Intro to Data Structures and Algorithms

(DS&A)

BeaconFire

What are Data Structures?

- Data Structures are used to store, organize, and manage data in specific formats enabling efficient access and modification
- It is a collection of stored values, the relationships between those values, and the functions or methods which can be applied to the values or structure.

Common Data Structures

Node

Queue

Stack

Binary Tree

Trees

Graphs

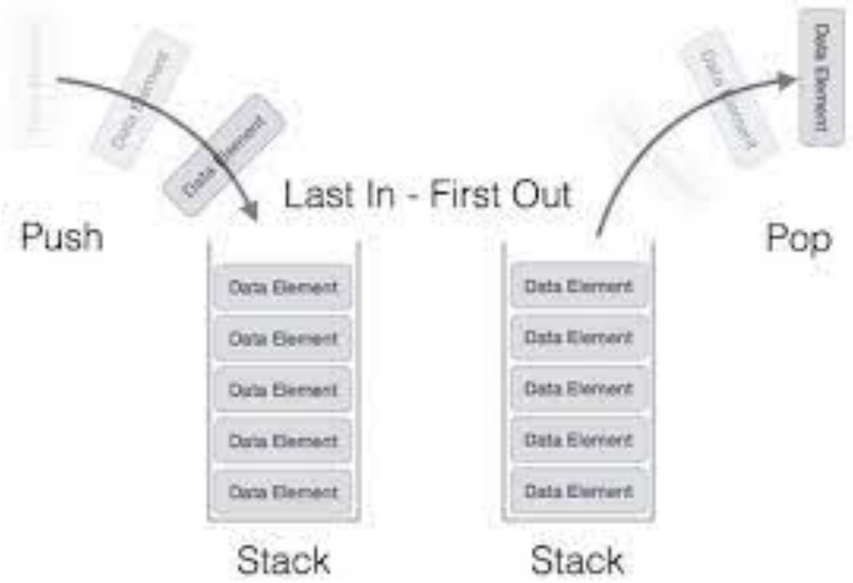
Linked Lists

Node

- Most basic data structure
- Building block of other data structures
- Can contain data and pointers to other nodes

Stack

- LIFO Principle - Last In, First Out
- Linear data structure
- Has a top and a bottom
- Items may only be added or removed from the top

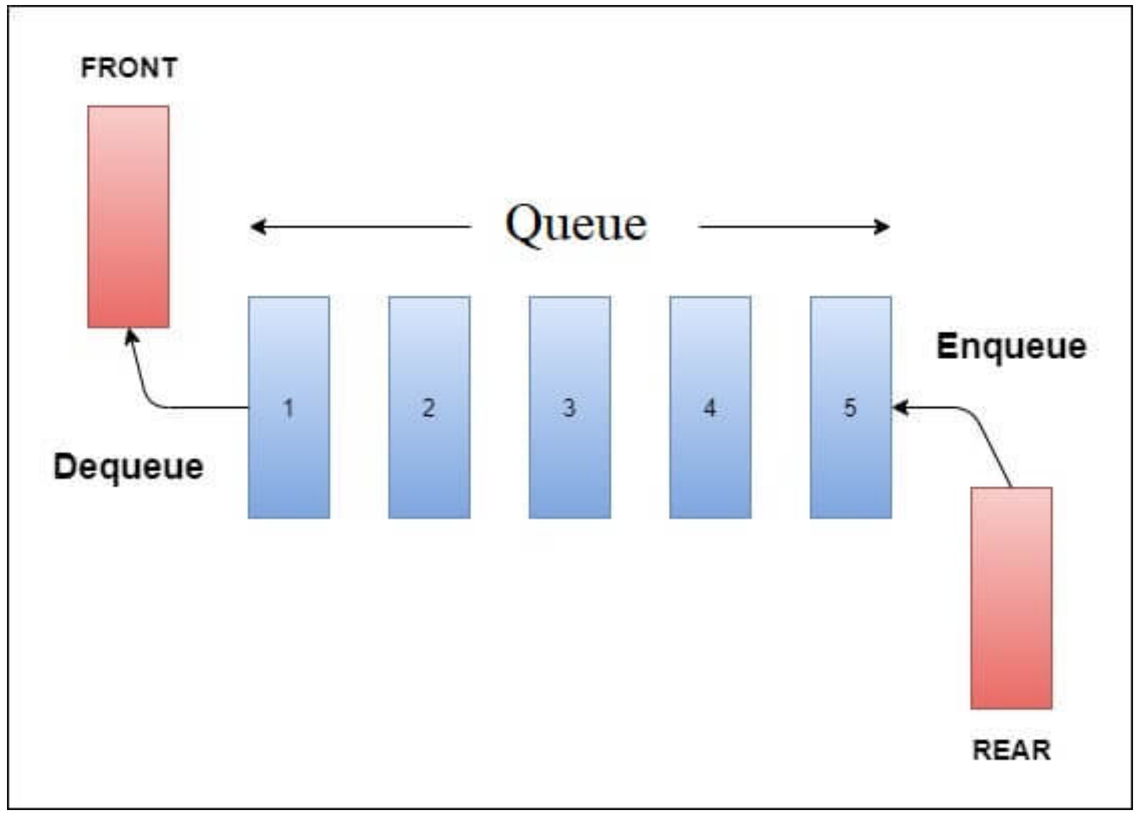

Last In - First Out
Push
Pop
Data Element
Data Element
Data Element
Data Element
Data Element
Stack
Data Element
Data Element
Data Element
Data Element
Data Element
Stack

