COL 351 : Analysis & Design of Algorithms

Lecture 8

Graph Algorithms I

Aug 07, 2024 | Rohit Vaish

REMINDERS

Sign up on Teams channel

Sign up on Gradescope

WHAT DOES POLYNOMIAL-TIME MEAN?

WHAT DOES POLYNOMIAL-TIME MEAN?

$$\text{input} \longrightarrow \boxed{\text{ALG}} \longrightarrow \text{output}$$

WHAT DOES POLYNOMIAL-TIME MEAN?

$$\text{input} \longrightarrow \boxed{\text{ALG}} \longrightarrow \text{output}$$

ALG is a polynomial-time algorithm if the worst-case running time of ALG is upper bounded by a polynomial function of the size of the input.

WHAT DOES POLYNOMIAL-TIME MEAN?

Problem	Input size	Poly-time?

WHAT DOES POLYNOMIAL-TIME MEAN?

Problem	Input size	Poly-time?
Given a number n, return its square.		

WHAT DOES POLYNOMIAL-TIME MEAN?

Problem	Input size	Poly-time?
Given a number n, return its square.	n	

WHAT DOES POLYNOMIAL-TIME MEAN?

Problem	Input size	Poly-time?
Given a number n, return its square.	$\cancel{n}$ $\log n$	

WHAT DOES POLYNOMIAL-TIME MEAN?

Problem	Input size	Poly-time?
Given a number n, return its square.	$\cancel{n}$ $\log n$	$O(n)$

WHAT DOES POLYNOMIAL-TIME MEAN?

Problem	Input size	Poly-time?
Given a number n, return its square.	$\cancel{n}$ $\log n$	$O(n)$ ✗

WHAT DOES POLYNOMIAL-TIME MEAN?

Problem	Input size	Poly-time?
Given a number n,	$\cancel{n}$	$O(n)$ ✗
return its square.	$\log n$	$O(n^2)$

WHAT DOES POLYNOMIAL-TIME MEAN?

Problem	Input size	Poly-time?
Given a number n, return its square.	~~n~~	$O(n)$ ✗
	$\log n$	$O(n^2)$ ✗

WHAT DOES POLYNOMIAL-TIME MEAN?

Problem	Input size	Poly-time?
Given a number n, return its square.	$\cancel{n}$ $\log n$	$O(n)$ ✗ $O(n^2)$ ✗ $O(\log^2 n)$

WHAT DOES POLYNOMIAL-TIME MEAN?

Problem	Input size	Poly-time?
Given a number n,	$\cancel{n}$	$O(n)$ ✗
return its square.	$\log n$	$O(n^2)$ ✗
		$O(\log^2 n)$ ✓

WHAT DOES POLYNOMIAL-TIME MEAN?

Problem	Input size	Poly-time?	
Given a number n, return its square.	$\cancel{n}$	$O(n)$	✗
	$\log n$	$O(n^2)$	✗
		$O(\log^2 n)$	✓
		$O(n \log n)$	

WHAT DOES POLYNOMIAL-TIME MEAN?

Problem	Input size	Poly-time?	
Given a number n, return its square.	$\cancel{n}$ $\log n$	$O(n)$	✗
		$O(n^2)$	✗
		$O(\log^2 n)$	✓
		$O(n \log n)$	✗

WHAT DOES POLYNOMIAL-TIME MEAN?

Problem	Input size	Poly-time?	
Given a number n, return its square.	~~n~~	$O(n)$	✗
	$\log n$	$O(n^2)$	✗
		$O(\log^2 n)$	✓
		$O(n \log n)$	✗
		$O(\sqrt{\log n})$	

WHAT DOES POLYNOMIAL-TIME MEAN?

Problem	Input size	Poly-time?
Given a number n, return its square.	$\cancel{n}$	$O(n)$ ✗
	$\log n$	$O(n^2)$ ✗
		$O(\log^2 n)$ ✓
		$O(n \log n)$ ✗
		$O(\sqrt{\log n})$ ✓

WHAT DOES POLYNOMIAL-TIME MEAN?

Problem	Input size	Poly-time?
Find the largest entry in an n-element array $A = \boxed{a_1 \mid a_2 \mid \cdots \mid a_n}$		

WHAT DOES POLYNOMIAL-TIME MEAN?

Problem	Input size	Poly-time?
Find the largest entry in an n-element array	$n \cdot c$	
$A = \begin{array}{\|c\|c\|c\|c\|}\hline a_1 & a_2 & \cdots & a_n \\ \hline\end{array}$	c = # bits to encode any entry a_i	

WHAT DOES POLYNOMIAL-TIME MEAN?

Problem	Input size	Poly-time?
Find the largest entry in an n-element array	$n \cdot c$	$O(n^2 c^2)$
$A = \begin{array}{\|c\|c\|c\|c\|} \hline a_1 & a_2 & \cdots & a_n \\ \hline \end{array}$	c = # bits to encode any entry a_i	

WHAT DOES POLYNOMIAL-TIME MEAN?

Problem	Input size	Poly-time?
Find the largest entry in an n-element array	$n \cdot c$	$O(n^2c^2)$ ✓
$A = \boxed{a_1 \mid a_2 \mid \cdots \mid a_n}$	c = # bits to encode any entry a_i	

WHAT DOES POLYNOMIAL-TIME MEAN?

Problem	Input size	Poly-time?
Find the largest entry in an n-element array	$n \cdot c$	$O(n^2 c^2)$ ✓
	c = # bits to encode any entry a_i	$O(n^{100} \sqrt{c})$

$$A = \boxed{a_1 \mid a_2 \mid \cdots \mid a_n}$$

WHAT DOES POLYNOMIAL-TIME MEAN?

Problem	Input size	Poly-time?
Find the largest entry in an n-element array	$n \cdot c$	$O(n^2 c^2)$ ✓
	c = # bits to encode any entry a_i	$O(n^{100}\sqrt{c})$ ✓

$$A = \boxed{a_1 \mid a_2 \mid \cdots \mid a_n}$$

WHAT DOES POLYNOMIAL-TIME MEAN?

Problem	Input size	Poly-time?
Find the largest entry in an n-element array	$n \cdot c$	$O(n^2 c^2)$ ✓
$A = \boxed{a_1 \mid a_2 \mid \cdots \mid a_n}$	c = # bits to encode any entry a_i	$O(n^{100}\sqrt{c})$ ✓
		$O(n^c)$

WHAT DOES POLYNOMIAL-TIME MEAN?

Problem	Input size	Poly-time?
Find the largest entry in an n-element array	$n \cdot c$	$O(n^2 c^2)$ ✓
$A = \boxed{a_1 \mid a_2 \mid \cdots \mid a_n}$	$c = \#$ bits to encode any entry a_i	$O(n^{100}\sqrt{c})$ ✓
		$O(n^c)$ ✗

WHAT DOES POLYNOMIAL-TIME MEAN?

Problem

Find the largest entry in an n-element array

$A =$

a_1	a_2	$\cdots$	a_n

Input size

$n \cdot c$

$c = \#$ bits to encode any entry a_i

Poly-time?

$O(n^2 c^2)$ ✓

$O(n^{100} \sqrt{c})$ ✓

$O(n^c)$ ✗

$O(n^2 \cdot 2^c)$

WHAT DOES POLYNOMIAL-TIME MEAN?

Problem	Input size	Poly-time?
Find the largest entry in an n-element array	$n \cdot c$	$O(n^2 c^2)$ ✓
$A = \boxed{a_1 \mid a_2 \mid \cdots \mid a_n}$	c = # bits to encode any entry a_i	$O(n^{100}\sqrt{c})$ ✓
		$O(n^c)$ ✗
		$O(n^2 \cdot 2^c)$ ✗

WHAT DOES POLYNOMIAL-TIME MEAN?

Problem

Find the largest entry in an n-element array

$$A = \begin{array}{|c|c|c|c|}\hline a_1 & a_2 & \cdots & a_n \\ \hline\end{array}$$

Input size

$n \cdot c$

c = # bits to encode any entry a_i

Poly-time?

$O(n^2 c^2)$ ✓

$O(n^{100} \sqrt{c})$ ✓

$O(n^c)$ ✗

$O(n^2 \cdot 2^c)$ ✗

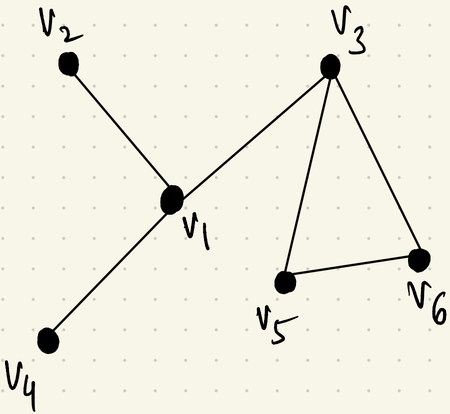
$O(a_3)$

WHAT DOES POLYNOMIAL-TIME MEAN?

