"Graph - Representaions and their comparisons"

Graph $G = (V, E)$

$V \rightarrow$ set of Vertices

$E \rightarrow$ set of Edges

$|V| = n \qquad |E| = m$

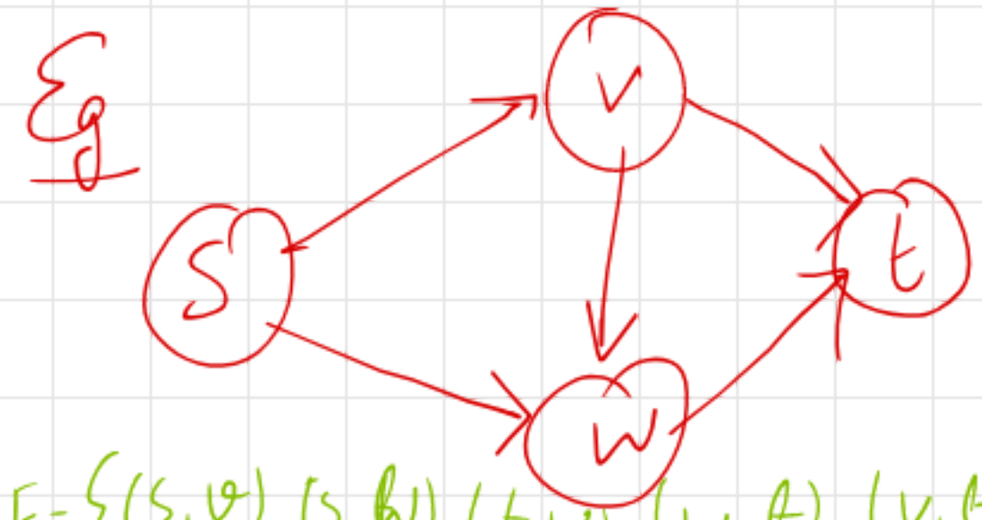
Directed graph Each Edge is an ordered pair

(v, w) is an edge

$v \longrightarrow w$

head ↑ tail

Eg

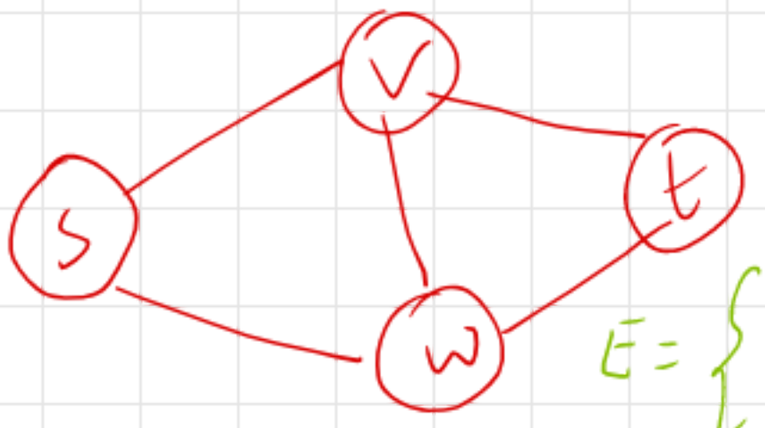


$E = \{(s,v), (s,w), (t,w), (w,t), (v,t)\}$

Undirected graph Each edge is an unordered pair

$\{v, w\}$

$v \text{——} w$



$E = \{\{s,v\}, \{s,w\}, \{v,w\}, \{v,t\}, \{t,w\}\}$

In an undirected graph, an edge with endpoints v and w can be denoted by (v, w) or (w, v) — there is no difference between the two.

Directed graph

$$|E| \leq 2\binom{|V|}{2} = O(|V|^2)$$

Undirected graph

$$|E| \leq \binom{|V|}{2}$$

$$= O(|V|^2)$$

$$|E| = O(n^2)$$

Thm:

$$\sum_{v \in V} \deg(v) = 2|E|$$

↳ Undirected graph

for directed graph,

degree →
- indegree → $\deg^-(\,)$
- Out-degree → $\deg^+(\,)$

$$\sum_{v \in V} \deg^+(v) = |E|$$

It cld hv been summation of out degrees exclusively as well

Representation of graph

Adjacency Matrix

n-vertices

$A =$ matrix

	V_1	V_2	V_3	...	V_n
V_1					
V_2					
V_3					
⋮					
V_n					

$$A[i,j] = \begin{cases} 1 & \text{if } (v_i, v_j) \in E \\ 0 & \text{otherwise} \end{cases}$$

