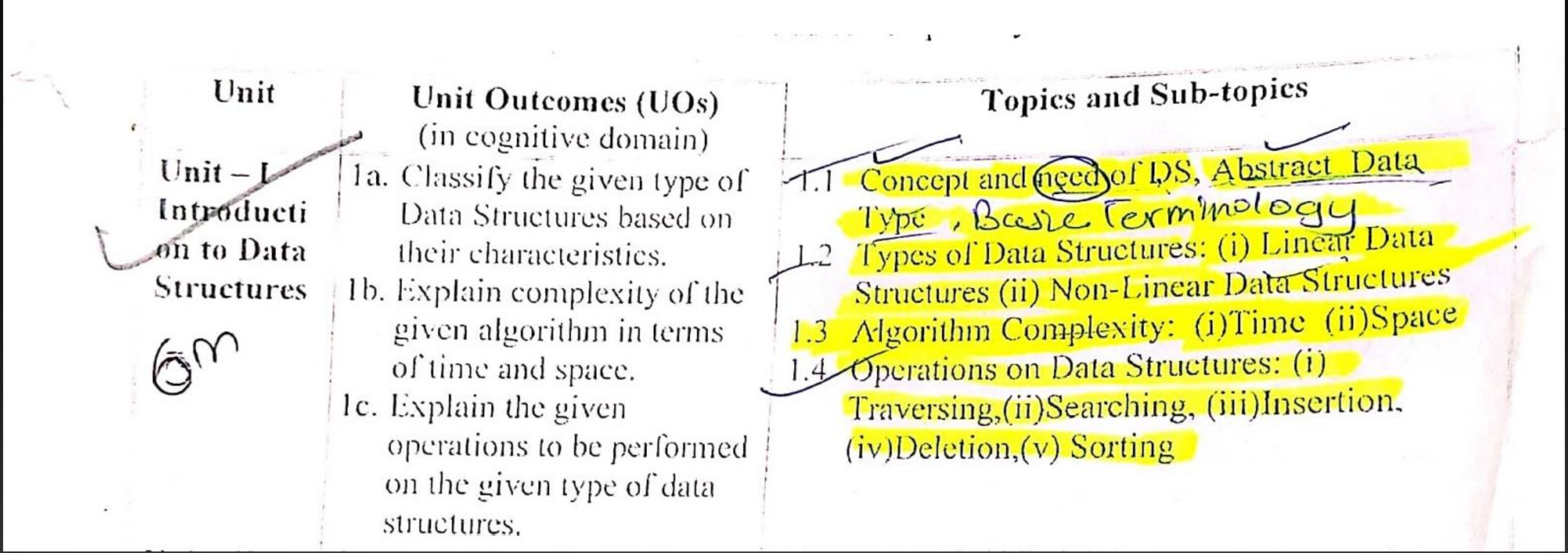
Pranjal Save (SY-COMPS)

DATA STRUCTURE USING ‘C’

UNIT 1 : INTRODUCTION TO DATA STRUCTURES

(WEIGHTAGE MARKS -6MARKS)



Unit	Unit Outcomes (UOs) (in cognitive domain)	Topics and Sub-topics
Unit – I Introduction to Data Structures 6M	1a. Classify the given type of Data Structures based on their characteristics. 1b. Explain complexity of the given algorithm in terms of time and space. 1c. Explain the given operations to be performed on the given type of data structures.	1.1 Concept and need of DS, Abstract Data Type , Basic Terminology 1.2 Types of Data Structures: (i) Linear Data Structures (ii) Non-Linear Data Structures 1.3 Algorithm Complexity: (i)Time (ii)Space 1.4 Operations on Data Structures: (i) Traversing,(ii)Searching, (iii)Insertion, (iv)Deletion,(v) Sorting

Pranjal Save (SY-COMPS)

CONCEPT OF DATA STRUCTURE

- Data Structure can be defined **as the group of data elements which provides an efficient way of storing and organizing data in the computer so that it can be used efficiently.**
- Some examples of Data Structures are arrays, Linked List, Stack, Queue, etc.
- Data Structures are widely used in almost every aspect 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.
- It plays a vital role in enhancing the performance of a software or a program as the main function of the software is to store and retrieve the user's data as fast as possible.

BASIC TERMINOLOGIES

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

Data:

File:

Group Items

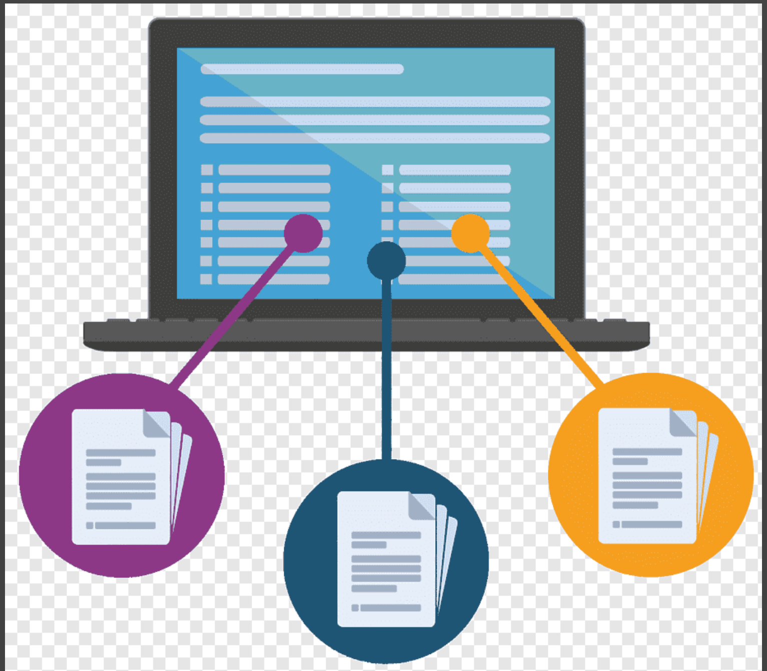
Attribute and Entity:

Record:

Field:

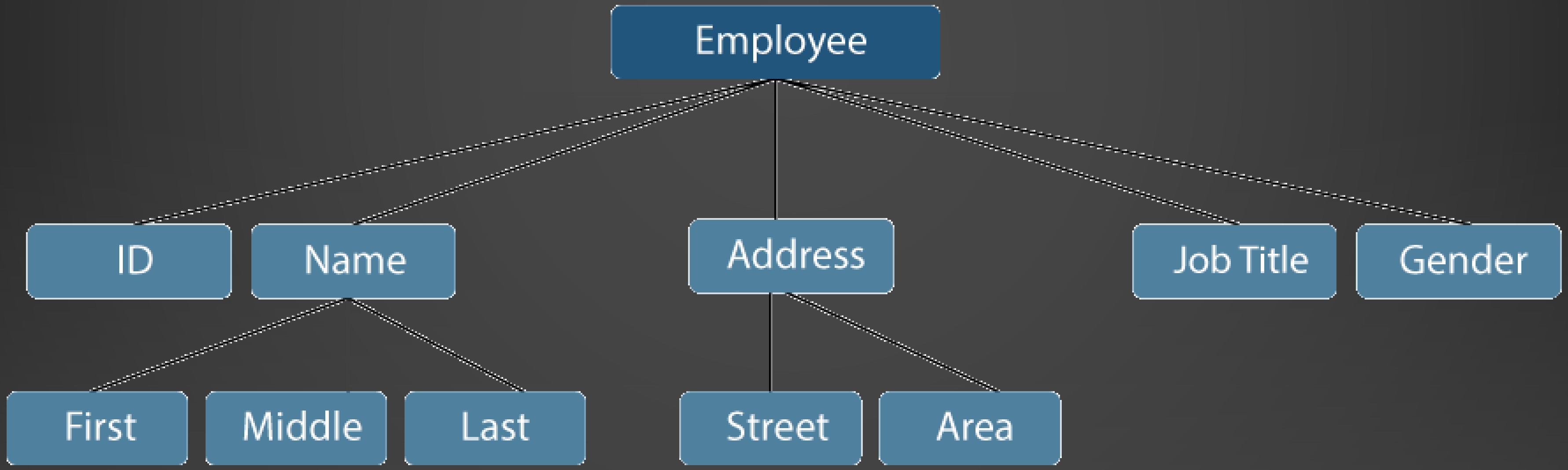
DATA

- **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.**



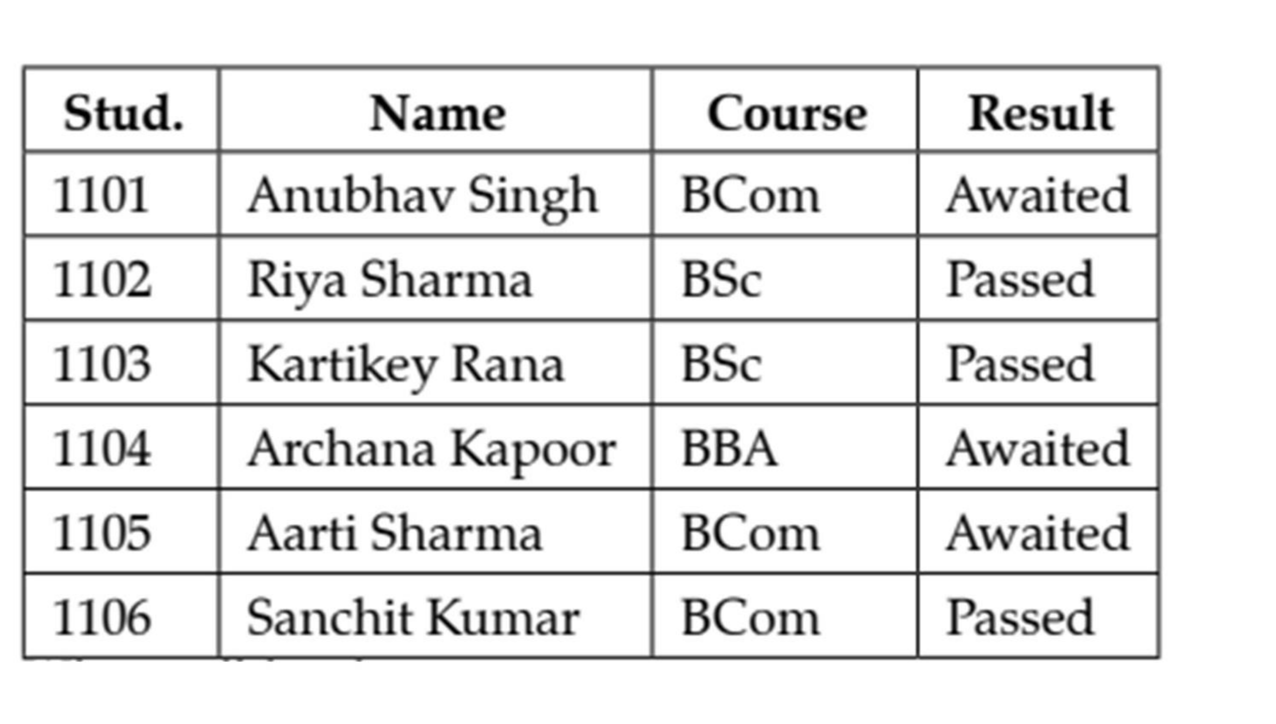
GROUP ITEMS

- **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.**



RECORD

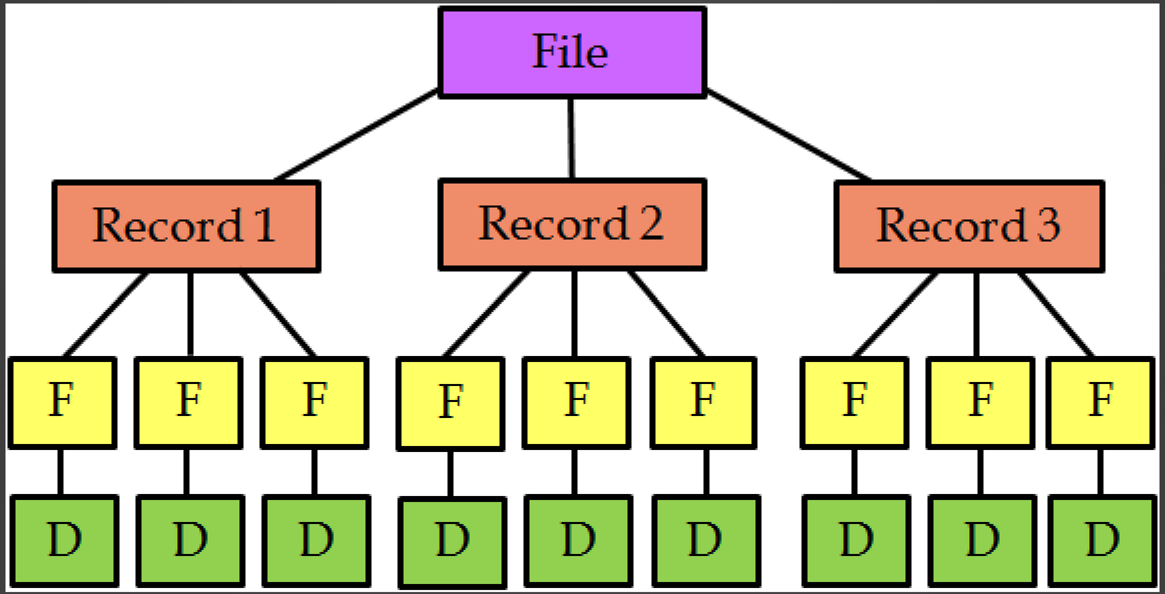
- **Record can be defined as the collection of various data items, for example, if we talk about the student entity, then its name, address, course and marks can be grouped together to form the record for the student.**