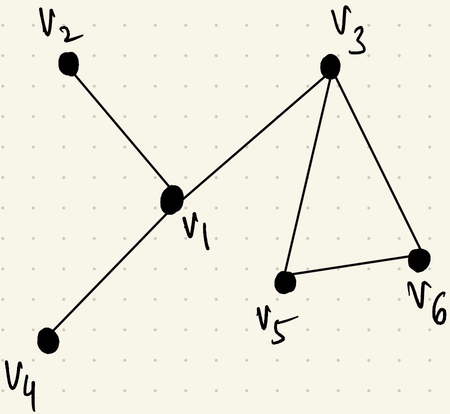
Problem	Input size	Poly-time?
Find the largest entry in an n-element array	$n \cdot c$	$O(n^2c^2)$ ✓
$A = \boxed{a_1 \mid a_2 \mid \cdots \mid a_n}$	$c = \#$ bits to encode any entry a_i	$O(n^{100}\sqrt{c})$ ✓
		$O(n^c)$ ✗
		$O(n^2 \cdot 2^c)$ ✗
		$O(a_3)$ ✗

Graph Algorithms

GRAPH

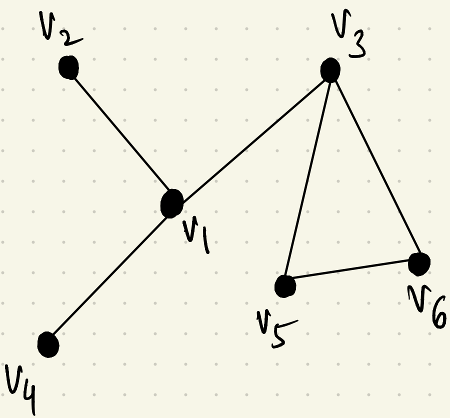


GRAPH



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

GRAPH

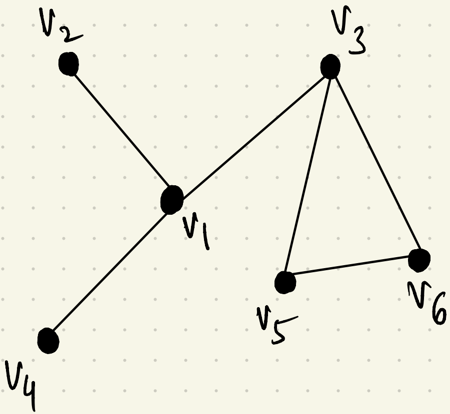


A graph G is a pair of sets (V, E)

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

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

GRAPH



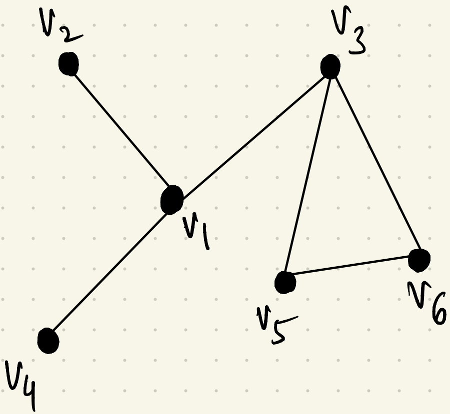
A graph G is a pair of sets (V, E)

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)

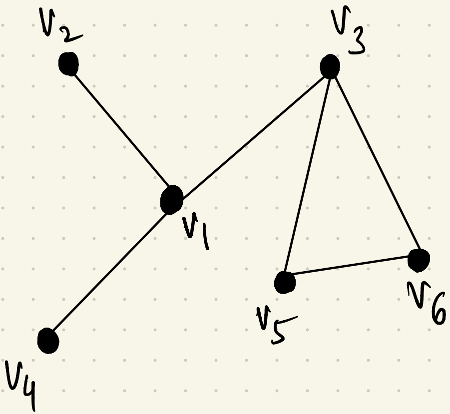
V : non-empty set of items called vertices / nodes

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

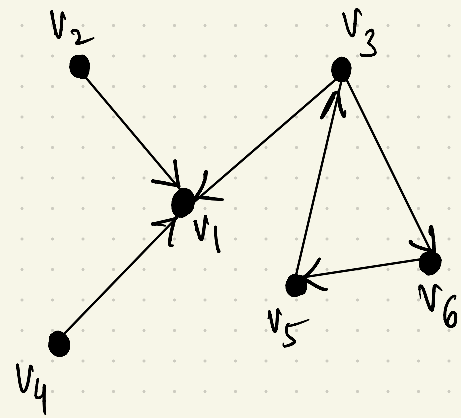
e.g., $E = \{ \{v_1, v_4\}, \{v_1, v_2\}, \{v_1, v_3\}, \{v_3, v_5\}, \{v_5, v_6\}, \{v_3, v_6\} \}$

GRAPH



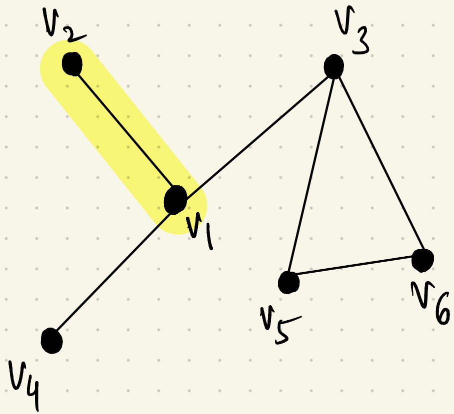
Edges can be undirected (unordered pair)

or directed (ordered pair).

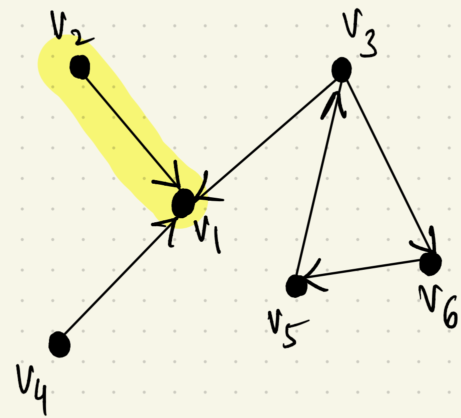


GRAPH

Edges can be undirected (unordered pair) or directed (ordered pair).



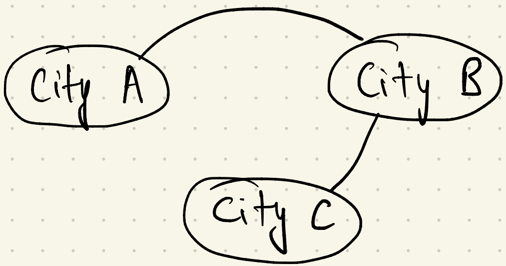
edge $\{v_1, v_2\}$ or $\{v_2, v_1\}$



edge (v_2, v_1)

