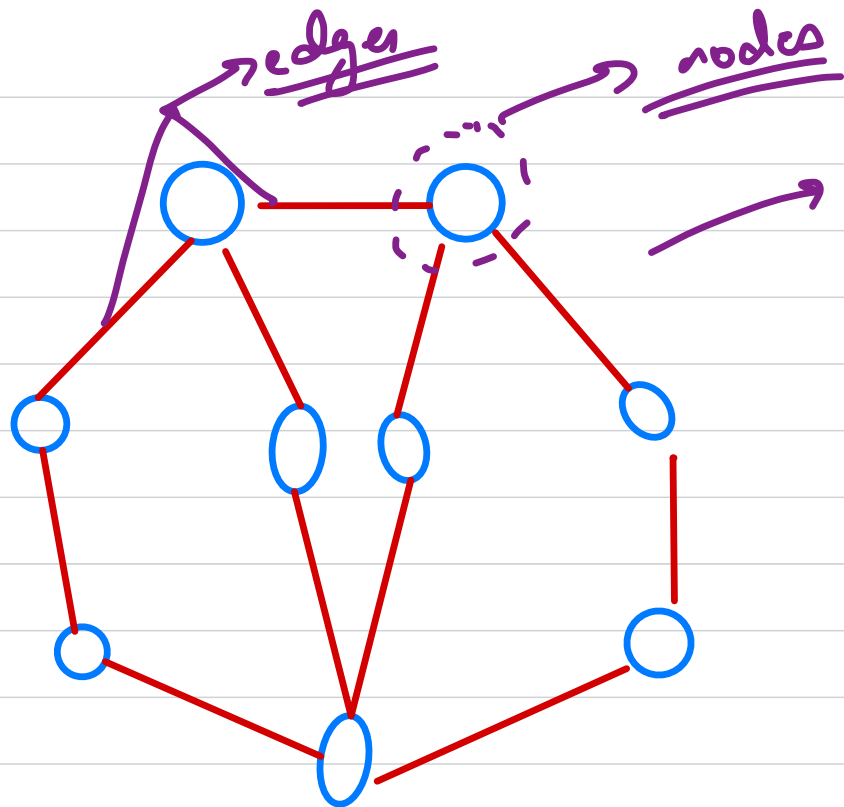


Graphs

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.



A typical graph

Biology → metabolic networks
→ PPI (Protein Protein Interactions)

Electrical → Circuit Organization

CSE → shortest path
→ Routing algo
→ graph dbs

Uber
Ola
Sunggy
Zomato
Foodpanda

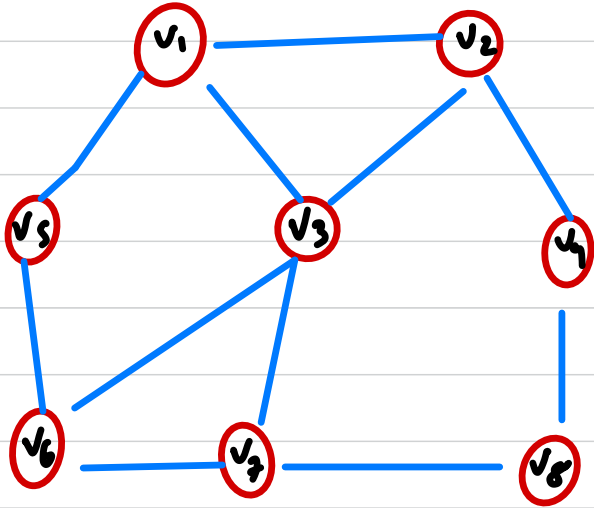
Map → gmail
→ her
→ apple us

(twitter
instagram
fb
:
)

formal definition of graph : $V = \{v_1, v_2, v_3, v_4, \dots, v_n\}$

$E = \{ \{v_1, v_2\}, \{v_2, v_4\}, \{v_2, v_3\}, \dots \}$
edges pairs are unordered

8 vertices
11 edges



$G = (V, E) \rightarrow$

mathematical notation of a
graph

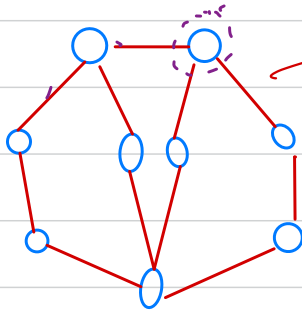
A graph G is an ordered pair of a set V vertices & E , a set of edges.

Type of graph :

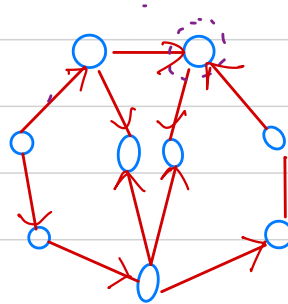


① Undirected → facebook

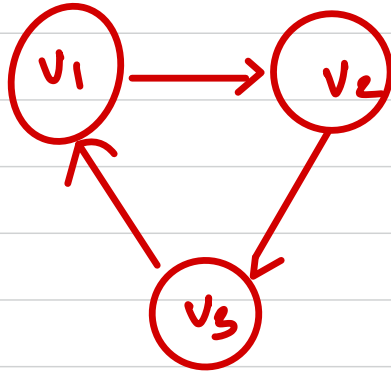
② Directed → Instagram, Twitter



→ undirected graph



→ Directed graph



$$G = (V, E)$$

$$E = \{ \{v_1, v_2\}, \{v_2, v_3\}, \{v_3, v_1\} \}$$

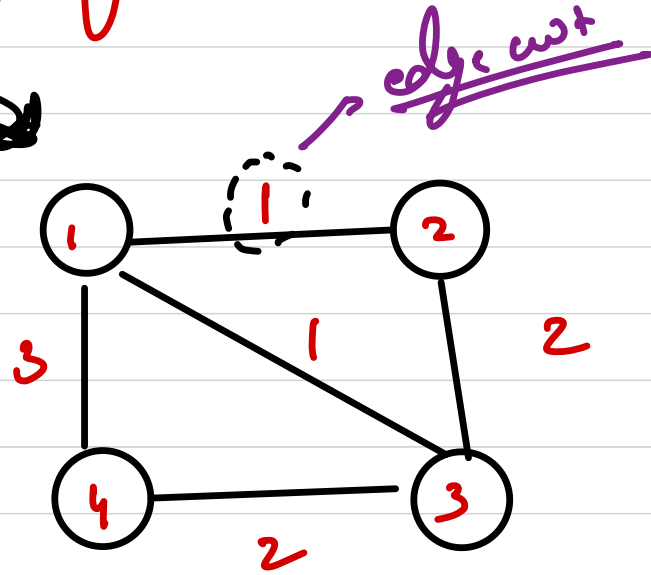
Set of
ordered
pairs

Based on edge property

① weighted



② Unweighted



Based on no. of edge

(1) Sparse

(2) Dense

Graph data structure →

↳ It is a non-linear data structure,
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.

