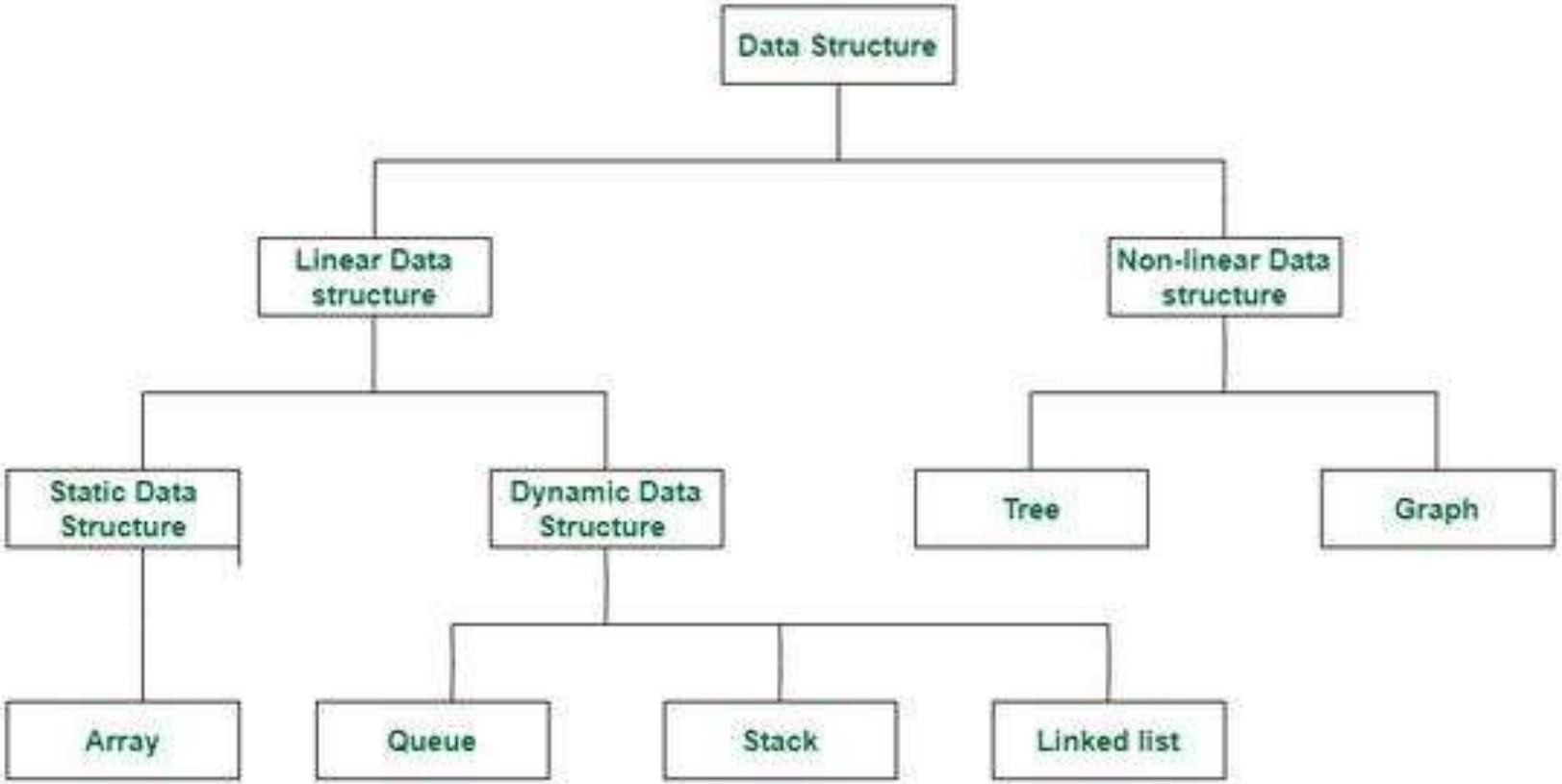
Data Structures

Let's dive in

- What is data structure ?
- Categories
  - Primitive data structure
  - Non primitive data structure
    - Linear
    - Non - linear

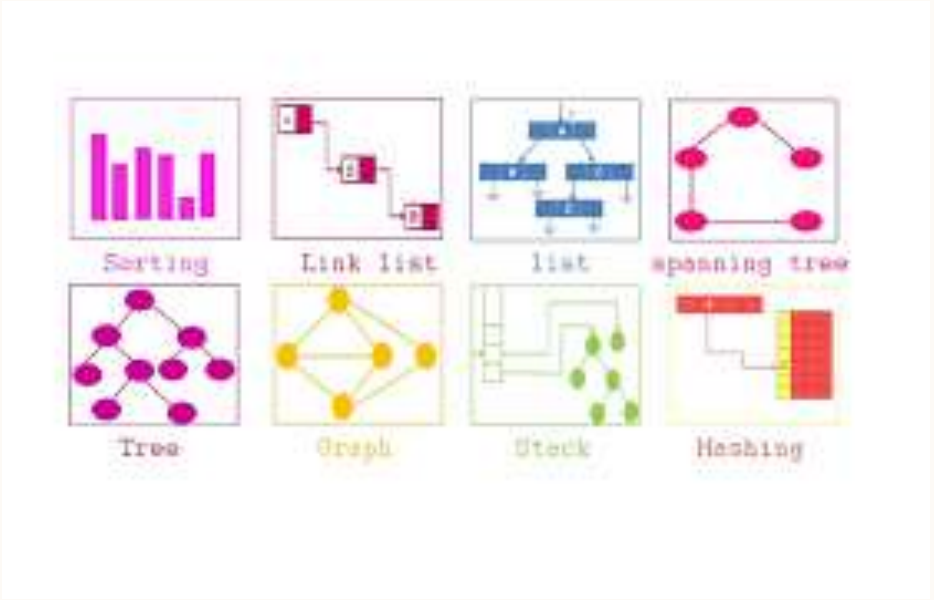
Classification of Data Structure



Types of operations

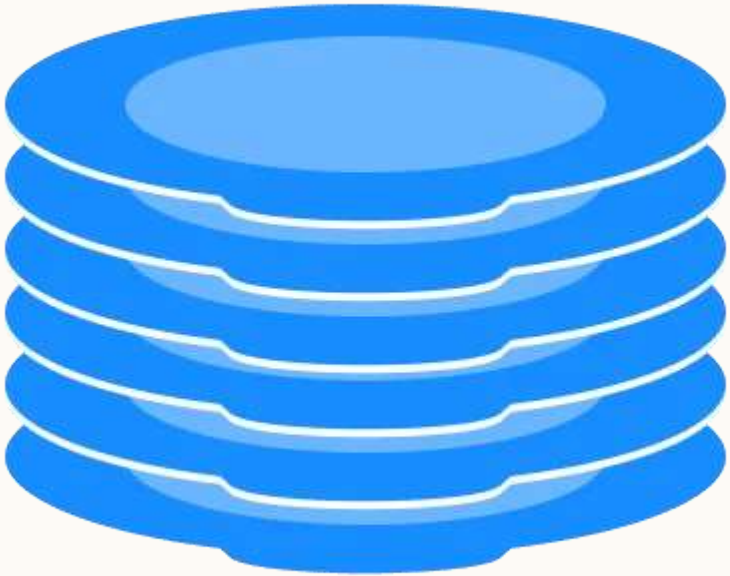
- Traversing
- Searching
- Insertion
- Deletion

Why data structures ?

