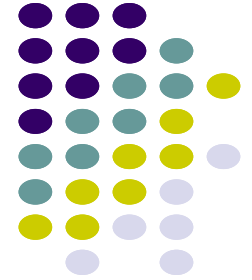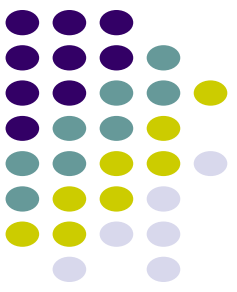# Data Structures & Algorithms

Learning data structures and algorithms allow us to write efficient and optimized computer programs.
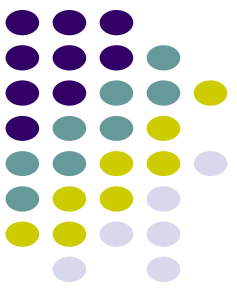
What is an Algorithm?

- In computer programming terms, an algorithm is a set of well-defined instructions to solve a particular problem.

- It takes a set of input(s) and produces the desired output.

- For example, An algorithm to add two numbers:
  - Take two number inputs
  - Add numbers using the + operator
 - Display the result

Qualities of a Good Algorithm

- Input and output should be defined precisely.

- Each step in the algorithm should be clear and unambiguous.

- Algorithms should be most effective among many different ways to solve a problem.

- An algorithm shouldn't include computer code. Instead, the algorithm should be written in such a way that it can be used in different programming languages.
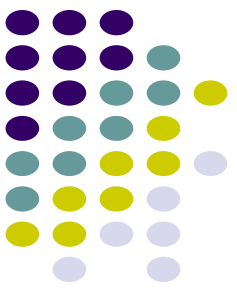
Data Structure and Types
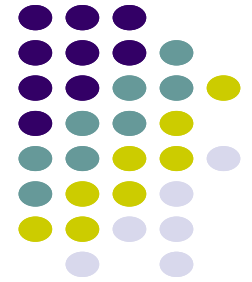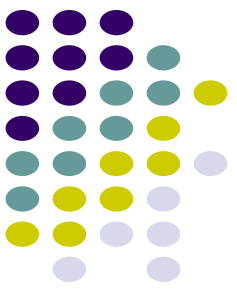
What are Data Structures?

- Data structure is a storage that is used to store and organize data.

- It is a way of arranging data on a computer so that it can be accessed and updated efficiently.

- Depending on your requirement and project, it is important to choose the right data structure for your project.

- For example, if you want to store data sequentially in the memory, then you can go for the Array data structure.

memory locations

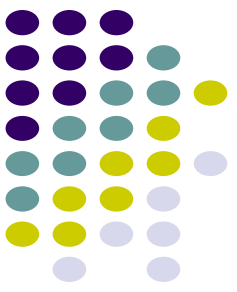| 1004	1005	1006	1007	1008	1009	1010
...	2	1	5	3	4	...
	0	1	2	3	4	

index

Types of Data Structure

- Data Structures are divided into two categories:
  - Linear data structure
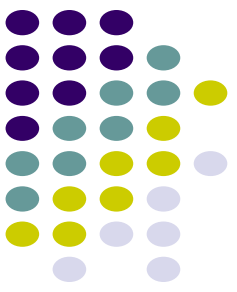 - Non-linear data structure

Linear data structures

- In linear data structures, the elements are arranged in sequence one after the other. Since elements are arranged in particular order, they are easy to implement.

- However, when the complexity of the program increases, the linear data structures might not be the best choice because of operational complexities.

- **Some Popular linear data structures are described in the subsequent slides**
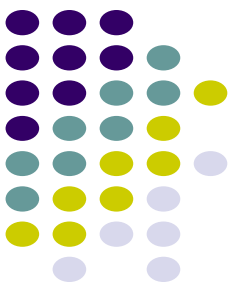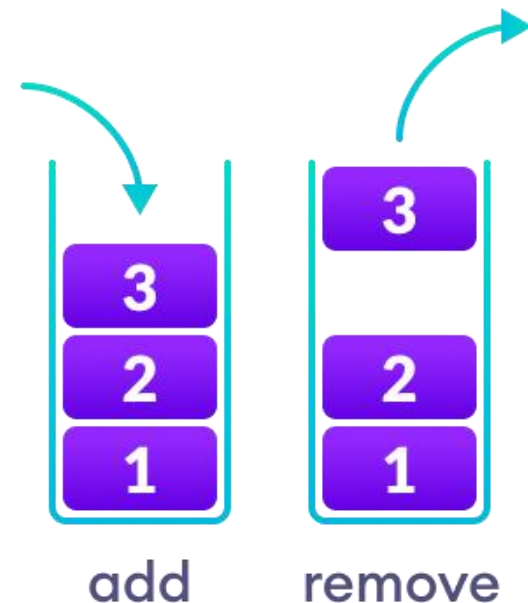
Array Data Structure

