

COL 351 : ANALYSIS & DESIGN OF ALGORITHMS

LECTURE 8

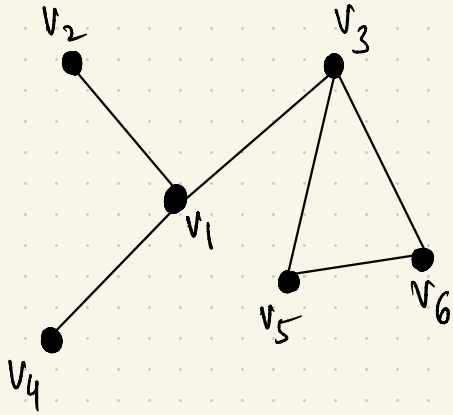
GRAPH ALGORITHMS I : BASIC DEFINITIONS AND REPRESENTATION

AUG 11, 2026

|

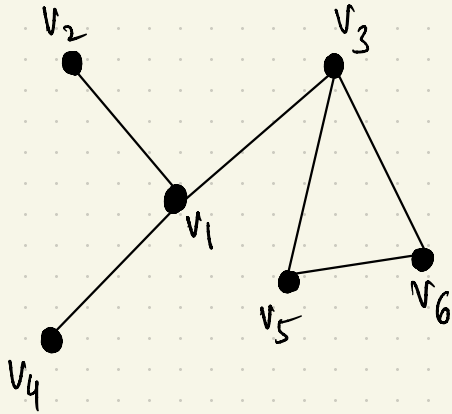
ROHIT VAISH

GRAPH

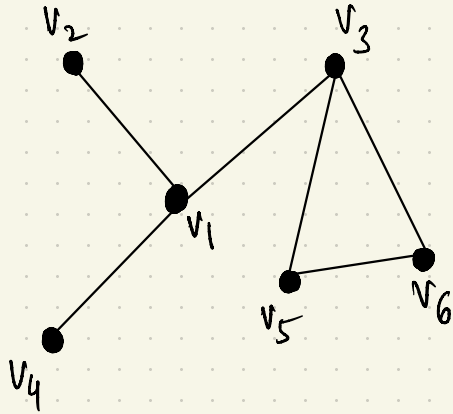


GRAPH

A graph G is a pair of sets (V, E)



GRAPH

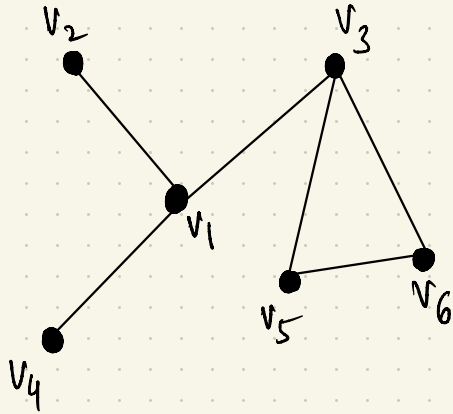


A graph G is a pair of sets (V, E)

V : non-empty set of items called
vertices / nodes

E : a (possibly empty) set of 2-item
subsets of V called edges

GRAPH



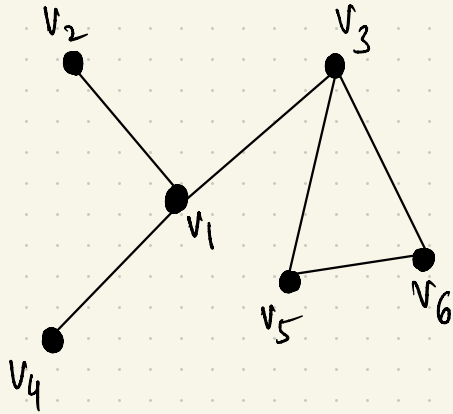
A graph G is a pair of sets (V, E)

V : non-empty set of items called
vertices / nodes

e.g., $V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$

E : a (possibly empty) set of 2-item
subsets of V called edges

GRAPH



A graph G is a pair of sets (V, E)

V : non-empty set of items called
vertices / **nodes**

e.g., $V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$

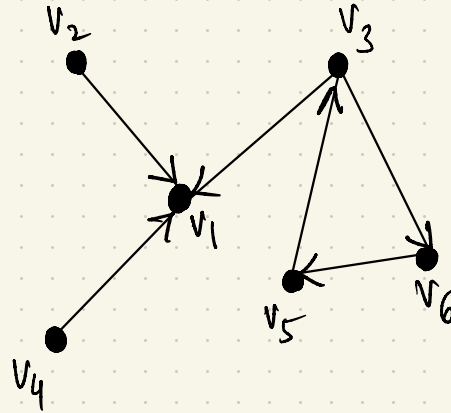
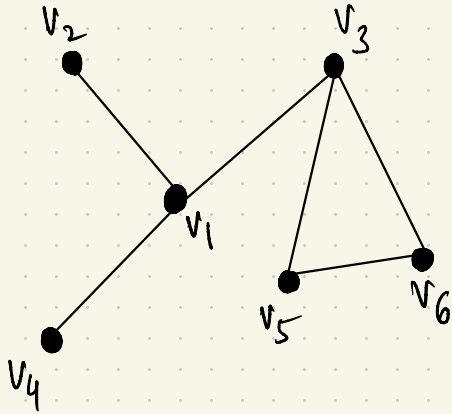
E : a (possibly empty) set of 2-item
subsets of V called **edges**

e.g., $E = \left\{ \{v_1, v_4\}, \{v_1, v_2\}, \{v_1, v_3\}, \right.$
 $\left. \{v_3, v_5\}, \{v_5, v_6\}, \{v_3, v_6\} \right\}$

GRAPH

Edges can be undirected (unordered pair)

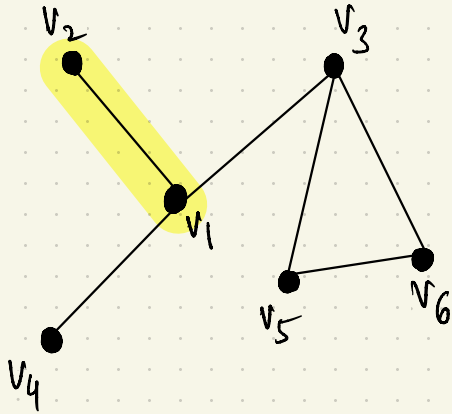
or directed (ordered pair).



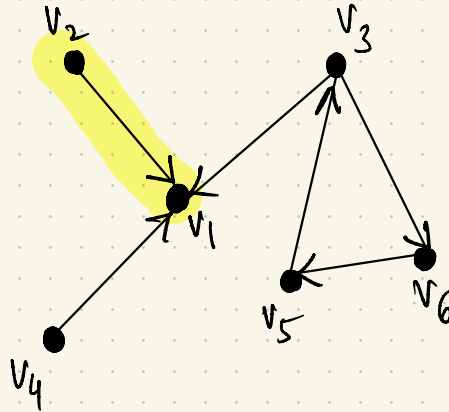
GRAPH

Edges can be undirected (unordered pair)

or directed (ordered pair).



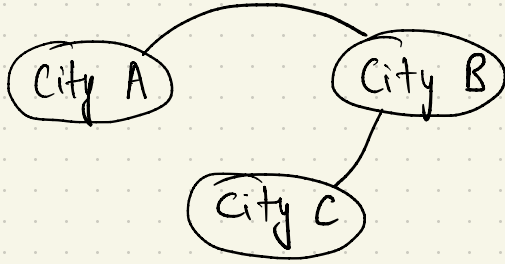
edge $\{v_1, v_2\}$ or $\{v_2, v_1\}$



edge (v_2, v_1)