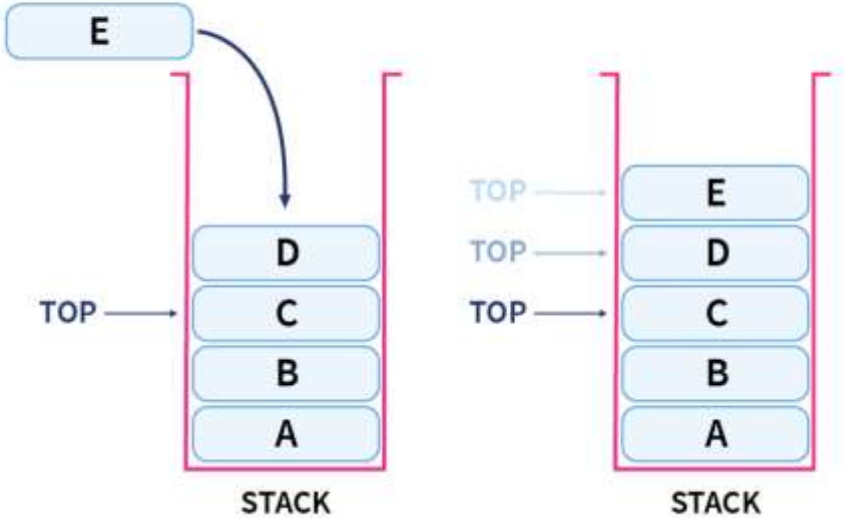


STACK

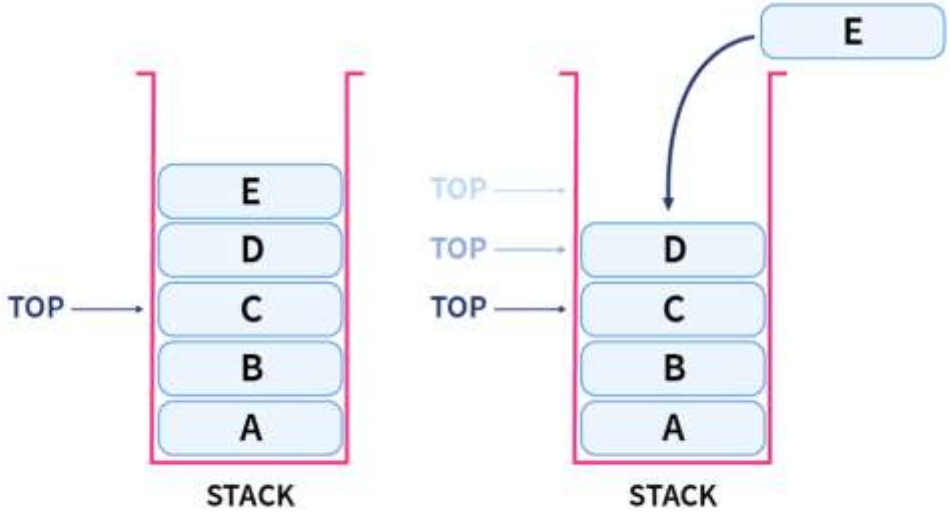
- LIFO
- Example: UNDO in editor



Push Operation

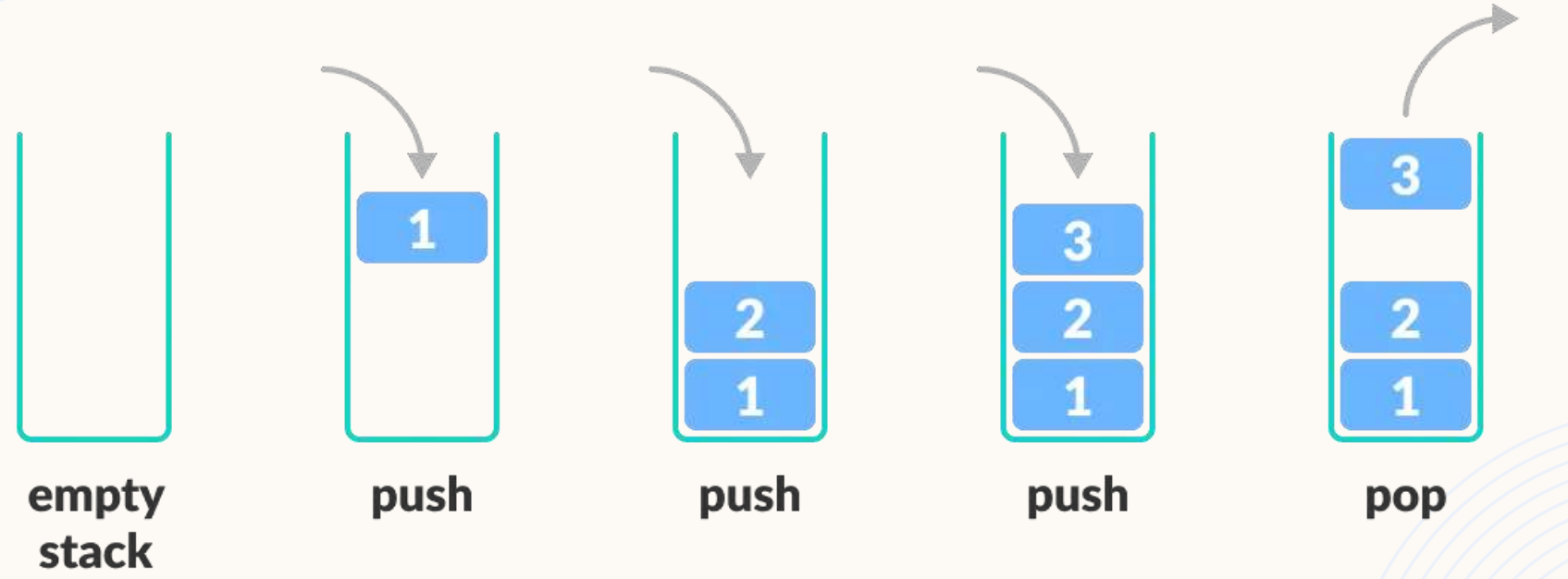


Pop Operation



METHODS IN STACK

- top()
- push()
- pop()
- size()
- peek()


TOP = -1
TOP = 0
stack[0] = 1
TOP = 1
stack[1] = 2
TOP = 2
stack[2] = 3
TOP = 1
return stack[2]
1
2
1
3
2
1
3
2
1
empty
stack
push
push
push
pop

