Introduction to Data Structure and Algorithms

Agenda

- Syllabus abstract and Textbook
- Definitions of basic terms
- Example
- Why study Data structure and algorithms?
- What are the different classifications
- Some example data structures you study in this course

Course Objectives:

ICT 4303: FUNDAMENTALS OF DATA STRUCTURES AND ALGORITHMS [3 0 0 3]

- Design efficient algorithms for various problems
- Understand the basic concepts of linear and nonlinear data structures.
- Compare and contrast various searching and sorting techniques
- To apply data structure concepts for efficient representation of data

Abstract:-

ICT 4303: FUNDAMENTALS OF DATA STRUCTURES AND ALGORITHMS [3 0 0 3]

- Introduction to algorithms,
- Arrays: Elementary operations, Applications, Performance Analysis, Sparse matrix representation, Transpose of sparse matrix,
- Stacks operations, Arithmetic expression conversion and evaluation using stack,
- Queue Operations,
- Singly linked Lists, Circular lists, Doubly linked lists,
- Trees, Binary Tree traversals and different operations, Binary search Tree, Heaps,
- Graph Abstract type: Representations and elementary operations,
- Sorting and searching techniques, Analysis of algorithm.

Syllabus:

Introduction:

- Performance Analysis and Measurements – Asymptotic notations, introduction to data structure, classification of data structure, Abstract data types **[4 Hours]**

Arrays:

- The Array as Abstract Data type, Sparse Matrix Representation, Transpose of a sparse matrix, Representation of multidimensional arrays, The String abstract data type, Pattern matching. **[3 Hours]**

Stacks:

- Definition, operations on stacks, Evaluation of Arithmetic Expressions, Conversion of arithmetic expressions, Recursion, Multiple Stacks **[3 Hours]**

Queues:

- Definition, operations, application of circular queues. **[2 Hours]**

Linked Lists:

- Introduction to pointers and Dynamic memory allocation, Singly linked lists, Circular lists, Dynamically Linked Stacks and Queues, Polynomial representation and polynomial operations using singly linked list, Singly circular linked list, Doubly linked lists, Analysis of linked list operations. **[8 Hours]**

Trees:

- Tree terminology, Binary trees, Properties, Binary tree representations, Binary Tree Traversal algorithms, Expression tree, Heaps, Binary Search Trees. Complexity associated with various algorithms. [**8 Hours]**

Graphs:

- Definitions and Representations, Depth First Search, Breadth First Search, Connected components, Spanning trees, Complexities associated with each of the searching techniques. **[4 Hours]**

Sorting and Searching:

- Insertion Sort, Quick Sort, Merge sort, Heap sort, Shell sort, Linear search, Binary search, analysis of algorithms with respect to time complexity **[4 Hours]**

Text books

- Ellis Horowitz, Sartaj Sahni, Dinesh Mehta, *Fundamentals of Data Structures in C++ (2e),* Galgotia Publications, 2008.
- Mark Allen Weiss, *Data Structures and Algorithm Analysis in C++ (3e),* Pearson Education, 2009.
- Michael T. Goodrich, Roberto Tamassia, David Mount, *Data Structures and Algorithms in C++ (2e),* John Wiley & Sons, 2011.
- **Ellis Horowitz, Sartaj Sahni, Susan Anderson-Freed, *Fundamentals of Data structures in C (2e),* Silicon Press, 2008.**

Basic Terminologies

- **Data:** are simply a value or a set of values of different or same type which is called data types like string, integer, char etc.

