CM1602 : Data Structures and Algorithms for AI

Overview

ROBERT GORDON UNIVERSITY ABERDEEN
TEF Gold
INFORMATICS INSTITUTE OF TECHNOLOGY

MODULE LEARNING OUTCOMES (LO)

On completion of this module, students are expected to be able to:

- LO1 : Describe the fundamental concepts of algorithms and data structures.
- LO2 : Evaluate algorithms and data structures using the theory of complexity analysis (for performance).
- LO3 : Apply appropriate data structures given a real-world problem to meet requirements of programming language APIs.
- LO4 : Adapt and extend algorithms to real-world problems and address implementation requirements.

MODULE DETAILS

- Full Module Title: Data Structures and Algorithms for AI
- Module Code : CM1602
- Length : 1 Semester
- Lecturers

Module Leader

- Ragu Sivaraman (email: ragu.s@iit.ac.lk)

ASSESSMENT PLAN

- Individual Coursework (LO3 & LO4) – 40%
- Closed Book Examination (LO1 & LO2) – 60%
- Qualifying marks for each component is 30% hence the student should achieve minimum 30/100 for each component

MODULE DELIVERY

- This is a one semester Module
- Lectures
  - 2 hours per week
- Tutorial
  - 2 hours per week

MODULE CONTENT

Lecture	Topic
Lecture 01	Introduction to Fundamentals of Algorithms
Lecture 02	Analysis of Algorithms
Lecture 03	Array and Linked Lists
Lecture 04	Stack
Lecture 05	Queue
Lecture 06	Searching algorithms and Sorting algorithms
Lecture 07	Trees
Lecture 08	Maps, Sets, and Lists
Lecture 09	Graph algorithms

ESSENTAIL READING

- Sedgewick, R. (2011). Algorithms in Java, 4th ed. Addison-Wesley Professional.
- Cormen, T. (2009). Introduction to Algorithms, 3rd ed. London: The MIT Press.
- Skiena, S. (2008). The Algorithm Design Manual. 2nd ed. Springer.

CM1602 : Data Structures and Algorithms for AI

1. Introduction to Fundamentals of Algorithms

Lecture 1 | R. Sivaraman

ROBERT GORDON UNIVERSITY ABERDEEN
INFORMATICS INSTITUTE OF TECHNOLOGY

MODULE CONTENT

Lecture	Topic
Lecture 01	Introduction to Fundamentals of Algorithms
Lecture 02	Analysis of Algorithms
Lecture 03	Array and Linked Lists
Lecture 04	Stack
Lecture 05	Queue
Lecture 06	Searching algorithms and Sorting algorithms
Lecture 07	Trees
Lecture 08	Maps, Sets, and Lists
Lecture 09	Graph algorithms