Representation of graphs :

(1) Adjacency Matrix

(2) Adjacency List \longrightarrow Adjacency Map

(3) Incidence Matrix

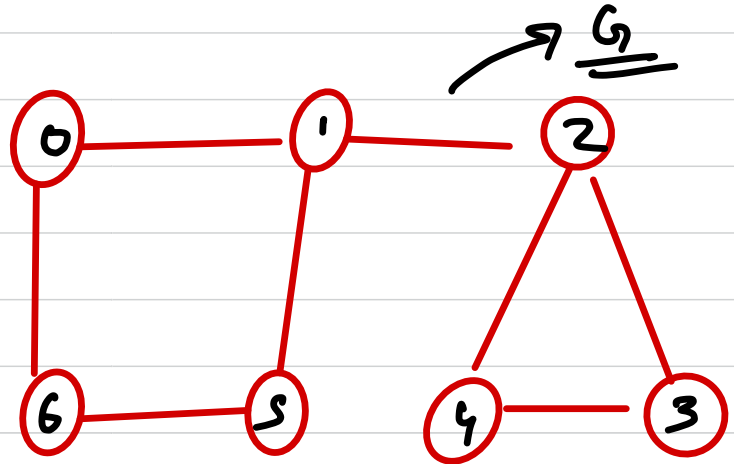
\longrightarrow edge List

Adjacency Matrix

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

$A =$

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



$\rightarrow \underline{\underline{O(V^2)}}$

$\underline{\underline{(V \times V)}} \rightarrow \underline{\underline{\text{dense}}}$

$\underline{\underline{|V|}} \Rightarrow \text{no of vertices}$

$V \times V$

Adjacency list

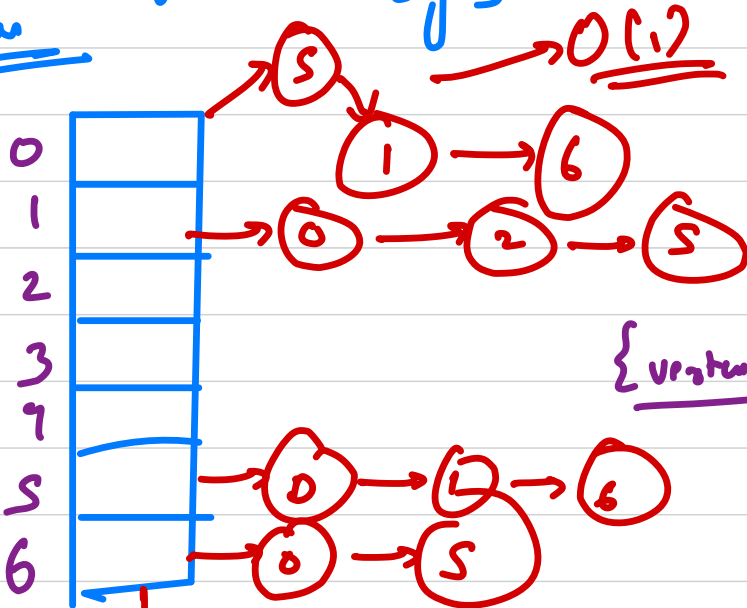
Array

of LL

→ space option

vertices

→ edges

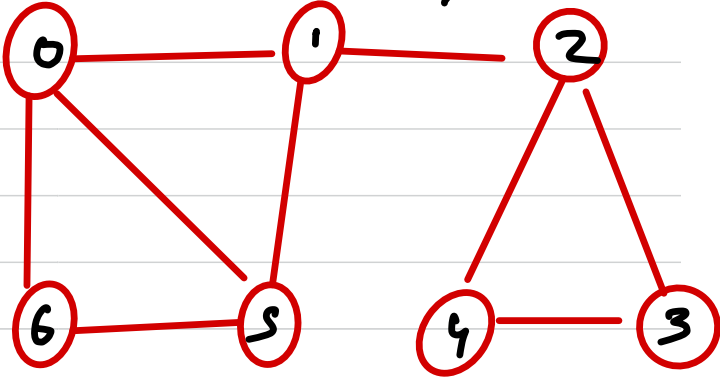


{ vertex, wt } → node

$$(V + 2E) \approx \underline{\underline{O(V+E)}}$$

Bucket array → index stores info about its edges

add at head
good for sparse graph



adjacency map

↓
array of hashmap

hasedge (v, v) → OG

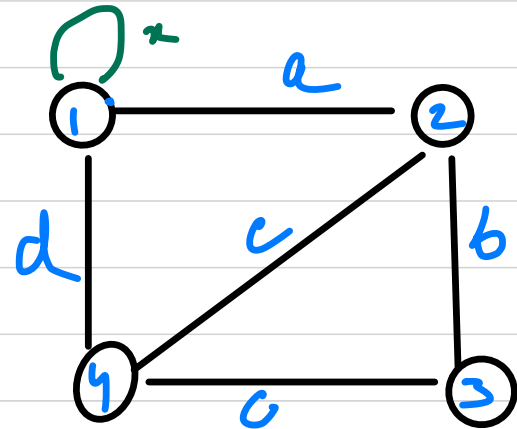
Incidence Matrix

$V = \{1, 2, 3, 4\}$
 $E = \{a, b, c, d, e\}$

$M =$

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

$(V \times E)$
 rows \rightarrow vertex
 column \rightarrow edge



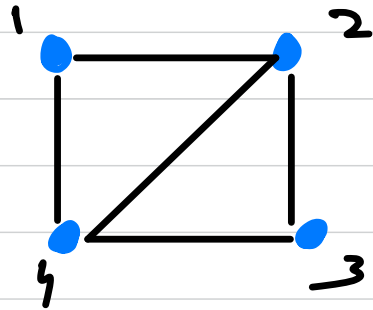
$(V \times E)$ dimension

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

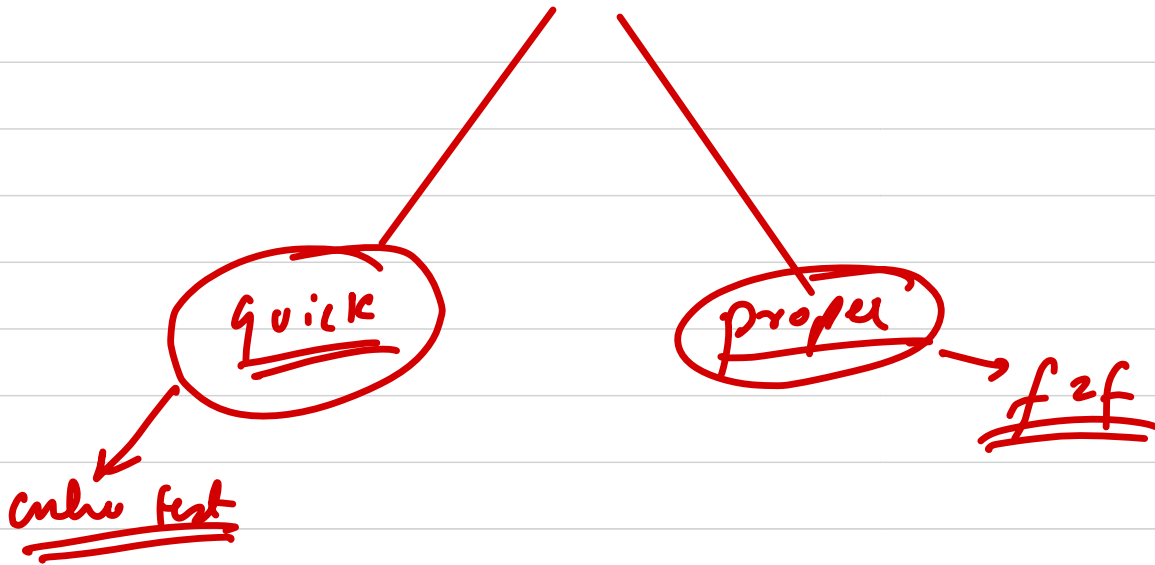
row-sum = deg
 column-sum = 2

Q₂ What is a degree??

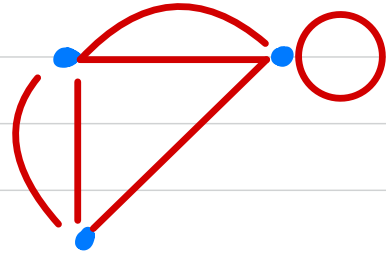
Degree of a vertex in a graph G , is the total no. of edges incident to it / associated with it (undirected graphs)



#note \rightarrow In a directed graph, the outdegree of a vertex is total no. of outgoing edges & indegree is total no. of incoming edges.

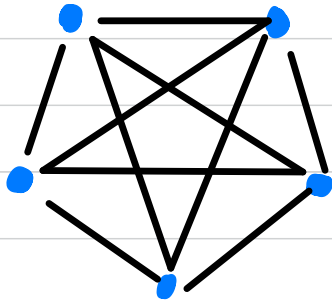


Multi graph \rightarrow an undirected graph with multiple edges and loops allowed.



