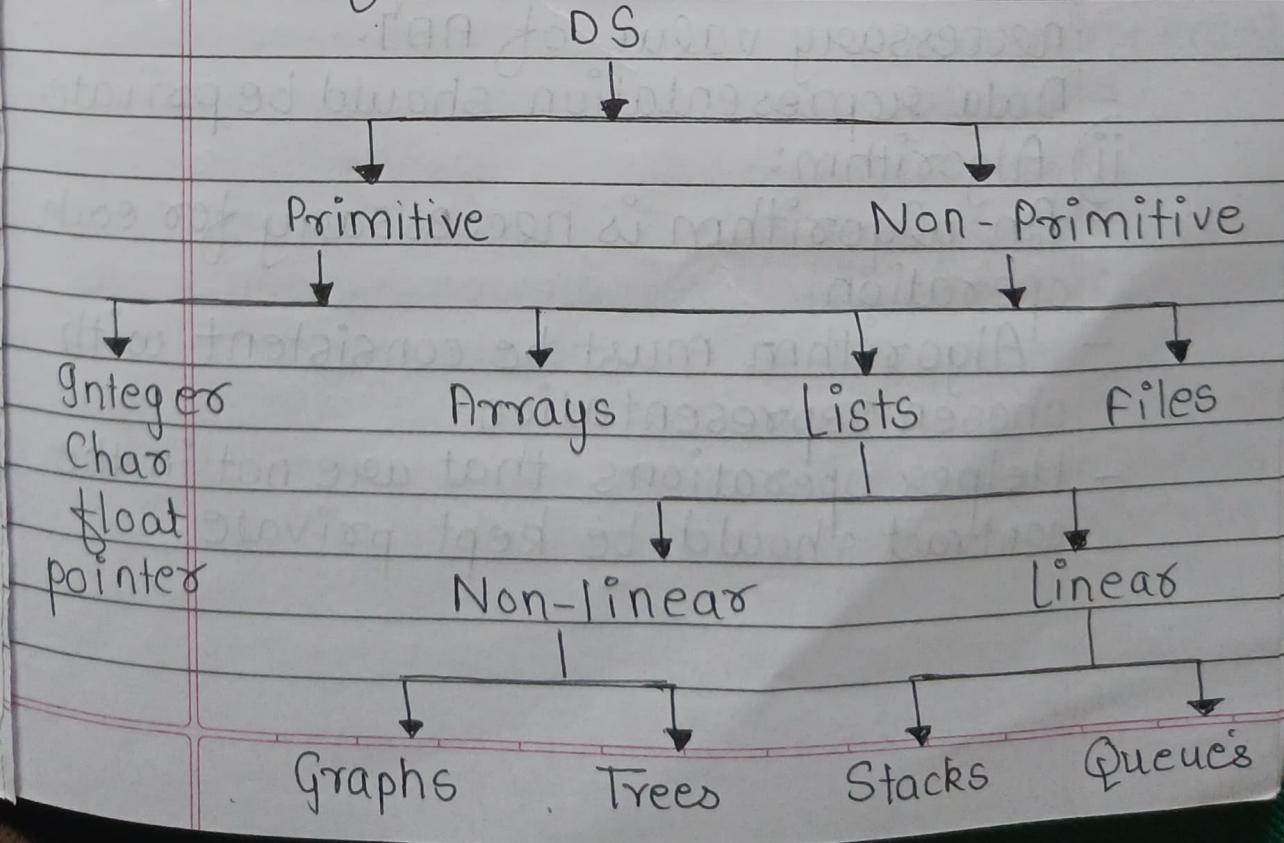
PAGE No.
DATE / /

Unit I & 2 - Introduction to DS

* Data Structure:

1) It is a way of storing and organizing data in a computer so that it can be used efficiently
2) Different kinds of data structures are suitable for different kinds of applications
3) Storing and retrieving can be carried out on both data stored in both main memory and in secondary memory.
4) Many algorithms have been developed for storing data efficiently
5) Data structures help us to understand of one element with anothe ~~e~~element

Classification:



PAGE No.
DATE / /

* Abstract Data types [ADT]

1) It is a data type organized in such a way that the specification of objects and specification of operations on the object is separated from representation of objects and implementation of operation
- User of ADT only needs to know the set of operations available for the data type.
- It is okay even if the user doesn't know how the operations are applied.

Examples:
Lists, sets, graphs, stacks are examples of ADT

eg] Implementation of ADT
- To implement ADT you need to choose:

i) A data representation:
- User must be able to represent all neccessary value of ADT.
- Data representation should be private.

ii) Algorithm:
- An algorithm is neccessary for each operation.
- Algorithm must be consistent with chosen representation.
- Helper operations that are not in the contract should be kept private.

