DATA STRUCTURES

SUBJECT CODE -18B11CI311

L-T scheme: 3-1 and Credits: 4

Syllabus

- Introduction to Data Structures
- Algorithm and Complexity
- Array
- Searching And Sorting Algorithms
- Stack
- Queue
- Linked List
- Graph and Tree

Recommended Books

TexT Books:

1. Sartaj Sahni, “Data Structures, Algorithms”, Tata Mc Graw Hill, New York
2. E Balagurusamy,”Data Structures using C”, McGraw Hill
3. **Narasimha Karumanchi, “Data Structures and Algorithms” Made Easy**

RefeRence Books:

1. **Corman et al., “Introduction to Algorithms”**
2. Kruse, Tonso, Leung, “Data Structures and Program Design in C”
3. Langsam, Augestein, Tanenbaum, “Data Structures using C and C++”
4. Weiss, “Data Structures and Algorithm Analysis in C/C++”
5. Carrano and Prichard, “Data Abstraction and Problem solving with C++”

Marks Distribution(Data Structures)

Component & Nature	Duration	Marks / Weightage
T1	1 hr	15
T2	1&1/2 hrs	25
T3	2hrs	35
Tutorials		05
Attendance		05
Quiz		05
Assignments		10
Total		100

Marks Distribution(Data Structures Lab)

Component & Nature	Duration	Marks / Weightage
P1 (Mid sem lab – Viva/Test)	2 hrs	15
P2 (Mid sem lab – Viva/Test)	2 hrs	15
Continuous Evaluation(Lab work)		**40**
Lab record		**15**
Attendance & discipline in lab		15
Total		100

Discussion Hours

Batch	Day	Time (From-To)
B1B2	Monday	4 PM to 6 PM
	Friday	4 PM to 6 PM
B3B4	Tuesday	4 PM to 6 PM
	Saturday	1 PM to 1.30 PM

For any doubt or query, you may visit my cabin during your respective discussion hours.

Prerequisite Knowledge

1. Basic programming constructs from C programming, like data types, operators, decision control, loops, string and functions etc.
2. Dynamic memory allocation
3. Pointers
4. Structures

Introduction to Data Structures

Why This Course?

- You will be able to write fast programs
- You will be able to solve new problems
- You will be able to evaluate the quality of a program (Analysis of Algorithms: Running time and memory space)
- You will be able to give non-trivial methods to solve problems.
- Your algorithm (program) will be faster than others.

Preface of Data structure

- Data: simply a value or a set of values of different type which is called data types like string, integer, char etc.
- Structure: Way of organizing information, so that it is easier to use –can be easier to use.

Data Structure?

A **data structure** is a particular way of organizing data in a computer memory so that it can be used **efficiently**.

- Even though computers can perform literally millions of mathematical computations per second, when a problem gets large and complicated, performance can nonetheless be an important consideration.
- One of the most crucial aspects to how quickly a problem can be solved is how the data is stored in memory.