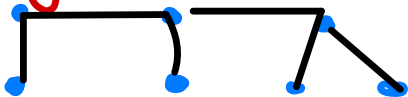
Simple Graph \rightarrow An undirected graph in which both multiple edge & loops are not allowed.

Complete Graph \rightarrow A graph in which every vertex is directly connected to every other vertex.

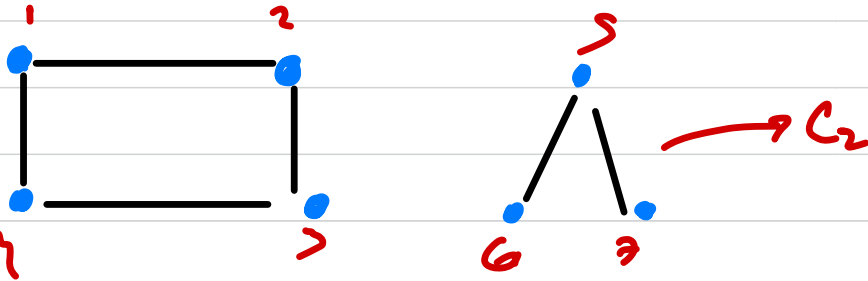


complete graph

Connected graph \rightarrow A connected graph has a path from every vertex to other vertices, not necessarily direct.



Disconnected graph \rightarrow at least 2 vertices not have a path to any other vertex.

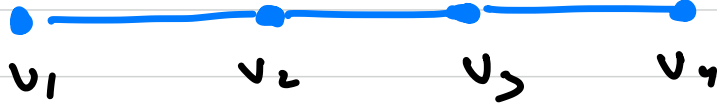


C_1

$$C_1 = (\{1, 2, 3, 4, 5, 6, 7\}, \{\{1, 2\} \{2, 3\} \{3, 4\} \{4, 1\} \{5, 6\} \{6, 7\} \{7, 5\}\})$$

component \rightarrow a subset of a disconnected / connected graph which is connected.

path \rightarrow A path P_n is a graph whose vertices can be arranged in some sequence



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

such that edge set of a graph is

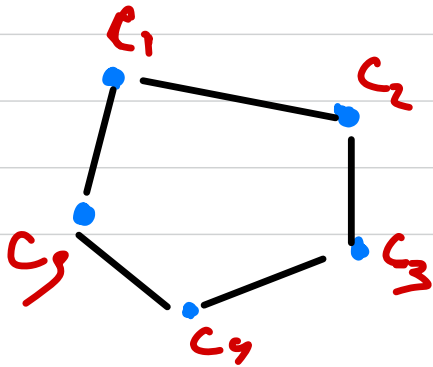
$$E = \{v_i v_{i+1} \mid \forall i \in \underline{1, n-1}\}$$

Cycle \rightarrow A cycle C_n is a graph whose vertices can be arranged in a cycle sequence

$$V = \{v_1, v_2, v_3, v_4, \dots, v_n\} \text{ such}$$

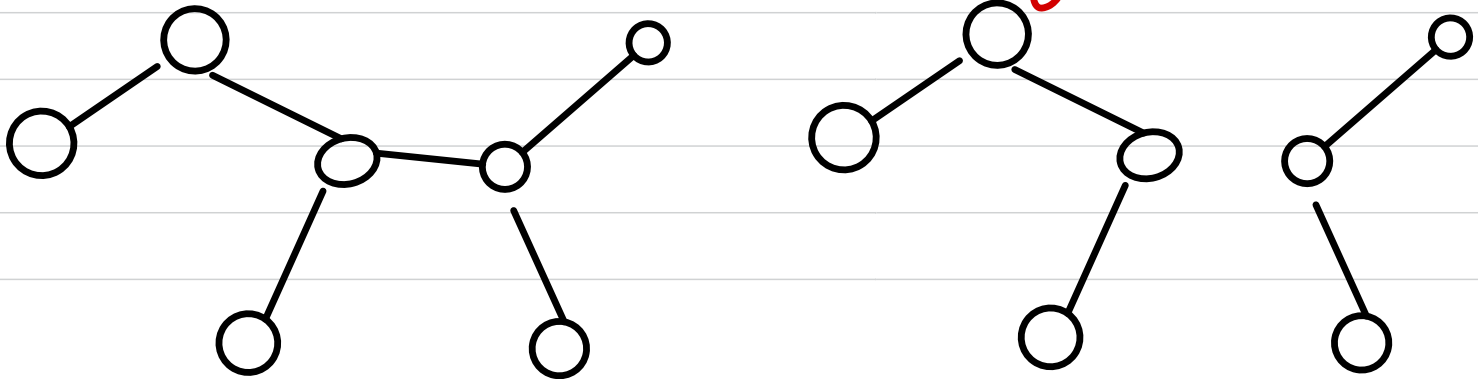
that edge set is

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

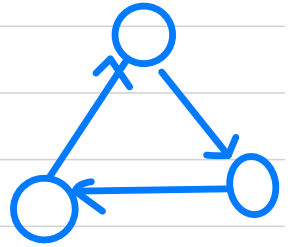
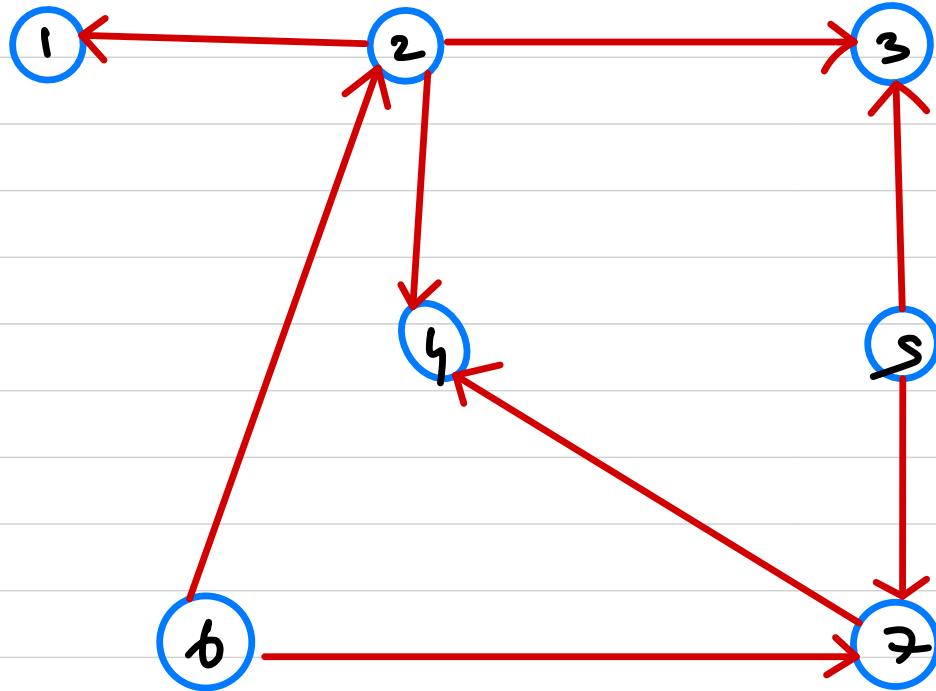


TREE \rightarrow Tree is a connected graph with no cycles.

forest \rightarrow If we remove an edge from tree, we get a forest viz collection of trees.



