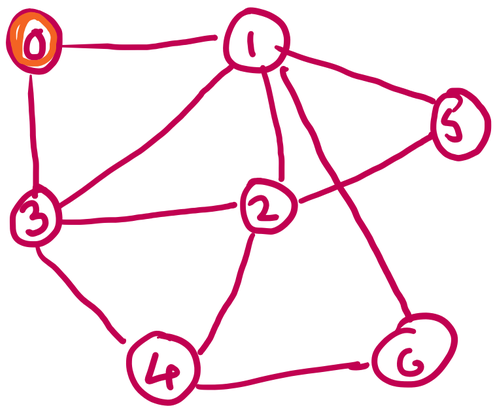
Traversing Algo

- BFS - Breadth first Search (Level order) (Queue)
- DFS - Depth first Search (Stack)

} u can start from any node if not mentioned, or specific node if mentioned.



Queue: 0 1 3 2 6 5 4

0→1→3→2→6→5→4

possibility of different answer there can be Multiple solutions.

Result: 0,1,3,2,6,5,4

Adjacent of 0 - not visited - 1,3 [or 3,1]

Adjacent of 1 - ~~3~~,2,6,5,~~0~~

3 - on Queue,
0 - visited

insert 2,6,5 on Queue, delete 1

Adjacent of 3 - ~~0~~, ~~1~~, ~~7~~, 4

0,1 - visited, 2 is on Queue
remaining 4 is inserted in Queue

Adjacent of 2 → ~~1~~, ~~3~~, ~~5~~, ~~4~~

1 & 3 are visited, 5 & 4 are on Queue
no unvisited node to be inserted

delete (2)

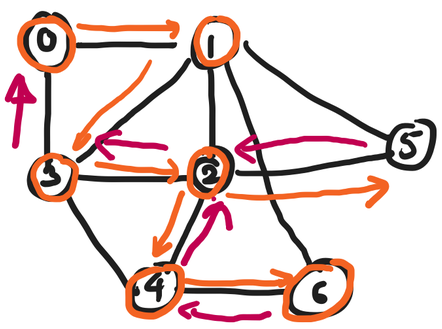
Adjacent of 6 → ~~4~~, ~~1~~ 1-visited
4 - on Queue
no unvisited node

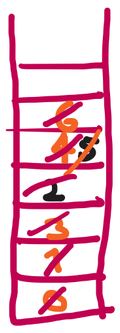
Adjacent of 5 → 1, 2
Both are visited
no unvisited node.

Adjacent of 4, - 2, 3, 6
all visited, no unvisited node & 4 delete, know Queue is empty

DFS - Depth first Search (Stack) - Size of stack is same as that no of node & in BFS size of Queue = number of nodes.

process steps.





← used for Back Tracking

Result: →0,1,3,2,4,6,5

pop (6), Tos - 4

Adjacent of 4 - 2,3,6
all visited no node to push.

pop (4) Tos - 2

Adjacent of 2 - ~~3~~, ~~4~~, 5, ~~1~~ (3, 4, 1 - visited)
push (5)

Adjacent of 5 - 1,2 (both visited) - No further node

Back Tracking pop (5)

Adjacent of 2 are visited, pop (2)

Adjacent of 3 are visited pop (3)

Adjacent of 3 are visited pop (1)

Adjacent of 4 are visited pop (0)

Stop process since stack is empty.

Adjacent 0 - 1,3 push (1)

Adjacent 1 - 2, 3, 5, ~~0~~, 6 0-visited
push (3)

Adjacent of 3 - ~~0~~, ~~1~~, 2, 4 0,1 - visited
push (2)

Adjacent of 2 - ~~1~~, ~~3~~, 4, 5 (1,3 - visited)
push (4)

Adjacent of 4 - ~~2~~, ~~3~~, 6 (2,3 - visited)

Adjacent of 6 - 1, 4 (Both are visited)
We Cannot go further (Backtracking)