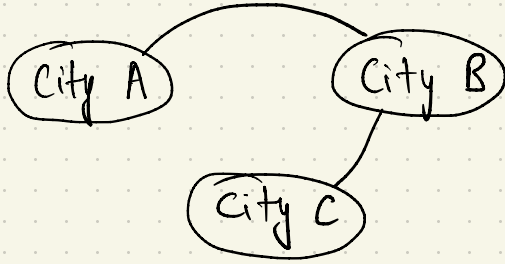
APPLICATIONS

APPLICATIONS

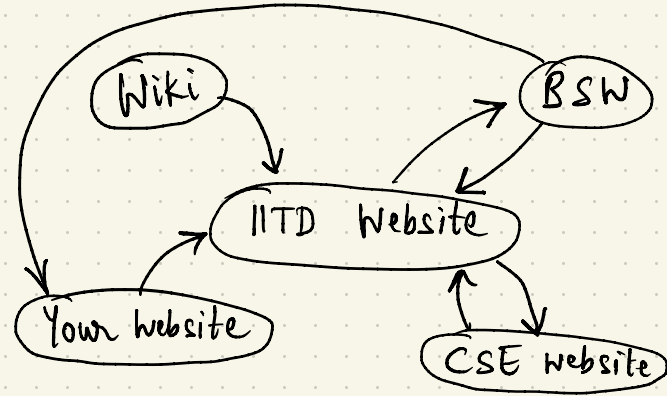


Road networks

APPLICATIONS

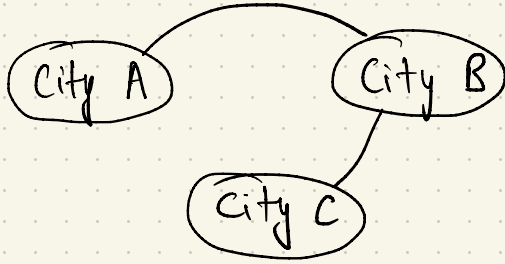


Road networks

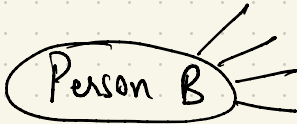
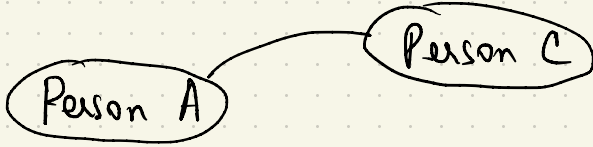


The web

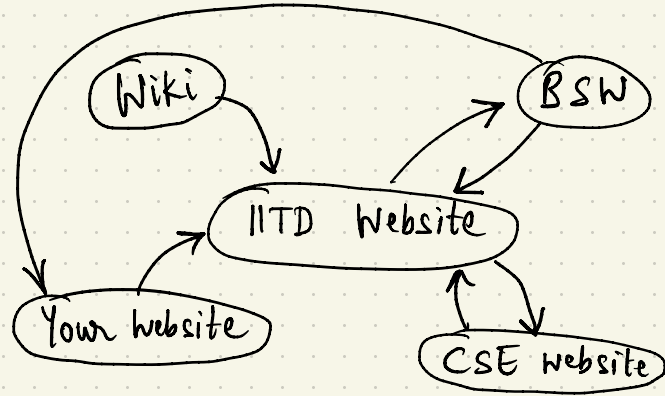
APPLICATIONS



Road networks

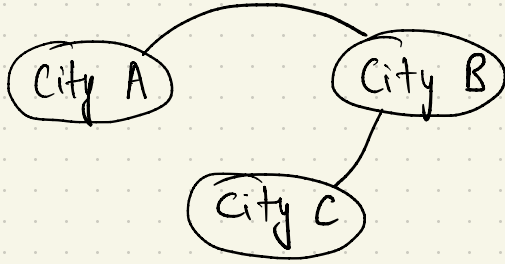


Social networks

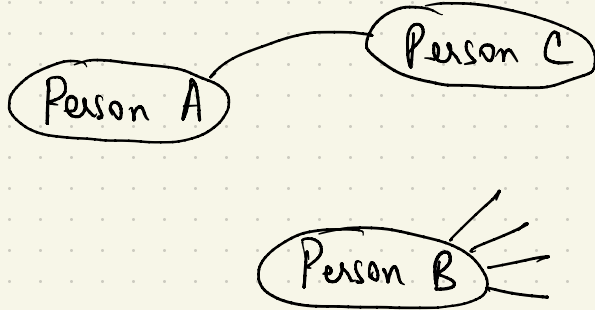


The web

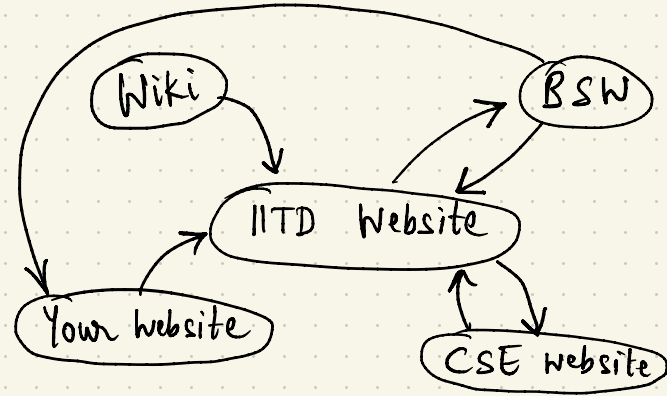
APPLICATIONS



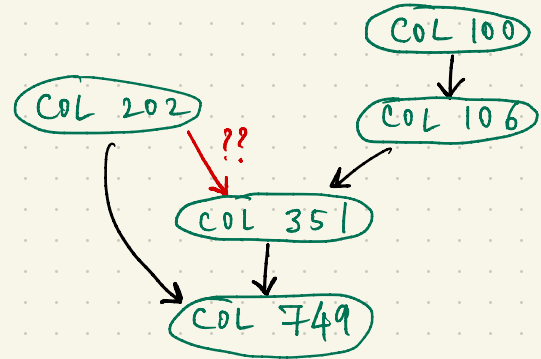
Road networks



Social networks

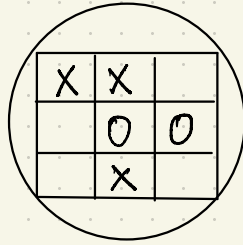


The web

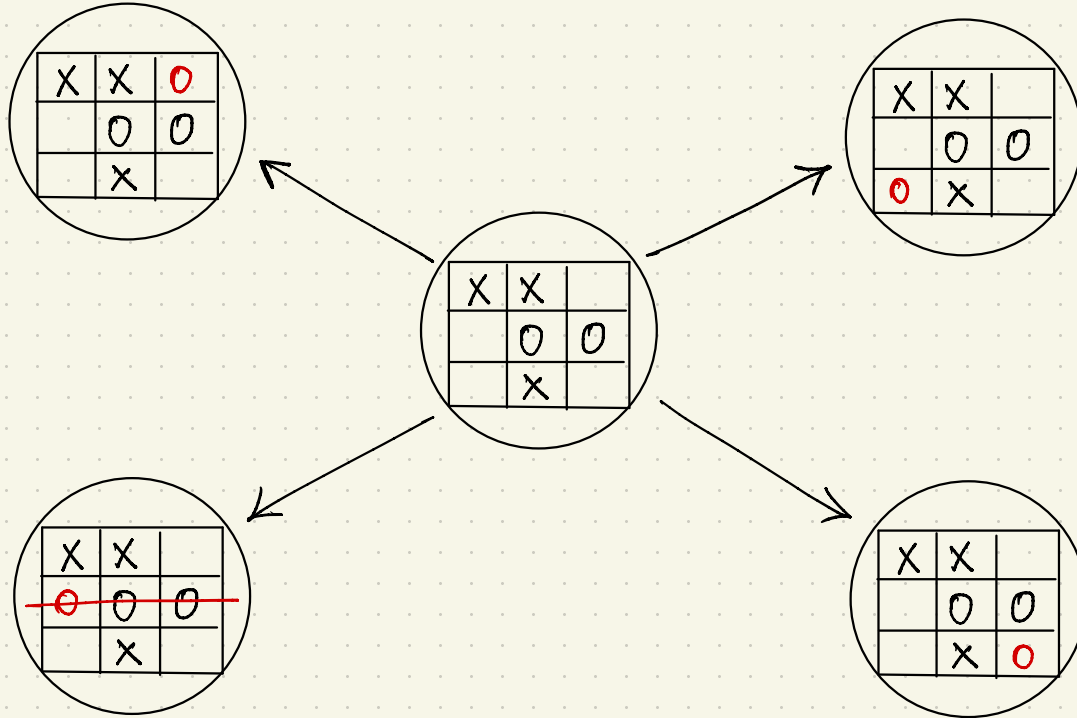


Precedence constraints

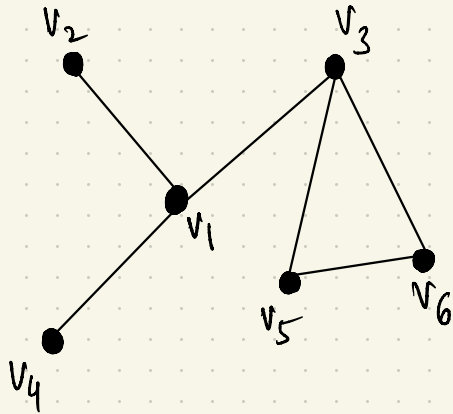
APPLICATIONS



APPLICATIONS

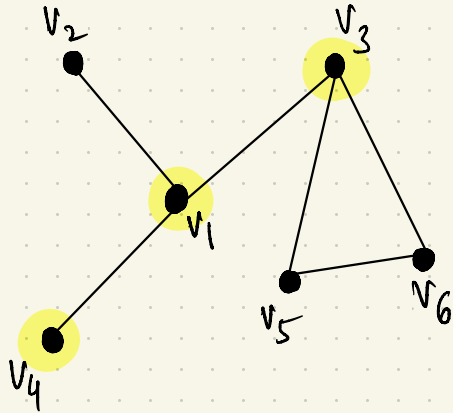


GRAPH



Two vertices v_i and v_j are adjacent if $\{v_i, v_j\} \in E$.

