Lecture 5

Announcements: HW1 DUE TODAY AT 6:59 PM

HW2 WILL BE POSTED TODAY

Quiz 3 WILL ALSO GO UP TODAY (10 MINUTES).

Last class: Graphs

$G = (V, E)$

V: Vertices / Objects

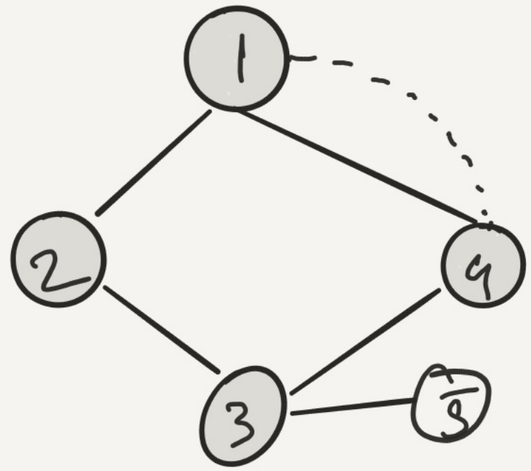
E: "Relationships / Edges"

UNDIRECTED GRAPHS

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

$E = \{$ pairs of vertices $\}$

Ex:



$V = \{1, 2, 3, 4, 5\}$

$E = \{\{1,2\}, \{1,4\}, \{2,3\}, \{3,4\}, \{3,5\}\}$

DIRECTED GRAPHS

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

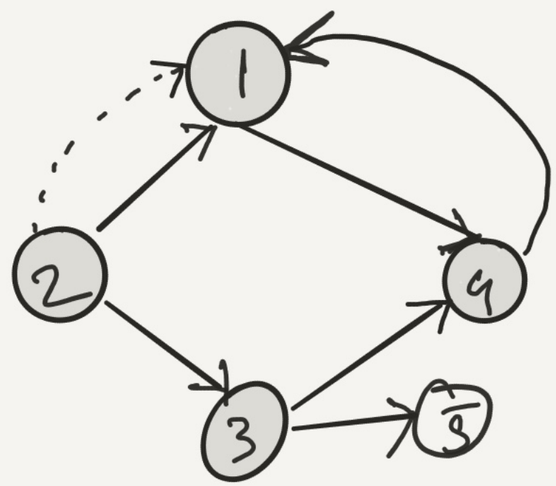
$E = \{$2-tuples of vertices$\}$

$= \{$"ordered pairs" of vertices$\}$

$e = (u, v)$

$V = \{1, 2, 3, 4, 5\}$

Ex:



$E = \{(1,4), (2,1), (3,3), (3,5), (3,4), (4,1)\}$

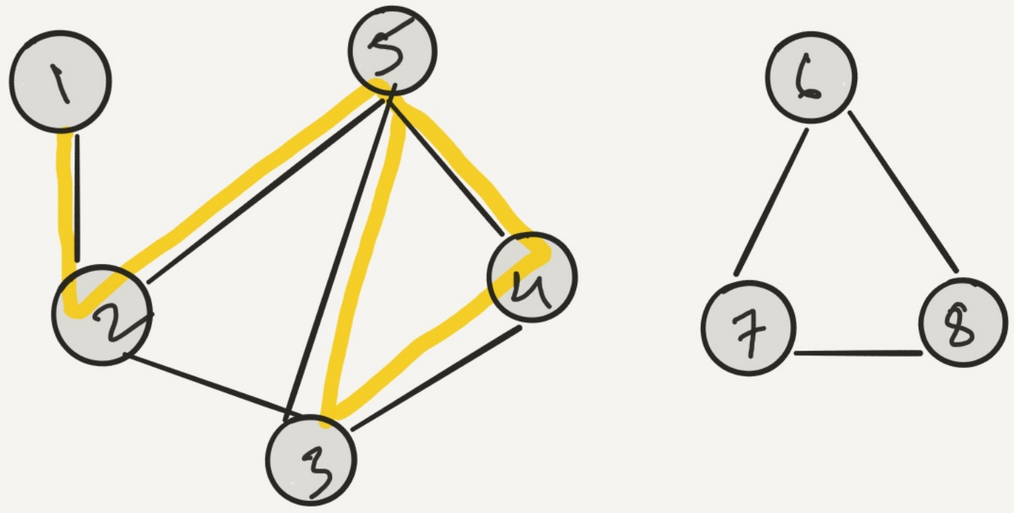
Examples of directed graphs: Geneology trees, Flight schedules, Dependencies among processes, Client-Server communication, Twitter "followers" graph

---

Path: $G = (V, E)$. A sequence of vertices $u_1, u_2, \ldots, u_k$ is a path if $(u_\ell, u_{\ell+1})$ is an edge in G.

$\forall \quad \ell = 1, \ldots, k-1$.