IMPLEMENTATIONS

- list
- collections.deque
- queue.LifoQueue

ADVANTAGES and APPLICATIONS

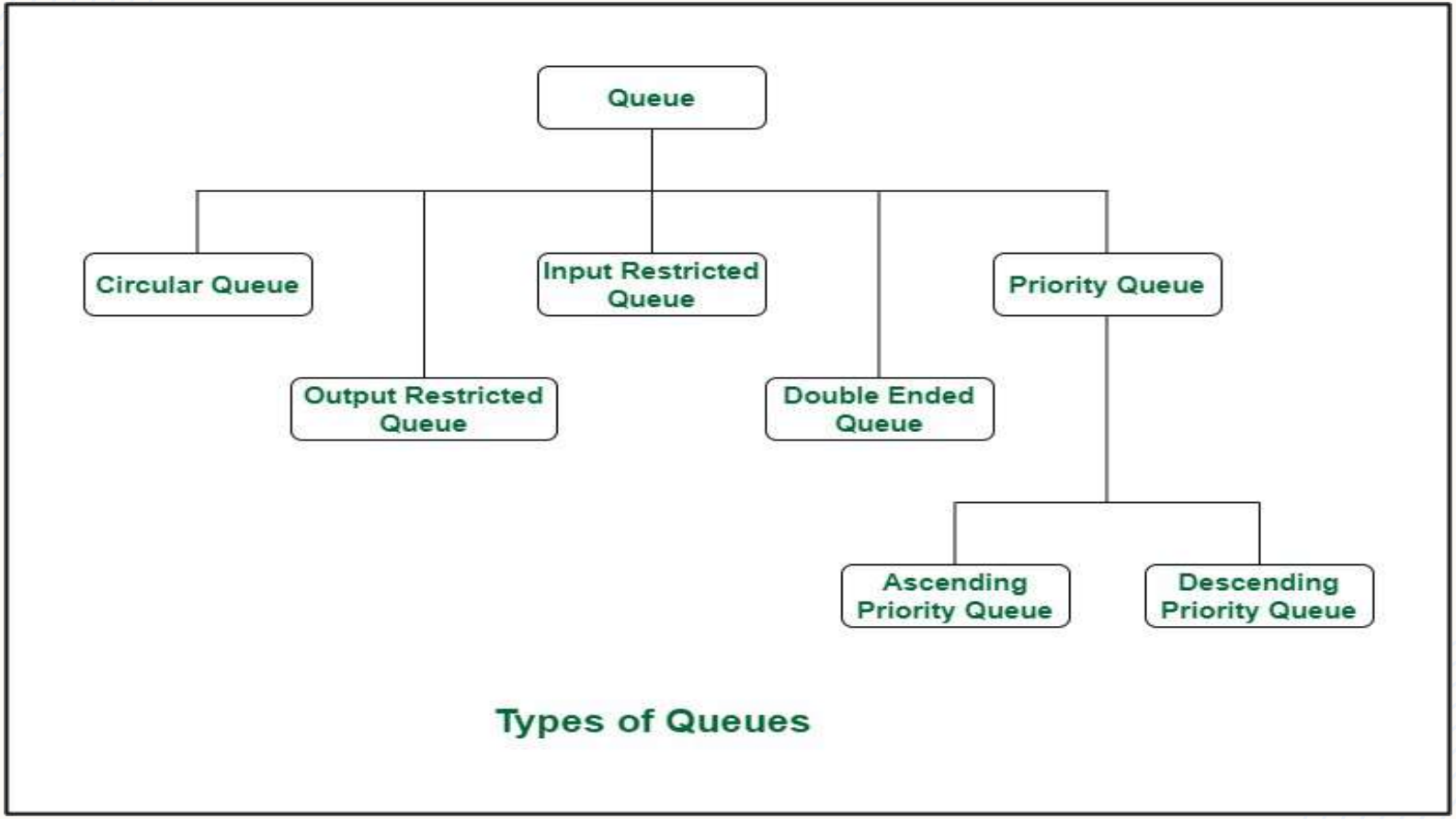
- Easy implementation
- Efficient memory utilization
- Fast access time
- Helps in function calls
- Supports backtracking
- Used in Compiler Design
- Enables undo/redo operations

DISADVANTAGES

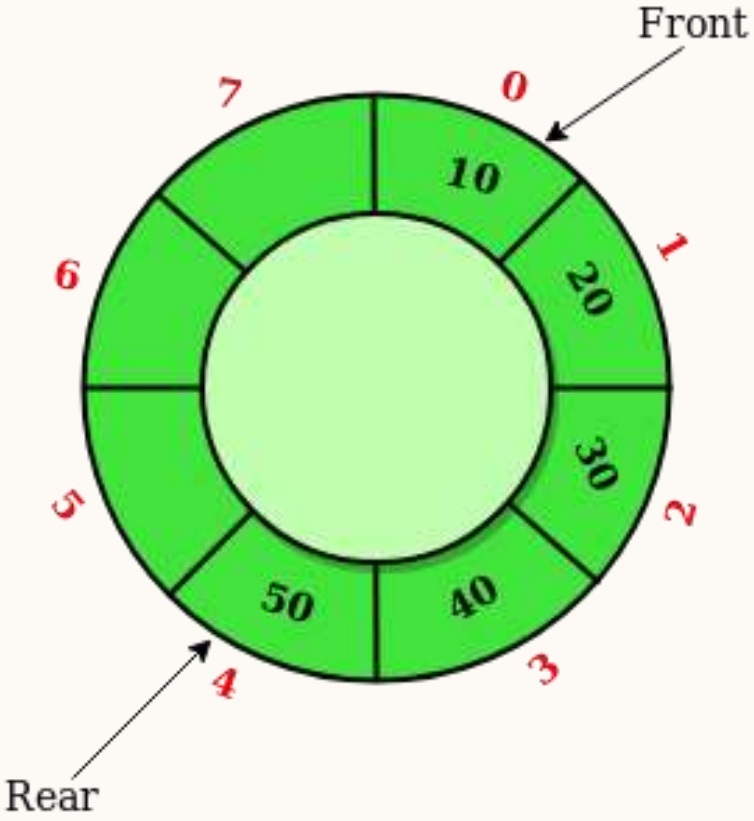
- Limited capacity
- No random access
- Memory management
- Not suitable for certain applications
- Stack overflow and underflow
- Recursive function calls limitations