- In an array, elements in memory are arranged in continuous memory.
- All the elements of an array are of the same type.
- And, the type of elements that can be stored in the form of arrays is determined by the programming language
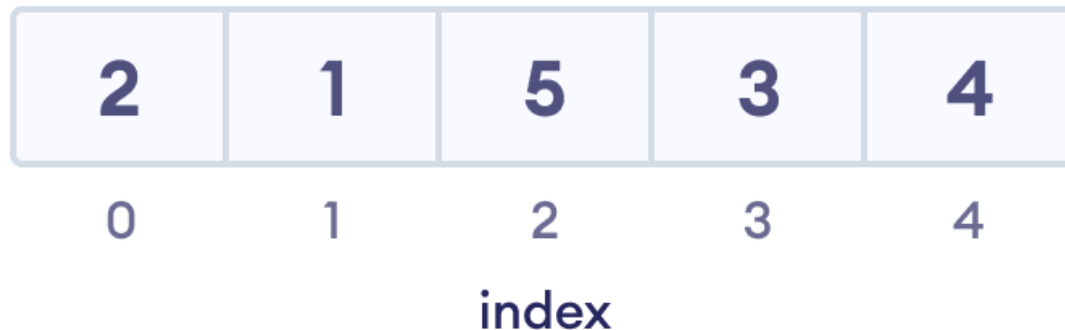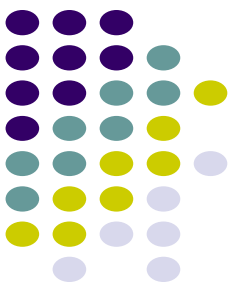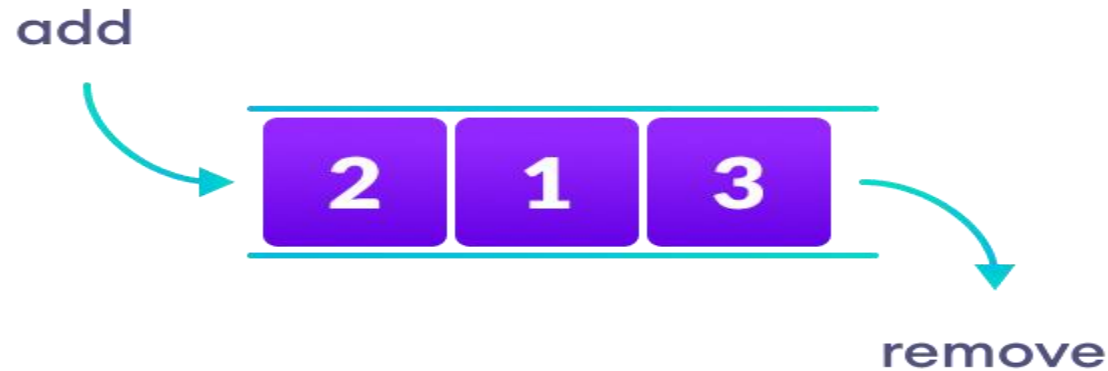
Stack Data Structure

- In stack data structure, elements are stored in the LIFO principle.
- That is, the last element stored in a stack will be removed first.
- It works just like a pile of plates where the last plate kept on the pile will be removed first.
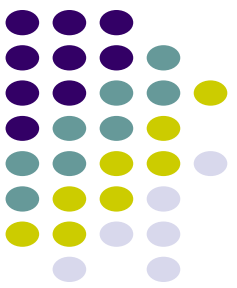

add   remove

Queue Data Structure

- Unlike stack, the queue data structure works in the FIFO principle where first element stored in the queue will be removed first.

- It works like a queue of people at the ticket counter where first person on the queue will get the ticket first.
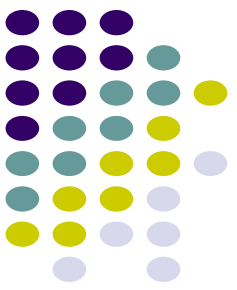
Linked List Data Structure

- In linked list data structure, data elements are connected through a series of nodes.

- And, each node contains the data items and address to the next node.
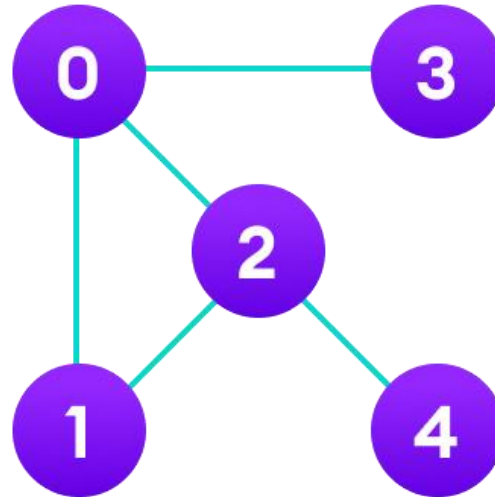
Non linear data structures

- Unlike linear data structures, elements in non-linear data structures are not in any sequence.