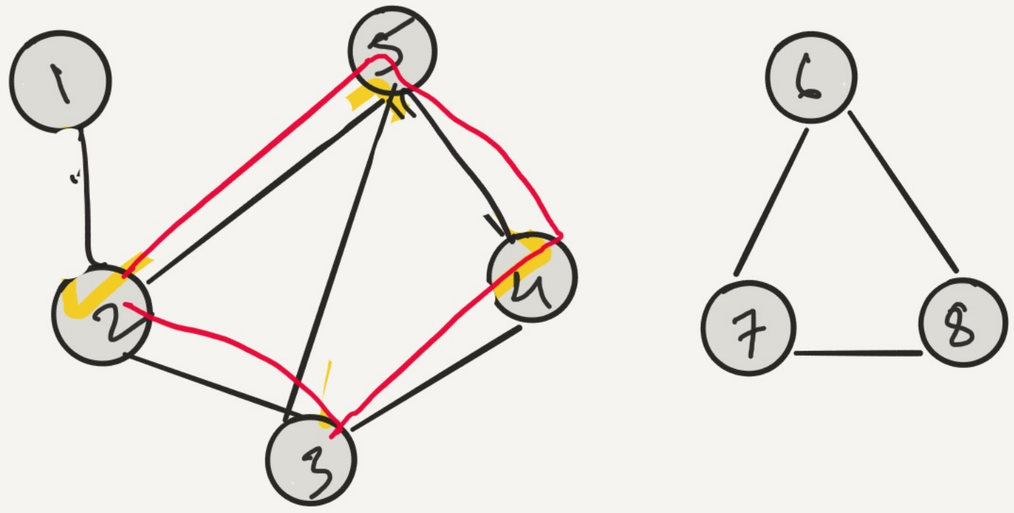
Path from u_1 to u_k.



(1, 2, 5, 4, 3) is a simple path.

(1, 2, 5, 4, 3, 5) is also a path

Simple Path Is a path where all vertices are distinct.



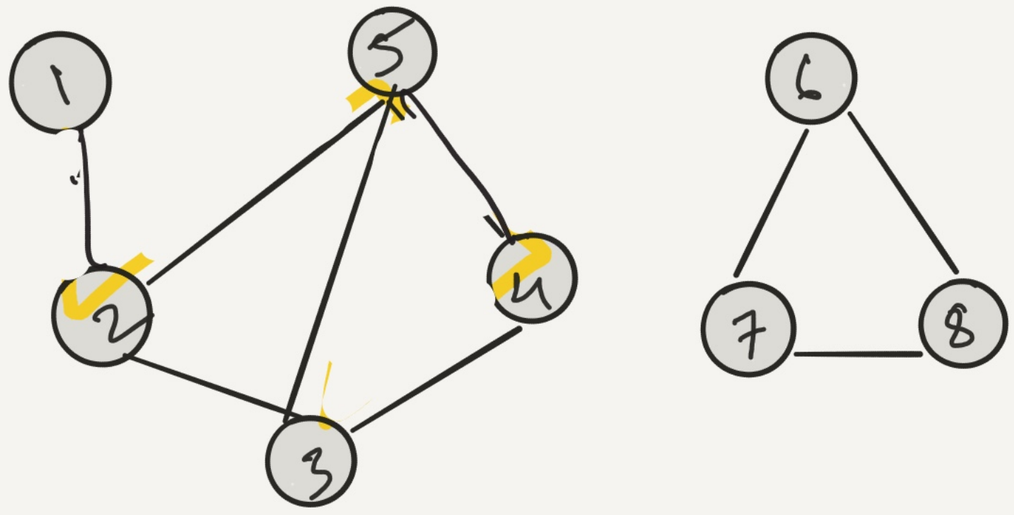
$(1, 2, 5, 4, 3)$ is a simple path.

$(1, 2, 5, 4, 3, 5)$ is also a path

$(2, 5, 4, 3, 2)$ is a CYCLE

Simple Path Is a path where all vertices are distinct.

CYCLE Is a sequence of vertices $v_1, v_2, \ldots, v_k$ where the end is the same as the start. and $(v_1, v_2, \ldots, v_{k-1})$ are distinct.



Simple
(1, 2, 5, 4, 3) is a path.