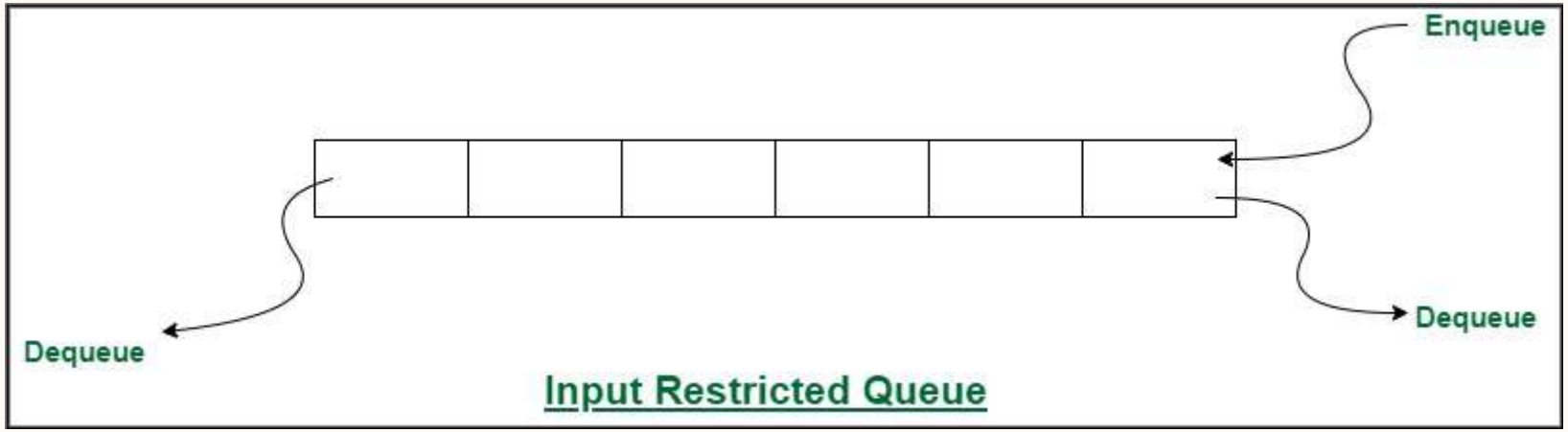
QUEUE

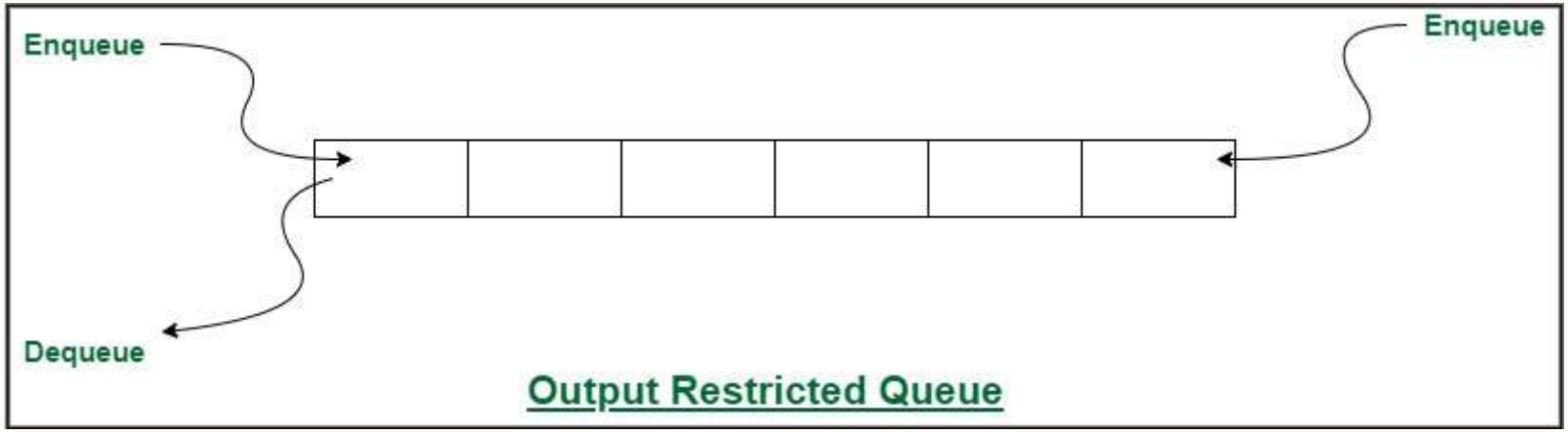
- FIFO
- Front and Rear

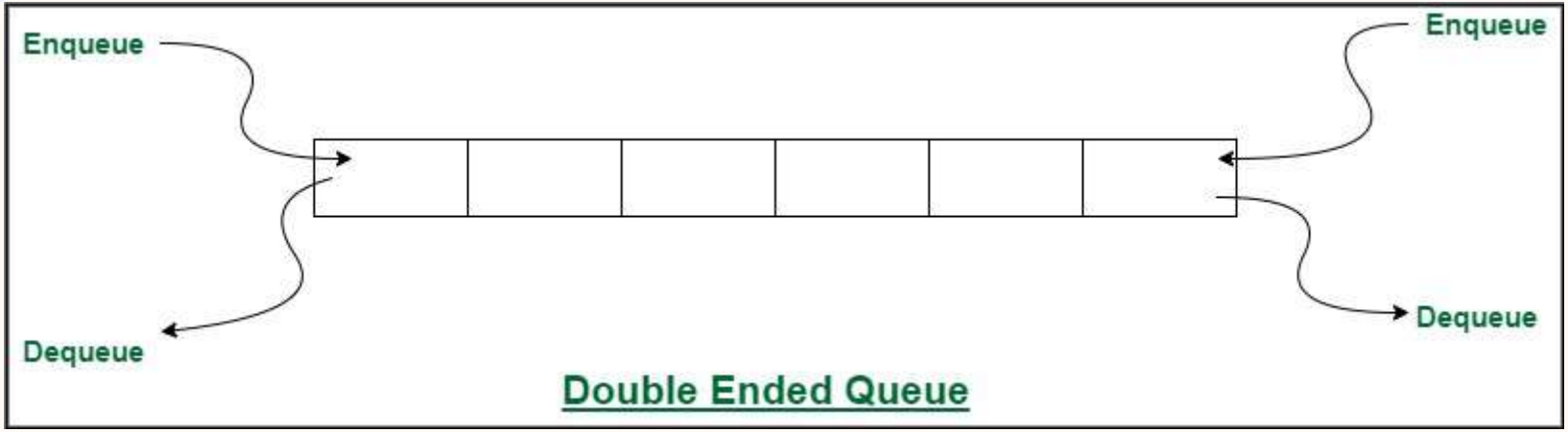


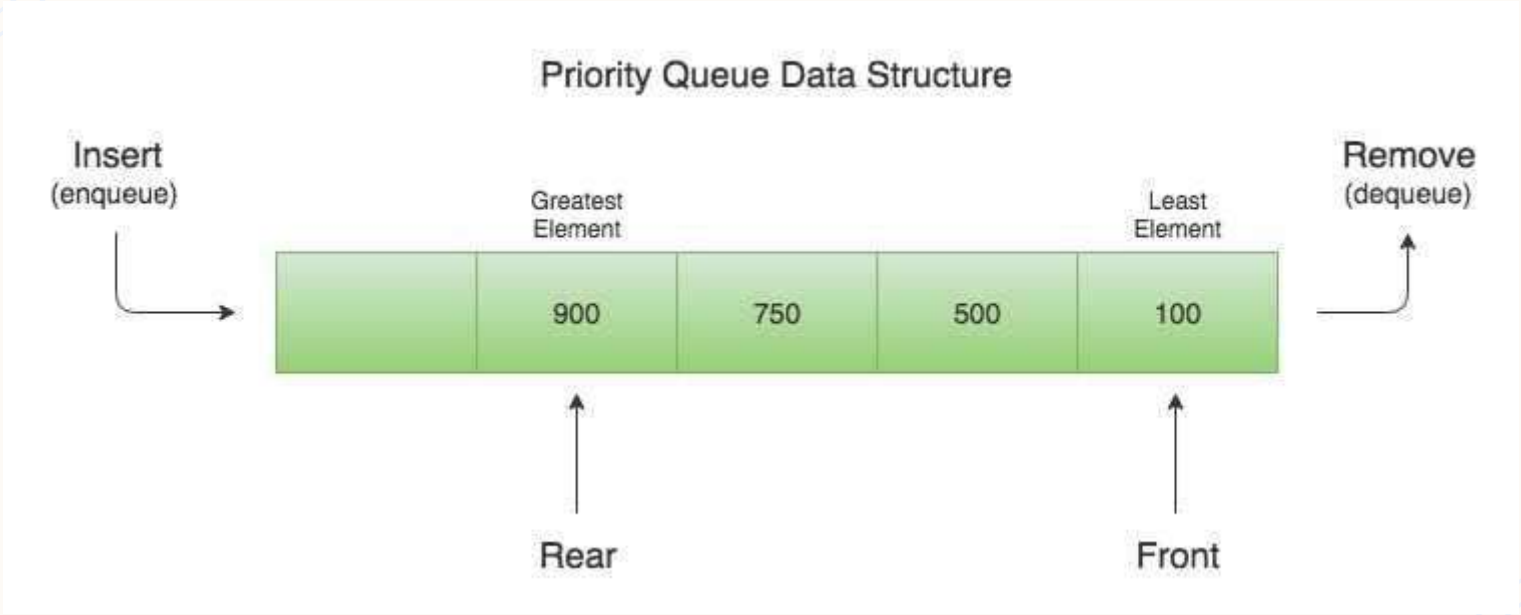
Types of Queues


Front
Rear
0
1
2
3
4
5
6
7
10
20
30
40
50


Enqueue
Dequeue
Dequeue
Input Restricted Queue


Enqueue
Enqueue
Dequeue
Output Restricted Queue


Enqueue
Enqueue
Dequeue
Dequeue
Double Ended Queue


Priority Queue Data Structure
Insert
(enqueue)
Remove
(dequeue)
Greatest
Element
Least
Element
900
750
500
100
Rear
Front