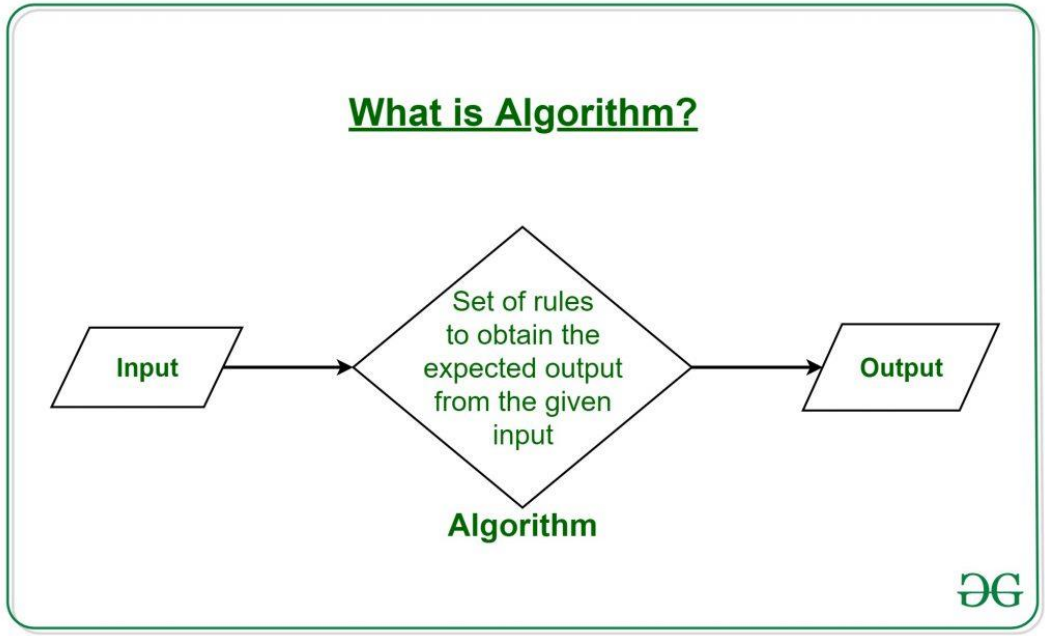
- **Structure:** Way of organizing information, so that it is easier to use

Data Structure

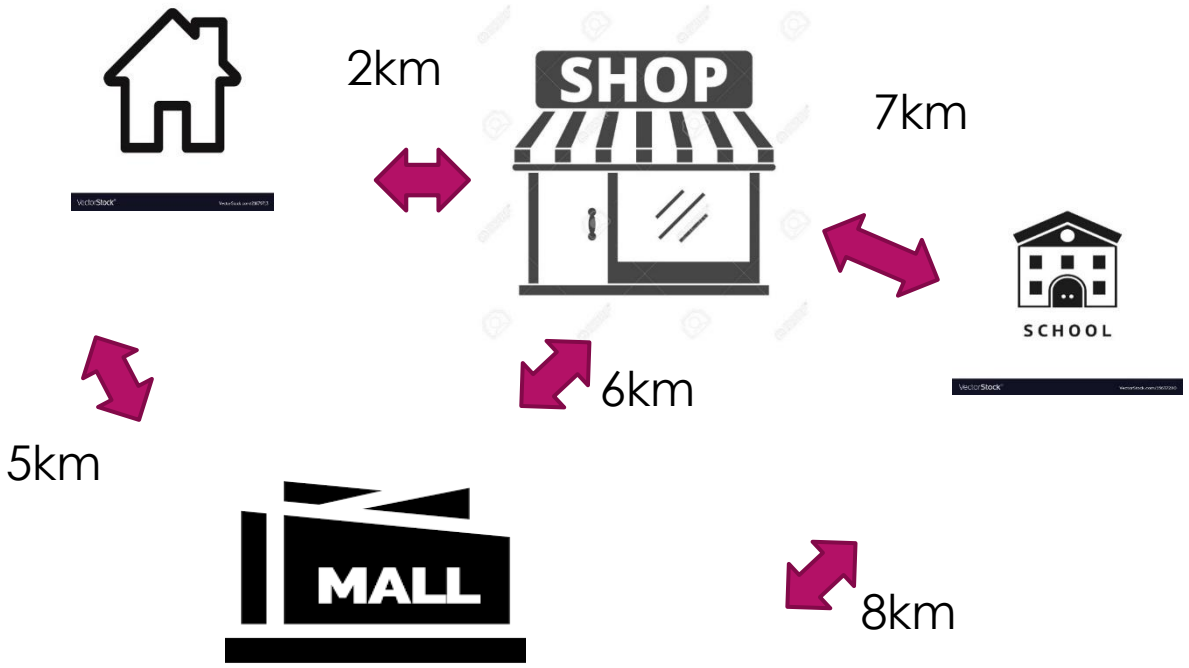
- In simple words we can define **data structures** as
  - Its a way of storing and organizing data in such a way that it is easier to use.
  - A data structure is a particular way of storing and organizing data in a computer so that it can be used efficiently in programs or algorithms.
  - A scheme for organizing related pieces of information.

