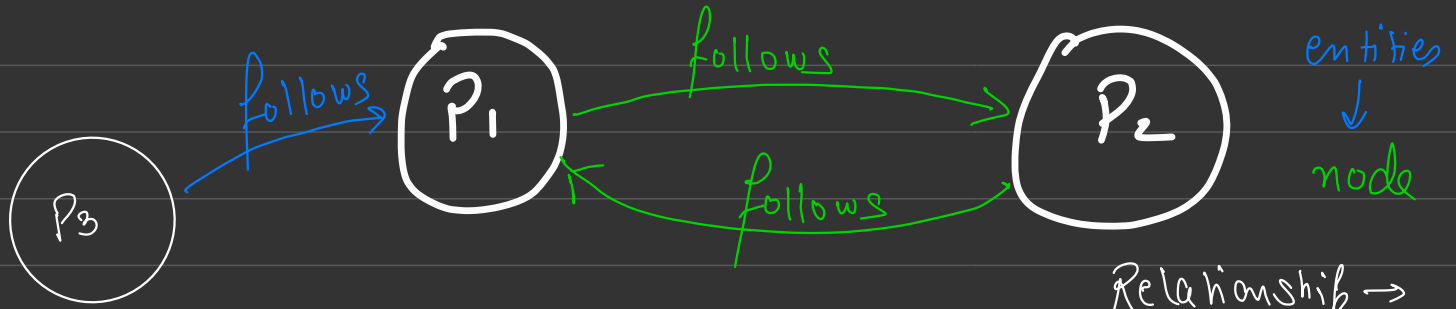
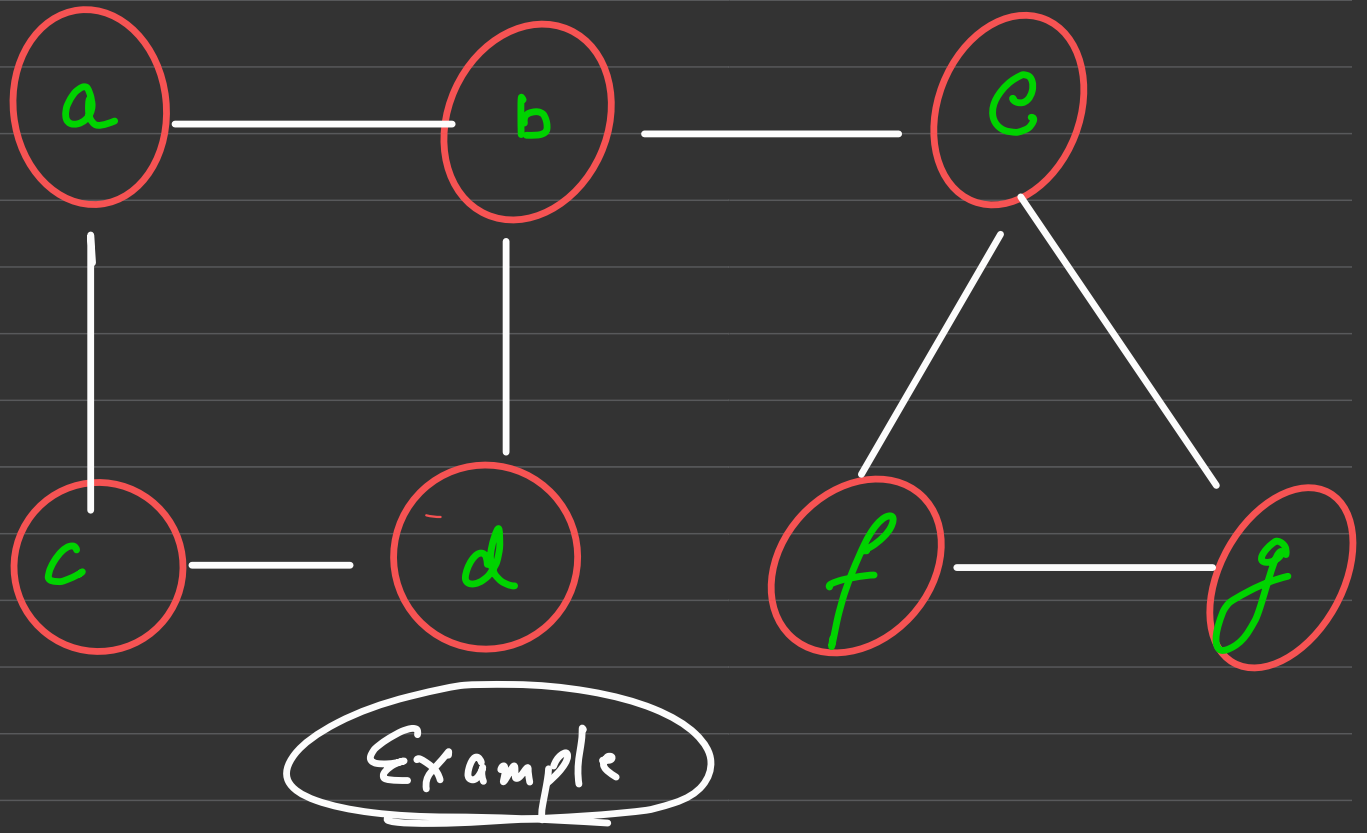
Graphs