OPERATIONS

- Enqueue
- Dequeue
- Front
- Rear

METHODS

- put(item)
- get()
- empty()
- qsize()
- full()
- maxsize()

IMPLEMENTATION

- Using python lists
- Using queue module
- Usings the collections.deque module

ADVANTAGE

- managed efficiently with ease.
- first in first out rule.
- multiple consumers.
- fast in speed
- implementation of other data structures.

DISADVANTAGE

- Time consuming
- Classical queue
- Defined size

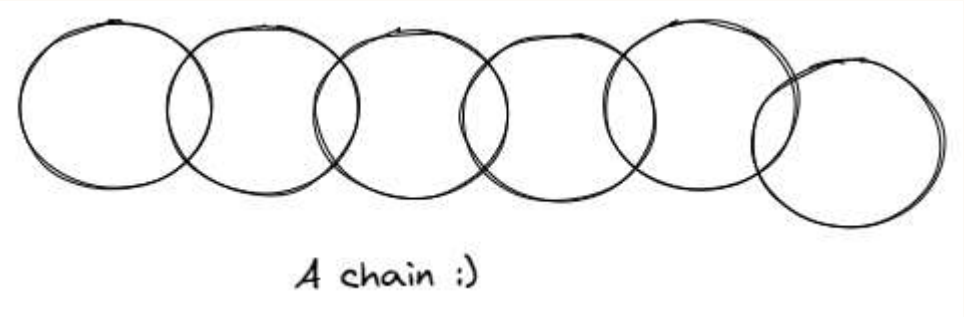
APPLICATIONS

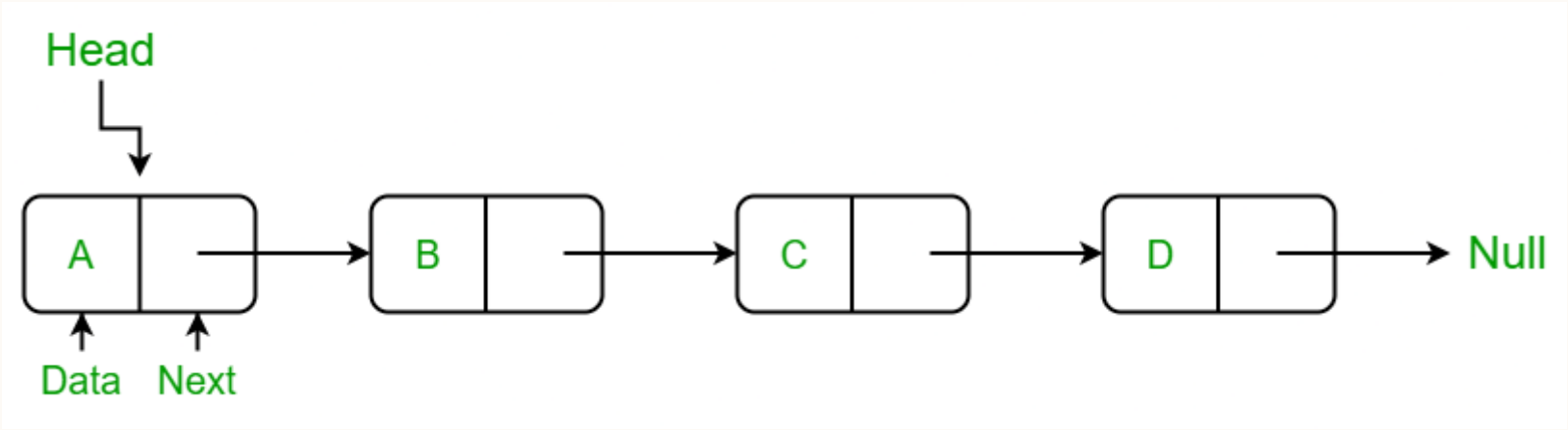
- Multi programming
- Network
- Job Scheduling
- Shared resources

REAL - LIFE APPLICATIONS

- Keyboard and CPU
- ATM Booth Line
- Ticket Counter Line
- CPU task scheduling
- Waiting time of each customer at call centers.

LINKED LIST

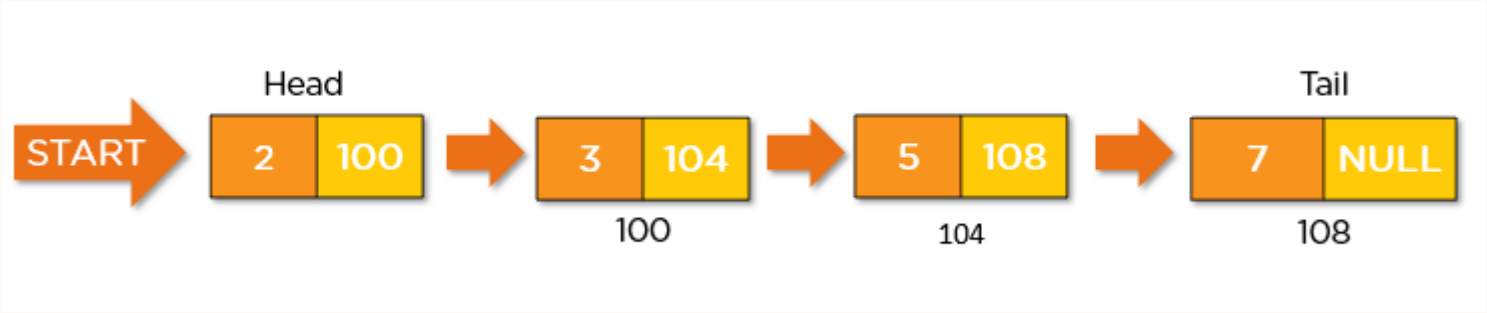



Head
A
B
C
D
Null
Data
Next

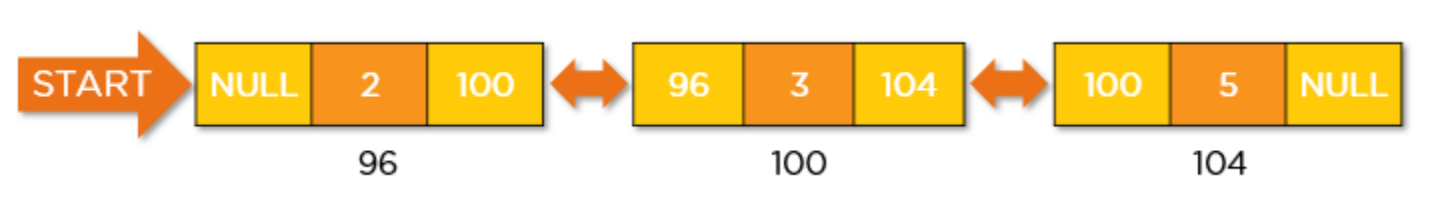
LINKED LIST

- Singly linked lists
- Doubly linked lists
- Circular linked lists
- Circular doubly linked lists

