**Data Structure & Algorithms - Narsimha Karumanchi**

**Refer - careermonk.com**

**Data Types -** Is a set of data with predefined values. Example- integer, float, string, character etc. However at the top level there are only 2 types of data types-

* System defined (a.k.a Primitive data type)
* User defined

**System Defined Data Type -** These are defined by the system and are provided by many programming languages - int, float, char, double, bool etc. The number of bits allocated for each data type are dependent on the programming languages, compiler an OS.For these, different languages may use different sizes.

Example- If int takes 2 bytes then (2^16) total possible values are -32768 to +32767. If it takes 4 bytes then (2^32) then total possible value are -2147483648 to + 2147483647

**User Defined Data Type -** These are defined by the user.Like, structures in C/C++ and classes in Java/Python.

Example- class newType (object):

def \_\_init\_\_(self,data1,data2):

self.data1=data1

self.data2=data2

**Data Structures-**

Based on above discussion, once we have data in variables, we need some mechanism for manipulating that data to solve problems. Data Structure is a particular way/special format of storing and organizing that data in a structured way. General data structure types includes arrays, files, linked lists,stacks, queues, trees, graphs etc. Data structures are classified into 2 types based on the organisation of elements-

* Linear Data Structures- Elements are accessed in a sequential order but it is not compulsory to store all elements sequentially - Stacks, Queues and linked lIsts
* Non-Linear Data Structures- Elements are accessed/stored in a nonlinear order. - Trees and graphs

**Abstract Data Types (ADTs)-**

To simplify the process of solving problems, we combine data structures along with their operations and call it ADTs. It consists of 2 parts-

* Declaration of data
* Declaration of operations

Commonly used ADTs include- Linked Lists, Stacks, Queues, Priority Queues, Binary Trees, Dictionaries, Disjoint sets (Union & Find), Hash Tables, Graphs and many other. Example- stacks uses LIFO mechanism while storing data in data structure.The last element inserted into the stack gets deleted first. Common operations of it are: creating the stack, pushing an element onto the stack, popping an element from stack, finding the current top of the stock, finding the number of elements in the stack etc.

**Algorithm-** Is the step by step instruction to solve a given problem.

**Running Time Analysis-** It is a process of determining how processing time increases as the size of the problem (input size) increases. Input size is the number of elements in the input. Following are common type of inputs-

* Size of an array
* Polynomial degree
* Number of elements in a matrix
* Number of bits in binary representation of the input.
* Vertices and edges in a graph

To compare algorithms, let us define few objective measures:

* Execution times? - Not a good measure as these are specific to a particular computer.
* Number of statements executed? - Not a good measure as these vary with the programming language as well as the style of an individual programmer.
* Ideal solution? - Let us assume that we expressed running time of given algorithm as a function of the input size and compare these different functions corresponding to running times. This type of comparison is independent of machine time, programming style etc.

**Rate growth -**