Data Structures

SunBeam Institute of Information & Technology, Hinjwadi, Pune & Karad.

Introduction

Mr. Sachin G. Pawar.

Educational Qualification:
1. B.Sc. Electronics.
2. M.Sc.Tech (Industrial Maths & Computer Applications).

Work Experience:
1. 7 Yrs of teaching experience in SunBeam, for PreCAT & CDAC Courses.
2. 3+ Yrs of Industry Experience of Software Development.

Technical Skills:
1. C Programming Language
2. C++ Programming Language
3. Data Structures
4. Operating System Concepts

 # DAY-01:

*** Introduction:**

- Cocept & Definition
- Types of data structures with examples

*** Algorithms:**

- What is an algorithm?
- Analysis of an algorithm: time complexity & space complexity
- Asymptotic analysis

*** Array:**

- Searching Algorithms:
  1. Linear Search: Implementation
  2. Binary Search: Implementation

 # DAY-02:

*** Array:**

- Sorting Algorithms:
    1. Selection Sort : Algorithm & Implementation
    2. Bubble Sort : Algorithm & Implementation
    3. Insertion Sort : Algorithm & Implementation

 # DAY-03:

*** Array:**

- Sorting Algorithms:
    4. Quick Sort : Algorithm
    5. Merge Sort : Algorithm
- Limitations of an Array data structure

 # DAY-03

*** Linked List:**

- Introduction: concept & definition
- Types of Linked List
- Implementation of Singly Linear Linked List Operations

 # DAY-04:

*** Linked List:**

- Implementation of Singly Linear Linked List Operations
- Singly Circular Linked List: Concept
- Doubly Linear Linked List: Concept
- Doubly Circular Linked List: Concept
- Applications of Linked List

 # DAY-05

*** Stack**

- Introduction: concept & definition
- Implementation of a static stack by using an array
- Applications of Stack
- Stack applications algorithms:

1. Converstion infix expression into its equivalent prefix & postfix.
2. Conversion of prefix into its equivalent postfix
3. Postfix evaluation

 # DAY-06:

*** Queue**

- Introduction: concept & definition
- Implementation of linear queue & circular queue
- Applications of queue

 # DAY-07 & DAY-08:

*** Tree Terminologies**

*** Graph Terminologies**

Q. What is Data Structure?

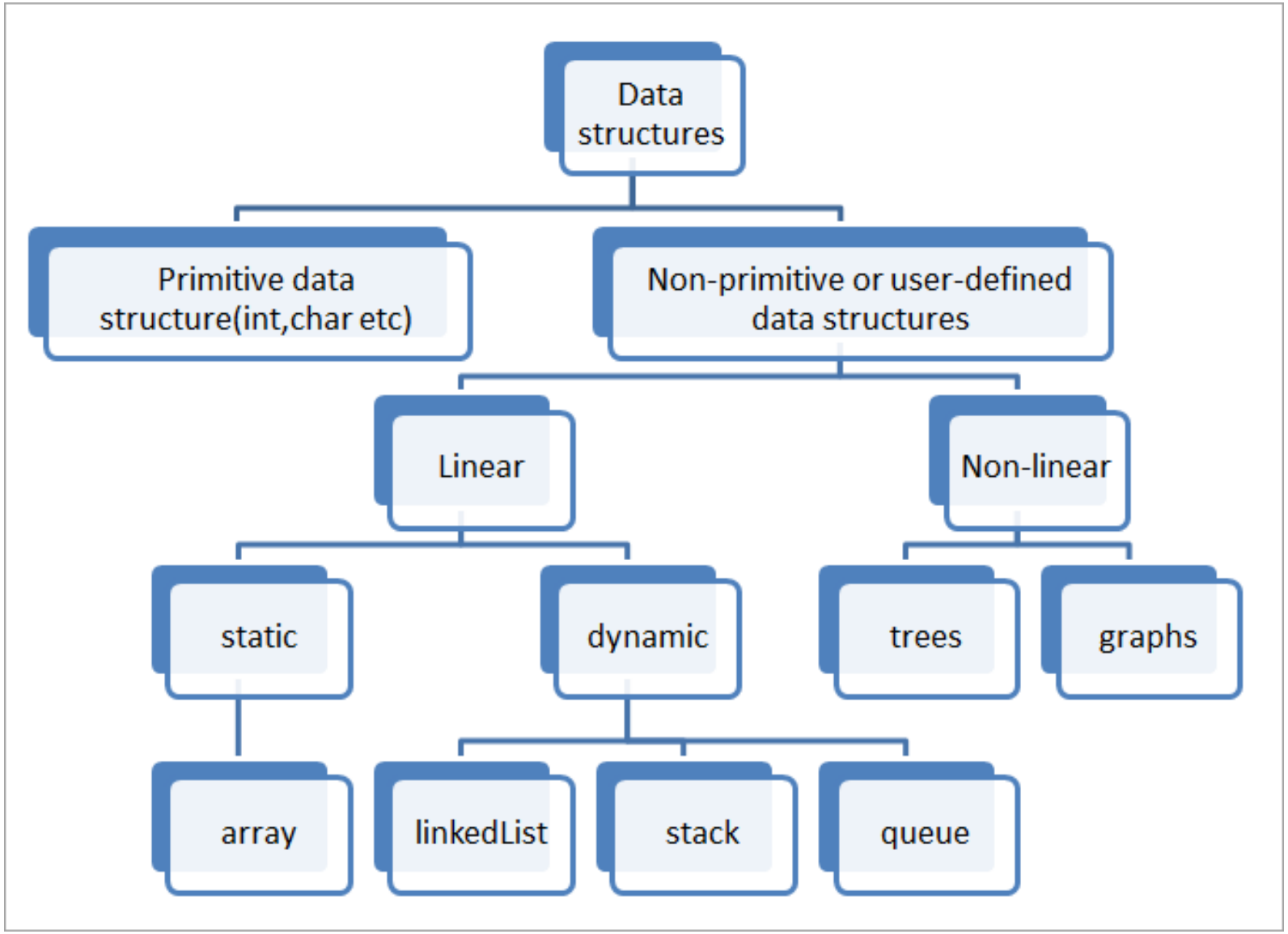
It is a way to store data into the memory (i.e. into the main memory) in an organized manner so that operations (like insertion, deletion, searching, sorting, traversal etc...) can be performed on it efficiently.