Q⇒ What is a graph ??

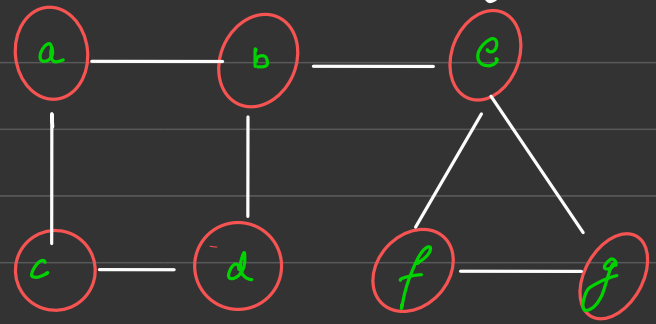
Graph is a non linear Data Structure which represents relationship btw 2 entities



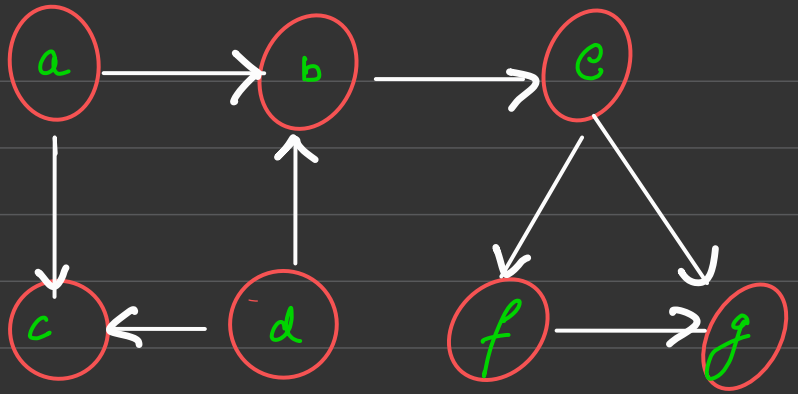

a
b
c
c
d
f
g
Example

Types of graph →

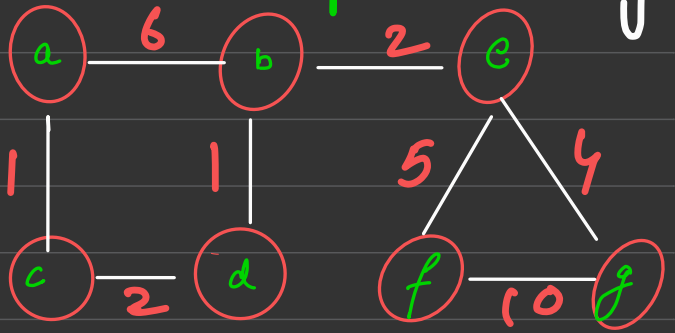
1) Undirected Graph → where edges donot have directions



