DSC 40B

Lecture 17 : Graphs

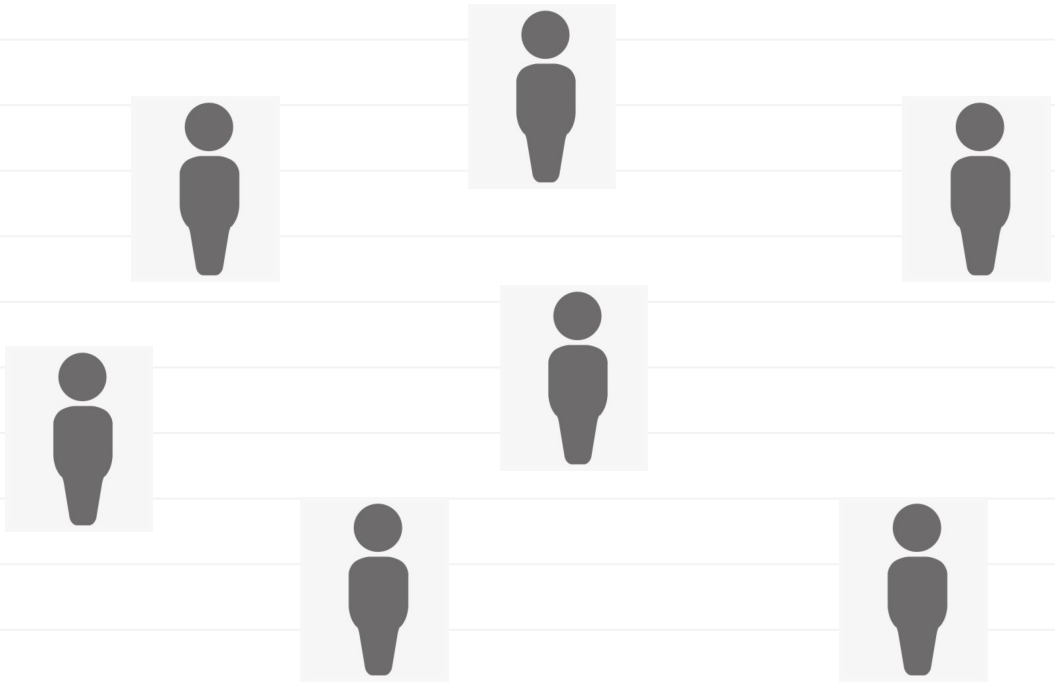
Back to the best/worst/case question

link

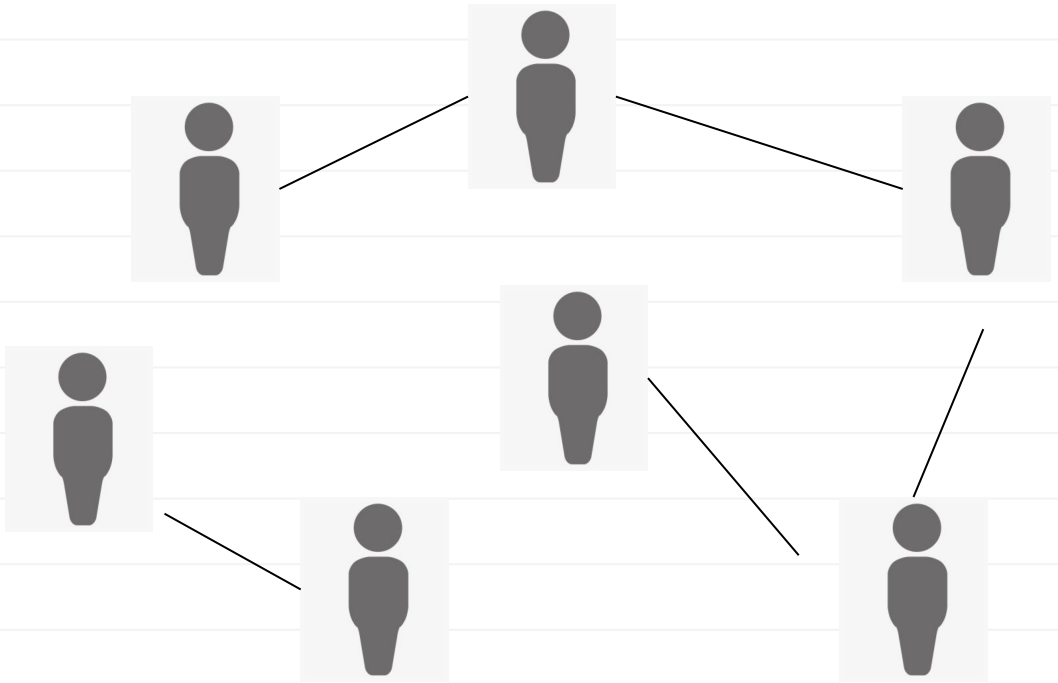
Data Types

- **Feature vectors**
  - We care about attributes of individuals.
- **Graphs**
  - We care about **relationships** between individuals.