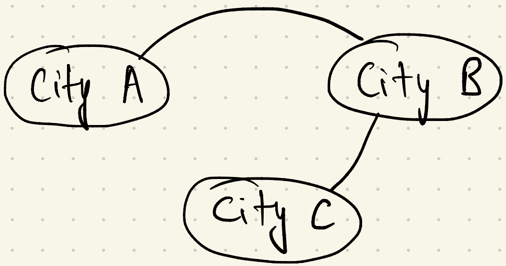
APPLICATIONS

APPLICATIONS

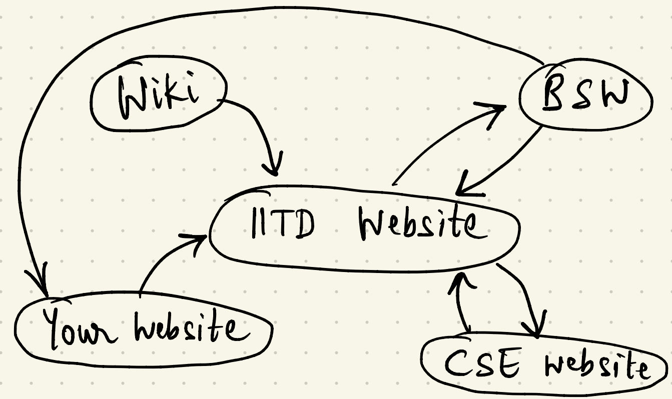


Road networks

APPLICATIONS

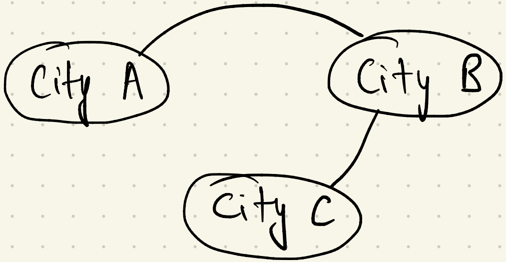


Road networks

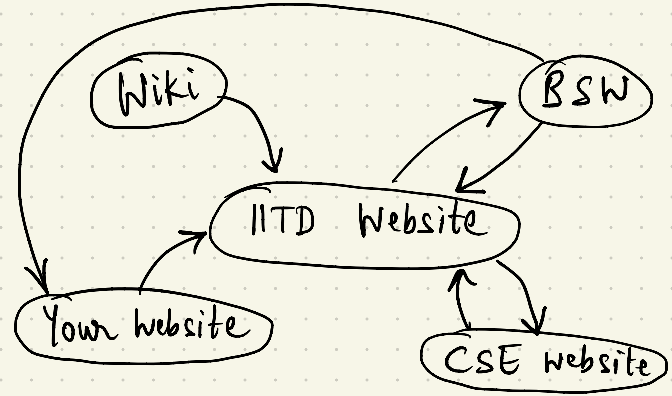


The Web

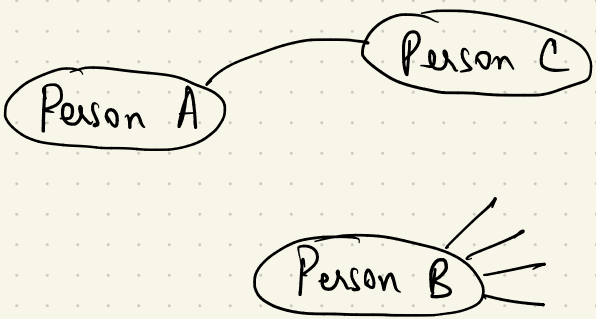
APPLICATIONS



Road networks

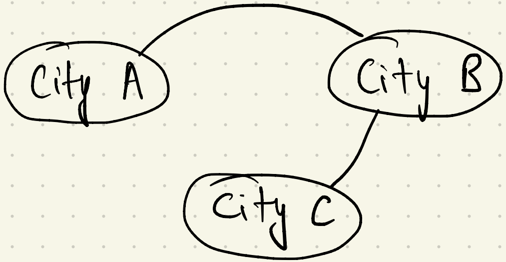


The Web

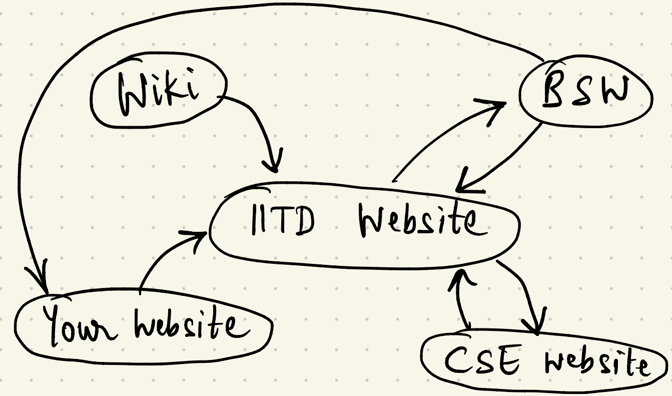


Social networks

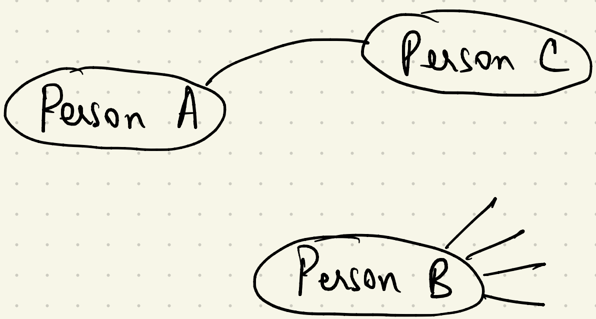
APPLICATIONS



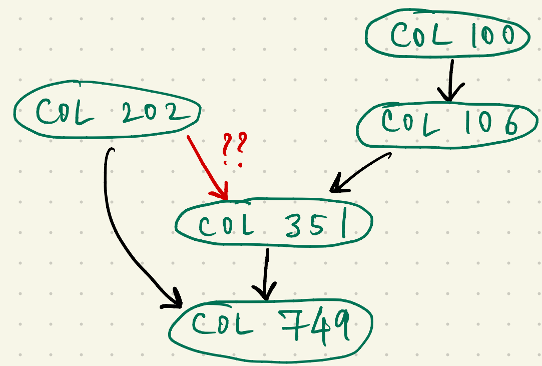
Road networks



The Web

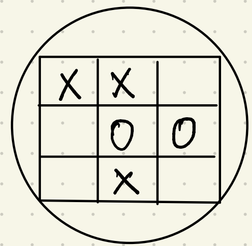


Social networks

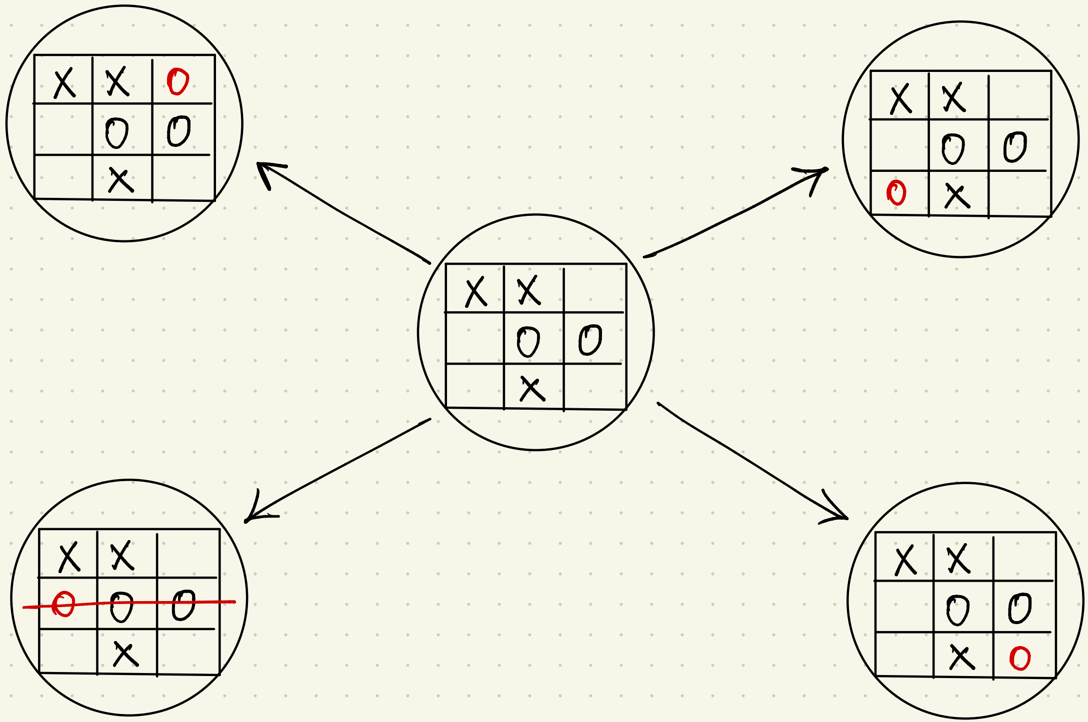


Precedence constraints

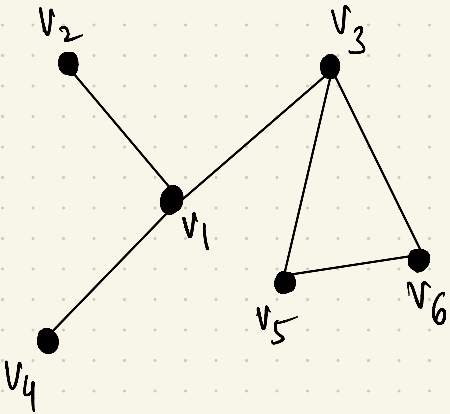
APPLICATIONS



APPLICATIONS



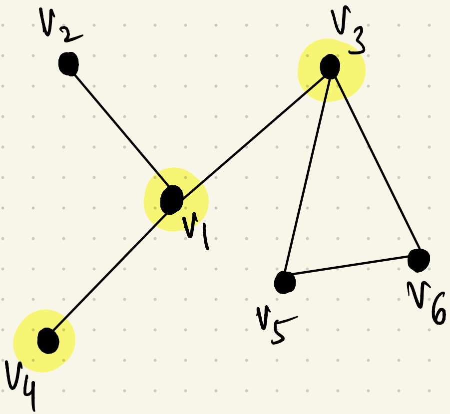
GRAPH



Two vertices v_i and v_j are adjacent

if $\{v_i, v_j\} \in E$.

GRAPH



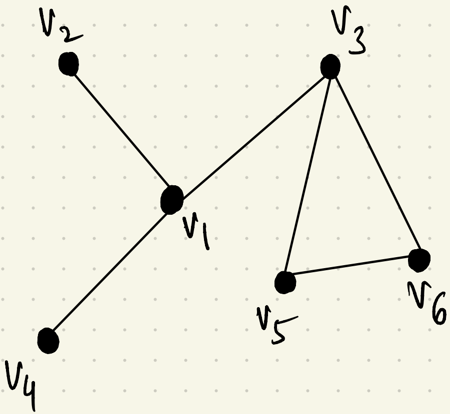
Two vertices v_i and v_j are adjacent

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 $\{v_i, v_j\} \in E$.

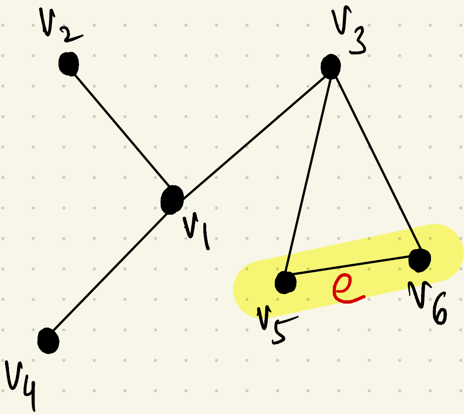
e.g., v_1 and v_3 are adjacent

v_3 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 $\{v_i, v_j\} \in E$.

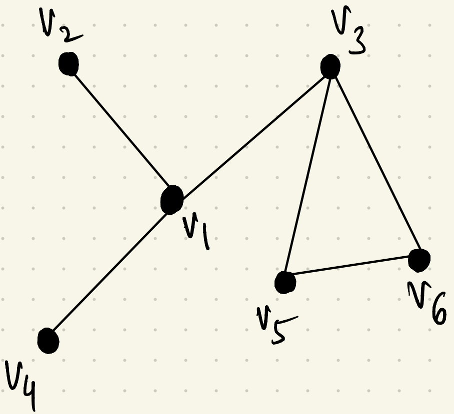
e.g., v_1 and v_3 are adjacent

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.