Common Stack Methods and Functions

- Push - adds a new element to the top of the stack
- Pop - removes and returns the top element of the stack
- Size - returns the length of the stack
- Peek - returns the top element of the stack but does not remove it

Queue

- FIFO Principle - First In, First Out
- Linear data structure
- Has a front and back
- Items may only be removed from the front
- Items may only be added to the back


FRONT
Queue
1
2
3
4
5
Enqueue
Dequeue
REAR

Common Queue Methods and Functions

- Enqueue - adds a new element to the back of the queue
- Dequeue - removes the first element (front) of the queue
- Size - returns the length of the queue

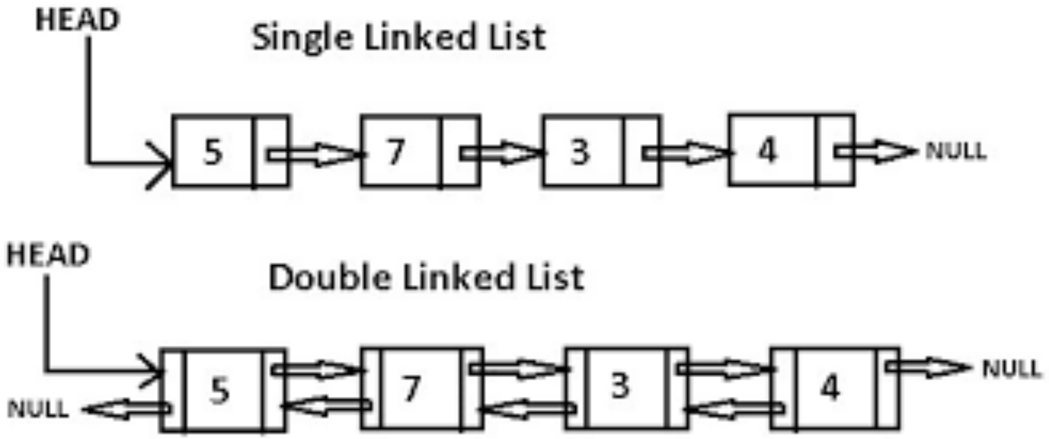
Linked Lists

Singly Linked List

- Constructed with Nodes, having a one way reference chain
- Linear data structure

Doubly Linked List

- Constructed with Nodes, having a two way reference chain
- Linear data structure

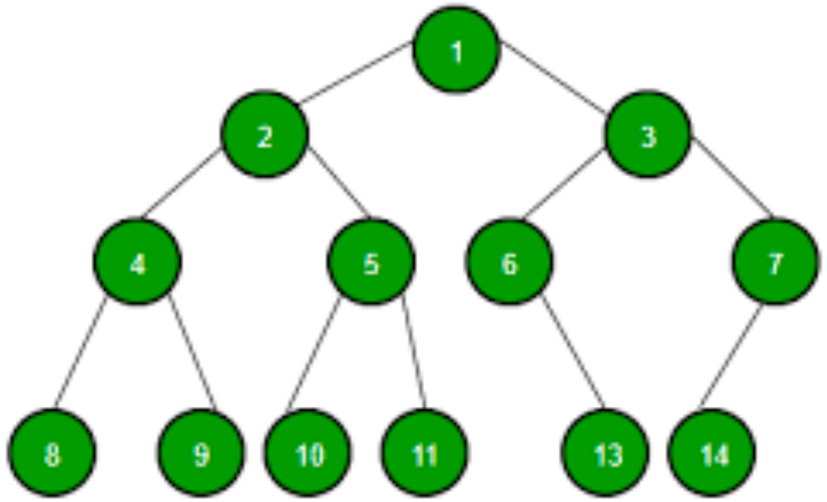

HEAD
Single Linked List
5
7
3
4
NULL
HEAD
Double Linked List
NULL
5
7
3
4
NULL

Common Linked List Methods and Functions

- Append - adds a new Node to the end of the list
- Prepend - adds a new Node to the beginning of the list
- Insert - adds a new Node at a given position (index) of the list
- Remove - removes the first Node with the given value
- NodeAtIndex - returns the Node at the given index
- Print - prints the list
- Size - returns the length of the list

Binary Tree

- Constructed with Nodes
- Each Node may have at most two children
- Tree structure (non-linear)
- Starting Node is referred to as the ‘root node’
- Nodes having no children are referred to as ‘leaf nodes’


