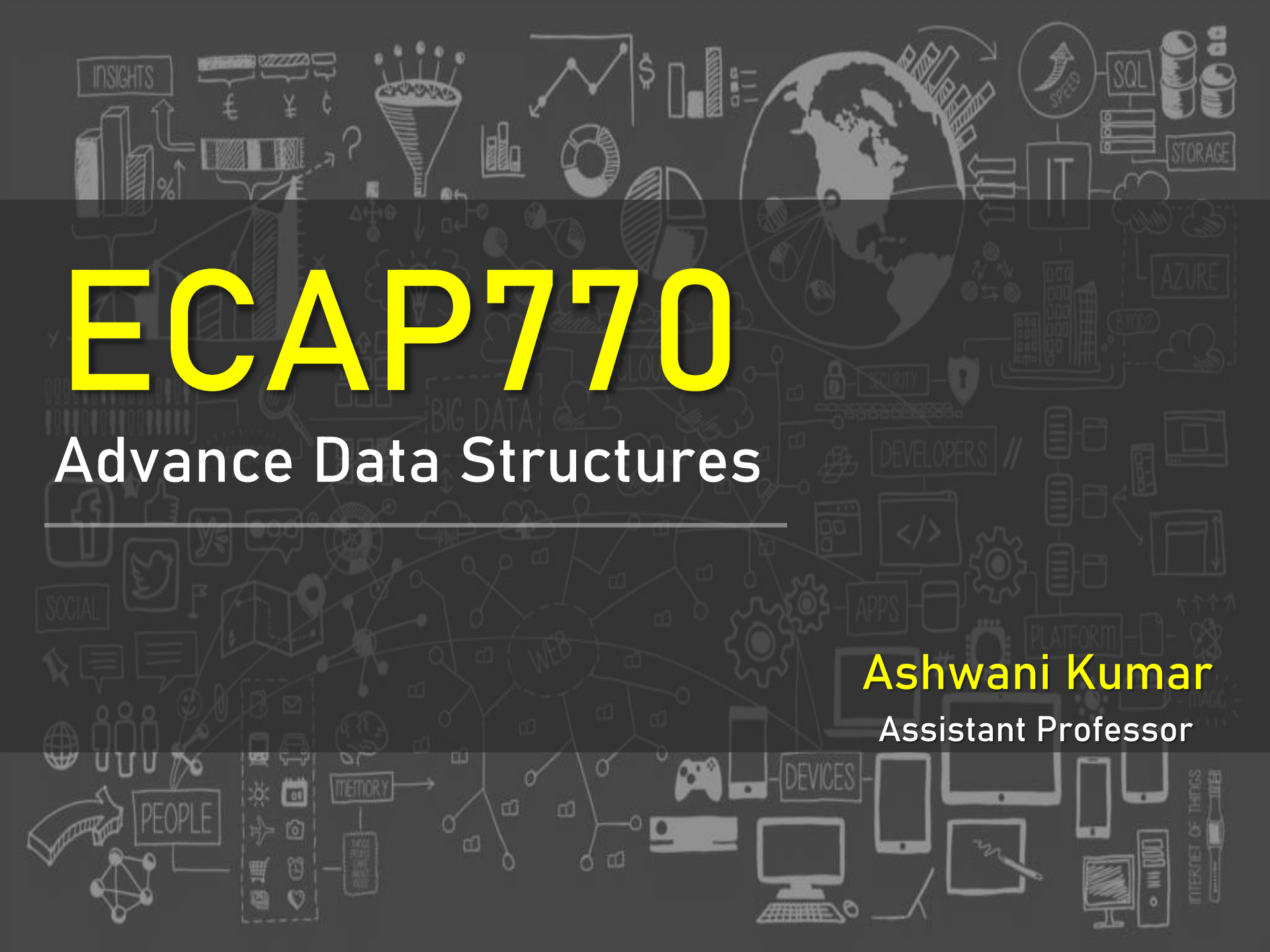


ECAP770

Advance Data Structures

Ashwani Kumar

Assistant Professor

Learning Outcomes

After this lecture, you will be able to

- learn Data structure,
- know the Types of Data structure.