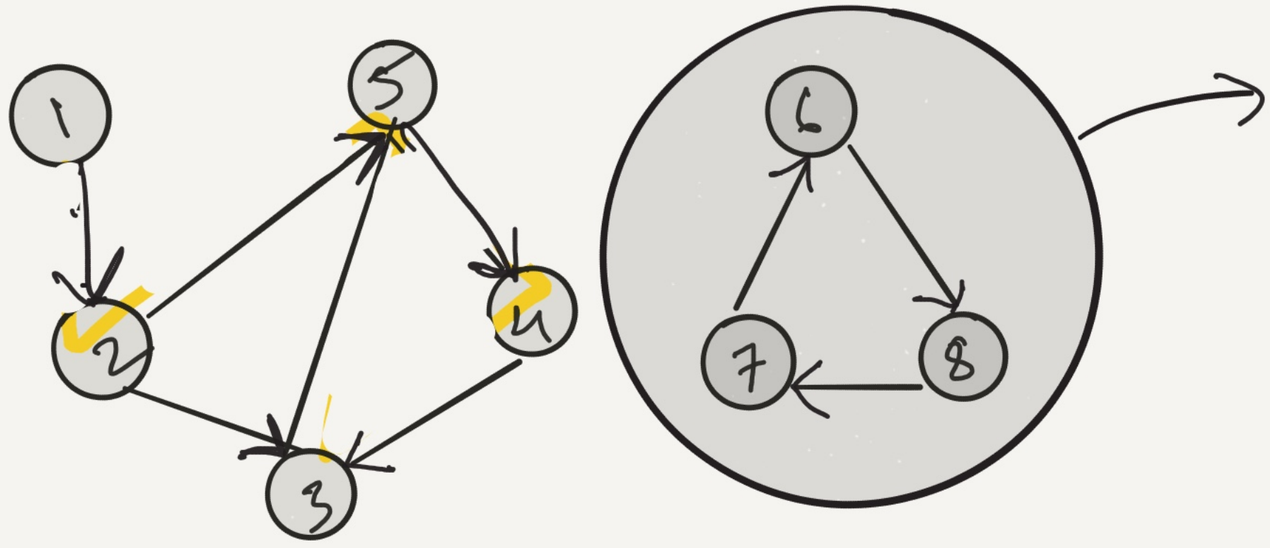
(1, 2, 5, 4, 3, 5) is also a path

→ NOT CONNECTED.

CONNECTIVITY: u and v are connected if there is a path from u to v.

∃ a path $P = (u, v_2, v_3, \ldots, v_{k-1}, v)$.

G IS <u>CONNECTED</u> IF ANY TWO VERTICES ARE CONNECTED

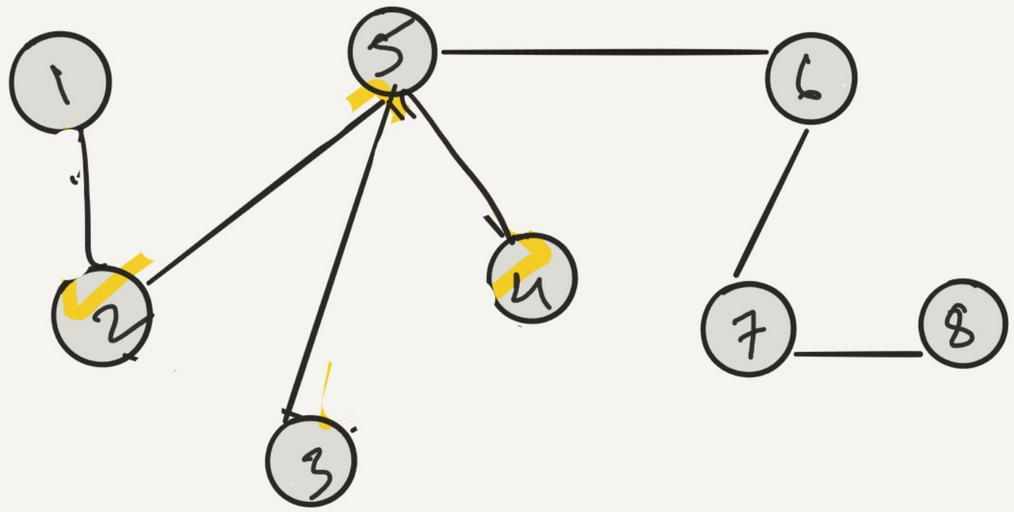


This piece is Strongly Connected.

CONNECTIVITY: u and v are connected if there is a path from u to v.

$\exists$ a path $P = (u, v_2, v_3, \ldots, v_{k-1}, v)$.

STRONGLY CONNECTED: G is Strongly connected if $\forall u, v$, $\exists$ a path from u to v and from v to u.

