The Graph Algorithms Course

Pre-Requisite

- Basic Programming (we will code in C++)
- Basic - Intermediate Recursion
- Arrays, HM, LL, Trees

Outcome Expectation

- Course will be starting from very basics but we will cover adv CP topics
- Beneficial with Interview perspective too.

(21 classes)

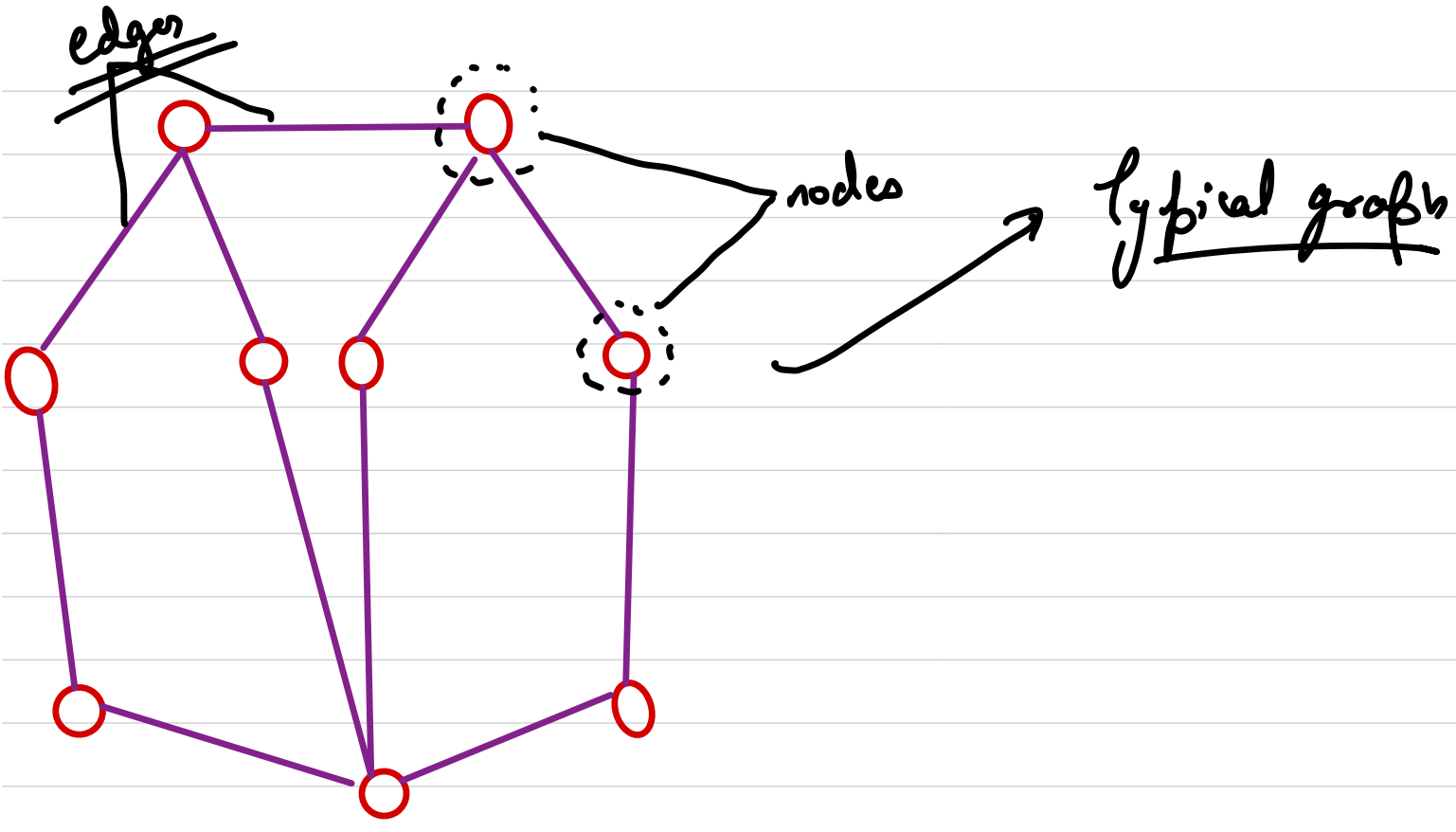
Topics

- Basic graphs & terminologies
- Traversal techniques BFS / DFS
- DP on graph (DP on DAGs)
- Matrix BFS / DFS , 0-1 BFS dc
- Trees (n-ary) , DP on trees
- Shortest path, MST
- DSU
- SCC , Bridges / articulation point

Graphs

Q⇒ What is a graph ??

