WEEK-1

Data Structure:

Organizing managing and storing data is important as it enables easier access and efficient modification.

- It enables you to store collection of data and relate them by organizing your data in such a way that we can perform operation on them.

Need of data structure

- Processor speed:

To handle very large amount of data high speed processing is required.

- Data search:

consider an inventory size of 106 items in a store. If an application needs to search for a particular item it leads traverse 106 items every time. In order to solve this problem data structures are used.

- Multiple requests:

If thousands of users are searching the data simultaneously in a web server then there is a chance of server fail so to overcome this data structures are used.

Advantages of Data structures

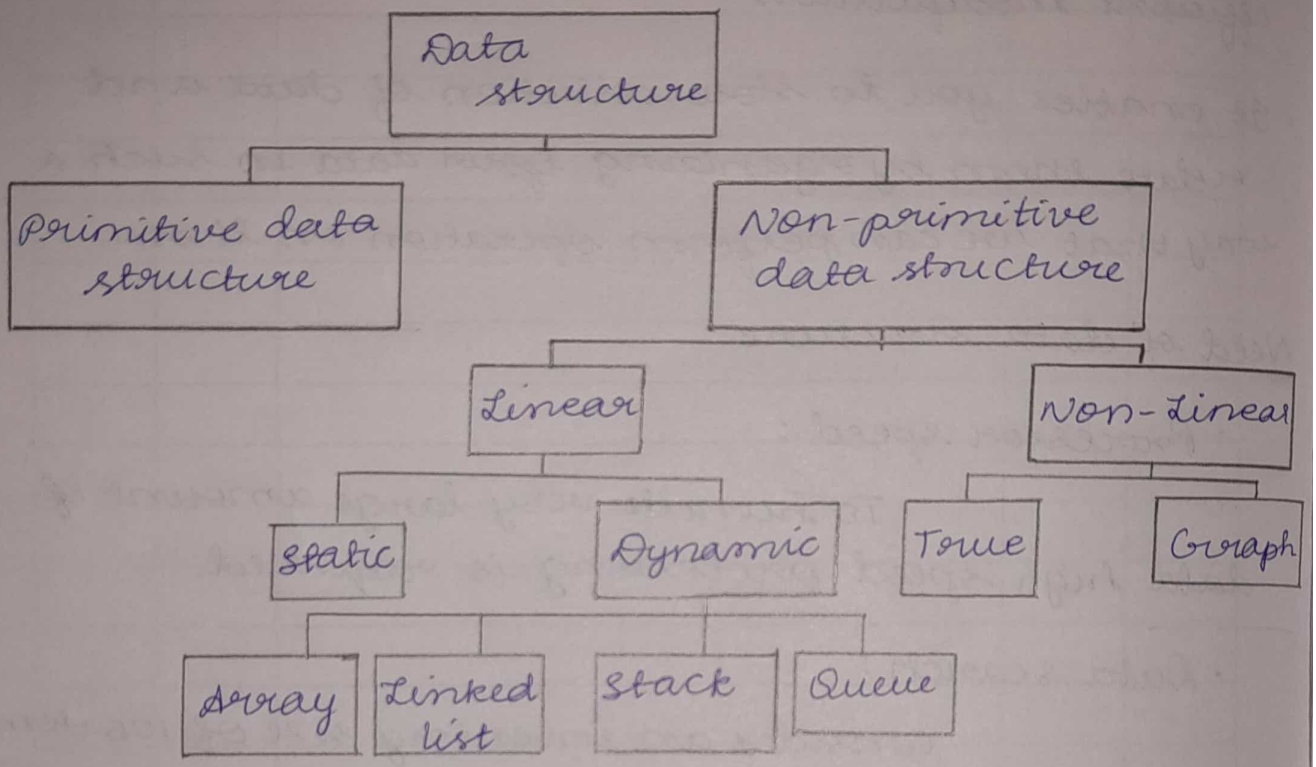
1. Efficiency:
   It depends upon the choice of data structure.

2. Reusability:
   Data structures are reusable that is once a particular data structure is implemented it can be

reused at any other places.

- Abstraction:

Data structure is specified ADT (Abstract data type) which provides a level of abstraction.



Operations on data structure:

- Traversing:

Every data structure consists a set of data elements. Traversing a data structure means visiting each element of the d.s inorder to perform some operation like searching, sorting.

- Insertion:

It is a process of adding the elements to data structure at any location.

- Deletion:

The process of removing the element from the d.s is called deletion. We can delete an element at any random location.

. Searching:

The process of finding the presence of an element within the d.s is called searching.

There are 2 algorithms:

1. Linear search
2. Binary search

. Sorting:

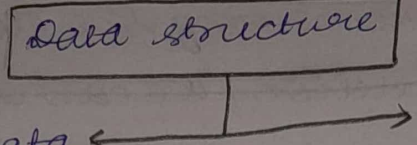
The process of arranging the data structure in a specific order is known as sorting. There are many algorithms for eg insertion sort, selection sort, bubblesort.

. Merging:

When 2 list i.e list A and list B of size M and N respectively of similar type of elements joined to form the third list list C of size (M+N) then this process is called merging.