PAGE No.
DATE / /

*) Data structure operations:
- Data in data structures is processed by means of certain operations:
- The operations are:
1) Traversing
2) Searching
3) Inserting
4) Deleting
5) Merging
6) Sorting

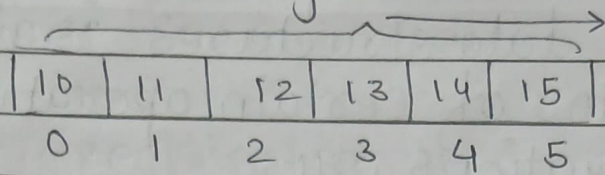
→ Data Structure [Linear]:
- They are classified as linear or non-linear
- If elements of a data structure form a sequence or linear list then the data structure is known as linear data structure
- Such linear structures are represented in two ways: Arrays, linked list.

Arrays:
Linear relationship between elements is represented by means of sequential memory locations.

Linked lists:
Linear relationship between elements is represented by means of pointers or links.

PAGE No.
DATE / /

* Linear Arrays:



1) It is a list of finite number of homogeneous data elements stored in adjacent memory locations.
2) Elements of array are accessed by index set of 'n' consecutive numbers.
   n = total no of elements in array
3) Length of Array = Upper bound - lower bound + 1.
4) Array declaration must give three items of information:

```
int      nums [n]  ← Index set or size
 ↑         ↑
datatype  name
```

5) Memory can be allocated statically during compile time or dynamically during run time by taking user input for length.
6) Address of first element is denoted by Base(Arr) or base address.
7) Using base address, address of any element of an array can be calculated by using formula:

Loc(Arr[k]) = Base(Arr) + W(k - lower bound)

PAGE No.
DATE / /

* Multidimensional Arrays:
- Its data elements are referenced by more than one subscript.
- Two dimensional array which consists of 'M' rows and 'N' columns is a collection of M * N elements such that each element is specified by pair of integers j, k called subscripts.
- A element with subscripts [j] and [k] is ~~dest~~ denoted by A[j][k].
- They are also known as Matrices.
- Two dimensional array can be stored in two ways:
  1) Column by Column:
  - It is known as Column major method.
  2) Row by Row
  - It is known as Row major method.

* Pointers:
1) They are special variables which contain the address of another memory location.
2) They are useful in accessing any memory location directly
3) Address of memory location is a long integer and is stored in pointer type variable.
4) Adding or subtracting two pointers gives number of bytes between two memory address.

PAGE No.
DATE / /

Ex:
Declare base variable:
int siu;
Declare pointer type variable to siu:
int *p;
Establish relation between base and pointer:
p = &siu;

* Structures:

1] In order to store a group of data items we need structures. Structure is a constructed data type for packing different data types that are logically related.
2] Structures are used for organizing complex data in a simple and meaningful way.
3] Structure is analggous to the record of a database.

Exp] Structure for:
a) Student info: Roll no, name, age, address.

25/10/23

PAGE No.
DATE / /

Unit 3: Stacks

→ Stack:

1) It is a non-primitive linear data structure. It is a (LIFO) data structure [Last-in-first-out].
2) It is an ordered collection of homogenous items.
3) Stack is a data structure in which data is added and removed at only one end called the top.
4) If an attempt is made to add new element beyond the maximum size, stack full condition is encountered.
5) Stack empty conditions is reached, if tried to remove elements beyond the base of stack

→ Why are stacks used over linear arrays:

1) Linear arrays allows insertion and deletion at any place in the list.
2) There are certain situations where one wants to restrict this property. For this purpose stacks and queues are used.

Examples of Stacks:

a) Stack of magazines
b) Stack of plates in cafeteria

c) Stack of books in library
d) Stack of news papers.

Arrays	Stacks
1) Insertion is allowed	1) Insertion and deletion both are allowed.
2) Arrays are static	2) Stacks are dynamic
3) It's a linear data structure	3) It's a linear data structure
4) You can deal with any element	4) You can deal with only top element

*) Representation of Stacks:
- Static [using Arrays]
- Dynamic [using linked list]

→ Procedures to be carried on stack:
1) Put something on top (PUSH).
2) Remove the top item (POP).
3) Look at the top item, without removing it from stack (PEEP).
4) Check to see if stack is empty.