1
2
3
4
5
6
7
8
9
10
11
13
14

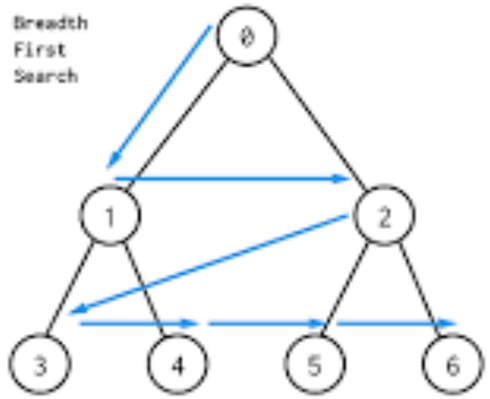
DFS vs BFS

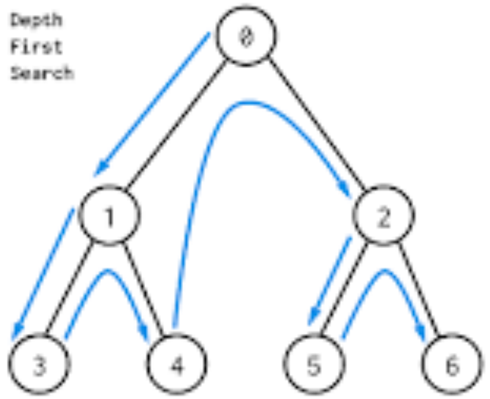
Depth First Search

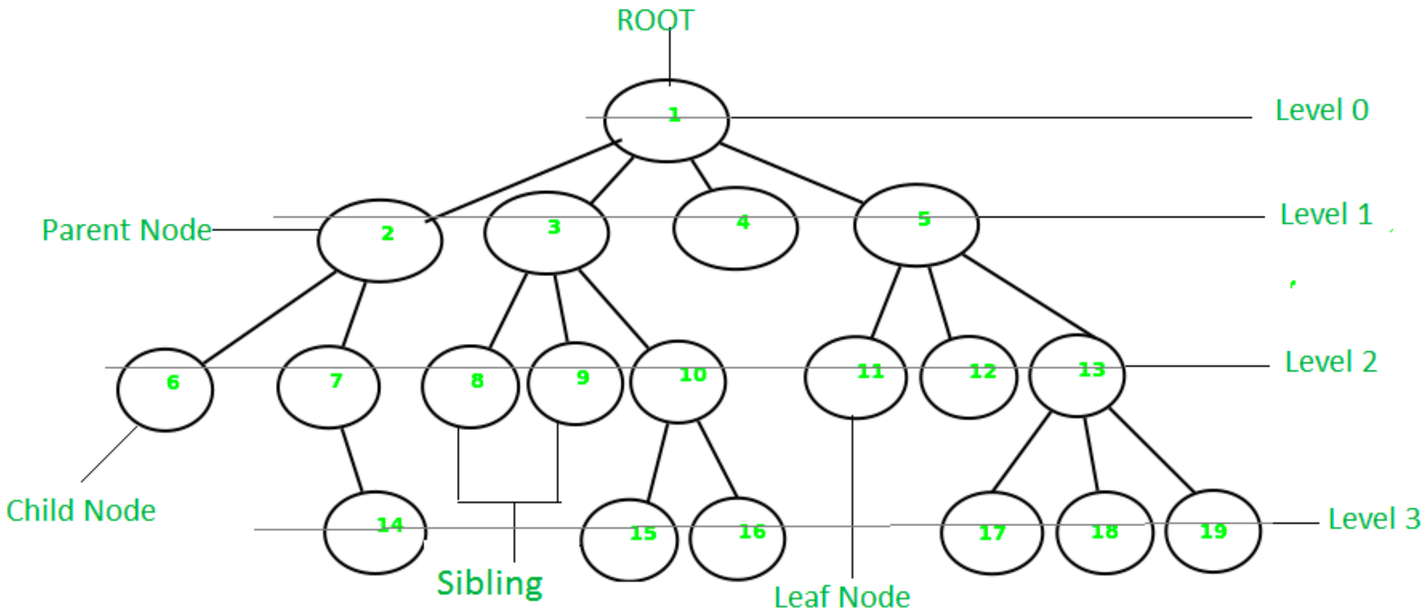
- An algorithm to traverse trees
- Starts from the root node and visits all possible routes to reach leaf nodes

Breadth First Search

- An algorithm to traverse trees
- Starts att the root node and visits all nodes within the same level (sibling nodes) before moving on to the next level


Breadth
First
Search
0
1
2
3
4
5
6


Depth
First
Search
0
1
2
3
4
5
6


ROOT
1
Level 0
Parent Node
2
3
4
5
Level 1
6
7
8
9
10
11
12
13
Level 2
Child Node
14
15
16
17
18
19
Level 3
Sibling
Leaf Node

Questions, comments, concerns?