Data Structures and Algorithm Analysis

1

Dr. Syed Asim Jalal
Department of Computer Science
University of Peshawar

about me

- Syed Asim Jalal
  - BS. University of Peshawar
  - MS. NUST
  - PhD. University of Southampton (UK)
- Past Teaching at:
  - Southampton University
  - University of Peshawar
  - Kohat University of S&T

Syllabus – tentative topics

- Includes the following topics and more..
  - Data Structure Types
  - Array
  - Linked List
  - Stack
  - Queues
  - Analysis of Algorithms
  - Sorting Algorithms and its Analysis
    - selection sort,
    - bubble sort,
    - merge sorting,
    - Quick sort
    - Radix sort

    - Bucket Sort (address-calculation sort)
    - Counting Sort
- Tree
    - Traversal, Insertion, Deletion Algorithms
- Graphs
    - Introduction, Depth First Search, Breadth First Search, Minimum Spanning Trees, etc.
- etc.

Objectives of the course

- Understanding different options available for storing different types of data
  - List/Array, Linked Lists, Trees, Queue, Graph, Hash Table etc.
- Being able to implement different operations on different data structures (Algorithms)
- Understanding when to use a particular data structure based on the ***requirement of a problem*** at hand
  - One data structure is not always better.
- Understanding Analysis of Algorithm and being able to analyse algorithms
- Understanding factors that influence the selection of data structures in some application
  - Memory efficiency
  - Time efficiency

Data Structure

- A Data Structure is a specific format to organize data in **computer memory** so that it can be used to efficiently **store and process** data by computer programs.
- Data Structure is organization of information, usually in memory, for better algorithm efficiency.
- A data structure organizes different data items based on its relationship to each other.

Efficient retrieval and access of data

- Complex applications requires processing of data, which include accessing data and writing new data.
- This data processing need to be efficient in terms of memory space, time or both.
- The choice of how the data is organised can make big difference in efficiency of the complex programs.