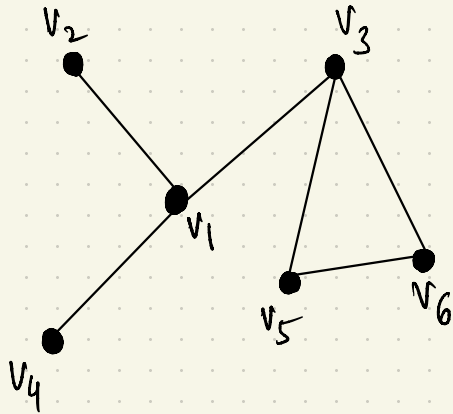
GRAPH



Two vertices v_i and v_j are adjacent
if $\{v_i, v_j\} \in E$.

e.g., v_1 and v_3 are adjacent
 v_3 and v_4 are NOT adjacent

GRAPH

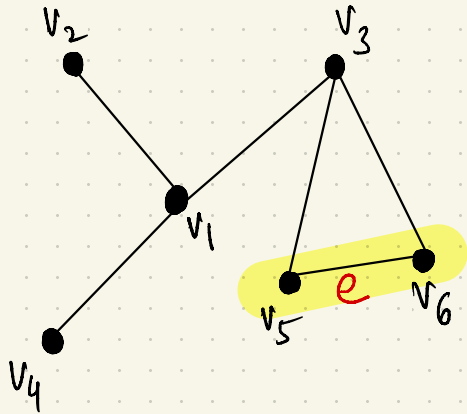


Two vertices v_i and v_j are adjacent if $\{v_i, v_j\} \in E$.

e.g., v_1 and v_3 are adjacent
 v_2 and v_4 are NOT adjacent

An edge $e = \{v_i, v_j\}$ is incident to the vertices v_i and v_j .

GRAPH



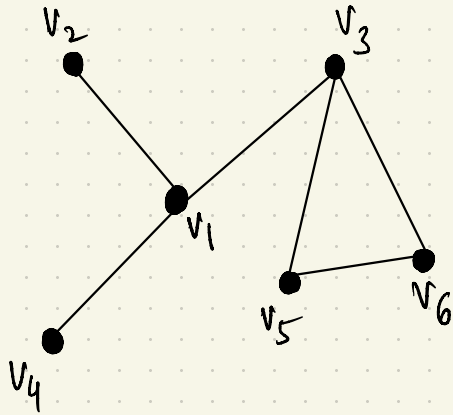
Two vertices v_i and v_j are adjacent if $\{v_i, v_j\} \in E$.

e.g., v_1 and v_3 are adjacent
 v_3 and v_4 are NOT adjacent

An edge $e = \{v_i, v_j\}$ is incident to the vertices v_i and v_j .

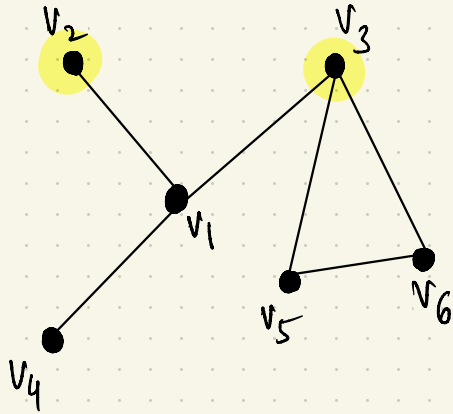
e.g., e is incident to v_5 and v_6
but not v_3 .

GRAPH



The number of edges incident to a vertex is called the **degree**.

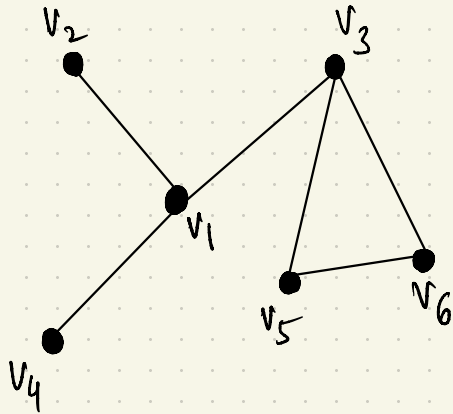
GRAPH



The number of edges incident to a vertex is called the **degree**.

e.g., $\deg(v_3) = 3$ $\deg(v_2) = 1$

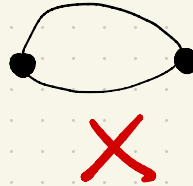
GRAPH



The number of edges incident to a vertex is called the **degree**.

e.g., $\deg(v_3) = 3$ $\deg(v_2) = 1$

A **simple** graph has no self-loops or multiedges.



WALKS AND PATHS

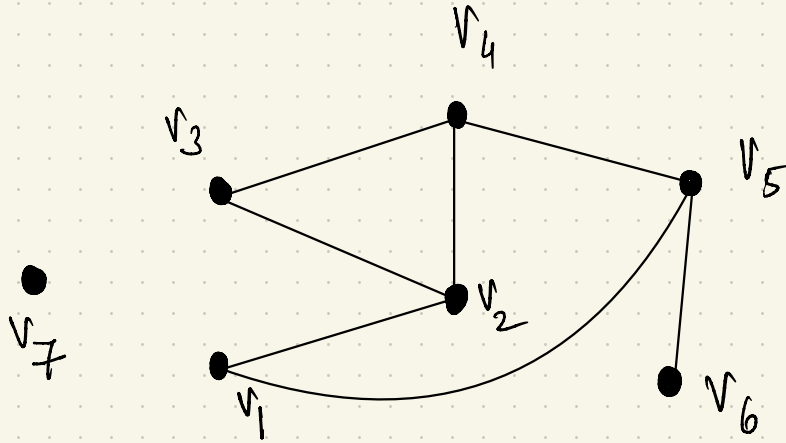
WALKS AND PATHS

A **walk** is a sequence of vertices connected by edges.

WALKS AND PATHS

A **walk** is a sequence of vertices connected by edges.

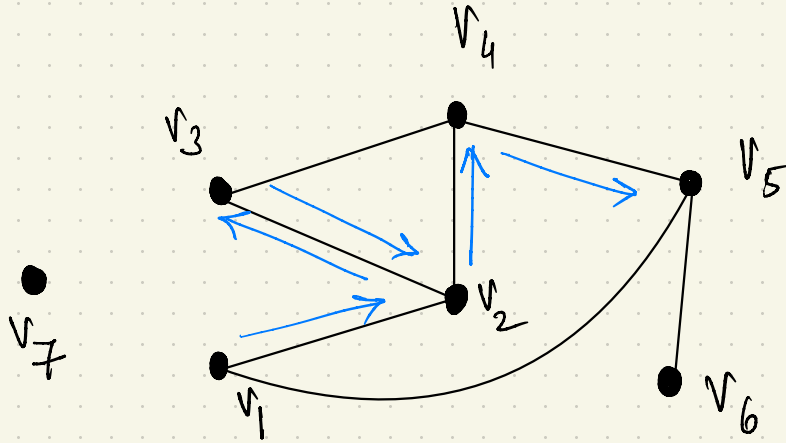
E.g.,



WALKS AND PATHS

A **walk** is a sequence of vertices connected by edges.