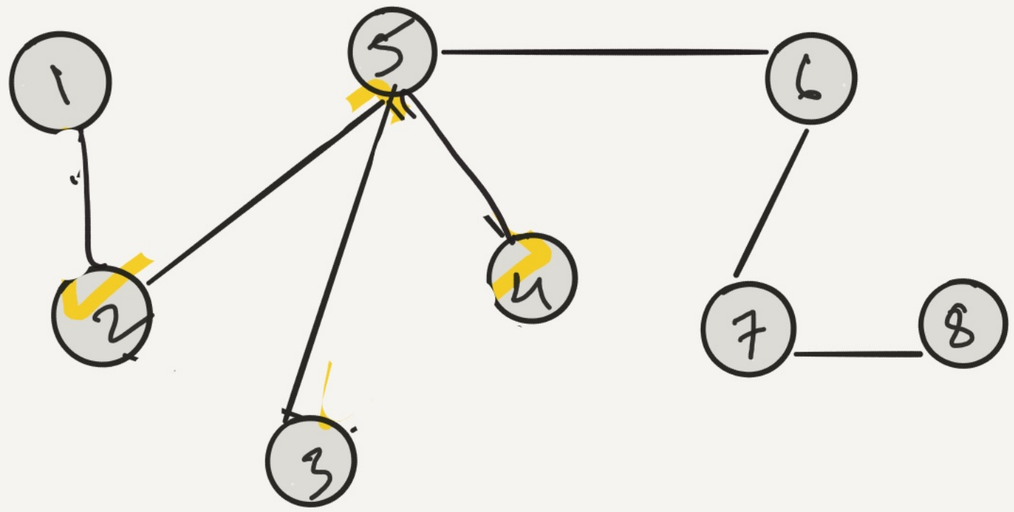


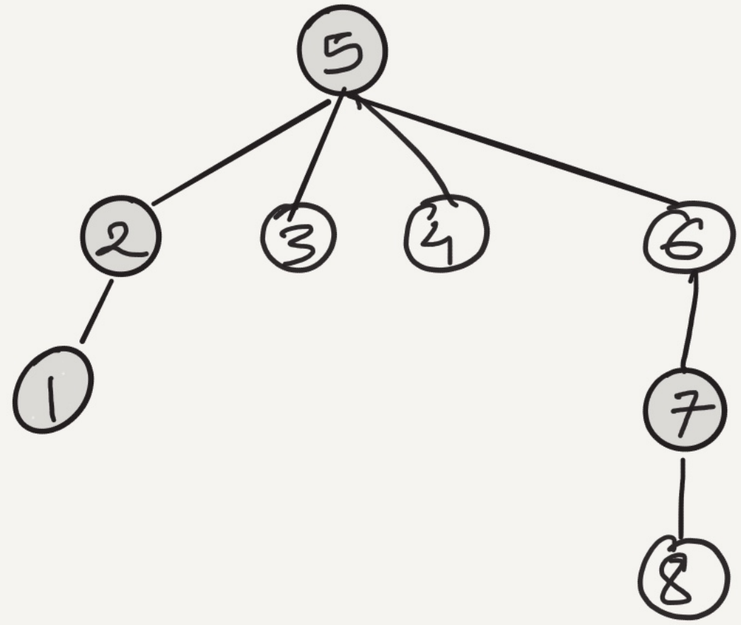
} A Tree.

TREE: A tree is a graph $G = (V, E)$ that is

(a) Connected

(b) There is no cycle in the graph.





ROOTED TREE: Is a tree where there is a designated "root" node & and an ordering of vertices so that edges go away from root.

GRAPH CONNECTIVITY (s-t CONNECTIVITY PROBLEM)

INPUT : $G = (V, E)$; TWO VERTICES s and t

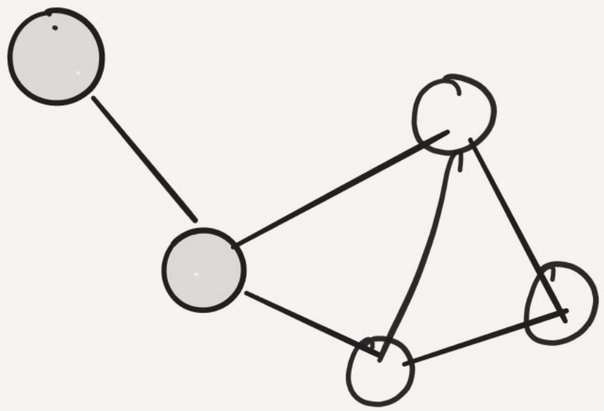
OUTPUT : Are s, t CONNECTED ? IF Yes give me a Path.

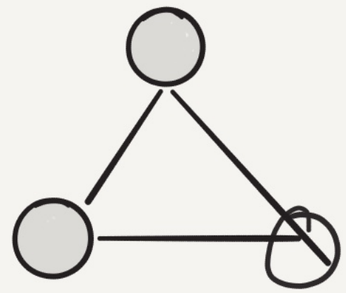
① Facebook "Are we friends (yet)?"