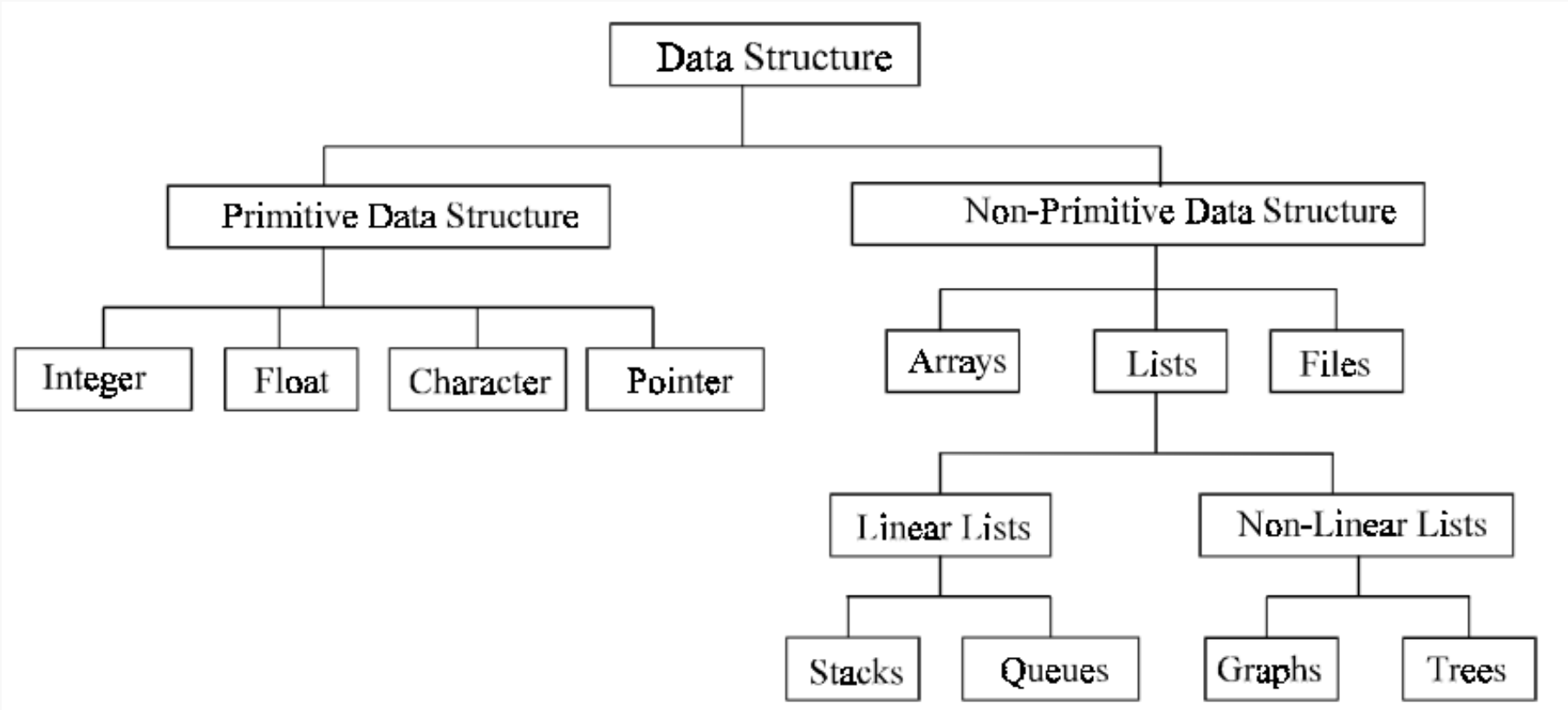
Data Structure vs Files

- Data Structure is often referred to data storage in main memory (RAM)
- Data storage representation in secondary storage is referred to as File Structure.
  - or Databases

- Data may be organized in many ways
  - e.g. Arrays, Linked Lists, Trees, etc.
- The choice of a particular data structure depends on two considerations:
  - It must be able to represent the actual relationships of data in the real world
  - The structure should efficiently process the data when necessary

- Data structure for storing records of data, for example, students names list.
  - Arrays or Linked Lists
- Issues
  - Depends on operations on data.
  - Operations efficiency (Time required to complete operations)
    - Are there going to be only Retrieval or Insertion operation on Data?
    - Are there going to be Deletions?

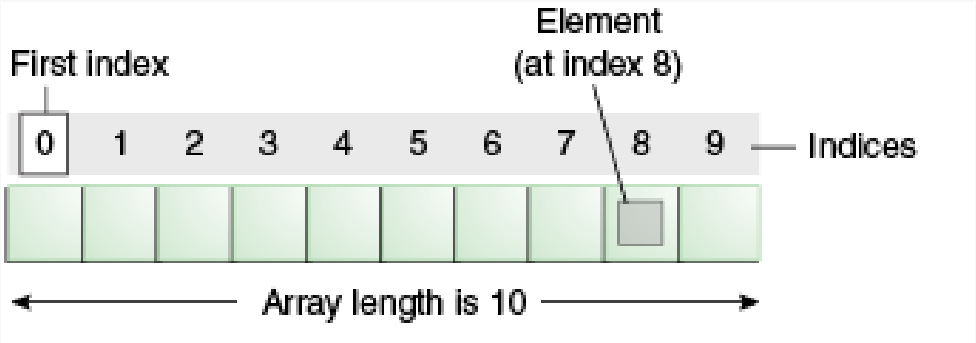
Classification of Data Structures

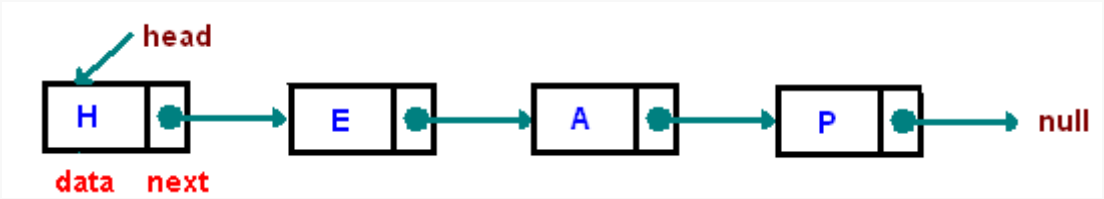


- Primitive data structures
  - These are the basic data structures and are directly operated upon by the machine instructions, which is in a primitive level.
  - They are integers, floating point numbers, characters, string constants, pointers etc.
- Non-primitive data structures
  - They are more sophisticated data structures based on structuring of a group of basic data structures or items.
  - Array, list, files, linked list, trees and graphs