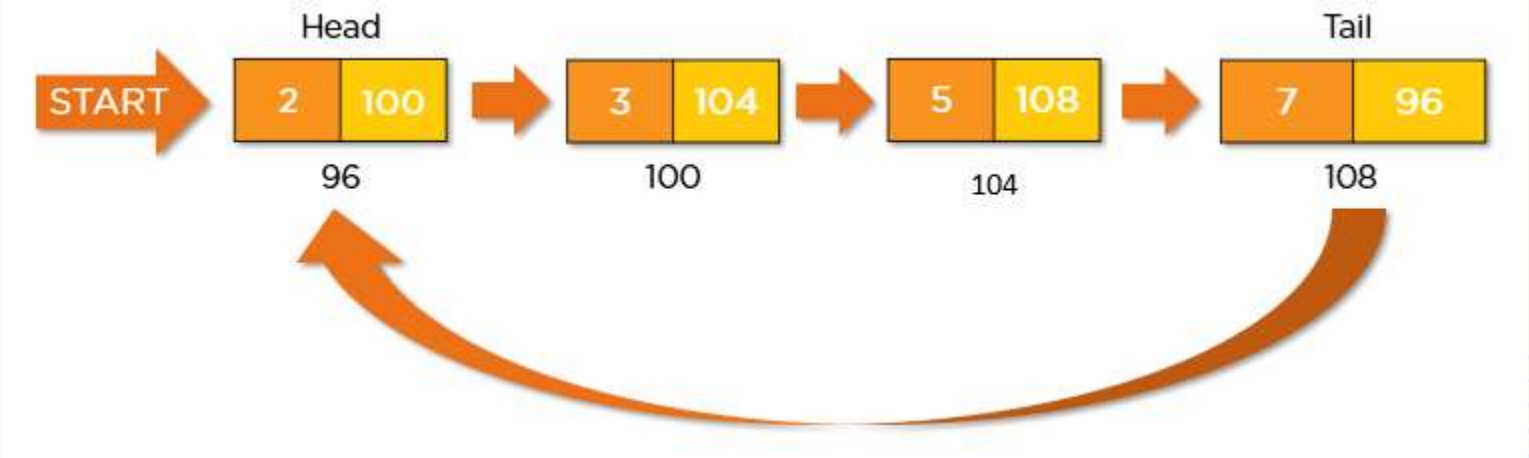
Singly Linked List



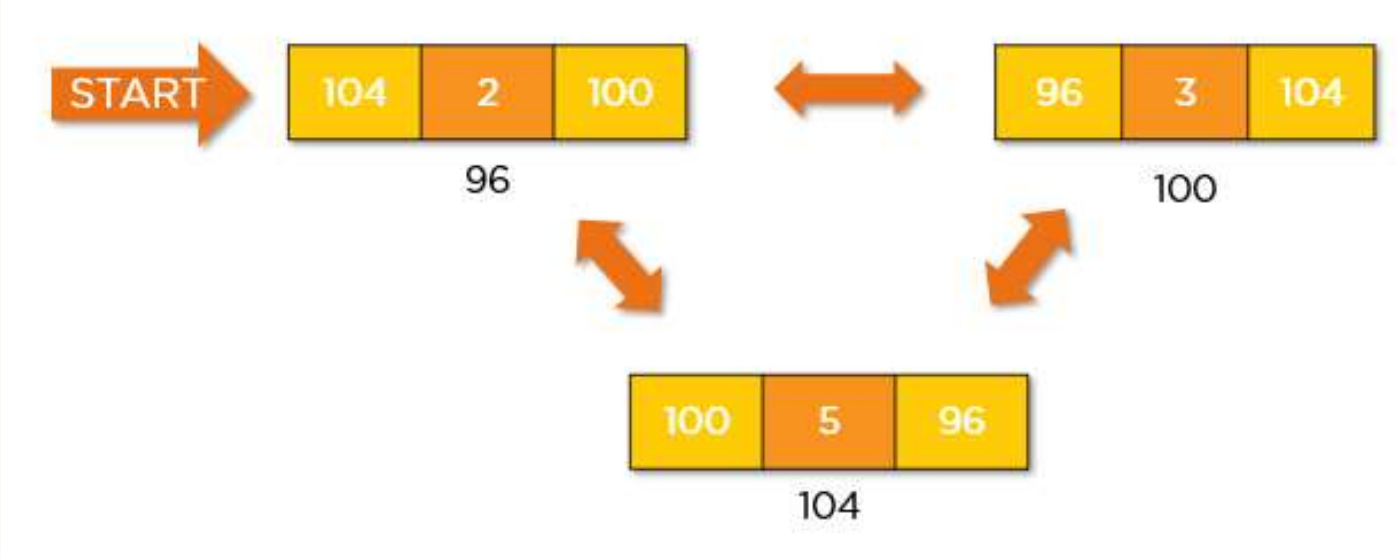
Doubly Linked List

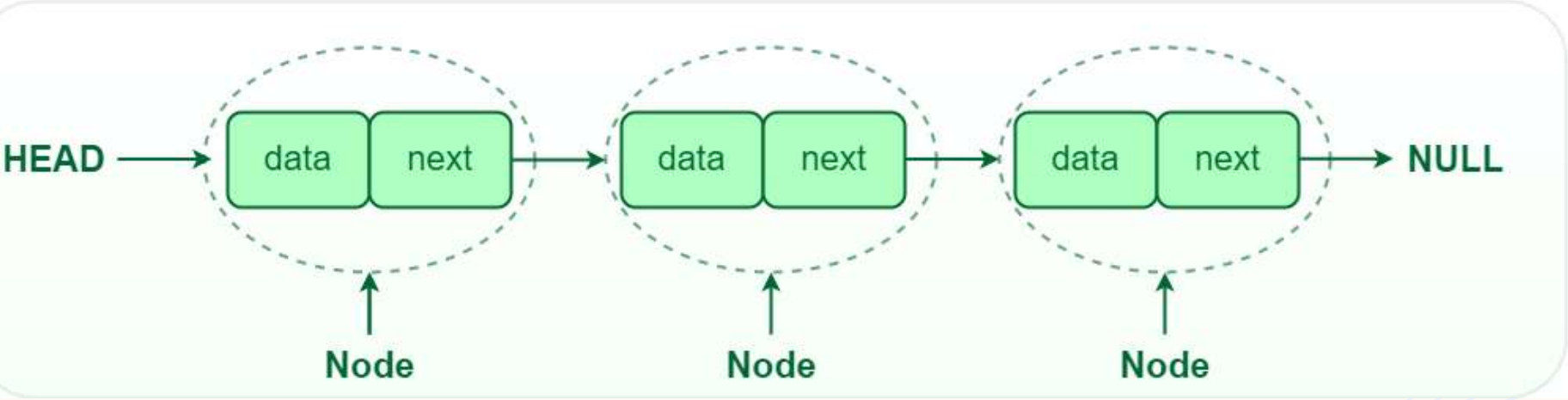


Circular Linked List



Circular Doubly Linked List




HEAD
data
next
data
next
data
next
NULL
Node
Node
Node

ADVANTAGES

- Dynamic Size
- Insertion and Deletion
- Flexibility

DISADVANTAGES

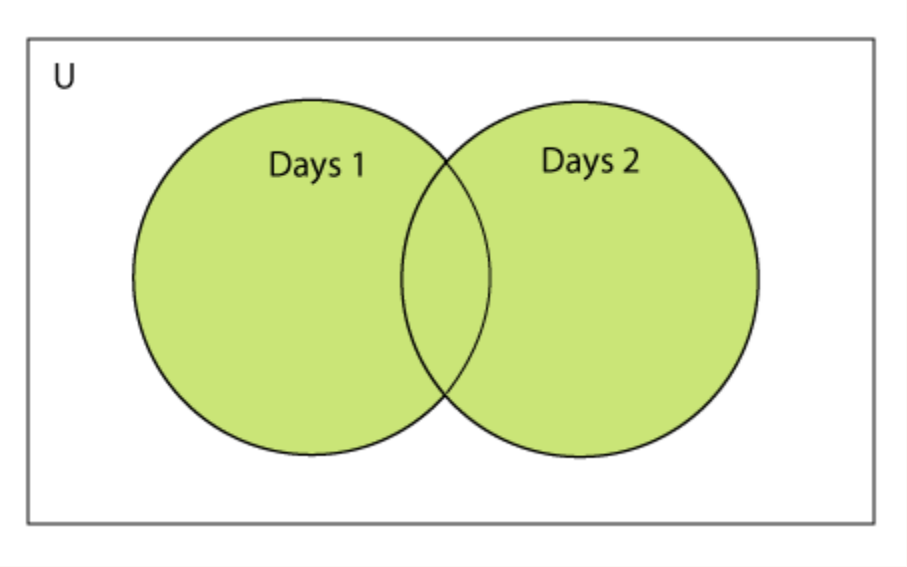
- Random Access
- Extra Memory

SETS

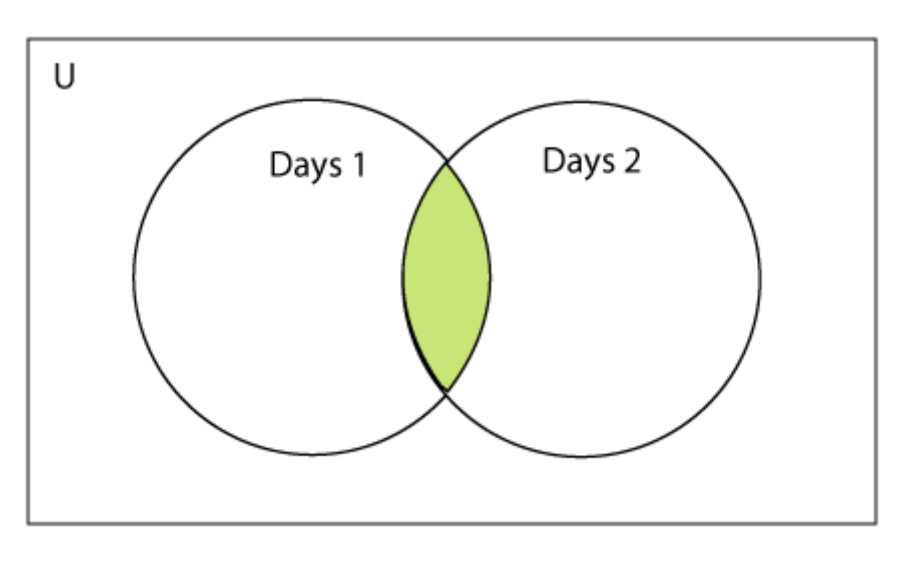
- What is a set ?
  - Unordered
  - Unindexed
  - Mutable
  - Iterable

- Applications
  - Removing duplicates
  - Checking set membership