Undirected graph → Symmetric matrix

$A[i,j] = 1 = A[j,i]$

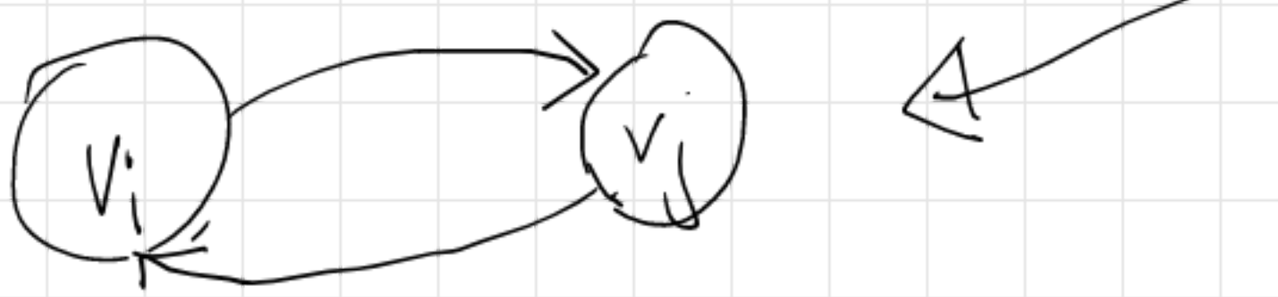
directed graph

$A[i,j] = 1$

$\not\Downarrow$ doesn't imply

$A[j,i] = 1$

$[A[i,j] = 1$ & $A[j,i] = 1]$



Space to store the matrix $(n \times n)$

$= O(n^2)$

→ Time to check if there is an edge between V_i and V_j,

→ Check entry of A[i,j]

↳ O(1)

→ Suppose the graph is sparse

⇓

less no. of edges

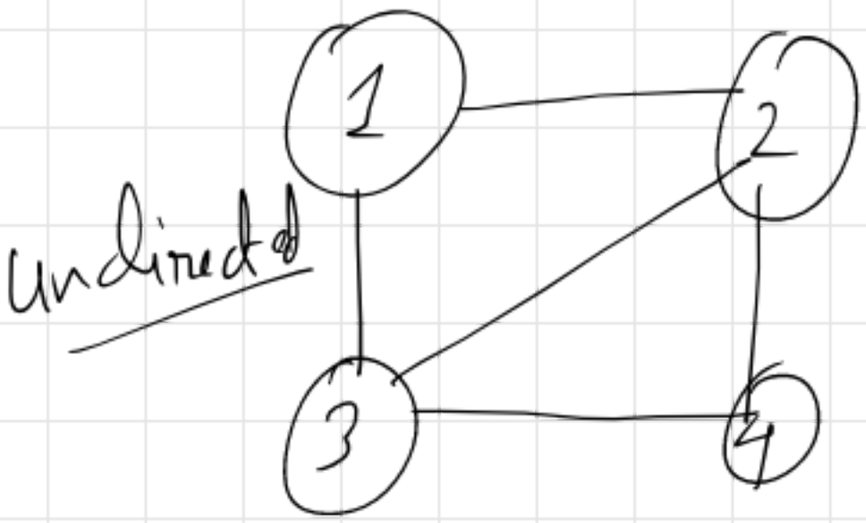
→ then this matrix representation will have more no. of zeros

