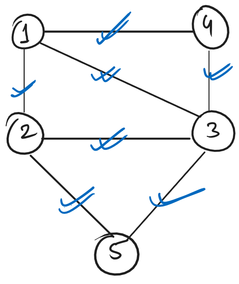
* Given a graph (undirected/directed), write a C++ code to display :→

(i) Adjacency Matrix

(ii) Adjacency List



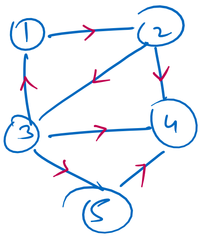
Adjacency Matrix : 5×5

	1	2	3	4	5
1	0	1	1	1	0
2	1	0	1	0	1
3	1	1	0	1	1
4	1	0	1	0	0
5	0	1	1	0	0

Adjacency List : Node → Neighbours

1 → 2, 3, 4
2 → 1, 3, 5
3 → 1, 2, 4, 5
4 → 1, 3
5 → 2, 3

Directed Graph :→

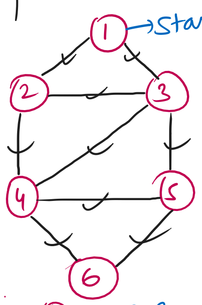


Adj List :→

Node : List of Neighbours

1 → 2
2 → 3, 4
3 → 1, 4, 5
4 →
5 → 4

Graph Traversals :→ *** Important (BFS/DFS)



Starting Node :→ (1) Who are your neighbours ?

Pre-requisites :→ * Adj List
* visited array
* queue data structure

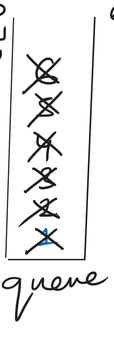
Adj List:

1 → 2, 3
2 → 1, 3, 4
3 → 1, 2, 4, 5
4 → 2, 3, 5, 6
5 → 3, 4, 6
6 → 4, 5

array → Output → 1, 2, 3, 4, 5, 6

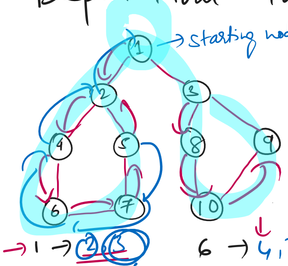
Visited array

T	T	T	T	T	T
F	F	F	F	F	F
1	2	3	4	5	6

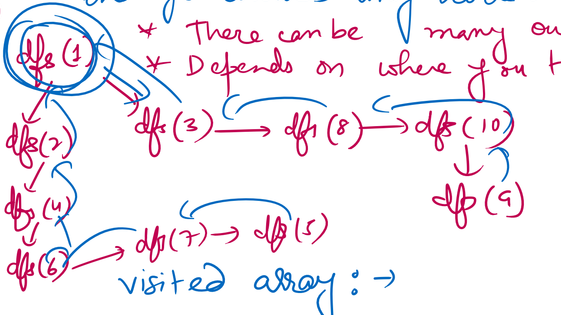


Depth First Traversal :→ Traverse the entire depth ; if you end go towards any node.

* There can be many outputs.
* Depends on where you traverse.



1 → 2, 3
2 → 1, 4, 5
3 → 1, 8, 9
4 → 2, 6
5 → 2, 7
6 → 4, 7
7 → 5, 6
8 → 3, 10
9 → 3, 10
10 → 8, 9



visited array :→

T	T	T	T	T	T	T	T	T	T
1	2	3	4	5	6	7	8	9	10

O/P → 1, 2, 4, 6, 7, 5, 3, 8, 10, 9