Graph is a collection of nodes and edges where each node might point to / connected to other nodes. The nodes represent real life entities and are connected by edges representing relationship between the nodes.


edges
nodes
typical graph

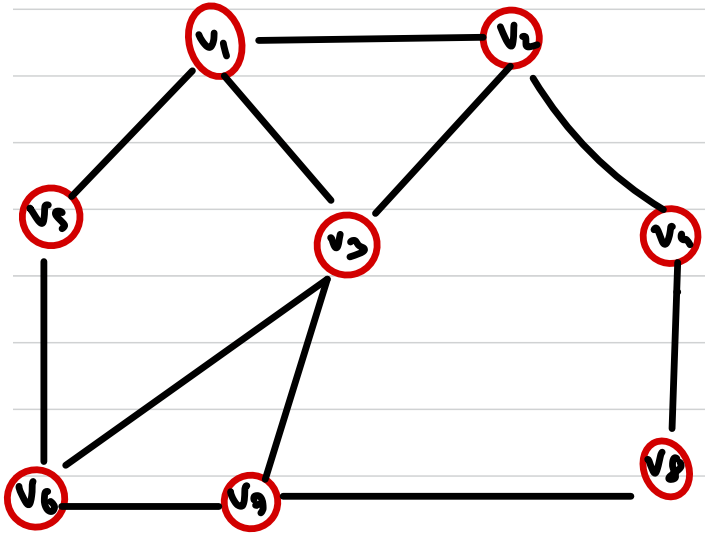
#Applications of graph

Biology → PPI graph
→ Metabolic network graph

Electrical → Circuit org graph

Computer Scene → Shortest path (ncp)
→ gsm frequency assignment
→ Routing mechanism
→ social media

formal definition of graph



V → set of vertices

$V = \{ v_1, v_2, v_3, \ldots\ldots, v_n \}$

Edge set

$E = \{ \{v_1, v_2\}$
$\{v_2, v_3\}$ → unordered set
$\vdots$
$\}$

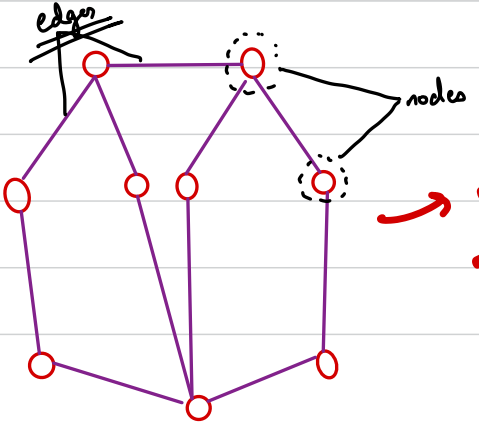
$G = (V, E)$ → mathematical notation of graph

A graph G is an ordered/unordered pair of a set V vertices & E, edges.