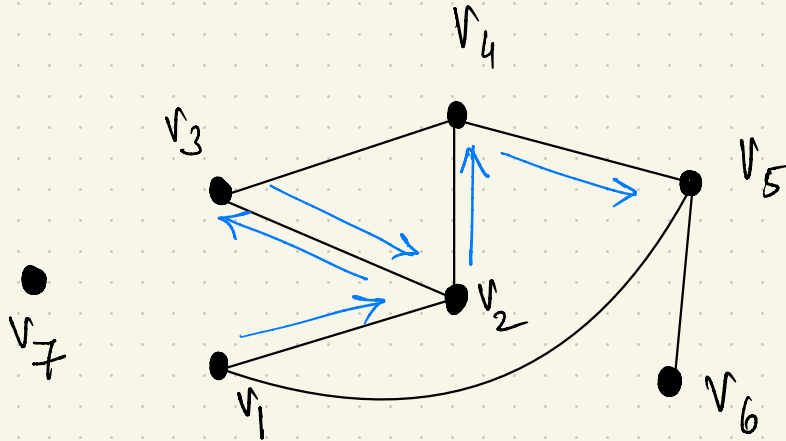
E.g., $v_1 - v_2 - v_3 - v_2 - v_4 - v_5$
start end



WALKS AND PATHS

A **walk** is a sequence of vertices connected by edges.

E.g., $v_1 \text{ --- } v_2 \text{ --- } v_3 \text{ --- } v_2 \text{ --- } v_4 \text{ --- } v_5$ } walk of length 5
start end



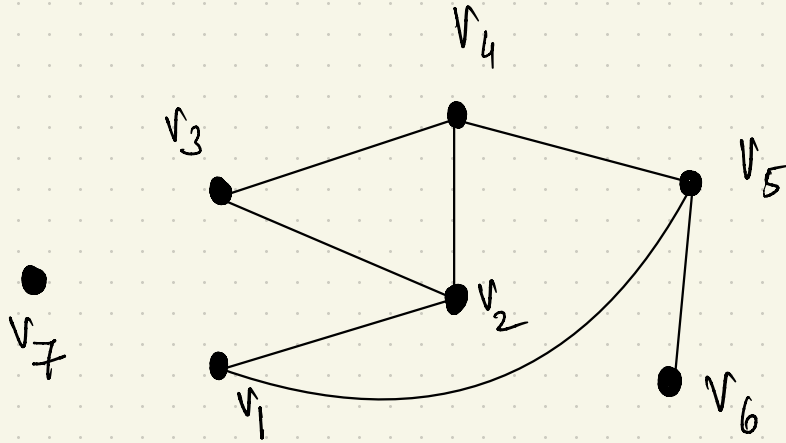
WALKS AND PATHS

A **path** is a walk where all vertices are different.

WALKS AND PATHS

A **path** is a walk where all vertices are different.

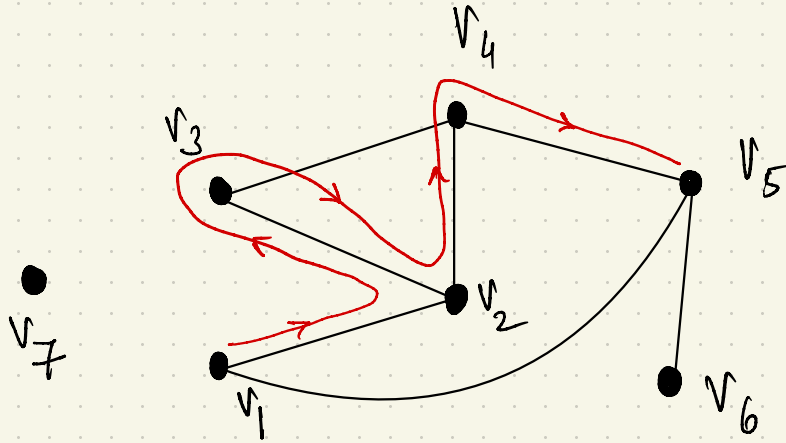
E.g.,



WALKS AND PATHS

A **path** is a walk where all vertices are different.

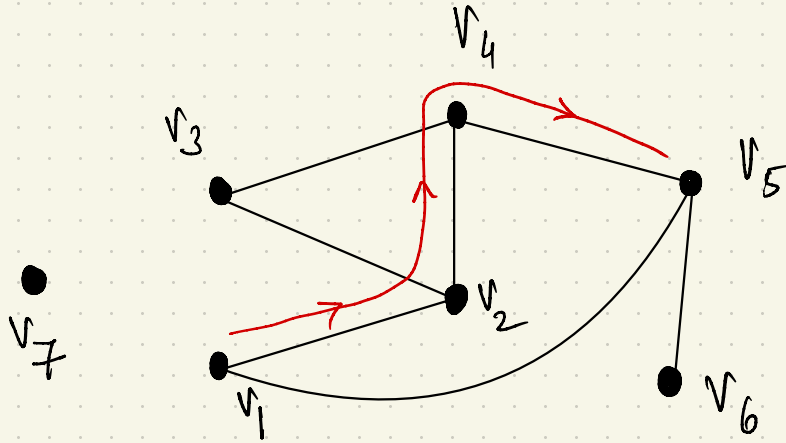
E.g., $v_1 - v_2 - v_3 - v_2 - v_4 - v_5$ X Not a path



WALKS AND PATHS

A **path** is a walk where all vertices are different.

E.g., $v_1 - v_2 - v_4 - v_5$ A valid path (length = 3)



Lemma: For any two distinct vertices u and v ,

there exists a **walk** between u and v $\iff$ there exists a **path** between u and v

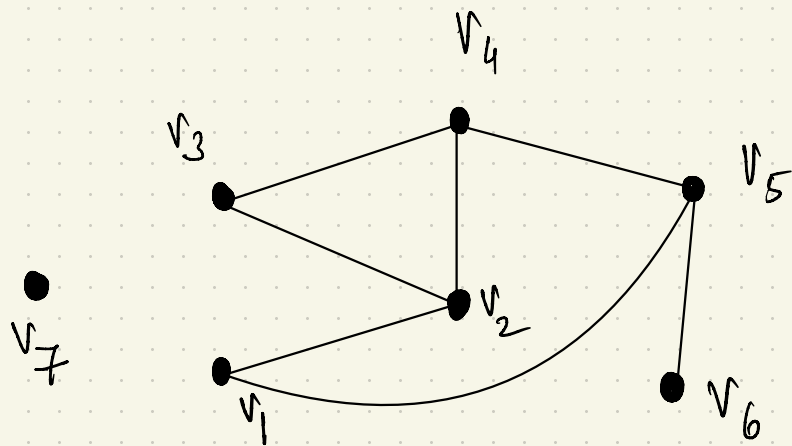
Proof: Exercise

CONNECTIVITY

A pair of vertices u and v are **connected** if there is a path between u and v .

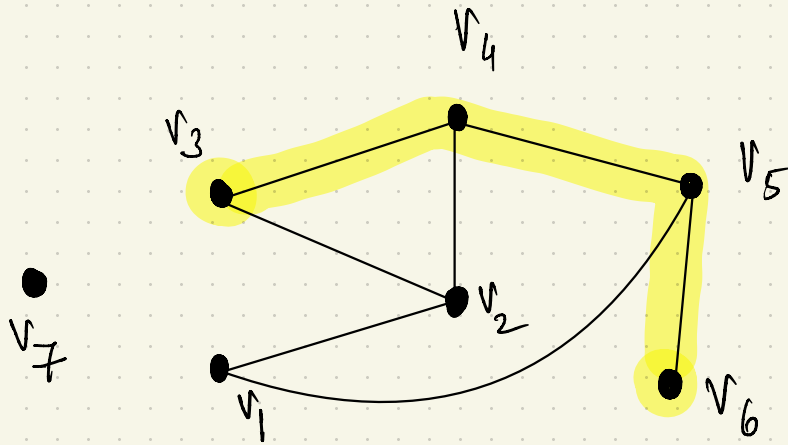
CONNECTIVITY

A pair of vertices u and v are **connected** if there is a path between u and v .