DAG (Directed acyclic graph)



not a
DAG

How to Read Graphs

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

Graph Traversals

- ① Depth first search
- ② Breadth first search

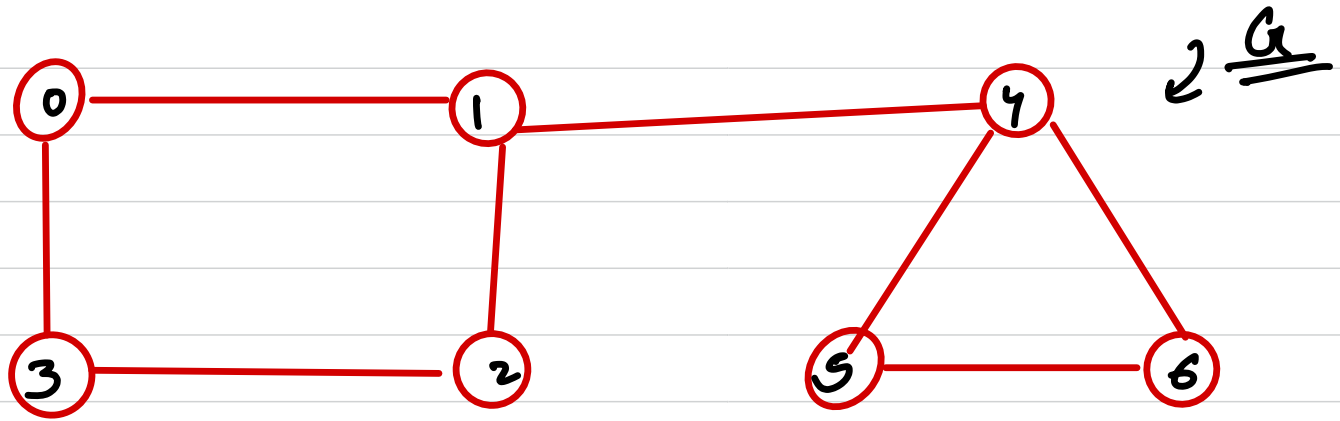
Depth first Search

Motivation problem:

Given a graph calculate all paths between 2 vertices

OR

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



is there a path from 0 to 5 ?

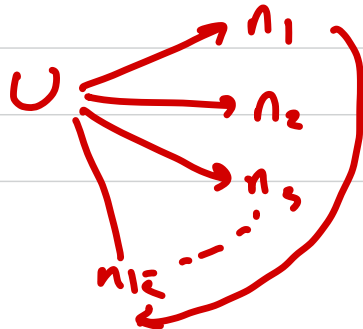
What is the most correct query for this ?

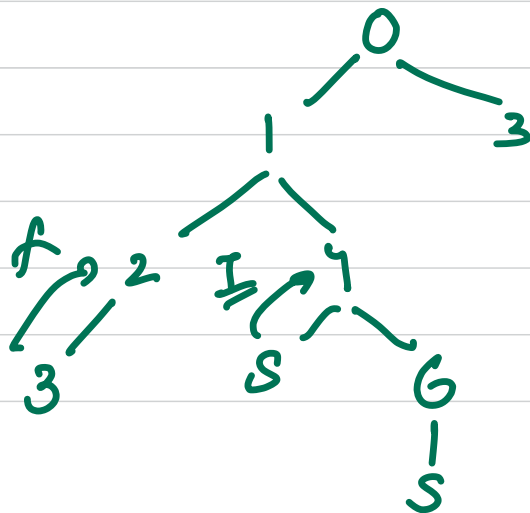
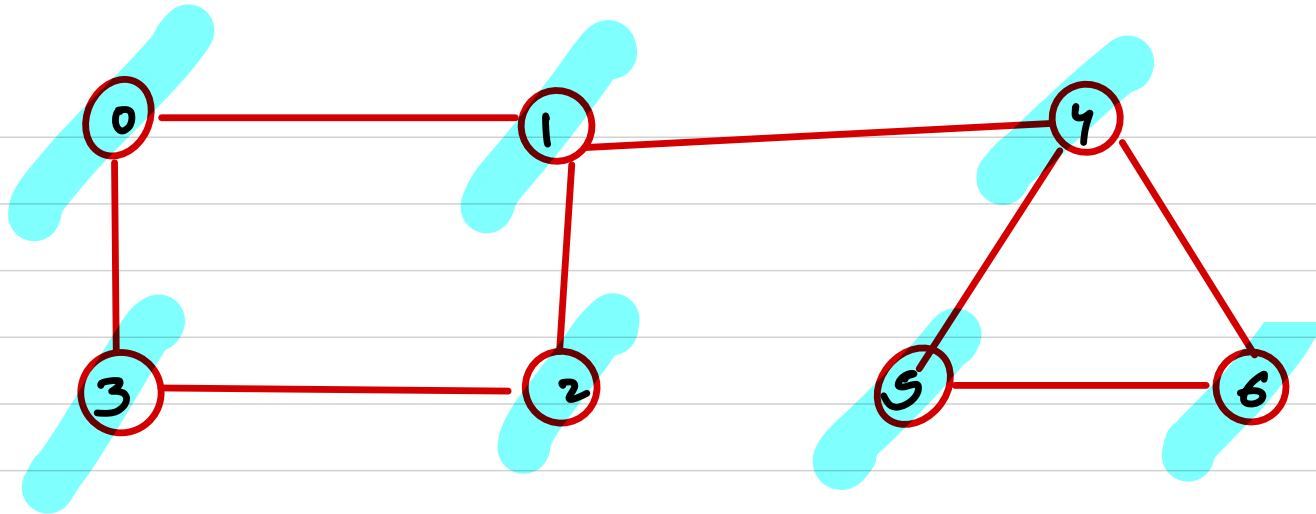
→ neighbors

path always exist for invalide
reply our

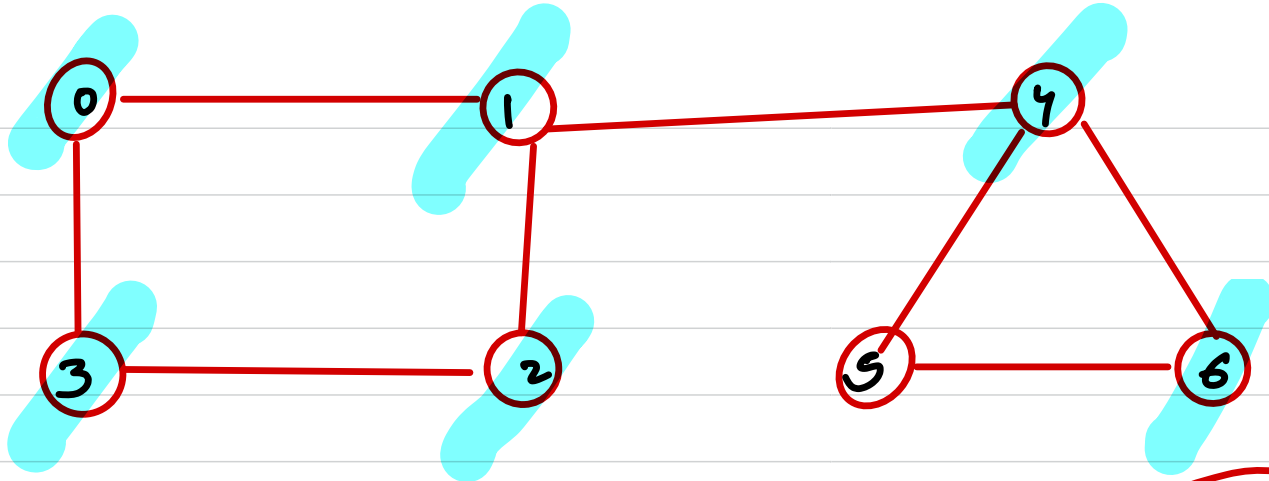
$$f(u, v) = \begin{cases} f(n_1, v) \\ \text{or} \\ f(n_2, v) \\ \text{or} \\ f(n_3, v) \\ \vdots \\ f(n_k, v) \end{cases}$$

whether there is
a path from
 u to v .



