Data Structures and Algorithms

Mr. Tahir Iqbal

tahir.iqbal@bahria.edu.pk

Lecture 1

COURSE OBJECTIVE

- To Understand and to implement numerous examples of relationships between data;
- To purpose and mathematical background of algorithm analysis and be able to apply this to determine the run time and memory usage of algorithms
- To understand and implement abstract data types of stacks, queues and de-queues; Variety of ways that linearly and weakly ordered data can be stored, accessed, and manipulated
- To understand and implement the characteristics and optimal behavior of hash tables for access and retrieval
- To understand and implement various sorting algorithms and the run-time analysis required to determine their efficiencies
- To understand and implement various graph algorithms; Numerous algorithm design techniques including greedy, divide-and-conquer, dynamic programming, randomized algorithms, and backtracking;

COURSE OUTCOMES

On the successful completion of the course, students will be able to:

Understand the principle behind the use of Abstract Data Types (ADT)

Learn the commonly used ADT's and their implementation

Learn basic algorithms and basic methods used in design of algorithms

Learn to compute the cost and benefits of different data structures and algorithms
Learn how to select algorithms and data structures

Course Information

Text Books

- Adam Drozdek, "Data Structure and Algorithms in C++", 4th Edition, Cengage Learning, ISBN-13: 978-1133608424, 2014
- D.S. Malik, "Data Structures using C++", 2nd Edition, Course Technology cengage learning, 2009. ISBN-13: 978-0324782011

Course Information

Reference Books

- D. Samantha., “Classic Data Structures” Latest Edition, PHI learning private limited, New Delhi. 2009, ISBN-978-81-203-3731-2
- Elliot Koffman., “Data Structures: Abstraction and Design Using Java”., John Wiley & Sons, Inc. 2010, (latest Edition), ISBN: 9780470128701
- Thomas H. Cormen. Charles E. Leiserson , Ronald L. Rivest, “Introduction to Algorithms”, 3rd Edition .2009. ISBN- 978-0262033848.
- Sahni , Sartaj., “Data Structures, Algorithms, and Application in C++”, Latest Edition
- Tenenbaum., Aaron M., Langsam., Yedidyah., Augenstein., Mosh., “Data Structures Using C”, Latest Edition, 1989

Course Outline

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

Grading

- Theory
  - Quizzes ----------10%
  - Project/Assignments ---------20%
  - Mid Term---------- 20%
  - Final---------- 50%

Introduction to Data Structure and Abstract Data Types

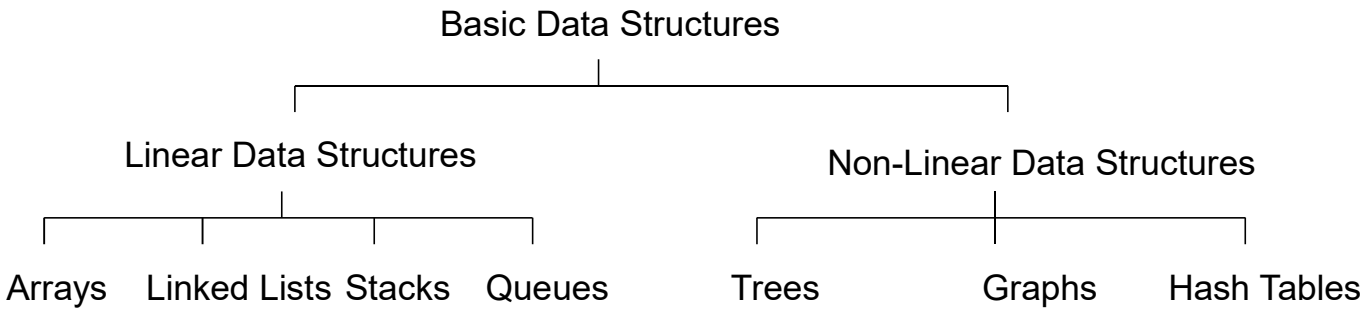
What is Data Structure?