Stud.	Name	Course	Result
1101	Anubhav Singh	BCom	Awaited
1102	Riya Sharma	BSc	Passed
1103	Kartikey Rana	BSc	Passed
1104	Archana Kapoor	BBA	Awaited
1105	Aarti Sharma	BCom	Awaited
1106	Sanchit Kumar	BCom	Passed

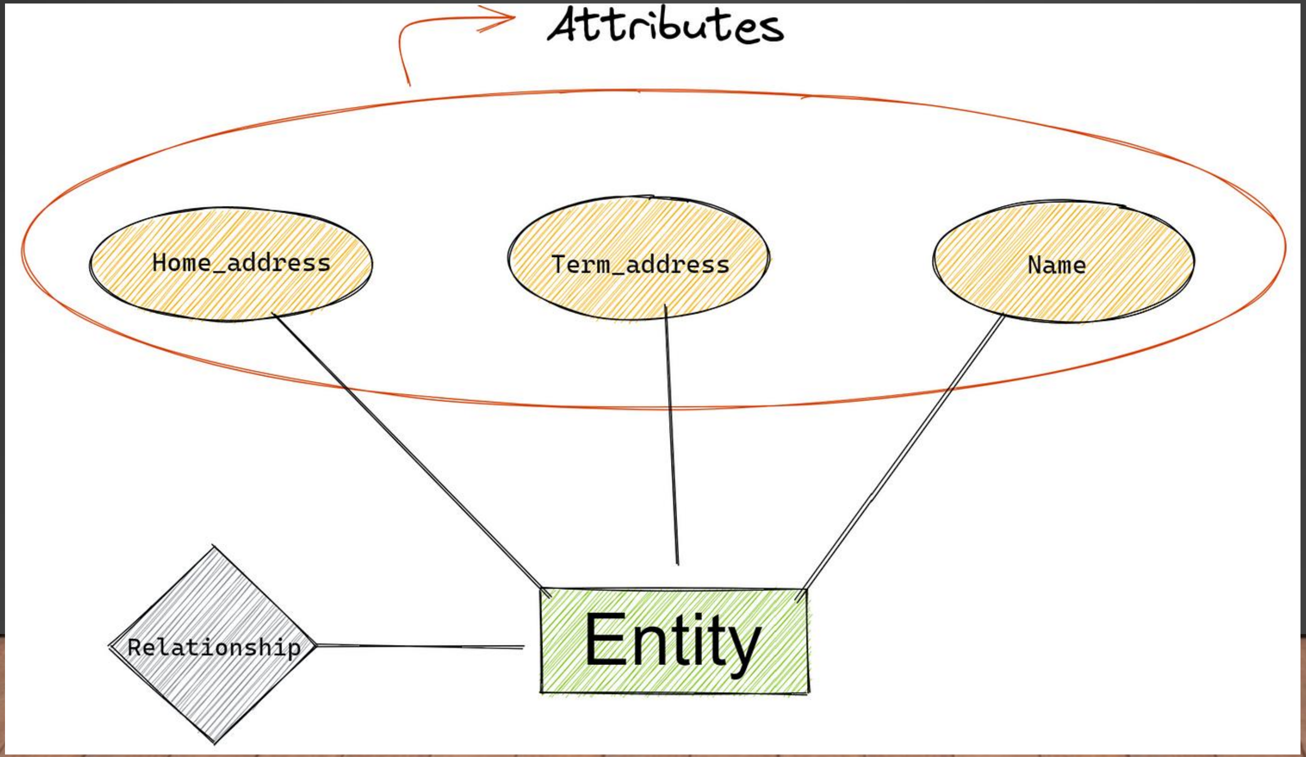
FILE

- : A File is a collection of various records of one type of entity, for example, if there are 60 employees in the class, then there will be 20 records in the related file where each record contains the data about each employee.



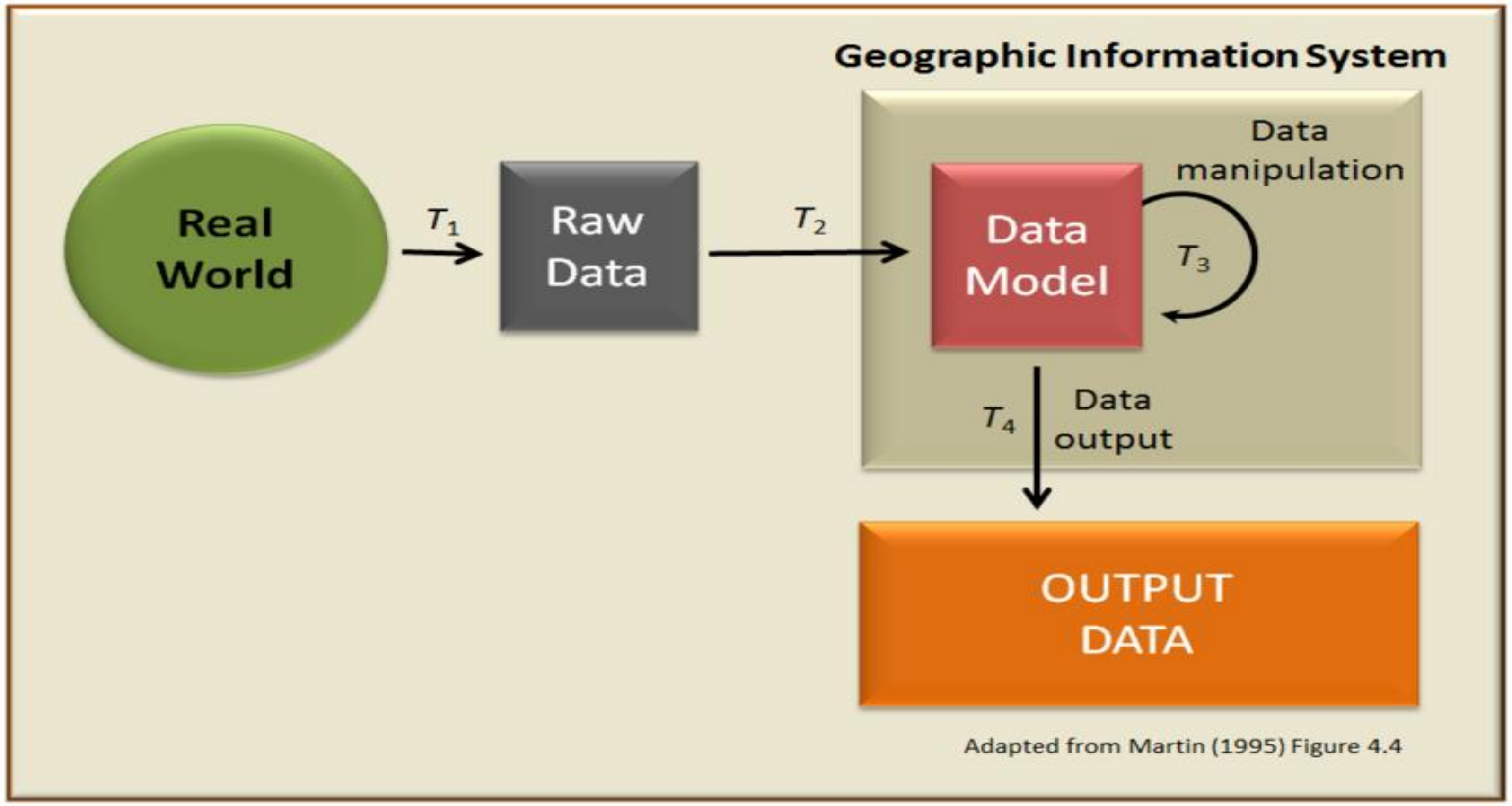
ATTRIBUTE AND ENTITY

- : 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.**



Adapted from Martin (1995) Figure 4.4

NEED OF DATA STRUCTURE

As applications are getting complexes and amount of data is increasing day by day, there may arise the following 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.
- **Data Search:** Consider an inventory size of 106 items in a store, if our application needs to search for a particular item, it needs to traverse 106 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 in order to solve the above problems, data structures are used. Data is organized to form a data structure in such a way that all items are not required to be searched and required data can be searched instantly.
- **Efficient Data Management:** Data structure help in storing and accessing data in a manner that improves efficiency of operations like search , insertion and deletion.
- **Optimized Memory Usage:** By utilizing appropriate data structures you can minimize memory wastage and make the most out of available memory.
- **Improved Algorithm Design:** Many algorithms rely on specific data structure to perform optimally.
- **Easy Maintenance:** Code using appropriate data structures tends to be more modular and easier .

Problem solving
Handling large data sets.

ADVANTAGES OF DATA STRUCTURES

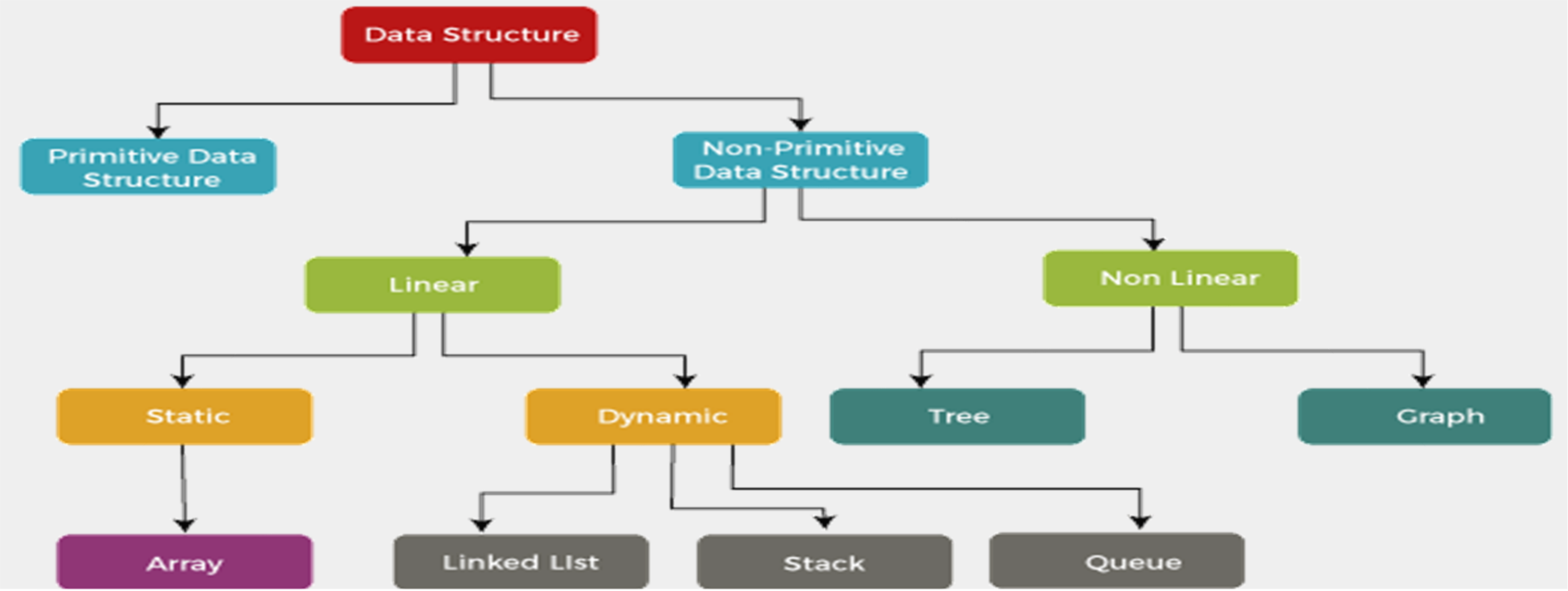
- **Reusability:** Data structures are reusable, i.e. once we have implemented a particular data structure, we can use it at any other place. Implementation of data structures can be compiled into libraries which can be used by different clients.

- **Abstraction:** Data structure is specified by the ADT which provides a level of abstraction. The client program uses the data structure through interface only, without getting into the implementation details.

- Efficiently ; Efficiency of a program depends upon the choice of data structures.
- Flexibility ; It allows developers to choose the most appropriate data structures for specific problem.
- Modularity and Maintainibilty ; By seprating store , manipulation , structures of data to promote modularity in designing and results in mainability.
- Memory optimization ; Data structures is efficient memory management ensures that memory resources are utlizies effectively.