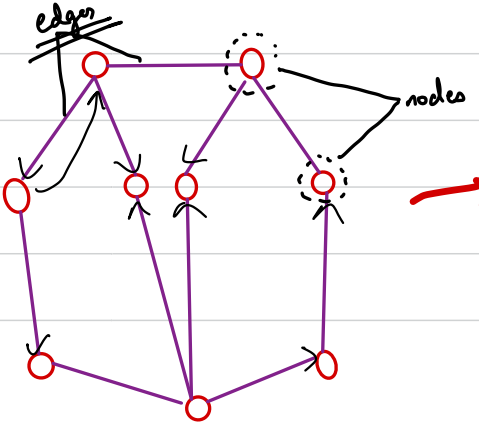
Types of graph

Based On direction

→ Directed → water flow, road network,
→ Undirected → facebook



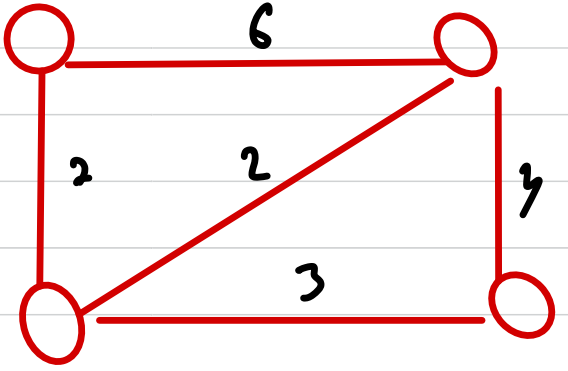
→ undirected



→ directed

Based on edge property

1. Weighted
2. Unweighted



Based on edge density

1) Sparse

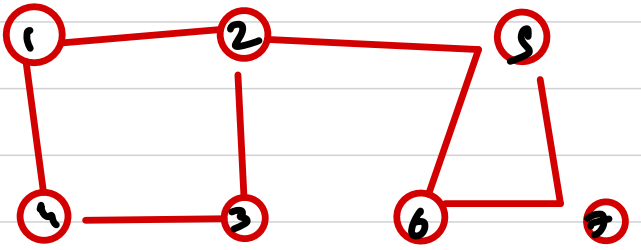
2) Dense

Representing graphs as a data structure

1) Adjacency Matrix
2) Adjacency List → adjacency set/map
   → edge list
3) Incidence Matrix

adjacency matrix

$A_{ij} = \begin{cases} 1 \rightarrow \text{if there is an edge from } i \text{ to } j \\ 0 \rightarrow \text{else} \end{cases}$



$A =$

	1	2	3	4	5	6	7
1	0	1	0	1	0	0	0
2	1	0	1	0	1	0	0
3	0	1	0	1	0	0	0
4	1	0	1	0	1	0	1
5	0	1	0	1	0	1	1
6	0	0	0	0	1	0	1
7	0	0	0	0	1	1	0

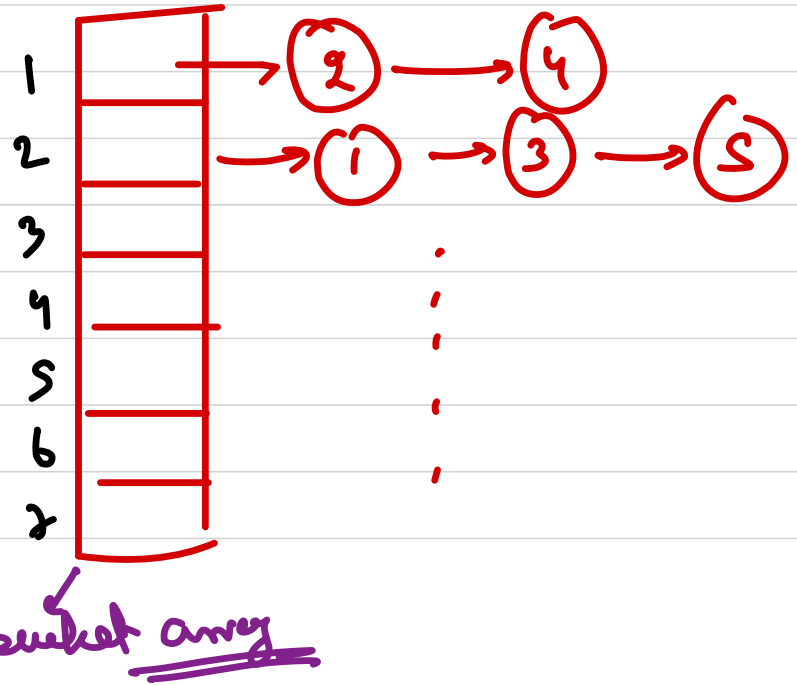
$\rightarrow O(V^2)$

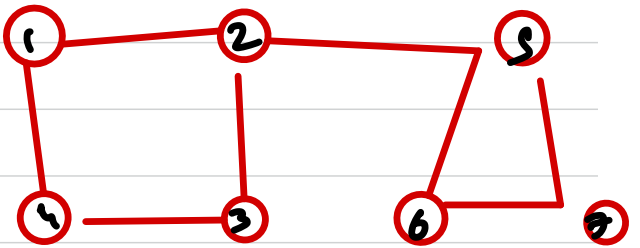
$N \times N$

adjacency List

{v, w}

Array of LL





O (V + E) → space

adjacency map/set

array of hashmap

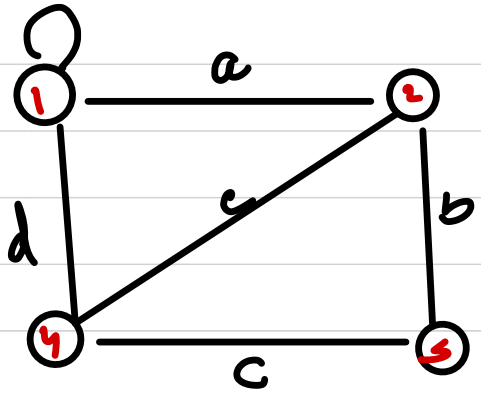
Incidence Matrix

Smia

M=

a	b	c	d	e	
1	0	0	1	0	1
1	1	0	0	1	2
0	1	1	0	0	3
0	0	1	1	1	4

(V*E)