Classification of Data structure.

Data structures are normally divided into 2 categories: 1) Built in data structures.
2) User defined data structures.



Built in data
- ↳ List
- ↳ Tuple
- ↳ Dictionary
- ↳ Set

User defined data structures
- ↳ Stack
- ↳ Queue
- ↳ Tree
- ↳ Linked list
- ↳ Graph
- ↳ Hash map

Built in data structure:

• These are the data structures already created by programmers which helps the users to use them inorder to obtain solution faster.

1. List: This is used to store the data of various datatypes.

2. Tuple: This is same as list. The main difference is that the data present in the list can be change but the data present in tuple can't be changed.

3. Dictionary:

It contains unordered info & are used to store data in key value pairs.

4. Set: It contains unordered elements which are unique, it doesn't include elements in the repetation.

User defined data structure

1) Stack:
It works under concept of last in first out (LIFO) and it is a linear data structure.

2) Queue:
It works in the concept of first in first out (FIFO) and it is a linear data structure.

3) Tree:
It is a user defined data structure that works on the concept of trees in nature. This d.s starts from the up and goes down a with its branches or nodes.

4) Linked list:
It is the order of data elements which are connected together with the links.

5) Graph:
It is an illustrative representation of a graph of objects where few pairs of objects are joined by links.

6) Hash map:
It is a data structure that matches the key with its value pairs.

Characteristics of data structure.

1. Linear:
It describes whether the data items are arranged in sequential form like an array.

2. Non-Linear:
Here data items are not in a sequenti-al form like tree, graph.

. Static:
Here the data items assosiated with memory location at compile time that are fixed.

4. Homogeneous:

In all the data types is same.

Eg: Array

5. Non-Homogeneous (Heterogeneous):

Here data types of an element may or may not be same.

6. Dynamic:

It defines the shrinking and expanding of data items at run time. Eg: Linked list.

7. Time complexity:

The execution time of a data structure should be minimum as possible.

8. Space complexity:

The memory usage for data items in a d.s should be less as possible.

Difference b/w linear and non-linear data structure

Linear	Non-linear
• Elements are ordered in a linear manner.	• Elements are ordered in a non linear manner.
• Each element is attached to the next as well as the previous element.	• A group of element or each element is attached to any other elements.
• Implementation is easy	• Implementation is problematic
• Memory not used efficiently	• Memory used efficiently.
• Used in s/w development.	• Used in AI, ML, research
• Eg: Arrays, list, linked list, stakes, queues.	Egs: tree and graphs

Abstract data type (ADT):

It is a special kind of data type whose behaviour is defined by set of values and set of operations. ADT mentions what operations are to be performed and how these operations will be implemented. There are 3 main ADTs:

1. List ADT
2. Stack ADT
3. Queue ADT

1. List ADT:

The data is generally stored in key sequence in a list which has head structure consist of count, pointers, address of compare function. The following operations can be performed under list:

1). get() - It returns element from the list at any given position.
2) insert() - Insert() any element at any positio -n of the list
3) Remove() - Remove() the first occurance of any element.
4) Replace() - Replace an element at any position.
5) size() - It returns the number of elements in a list.
6) is empty() - It returns tree if the list is empty.
7) Is full() - It returns true if the list is full. Otherwise it returns false.
8) RemoveAt() - Removes an element at a specific location from an non empty list.

Stack ADT:

Instead of data being stored in each node the pointer to data is stored. The program allocates memory for the data and address is passed to the stack ADT.

The following operations can be performed:

1. push ( ) -
Inside an element at one end of the stack called top

2. pop ( ) -
Remove and returns the element at the top of the stack if it is not empty.

3. Size ( ) -
It returns the number of elements in the stack.

4. is empty ( ) -
It returns true if the stack is empty otherwise it returns false.

5. is full ( ) -
It returns true if the stack is full otherwise returns false.

Queue ADT:

The Queue ADT follows basic design of stack ADT each note contains a pointer to the data and link pointer to the next element in the Queue. The following operations performed are

1. Enqueue:
Insert an element at the end of the queue

2. Dequeue:
Remove and return the first element of the queue.

3. Size:
It returns the number of elements in a queue

4. is empty ( ): it returns true if the queue is empty

otherwise it returns false.

5) is full():
It returns true if the ~~stack~~ queue is full otherwise returns false.

4/3

WEEK 2.

Algorithm Analysis

Alg are the set of instructions to get the expected output.

The algorithm can be analyzed in two levels:

1) Priori Analysis 2) Posterior Analysis

1) Priori Analysis:

It is a theoritical analysis of an algorithm which is done before implementing the algorithm

2) Posterior:

It is a practical analysis achieved by implementing the algorithm using any programming language. There are two techniques to measure the efficiency of an algorithm

1. space complexity:

An algorithm's space complexity is the amount of space required to solve a problem and produce an output

It is expressed in big 'O' notation

space complexity = auxiliary space + input size

where auxiliary space is extra space required by algorithm excluding input size. where input size may be of a space required for program, constant value, variable values, function calls, statements etc.