Pranjal Save (SY-COMPS)

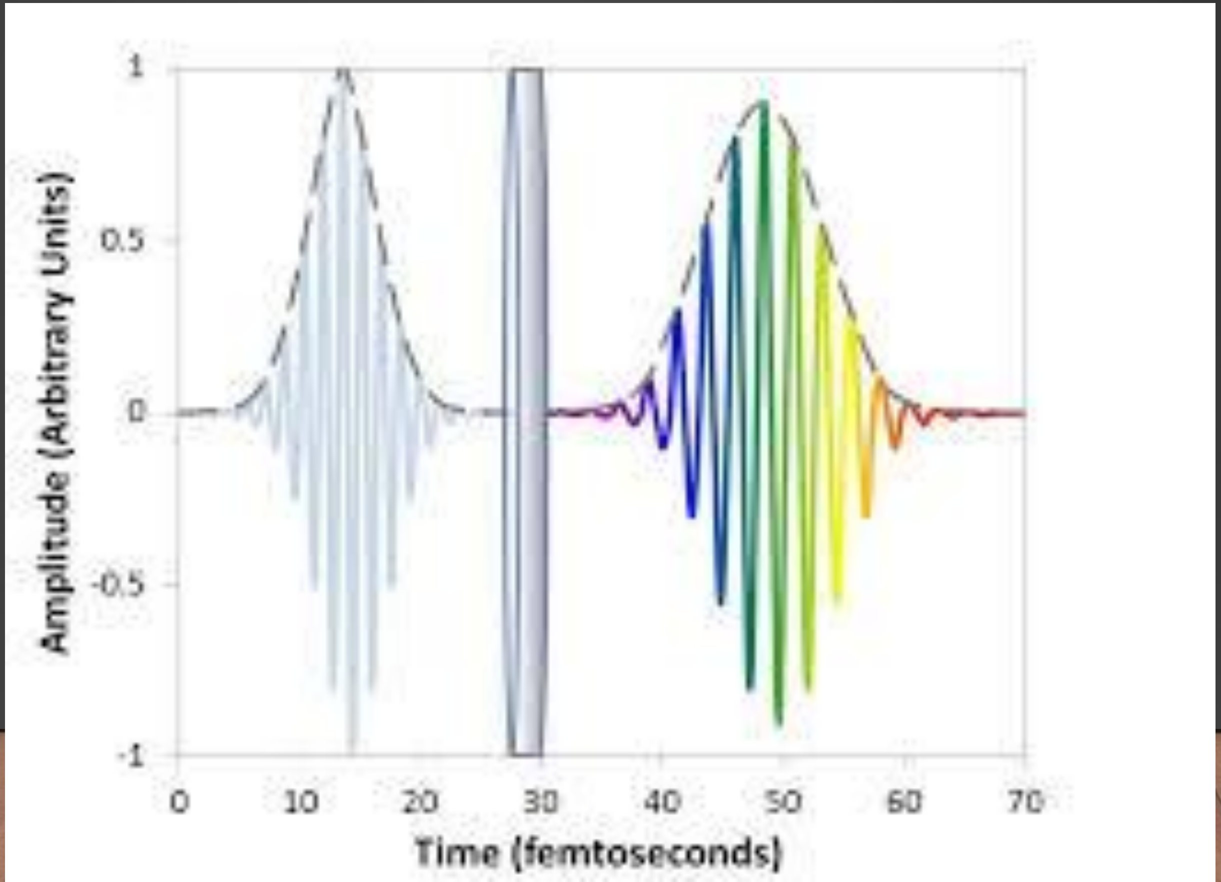
Data Structure Classification



Primitive types are predefined (already defined) i Non-primitive types are created by the programmer and is not defined by (except for String).

NON LINEAR DATA STRUCTURE

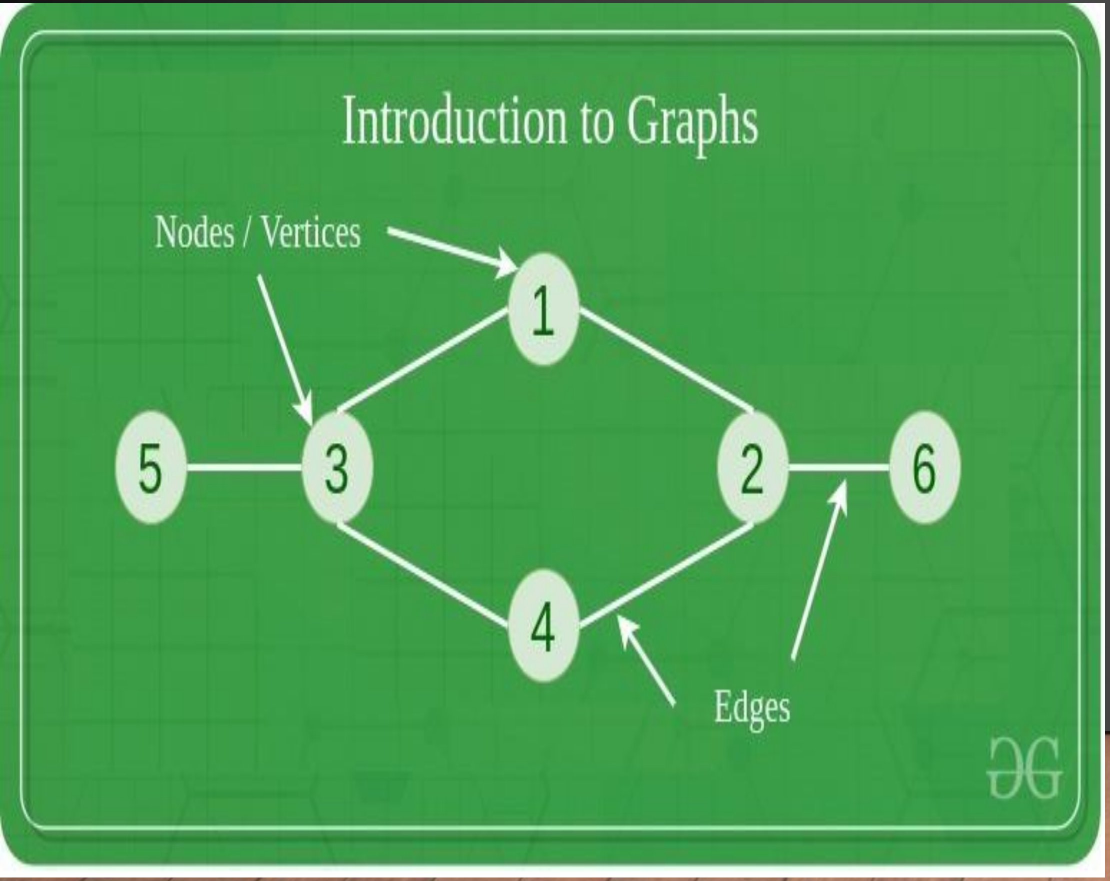
- **Non Linear Data Structures:** This data structure does not form a sequence i.e. each item or element is connected with two or more other items in a non-linear arrangement. The data elements are not arranged in sequential structure.



Pranjal Save (SY-COMPS)

GRAPHS

- A Graph is a non-linear data structure consisting of vertices and edges. The vertices are sometimes also referred to as nodes and the edges are lines or arcs that connect any two nodes in the graph. More formally a Graph is composed of a set of vertices(V) and a set of edges(E). The graph is denoted by G(E, V).a



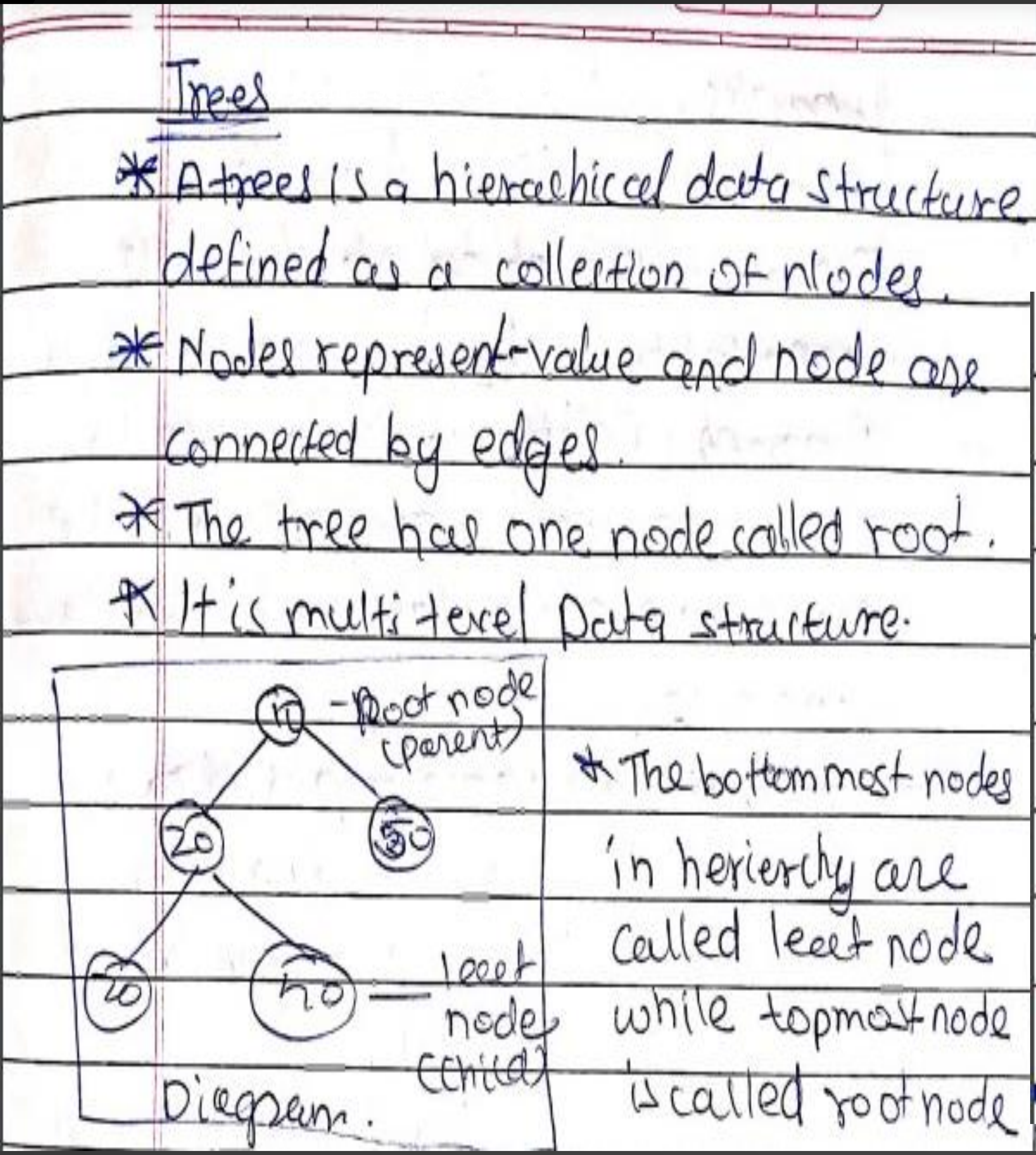
Components of a Graph

•**Vertices:** Vertices are the fundamental units of the graph. Sometimes, vertices are also known as vertex or nodes. Every node/vertex can be labeled or unlabelled.

•**Edges:** Edges are drawn or used to connect two nodes of the graph. It can be ordered pair of nodes in a directed graph. Edges can connect any two nodes in any possible way. There are no rules. Sometimes, edges are also known as arcs. Every edge can be labeled/unlabelled.

Graphs are used to solve many real-life problems. Graphs are used to represent networks. The networks may include paths in a city or telephone network or circuit network.

TREES



Trees

* A trees is a hierachical data structure defined as a collection of nlodes.