- 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.

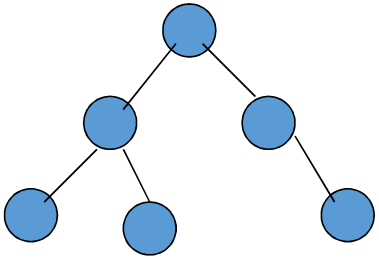
- Data Structure are the method of representing of logical relationships between individual data elements related to the solution of a given problem.

Basic Data Structure

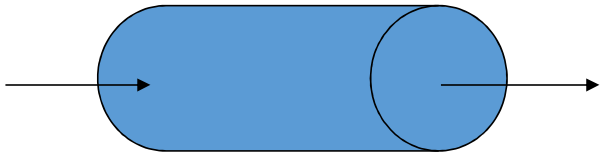


array

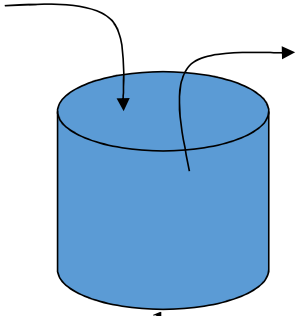
Linked list



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.
  - 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

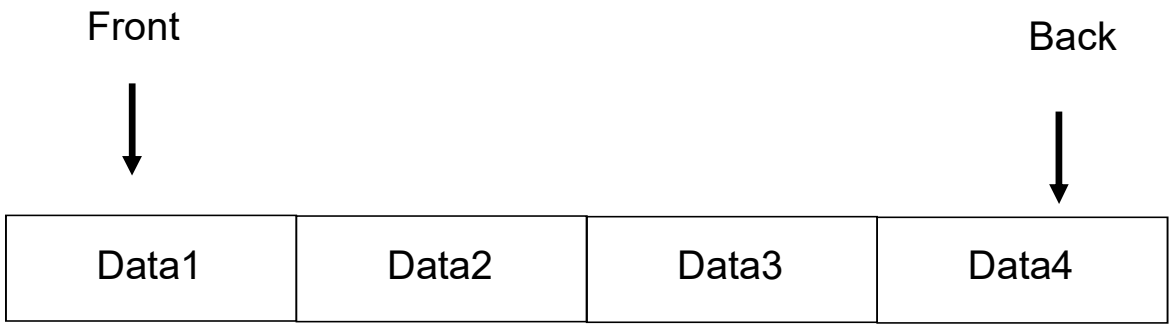
- No single data structure works well for all purposes, and so it is important to know the strengths and limitations of several of them

Stacks

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

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



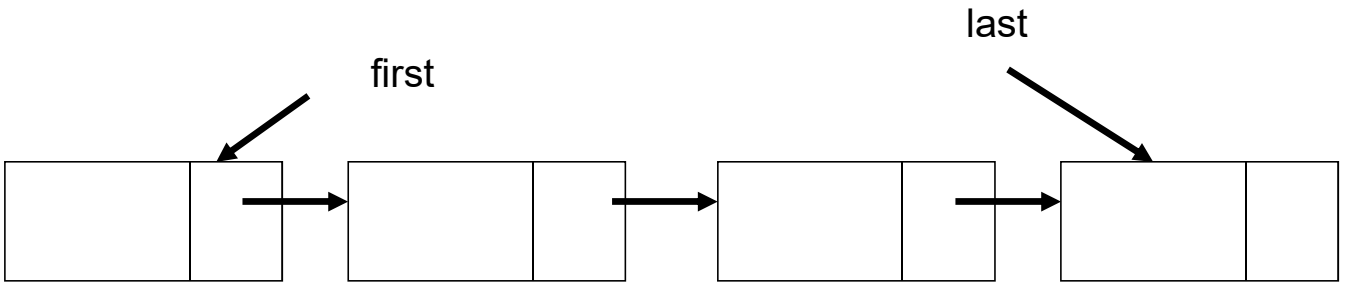
List

- 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.