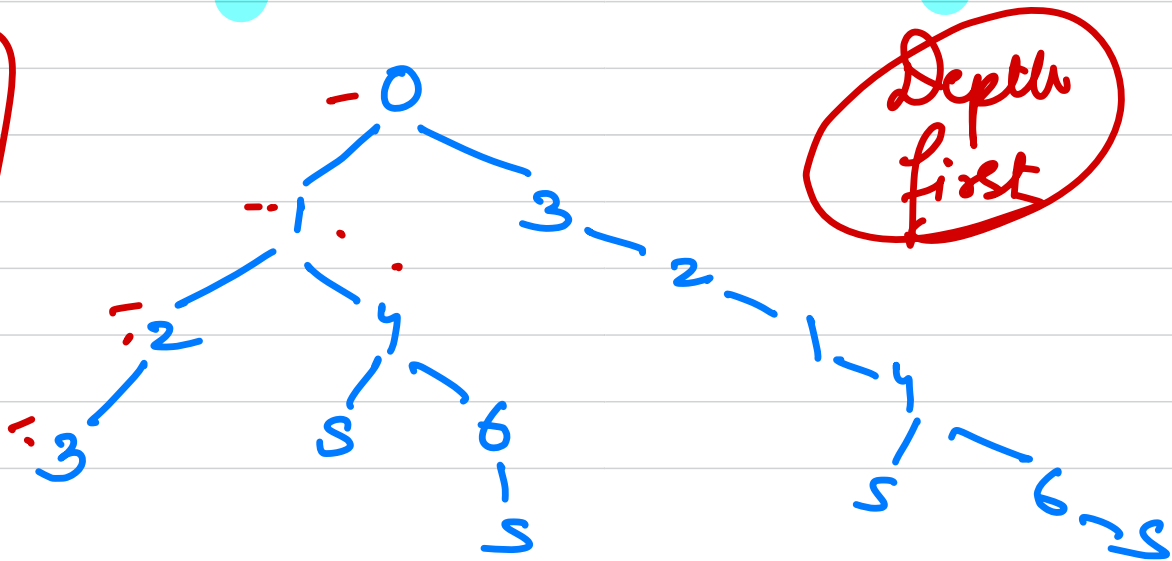


Back search

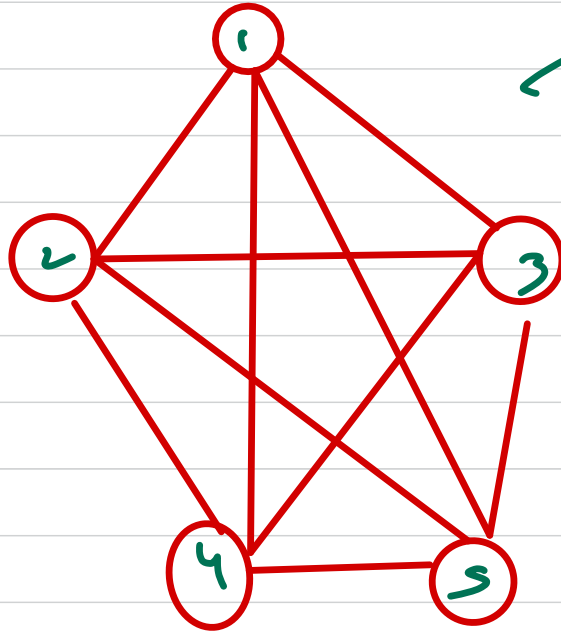


any path

$[0, 1, 2]$
 $[0, 1, 4, 5]$
 $[0, 1, 4]$



Depth first



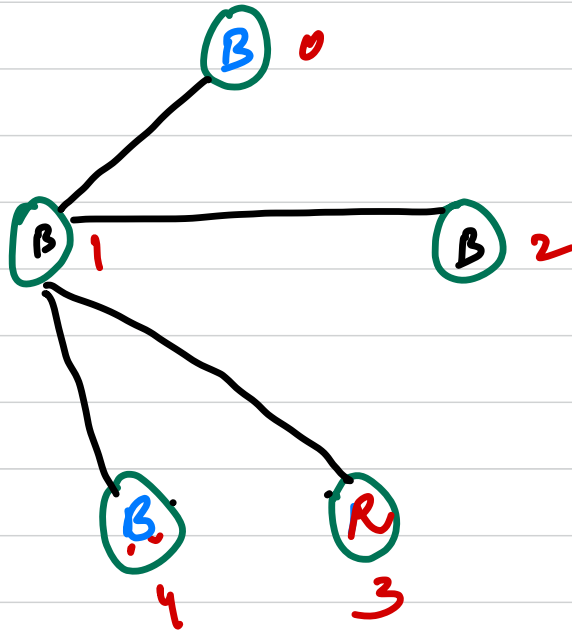
→ complete graph

1, 5

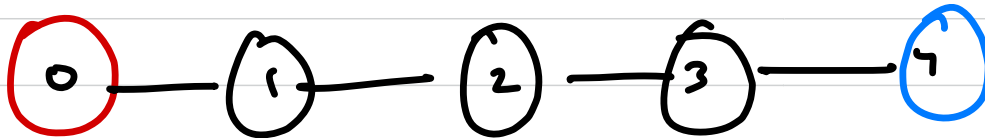
$O(v!)$

Q=1 You are given an undirected tree with v vertices. The vertices can be of any one of the following colors : Red, Blue, Black. Tree contains atleast one red and one blue node. You can remove an edge such that we get 2 trees where none of the trees has both red and blue color nodes together. Find how many such edges we can remove.

