Data Structures

lecture 4

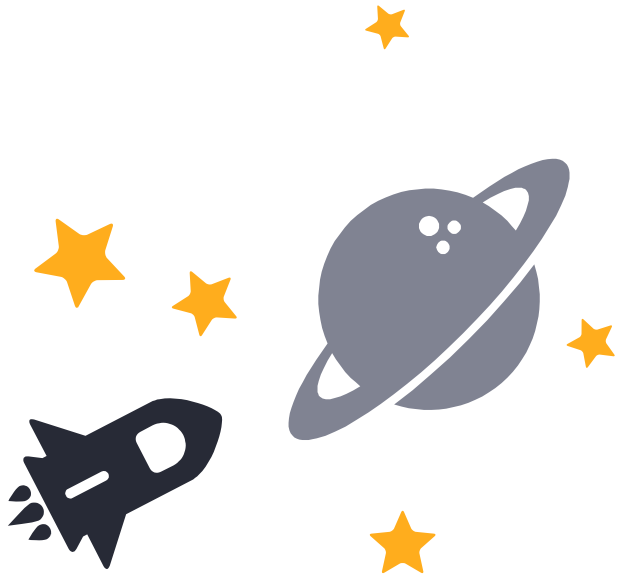
24-9-2022

Unit 1: Basics of Data Structures

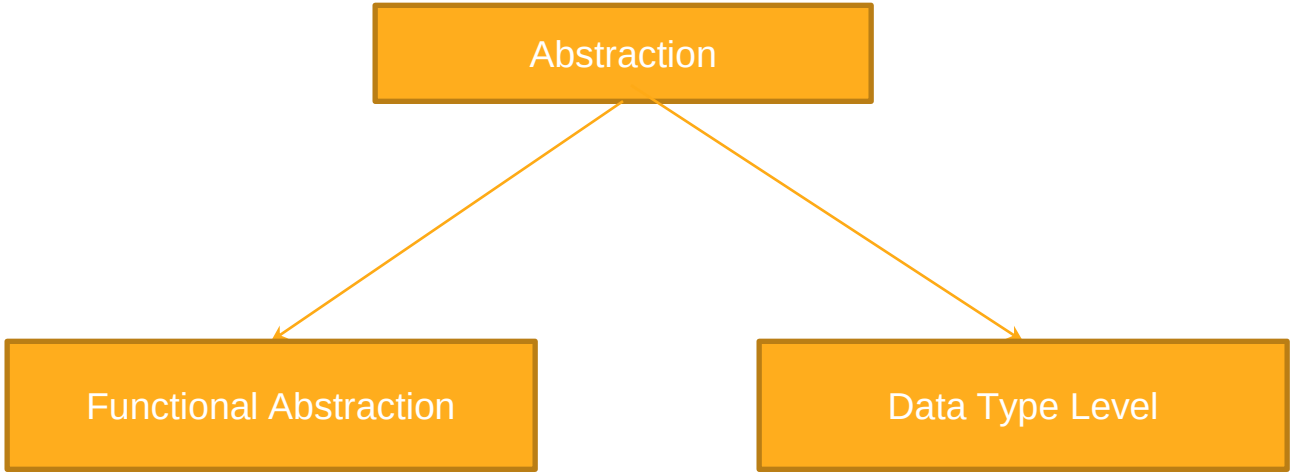
Abstraction

- **Abstract:** Something which focus on essential parts ignoring details.

- **Abstraction: something which is not in detail**



Abstraction in Programming



Functional Abstraction

- Every function performs some task.
- How task is performed implementation details hidden.
- Deals with **what a function does**
- **Not how** it does.

Data Type Level Abstraction

Atomic Data

- consist single piece of information.
- cannot be subdivided into other meaningful pieces.
- **Example:**
  - Some integer 457

Composite Data

- Opposite to Atomic
- can be subdivided into other meaningful pieces.
- **Example:**
  - Vehicle no, Phone No.
  - MH 09 PQ 2647

Data Type Level Abstraction

- Tells **what data type is**
- **what operations** can be performed on that data type
- Example:
  - int (supports bitwise operator)
  - float (does not support bitwise operator)

