Data Structures CS-204

Lecture 1,2

Course Information

- **Textbooks**
  - Introduction to Data Structures in C
    by Ashok N. Kamthane
  - Data Structures and Algorithms
    by A. V. Aho, J. E. Hopcroft, J. D. Ullman
  - Data Structures Using C and C++
    by Y. Langsam, M. J. Augenstein, A. M. Tenenbaum
  - Algorithms in C++
    by Robert Sedgewick

Course Outline

- Introduction to Data Structure
- Recursion
- Stacks
- Queues
- Lists and linked lists
- Trees
- Sorting
- Searching
- Graphs
- Hashing

Grading

- Theory
  - Quizzes ---------------5%
  - Assignments---------10%
  - Mid Term-------------- 25%
  - Final-------------------- 50%
  - Project------ 10%

Introduction to Data Structure and Abstract Data Types

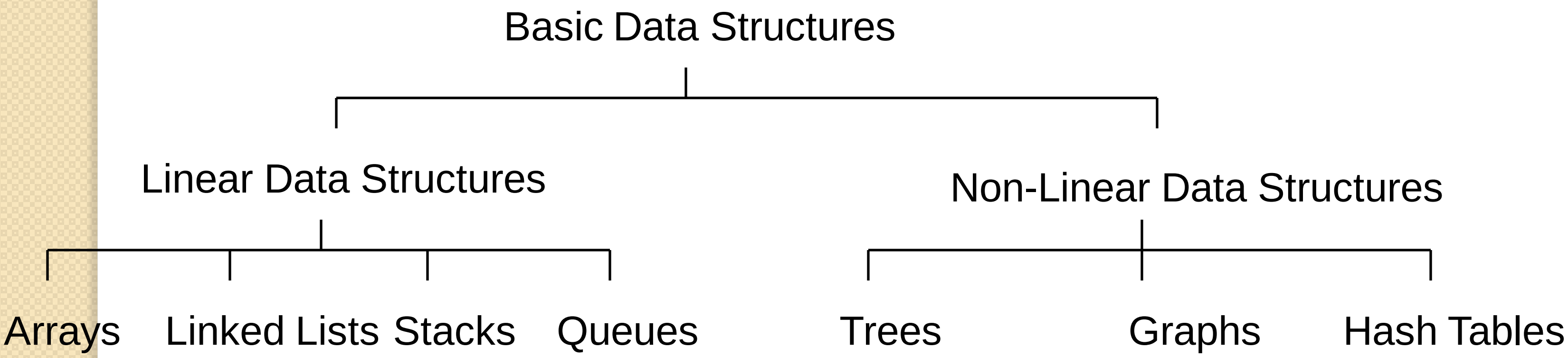
What is Data Structure?

- Data structure is a representation of data and the operations allowed on that data.

- A data structure is a way to store and organize data in order to facilitate the access and modifications.

- Data Structure are the method of representing of logical relationships between individual data elements related to the solution of a given problem.