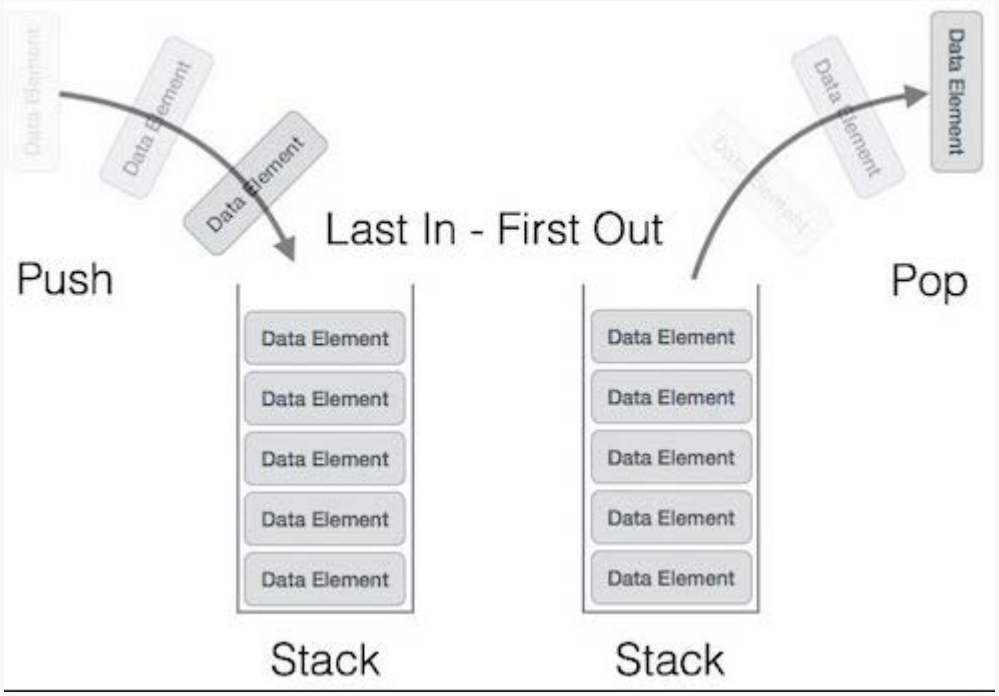
Linear and Non-linear Data Structures

- Linear Data Structures
  - Linear data structures organize their data elements in a linear fashion, where data elements are attached one after another.
  - All linear data structures look like a list.
  - In Linear data structures data elements are traversed one after the other sequentially.
  - Some commonly used linear data structures are arrays, linked lists, stacks and queues.