Algorithm

- Set of instructions which work on the data structure to solve a particular problem

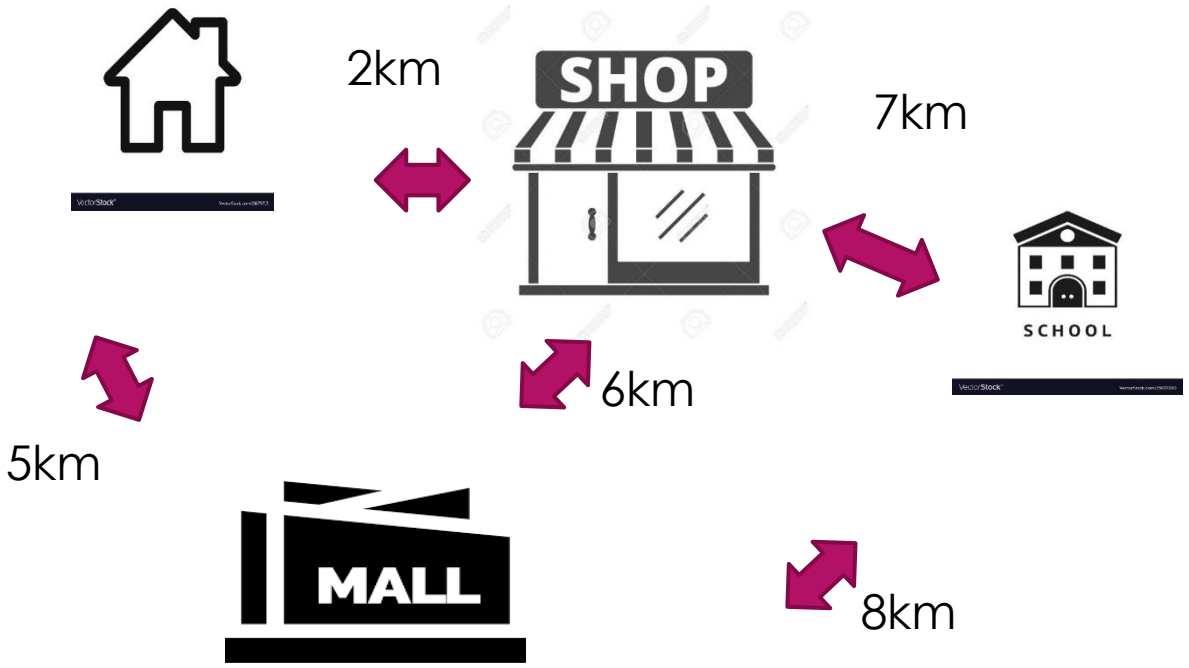


Example



	home	Mall	shop	school
home	0	5	2	9
Mall	5	0	6	8
shop	2	6	0	7
school	9	8	7	0

Example



	home	Mall	shop	school
home	0	5	2	infinity
Mall	5	0	6	8
shop	2	6	0	7
school	infinity	8	7	0

What is the shortest distance between home and school?

Algorithm to answer question

- Home to school direct path does not exist according to our convention
- So calculate the distance to school from home via shop
- Calculate the distance to school from home via mall
- Find smallest distance among these 2

Why study data structure and algorithms

- You know computer is a data processor. Without storing and knowing how to store data in different ways how u will process?
- How u will decide which approach is better ?
- How u will decide what data structure to use?
- Which processing method should be used?
- ……..
- So, It is very very important to know data structures

Classification of Data Structure …

- **Simple Data Structure:** Simple data structure can be constructed with the help of primitive data types. A primitive data structure used to represent the standard data types of any one of the computer languages (integer, Character, float etc.).

- **Compound Data Structure:** Compound data structure can be constructed with the help of any one of the primitive data structure and it is having a specific functionality. It can be designed by user.
  - It can be classified as **Linear and Non-Linear** Data Structure.