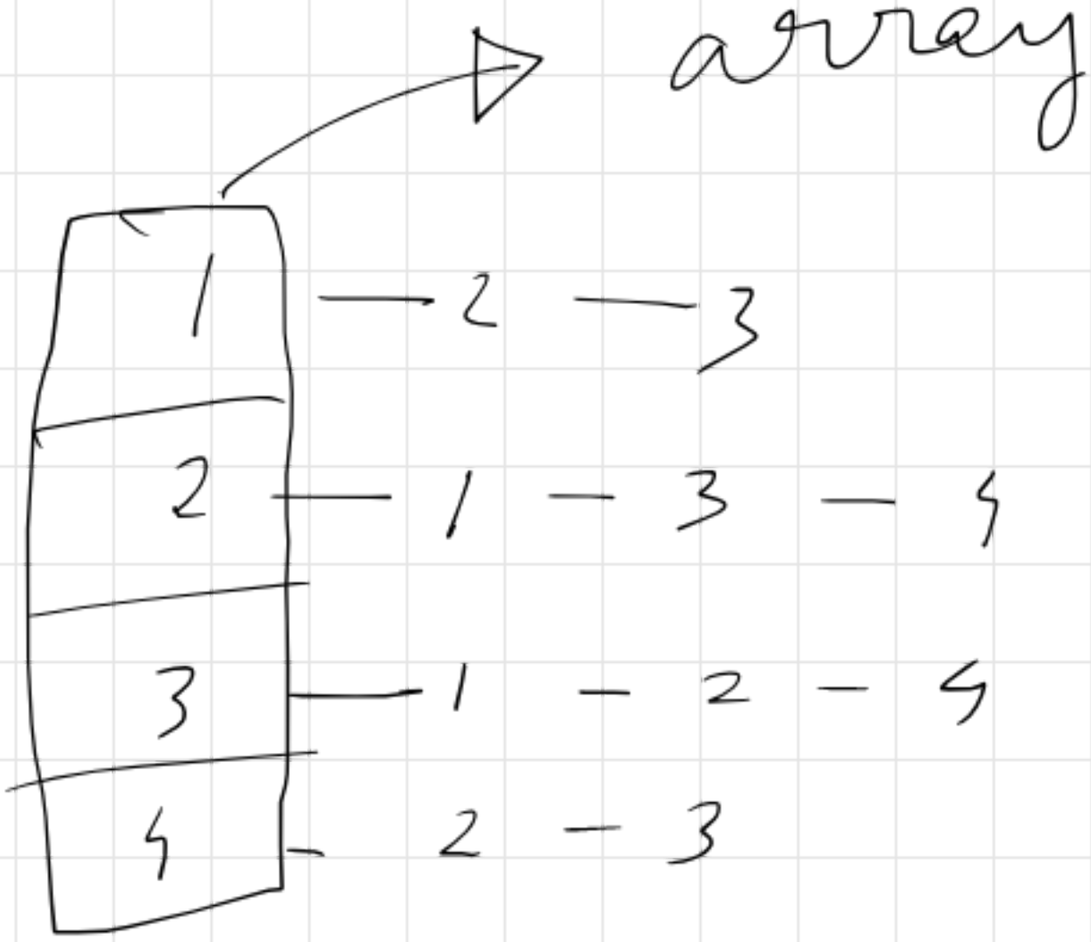
⇓

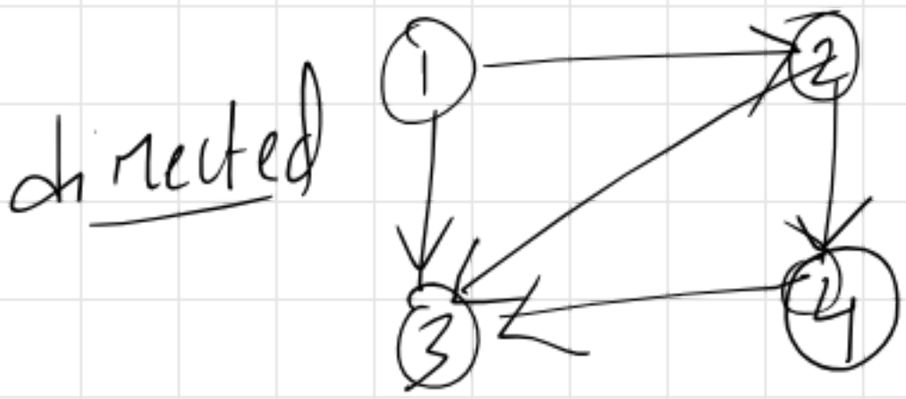
wastage of memory

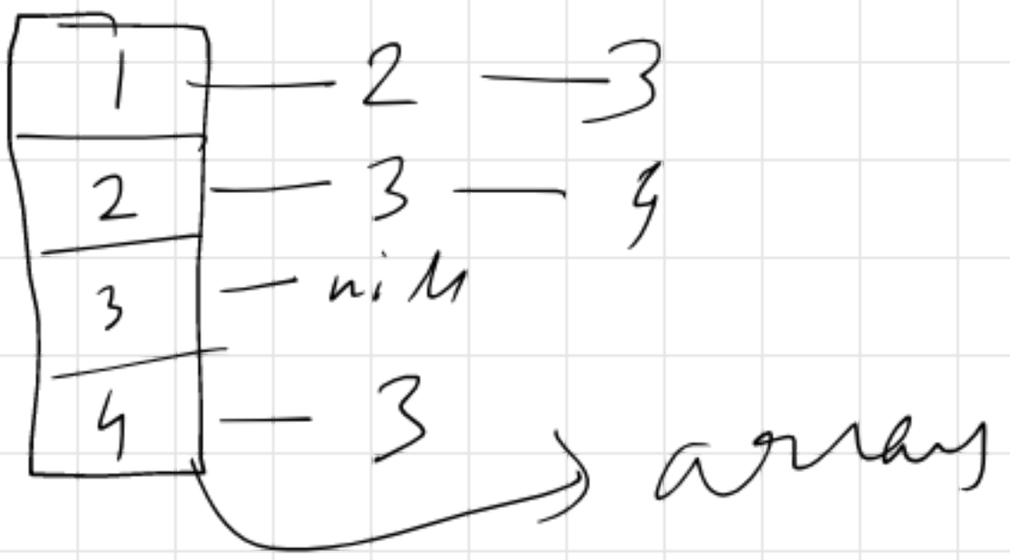
So, for sparse graph : not a good representation

Adjacency List









Space require to store Adjacency List for graph $G = (V, E)$ (both directed and undirected) is $O(V + E)$

Time to check there exists an edge between V_i and V_j (undirected)

→ Go to i^{th} index of array, and traverse the link list associated with it → $O(\deg(V_i))$

For undirected graph G,

$O(\min\{deg(v_i), deg(v_j)\})$

by simultaneously scanning the neighbors of both v_i and v_j; by stopping either when we locate the edge.

More insightful Thoughts

Link Lists are not the only Data Structures we could use;

any other structure that support Searching, Listing, insertion, and deletion will do.

For eg: we can reduce the time to determine whether $v_i v_j$ is an edge to $O(\log(deg(v_i))$ by using BBST to store the neighbors of v_i.