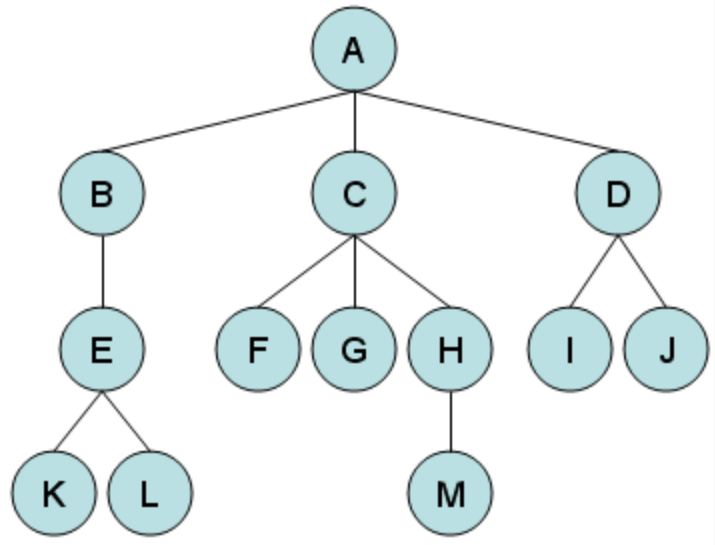


Linked List Data Structure

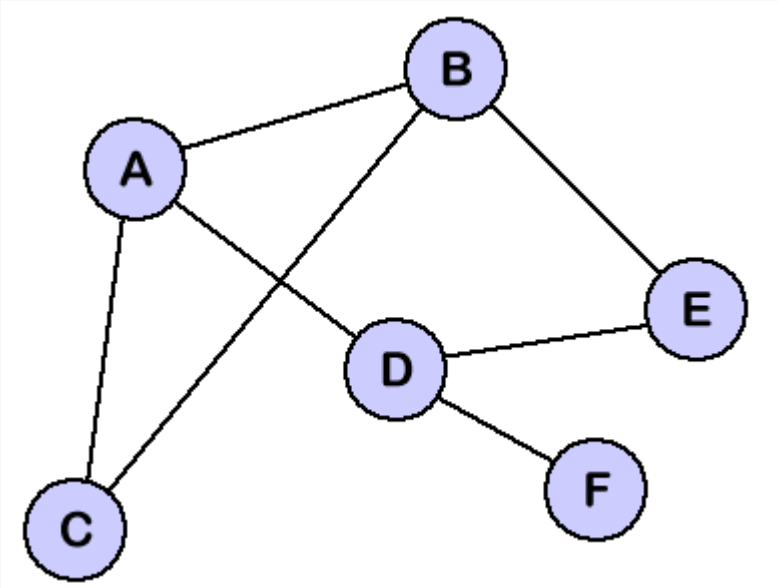


- Non-Linear Data Structures
  - In nonlinear data structures, data elements are not organized in a sequential fashion.
  - A data item in a nonlinear data structure could be attached to several other data elements based on their relationship among.
  - All data items cannot be traversed sequentially, we need to backtrack.
  - Data structures like Trees and Graphs are examples nonlinear data structures.

- A tree is a data structure that is made up of a set of linked nodes, which can be used to represent a hierarchical relationship among data elements.
- A graph is a data structure that is made up of a finite set of edges and vertices. Edges represent connections or relationships among vertices that stores data elements.



Tree Data Structure



Graph Data Structure

Static and Dynamic Data Structures

- Static Data Structures
  - In static data structures the size of the data structure is known in advance and is fixed.
  - Fixed number of data items can be stored in static data structures
  - Advantage:
    - It has no memory overflow problems
  - Disadvantage:
    - Storing less elements than the maximum number will results in memory wastage.
  - For example, int, float, array etc.

- Dynamic Data Structures
  - Dynamic Data Structures have no fixed size and any number of data items can be stored
  - The memory used can grow and shrink based on the number of elements stored.
  - Advantage:
    - Store any number of data elements
    - No wastage of memory
  - Disadvantage:
    - Memory overflow can happen.

Common Operations on Data Structures

- Insertion
  - The operation that add a new data element item in a data structure
- Deletion
  - Deletion operation removes a data elements from a data structure
- Traversing
  - Accessing each data element in a data structure in order to perform some data processing.

- Searching
  - Finding data elements in a structure that contains some value or meets some search criteria or condition.
- Sorting
  - Arranging data elements in some order. For example, ascending or descending order.
  - Sorting data has many useful applications.
- Merging
  - Combining records in two different data structures in to one data structure.

Data Representation

Data Representation

- The basic unit of data representation is a bit – one 1 or 0
  - A bit is implemented as a switch that can take one of two possible values
    - One bit can represent information about any two states

Why is Computer based on 1 or 0s?

So a 1 or 0 is a data and they mean nothing by itself, we need to assign meaning to 1 or 0 to make it represent some information.

- 2 Bits can represent 4 states
  - 00,01,10,11
- 3 Bits can represent 8 states
- In general *n* Bits can represent 2^n states.
  - We can assign different meaning to each state.
  - For example, a number, a character.
  - It is up to the programmer to determine how to interpret a series of 1s and 0s.

Representing Positive and Negative Numbers

- We can represent positive and negative numbers using 1s and 0s.
  - e-g: ***0***0000111 represent 7 using binary number system.
  - Decimal to Binary conversion method is used to represent decimal numbers using binary
- Negative numbers are represented using 2's complement
  - We use the process of 2s complement to store any negative number
  - ***1***1111001 represents -7
- Left most bit represents the sign and indicates to the processor about how to treat the remaining bits as positive or negative.

- Real Numbers are also represented in binary form
  - 387.53 is a real number.
- Real numbers are represented through two numbers: mantissa and integer power.
  - Real number is usually implemented through 4 bytes
- e-g: 387.53 is represented as $.38753 \times 10^2$
  - Mantissa: **38753**, Power: **2**
  - Mantissa in Binary: 00000000100101110110000 1
  - Exponent in Binary: **00000010**
  - 387.53 = 00000000100101110110000**100000010**

- e-g: 0.0038753 can be represented as 0.38753×10^{-2}
  - Mantissa: **38753**, Power: **-2**
  - Mantissa: **38753**
    - Mantissa: 00000000100101110110000 1
  - Power: **-2**
    - Power: **1111111 0**
  - 387.53 = 00000000100101110110000**1111111 0**