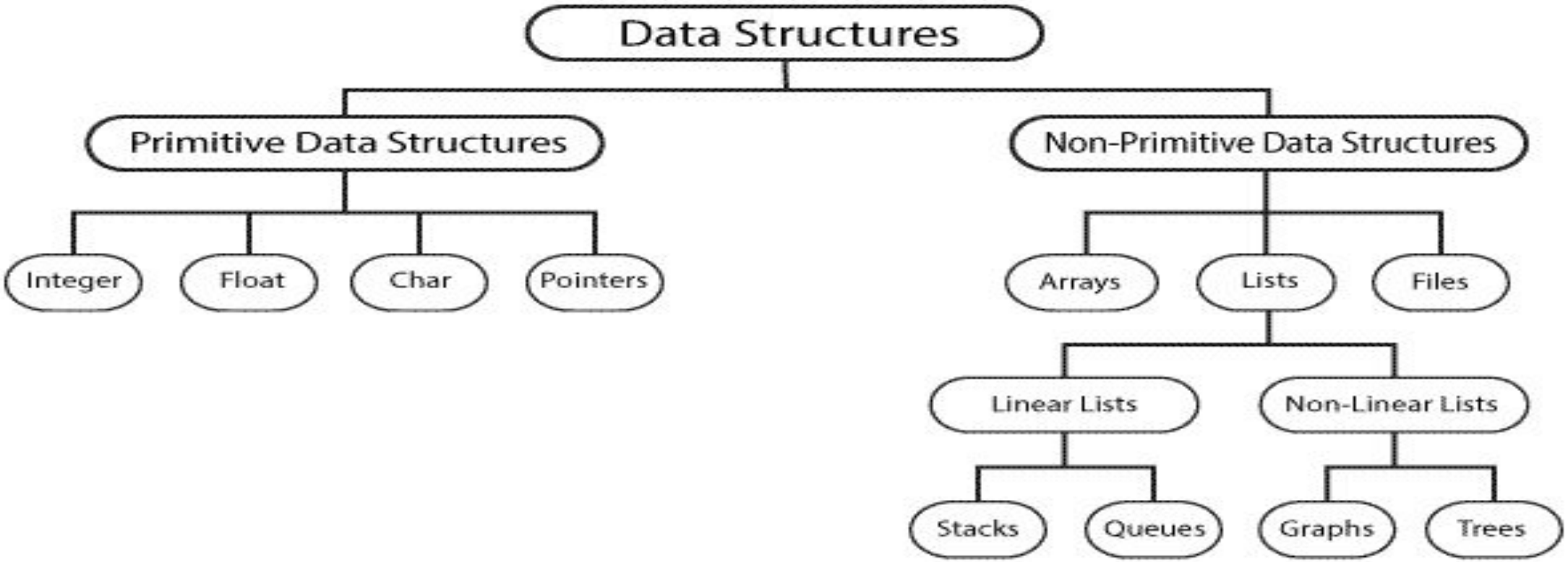
Learning Outcomes

- Covers LO1 (LO1 : Describe the fundamental concepts of algorithms and data structures) for Module
- On completion of this lecture, students are expected to be able to:
  - Define algorithms and the importance of algorithms

Data Structures, Algorithms, and Programs

- **Data structure** – Organization of data to solve the problem at hand
- **Algorithm** – Algorithm is a step-by-step procedure, which defines a set of instructions to be executed in a certain order to get the desired output.
- **Program** – implementation of an algorithm in some programming language

Types of Data Structures



Types of Data Structures

- **Static data structures**(SDS) are fixed sized (e.g. Arrays), the amount of memory once allocated to them cannot change on run time whereas.

- **Dynamic data structures**(DDS)(e.g. Linked Lists) have flexible size , they can grow or shrink as needed to contain the **data** to be stored.

Algorithms

- Algorithm is a step-by-step procedure, which defines a set of instructions to be executed in a certain order to get the desired output.
- Algorithms are generally created independent of underlying languages.

Characteristics of an Algorithm

- **Unambiguous** − Algorithm should be clear and unambiguous. Each of its steps (or phases), and their inputs/outputs should be clear and must lead to only one meaning.
- **Input** − An algorithm should have 0 or more well-defined inputs.
- **Output** − An algorithm should have 1 or more well-defined outputs, and should match the desired output.
- **Finiteness** − Algorithms must terminate after a finite number of steps.
- **Feasibility** − Should be feasible with the available resources.
- **Independent** − An algorithm should have step-by-step directions, which should be independent of any programming code.

Simple Algorithm Example

- Design an algorithm to add two numbers and display the result.

```
Step 1 – START ADD
Step 2 – get values of a & b
Step 3 – c ← a + b
Step 4 – display c
Step 5 – STOP
```

Recursion

- A part of the object refers to the entire object, or
- One part refers to another part and vice versa

Factorial

- A factorial is a function that multiplies a number by every number below it.

- E.g. - 5!= 5*4*3*2*1=120

Factorial Program using Loop

```
num = ...
factorial = 1

if num < 0:
    print("must be positive")
elif num == 0:
    print("factorial = 1")
else:
    for i in range(1,num + 1):
        factorial = factorial*i
    print(num, factorial)
```

Factorial Program using Recursion

```
# Factorial of a number using recursion

def recur_factorial(n):
    if n == 1:
        return n
    else:
        return n*recur_factorial(n-1)
```

Activity

- Why learn Data Structures and Algorithm?
- Recursion vs Iteration