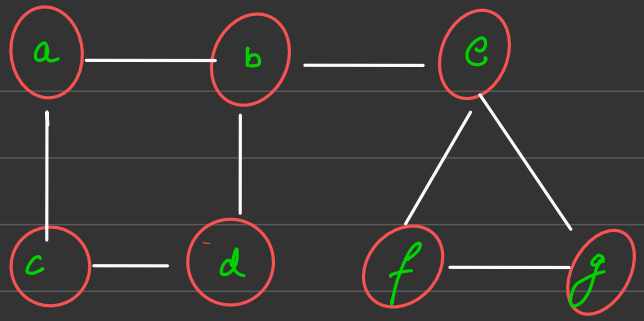
2) Directed Graphs → edges have directions



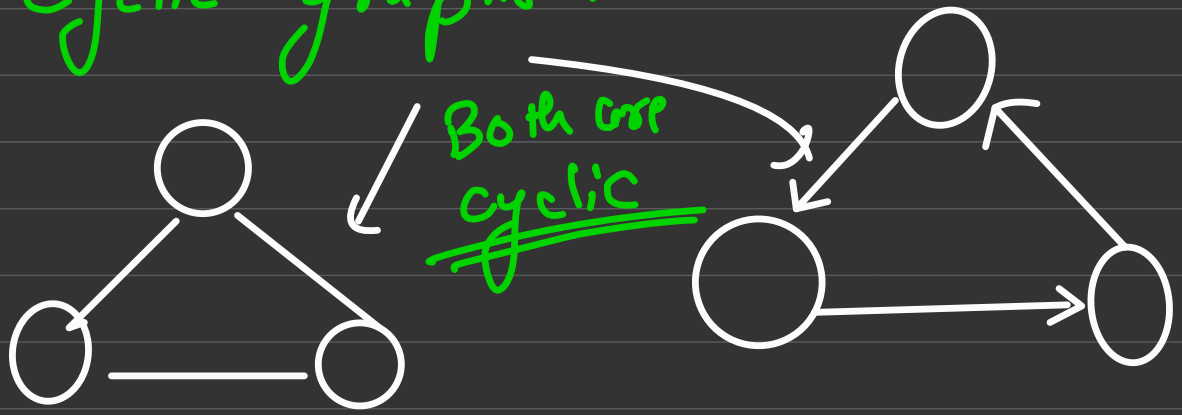
3) Weighted Graph → every edge has a weight