CONNECTIVITY

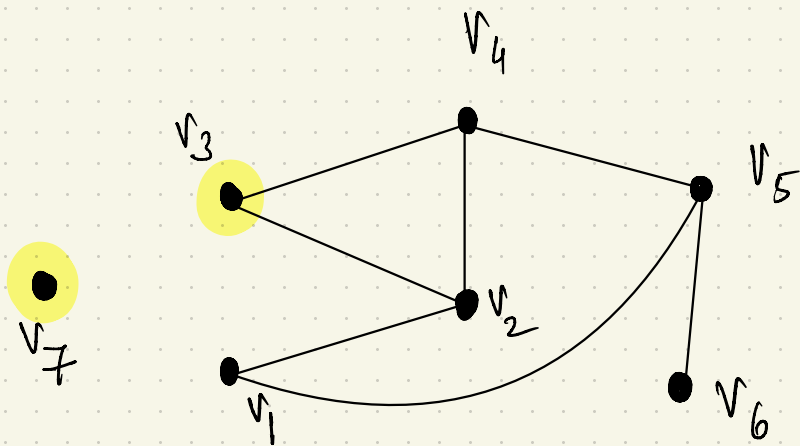
A pair of vertices u and v are **connected** if there is a path between u and v .



$v_3, v_6 \rightarrow$ Connected

CONNECTIVITY

A pair of vertices u and v are **connected** if there is a path between u and v .

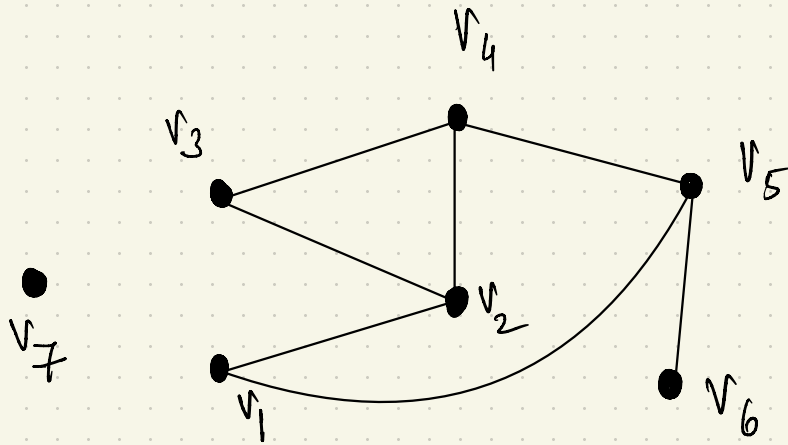


$v_3, v_6 \rightarrow$ Connected

$v_3, v_7 \rightarrow$ Not connected

CONNECTIVITY

A pair of vertices u and v are **connected** if there is a path between u and v .



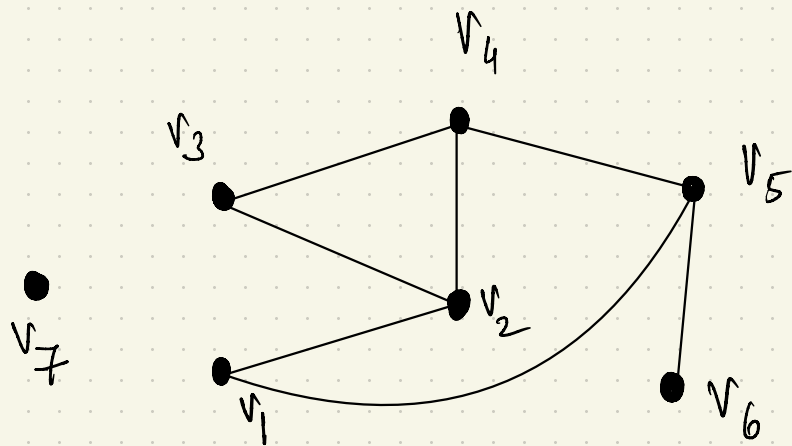
$v_3, v_6 \rightarrow$ Connected

$v_3, v_7 \rightarrow$ Not Connected

A graph is **connected** if every pair of vertices are connected.

CONNECTIVITY

A pair of vertices u and v are **connected** if there is a path between u and v .

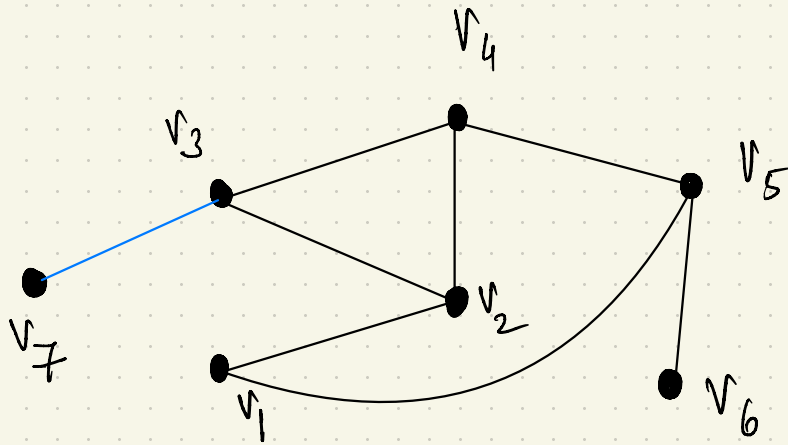


Not a connected graph

A graph is **connected** if every pair of vertices are connected.

CONNECTIVITY

A pair of vertices u and v are **connected** if there is a path between u and v .



Connected

A graph is **connected** if every pair of vertices are connected.

CONNECTIVITY

Given an undirected and connected graph on n vertices

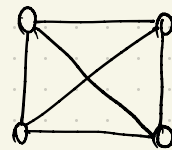
Maximum # edges = ?

Minimum # edges = ?

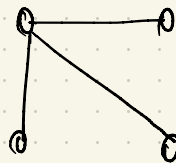
CONNECTIVITY

Given an undirected and connected graph on n vertices

Maximum # edges = $n C_2$



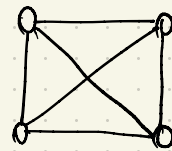
Minimum # edges = $n - 1$



CONNECTIVITY

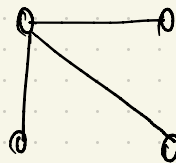
Given an undirected and connected graph on n vertices

Maximum # edges = nC_2



complete graph

Minimum # edges = $n-1$



tree

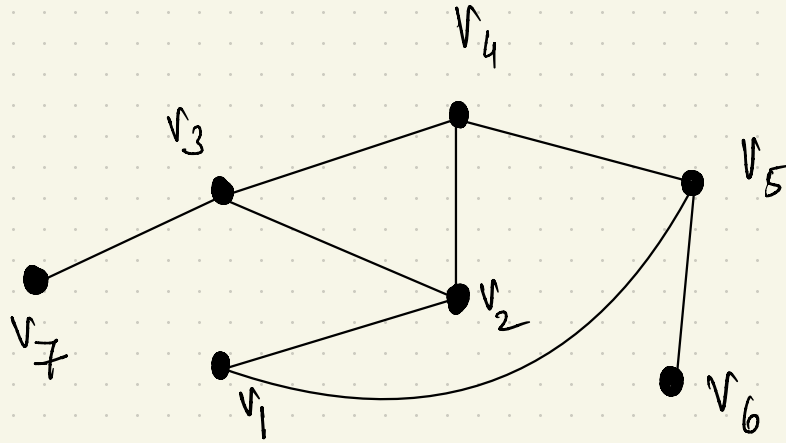
CYCLES AND CLOSED WALKS

CYCLES AND CLOSED WALKS

A closed walk is a walk that starts and ends at the same vertex.

CYCLES AND CLOSED WALKS

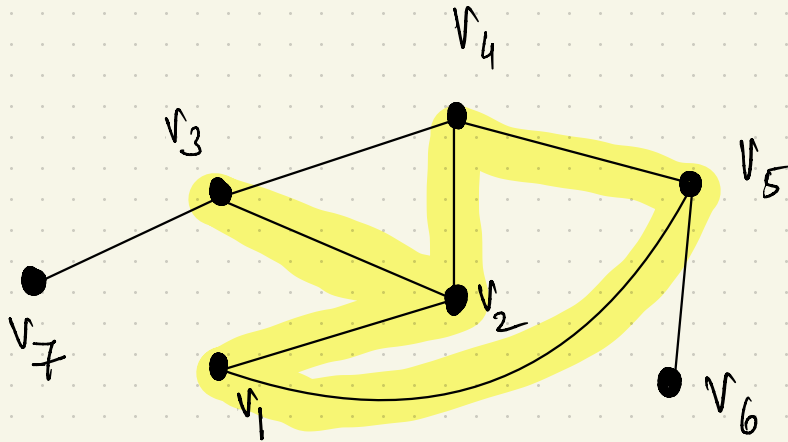
A **closed walk** is a walk that starts and ends at the same vertex.



