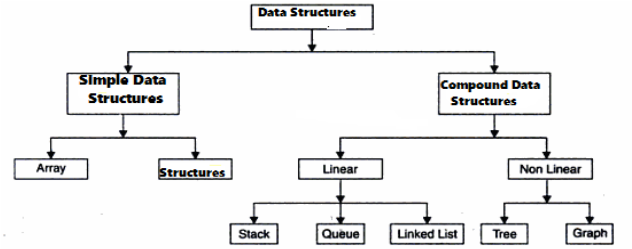
Chapter3

Data Structures and Operations

DataStructure

A data structure is a particular way of organizing data.

Classification of Data Structures



Data structures can be classified into two types,Simple Data Structure and Compound Data Structure.

Simple data structues are normally built from primitive data types like integer,character etc.

Simple data structures can be combined to form more complex structures called compound data **structures**.Compound data structures are classified into two,**Linear data structures** and **Non-Linear data structures**.

Depending upon memory allocation,data structures are classified into two,Static data structures and **Dynamic data structures.**

The size of static data structure is fixed ie,it cannot be changed.Data structure implemented using array are static in nature.In case of dynamic data structure ,memory is allocated during run-time.Data structure implemented using Linked list are dynamic in nature.They grow and shrink during execution.

Operations on data structures

The major operations performed on data structures are **traversal**,**searching**,**inserting**,**deletion** ,**sorting** and **merging**.

Traversal

The process of visiting each elements in a data structure is called traversal.It starts from first element to the last element.

Searching

The process of finding the location of a particular element in a data structure is called searching.It is based on a condition.

Inserting

The process of adding a new data at a particular position in a data structure is called inserting.The position is identified first and insertion is then done.

Deletion

The process of removing a particular element from the data structure is called deletion.

Sorting

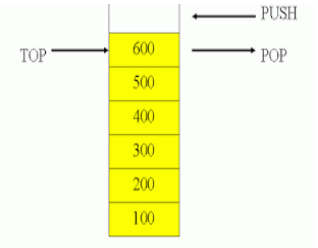
The process of arranging elements of a data structure in an order(ascending or descending) is called sorting.

Merging

The process of combining elements of two data structures is called merging.

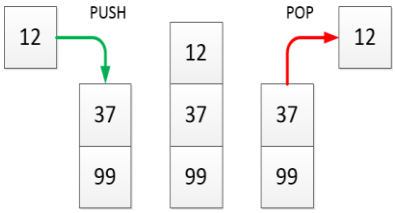
Stack

A stack is a linear data structure in which insertion and deletion takes place at one end called **Top**.It follows LIFO(LastInFirstOut)principle.



Operations on Stack

The insertion operation is referred to as **Push** and deletion operation as **Pop**.



Stack overflow occur when a program attempt to add elements to a fullstack.Stack underflow occur when a program attempts to remove an element from an empty stack.

Implementation of Stack

A Stack can be implemented in two ways

1)Using an array.

2)Using a Linked list.

In implementing a stack using an array,the number of elements is limited to the size of array.When an element is added to the stack top of stack(TOS) is incremented by1.In array implementation a stack can towards end index of the array.The array implementation has following dis-advantages

1. Memory space is wasted.
2. The size of stack is fixed.
3. Insertion and deletion operation involved more data movement.

In Linked list implementation of a stack the first element inserted points to second element,second element points to third element and so on.Here stack is not of fixed size.

Applications of Stack

The following are the applications of a stack

- Decimal to binary conversion.
- Reversing a string.
- Infix to postfix conversion.

Queue

A Queue is a linear data structure in which insertion takesplace at one end(**Rear end**) and deletion takes place at other end(**Front end**).It follows FIFO(FirstInFirstOut) principle.The elements are processed in the order in which they are received.

Note:The number of elements in a queue is called Length of the queue.

Operations on a Queue

The operations that can be performed on a Queue are,

a)Inserion operation(**ENQUEUE**).

b)Deletion operation(**DEQUEUE**).

Insertion operation

The insertion operation adds a new element adds a new element at the rear end of the queue.

Deletion operation

The deletion operation removes an element from the front end of the queue.

An attempt to delete an element from an empty queue is called **queue underflow**.

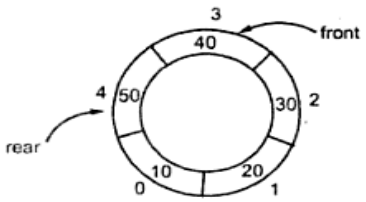
Queue overflow results from trying to add an element onto a full queue.

Applications of Queue

- Job Scheduling.
- Resource scheduling.
- Handling of interrupts in real-time systems.

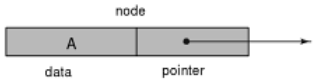
Circular Queue

A circular queue allows to store elements without shifting any data within the queue.The main advantage of circular queue is that we can utilize the space of queue fully.It allows to store data without shifting any data in the queue.



Linked list

A linked list is a collection of data item(nodes).Each node consists of data and a link(pointer) to the next node.



A Node in a Linked list

The last node in a linked list has a Null,which marks the end of the list.

Linked list

Implementing linked list

Linked list can be implemented in two ways,both as stack and queue ie,

- using array (static linked list)
- using self-referential structures(Dynamic linked list)

Difference between Array and Linked list

The following are the difference between array and linked list

- The size of array is fixed whereas a linked list can grow or shrink.
- Array elements are accessed randomly.Linked list elements are accessed sequentially.

Operations on linked list

The following operations can be performed on a linked list

a. Creating a Linked list.
b. Traversing a Linked list.
c. Inserting elements to a Linked list.
d. Deleting elements from a Linked list

Creating a Linked list

A Linked list can be created by dynamically allocating memory(using **new** operator).

Traversing a Linked list

Traversing means visiting all elements.In a linked list traversal begins from first node.The pointer start gives address of first node.The **link part** of first node points to **address** of second node and so on.The traversal continues till **last node** is reached(Contains Null).

Inserting elements to a Linked list

The process of placing a node in a particular position is called insertion.A node can be placed anywhere.To place a node at the beginning by copying the content of **Start** into **link part**(**Pointer**) of new node.To insert a node at the end the address of new node is copyed into **link part** of last node.

Deleting elements from a Linked list

Deleting is the process of removing a node from the list.To remove a node either its position or data is given.To remove first node the link part content of **First Node** is copied to **Start**.

Note:*In a doubly linked list each node points to next node aswell as to previous node.Complex data structures like trees are created using doubly linked list.*