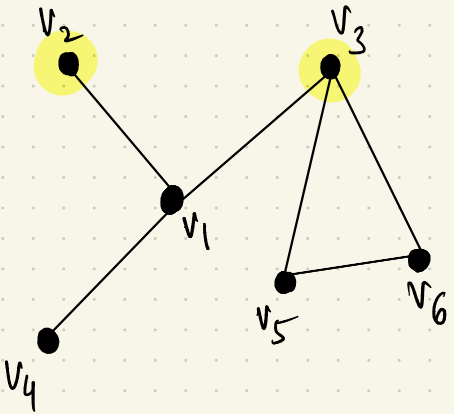
e.g., e is incident to v_5 and v_6 but not v_3.

GRAPH



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

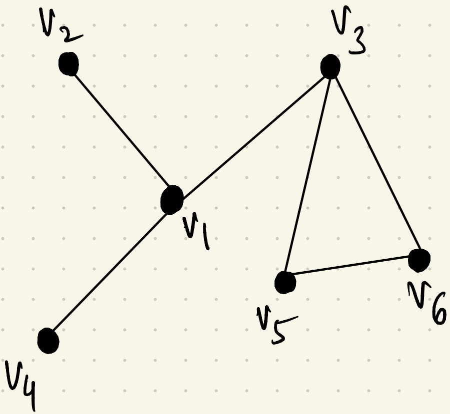
GRAPH



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

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

GRAPH

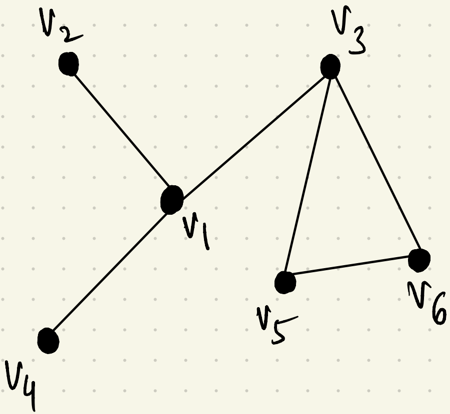


The number of edges incident to a vertex is called the degree.

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

A simple graph has no loops or multiedges.

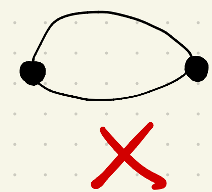
GRAPH



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

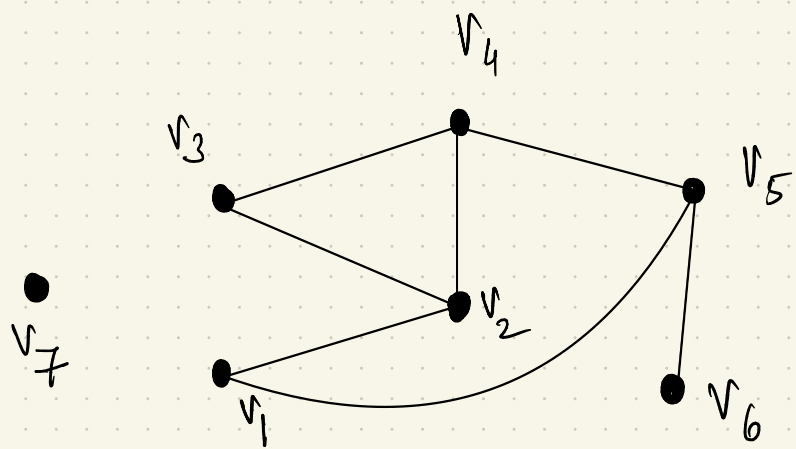
WALKS AND PATHS

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.,

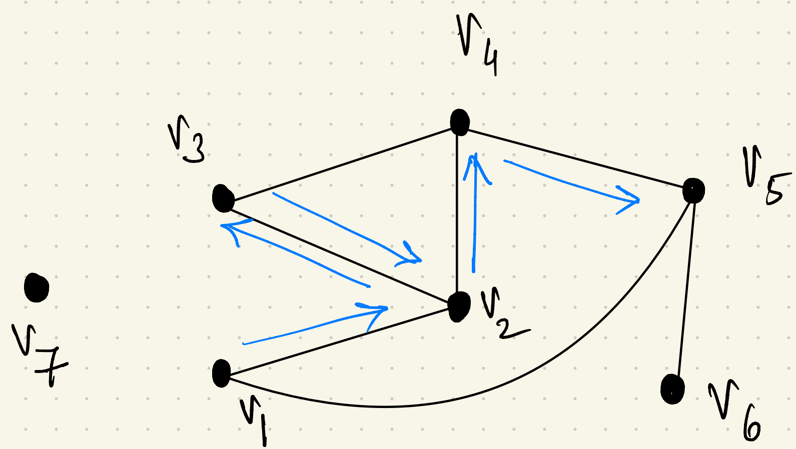


WALKS AND PATHS

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

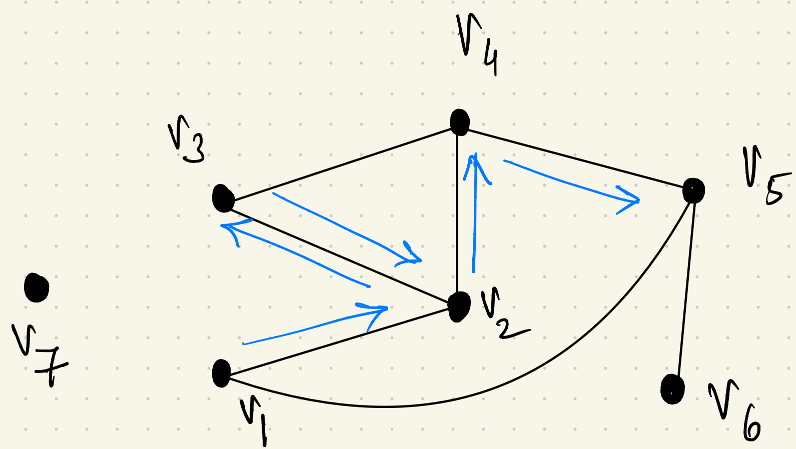


WALKS AND PATHS

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

E.g., $v_1 - v_2 - v_3 - v_2 - v_4 - v_5$ } walk of length 5

start ... end



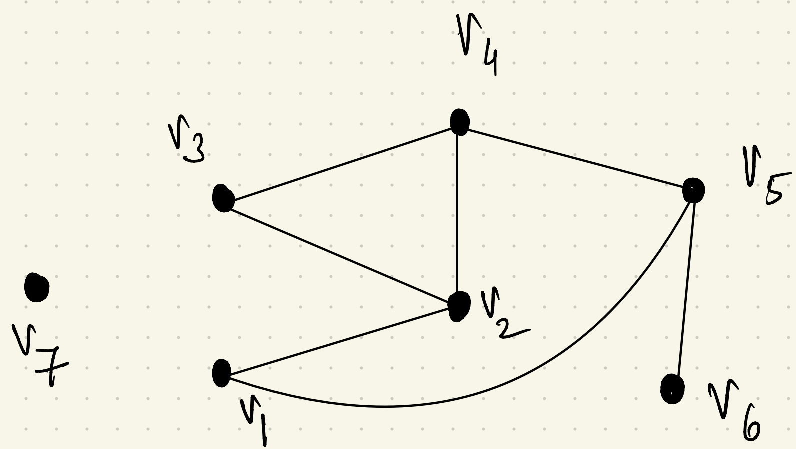
WALKS AND PATHS

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

WALKS AND PATHS

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

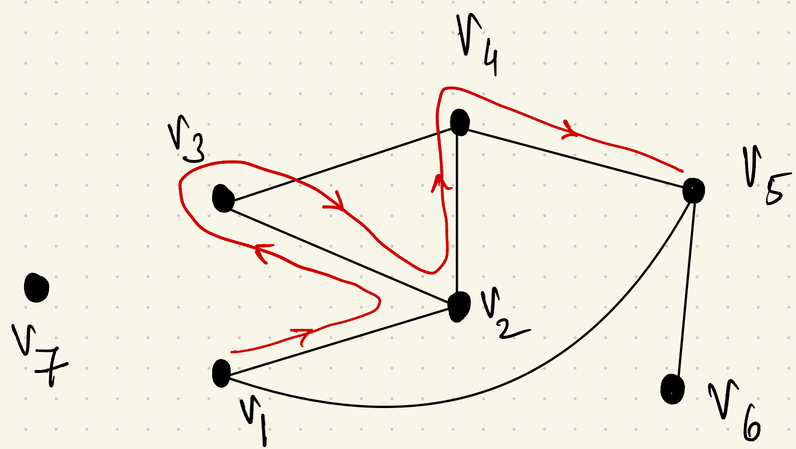
E.g.,



WALKS AND PATHS

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

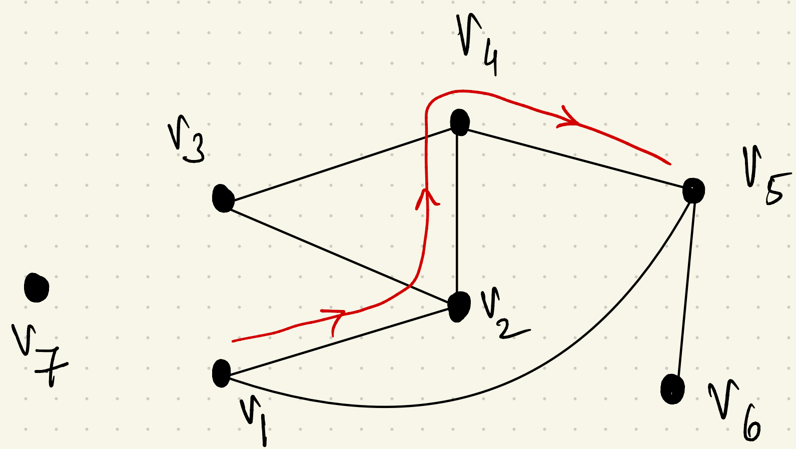
E.g., $v_1 - v_2 - v_3 - v_2 - v_4 - v_5$ ✗ Not a path