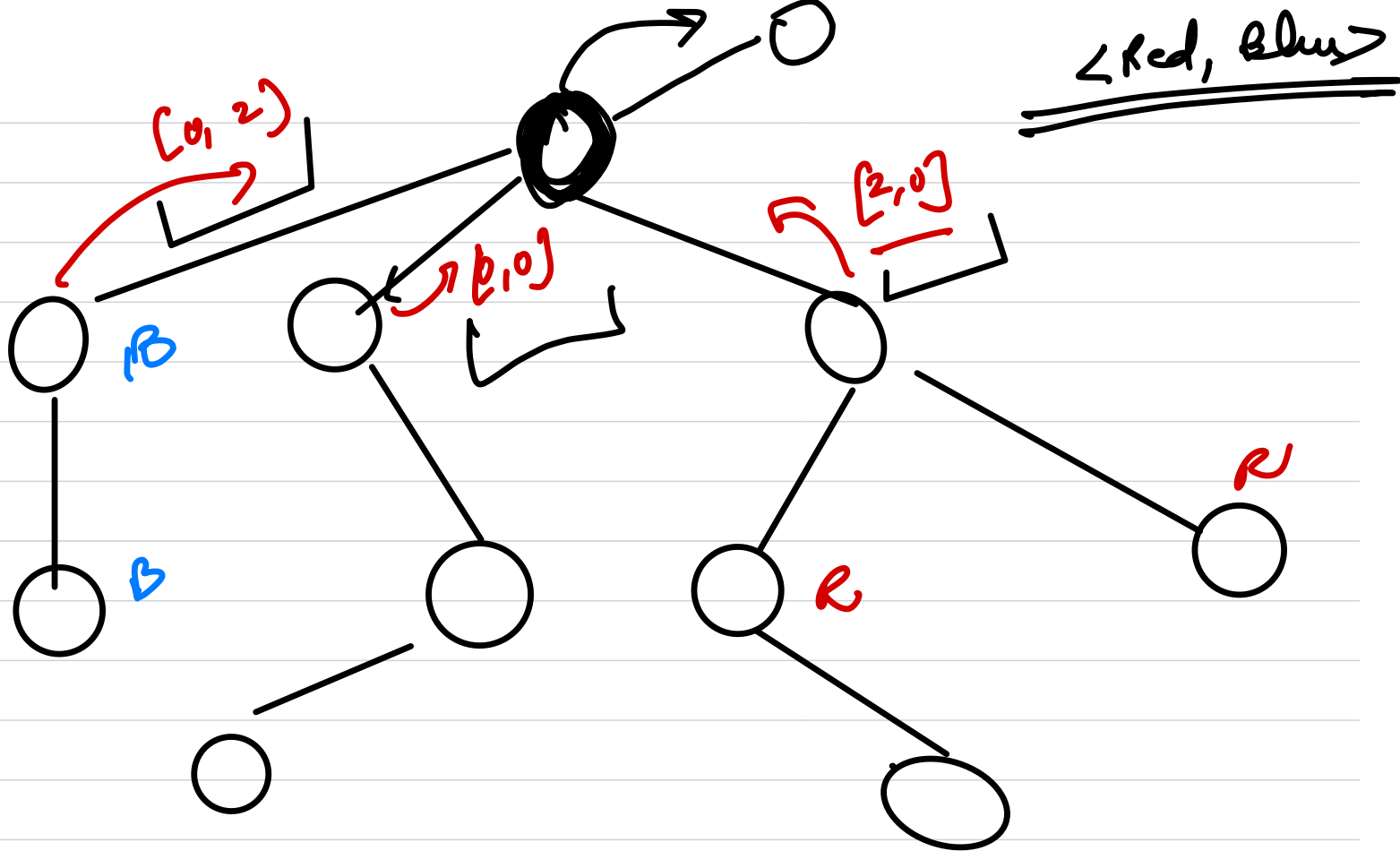
$$v \leq 3 \times 10^5$$



1 ans

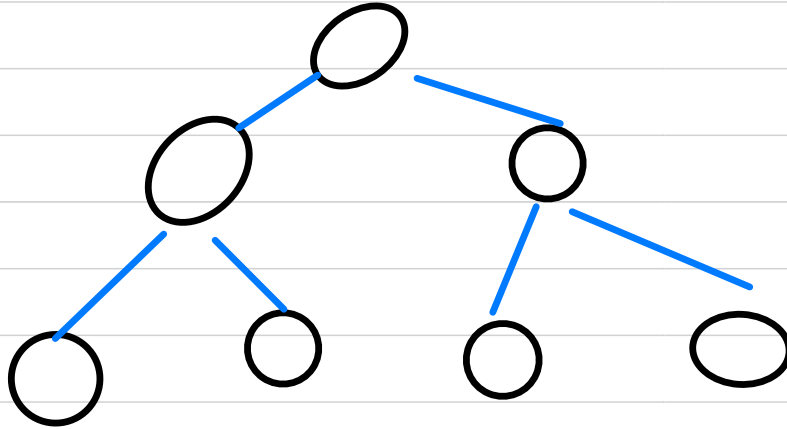


4



Tree \rightarrow V nodes

edges ???



Prove that a tree with n nodes has $n-1$

edges.

PM1 \rightarrow assume $f(n) \rightarrow n-1$

no. of edg for
 n nodes.

$$f(1) \rightarrow 0$$

$$f(2) \rightarrow 1$$

}

Base Case

assume $f(n) \rightarrow$ True

To prove $f(n+1)$

no. of edges will be $(n-1)$ + no. of edges
required for $(n+1)$ th node.

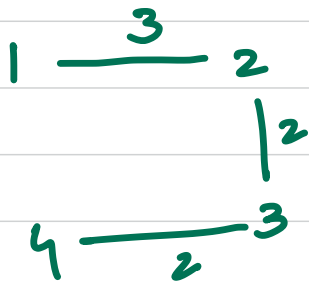
Every node that will be added to a tree
needs one edge.

$$\begin{aligned} f(\underline{n+1}) &= f(n) + 1 \\ &= (n-1) + 1 \Rightarrow \underline{\underline{(n+1) - 1}} \end{aligned}$$

$$f(\underline{n}) = \underline{\underline{n-1}} \quad \underline{\underline{H.P.}}$$

→ Koli:

(Spoj)



1 — 2

4 — 3

1 → 3

2 → 3

4 — 2

3 — 2

10



1 → 3 + 2 → 5

3 → 2 + 3 → 5

4 → 2 + 2 → 4

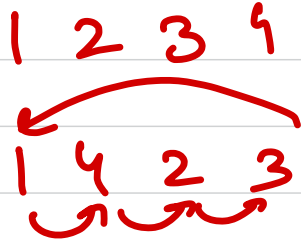
2 → 2 + 2 → 4

18 ← best

Brute force

1. 2. 3. N → we can have $N!$

permutations



N nodes



$(N-1)$ edges

2-9

1-4

$\epsilon \times O(v \times \epsilon)$

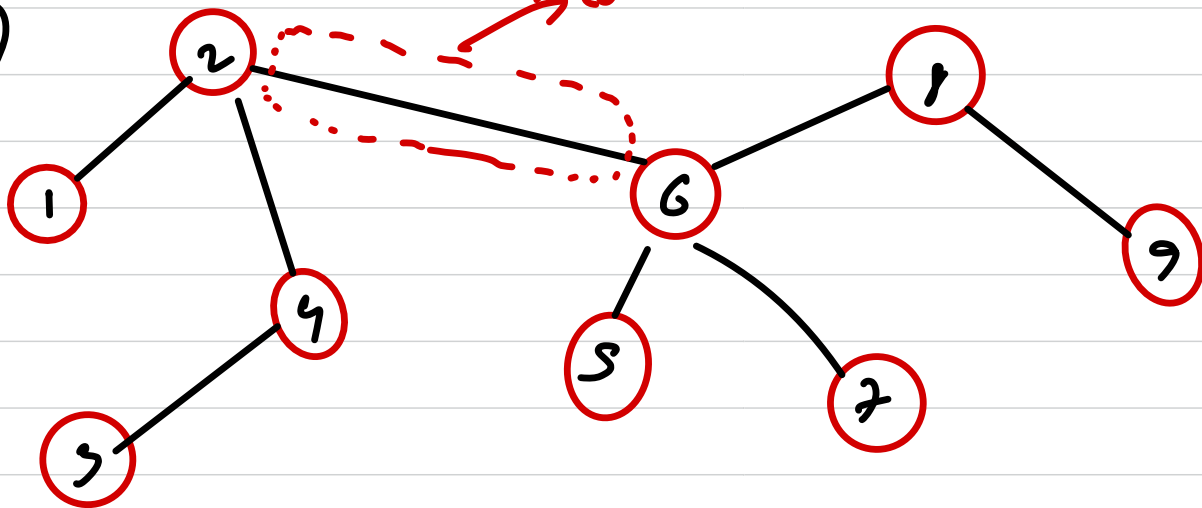
$v \times \epsilon \times \epsilon^2$

$O(v^4)$

Tree (weighted)