- Instead they are arranged in a hierarchical manner where one element will be connected to one or more elements.

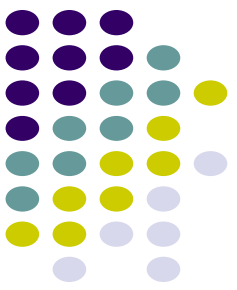- Non-linear data structures are further divided into graph and tree based data structures.
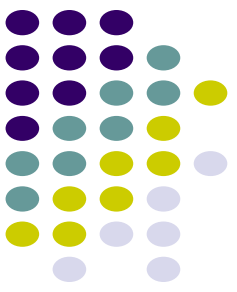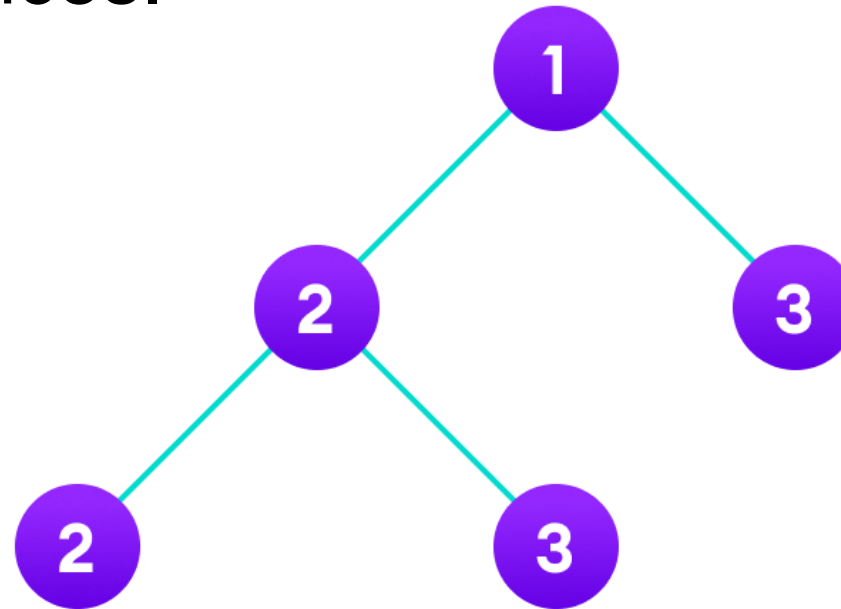
Graph Data Structure

- In graph data structure, each node is called vertex and each vertex is connected to other vertices through edges
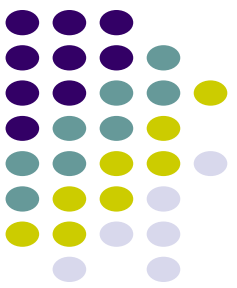
Trees Data Structure

- Similar to a graph, a tree is also a collection of vertices and edges. However, in tree data structure, there can only be one edge between two vertices.
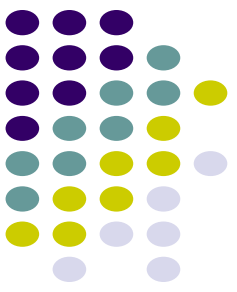
Linear Vs Non-linear Data Structures

Linear Data Structures	Non Linear Data Structures
The data items are arranged in sequential order, one after the other.	The data items are arranged in non-sequential order (hierarchical manner).
All the items are present on the single layer.	The data items are present at different layers.
It can be traversed on a single run. That is, if we start from the first element, we can traverse all the elements sequentially in a single pass.	It requires multiple runs. That is, if we start from the first element it might not be possible to traverse all the elements in a single pass.
The memory utilization is not efficient.	Different structures utilize memory in different efficient ways depending on the need.
The time complexity increase with the data size.	Time complexity remains the same.
Example: Arrays, Stack, Queue	Example: Tree, Graph, Map

Why Data Structure and Algorithms?

- Knowledge about data structures help you understand the working of each data structure.

- And, based on that you can select the right data structures for your project.

- This helps you write memory and time efficient code.

- Programming is all about data structures and algorithms.

- Data structures are used to hold data while algorithms are used to solve the problem using that data.

- Data structures and algorithms (DSA) goes through solutions to standard problems in detail and gives you an insight into how efficient it is to use each one of them.

- It also teaches you the science of evaluating the efficiency of an algorithm. This enables you to choose the best of various choices.