Comparisons

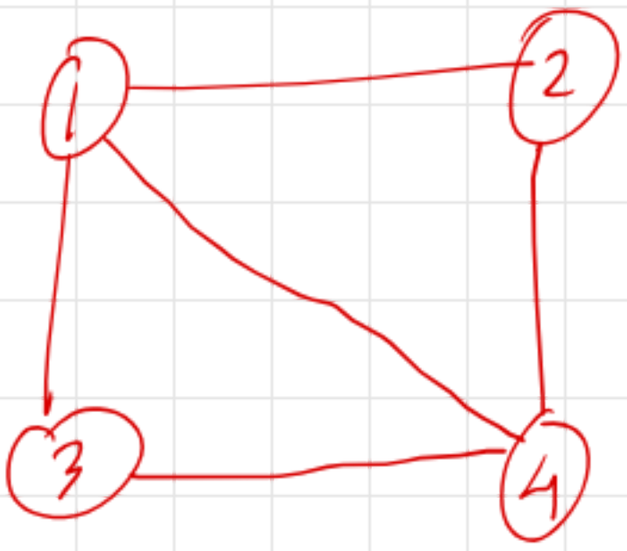
$|V| = n \quad |E| = m$

$G = (V, E)$

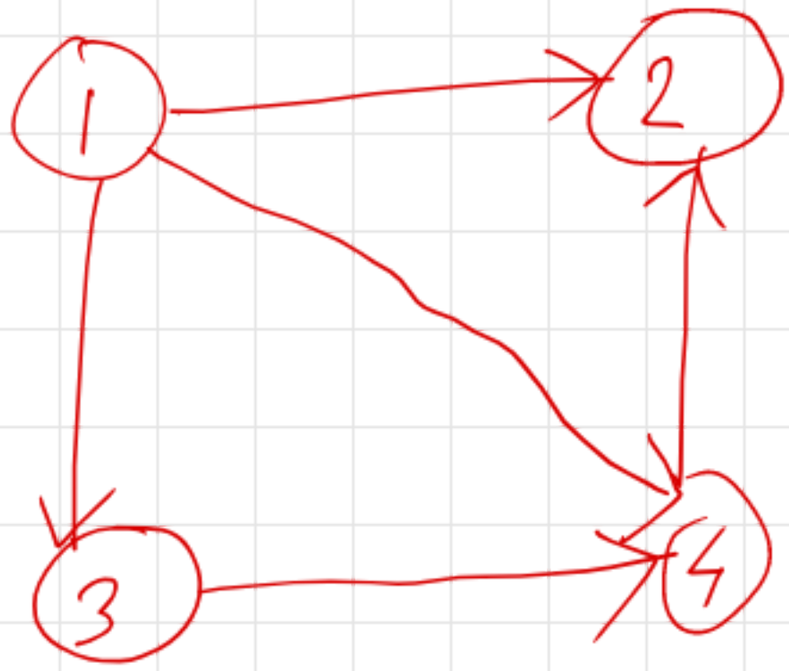
	Adjacency List	Adjacency Matrix
Space	$O(m+n)$	$O(m+n)$
Test if $uv \in E$	$O(\min(\deg(u), \deg(v)))$	$O(1)$
Test if $u \to v \in E$	$O(\deg(u))$	$O(1)$
List all edges	$O(m+n)$	$O(n^2)$
insert an edge	$O(1)$	$O(1)$
delete an edge uv	$O(\deg(u) + \deg(v))$	$O(1)$

⟨Important Assignment⟩

a)



b)



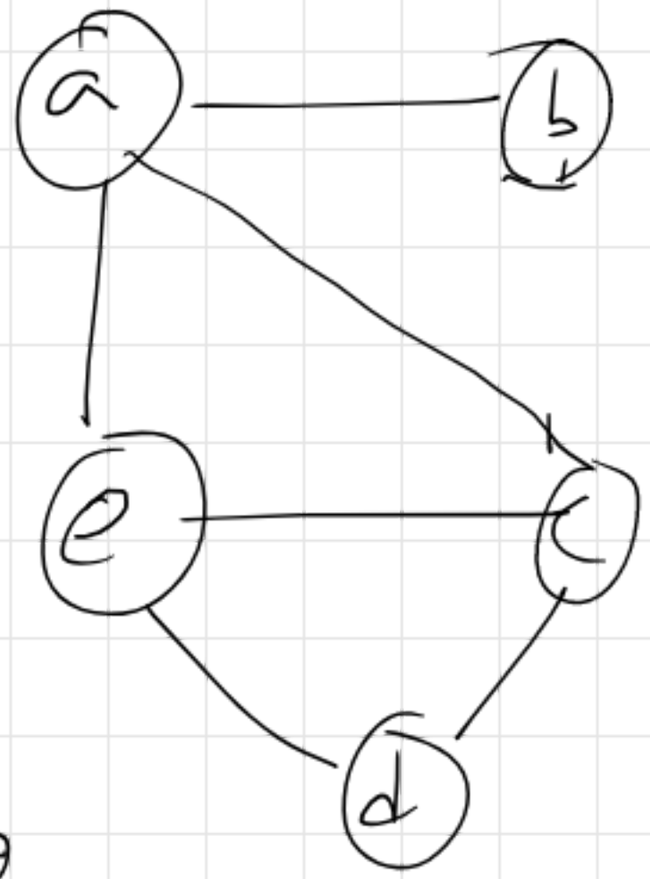
> Implement both graphs using Adjacency Matrix and Adjacency List

> Find neighbor of node 1 in the above graphs using both representations

> Look at the comparison table, and try to implement all the operations using both representations on the above mentioned graphs.

Terminolog (contd..)

