CS301
Data Structure and Algorithms
Lecture 1: Introduction

Pandav Patel
Assistant Professor

Computer Engineering Department
Dharmsinh Desai University
Nadiad, Gujarat, India

Objective

- Course outline
- Understand what is data structure
- Know importance of data structure

Overview

General Information

- Teaching Scheme
  - Lectures: 4 hours per week
  - Practical: 2 hours per week
  - Total: 6 hours per week
- Credits: 5
- Examination Scheme
  - External: 60 marks
  - Sessional: 40 marks
  - Term work: 25 marks
  - Practical: 25 marks
  - Total: 150 marks

Syllabus

- Basic Concepts
- Arrays
- Stacks and Queues
- Linked Lists
- Trees
- Graphs
- Sorting
- Hashing
- Search Techniques

Sessionwise Syllabus

- Session-I
  - Arrays
  - Linked Lists
- Session-II
  - Graphs
- Session-III
  - Sorting
  - Hashing

Reference Books

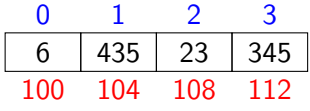
- Data Structures and Algorithms in Java (fourth edition) *by Michael T. Goodrich and Roberto Tamassia*, Publisher: John Wiley & Sons Inc
- Data Structures and Program Design in C (Second Edition) *by Robert L. Kruse and Bruce P. Leung*, Pearson Education
- Data Structures And Algorithms Made Easy In JAVA *by Narasimha Karumanchi*, Publisher: Careermonk Publications
- An Introduction to Data Structures with Applications (Second Edition) *by Tremblay and Sorenson*, McGraw Hill Pulbications

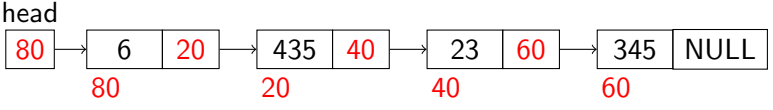
What is data structure?

- Data structure is a way of representing data in computer memory.
- Same data can be represented in different ways in computer memory.
- For example, a list of integers can be stored in computer memory using an array or linked list.
  - Elements of an array are stored in contiguous memory locations while elements of linked list are not stored in contiguous memory locations.

Array Vs Linked List

- Say we want to store list of integers in computer memory.
- 6, 435, 23, 345





Classification of data structures

- Primitive data types
- Non-primitive data structures
  - Linear data structures
    - Arrays
    - Stacks
    - Queues
    - Linked lists
  - Non-linear data structures
    - Trees
    - Graphs

Why learn data structure?

- We can represent same data in different ways in computer memory.
- Time required for storage and retrieval of data depends on its storage structure in memory.
- Data structure for storing given data should be chosen based on access pattern of the algorithm that would operate on the data.
  - For example, if you frequently need to access integer at random index in a list of integers; then storing the list as an array is better than storing it as linked list.
- Thorough understanding of different data structures is very important to be able to choose right data structure for storing given data (after considering an algorithm that will operate on the data).

Arrays

- Elements are stored contiguously in memory
- Memory is wasted due to allocation in advance
- Random access is efficient (e.g. access 100^{th} element)
- Traversing all elements is efficient (e.g. find sum)
- Inserting element in the middle is inefficient

Linked Lists

- Elements are NOT stored contiguously in memory
- Memory is wasted in storing links
- Random access is NOT efficient
- Traversing all elements is efficient (e.g. find sum)
- Inserting element in the middle is efficient (given node address after which to insert)

Arrays

- If memory is available, inserting element at the end is efficient
- preferred when number of elements is known in advance and elements are NOT inserted frequently in the middle

Linked Lists

- Inserting an element at the end is efficient if tail pointer is maintained
- preferred when number of elements is NOT known in advance and elements are inserted frequently in the middle

Problem I

What data structure would you prefer?

- Input
  - First line contains number of elements (n).
  - Second line contains n positive integers separated by space.
- Output
  - Sum of all the elements
- Example

Input	Output
5 32 4 22 5 42	105

Problem II

What data structure would you prefer?

- Input
  - First line contains positive integers separated by space and -1 represents end of list.
- Output
  - Sum of all the elements
- Example

Input	Output
32 4 22 5 42 -1	105

Problem III

What data structure would you prefer?

- Input
  - First line contains number of elements (n).
  - Second line contains n positive integers separated by space.
- Output
  - Mean, median and mode separated by space
- Example

Input	Output
5 2 2 3 2 6	3 2 2

Problem IV

What data structure would you prefer?

- Input
  - First line contains positive integers separated by space and -1 represents end of list.
- Output
  - Mean, median and mode separated by space
- Example

Input	Output
2 2 3 2 6 -1	3 2 2

Problem V

What data structure would you prefer?

- Input
  - First line contains positive integers separated by space and -1 represents end of list.
  - Second line contains non-negative integers separated by space and end is marked by -1.
  - NOTE: numbers in second line are in range (0 to $number\_of\_element\_in\_first\_line - 1$).
- Output
  - Values (as represented in first line in input) present at respective index in the second line of the input
- Example

Input	Output
11 14 12 34 54 23 -1 2 3 1 5 0 -1	12 34 14 23 11

Problem VI

What data structure would you prefer?

- Input
  - First line contains positive integers in ascending order separated by space and -1 represents end of list.
  - Second line contains positive integers separated by space in ascending order. End is marked by -1.
- Output
  - Insert all elements of the second line to the elements in the first line. Keep list sorted throughout the execution. Print elements (separated by space) from the resultant list after each step.
- Example

Input	Output
11 14 22 34 54 63 -1	5 11 14 22 34 54 63
5 12 35 -1	5 11 12 14 22 34 54 63
	5 11 12 14 22 34 35 54 63

Problem Solutions

Preferred Data Structures

- Problem I: No data structure is required
- Problem II: No data structure is required
- Problem III: Array
- Problem IV: Linked list or Array
- Problem V: Array
- Problem VI: Linked list