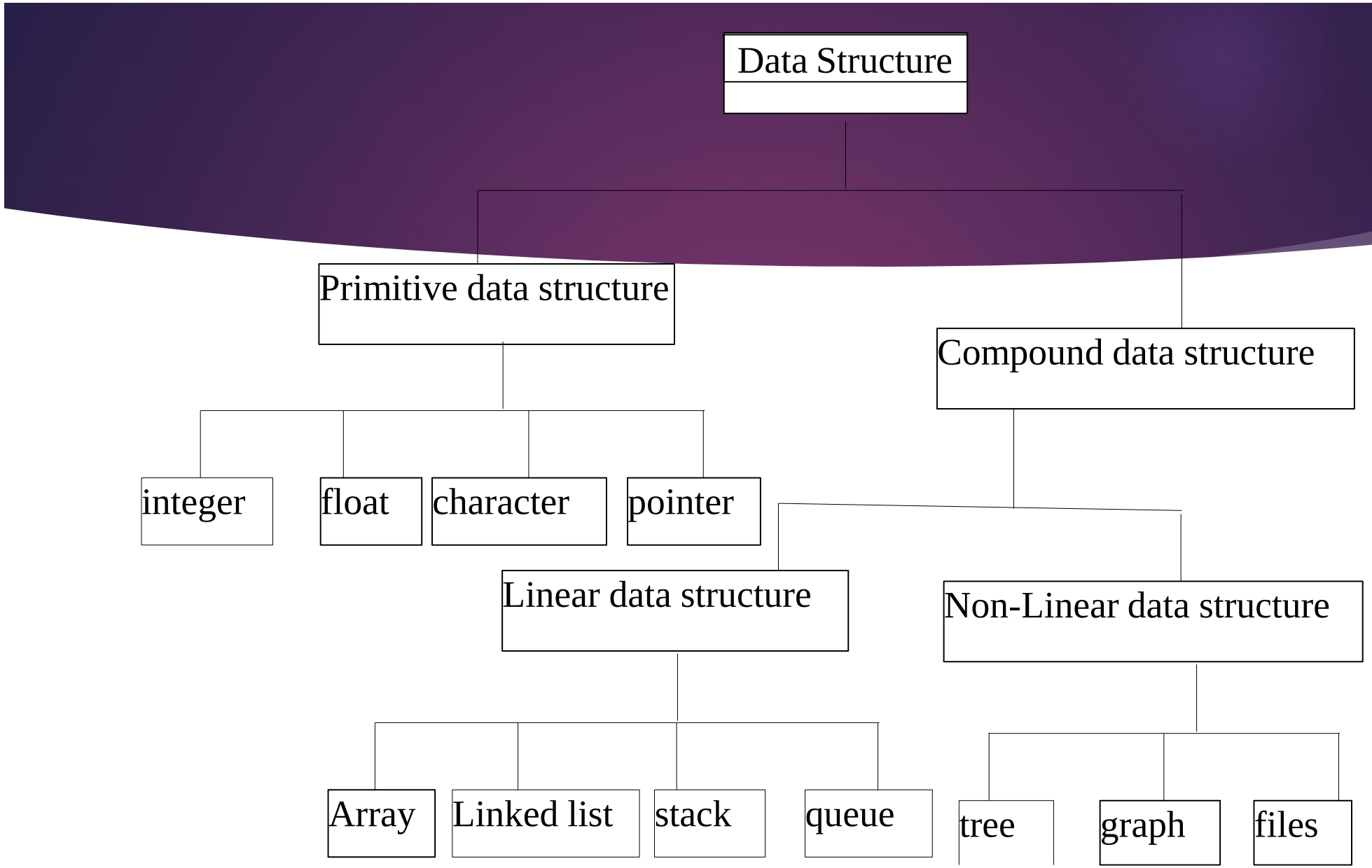
Compound data structure classification

- **Linear Data Structures:** A linear data structure traverses the data elements sequentially, in which only one data element can directly be reached.

  Ex: Arrays, Linked Lists

- **Non-Linear Data Structures:** Every data item is attached to several other data items in a way that is specific for reflecting relationships. The data items are not arranged in a sequential structure.