col → edges
rows → vertex

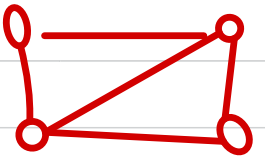
$$M_{ij} = \begin{cases} 1 & \text{if the } i^{th} \text{ vertex belongs to the } j^{th} \text{ edge} \\ 0 & \text{else} \end{cases}$$

col sum → 2

row sum → degree

Q. What is a degree ??

Degree of a vertex in a graph G, is the total no. of edges incident / associated with it.



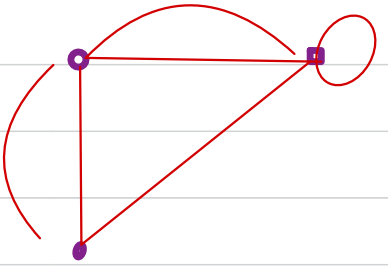
directed graph → indegree / outdegree

outdegree → no. of outgoing edges

indegree → no. of incoming edges

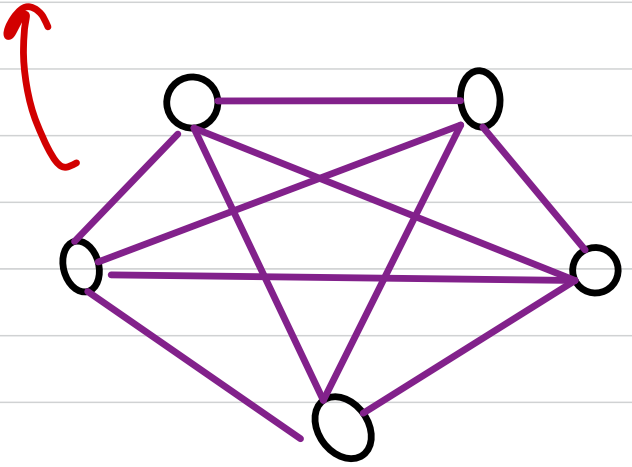
Multi graph → an undirected graph with multiple edges b/w two vertices & self loops allowed.

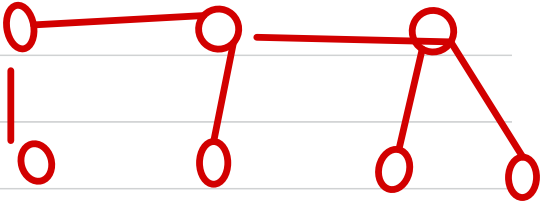
Multi graph



Simple graph → An undirected graph in which both multiple edges & loops are not allowed.

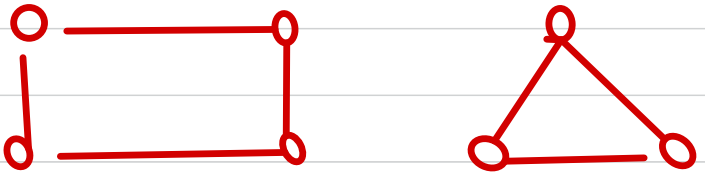
Complete graph → A graph in which every vertex is directly connected to every other vertex.





Connected graph → A graph in which there is some path between 2 vertices but not necessarily direct

Disconnected graph → atleast 2 vertices do not have a path to every other vertex.



Component → a subset of a disconnected/connected graph which is connected.