→ Applications of Stack:
1) Illustration on function call.
2) Reversing a list.
3) Conversion of an infix expression into a postfix expression

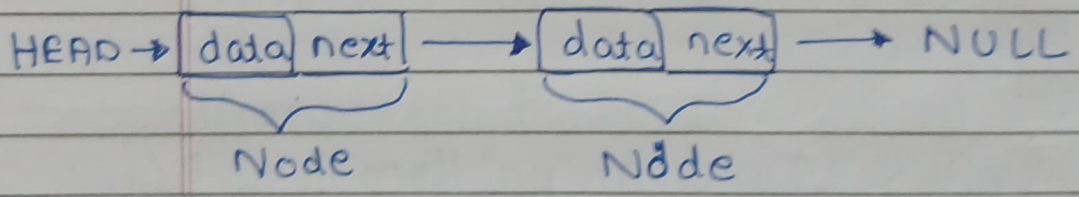
4) Evaluation of a postfix expression
5) Recursion
6) Parenthese checker

* Infix Expression
1) Arithmetic operator exists in between two operands.
(A+B)

* Postfix Expression
1) It is the form of an arithmetic expression in which arithmetic operator is fixed (place) after (past) its operands
2) It is also reverse polish notation.
3) It does not require brackets.

Unit 5 - Linked List

* What is linked list:
- It is a linear data structure which looks like a chain of nodes, where each node is a different element.
- Linked list elements are not stored at a contiguous location.
- It is a chain of nodes, each node contains information such as data and a pointer to the next node in the chain.
- LL has a head pointer which points to the first element of the linked list
- If list is empty head points to null or nothing.



* Features of LL / Why are linked lists needed:
1) Dynamic Data: Size or memory can be allocated or de-allocated at run time based on operation.
2) Ease of Insertion / Deletion:
Since no elements need to be shifted after insertion and deletion only address is updated thus insertion /

deletion is easier.

3) Efficient memory utilization:
Size of linked list increases or decreases as per requirement, this avoids wastage of memory.

4) Implementation:
Various advanced data structures can be implemented using a linked list like a stack, queue, graph, hash maps etc.

* Types of linked list:
1) Single-linked list
2) Double linked list
3) Circular linked list

→ Operations of Linked List:

1) Insertion:
- Insertion operation can be performed in three ways:
  a) Inserting at beginning of list.
  b) Inserting at end of list.
  c) Inserting at specific location in the list.

2) Deletion:
- Deletion operation can be performed in three ways:
  a) Deleting from beginning of list.
  b) Deleting from end of list
  c) Deleting a specific node.

3) Search:
- It is a process of retrieving a specific node from anywhere in the list.

4) Display:
- This process displays the elements of a single-linked list.

→

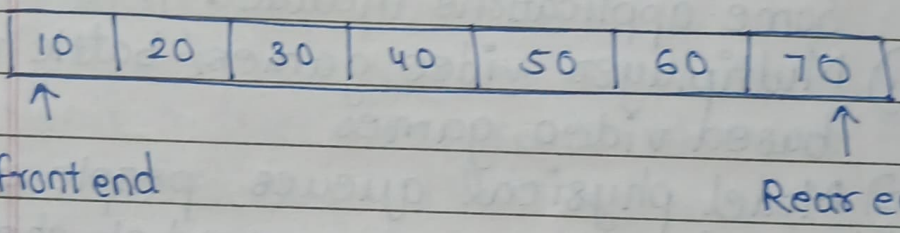
Array	Linked list
~~linked list~~	~~Array~~
1) Arrays are stored in contiguous location	1) Linked lists are not stored in contiguous location
2) Fixed in size	Dynamic in size
3) Memory is allocated at compile time	Memory is allocated at runtime.
4) Uses less memory than linked lists	Uses more memory because it stores both data and address of next node
5) Elements can be accessed easily	Element accessing requires the traversal of whole linked list.
6) Insertion and deletion operation takes time	6) Insertion and deletion operation is faster

PAGE NO
DATE / /

→ Unit 4: Queues

*) Queue

1) It is a non-primitive linear data structure
2) It is a ordered homogeneous group of elements in which elements are added at one end called rear. Deletion is done ~~to~~ from other end called front.
3) It is a First In First Out (FIFO) data structure.



→ Representation of Queue

i) Static representation using arrays.
ii) Dynamic representation using linked list.

Array declaration: int queue[Maxsize];

→ Conditions:

1) FRONT = REAR — Queue is empty.
2) Therefore initial condition of a queue is: FRONT = REAR = -1 or FRONT = NULL
3) When an element is deleted from the queue: Front = Front + 1;
4) When an element is added to the queue

value of rear is increased by 1
Rear = Rear + 1;

→ ~~Queue Operations:~~

→ Application / Uses of queue data structure

* There are multiple use cases of a queue data structure. Something that involves first come, first serve might use queues.
Some applications include:
1) Multiplayer video games / battle royale based video games
2) Model physical queues, people waiting in a line for a supermarket checkout.
3) When sending data over the internet, various data ~~packests~~ packets wait in a queue to be sent.
4) A server responding to requests. Usually, these requests are stored in queues. The first come, first respond policy is followed in such cases.