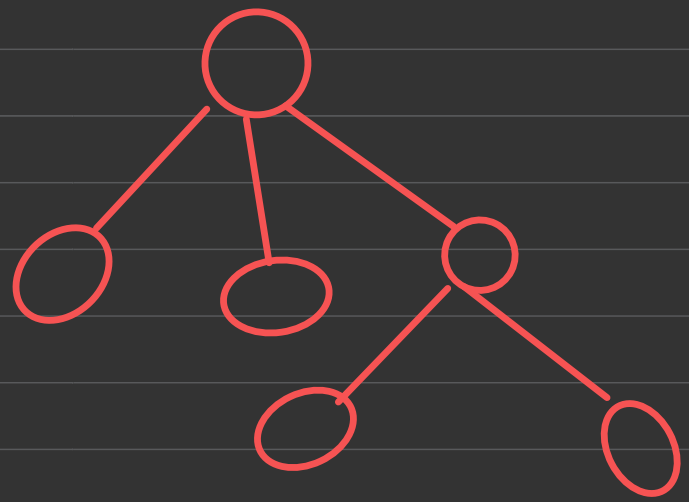
4) Unweighted graphs



5) Cyclic graphs ⇒



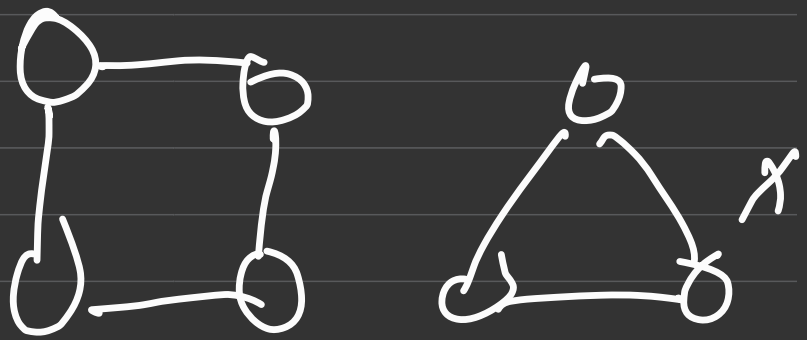
TREE → It is an undirected acyclic graph where any 2 nodes are connected by only one path