#path → A path P_n is a graph whose vertices can be arranged in some sequence such that

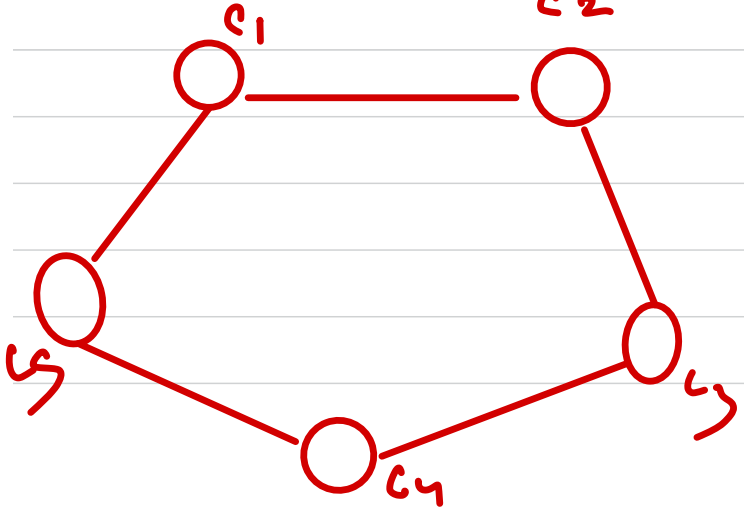
$V = \{v_1, v_2, v_3, v_4\}$

edge set of the graph is

$$E = \{ v_i v_{i+1} \quad \forall \; i \in [1, n-1] \}$$

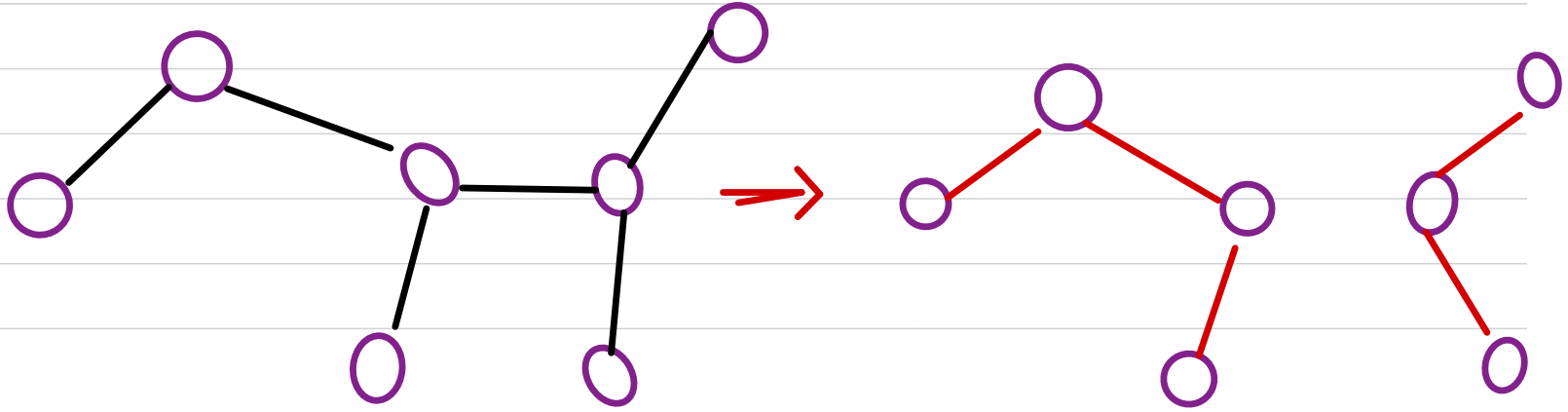
Cycle → A cycle C_n is a graph whose vertices can be arranged in a cyclic sequence. such that edge set is

$$E = \{v_i v_{i+1} \ \forall \ i \in [1, n-1]\} \cup \{v_1 v_n\}$$

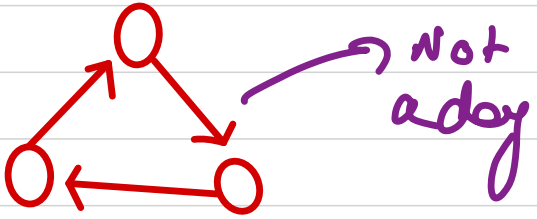


TREE → Tree is a connected graph with no cycles

forest → If we remove an edge from tree we get a forest which is collection of trees



DAGs (directed acyclic graphs) →



How to read graphs

As graphs are non-linear, we need some mechanism to read graphs.