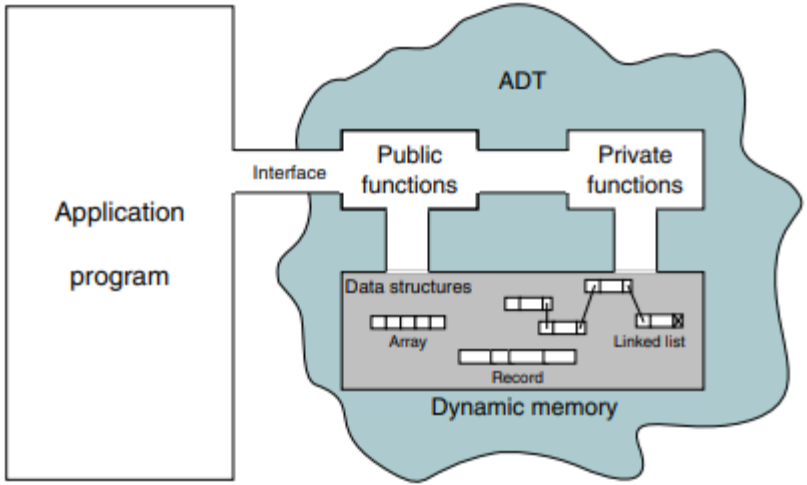
Abstraction in Programming tells us

About **what** a can be done

NOT How it can be done

Abstract Data Type (ADT)

ADT = Functional Abstraction + Data Type Level Abstraction



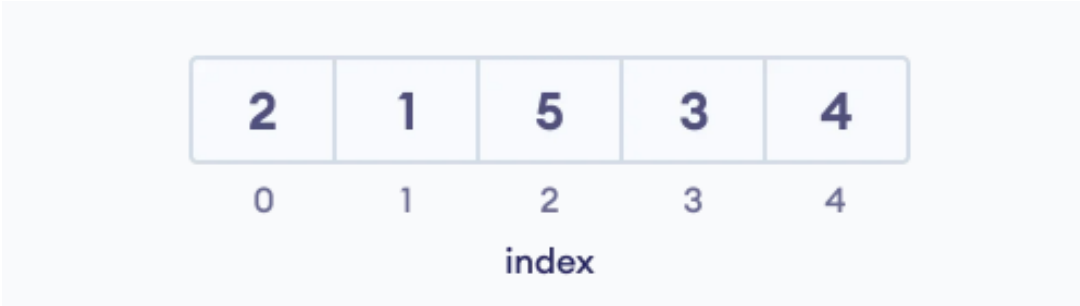
Types of Data Structures

- Basically, data structures are divided into **two categories**:
  - **Linear** data structure
  - **Non-linear** data structure

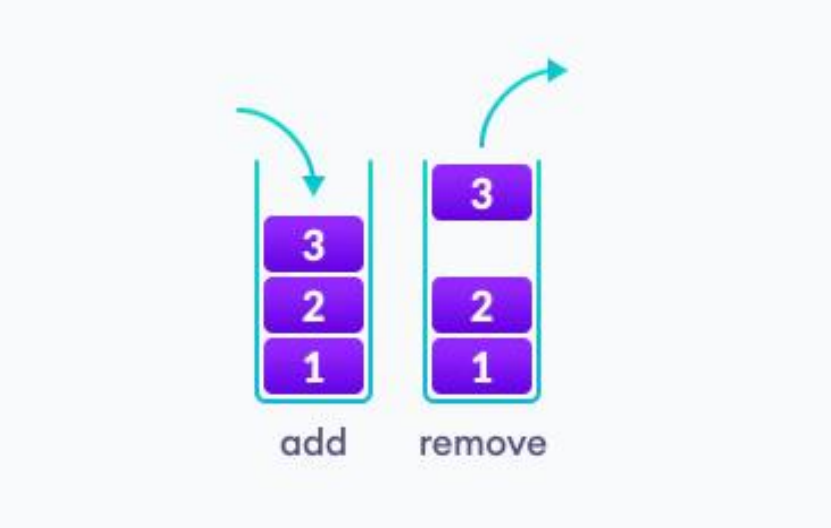
Linear Data Structures

- Elements are arranged sequentially
- every element is attached to its previous and next adjacent element
- **Example:**
  - Array
  - Stack
  - Queue
  - Linked List