Dense graph

Sparse graph

Connected graph



How To Represent A Graph

→ Adjacency Matrix

→ Adjacency List

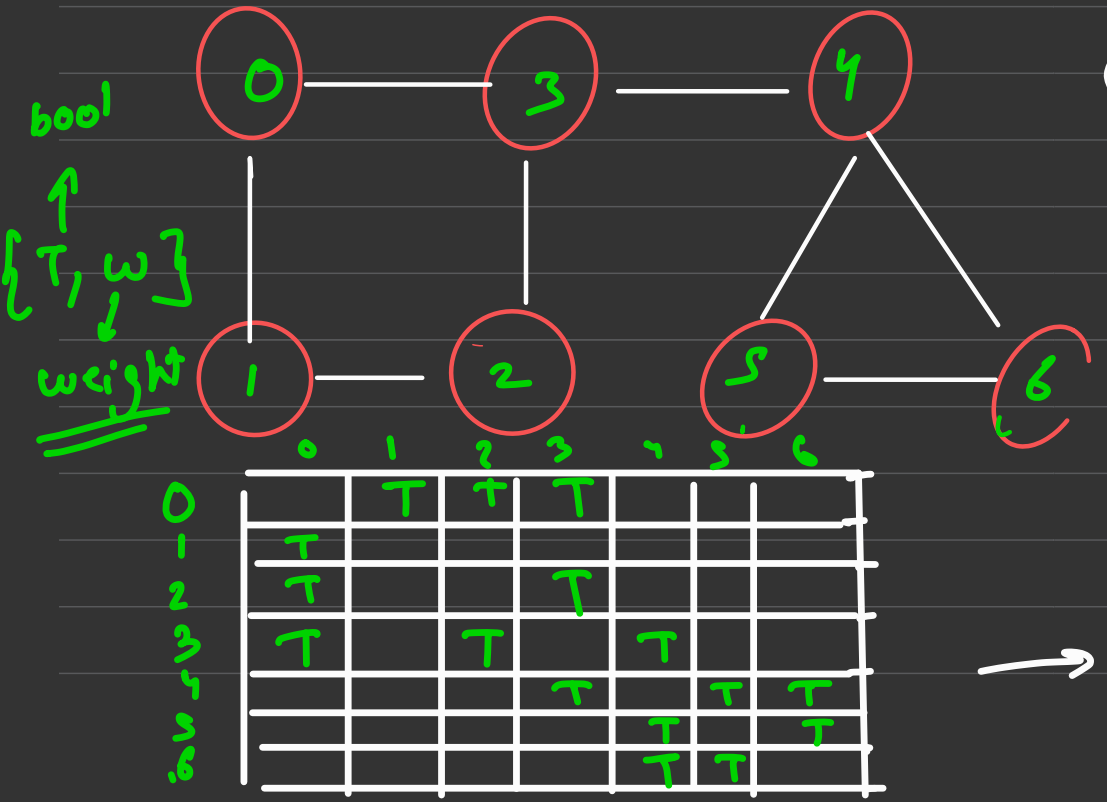
→ Adjacency Map

→ Edge list

(1) Adjacency Matrix → V vertices

↓

V x V matrix

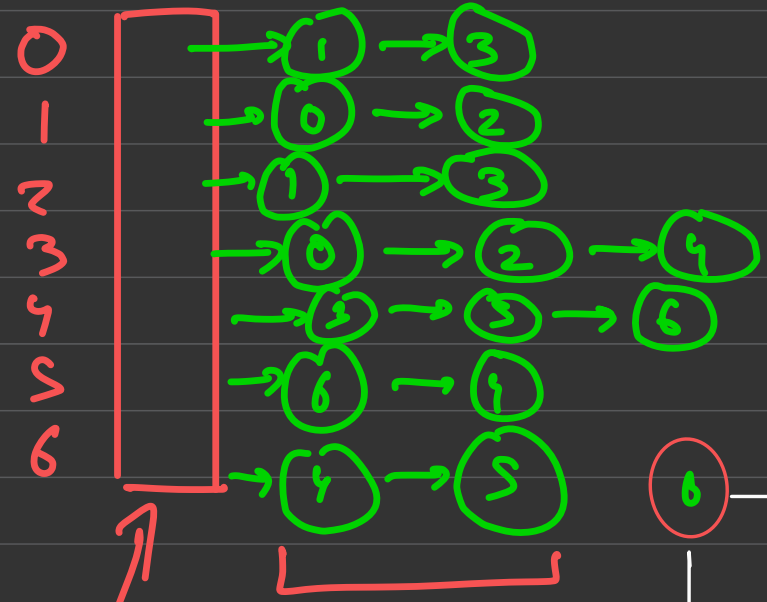


(i, j) → there is an edge b/w i^{th} node & j^{th} node or not

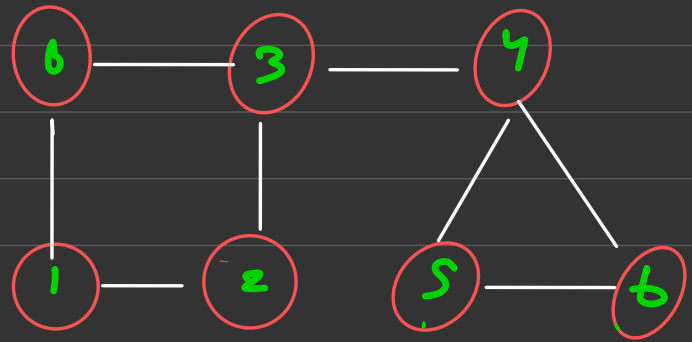
→ wastage of space

Adjacency List → array of LL

vector of vector

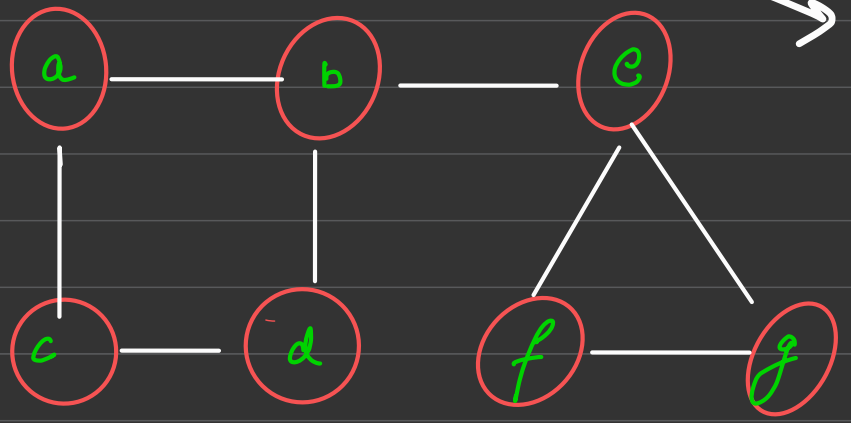


→ O(V + E)