Basic Data Structure



Basic Data Structures

Linear Data Structures

Non-Linear Data Structures

Arrays

Linked Lists

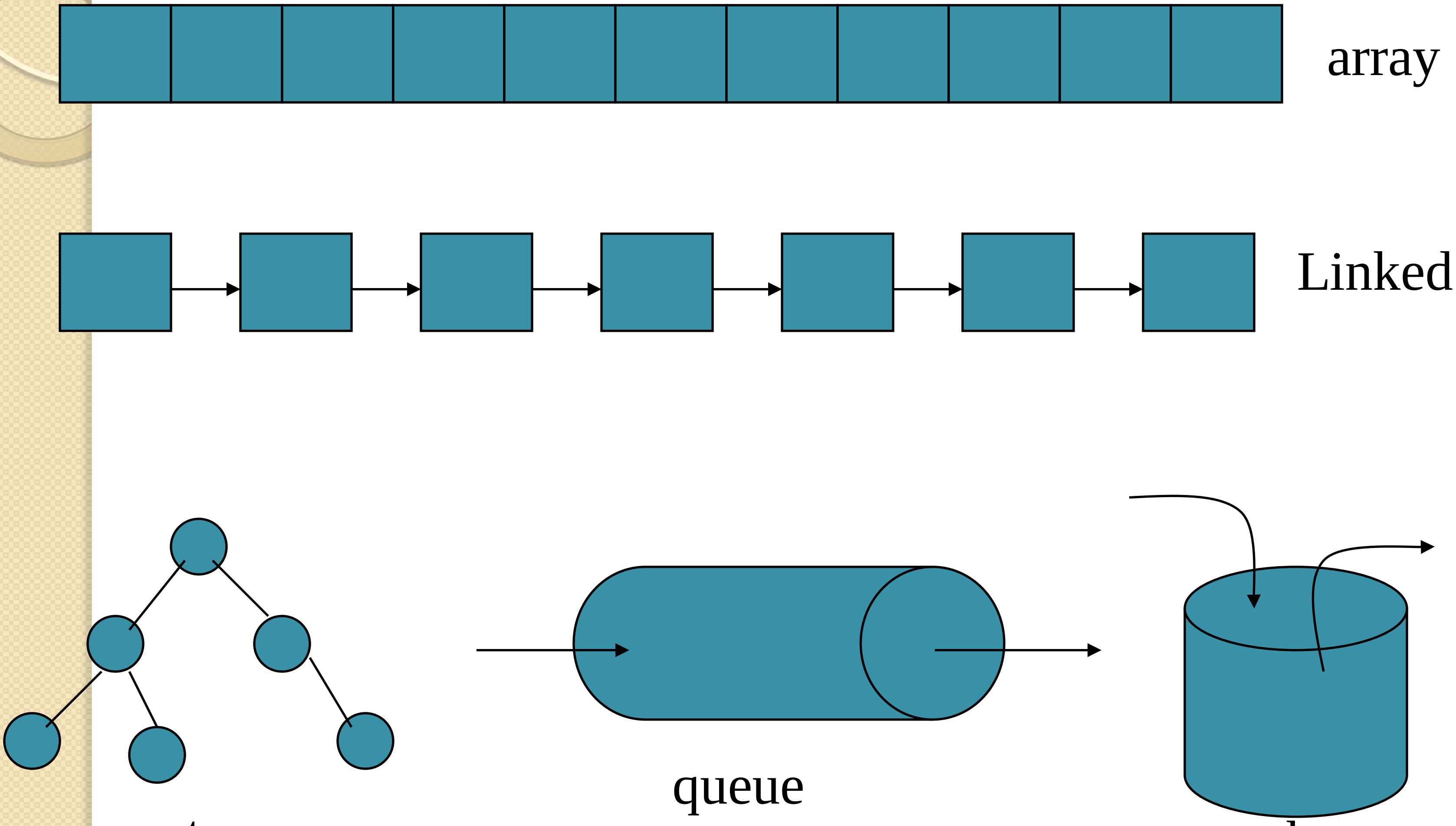
Stacks

Queues

Trees

Graphs

Hash Tables



array

Linked list

tree

queue

stack

Selection of Data Structure

- The choice of particular data model depends on two consideration:
  - It must be rich enough in structure to represent the relationship between data elements
  - The structure should be simple enough that one can effectively process the data when necessary

Types of Data Structure

- Linear: In Linear data structure, values are arrange in linear fashion.
  - Array: Fixed-size
  - Linked-list: Variable-size
  - Stack: Add to top and remove from top
  - Queue: Add to back and remove from front
  - Priority queue: Add anywhere, remove the highest priority

Types of Data Structure

- Non-Linear: The data values in this structure are not arranged in order.
  - Hash tables: Unordered lists which use a ‘hash function’ to insert and search
  - Tree: Data is organized in branches.
  - Graph: A more general branching structure, with less strict connection conditions than for a tree

Type of Data Structures

- Homogenous: In this type of data structures, values of the same types of data are stored.
  - Array
- Non-Homogenous: In this type of data structures, data values of different types are grouped and stored.
  - Structures
  - Classes

Abstract Data Type and Data Structure

- Definition:-
  - *Abstract Data Types (ADTs)* stores data and allow various operations on the data to access and change it.
  - A mathematical model, together with various operations defined on the model
  - An ADT is a collection of data and associated operations for manipulating that data

- Data Structures
  - Physical implementation of an ADT
  - data structures used in implementations are provided in a language *(primitive* or *built-in)* or are built from the language constructs *(user-defined)*
  - Each operation associated with the ADT is implemented by one or more subroutines in the implementation

Abstract Data Type

- ADTs support *abstraction*, *encapsulation*, and *information hiding*.

- *Abstraction* is the structuring of a problem into well-defined entities by defining their data and operations.

- The principle of hiding the used data structure and to only provide a well-defined interface is known as *encapsulation.*

The Core Operations of ADT

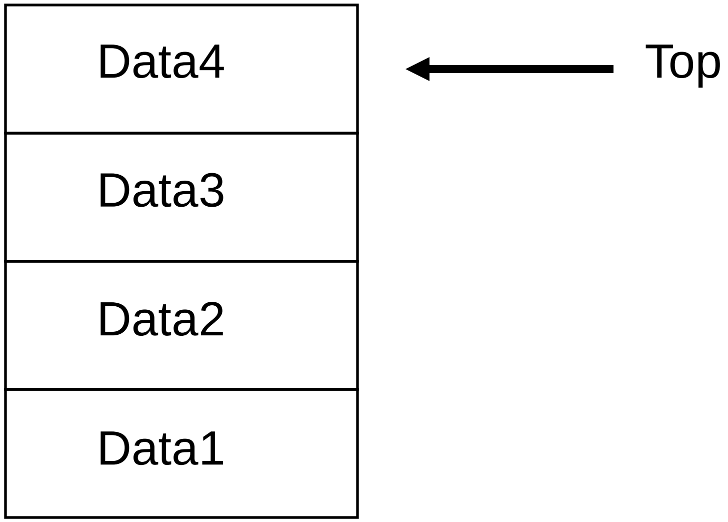
- Every Collection ADT should provide a way to:
  - add an item
  - remove an item
  - find, retrieve, or access an item