PAGE No.
DATE / /

Module 6: Searching & Sorting

*) Searching techniques:
1) Sequential search
2) Binary search

*) Sorting definitions:
1) Bubble sort
2) Selection sort
3) Insertion sort
4) Radix sort

→ Searching Algorithms:

1) They are methods or procedures used to find a specific item or element within a collection of data.
2) They are widely used in computer science and are crucial for tasks like searching for a particular record in a database, finding an element in a sorted list or locating a file on a computer.
3) Commonly used searching algorithms:
   i) Linear search
   ii) Binary search
   iii) Hashing
   iv) Interpolation search
   v) Tree-based searching
   vi) Ternary search

→ Linear / Sequential Search

a) It is a searching algorithm where we start from one end and check every element of the list until the desired element is found.
b) It is the simplest searching algorithm.

→ Binary Search

a) It is a searching algorithm for finding an element's position in a sorted array.
b) It can be implemented only ~~one o~~ on a sorted list of items.
c) If elements are not sorted already, we need to ~~stop the~~ sort them first.
d) It can be implemented in two ways:
   i) Iterative method
   ii) Recursive method
   Recursive method follows the divide and conquer approach.

* SORTING

1) Bubble Sort:
a) It is a sorting algorithm that compares two adjacent elements

and swaps them until they are in the intended order.

Working:

a) Starting from first index, compare the first and second elements.

b) If first element is greater than second element, they are swapped.

c) Now, compare second and third elements swap them if they are not in order

d) The above process goes on untill the last element.

~~e) All~~

2) Selection sort:

a) It is a sorting algorithm that selects the smallest element from an unsorted list in each iteration and places that element at the begining of the unsorted list.

working:

i) Compare minimum with second element. If second element is smaller than minimum assign second element as minimum.

ii) Compare minimum with third element. If the third element is smaller, then assign minimum to third element otherwise do nothing. The process goes on until the last element.

iii) After each iteration, minimum is placed in front of unsorted list.

3) Insertion Sort

a) It is a sorting algorithm that places an unsorted element at its suitable place in each iteration

b) It works similarly as we sort cards in one hand in a card game.

Working:

1) First element in the array is assumed to be sorted. Second element is stored separately in key.

2) Compare key with first element. If first element is greater than key, then key is placed in front of first element

3) Now first two elements are sorted.

4) Take third element and compare it with the elements on the left of it. Place it just behind the element smaller than it. If there is no element smaller than it then place it at the beginning of the array.

4) Radix Sort

a) It is a linear sorting algorithm that sorts elements by processing them digit by digit.

b) It is an efficient sorting algorithm

for integers or strings with fixed-size keys.

c) Elements are distributed into buckets based on each digits value.

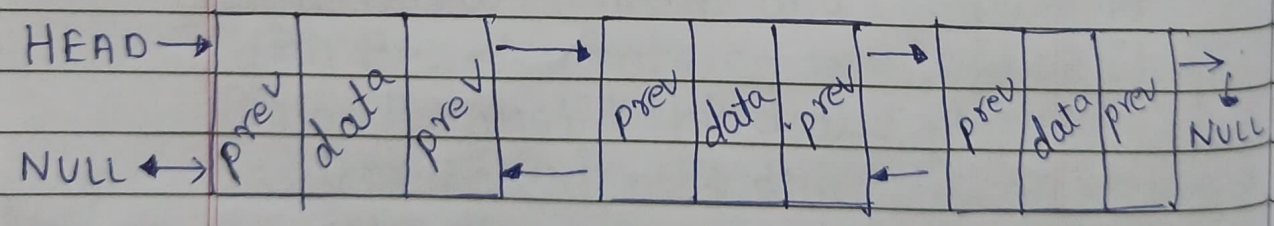
d) By repeatedly sorting elements by their significant digits, from least to most significant, radix sort achieves final sorted order.

PAGE No.
DATE / /

Unit 5: Linked list - Continued

* Doubly linked list / Double linked list

a) It is a type of linked list. It is a linear data structure which looks like a chain of nodes where each node is a different element.
b) Linked list elements are not stored in contiguous location. It is a chain of nodes each node contains information such as data and a pointer to the next node in chain.
c) Traversal of items can be done in both forward and backward directions as every node contains an additional previous pointer that points to previous node.



* Circular linked list:

a) It is a type of linked list. It is a linear data structure which looks like a chain of nodes where each node is a different element.
b) Linked list elements are not stored in contiguous location. It is a chain of

nodes, each node contains information such as data and a pointer to the next node in chain.

c) It is a type of linked list in which the first and last nodes are also connected to each other to form a circle, there is no NULL at the end.