max contribution

of each edge

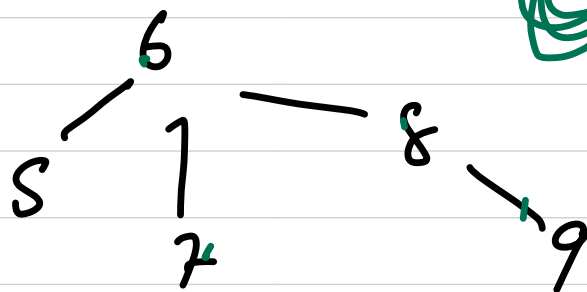
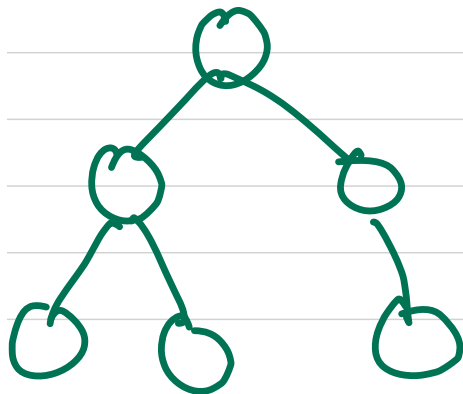
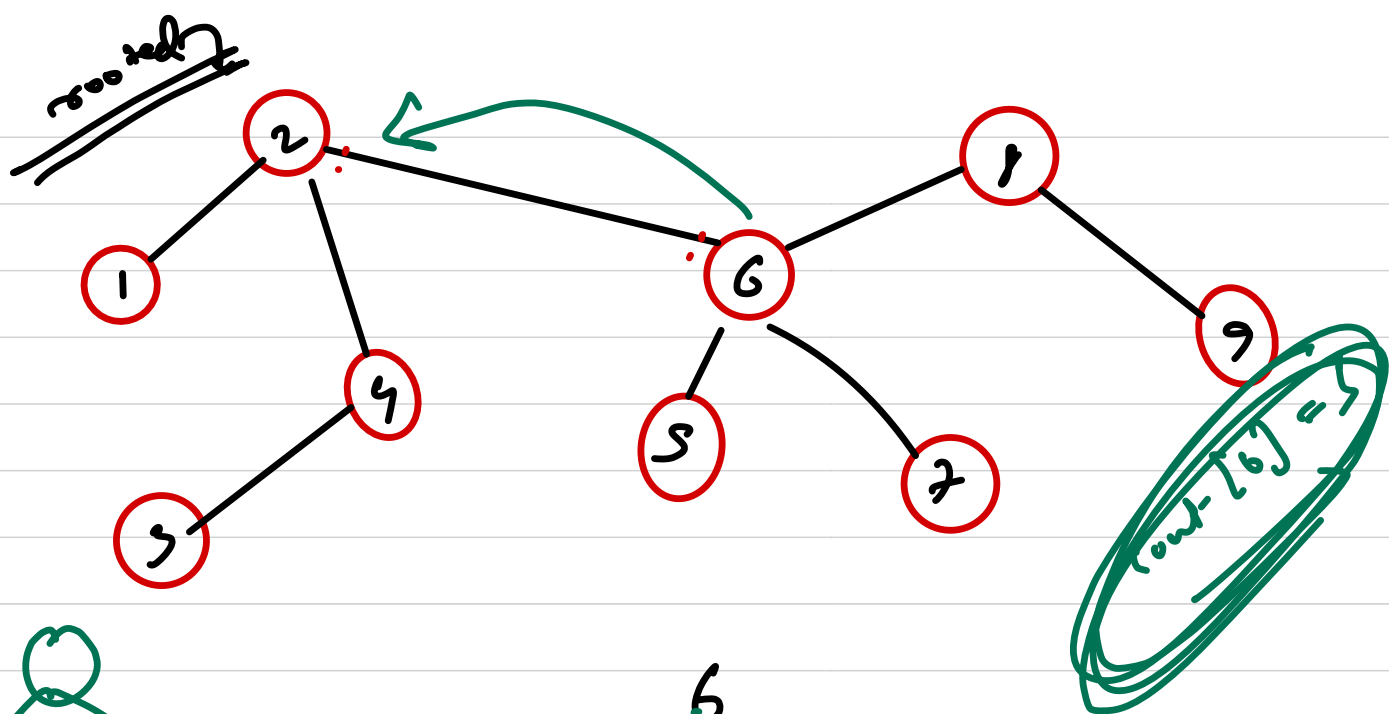


1-5
2-6

5-2
6-3 - - -

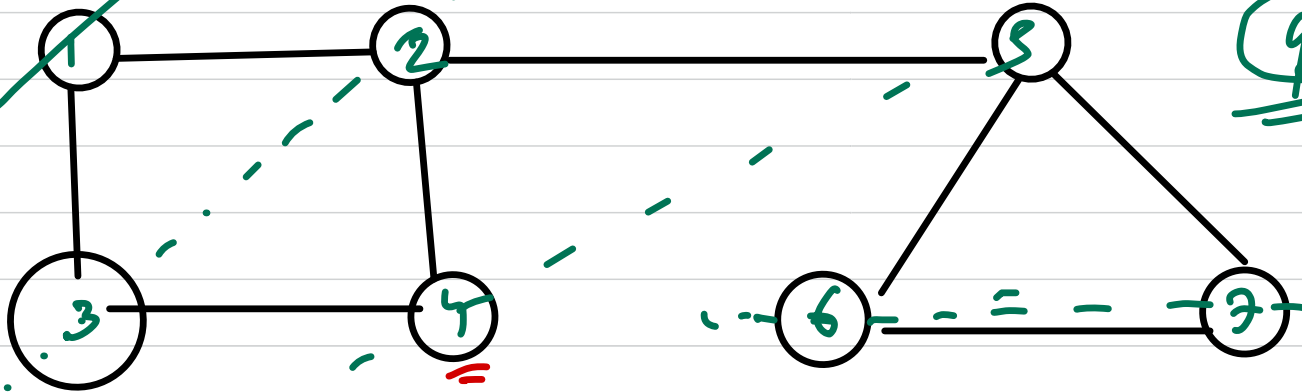
for any edge e ,

$$\text{contribution} = 2xw + x \min(s_2 - c_1, s_2 - c_2)$$



Breadth first Search

FIFO
Queue



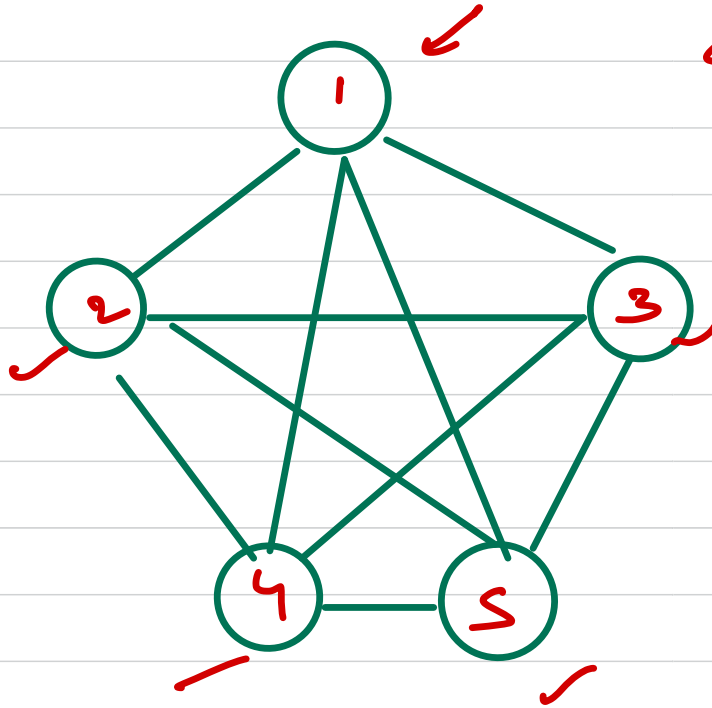
Queue

1, 2, 3, 4, 5, 6, 7

1, 2, 3, 4, 5, 6, 7

level order traversal

TC $\rightarrow O(V+E)$



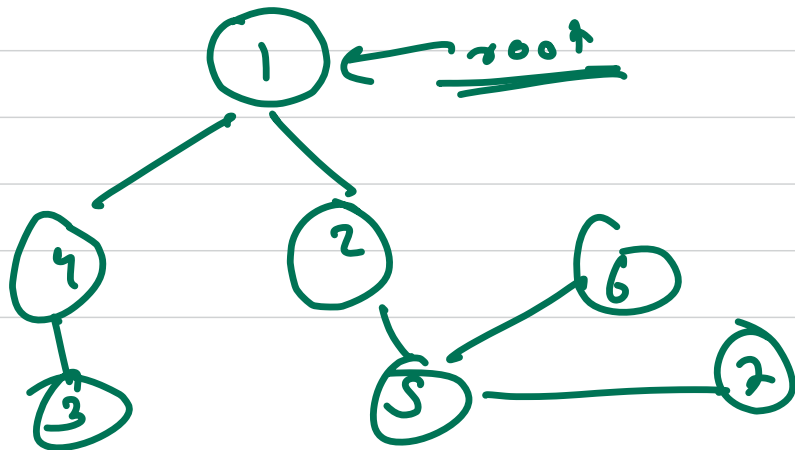
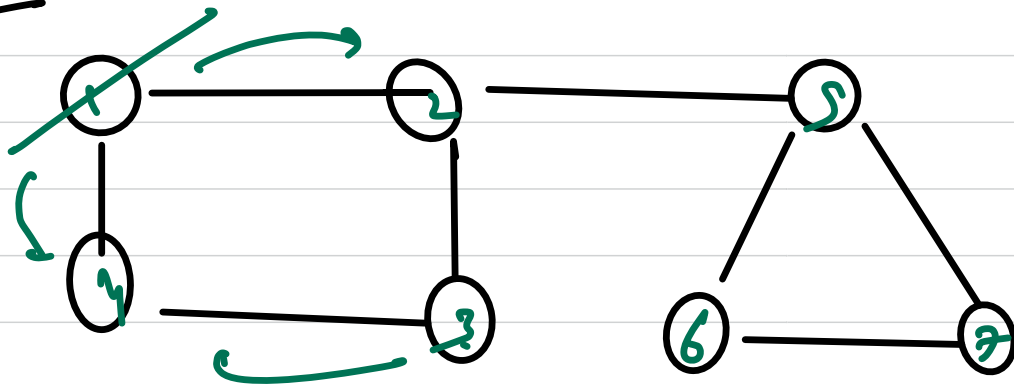
1	2	3	4	5
x	x	x	x	x