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

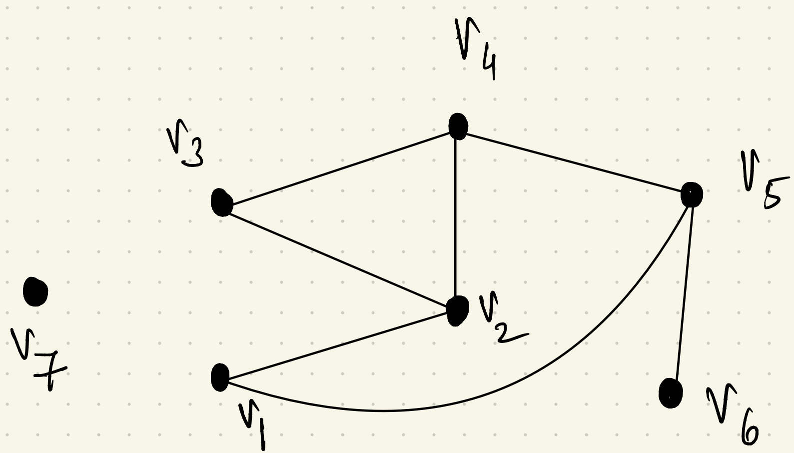
Proof: Exercise

CONNECTIVITY

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

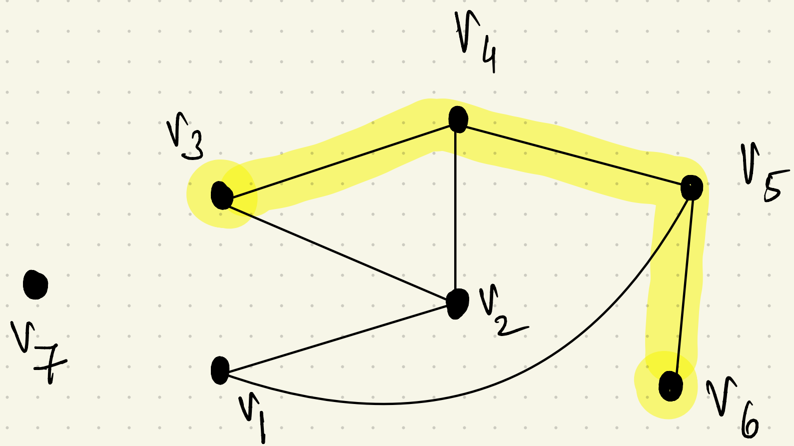
CONNECTIVITY

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



CONNECTIVITY

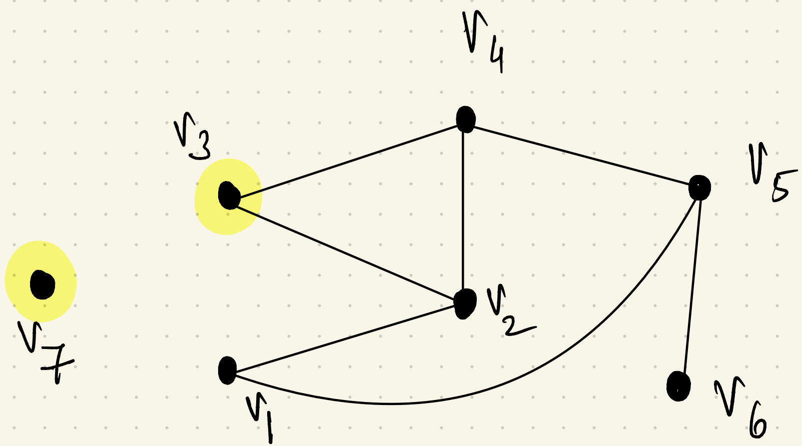
A pair of vertices u and v are connected if there is a path between u and v.



$v_3, v_6 \rightarrow$ Connected

CONNECTIVITY

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

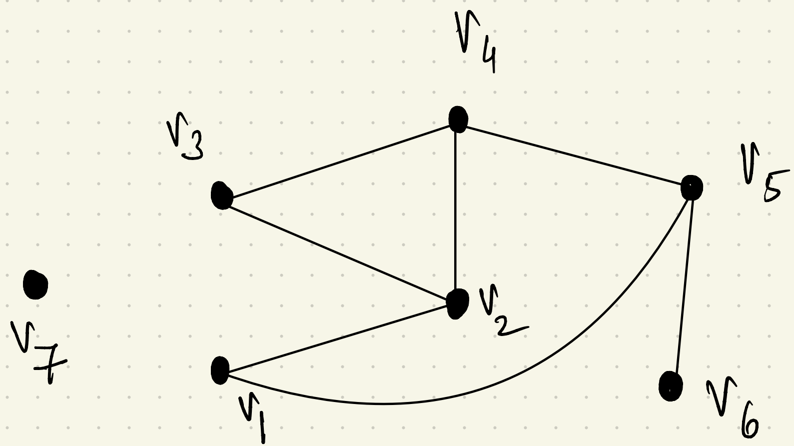


$v_3, v_6 \rightarrow$ Connected

$v_3, v_7 \rightarrow$ Not connected

CONNECTIVITY

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



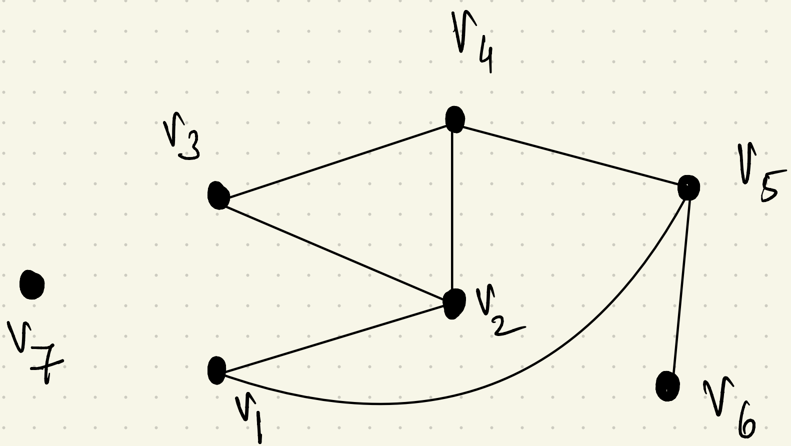
$v_3, v_6 \rightarrow$ Connected

$v_3, v_7 \rightarrow$ 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.

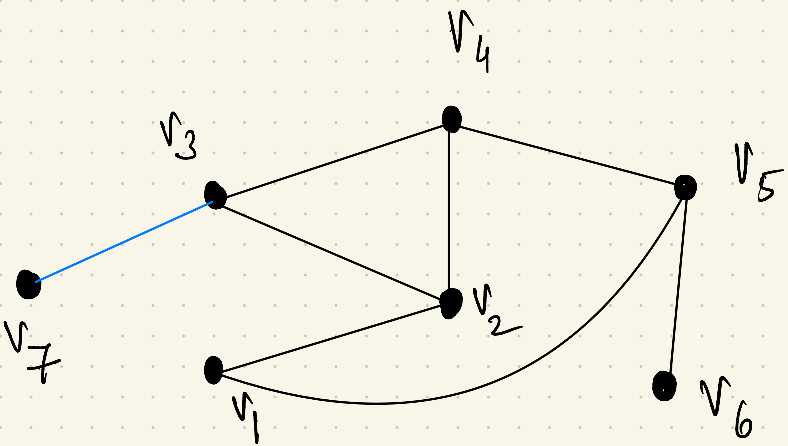


Not a connected graph

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

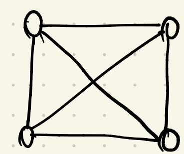
Maximum # edges =

Minimum # edges =

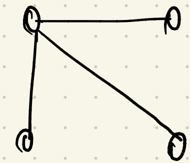
CONNECTIVITY

Given an undirected and connected graph on n vertices

Maximum # edges = ${}^{n}C_2$



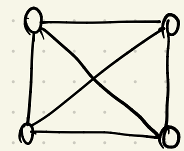
Minimum # edges = $n-1$



CONNECTIVITY

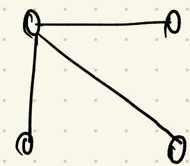
Given an undirected and connected graph on n vertices