Adjacency Map → array of map

Edge List → array of edge object



storing the info of edge

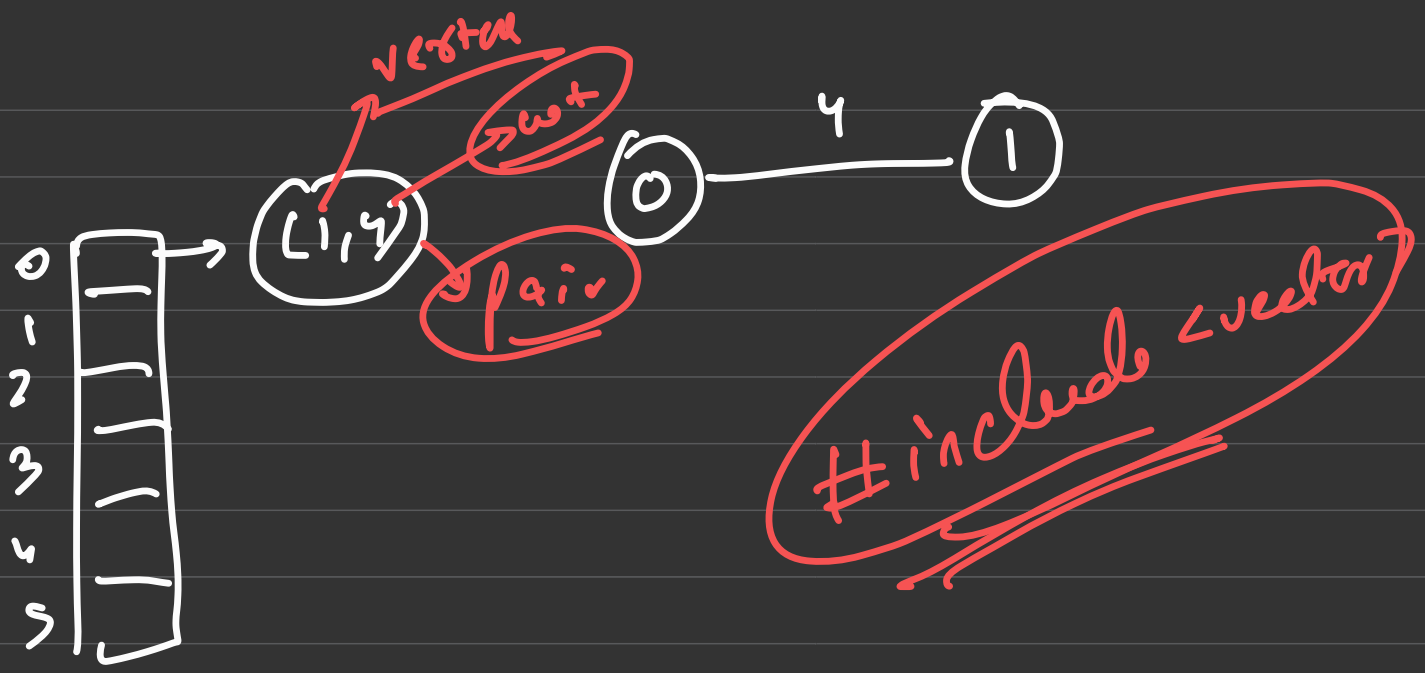
src
dest
wt

[(a,b), (a,c), (b,d), (c,d), (b,e) ----]