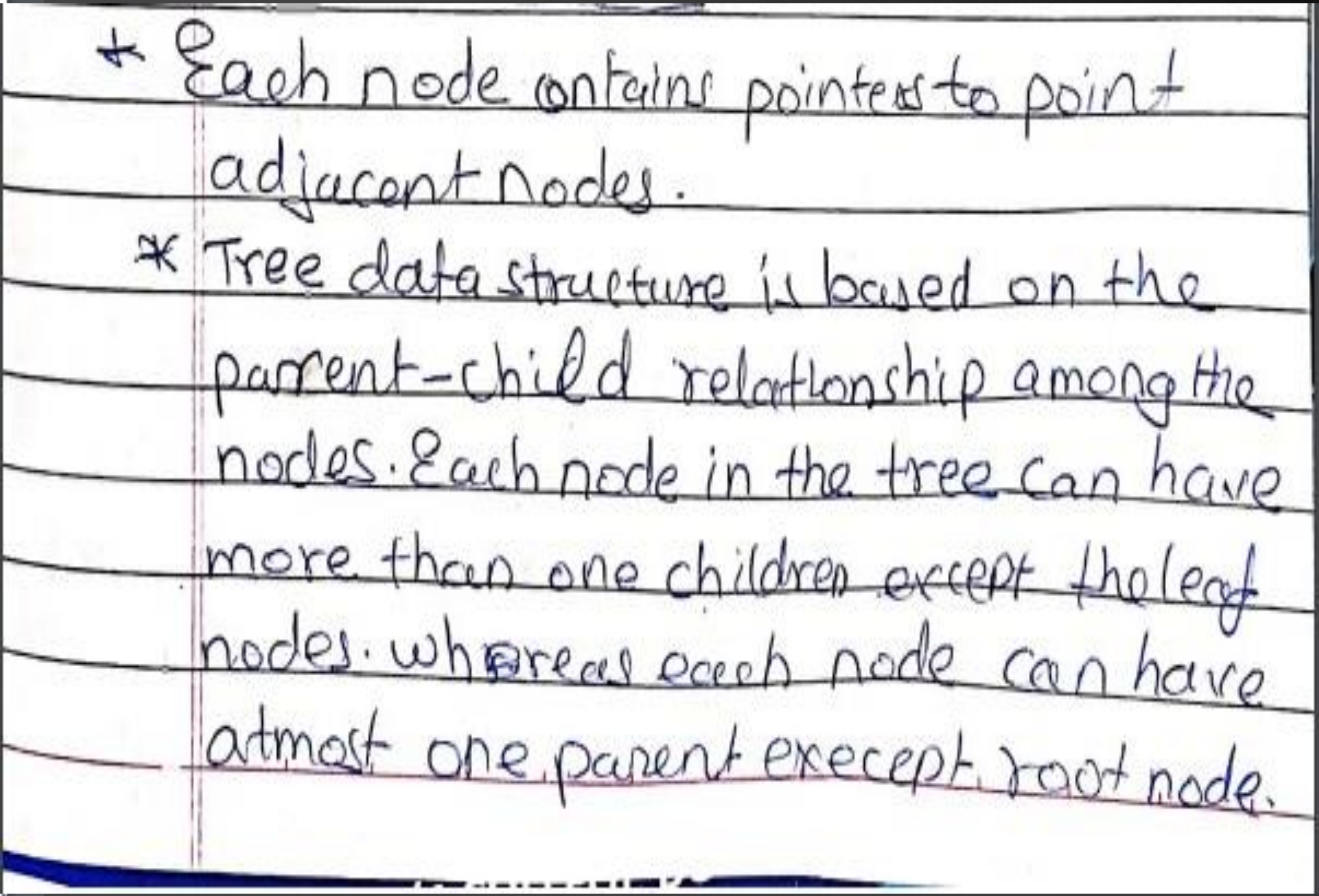
* Nodes represent value and node are connected by edges.

* The tree has one node called root.

* It is multi level Data structure.

10
- Root node (parent)
20
50
30
40
— leaf node (child)
Diagram.

* The bottommost nodes in herierchy are called leaf node while topmost node is called root node



* Each node contains pointers to point adjacent nodes.

* Tree data structure is based on the parent-child relationship among the nodes. Each node in the tree can have more than one children except the leaf nodes. whereas each node can have atmost one parent except root node.

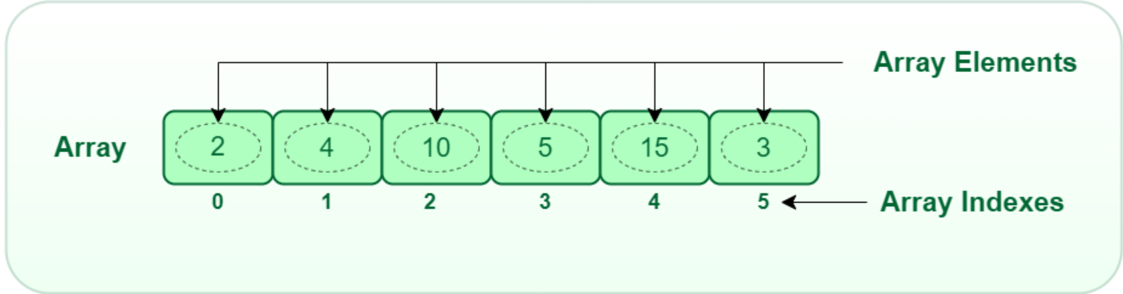
LINEAR DATA STRUCTURES

- Linear data structures are data structures in which data elements are stored in a linear sequence.

Hospital Emergency Queue
The person who requires quick medical attention will be treated immediately!

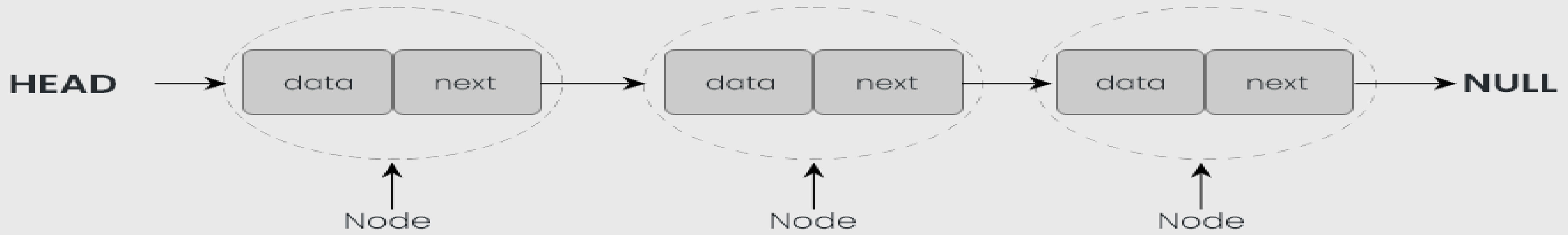
ARRAY

- An array is a collection of similar data elements stored at contiguous memory locations.
- It is the simplest data structure where each data element can be accessed directly by only using its index number.
- Array can be of any data type int , float double & char.
- There are types of array – 1-D,2-D, Multi-D.
- Syntax- data type variable[size].(1D)
- 2-d a[3][4]…



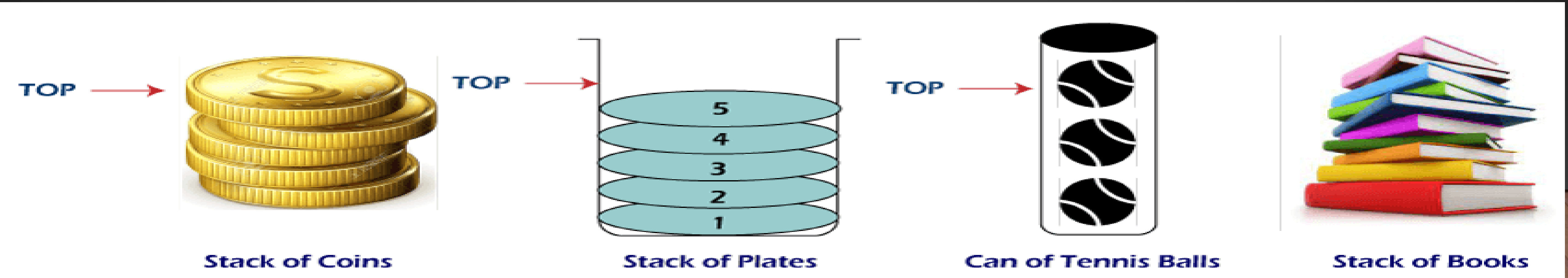
LINKED LISTS

- **:** 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.
- *The elements in a linked list are linked using pointers as shown in the below image:*
- In simple words, a linked list consists of nodes where each node contains a data field and a reference(link) to the next node in the list.



STACKS

- 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 (ADT), 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.
- A stack is a linear data structure that follows the principle of Last In First Out (LIFO). This means the last element inserted inside the stack is removed first.

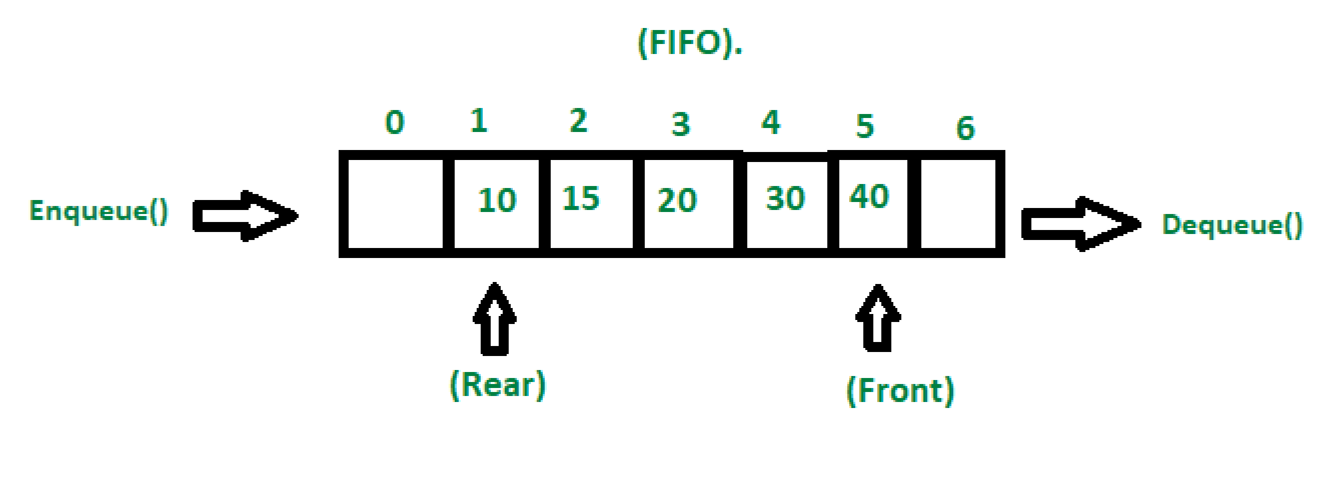


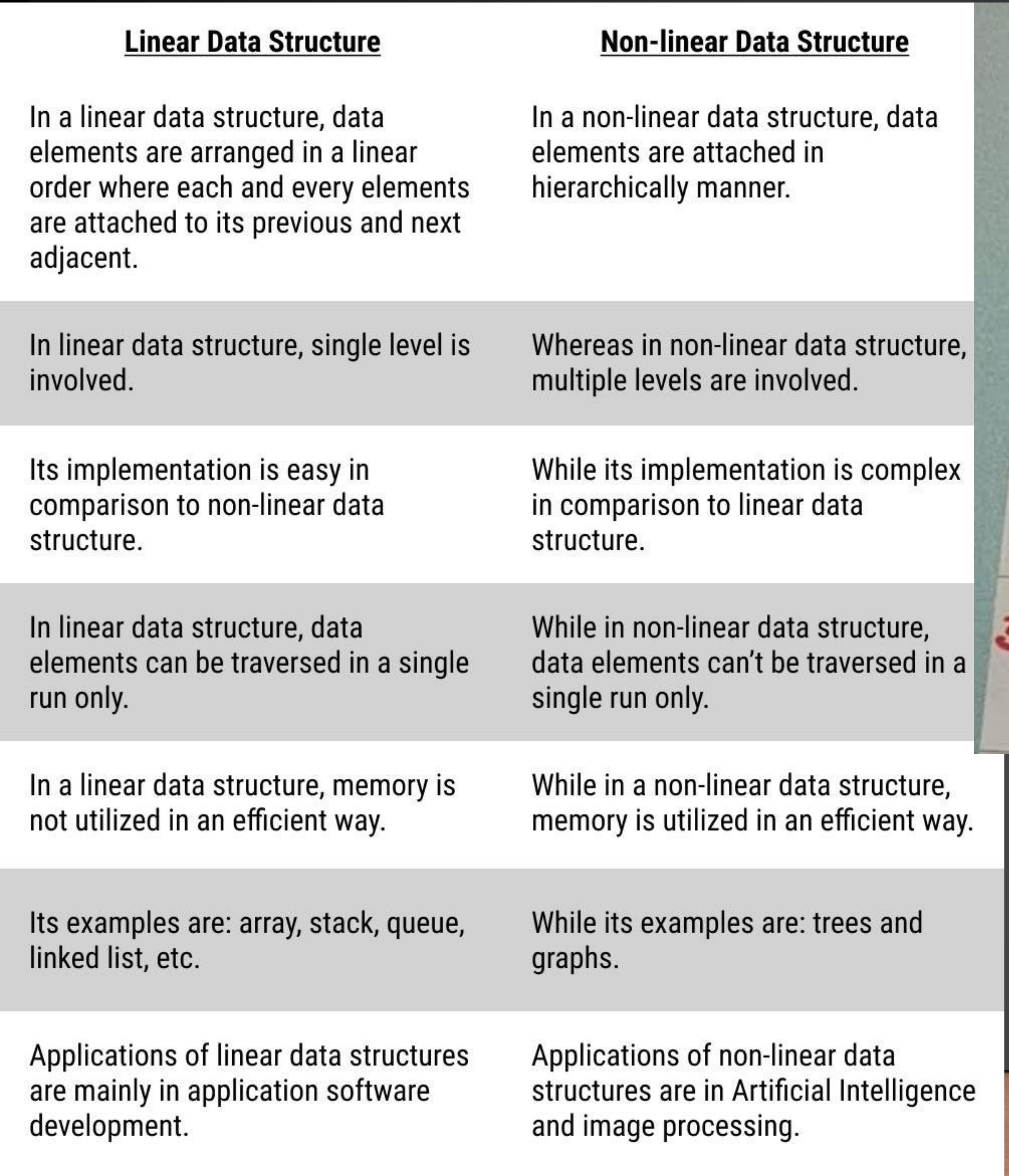
QUEUE

Queue is a linear list in which elements can be inserted only at one end called **rear** and deleted only at the other end called **front**.