Data Structures

- Data Structure is method or technique to data organization, management, and storage format in the computer so we can perform operations on the stored data more efficiently.
- It is a set of algorithms that we can use in any programming language to structure the data in the memory. Examples are arrays, Linked List, Stack, Queue, etc.

Data Structures

- Data Structures are widely used in the field of Computer Science i.e. Operating System, Compiler Design, Artificial intelligence, Graphics and many more.
- Data Structures are the main part of many computer science algorithms as they enable the programmers to handle the data in an efficient way.

Data Structures

- Data structures are the building blocks of any program or the software. Choosing the appropriate data structure for a program is the most challenging task for a programmer.

Basic Terminology

- Data: Data can be defined as an elementary value or the collection of values, for example, student's name and its id are the data about the student.
- Group Items: Data items which have subordinate data items are called Group item, for example, name of a student can have first name and the last name.

Basic Terminology

- Record: Record can be defined as the collection of various data items, for example, if we talk about the student entity, then its name, address, course and marks can be grouped together to form the record for the student.
- File: A File is a collection of various records of one type of entity, for example, if there are 60 students in the class, then there will be 20 records in the related file where each record contains the data about each student.

Basic Terminology

- Attribute and Entity: An entity represents the class of certain objects. it contains various attributes. Each attribute represents the particular property of that entity.
- Field: Field is a single elementary unit of information representing the attribute of an entity.

Need of Data Structures

- As applications are getting complex and amount of data is increasing day by day, there may arises many problems:
- Processor speed: To handle very large amount of data, high speed processing is required, but as the data is growing day by day to the billions of files per entity, processor may fail to deal with that much amount of data.

Need of Data Structures

- Data Search: Consider an inventory size of 100 items in a store, If our application needs to search for a particular item, it needs to traverse 100 items every time, results in slowing down the search process.
- Multiple requests: If thousands of users are Searching the data simultaneously on a web server, then there are the chances that a very large server can be failed during that process To solve these problems data structures are used.

Types of Data structure

There are two types of data structures:

- Primitive data structure
- Non-primitive data structure

Types of Data structure

- Primitive Data structure: The primitive data structures are primitive data types. The int, char, float, double, and pointer are the primitive data structures that can hold a single value.
- Non-Primitive Data structure: The non-primitive data structure is divided into two types:
  - Linear data structure
  - Non-linear data structure

Types of Data structure

- Static data structure: in this type data structure where the size is allocated at the compile time. Therefore, the maximum size is fixed.
- Dynamic data structure: It is a type of data structure where the size is allocated at the run time. Therefore, the maximum size is flexible.

Linear Data Structures

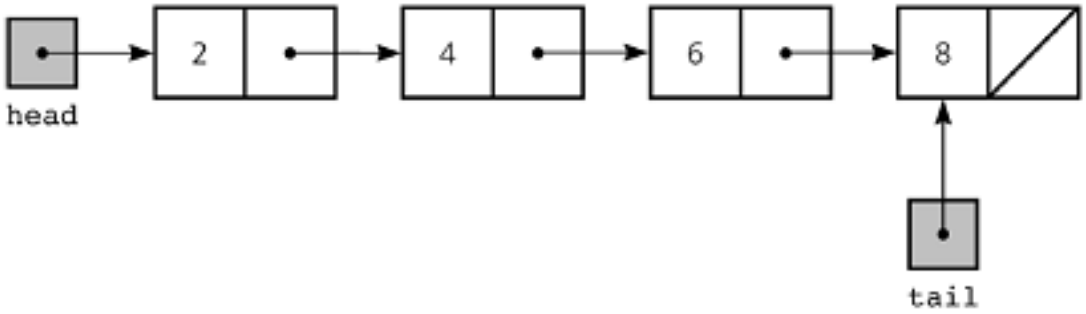
- A data structure is called linear if all of its elements are arranged in the linear order. In linear data structures, the elements are stored in non-hierarchical way where each element has the successors and predecessors except the first and last element
- The arrangement of data in a sequential manner. Data structures used for this purpose are Arrays, Linked list, Stacks, and Queues.