Graph traversal

→ Depth first

→ Breadth first

Depth first search

TC
2
$O(V+E)$

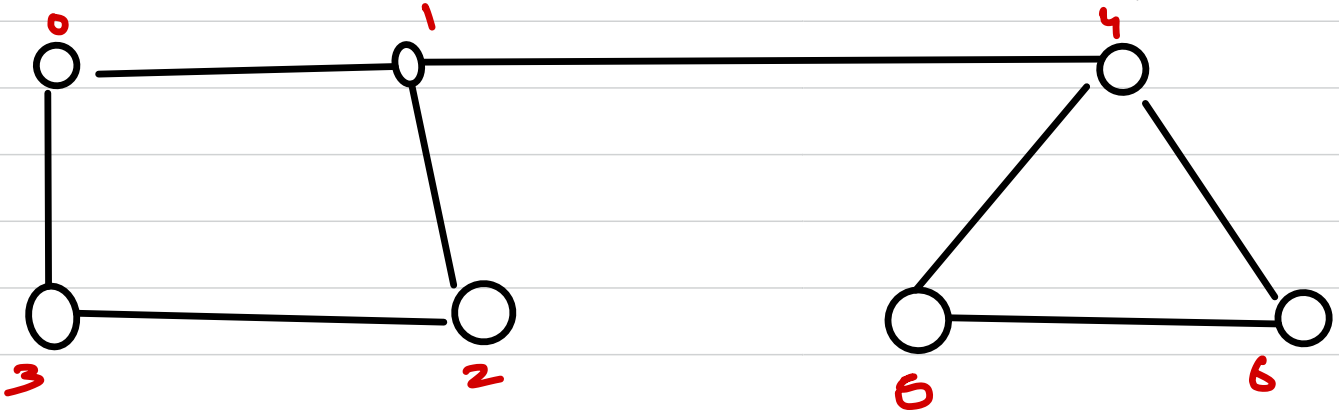
★ Motivation problem:

Q. Given a graph calc all paths between two vertices

OR

Q. Given a graph check whether there is a path between 2 vertices.

Last in → stack



is there a path from 0-5

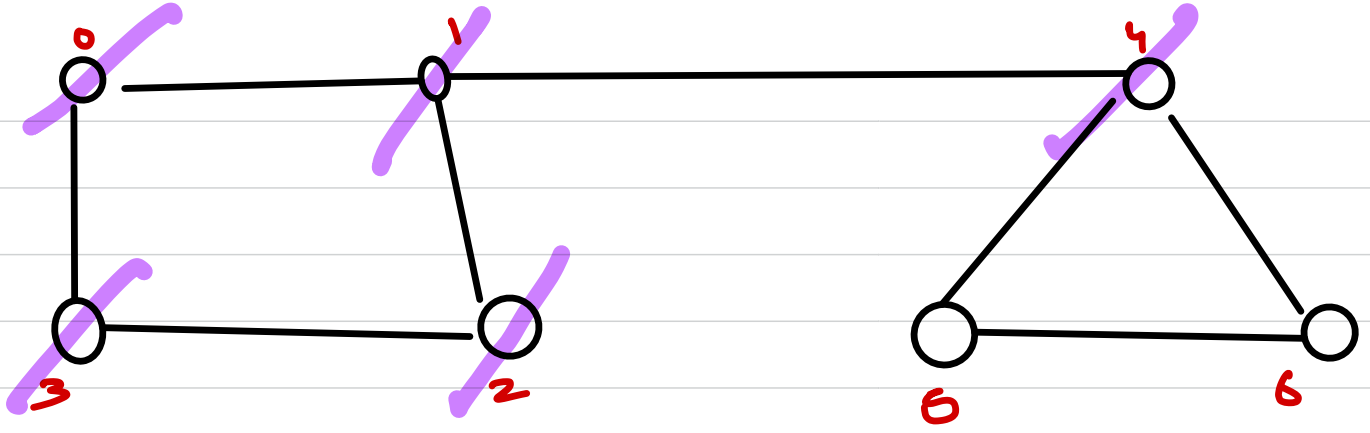
What is the most easiest query for this ??