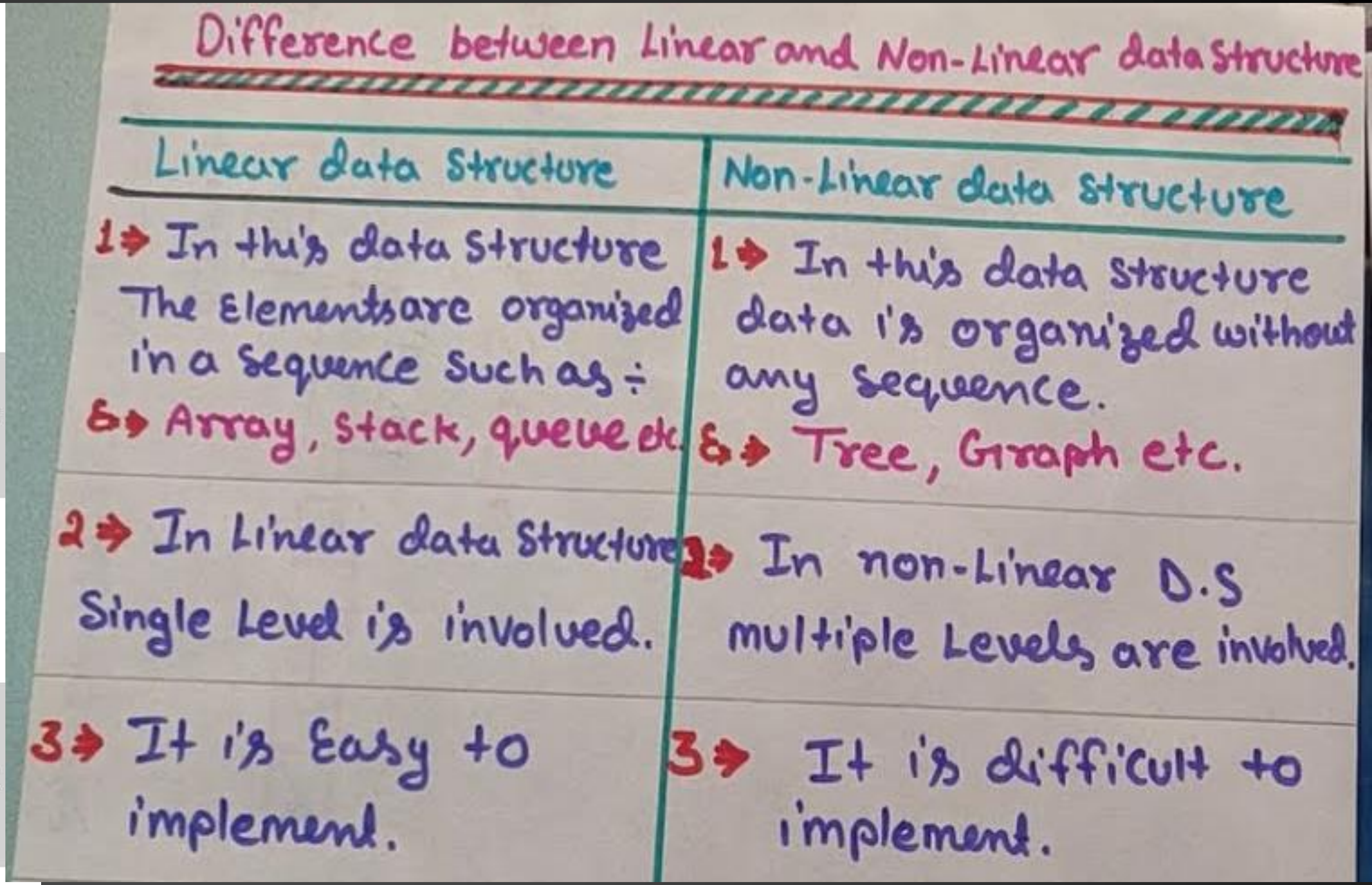
It is an abstract data structure, similar to stack. Queue is opened at both end therefore it follows First-In-First-Out (FIFO) methodology for storing the data items.

The data is inserted into the queue through one end and deleted from it using the other end.



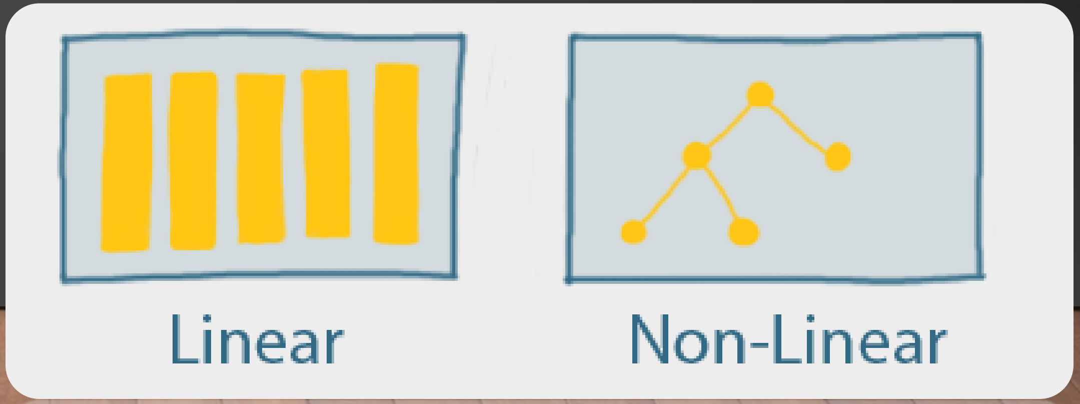


Linear Data Structure	Non-linear Data Structure
In a linear data structure, data elements are arranged in a linear order where each and every elements are attached to its previous and next adjacent.	In a non-linear data structure, data elements are attached in hierarchically manner.
In linear data structure, single level is involved.	Whereas in non-linear data structure, multiple levels are involved.
Its implementation is easy in comparison to non-linear data structure.	While its implementation is complex in comparison to linear data structure.
In linear data structure, data elements can be traversed in a single run only.	While in non-linear data structure, data elements can't be traversed in a single run only.
In a linear data structure, memory is not utilized in an efficient way.	While in a non-linear data structure, memory is utilized in an efficient way.
Its examples are: array, stack, queue, linked list, etc.	While its examples are: trees and graphs.
Applications of linear data structures are mainly in application software development.	Applications of non-linear data structures are in Artificial Intelligence and image processing.



Difference between Linear and Non-Linear data Structure

Linear data Structure	Non-Linear data structure
1➔ In this data Structure The Elements are organized in a Sequence Such as ÷ &➔ Array, Stack, queue etc.	1➔ In this data structure data is organized without any sequence. &➔ Tree, Graph etc.
2➔ In Linear data Structure Single Level is involved.	2➔ In non-Linear D.S multiple Levels are involved.
3➔ It is Easy to implement.	3➔ It is difficult to implement.

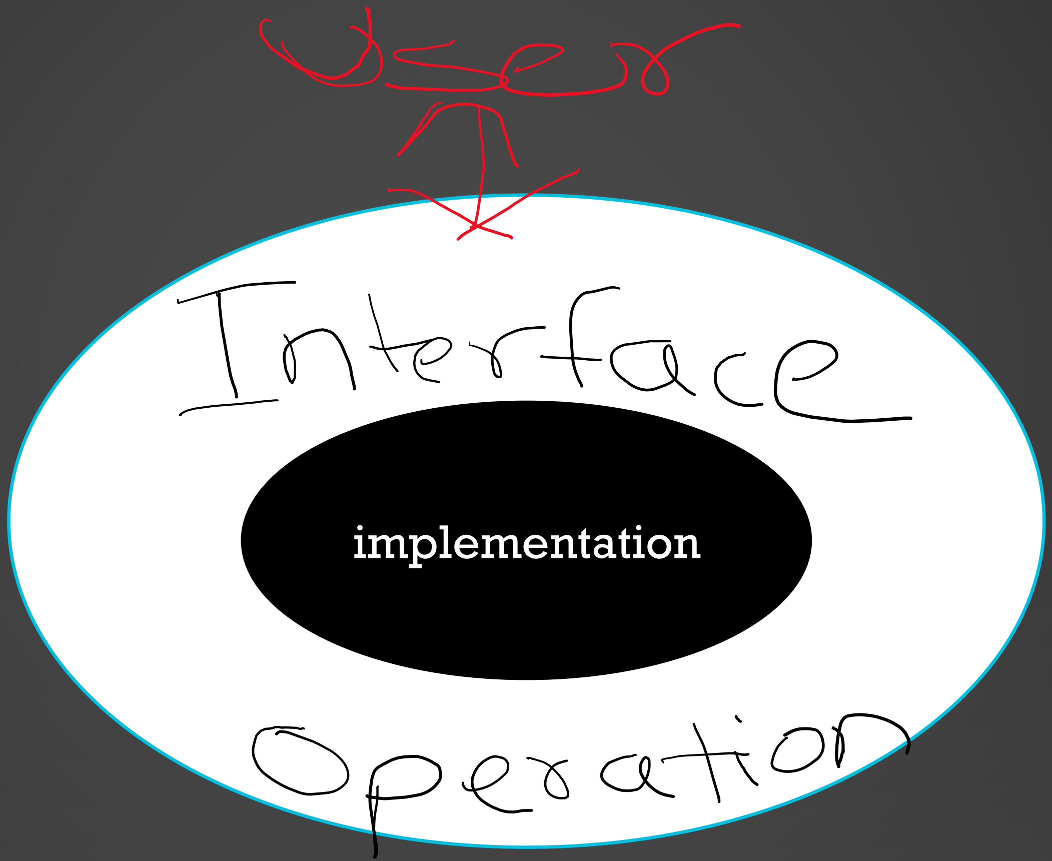


ABSTRACT DATA TYPE

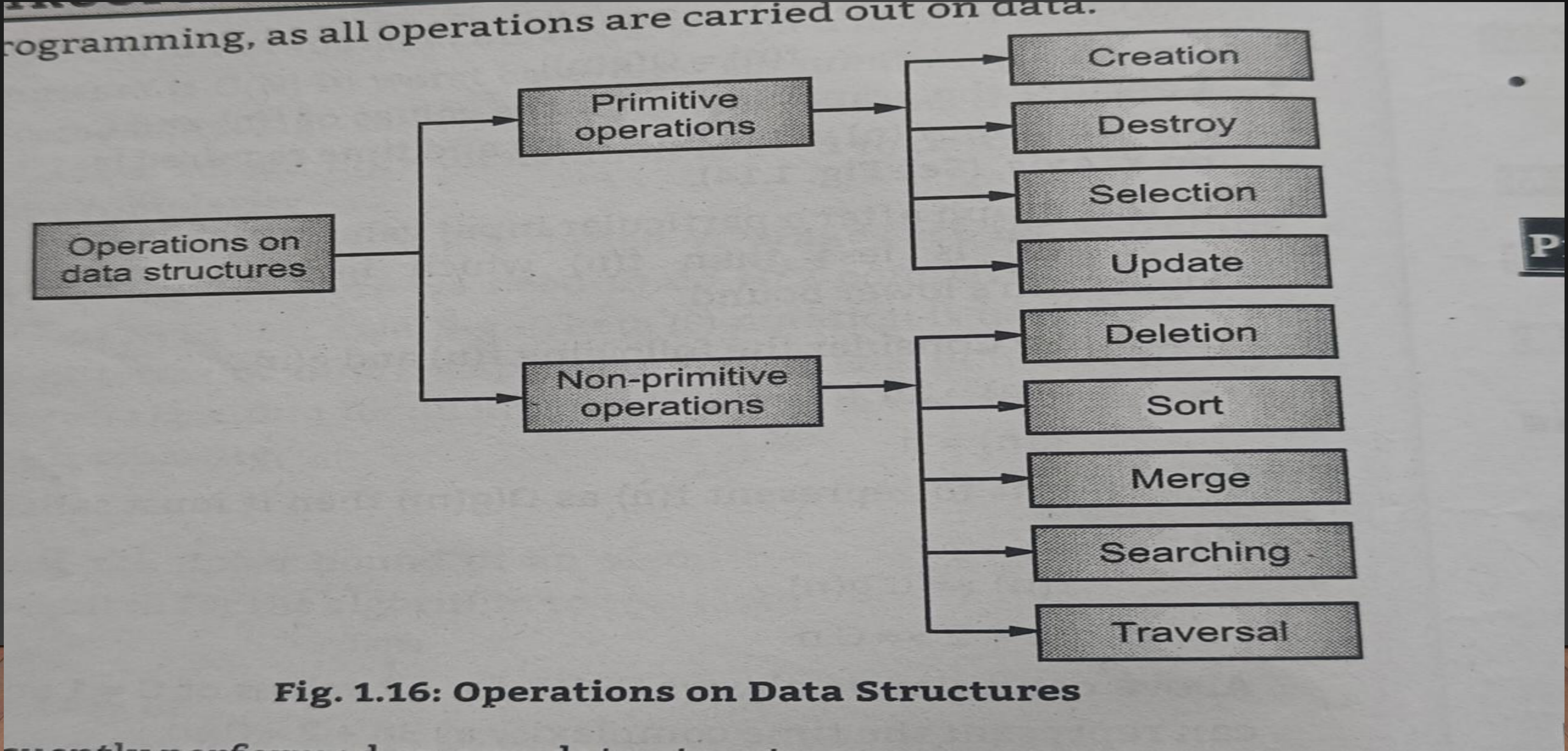
- An abstract data type is an abstraction of a data structure that provides only the interface to which the data structure must adhere.
- The interface does not give any specific details about something should be implemented or in what programming language.
- An abstract data type in data structure is a kind of a data type whose behavior is defined with the help of some attributes and some functions. An abstract data type in data structure can be that of a list data structure, stack data structure and a queue data structure.
- The user of data type does not need to know how that data type is implemented, for example, we have been using Primitive values like int, float, char data types only with the knowledge that these data type can operate and be performed on without any idea of how they are implemented.
- It is a special kind of data type , whose behaviour is defined by a set of values and set of operations .
- The keyword”abstract” is used as we can use these datatypes , we can perform operations.But operations are working that is totally hidden from user.
- Adt is non primitive data types . Operations logics are hidden.
- Examples , stacls queques ect.