② Road networks and routing

③ Packet routing

Representation of Graphs



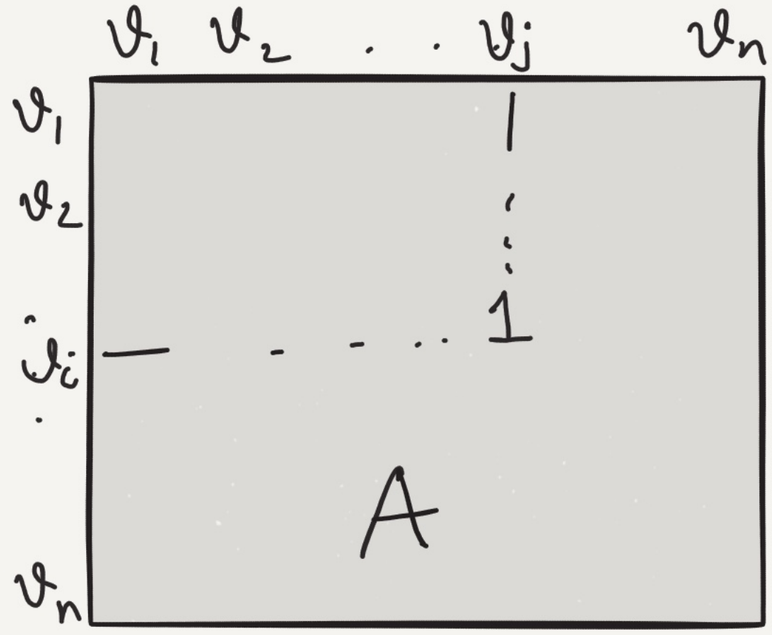


① Adjacency matrix Representation:

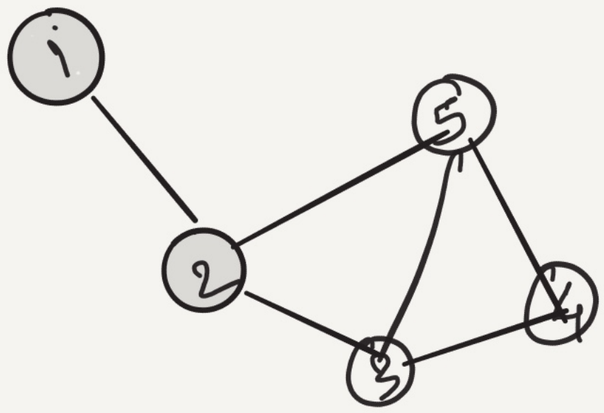
$G = (V, E)$ ⟶

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

$$A_{ij} = \begin{cases} 1 & \text{if } \{v_i, v_j\} \in E \\ 0 & \text{otherwise} \end{cases}$$



Representation of Graphs



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

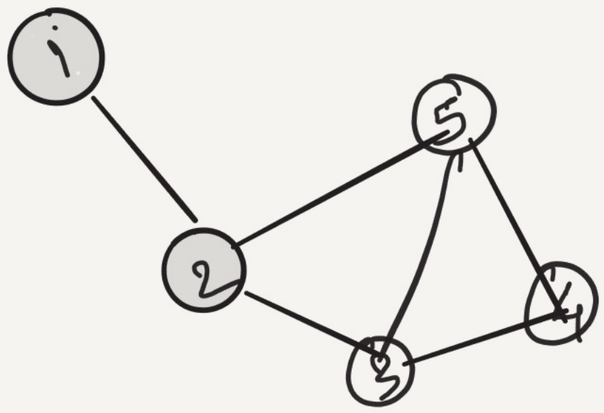
(1) Adjacency matrix Representation:

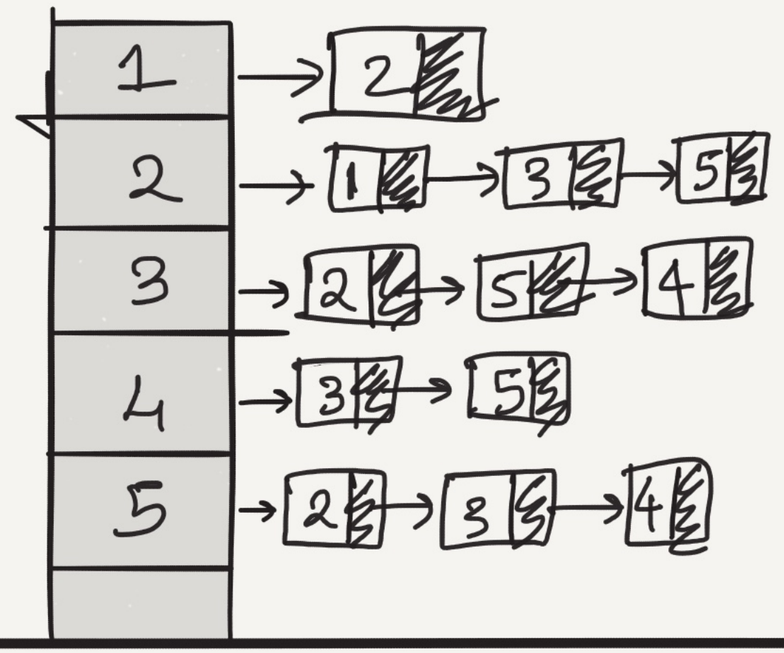
PROS	CONS
Can check presence of edges very fast → O(1).	Space complexity is n^2.

Ex: Facebook graph: 10^9 vertices; 10^{12} edges total!!