Maximum # edges = $^nC_2$



complete graph

Minimum # edges = $n-1$



tree

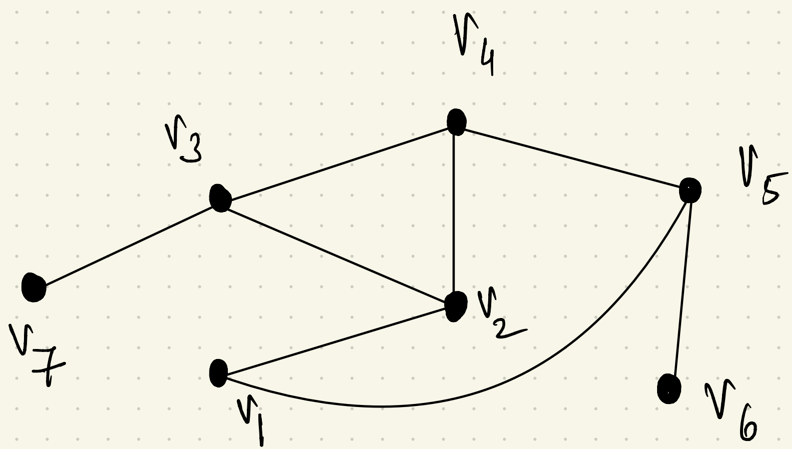
CYCLES AND CLOSED WALKS

CYCLES AND CLOSED WALKS

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

CYCLES AND CLOSED WALKS

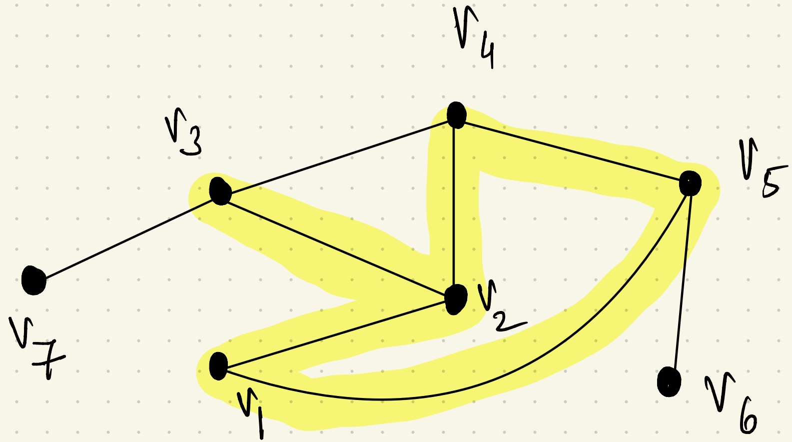
A closed walk is a walk that starts and ends at the same vertex.



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.

disallow single vertices or single edges

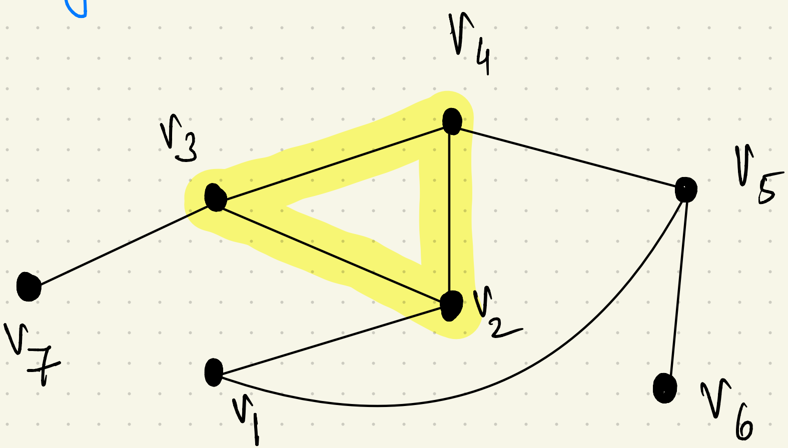
A **cycle** is a closed walk in which all vertices (except start and end) are different.

CYCLES AND CLOSED WALKS

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

disallow single vertices or single edges

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



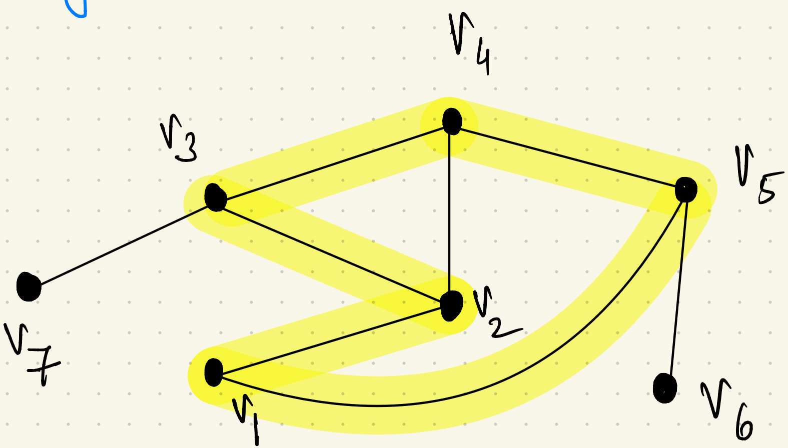
$v_2 - v_3 - v_4 - v_2$ is a cycle of length three

CYCLES AND CLOSED WALKS

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

disallow single vertices or single edges

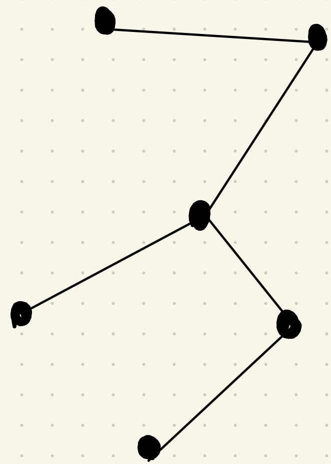
A **cycle** is a closed walk in which all vertices (except start and end) are different.



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

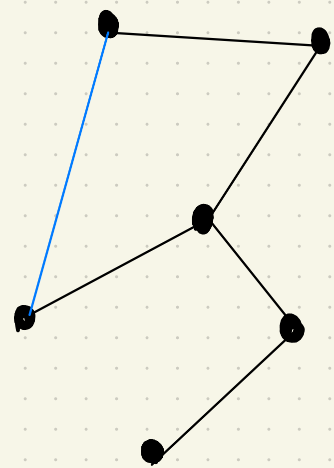
TREES

A tree is a connected and acyclic graph.



TREES

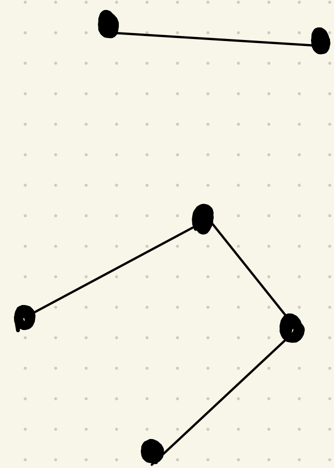
A tree is a connected and acyclic graph.



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 $\qquad$ 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 $\qquad$ m : number of edges

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

roughly,

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

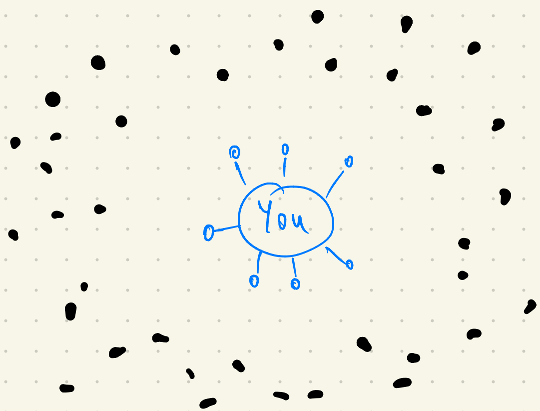
dense : $m = \Omega\left(\frac{n^2}{\log n}\right)$ $\qquad$ close to quadratic

SPARSE vs DENSE GRAPHS

n : number of vertices $\qquad$ m : number of edges

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

Facebook graph is sparse ...