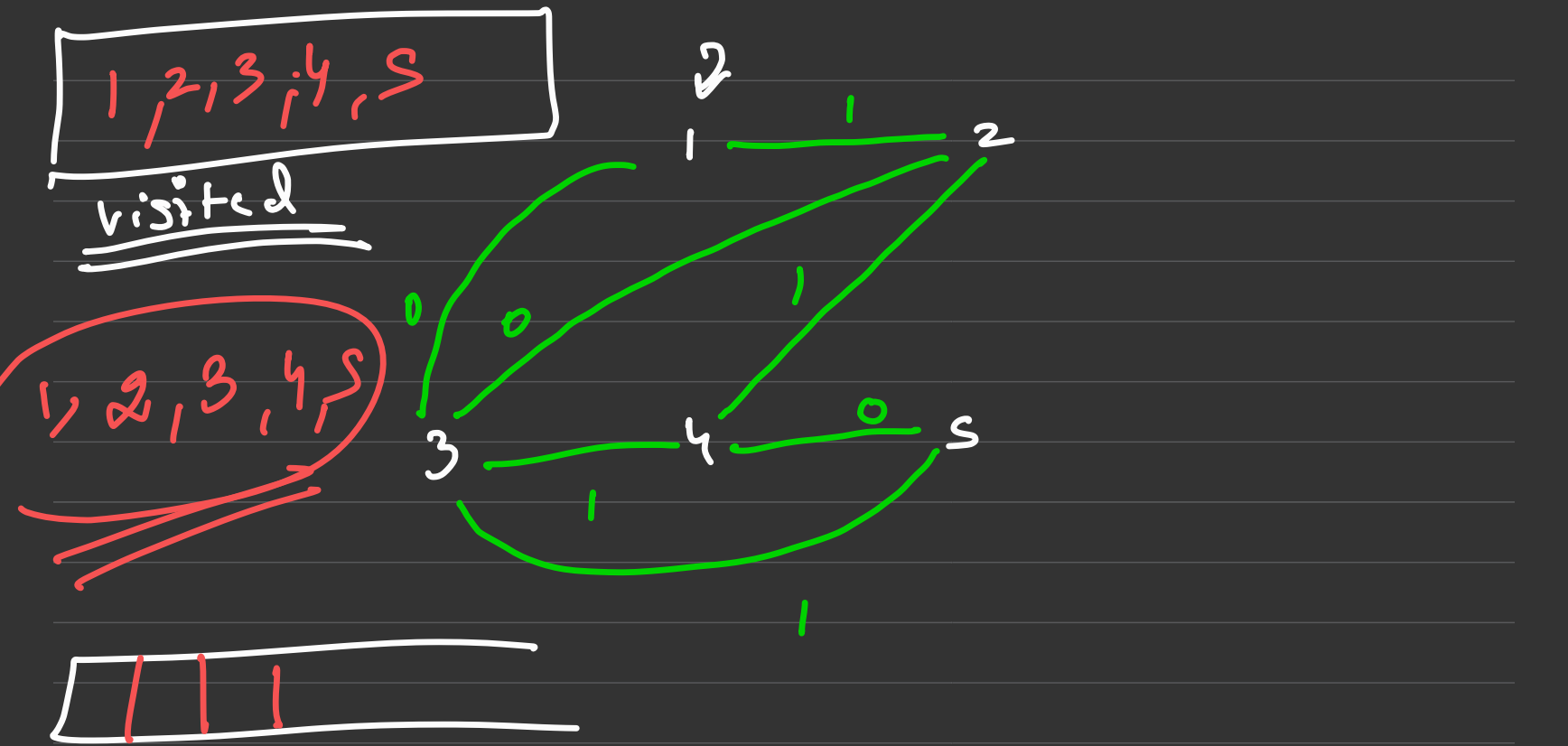
→ Kruskals

p ← {
first: x
second: y
}

p.first → x
p.second → y



0
1
2
3
4
5
(1,4)
vertex
wt
pair
0
4
1
#include <vector>

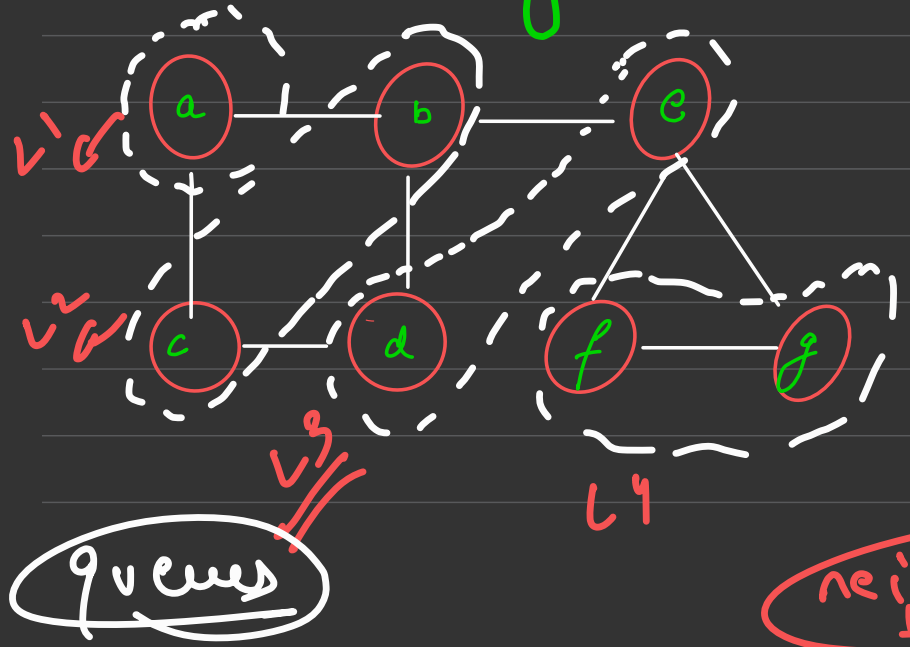

1,2,3,4,5
visited
1,2,3,4,5
2
1
2
0
0
1
3
4
0
5
1
1

Graph Traversal

1) BREADTH FIRST TRAVERSAL/SEARCH (BFS)