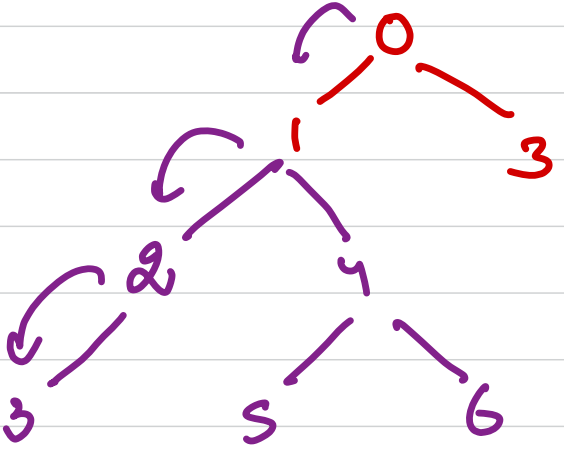
→ neighbours

$f(u,v)$ = whether there is a path from u to v.

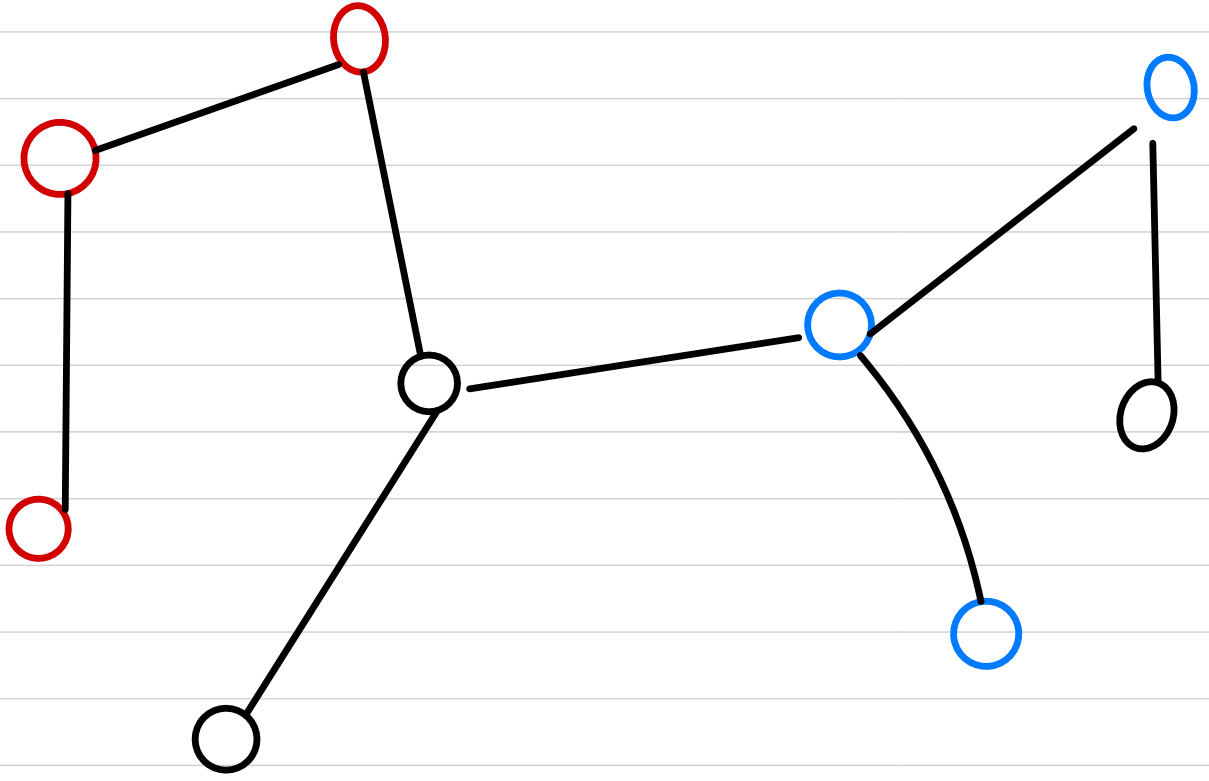
$$f(u,v) = \begin{cases} f(n_1, v) \\ \text{or} \\ f(n_2, v) \\ \text{or} \\ f(n_3, v) \\ \vdots \end{cases}$$

where $(n_1\ n_2\ n_3 \dots) \in$ neighbours of u.


0
1
4
3
2
6
6


0
1
3
2
4
3
5
6

→ total red
total blue]



Proof that a tree with n nodes has n-1 edges.