  Ex: Trees, Graphs


Data Structure
Primitive data structure
Compound data structure
integer
float
character
pointer
Linear data structure
Non-Linear data structure
Array
Linked list
stack
queue
tree
graph
files

Types of Data Structure

- **Array:** is commonly used in computer programming to mean a contiguous block of memory locations, where each memory location stores one fixed-length data item. e.g. Array of Integers int a[10], Array of Character char b[10]

Array of Integers									
0	1	2	3	4	5	6	7	8	9
5	6	4	3	7	8	9	2	1	2

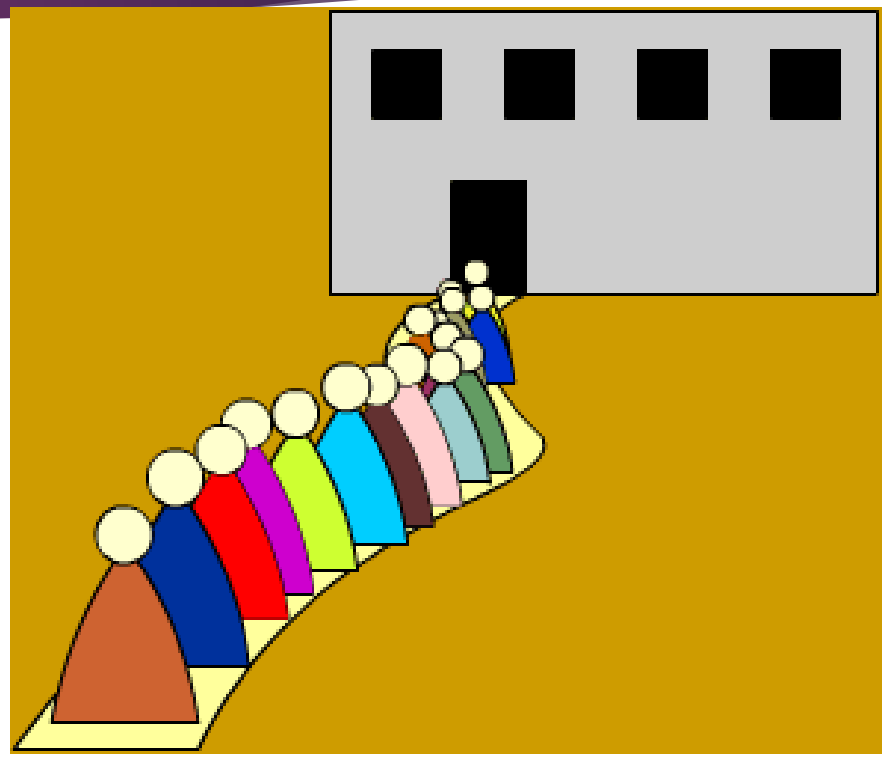
Array of Character									
0	1	2	3	4	5	6	7	8	9
5	6	4	3	7	8	9	2	1	2

Types of Data Structure ...

- **Stack:** A stack is a data structure in which items can be inserted only from one end and get items back from the same end. There , the last item inserted into stack, is the the first item to be taken out from the stack. In short its also called Last in First out [LIFO].

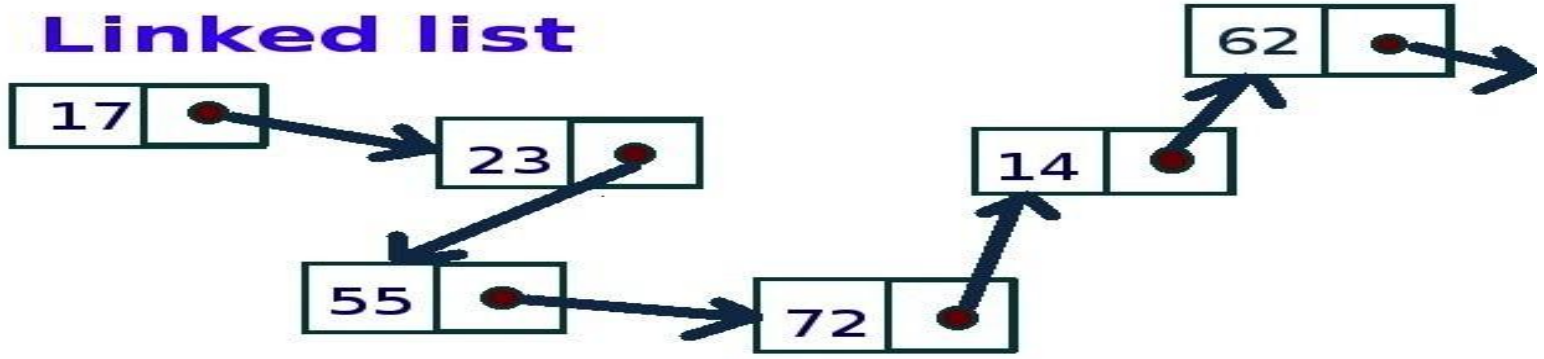
Types of Data Structure ...