Adjacency representation: 10^{18} bits of memory

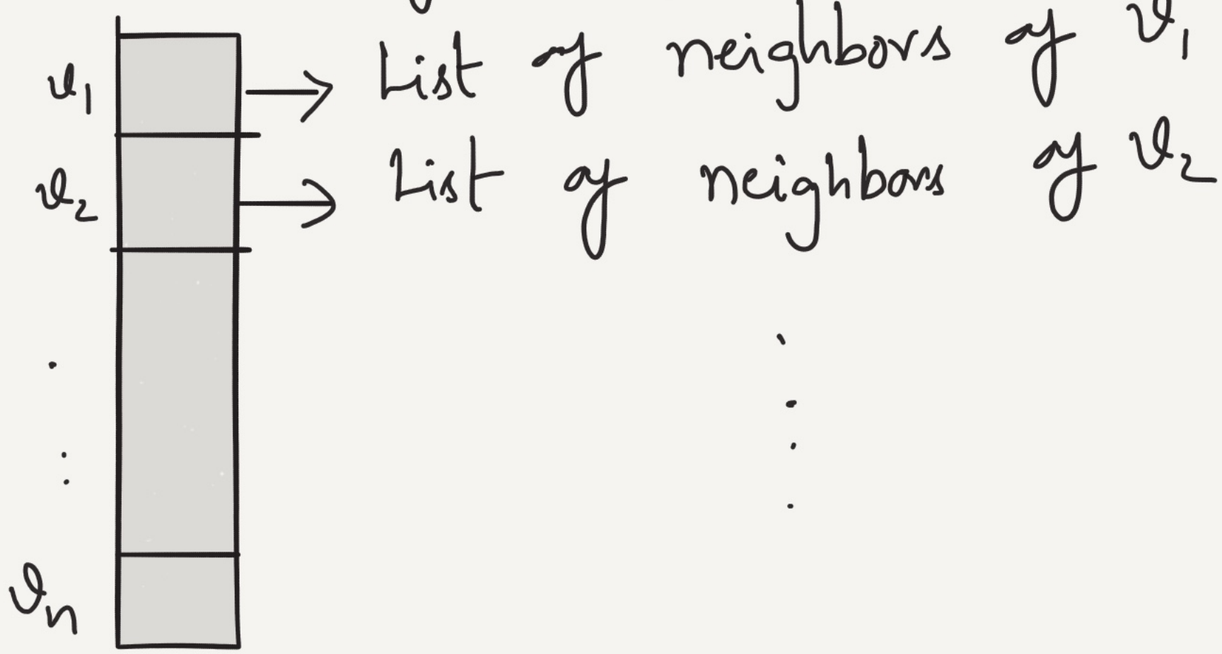
Representation of Graphs



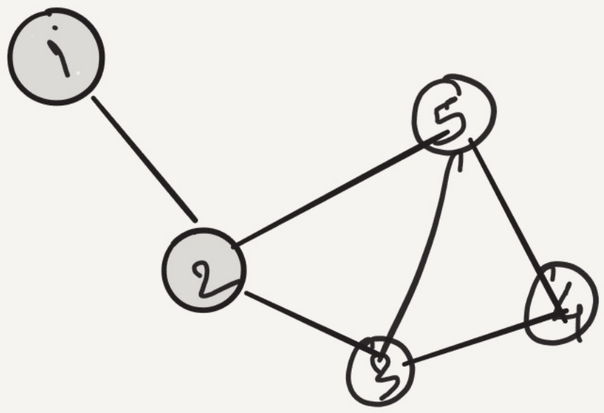


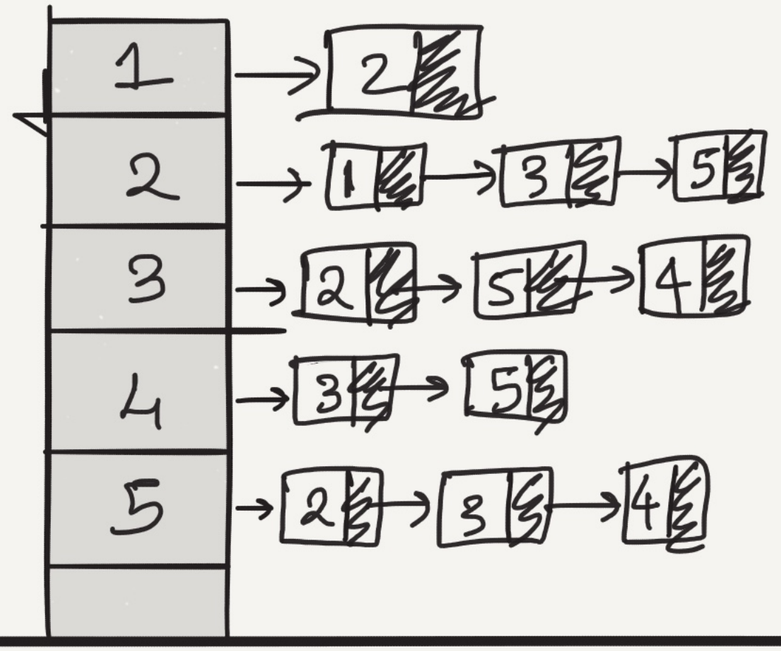
Adjacency List Representation:

$G = (V, E)$ → An array of "Lists".