  - Performing operations

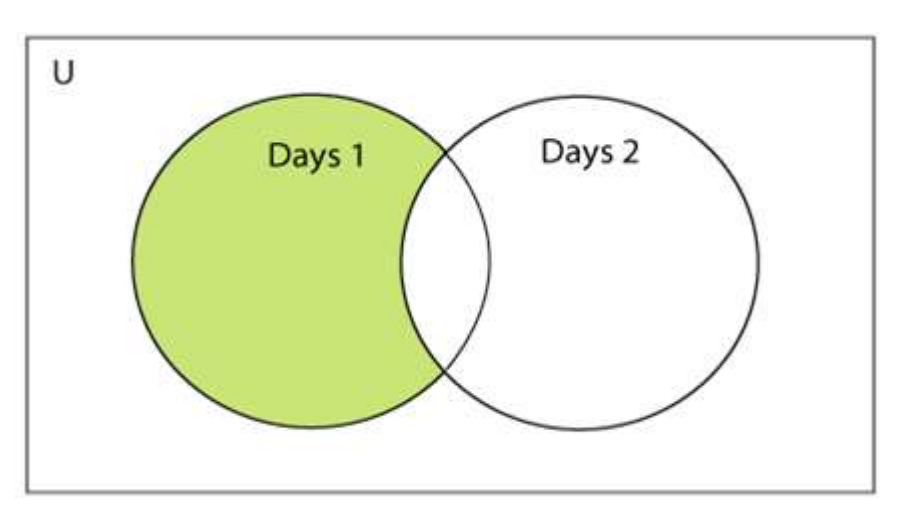
Union of two sets



Intersection



Difference



Real Time Applications

- Music player app
- Distinct values in an array
- Detect cycles in a Linked List
- Database operations

ADVANTAGES

- Unique values
- Efficient
- Dynamic
- Fast and efficient operations
- Implemented using different data structures
- Improve performance

DISADVANTAGES

- No indexing
- Complex to implement
- Large data sets
- Specific data type
- More memory

ARRAY

- Contiguous memory locations
- Fixed number of items
- Same type
- Comparison: Fleet of stairs
- Terms :
  - Elements
  - Index