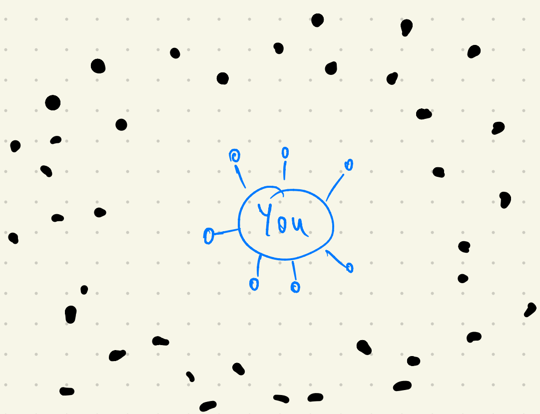


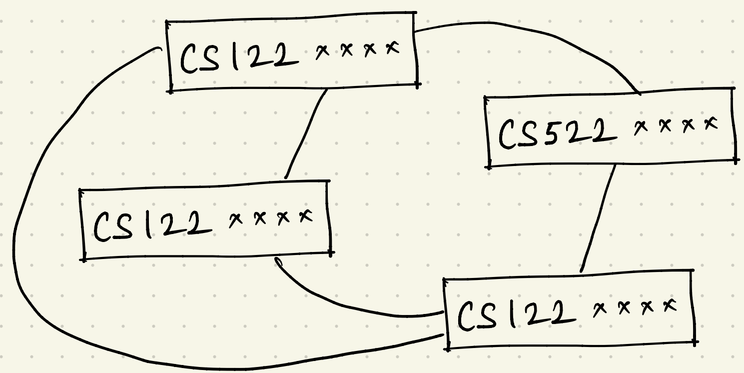
SPARSE vs DENSE GRAPHS

n : number of vertices $\qquad$ m : number of edges

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

Facebook graph is sparse ... $\qquad$ ... but also dense





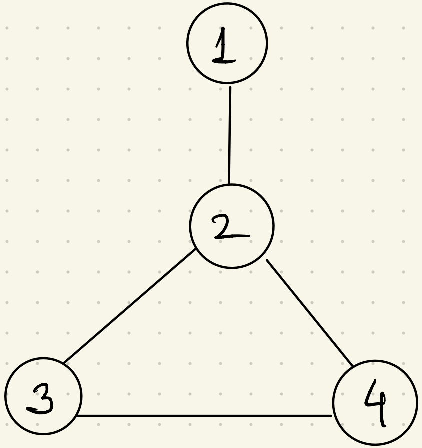
REPRESENTING GRAPHS

REPRESENTING GRAPHS

Adjacency matrix

REPRESENTING GRAPHS

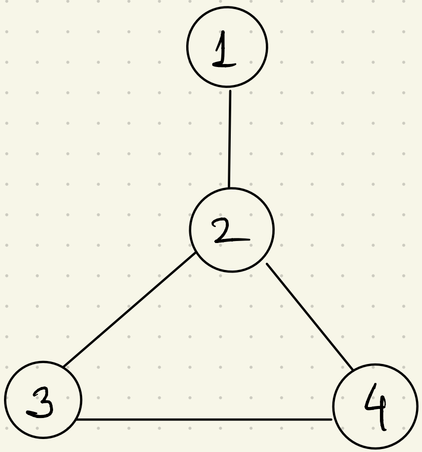
Adjacency matrix



REPRESENTING GRAPHS

Adjacency matrix

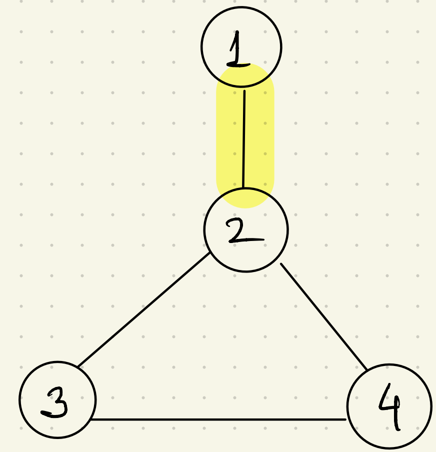
$$A = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & & & & \\ 2 & & & & \\ 3 & & & & \\ 4 & & & & \end{array}$$



REPRESENTING GRAPHS

Adjacency matrix

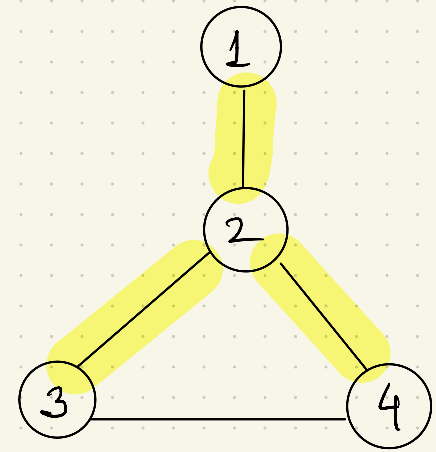
$$A = \begin{array}{c c} & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \left[\begin{array}{cccc} 0 & 1 & 0 & 0 \\ & & & \\ & & & \\ & & & \end{array}\right] \end{array}$$



REPRESENTING GRAPHS

Adjacency matrix

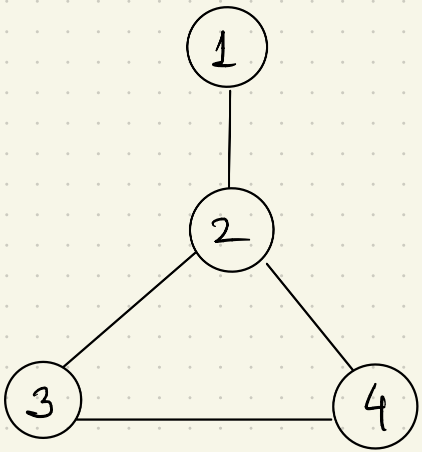
$$A = \begin{array}{cc} & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ & & & \\ & & & \end{bmatrix} \end{array}$$



REPRESENTING GRAPHS

Adjacency matrix

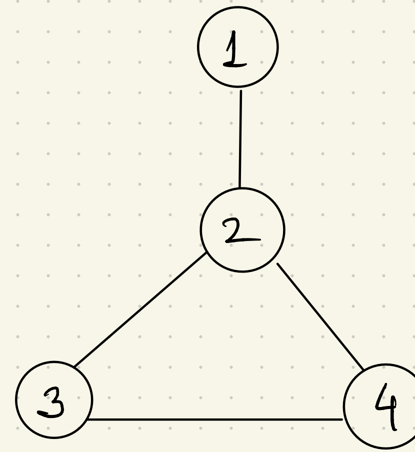
$$A = \begin{array}{c c} & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \end{array}$$



REPRESENTING GRAPHS

Adjacency matrix

$$A = \begin{array}{c c} & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

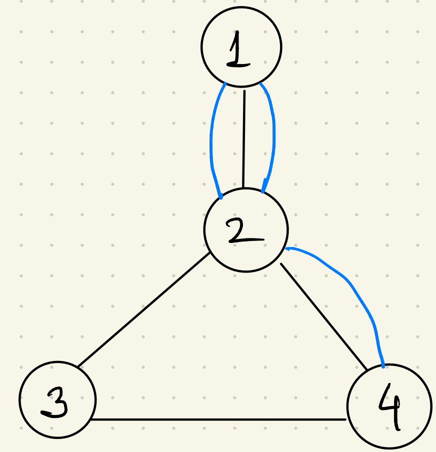


symmetric

REPRESENTING GRAPHS

Adjacency matrix

$$A = \begin{array}{c c} & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{bmatrix} 0 & 3 & 0 & 0 \\ 3 & 0 & 1 & 2 \\ 0 & 1 & 0 & 1 \\ 0 & 2 & 1 & 0 \end{bmatrix} \end{array}$$

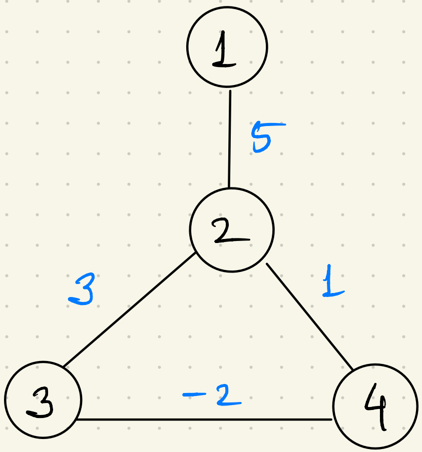


parallel edges

REPRESENTING GRAPHS

Adjacency matrix

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



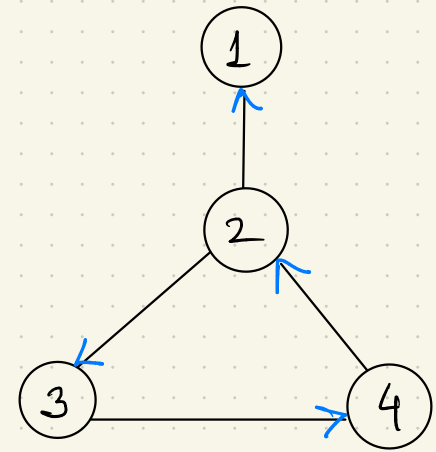
edge weights

REPRESENTING GRAPHS

Adjacency matrix

$$A = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & -1 & 0 & 0 \\ 2 & +1 & 0 & +1 & -1 \\ 3 & 0 & -1 & 0 & +1 \\ 4 & 0 & +1 & -1 & 0 \end{array}$$

skew symmetric



Directed edges

REPRESENTING GRAPHS

Adjacency matrix

$$A = \begin{array}{c c} & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{bmatrix} \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \end{bmatrix} \end{array}$$