- **Queue:** A queue is two ended data structure in which items can be inserted from one end and taken out from the other end. Therefore , the first item inserted into queue is the first item to be taken out from the queue. This property is called First in First out [FIFO].



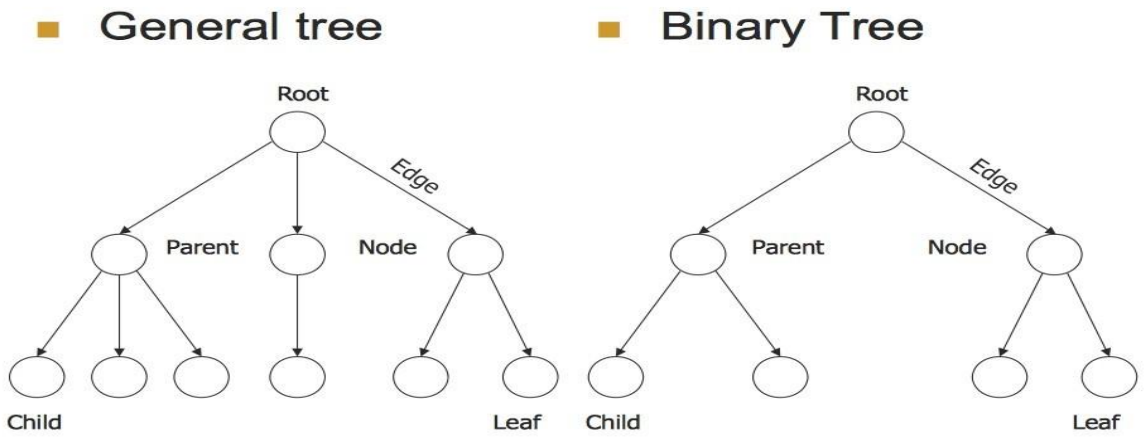
Types of Data Structure ...

- **Linked List:** Could alternately used to store items. In linked list space to store items is created as is needed and destroyed when space no longer required to store items. Hence linked list is a dynamic data structure space acquire only when need.



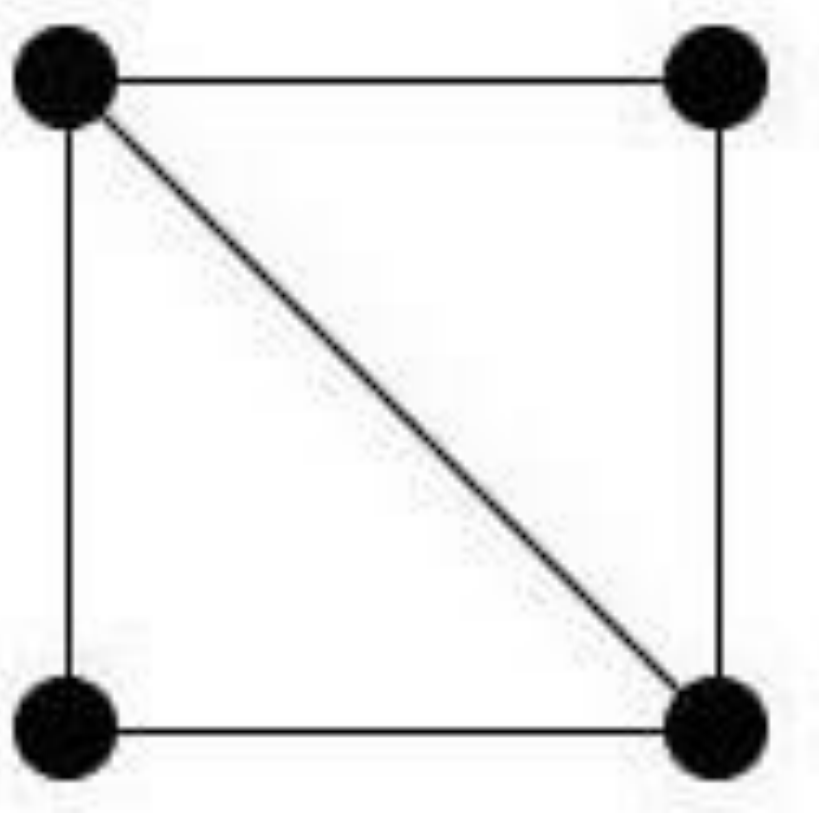
Types of Data Structure ...