ARRAY

- Index starts from 0
- Length defines capacity

OPERATIONS

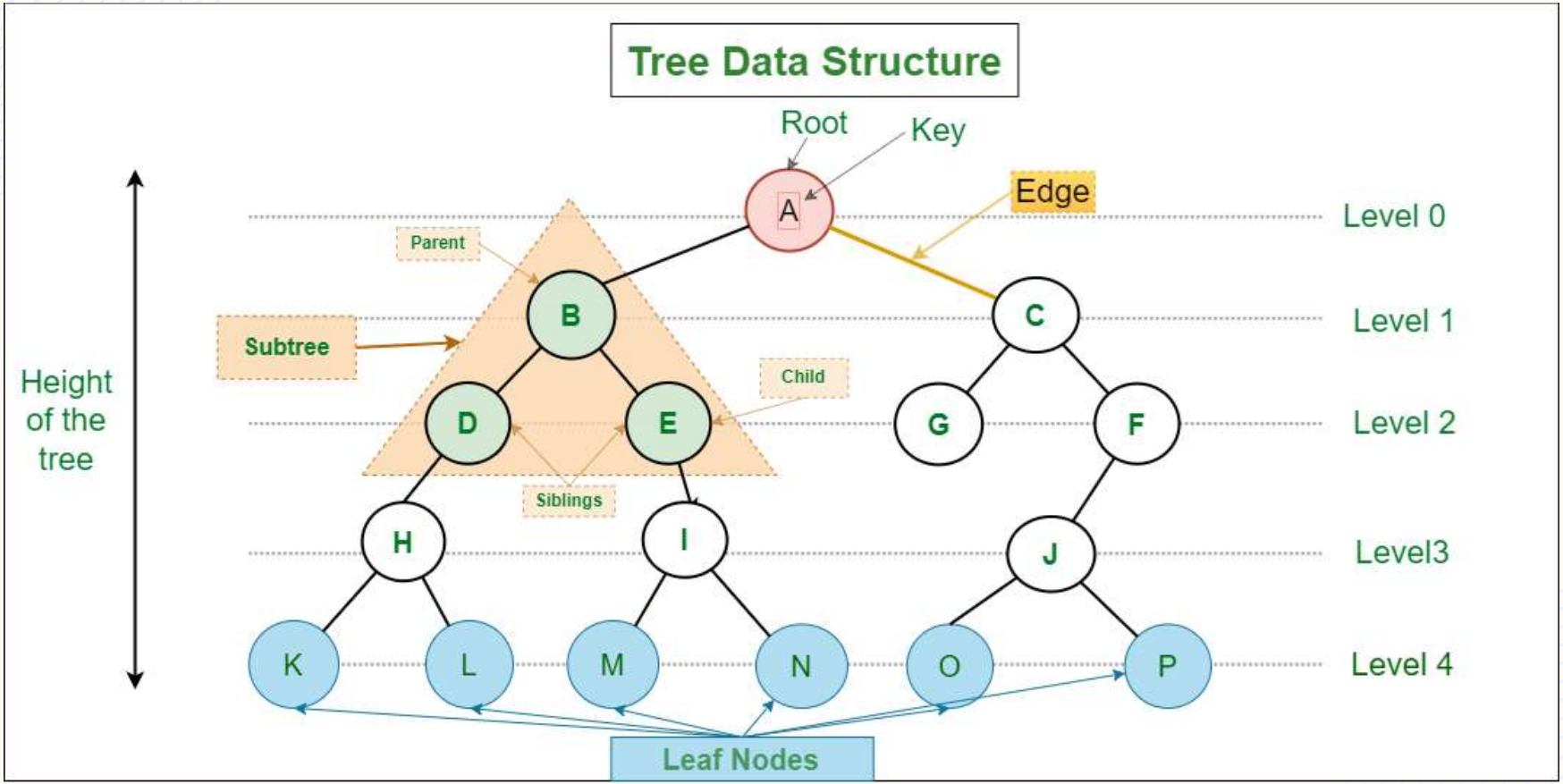
- Traverse
- Insertion
- Deletion
- Search
- Update

OPERATIONS

- Traverse
- Insertion.
- Deletion
- Search
- Update

TREES

- Hierarchical manner
- Root node
- Zero or more nodes as children


Tree Data Structure
Root
Key
Edge
A
Parent
B
C
Subtree
Child
D
E
G
F
Siblings
H
I
J
K
L
M
N
O
P
Leaf Nodes
Height of the tree
Level 0
Level 1
Level 2
Level3
Level 4

BINARY TREES

- 2 children per node
- Left and right child

Why TREES ?

- Hierarchy
- Moderate access or search
- No upper limit

APPLICATIONS

- Data storage and retrieval
- Expression evaluation
- Network routing
- Game AI
- Algorithms

BASIC OPERATIONS

- Inserting an element
- Removing an element
- Searching for an element
- Traversing the tree

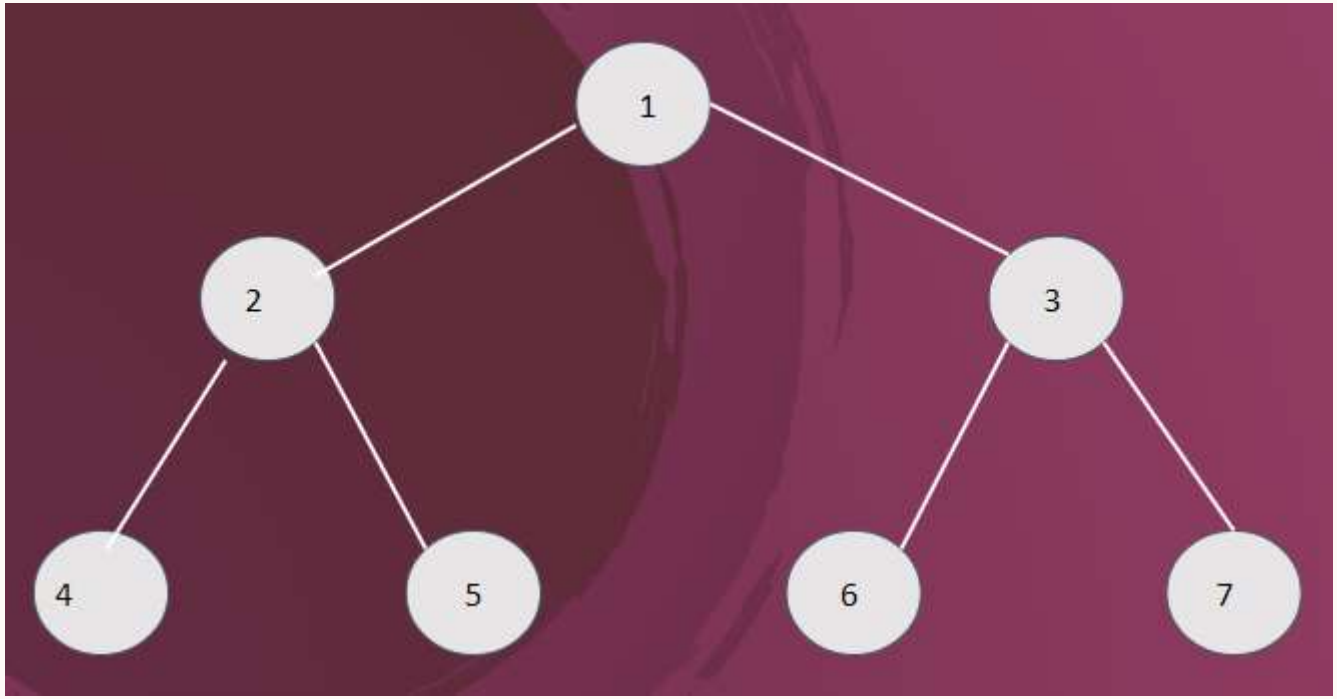
AUXILIARY OPERATIONS

- Finding the height of the tree
- Find the level of a tree
- Finding the size of the entire tree

BINARY TREE TRAVERSAL

- Depth-First Search (DFS) Algorithms
  - Preorder Traversal
  - Inorder Traversal
  - Postorder Traversal

- Breadth-First Search (BFS) Algorithms
  - Level order Traversal


1
2
3
4
5
6
7

GRAPHS

- Pictorial representation
- Objects are connected by links
- Vertices, V
- Edges, E

BASIC OPERATIONS

- Display graph vertices
- Display graph edges
- Add a vertex
- Add an edge
- Creating a graph

Types

- Null graph
- Trivial graph
- Undirected graph and Directed graph
- Connected graph and Disconnected graph
- Regular graph
- Complete graph
- Cycle and cyclic graph
- Directed Acyclic graph
- Bipartite graph
- Weighted graph

REPRESENTATION

- Adjacency matrix
- Adjacency list

NetworkX

- Software package
  - Creation
  - Manipulation
  - Study of the structure
  - Dynamics
  - Function of complex networks.

Matplotlib

- Comprehensive library
  - Creating static
  - Animated
  - Interactive visualizations in Python.