CYCLES AND CLOSED WALKS

A **closed walk** is a walk that starts and ends at the same vertex.

$$v_1 - v_2 - v_3 - v_2 - v_4 - v_5 - v_1$$



CYCLES AND CLOSED WALKS

A closed walk is a walk that starts and ends at the same vertex.

A cycle is a closed walk in which all vertices (except start and end) are different.

CYCLES AND CLOSED WALKS

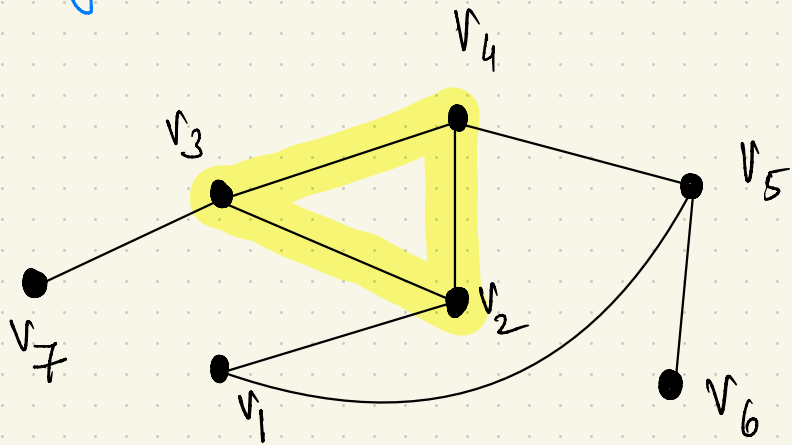
A closed walk is a walk that starts and ends at the same vertex.

A cycle is a closed walk in which all vertices (except start and end) are different.  disallow single vertices or single edges

CYCLES AND CLOSED WALKS

A **closed walk** is a walk that starts and ends at the same vertex.

A **cycle** is a closed walk in which all vertices (except start and end) are different. ↙ disallow single vertices or single edges

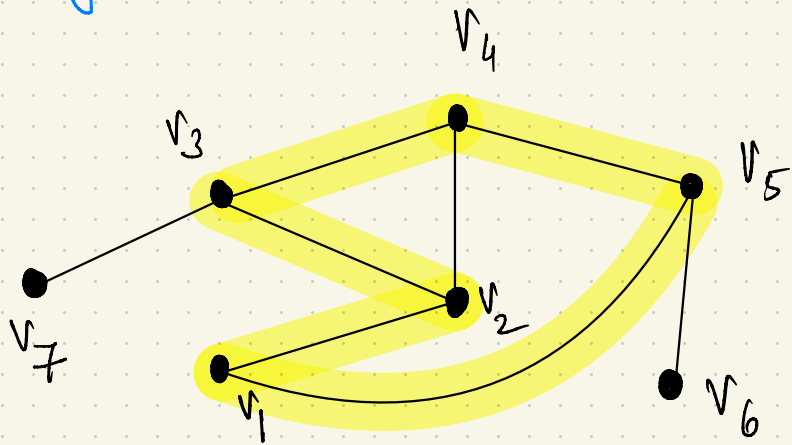


$v_2 - v_3 - v_4 - v_2$ is a cycle of length three

CYCLES AND CLOSED WALKS

A **closed walk** is a walk that starts and ends at the same vertex.

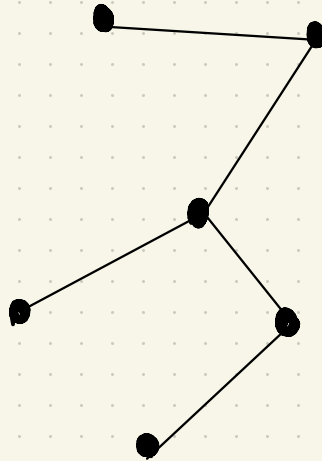
A **cycle** is a closed walk in which all vertices (except start and end) are different. ↙ disallow single vertices or single edges



$v_1 - v_2 - v_3 - v_4 - v_5 - v_1$ is a cycle of length five

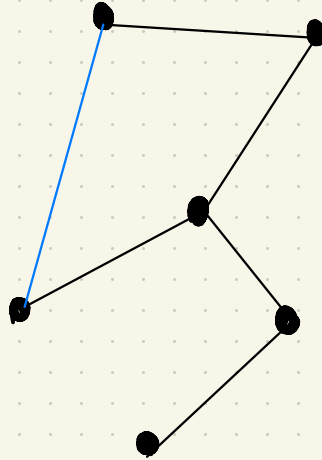
TREES

A tree is a connected and acyclic graph.



TREES

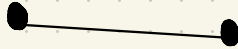
A tree is a connected and acyclic graph.



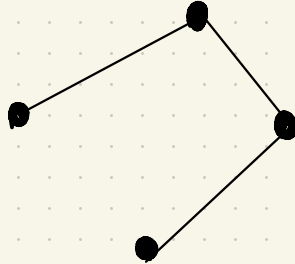
Not a tree

TREES

A tree is a connected and acyclic graph.



Not a tree



(It's a forest!)

SPARSE vs DENSE GRAPHS

SPARSE vs DENSE GRAPHS

n : number of vertices

m : number of edges

SPARSE vs DENSE GRAPHS

n : number of vertices

m : number of edges

Many applications : m is $\Omega(n)$ and $O(n^2)$.
(but not always)

SPARSE vs DENSE GRAPHS

n : number of vertices

m : number of edges

Many applications : m is $\Omega(n)$ and $O(n^2)$.
(but not always)

roughly,

sparse : $m = O(n \log n)$ close to linear

dense : $m = \Omega\left(\frac{n^2}{\log n}\right)$ close to quadratic

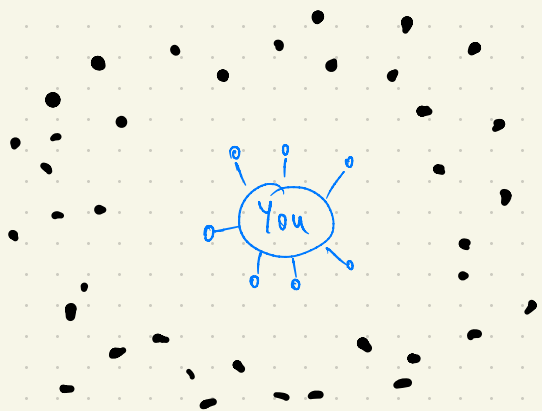
SPARSE vs DENSE GRAPHS

n : number of vertices

m : number of edges

Many applications : m is $\Omega(n)$ and $O(n^2)$.
(but not always)

Facebook graph is sparse...



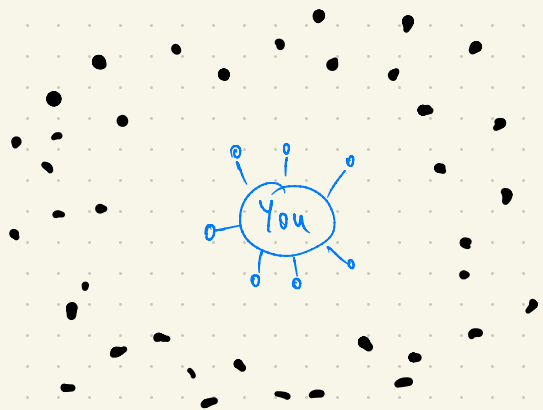
SPARSE vs DENSE GRAPHS

n : number of vertices

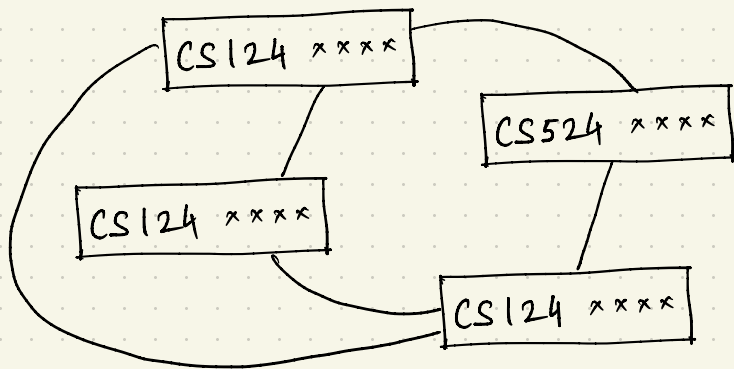
m : number of edges

Many applications : m is $\Omega(n)$ and $O(n^2)$.
(but not always)

Facebook graph is sparse ...