Linear Data Structures : Arrays

- Arrays: An array is a collection of similar type of data items and each data item is called an element of the array.
- The data type of the element may be any valid data type like char, int, float or double. — The individual elements of the array age are: — age[0], age[1], age[2], age[3], age[98], age[99].

Linear Data Structures : Linked list

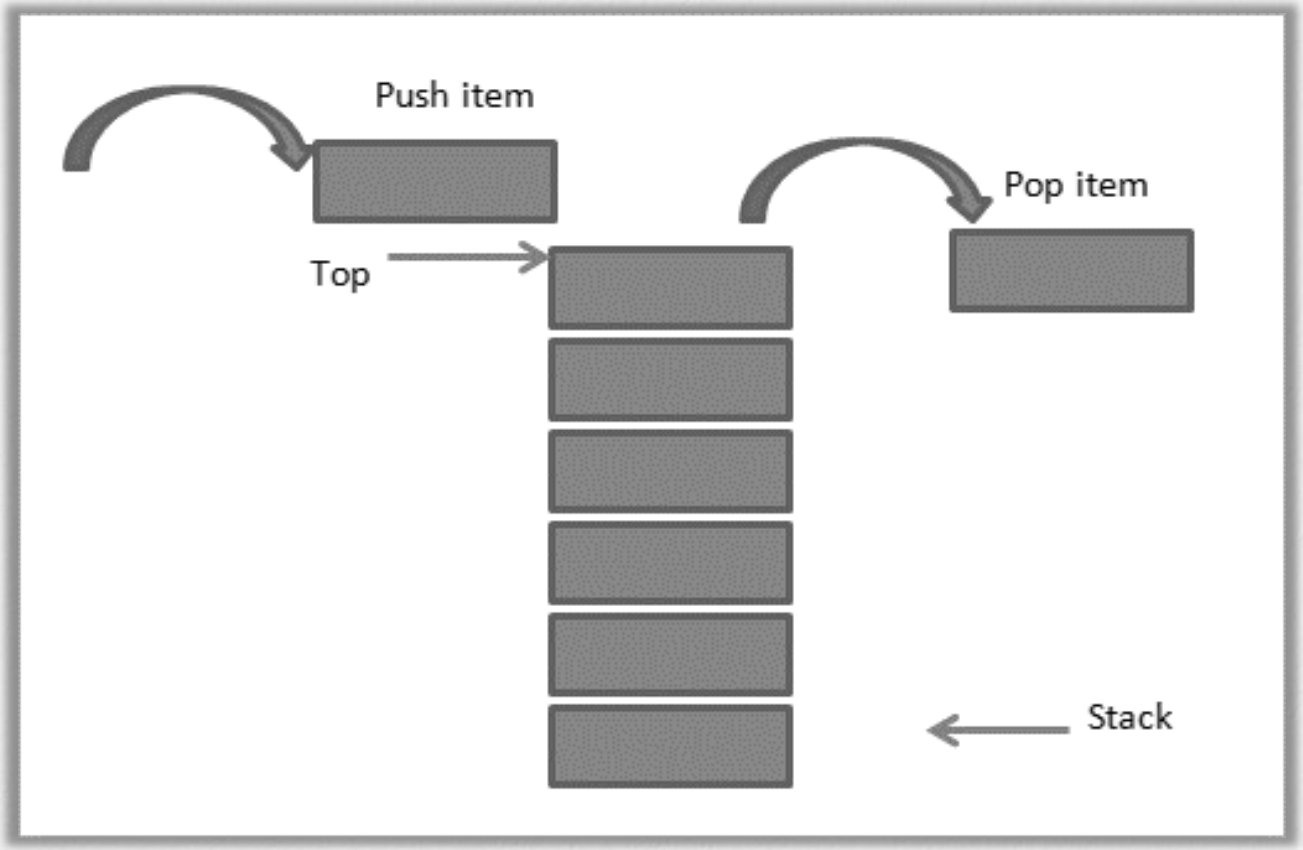
- Linked List: Linked list is a linear data structure which is used to maintain a list in the memory.
- It can be seen as the collection of nodes stored at non-contiguous memory locations. Each node of the list contains a pointer to its adjacent node