$\Theta(n^2)$ space

OK for dense graphs

Not OK for sparse graphs

REPRESENTING GRAPHS

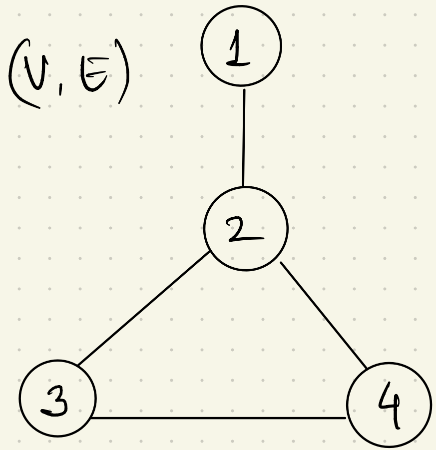
Adjacency list

REPRESENTING GRAPHS

Adjacency list

array of $|V|$ linked lists

$G = (V, E)$



REPRESENTING GRAPHS

Adjacency list

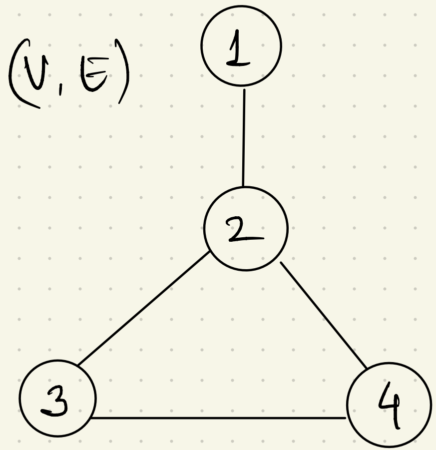
array of $|V|$ linked lists

for each vertex $u \in V$,

Adj[u] stores neighbors of u

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

$G = (V, E)$



REPRESENTING GRAPHS

Adjacency list

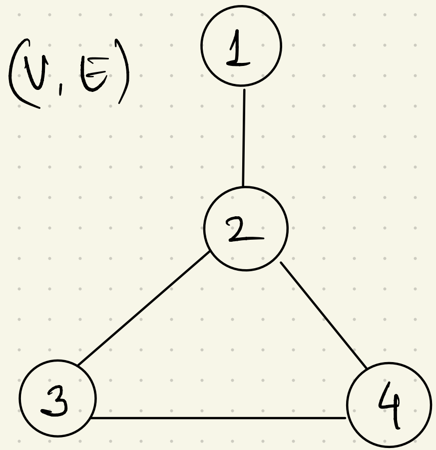
array of $|V|$ linked lists

for each vertex $u \in V$,

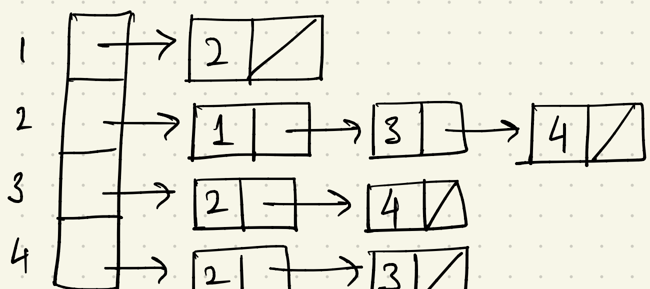
Adj[u] stores neighbors of u

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

$G = (V, E)$



Adj

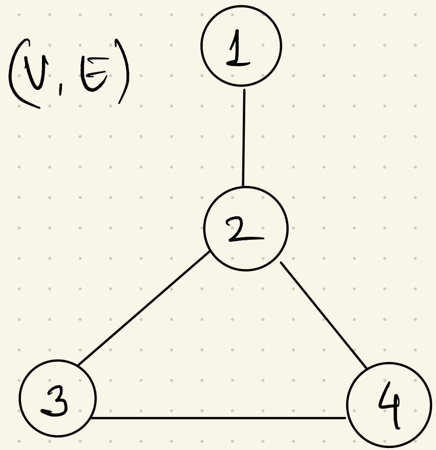


REPRESENTING GRAPHS

Adjacency list

array of $|V|$ linked lists

$G = (V, E)$

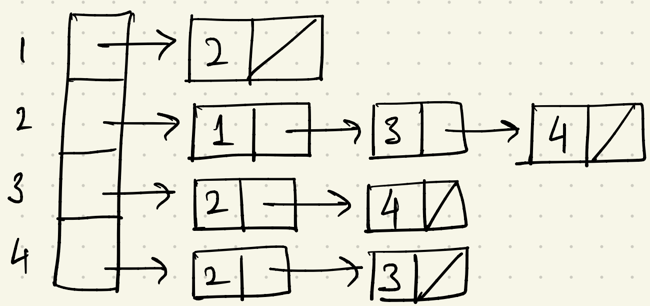


for each vertex $u \in V$,

Adj[u] stores neighbors of u

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

Adj



Space : ?

REPRESENTING GRAPHS

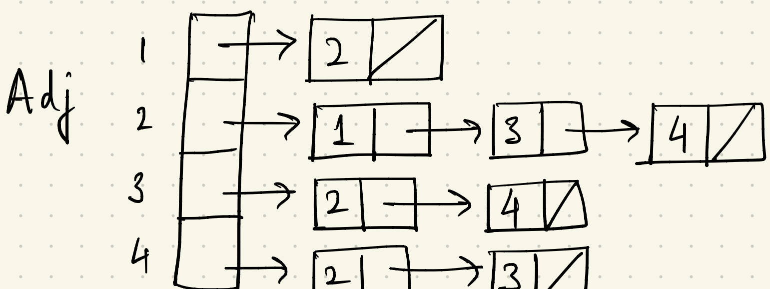
Adjacency list

array of $|V|$ linked lists

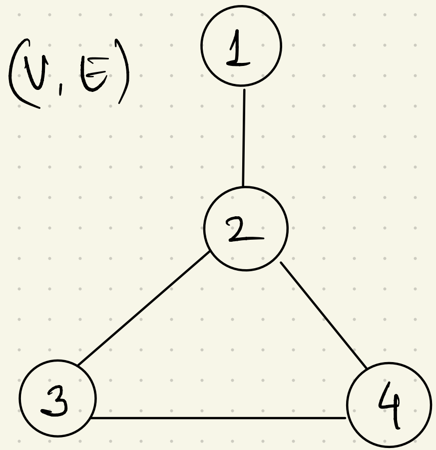
for each vertex $u \in V$,

Adj[u] stores neighbors of u

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



$G = (V, E)$



Space : $\Theta(|V| + |E|)$

or $\Theta(m+n)$

REPRESENTING GRAPHS

Which is better : adjacency matrix or adjacency list ?

REPRESENTING GRAPHS

Which is better : adjacency matrix or adjacency list ?

Depends on graph density and operations needed.

REPRESENTING GRAPHS

This course

Which is better : adjacency matrix or adjacency list ?

Depends on graph density and operations needed.

REPRESENTING GRAPHS

This course

Which is better : adjacency matrix or adjacency list?

Depends on graph density and operations needed.

Adjacency lists

- great for "follow your nose" operations
- better for massive sparse graphs (e.g., the web graph)

GRAPH SEARCH

GRAPH SEARCH

Checking connectivity

- Is there a way to reach Hauz Khas from Connaught Place?

GRAPH SEARCH

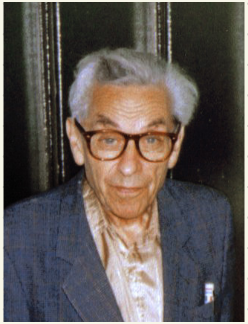
Checking connectivity

- Is there a way to reach Hauz Khas from Connaught Place?
- Is my Erdös number ≤ 3?

GRAPH SEARCH

Checking connectivity

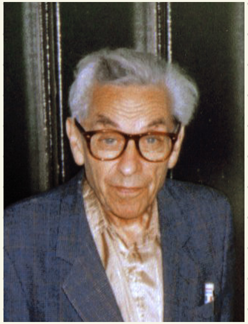
- Is there a way to reach Hauz Khas from Connaught Place?
- Is my Erdös number ≤ 3?



GRAPH SEARCH

Checking connectivity

- Is there a way to reach Hauz Khas from Connaught Place?
- Is my Erdős number ≤ 3?

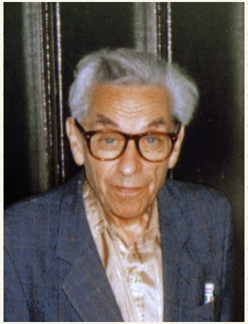


0

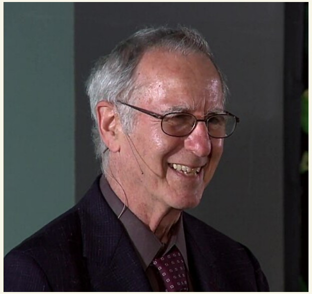
GRAPH SEARCH

Checking connectivity

- Is there a way to reach Hauz Khas from Connaught Place?
- Is my Erdös number ≤ 3?



0

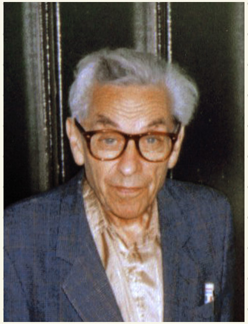


1

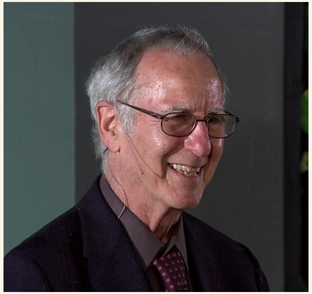
GRAPH SEARCH

Checking connectivity

- Is there a way to reach Hauz Khas from Connaught Place?
- Is my Erdös number ≤ 3?



0



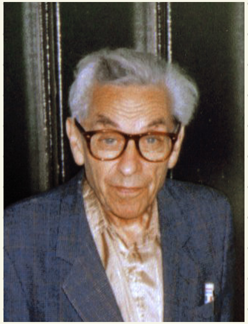
1

2

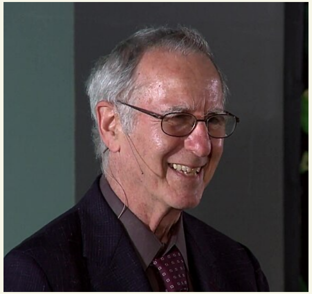
GRAPH SEARCH

Checking connectivity

- Is there a way to reach Hauz Khas from Connaught Place ?
- Is my Erdös number ≤ 3 ?



0



1

2

3

GRAPH SEARCH

Checking connectivity

- Is there a way to reach Hauz Khas from Connaught Place?
- Is my Erdős number ≤ 3?

GRAPH SEARCH

Checking connectivity

- Is there a way to reach Hauz Khas from Connaught Place?
- Is my Erdös number ≤ 3?

Shortest paths

- cheapest combination of flight tickets

GRAPH SEARCH

Checking connectivity

- Is there a way to reach Hauz Khas from Connaught Place?
- Is my Erdös number ≤ 3?

Shortest paths

- cheapest combination of flight tickets

Planning (abstract)

- e.g., Sudoku, Rubik's cube, locomotion, painting