Path: sequence of nodes $v_1 v_2 v_3 \ldots v_{k-1} v_k$
such that each consecutive pair
v_i, v_{i+1} is adjacent



1. aecd ? $\xrightarrow{\text{Path}}$ Yes

Check
(a, e) → Yes
(e, c) → Yes
(c, d) → Yes
So path

2. aecde → is this a path ?

Yes

abcd ? → no (b, c) ∉ E

Simple path all nodes are distinct

1. aecd is a simple path
2. aecde is not a " "

Cycle is a path $v_1 v_2 \cdots v_k$

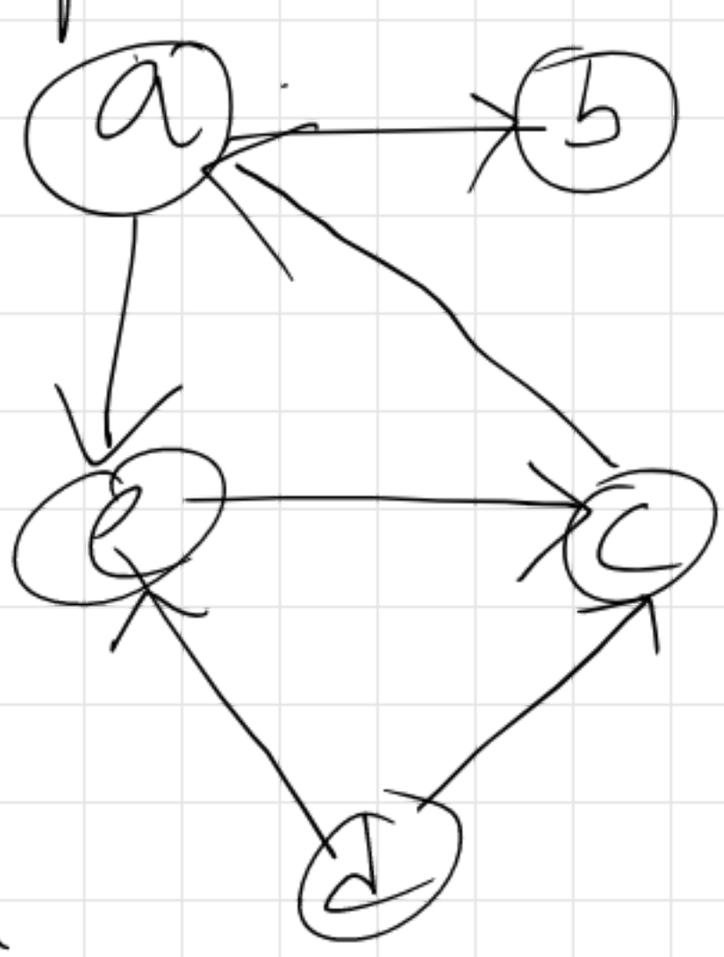
$[k > 2]$

→ first $k-1$ nodes are distinct.

→ $v_1 = v_k$

a simple path where first and last nodes are same

Path in a directed graph



(a e c d)