- Many, many more possibilities
  - is the collection empty
  - make the collection empty
  - give me a sub set of the collection

- No single data structure works well for all purposes, and so it is important to know the strengths and limitations of several of them

Stacks

- Collection with access only to the last element inserted
- Last in first out
- insert/push
- remove/pop
- top
- make empty



Data4

Data3

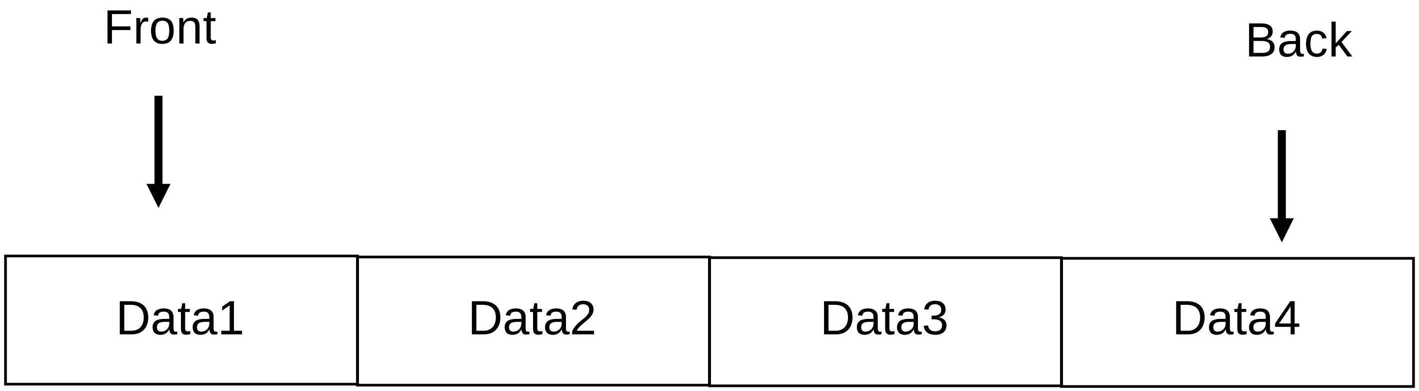
Data2

Data1

Top

Queues

- Collection with access only to the item that has been present the longest
- Last in last out or first in first out
- enqueue, dequeue, front
- priority queues and dequeue



Front

Back

Data1

Data2

Data3

Data4

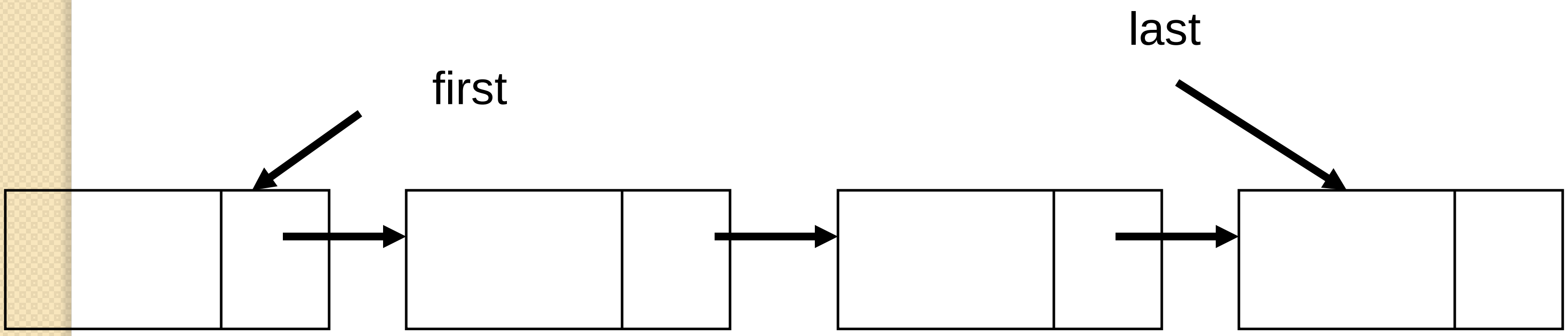
List

- A ***Flexible*** structure, because can grow and shrink on demand.

Elements can be:

- Inserted
- Accessed
- Deleted

At ***any*** position



Tree

- A ***Tree*** is a collection of elements called ***nodes*.**
- One of the node is distinguished as a ***root***, along with a relation (“parenthood”) that places a hierarchical structure on the nodes.