Pranjal Save (SY-COMPS)

ADT



OPERATIONS ON DATA STRUCTURE

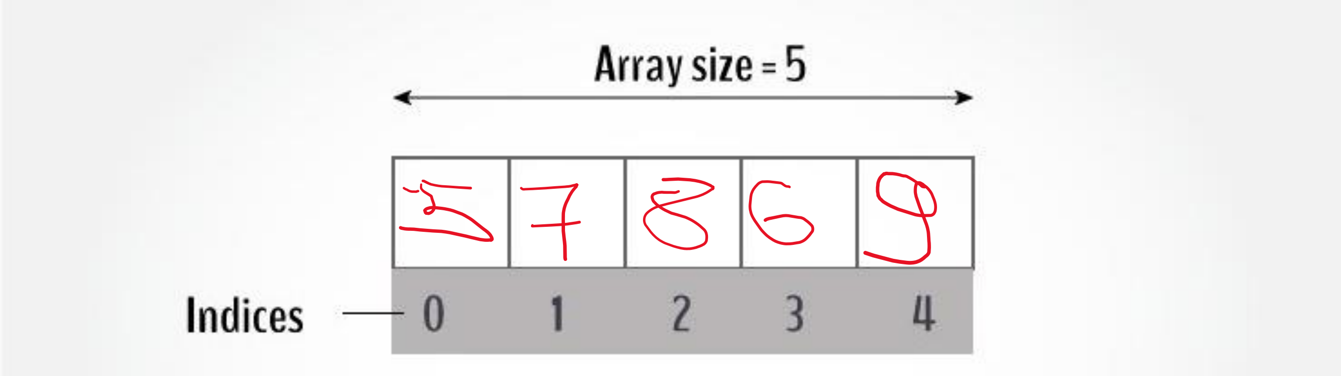


rogramming, as all operations are carried out on data.

Fig. 1.16: Operations on Data Structures

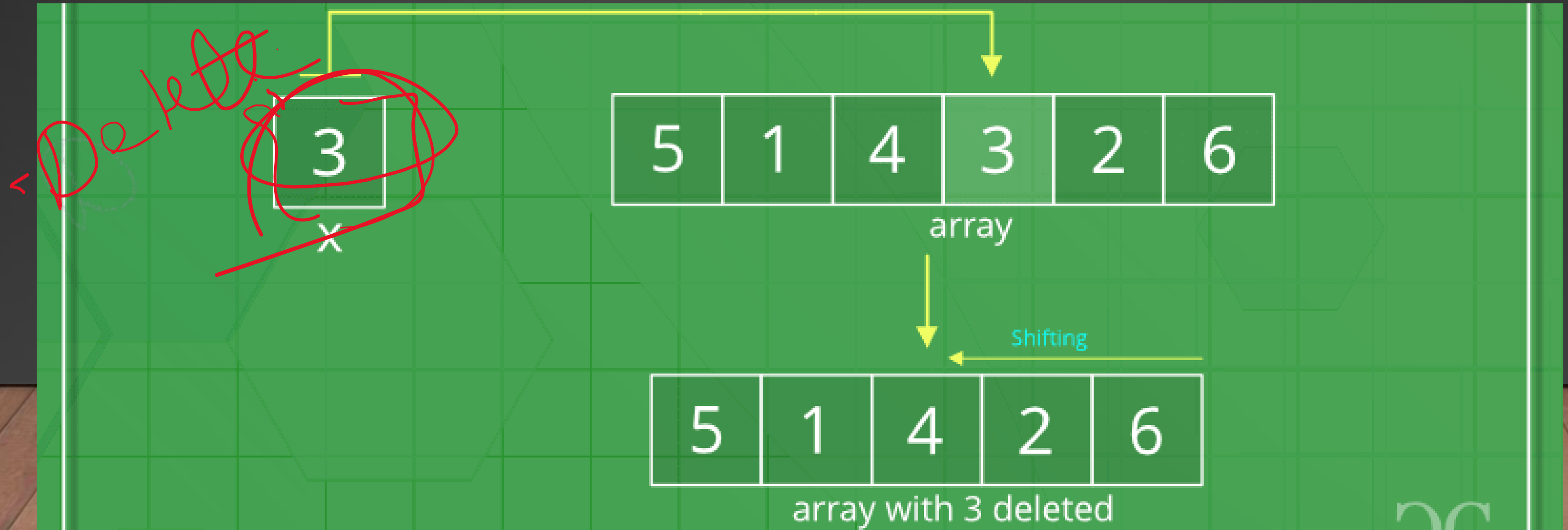
CREATION

- Creation: To create new Data Structure
- To create a new data structure using data types in system.
- For example, int a[5]



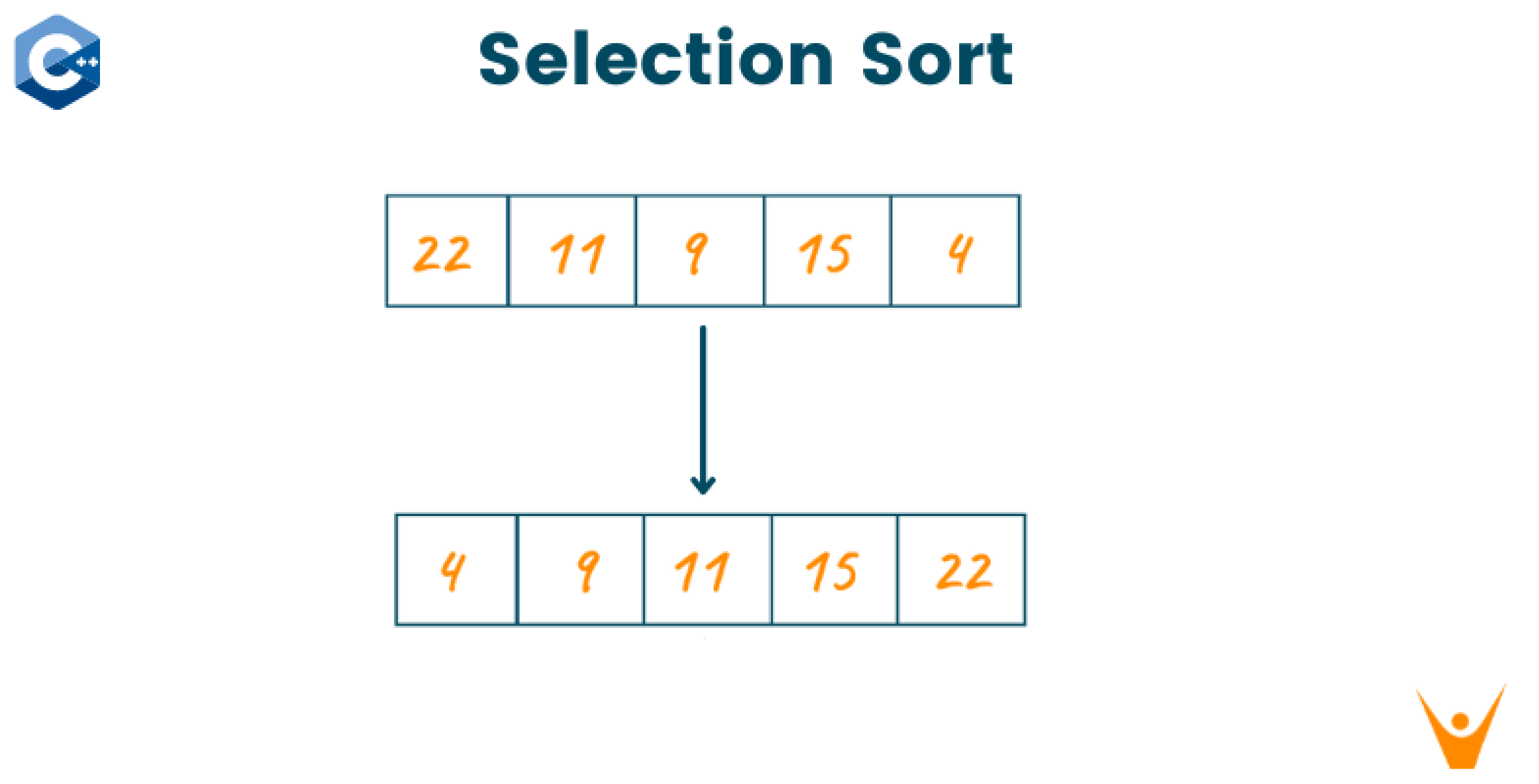
DELETEION

- Remove a data from an data structure.
- Destroy: To delete Data Structure
- The process of removing an element from the data structure is called Deletion. We can delete an element from the data structure at any random location.
- If we try to delete an element from an empty data structure then **underflow** occurs.
- Removing a data from the data structure is referred as deletion



SELECTION

- Selection: To access (select) data from the data structure



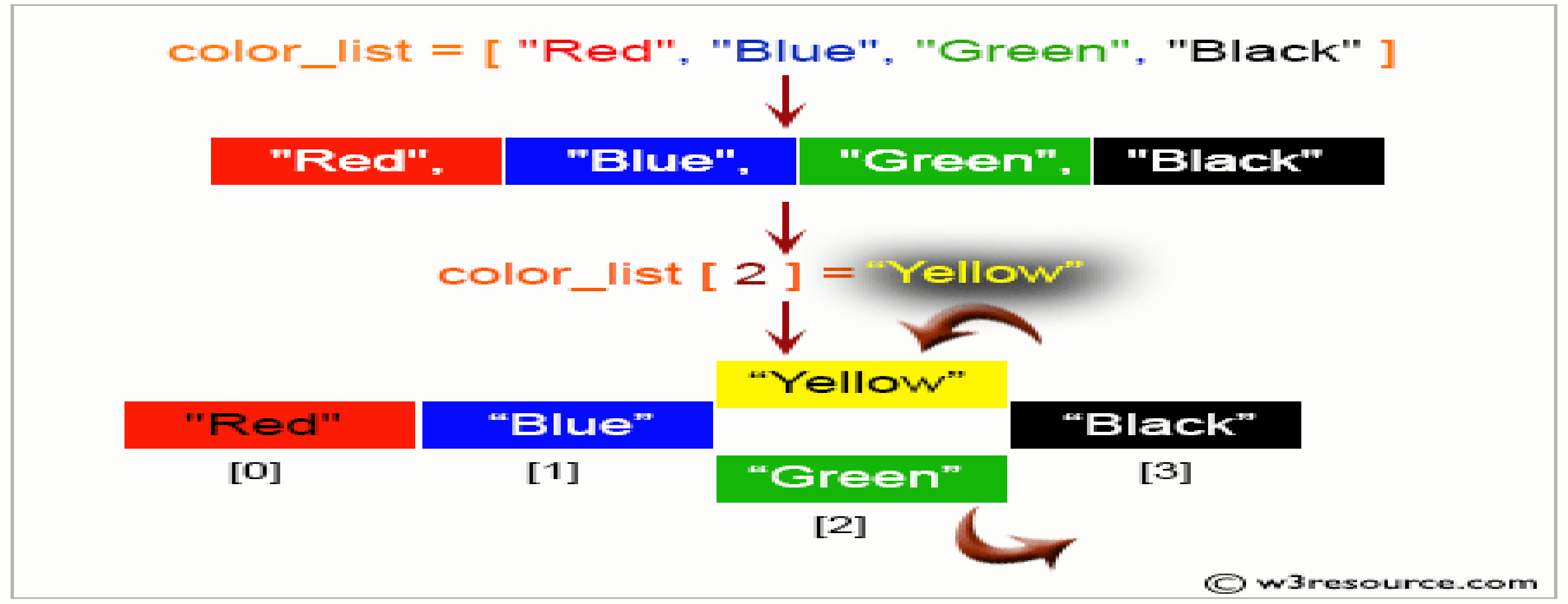
Selection Sort

22	11	9	15	4

4	9	11	15	22

UPDATION

- Updating: To edit or change the data within the data structure
- It is known as minor changes in data structure .