Q. Why there is a need of Data Structure?

To achieve three things in programming:

1. Efficiency
2. Reusability
3. Abstraction

Introduction: DS Classification



Introduction

Q. What is an Algorithm?

- An algorithm is a finite set of instructions written in human understandable language (e.g. like an english), if followed, acomplishesh a given task.
- An algorithm is a finite set of instructions written in human understandable language (e.g. like english) with some programming constraints, if followed, acomplishesh a given task, such an algorithm is reffered as a **pseudocode**.

 # Analysis of an algorithm:

Analysis of an algorithm is, to calculate how much computing time i.e. **time complexity** and computer memory/space i.e. **space complexity**, it needs to run to completion.

1. Time Complexity: time complexity of an algorithm is the amount of **time i.e. computer time,** it needs to run to completion.

2. Space Complexity: space complexity of an algorithm is the amount of **space i.e. computer memory,** it needs to run to completion.

Introduction

 # Best Case Time Complexity:

- If an algorithm takes minimum amount of time to complete its execution then it is reffered as best case time complexity.
- **Big Omega (Ω)**, notation can be used to represent best case time complexity of an algorithm.

 # Worst Case Time Complexity:

- If an algorithm takes maximum amount of time to complete its execution then it is reffered as worst case time complexity.
- **Big Oh (O)**, notation can be used to represent worst case time complexity of an algorithm.

 # Average Case Time Complexity:

- If an algorithm takes neither minimum nor maximum amount of time to complete its execution then it is reffered as an average case time complexity.
- **Big Theta (θ)**, notation can be used to represent worst case time complexity of an algorithm.

Array: Searching Algorithms

 # Searching:

To search/find a key element in a given collection/list of elements.

Two seaching algorithms are there:

1. **Linear Search**
2. **Binary Search**

1. Linear Search: In this algorithm key element gets compared with each element in a list/collection of elements sequentially from the first element either match is not found or maximum till last element, if match is found it returns true and if match not found it returns false.

- **Best Case Time Complexity : Ω(1)**
- **Worst Case Time Complexity : O(n)**
- **Average Case Time Complexity : θ(n)**

2. Binary Search:

- This algorithm can be applied only on an array.
- Array elements must be sorted
- This algorithm uses **divide-and-conquer** stratergy.

- **Best Case Time Complexity : Ω(1)**
- **Worst Case Time Complexity : O(log n)**
- **Average Case Time Complexity : θ(log n)**

Array: Sorting Algorithms

1. Selection Sort:

- Best Case Time Complexity : $\Omega(n^2)$
- Worst Case Time Complexity : $O(n^2)$
- Average Case Time Complexity : $\theta(n^2)$

2. Bubble Sort:

- Best Case Time Complexity : $\Omega(n)$
- Worst Case Time Complexity : $O(n^2)$
- Average Case Time Complexity : $\theta(n^2)$

3. Insertion Sort:

- Best Case Time Complexity : $\Omega(n)$
- Worst Case Time Complexity : $O(n^2)$
- Average Case Time Complexity : $\theta(n^2)$

4. Quick Sort:

- Best Case Time Complexity : $\Omega(n \log n)$
- Worst Case Time Complexity : $O(n^2)$
- Average Case Time Complexity : $\theta(n \log n)$

5. Merge Sort:

- Best Case Time Complexity : $\Omega(n \log n)$
- Worst Case Time Complexity : $O(n \log n)$
- Average Case Time Complexity : $\theta(n \log n)$