and Algorithm Design Goals

Correctness

Efficiency

Implementation Goals

Robustness

Adaptability

Reusability

Data Structure	**Advantages**	**Disadvantages**
Array	Quick insertion, very fast access if index known.	Slow search, slow deletion, fixed size.
Ordered array	Quicker search than unsorted array.	Slow insertion and deletion, fixed size.
Stack	Provides last-in, first-out access.	Slow access to other items.
Queue	Provides first-in, first-out access.	Slow access to other items.
Linked List	Quick insertion, quick deletion.	Slow search.
Binary Tree	Quick search, insertion, deletion (if tree remains balanced)	Deletion algorithm is complex.

DATA STRUCTURES VS. ALGORITHMS

- Data Structures

Represent objects of the Abstract data type

- Algorithms

Manipulate the data structures to implement the operations of the ADT

Data structures and algorithms are patterns for solving problems

WHY WE NEED DATA STRUCTURES?

- Allow us to achieve an important object-oriented programming goal: component reuse.
- Once each data structure has been implemented, it can be used over and over again in various applications.
- Data structure is a particular way of storing and organizing information in a computer, so that it can be retrieved and used most productively.

WHY STUDY DATA STRUCTURES AND ALGORITHMS?

Programs are comprised of two things:

data and **algorithms. …**

The reason for learning about **data structures** is because adding **structure** to our **data** can make the algorithms much simpler, easier to maintain, and often faster.

END