Linear Data Structures : Stack

- Stack: Stack is a linear list in which insertion and deletions are allowed only at one end, called top.
- A stack is an abstract data type, can be implemented in most of the programming languages. It is named as stack because it behaves like a real-world stack, for example: - piles of plates or deck of cards etc.

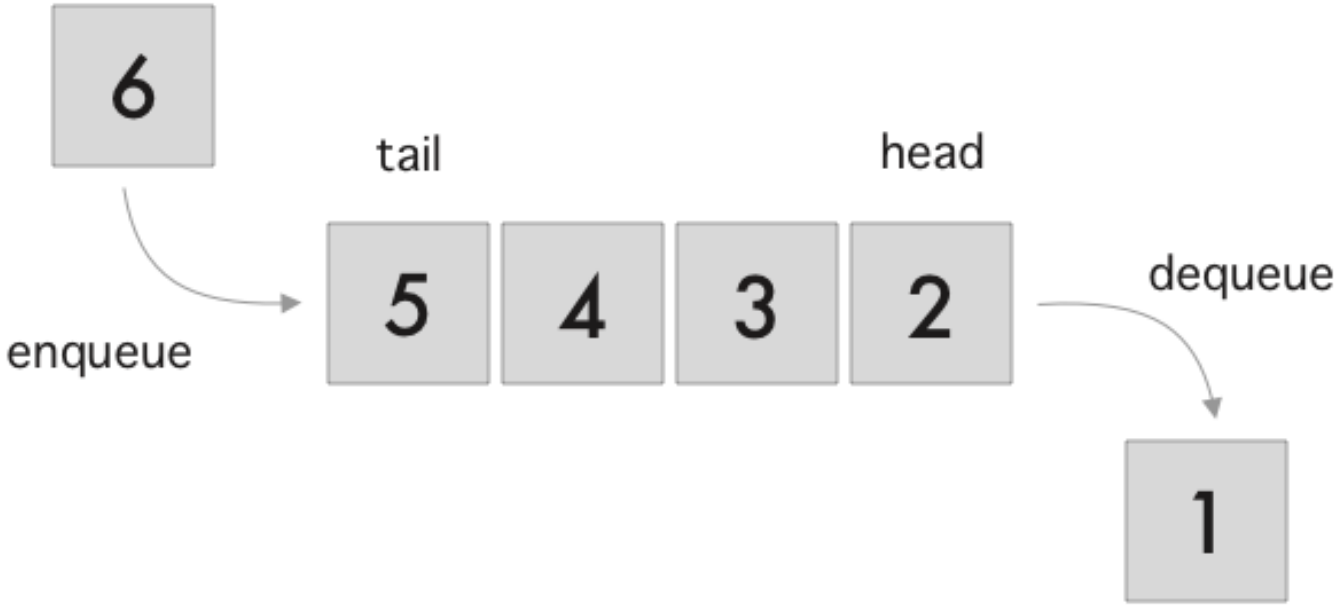
Stack



Linear Data Structures : queue

- Queue is a linear list in which element can be inserted only at one end called rear and deleted only at other end called front.
- It is abstract data structure, similar to stack. It is open at both end therefore if follows first-in-first-out (FIFO) technique for storing the data items.