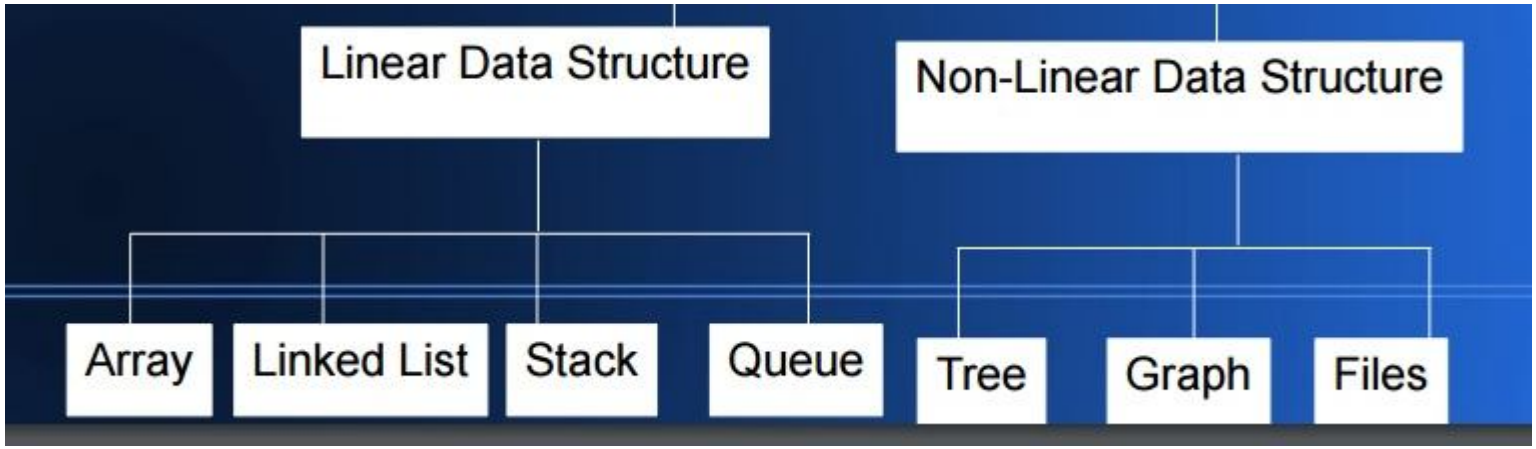
Why Data Structure

- Human requirement with computer are going to complex day by day.
- To solve the complex requirements in efficient way we need this study.
- Provide fastest solution of human requirements.
- Provide efficient solution of complex problem
  - Space
  - Time

Classification of Data Structure

- **Simple Data type or data structure**: Simple data structure can be constructed with the help of primitive data structure. 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.

Classification of Data Structure



Classification of Data Structures

- **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

Operations on Data Structures

- **Basic operations –**
- Traversing – Accessing and processing record exactly once
- Insertion – Adding new record to data structure
- Deletion – Removing particular record from the data structure
- Searching – Finding the location of the record in data structure
- **Special operations –**
- Sorting – Arranging the record in some specific order
- Merging – Combining the records from different data structures to single data structure

Type of data structures

No single data structure works well for all purposes,

1. Linked list
2. Stack
3. Queue
4. Tree
5. Graph

Abstract Data Type (ADT)

- A data type is considered **abstract** when it is defined in terms of operations on it, and its implementation is hidden. i.e. An ADT is implementation independent
- Hence in case ADT we know what it does, but not necessarily how it will do it.

Examples of ADT:

1. stack: operations are "push an item onto the stack", "pop an item from the stack", "ask if the stack is empty"; implementation may be as array or linked list or whatever.

2. queue: operations are "add to the end of the queue", "delete from the beginning of the queue", "ask if the queue is empty"; implementation may be as array or linked list or heap.

Cost of Data Structures

Cost of data structure or algorithm is always calculated in terms of following parameters –

1. **Time Complexity**: Time required to perform a basic operation on data
2. **Space Complexity**: Memory space required to store the data items of data structure
3. **Programming efforts to write code for basic operations.**

Space – Time Tradeoff

Reduction of needed space could increase the time of execution.

Example –

1. **Compressed and uncompressed data –**

A space–time tradeoff can be applied to the problem of data storage. If data is stored uncompressed, it takes more space but less time than if the data were stored compressed (since compressing the data reduces the amount of space it takes, but it takes time to run the **decompression algorithm**).
Depending on the particular instance of the problem, either way is practical.

2. **Smaller code and loop unrolling –**

- Larger code size can be traded for higher program speed when applying **loop unrolling(**A basic compiler optimization is known as 'loop unrolling. Loop unrolling means going back to some previous statement.**)**.
- This technique makes the code longer for each iteration of a loop, but saves the computation time required for jumping back to the beginning of the loop at the end of each iteration.

Selection of data structure

- Organization of data in memory affects the performance of algorithm
- So, selection of data structure plays important role to make algorithm efficient

  But, it depends on the nature of problem and operations to be supported.

Typically, following steps are to be followed to select an efficient data structure –

1. Analyze the problem very carefully and estimate the resources so that solution must meet.
2. Determine the basic operation to be supported
3. Select the best data structure which meets the above requirements.