is not

a path

$(v_i\ v_{i+1})$ must be an edge

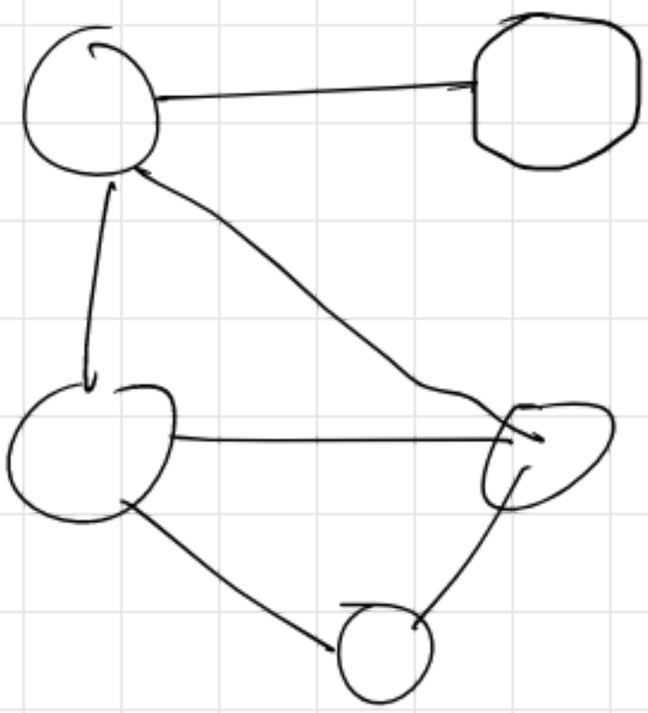
$v_i \longrightarrow v_{i+1}$

Connected graph

for every pair of nodes u & v
there is a path between u and v

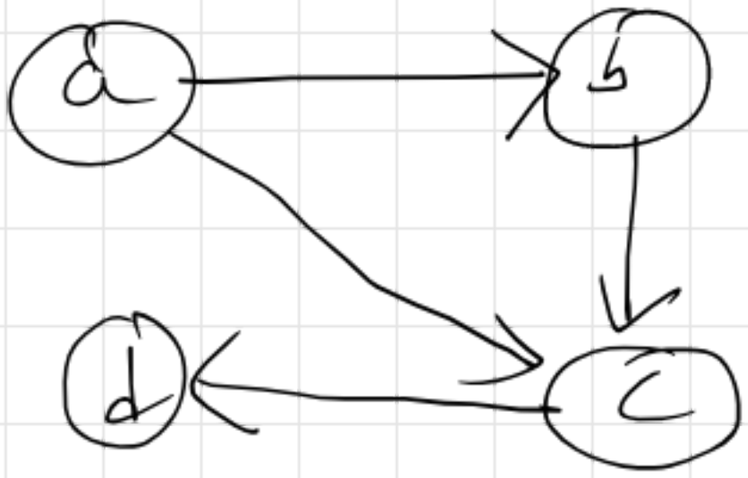
[undirected graph]

is this connected?



Yes

Notion of connected means in Directed graph



a ⇝ d
path ✓

d ⇝ a
No path exists

path from a to d doesn't imply d to a

Strongly connected:
for every pair u & v,
u ⇝ v v ⇝ u

Path with as less hops possible

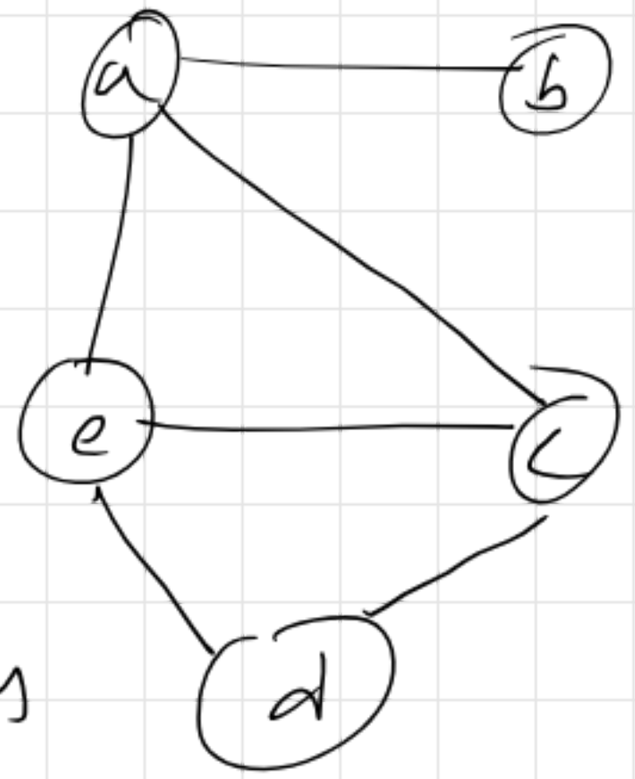
no. of edges on a path

Path b^tn a and d

a e c d → 3 edges

a c d → 2 edges

a e d → 2 edges



I am interested in path with minimum hops b^tn two vertices

Distance b^tn 2 nodes: [minimum no. of edges on a path b^tn 2 nodes]