PMI → $f(n)$ → no. of edges for n nodes

$f(1) \rightarrow 0$

$f(2) \rightarrow 1$

To proove → $f(n+1)$

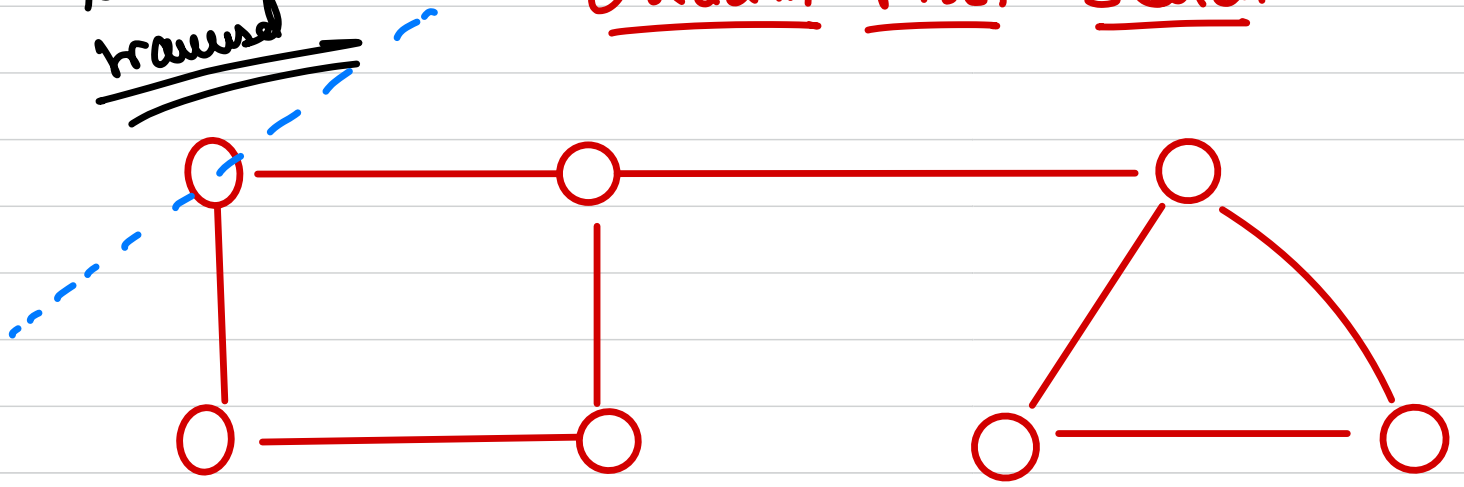
we assume function works for $f(n) \rightarrow n-1$

Total no. of edges → no. of edges required for $(n+1)^{th}$ node
+ $(n-1)$ edges

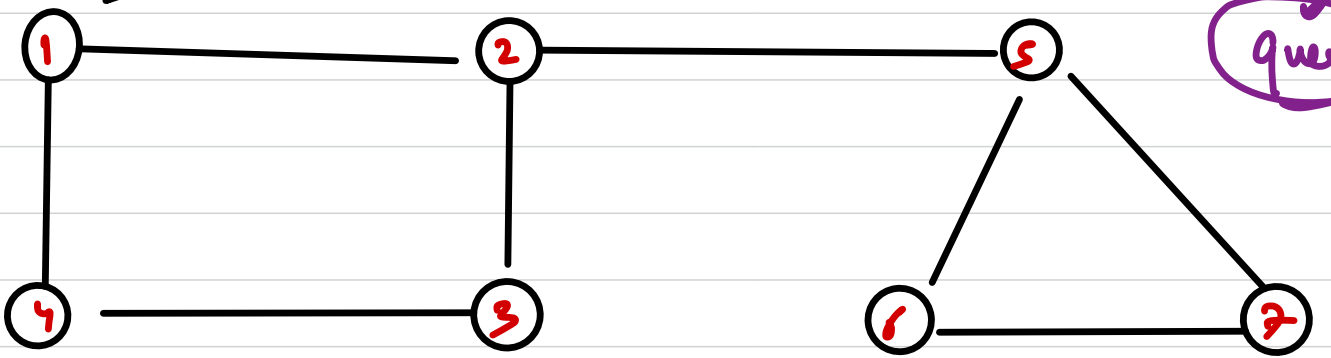
Every node that will be added to a tree requires only 1 node.

$f(n+1) = (n-1) + 1 \Rightarrow n$ H.P

Breadth First Search



Say we have an arbitrary src node then all the nodes with same shortest (by edges) distance from src node are said to be on the same level.



1, 2, 4, 3, 5, 6, 7 → Breadth first traversal

→ queue