Representation of Graphs





Adjacency List Representation:

PROS

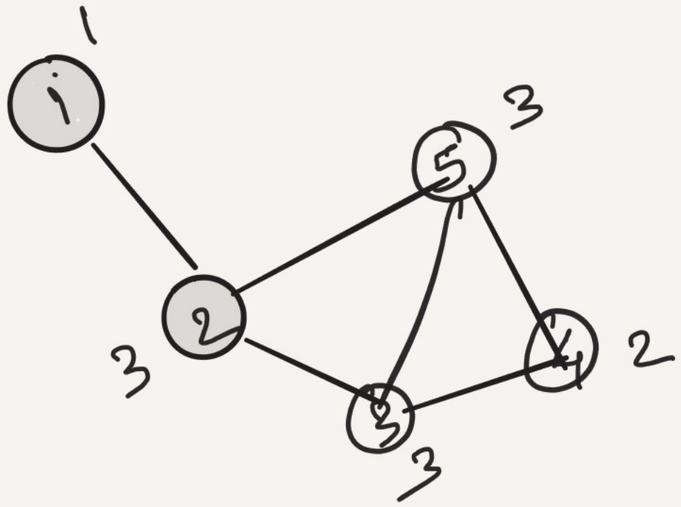
Space efficient:

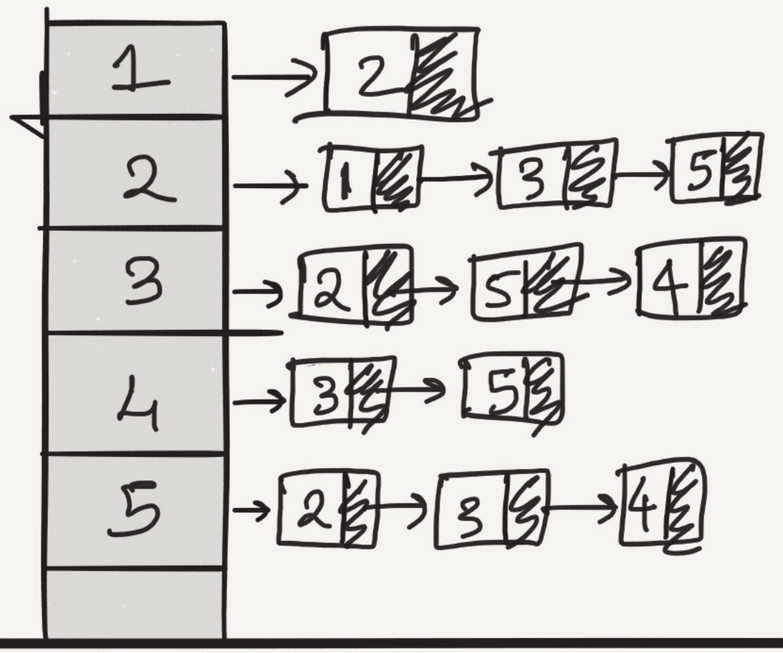
Size of representation $= O(\#\text{edges in } G)$.

CONS

To check if there is an edge between (u,v), we need to go through the entire list of u.

Representation of Graphs





Size of the list Corresponding to a vertex u

= the number of neighbors of u.

= degree(u)

$\Rightarrow$ Size of the representation = $\sum_{u \in V}$ degree(u).

= $2.|E|$.

Lemma: For any graph $G = (V, E)$

$$\sum_{u \in V} \text{degrees}(u) = 2 \cdot \underbrace{|E|}_{\#\text{ of edges}}$$

Proof: Each edge (u, v) "contributes two" to the sum on the left.

$\Rightarrow$ the lemma.

As a Consequence: facebook graph: 10^9 vertices, $\approx 10^{12}$ edges

Adjacency matrix requires $O(10^{18})$ bits of memory.

Adjacency list requires $O(10^{12})$

GRAPH CONNECTIVITY (s-t CONNECTIVITY PROBLEM)

INPUT : G is given as adjacency list; s, t

OUTPUT : Are s, t CONNECTED ? IF Yes give me a Path.

① Facebook "Are we friends (yet)?"

② Road networks and routing

③ Packet routing

How to Solve? Idea: Explore the graph.

→ If t is a neighbor of s.

Does $t \in A(s)$?

$A(s)$ → adjacency list of s.

If Yes, DONE!

→ If not check if t is a neighbor of a vertex in $A(s)$.

"Combine the lists of all my neighbors" & then check if t belongs to this new list.

How to Solve? Idea: Explore the graph.

→ If t is a neighbor of s.

Does $t \in A(s)$?