... but also dense



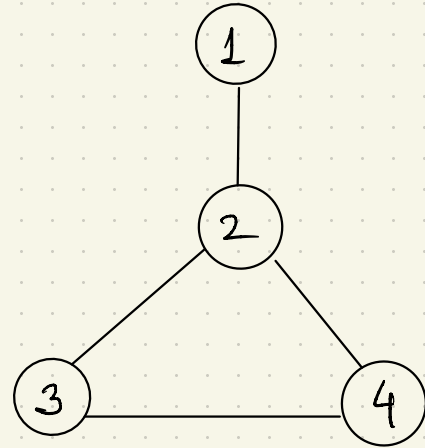
REPRESENTING GRAPHS

REPRESENTING GRAPHS

Adjacency matrix

REPRESENTING GRAPHS

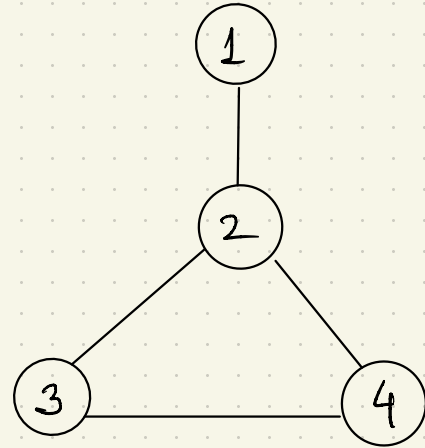
Adjacency matrix



REPRESENTING GRAPHS

Adjacency matrix

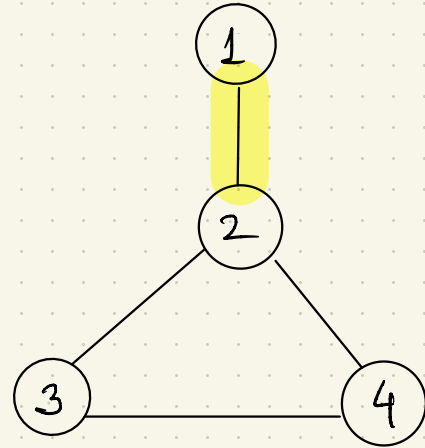
$$A = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{bmatrix} & & & \\ & & & \\ & & & \\ & & & \end{bmatrix} \end{matrix}$$



REPRESENTING GRAPHS

Adjacency matrix

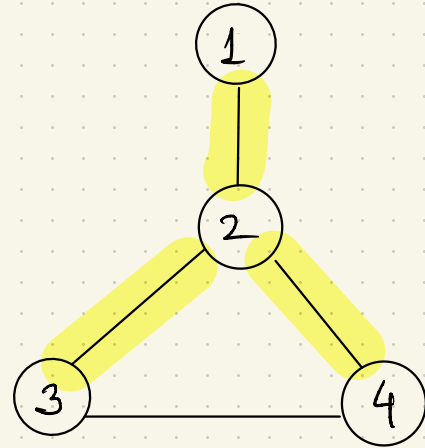
$$A = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{matrix}$$



REPRESENTING GRAPHS

Adjacency matrix

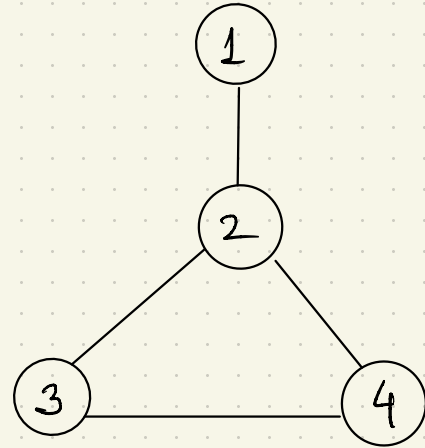
$$A = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \end{matrix}$$



REPRESENTING GRAPHS

Adjacency matrix

$$A = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \end{matrix}$$

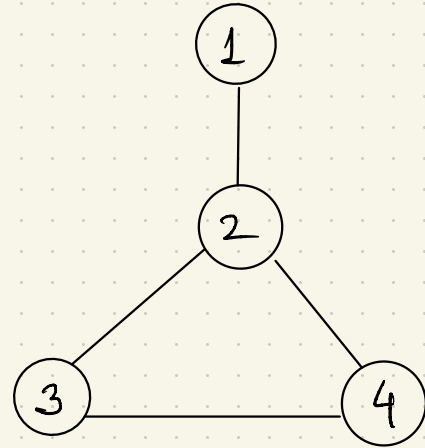


REPRESENTING GRAPHS

Adjacency matrix

$$A = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \end{matrix}$$

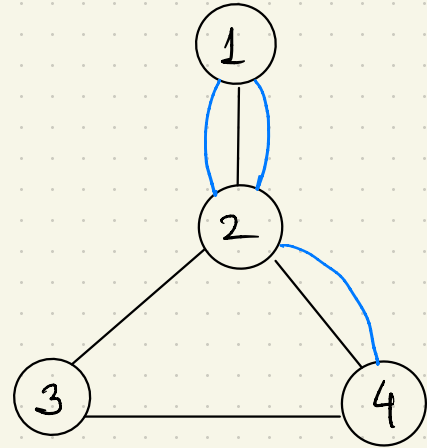
↘ Symmetric



REPRESENTING GRAPHS

Adjacency matrix

$$A = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{bmatrix} 0 & 3 & 0 & 0 \\ 3 & 0 & 1 & 2 \\ 0 & 1 & 0 & 1 \\ 0 & 2 & 1 & 0 \end{bmatrix} \end{matrix}$$

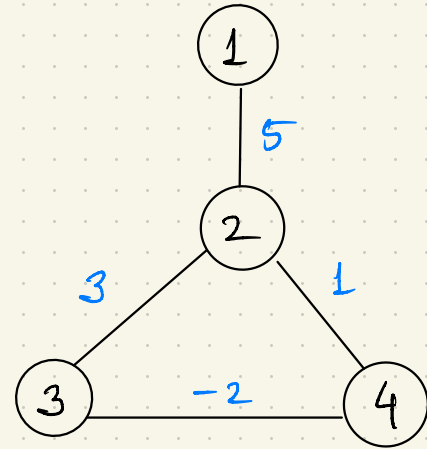


parallel edges

REPRESENTING GRAPHS

Adjacency matrix

$$A = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{bmatrix} 0 & 5 & 0 & 0 \\ 5 & 0 & 3 & 1 \\ 0 & 3 & 0 & -2 \\ 0 & 1 & -2 & 0 \end{bmatrix} \end{matrix}$$



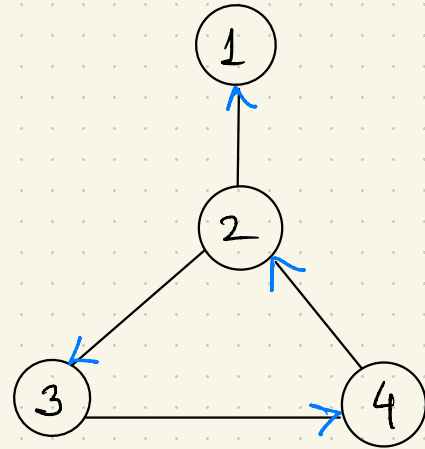
edge weights

REPRESENTING GRAPHS

Adjacency matrix

$$A = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{bmatrix} 0 & -1 & 0 & 0 \\ +1 & 0 & +1 & -1 \\ 0 & -1 & 0 & +1 \\ 0 & +1 & -1 & 0 \end{bmatrix} \end{matrix}$$

skew symmetric



Directed edges

REPRESENTING GRAPHS

Adjacency matrix

$$A = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{bmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{bmatrix} \end{matrix}$$