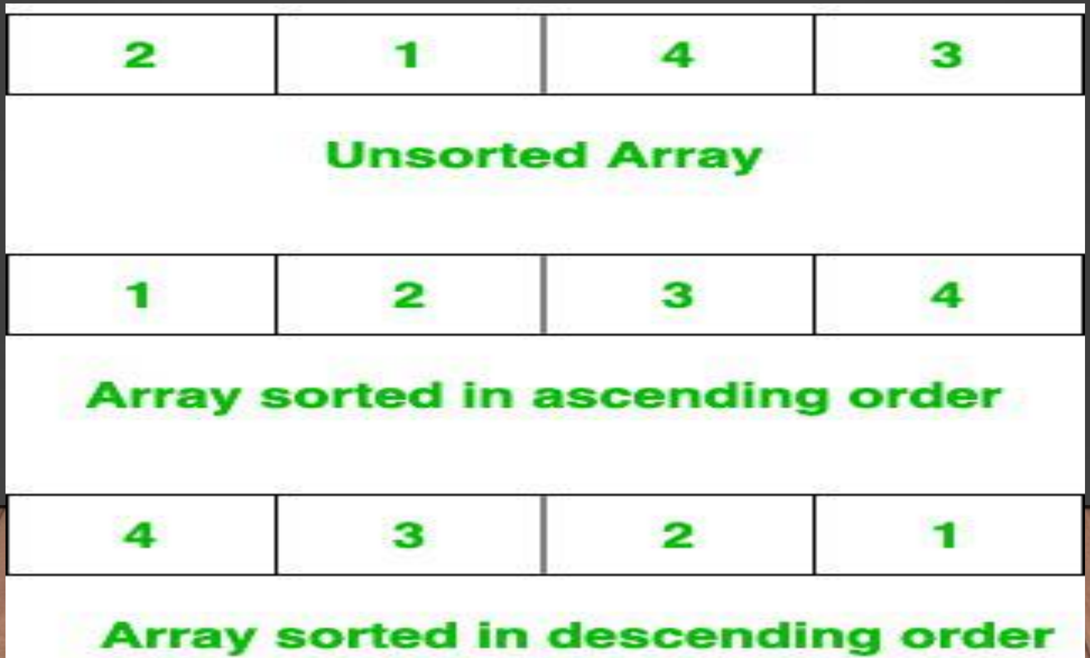


```
color_list = [ "Red", "Blue", "Green", "Black" ]
```

```
color_list [ 2 ] = "Yellow"
```

SORTING

- **Arranging the data in some logical order (ascending or descending, numerically or alphabetically).**
- The process of arranging the data structure in a specific order is known as Sorting. There are many algorithms that can be used to perform sorting, for example, insertion sort, selection sort, bubble sort, etc.



2	1	4	3

Unsorted Array

1	2	3	4

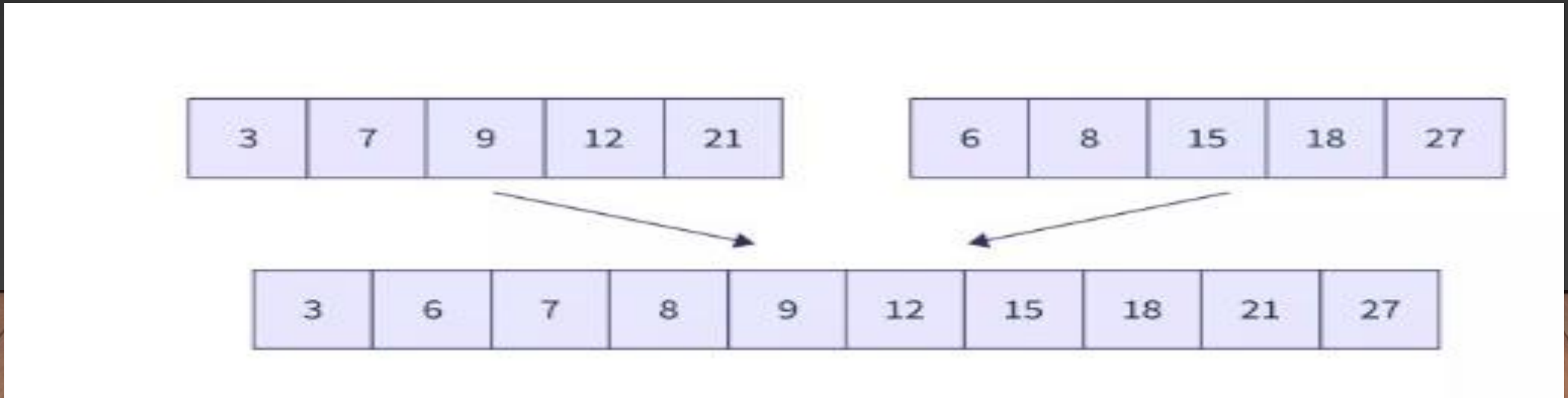
Array sorted in ascending order

4	3	2	1

Array sorted in descending order

MERGING

- When two lists List A and List B of size M and N respectively, of similar type of elements, clubbed or joined to produce the third list, List C of size (M+N), then this process is called merging
- Combining the data of two different sorted files into a single sorted file.
- Combining data elements



3	7	9	12	21

6	8	15	18	27

3	6	7	8	9	12	15	18	21	27

SEARCHING

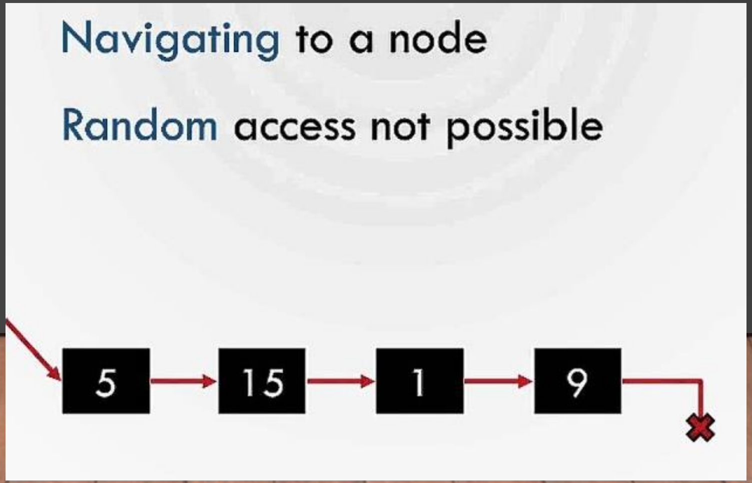
- **Finding the location of data within the data structure which satisfy the searching condition**
- The process of finding the location of an element within the data structure is called Searching. There are two algorithms to perform searching, Linear Search and Binary Search. We will discuss each one of them later in this tutorial
- For example, Find 15 = a[i];

TRAVERSING

- Every data structure contains the set of data elements. Traversing the data structure means visiting each element of the data structure in order to perform some specific operation like searching or sorting.

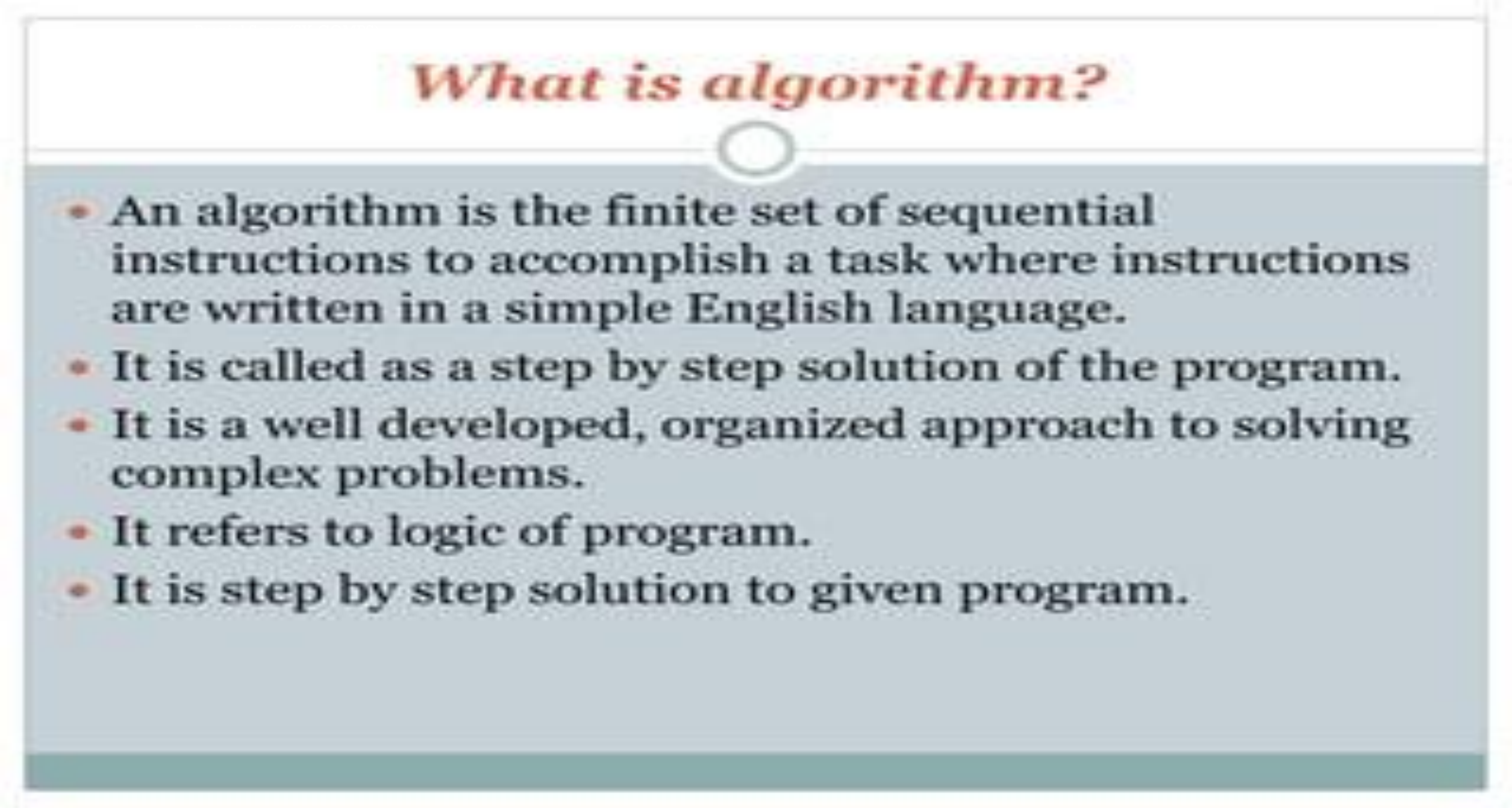
- **Example:** If we need to calculate the average of the marks obtained by a student in 6 different subject, we need to traverse the complete array of marks and calculate the total sum, then we will devide that sum by the number of subjects i.e. 6, in order to find the average.

- Accessing each data exactly once in the data structure so that each data item is traversed or visited.