$A(s)$ → adjacency list of s.

If Yes, DONE!

→ If not check if t is a neighbor of a vertex in $A(s)$.

Form a list of all "new vertices" in the lists of my neighbors.

Breadth-First Search Algorithm

High-level idea:

L_0 : [s]

L_1 : Neighbors of s.

L_2 : Neighbors of vertices in L_1, but are not in L_1.

L_3 : " " " L_2, but are not in L_1, L_2.

.

.

.

L_j :

L_{j+1} : Neighbors of vertices in L_j, but are not in $L_1, L_2, \ldots, L_j$.

Implementation: **Breadth-First Search (BFS)**

DISCOVERED[u] = false for $u \neq s$

DISCOVERED[s] = true

$L[0] \leftarrow s$; $i \leftarrow 0$

WHILE L[i] IS NOT EMPTY

$L[i+1] \leftarrow \phi$

For each vertex $u \in L[i]$

For each neighbor v of u $(v \in A[u])$

If DISCOVERED[v] = true, then

do Nothing.

Else,

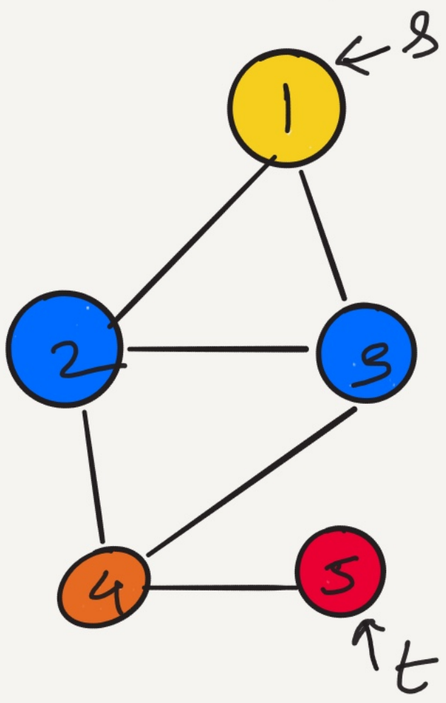
Set DISCOVERED[v] $\leftarrow$ true,

Add $v \leftarrow L[i+1]$.

$i \leftarrow i+1$.

→ CHECK IF DISCOVERED[t] = true.

Example:

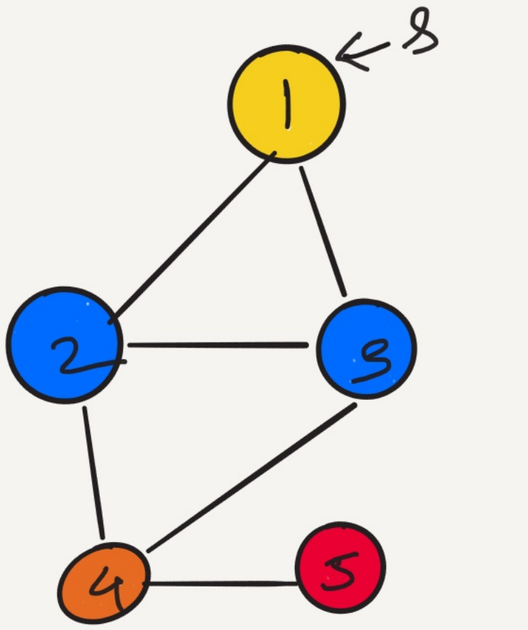


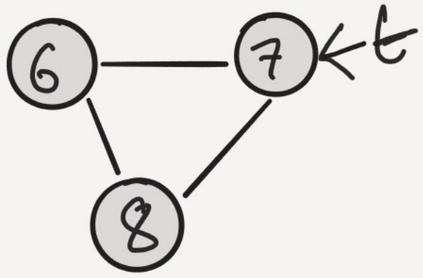
DISCOVERED

	1	2	3	4	5
$L_0 = [s]$	t	f	f	f	f
$L_1 = \{2, 3\}$	t	t	t	f	f
$L_2 = \{4\}$	t	t	t	t	f
$L_3 = \{5\}$	t	t	t	t	t
$L_4 = \emptyset$					

If DISCOVERED[5] is true?? YES, so (1,5) are Connected.

Example:





DISCOVERED

	1	2	3	4	5	6	7	8
$L_0 = [s]$	t	f	f	f	f	f	f	f
$L_1 = \{2, 3\}$	t	t	t	f	f	f	f	f
$L_2 = \{4\}$	t	t	t	t	f	f	f	f
$L_3 = \{5\}$	t	t	t	t	t	f	f	f
$L_4 = \emptyset$								

If DISCOVERED[7] is true?? NO! So (1, 7) are not connected.

Properties of BFS

1. The algorithm is correct!
2. Time Complexity: $O(\underbrace{n+|E|}_{\text{("size" of the input)}})$
3. The set of all vertices that are marked Discovered is precisely the set of all vertices that are connected to s.