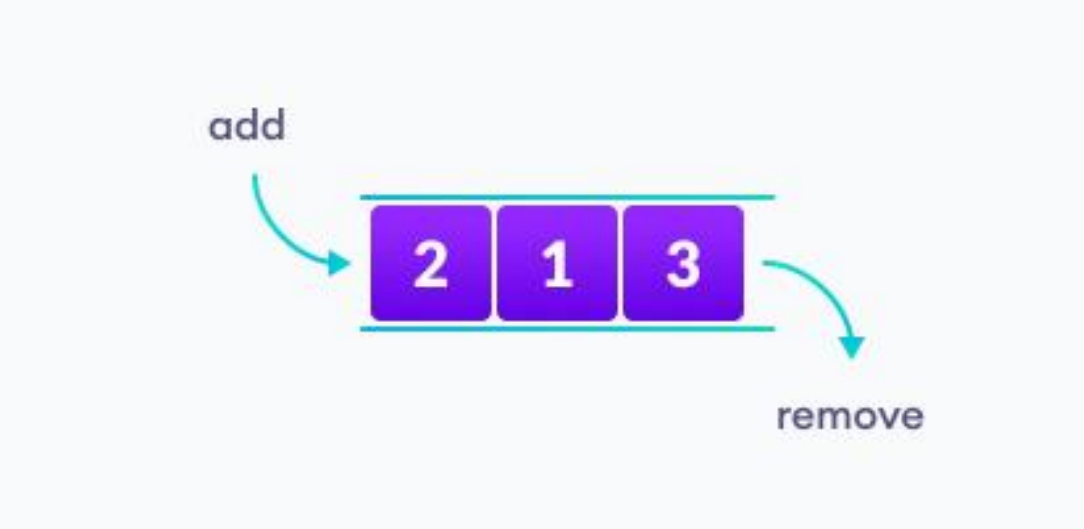
Array



Stack



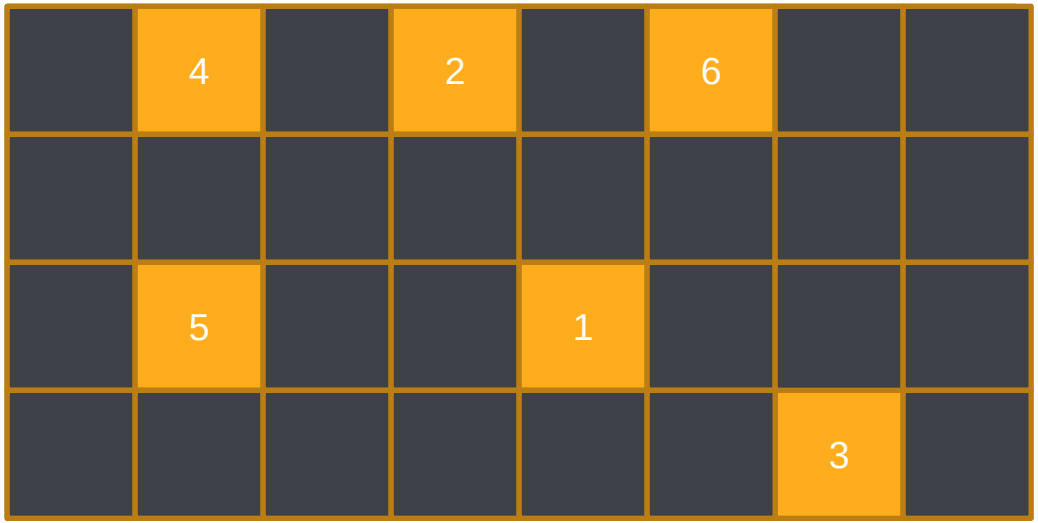
Queue



Linked List

Need of Linked List

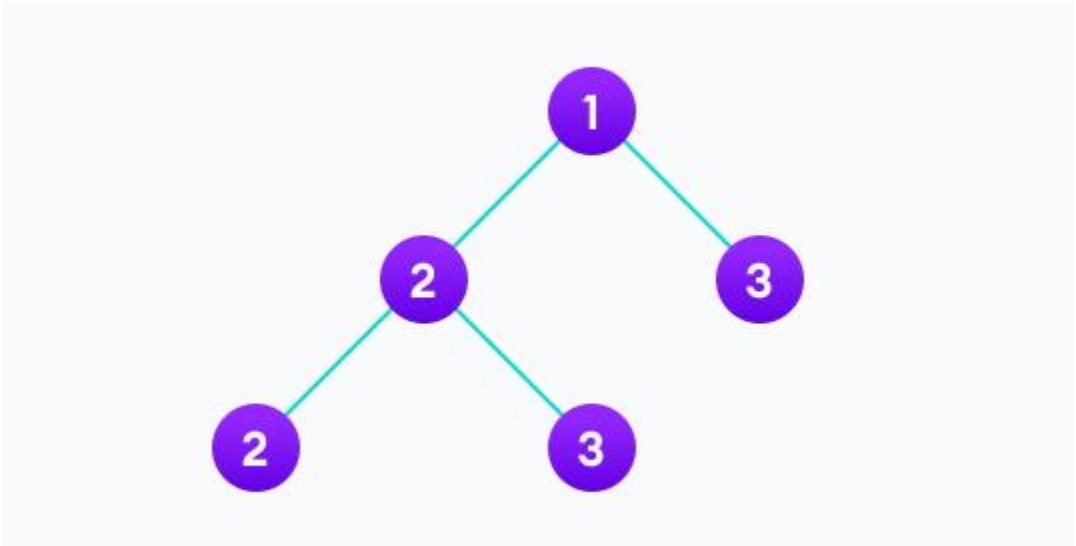
- Data: 4,2,6,5,1,3



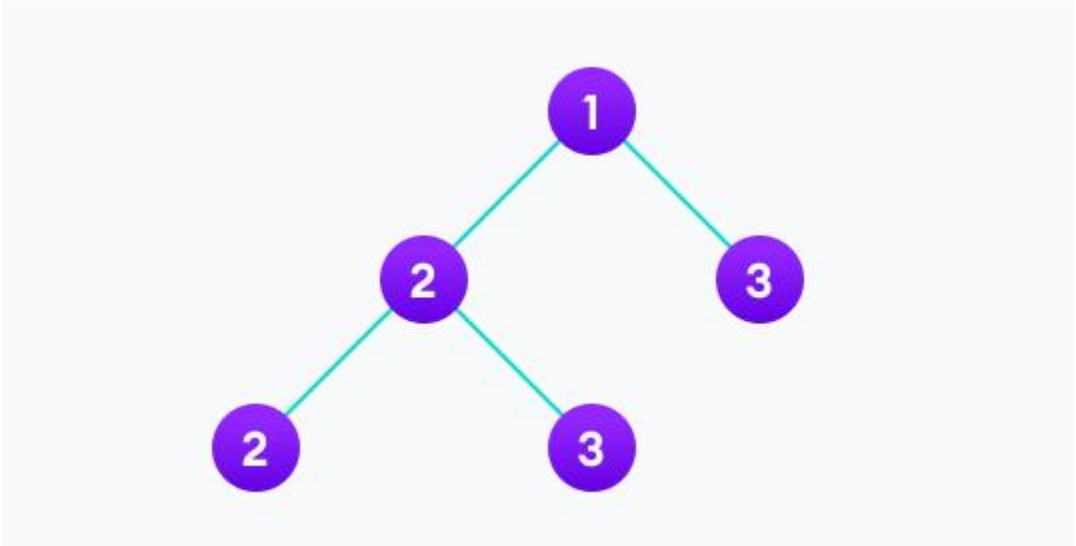
Non-Linear Data Structures

- Elements are not arranged sequentially.
- Every element is attached to more than one other elements.
- **Example:**
  - Tree
  - Graph

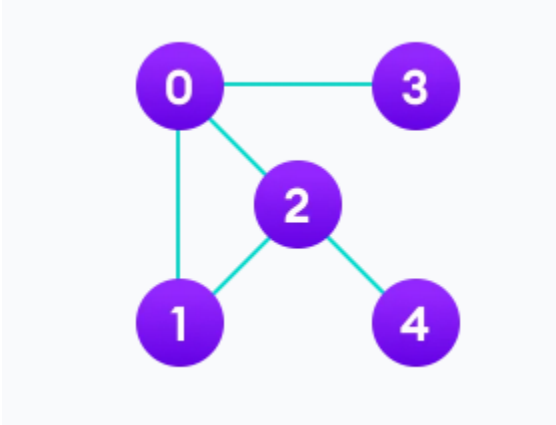
Tree



Tree



Graph



Quick Compare

Linear Data Structures	Non Linear Data Structures
arranged in sequential order, one after the other.	arranged in non-sequential order (hierarchical manner).
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 etc.	Example: Tree, Graph

Algorithm efficiency.

- There are more than one solutions (algorithms) to any problem
- We need to choose most efficient one.
- Major factor affecting efficiency is repetitive operations
  - Recursion / Loops

- Therefore Loops are very important in efficiency calculation

Efficiency

- Efficiency can be considered as mathematical function of the number of elements to be processed in loop

  - **f (n) = efficiency**