Tree a graph
→ Connected graph
→ has no cycle

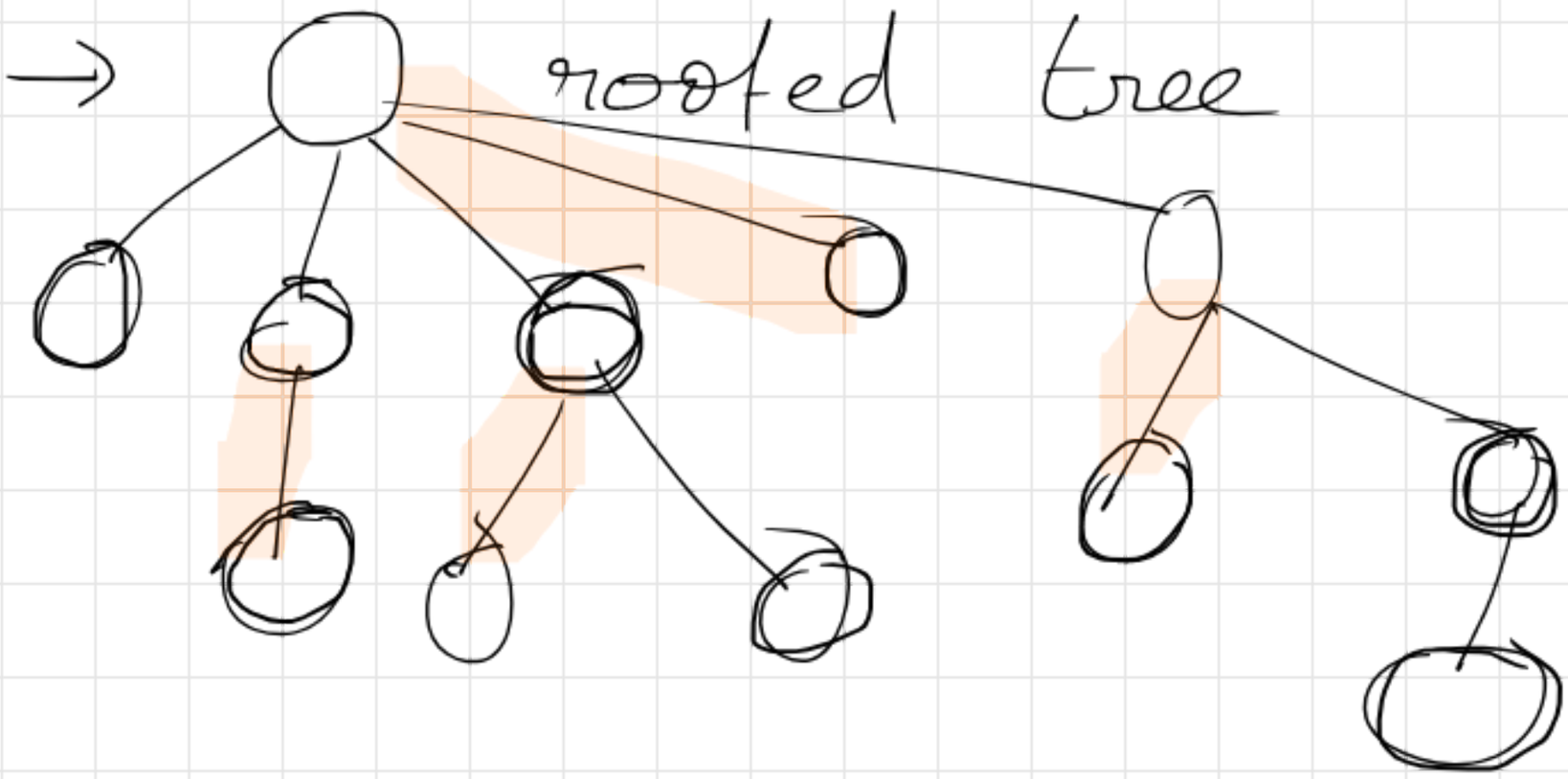
Tree is a minimally connected graph

[deleting an edge would disconnect it]

Leaf node

degree = 1

How many edges does a tree with n nodes? $n-1$ edges

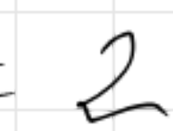


[idea]

[There is a unique edge corresponding to every node except root node.]

