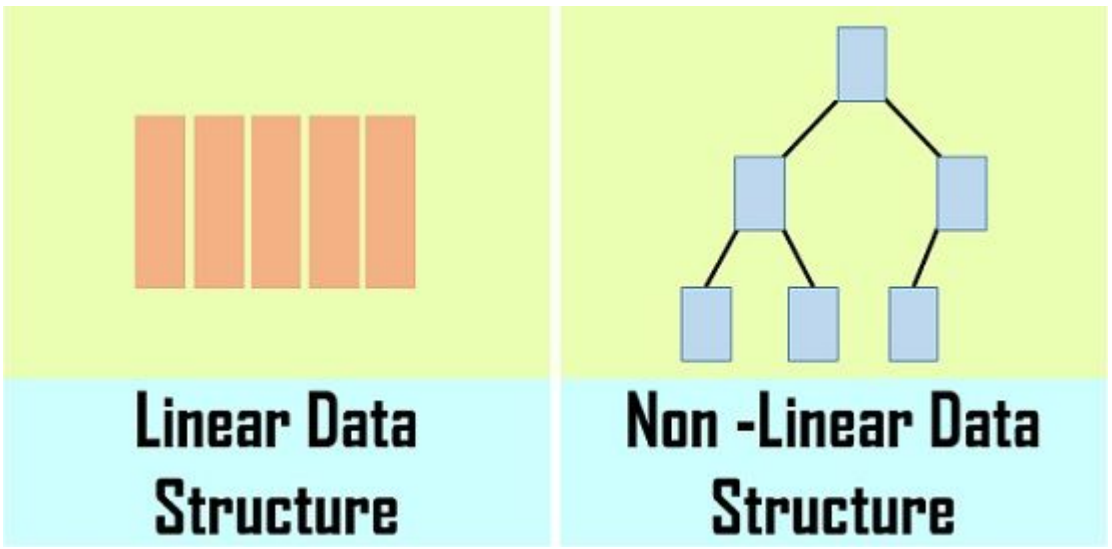
Data Structures & Algorithms

- It is a way to organize data.
- After organizing data it becomes easy to process it.

Data Structures Types:

- Linear:
  - They have data elements arranged in a sequential manner.
  - Each member is connected to its previous and next element.
  - Because they are connected sequentially it becomes easy to traverse them.
  - Array, Linked List, Stack, Queue are linear data structures.
  - They are single level.

- Non-Linear:
  - Data elements inside these data structures are not in sequence.
  - They are basically connected to one another through different paths.
  - They are basically stored in multi-level so in order to traverse each and every element in that non-linear data structure takes some amount of time.
  - Tree and Graph are non-linear data structures.



Algorithms:

- An algorithm is a set of instructions to perform a task or solve a given problem.

Analysis of Algorithms:

- There are several different algorithms to solve a given problem.
- Analysis of algorithm deals in finding the best algorithm which runs fast and takes in less memory.
- We should check time complexity and space complexity to find the best algorithm.

Time Complexity:

- Its amount of time taken by algorithm to run.
- The input processed by an algorithm helps in determining the time complexity.

Space Complexity:

- Its amount of memory or space taken by an algorithm to run.
- The memory required to process the input by an algorithm helps in determining the space complexity.

Asymptotic Analysis of an Algorithm:

- Asymptotic analysis helps in evaluating performance of an algorithm in terms of input size and its increase.
- Using asymptotic analysis we don't measure actual running time of the algorithm.
- It helps in determining how time and space taken by an algorithm increases with input size.

Asymptotic Notations:

- Asymptotic Notations are the mathematical tools used to describe the running time of an algorithm in terms of input size.

Types of Asymptotic Notations:

- There are three notations for performing runtime analysis of an algorithm.

1. Omega (Ω) Notation
2. Big O (O) Notation
3. Theta (θ) Notation

Omega (Ω) Notation:

- It is a formal way to express the lower bound of an algorithm's running time.
- Lower bound means for any given input this notation determines the best amount of time an algorithm can take to complete.

Big O (O) Notation:

- It is the formal way to express the upper bound of an algorithm's running time.
- Upper bound means for any given input this notation determines the longest amount of time an algorithm can take to complete.

Rules:

- It's a Single Processor
- It performs Sequential Execution of statements
- Assignment operation takes 1 unit of time
- Return statement takes in 1 unit of time
- Arithmetical operation takes 1 unit of time
- Logical operation takes 1 unit of time
- Other small/single operations takes 1 unit of time
- Drop lower order terms → T = n^2 + 3n + 1 → O(n^2)
- Drop constant multipliers → T = 3n^2 + 6n + 1 → O(n^2)

Theta (θ) Notation:

- It is the formal way to express both the upper and lower bound of an algorithm's running time.

- By Lower and Upper bound means for any given input this notation determines the average amount of time an algorithm can take to complete.

Array Data Structure:

- Array is the collection(box) of data elements of specified type.
- All members holding partitions are adjacent or contiguous.
- Each partition has two neighbors except the first and last one.
- Size of the array is fixed and cannot be modified.
- Being adjacent each partition is indexed and can be determined by its position.
- All data holding the partitions have contiguous memory locations.
- Index starts at 0 and for (one dimensional array) ends at length - 1