2) DEPTH FIRST TRAVERSAL / SEARCH (DFS)

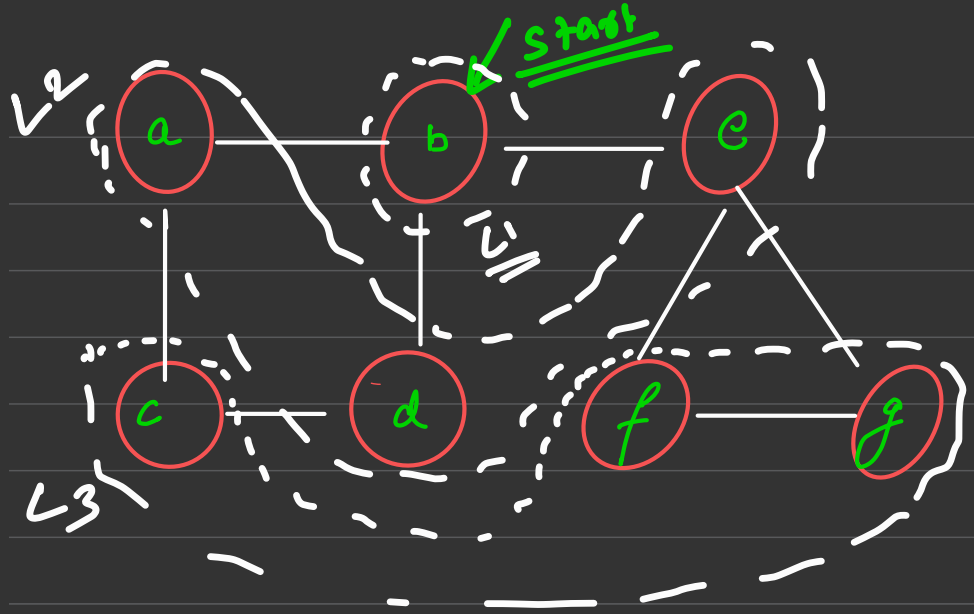
BFS → In this traversal, we select a node and traverse the graph layer wise (levelwise) thus exploring the neighbour nodes first



a, b, c, e, d, f, g

BFS

neighbour first



queue

b, a, d, e, c, f, g

visited

set

bool array

O(V+E)