Navigating to a node

Random access not possible

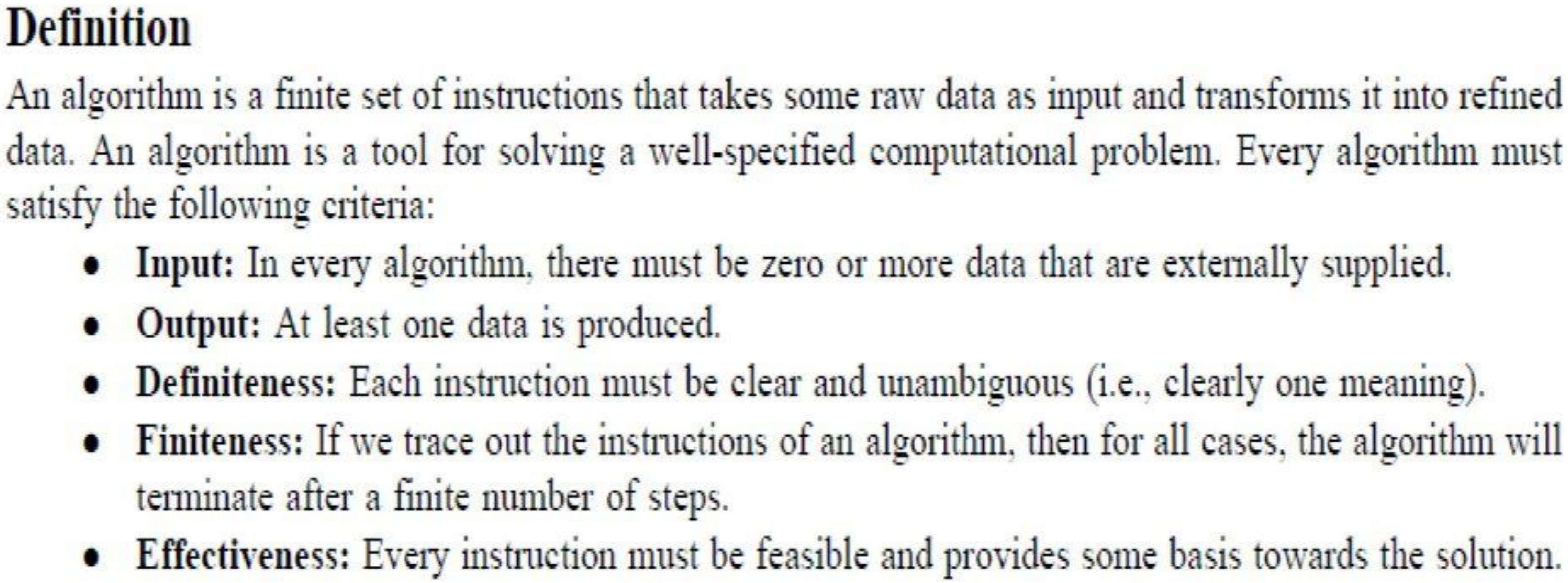


What is algorithm?

- An algorithm is the finite set of sequential instructions to accomplish a task where instructions are written in a simple English language.
- It is called as a step by step solution of the program.
- It is a well developed, organized approach to solving complex problems.
- It refers to logic of program.
- It is step by step solution to given program.

(SY-COMPS)

What is an Algorithm?



Definition

An algorithm is a finite set of instructions that takes some raw data as input and transforms it into refined data. An algorithm is a tool for solving a well-specified computational problem. Every algorithm must satisfy the following criteria:

- **Input:** In every algorithm, there must be zero or more data that are externally supplied.
- **Output:** At least one data is produced.
- **Definiteness:** Each instruction must be clear and unambiguous (i.e., clearly one meaning).
- **Finiteness:** If we trace out the instructions of an algorithm, then for all cases, the algorithm will terminate after a finite number of steps.
- **Effectiveness:** Every instruction must be feasible and provides some basis towards the solution.

Pranjal Save (SY-COMPS)

ALGORITHM COMPLEXITY

Algorithm complexity measures how many steps are required by the algorithm to solve the given problem.

It evaluates the order of count of operations executed by an algorithm as a function of input data size.

The complexity can be found in any form such as constant, logarithmic, linear, n*log(n), quadratic, cubic, exponential, etc.

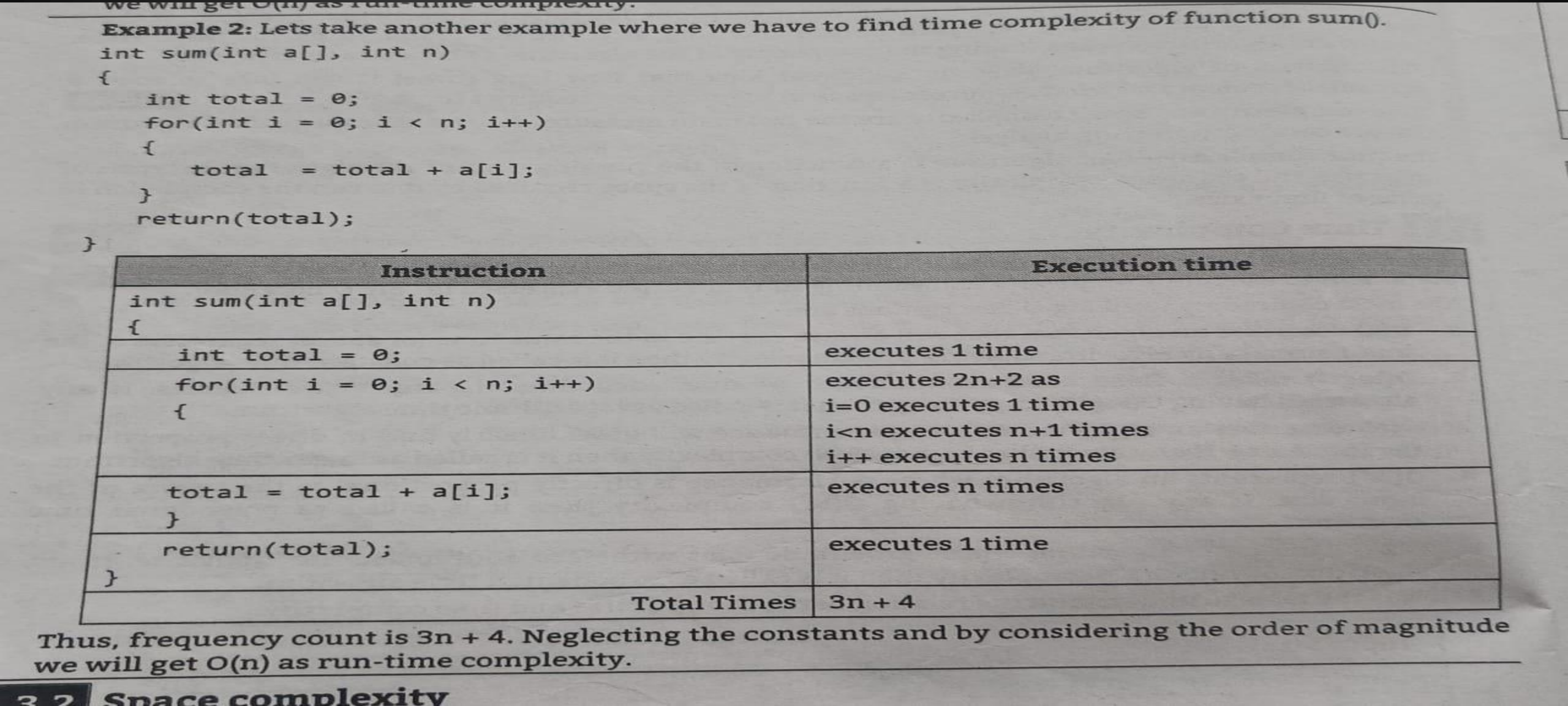
The efficiency of an algorithm depends on two parameters:

- **Time Complexity**
- **Space Complexity**

Time Complexity: It is defined as the number of times a particular instruction set is executed rather than the total time is taken. It is because the total time took also depends on some external factors like the compiler used, processor's speed, etc.

Space Complexity: It is the total memory space required by the program for its execution.

TIME COMPLEXITY EXAMPLE



we will get O(n) as run-time complexity.

Example 2: Lets take another example where we have to find time complexity of function sum().

```
int sum(int a[], int n)
{
   int total = 0;
   for(int i = 0; i < n; i++)
   {
      total  = total + a[i];
   }
   return(total);
}
```

Instruction	Execution time

`int sum(int a[], int n)`
`{`	

`int total = 0;`	executes 1 time

`for(int i = 0; i < n; i++)`
`{`
	executes 2n+2 as
	i=0 executes 1 time
	i<n executes n+1 times
	i++ executes n times

`total = total + a[i];`
`}`
	executes n times

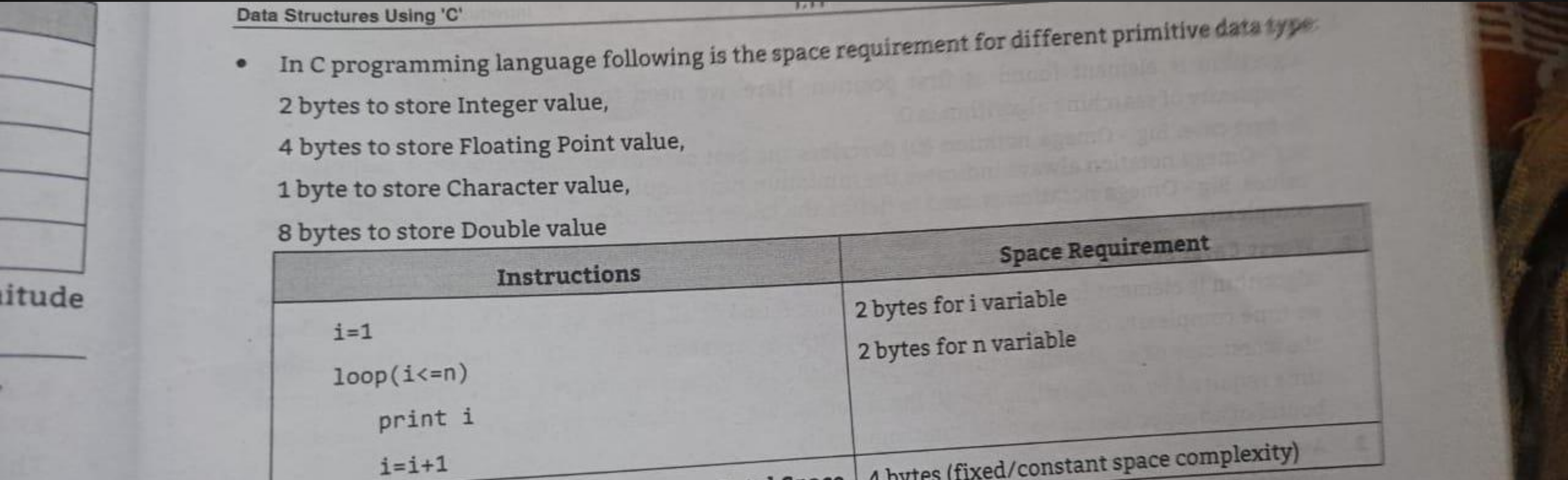
`return(total);`
`}`
	executes 1 time

Total Times	3n + 4

Thus, frequency count is 3n + 4. Neglecting the constants and by considering the order of magnitude we will get O(n) as run-time complexity.

3.2 Space complexity

SPACE COMPLEXITY EXAMPLE



Data Structures Using 'C'

- In C programming language following is the space requirement for different primitive data type:

  2 bytes to store Integer value,

  4 bytes to store Floating Point value,

  1 byte to store Character value,

  8 bytes to store Double value

Instructions	Space Requirement

i=1
loop(i<=n)
    print i
    i=i+1
	2 bytes for i variable
	2 bytes for n variable

[illegible]	4 bytes (fixed/constant space complexity)

Attempt any Three of the following:

Explain time and space complexity with an example.

Time Complexity: Time complexity of program or algorithm is amount of computer time that it needs to run to completion. To measure time complexity of an algorithm we concentrate on developing only frequency count for key statements.

```
Example:
#include<stdio.h>
void main ()
{
int i, n, sum, x;
sum=0;
printf("\n Enter no of data to be added");
scanf("% d", &n);
for(i=0 ; i<n; i++)
```

Statement	Frequency	Computational Time
sum=0	1	t_1
printf("\n Enter no of data to be added")	1	t_2
scanf("% d", &n)	1	t_3
for(i=0 ; i<n; i++)	n+1	$(n+1)t_4$
scanf("%d" , &x)	n	nt_5
sum=sum+x	n	nt_6
printf("\n Sum = %d ", sum)	1	t_7

Total computational time= t1+t2+t3+(n+1)t4 +nt6+nt5+t7

T= n(t4+t5+t6)+ (t1+t2+t3+t4+t7)

For large n , T can be approximated to

T= n(t4+t5+t6)= kn where k= t4+t5+t6

Thus T = kn or

Space Complexity: Total amount of computer memory required by an algorithm to complete its execution is called as space complexity of that algorithm. When a program is under execution it uses the computer memory for THREE reasons. They are as follows...

- Instruction Space: It is the amount of memory used to store compiled version of instructions.
- Environmental Stack: It is the amount of memory used to store information of partially executed functions at the time of function call.
- Data Space: It is the amount of memory used to store all the variables and constants.

If the amount of space required by an algorithm is increased with the increase of input value, then that space complexity is said to be Linear Space Complexity.

Example:

```
int sum(int A[ ], int n)
{
  int sum = 0, i;
  for(i = 0; i < n; i++)
    sum = sum + A[i];
  return sum;}
```

In the above piece of code it requires

'n*2' bytes of memory to store array variable 'a[ ]'
2 bytes of memory for integer parameter 'n'
4 bytes of memory for local integer variables 'sum' and 'i' (2 bytes each)
2 bytes of memory for return value.

That means, totally it requires '2n+8' bytes of memory to complete its execution. Here, the total amount of memory required depends on the value of 'n'. As 'n' value increases the space required also increases proportionately. This type of space complexity is said to be **Linear Space Complexity.**

OR

Thank you