1 2 3 4 5

BFS Tree

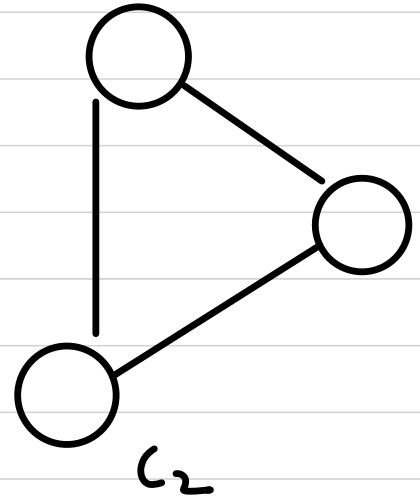
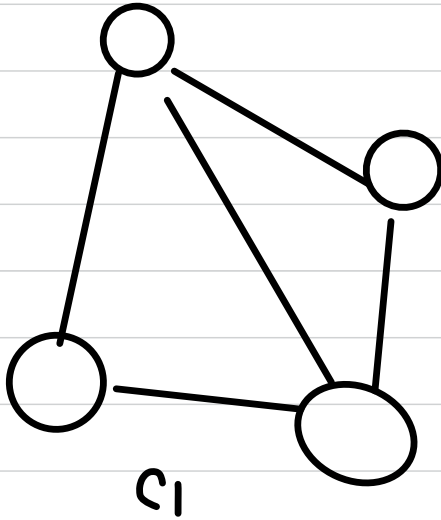
1, 4, 2, 3, 5
X X X X

parent 1 1 1 1 1



BFS Tree

* Connected Components



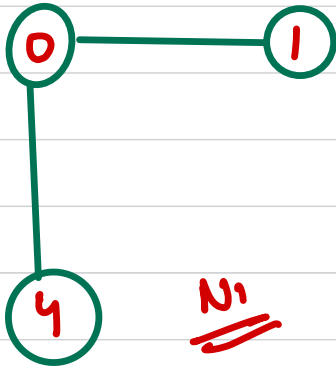
5/1

Journey to the moon

nC_2 $\textcircled{1}C_2$ \rightarrow Total

Total - Σ pairs of sum neighbors

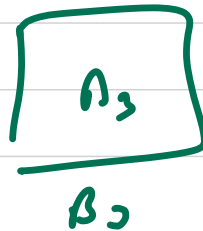
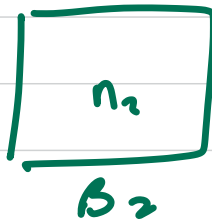
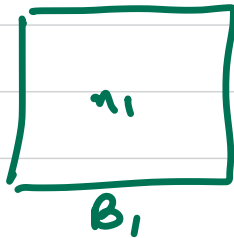
dfs



N_1

N_2

of sum of calls
dfs = # of cc



...

Q You have a permutation of N integers $1, 2, 3, \dots, N$.

You want to rearrange this permutation to some other permutation P . You can swap only M given pairs, whatever no. of times you want. Figure

out if this is possible or not?_?

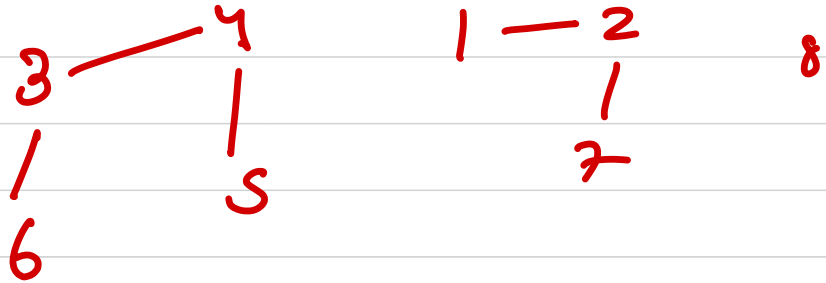
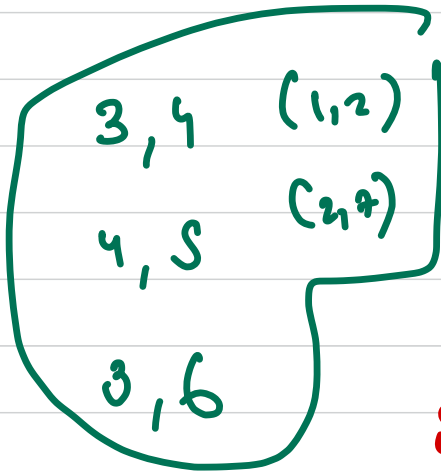
$G \rightarrow 1, 3, 2, 4$ $(3, 4)$

$P \rightarrow 1, 4, 2, 3$

\rightarrow Yes

→ connected component ← direct application

index cc



Set $\rightarrow G \rightarrow [3, 4, 5, 6]$

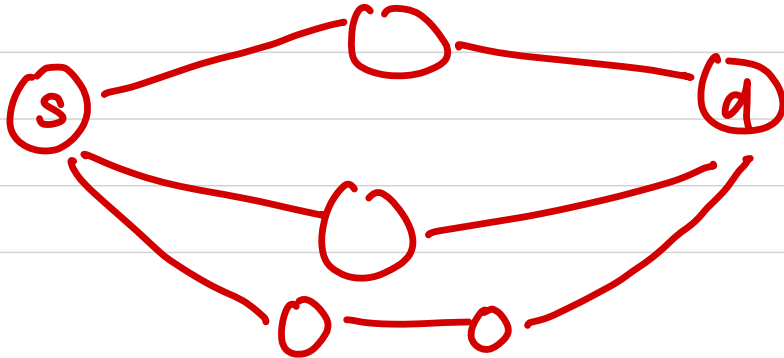
$G \rightarrow$ 1 2 3 4 5 6 7 8

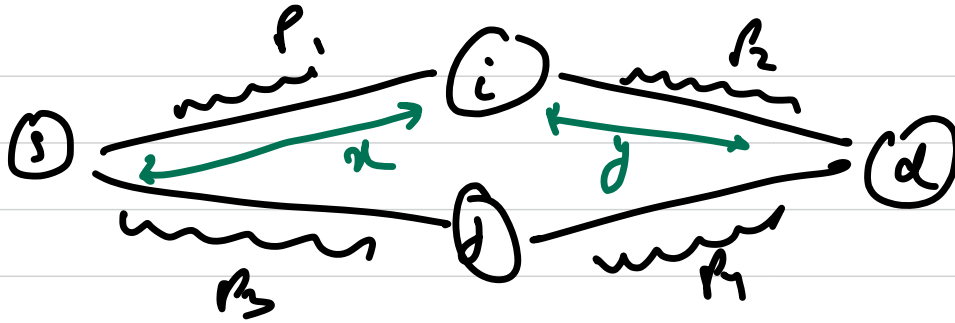
$P \rightarrow$ 7 1 4 6 5 3 2 8 //

all the elements in one connected component
can be arranged in any permutation under
some swaps

→ BFS → graph → Shortest path

Q ⇒ Given a graph, give src & dest nodes,
find all the nodes which are part of
atleast 1 Shortest path.





$$\underline{s[i] + d[i] = d}$$

$$\underline{O(V+E)}$$

$$\underline{y = d - x}$$

$$\underline{x + y = d}$$