- **Tree:** is a non-linear data structure which is mainly used to represent data containing a hierarchical relationship between elements.
  - **Binary Tree:** A binary tree is a tree such that every node has at most 2 child and each node is labeled as either left of right child.



Types of Data Structure ...

- **Graph:** It is a set of items connected by edges. Each item is called a vertex or node. Trees are just like a special kinds of graphs. Graphs are usually represented by G = (V, E), where V is the set vertices and E is the set of Edges.



Selecting a Data Structure

- Analyze the problem to determine the resource constraints a solution must meet.
- Determine the basic operations that must be supported. Quantify the resource constraints for each operation.
- Select the data structure that best meets these requirements.

Structure of C++ Program

```
1  // my first program in C++
2
3  #include <iostream>
4  using namespace std;
5
6  int main ()
7  {
8    cout << "Hello World!";
9    return 0;
10 }
```

```
Hello World!
```

Comment Line

```
1  // my first program in C++
2
3  #include <iostream>
4  using namespace std;
5
6  int main ()
7  {
8    cout << "Hello World!";
9    return 0;
10 }
```

```
Hello World!
```

Preprocessor Directives

```
1  // my first program in C++
2
3  #include <iostream>
4  using namespace std;
5
6  int main ()
7  {
8    cout << "Hello World!";
9    return 0;
10 }
```

```
Hello World!
```

namespace

```
1  // my first program in C++
2
3  #include <iostream>
4  using namespace std;
5
6  int main ()
7  {
8    cout << "Hello World!";
9    return 0;
10 }
```

```
Hello World!
```

Main Function

```
1  // my first program in C++
2
3  #include <iostream>
4  using namespace std;
5
6  int main ()
7  {
8    cout << "Hello World!";
9    return 0;
10 }
```

```
Hello World!
```

C++ statement

```
1  // my first program in C++
2
3  #include <iostream>
4  using namespace std;
5
6  int main ()
7  {
8    cout << "Hello World!";
9    return 0;
10 }
```

```
Hello World!
```

Writing code

```
int main ()
{
  cout << " Hello World!";
  return 0;
}
```

We could have written:

```
int main () { cout << "Hello World!"; return 0; }
```

```cpp
// my second program in C++

#include <iostream>

using namespace std;

int main ()
{
  cout << "Hello World! ";
  cout << "I'm a C++ program";
  return 0;
}
```

```
Hello World! I'm a C++ program
```

```
1  // my second program in C++
2
3  #include <iostream>
4
5  using namespace std;
6
7  int main ()
8  {
9    cout << "Hello World! ";
10   cout << "I'm a C++ program";
11   return 0;
12 }
```

```
int main ()
{
  cout <<
    "Hello World!";
  cout
    << "I'm a C++ program";
  return 0;
}
```

```
int main () { cout << " Hello World! "; cout << " I'm a C++ program "; return 0; }
```

Comments

C++ supports two ways to insert comments:

```
// line comment
/* block comment */
```

The first of them, known as **line comment**, discards everything from where the pair of slash signs (**//**) is found up to the end of that same line.

The second one, known as **block comment**, discards everything between the **/*** characters and the first appearance of the ***/** characters, with the possibility of including more than one line.