Queue



Non Linear Data Structures

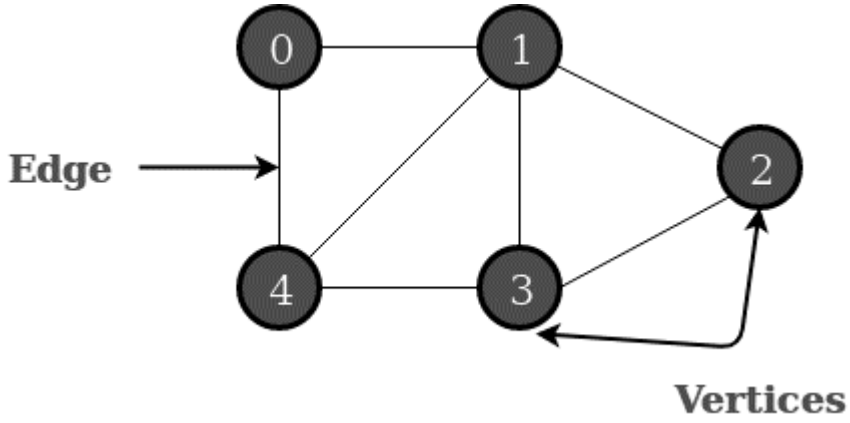
- This data structure does not form a sequence i.e. each item or element is connected with two or more other items in non-linear arrangement. The elements are arranged in a random manner.
- Examples : tree and graphs

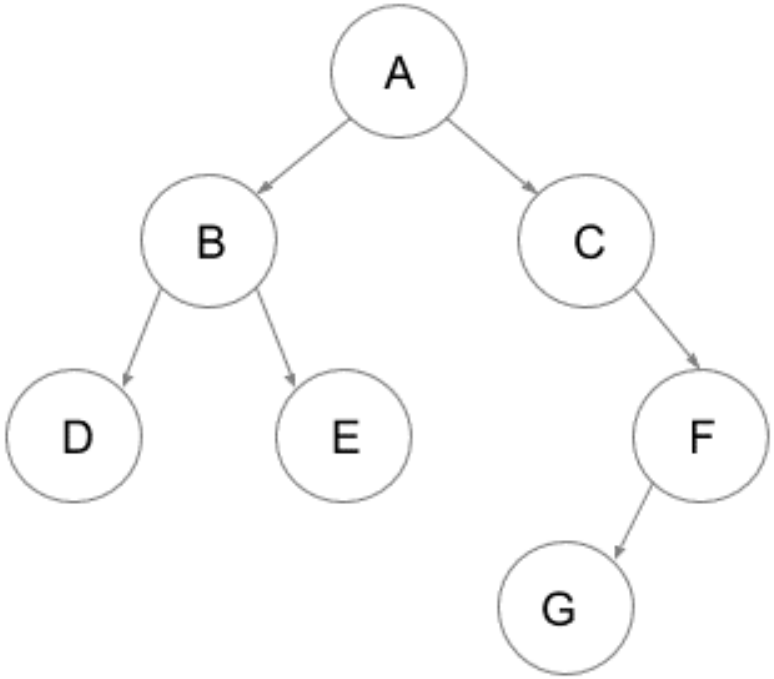
Non Linear Data Structures

- Trees: Trees are multilevel data structures with a hierarchical relationship among its elements known as nodes.
- Graphs: Graphs can be defined as the pictorial representation of the set of elements (represented by vertices) connected by the links known as edges

Graph and Tree

Graph





Tree

Advantages of Data structures

- Efficiency: If the choice of a data structure for implementing a particular ADT is proper, it makes the program very efficient in terms of time and space.
- Reusability: The data structure provides reusability means that multiple client programs can use the data structure.
- Abstraction: The data structure specified by an ADT also provides the level of abstraction. The client cannot see the internal working of the data structure, so it does not have to worry about the implementation part. The client can only see the interface.

That's all for now...