$\Theta(n^2)$ space

OK for dense graphs

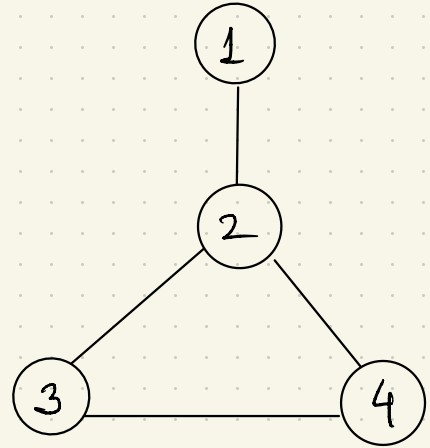
Not OK for sparse graphs

REPRESENTING GRAPHS

Adjacency list

REPRESENTING GRAPHS

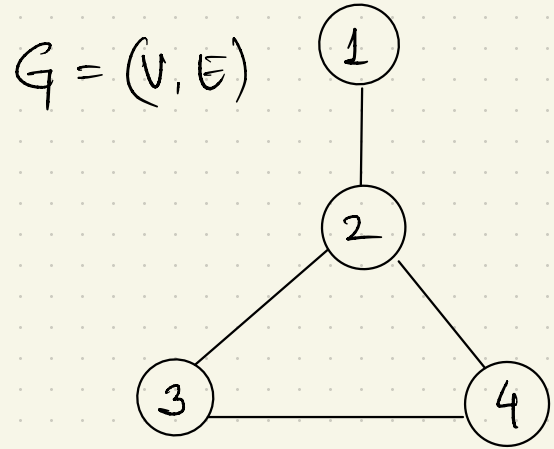
Adjacency list



REPRESENTING GRAPHS

Adjacency list

array of $|V|$ linked lists



REPRESENTING GRAPHS

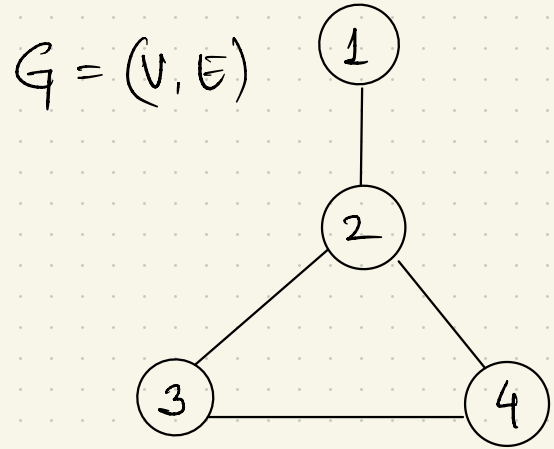
Adjacency list

array of $|V|$ linked lists

for each vertex $u \in V$,

$\text{Adj}[u]$ stores neighbors of u

$\{ v \in V : \{u, v\} \in E \}$.



REPRESENTING GRAPHS

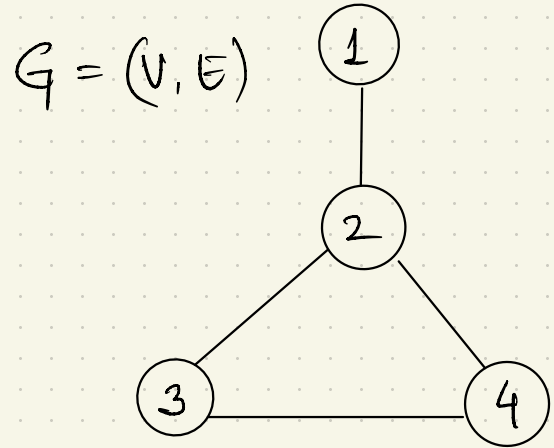
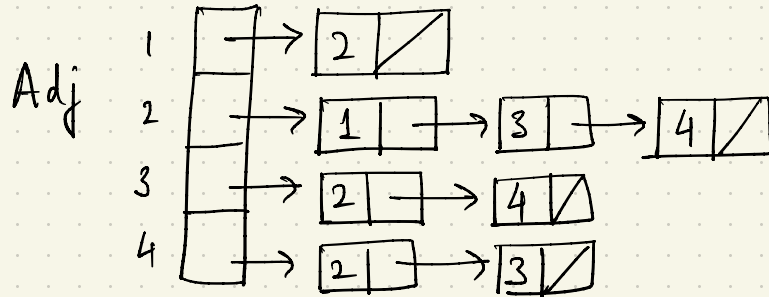
Adjacency list

array of $|V|$ linked lists

for each vertex $u \in V$,

$\text{Adj}[u]$ stores neighbors of u

$$\{ v \in V : \{u, v\} \in E \}.$$



REPRESENTING GRAPHS

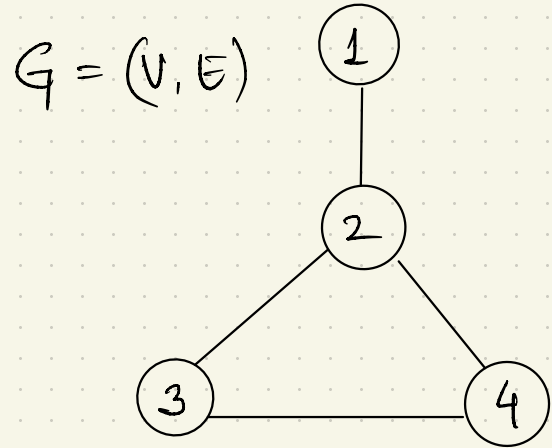
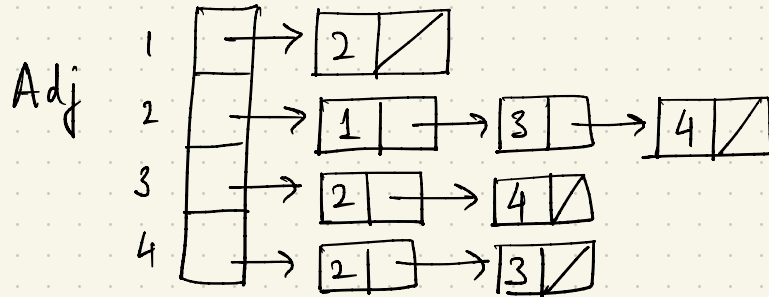
Adjacency list

array of $|V|$ linked lists

for each vertex $u \in V$,

$\text{Adj}[u]$ stores neighbors of u

$$\{ v \in V : \{u, v\} \in E \}.$$



Space : ?

REPRESENTING GRAPHS

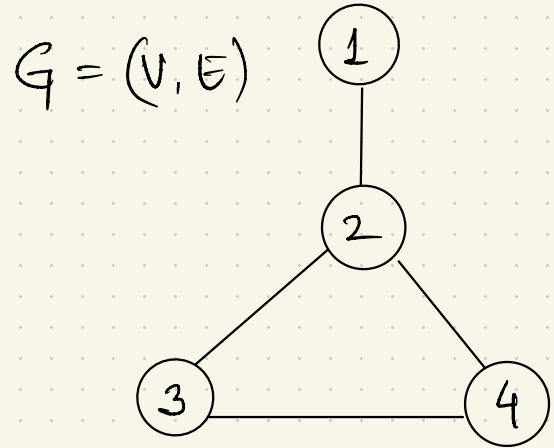
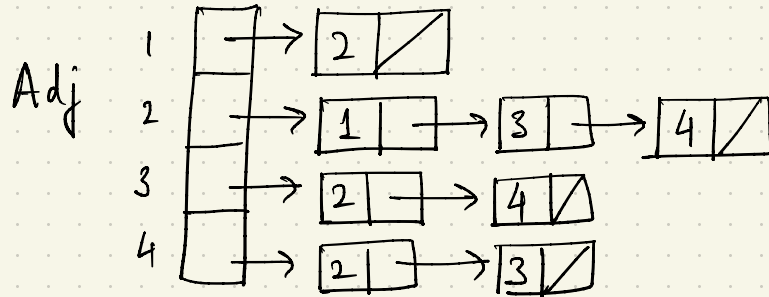
Adjacency list

array of $|V|$ linked lists

for each vertex $u \in V$,

$\text{Adj}[u]$ stores neighbors of u

$$\{ v \in V : \{u, v\} \in E \}.$$



Space : $\theta(|V| + |E|)$
or $\theta(m + n)$

REPRESENTING GRAPHS

Which is better : adjacency matrix or adjacency list ?