Proof by induction

Base case $n = 2$



∴ Tree is connected & no cycle

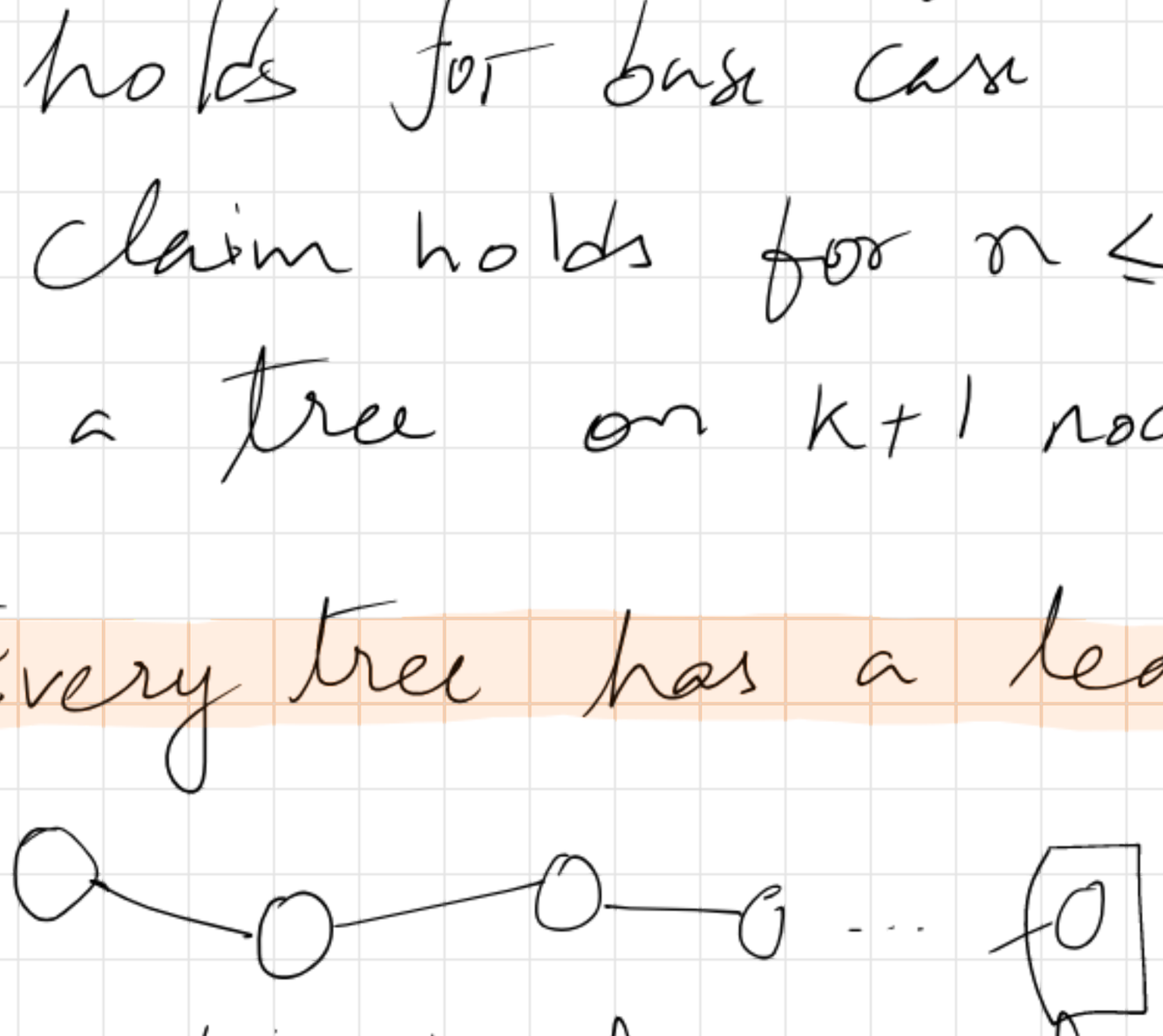
→  → only one edge possible

Claim holds for base case

Ind hyp: Suppose claim holds for $n \leq k$

Ind step: Given a tree on $k+1$ nodes

Observation: Every tree has a leaf



Can I come to already visited nodes ? → no

You stop when you arrive at a node from which you cannot go out

Given a tree on $k+1$ nodes,

Clip a leaf node from the tree

⇒ Will the resulting structure be a tree of k nodes?

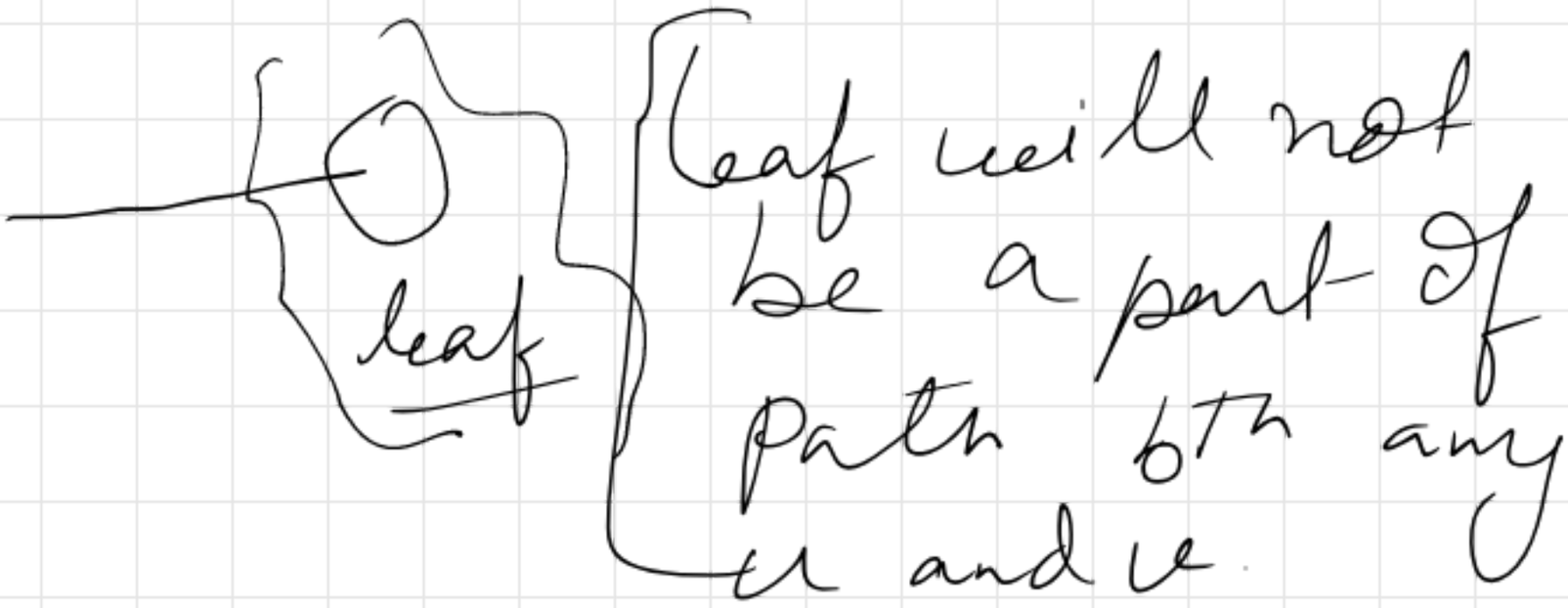
→ There will be k nodes ↩

→ Will it be a tree?

→ Yes -

→ removing a leaf node ensures that there will not be a cycle

→ If I remove a leaf, it is still connected ?? → why



→ So, after removing a leaf, will have a tree of k nodes.

Apply ind hyp → k-1 edges on the resulting tree

deleted 1 edge

Total edges = k-1 + 1 = k

— ged -

Let G be an undirected graph on n nodes.

Any two of the following statements imply the third

1) G is connected

2) G doesn't contain a cycle

3) G has $n-1$ edges

$$\left[\begin{array}{l} (1) \& (2) \longrightarrow 3 \\ (1) \& (3) \longrightarrow 2 \\ (2) \& (3) \longrightarrow 1 \end{array}\right]$$

HW check the above claim.