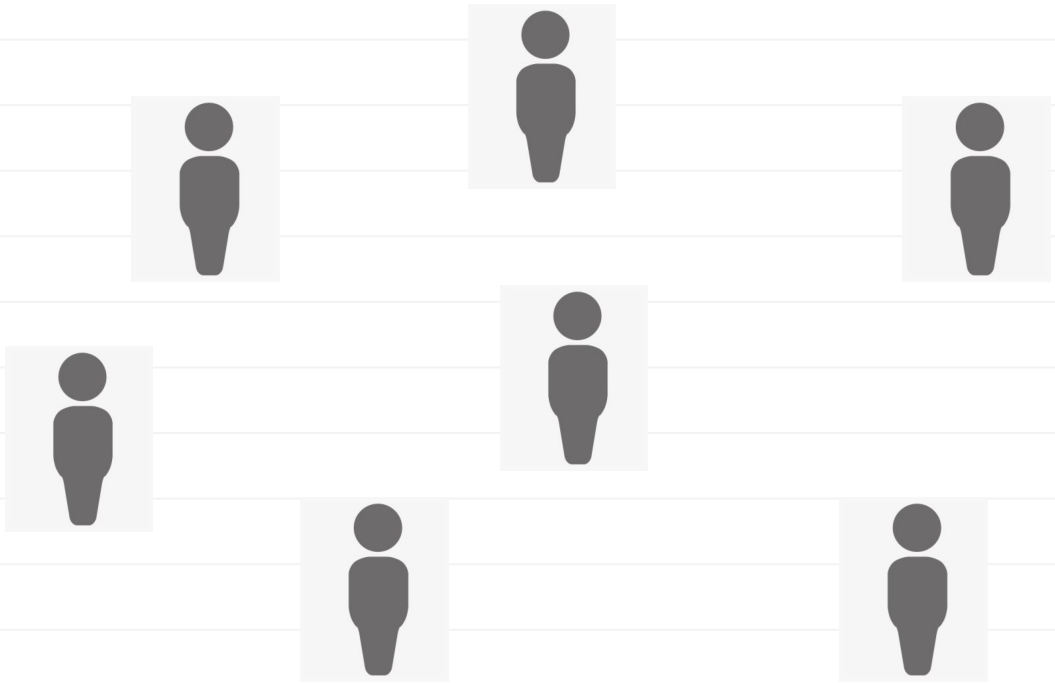
Example: Facebook



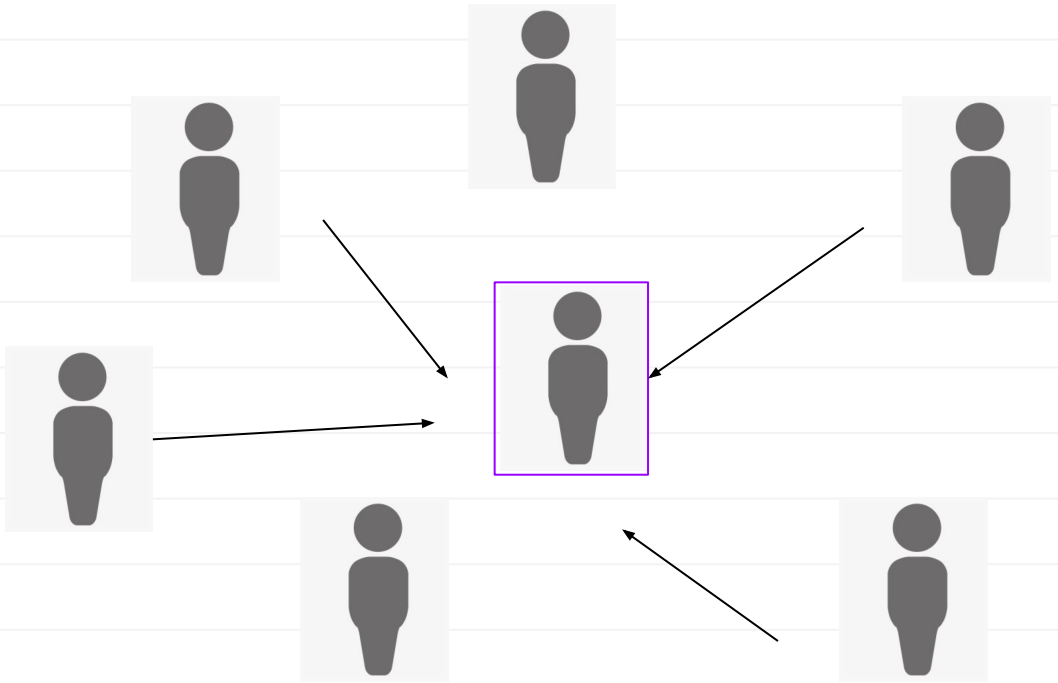
Example: Facebook



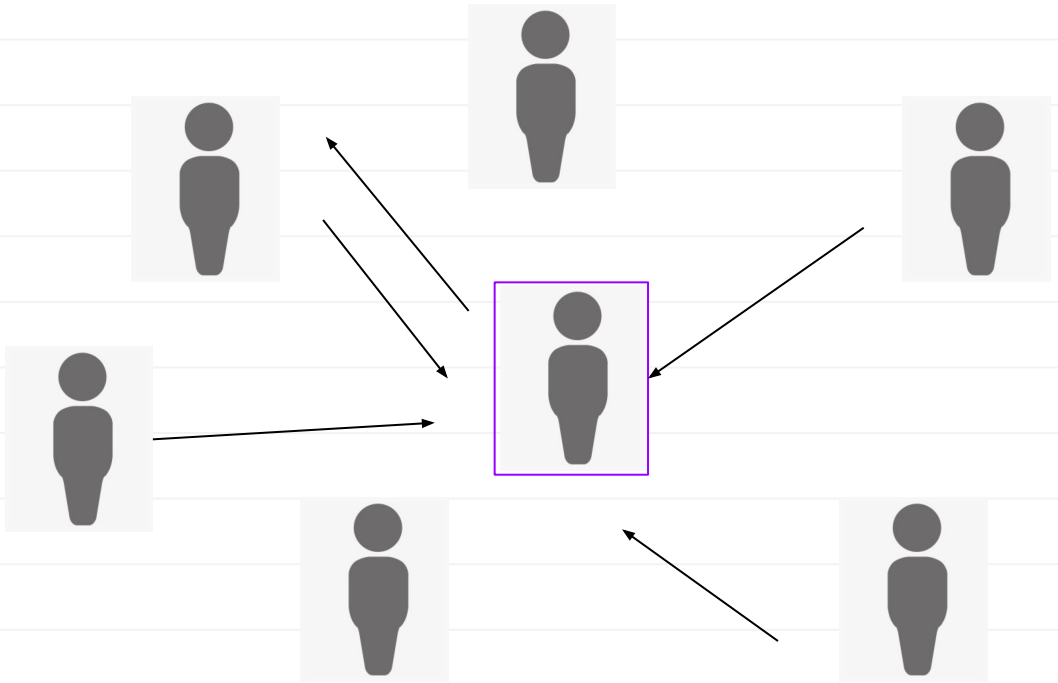
Example: Twitter



Example: Twitter



Example: Twitter



Definition

A directed graph (or digraph) G is a pair ($\boldsymbol{V}$, $\boldsymbol{E}$) where V is a finite set of nodes (or vertices) and E is a set of ordered pairs (the edges).

$V = \{a, b, c, d\}$

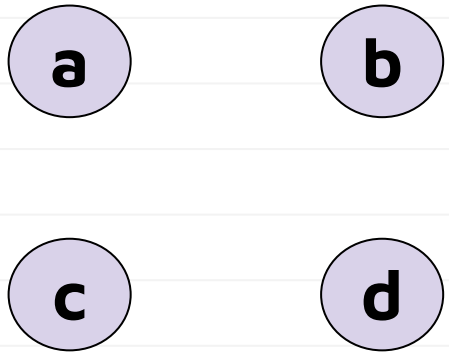
$E = \{(a, c), (a, b), (d, b), (b, d), (b, b)\}$

Definition

A directed graph (or digraph) G is a pair ($\boldsymbol{V}$, $\boldsymbol{E}$) where V is a finite set of nodes (or vertices) and E is a set of ordered pairs (the edges).

$\boldsymbol{V = \{a, b, c, d\}}$

$E = \{(a, c), (a, b), (d, b), (b, d), (b, b)\}$

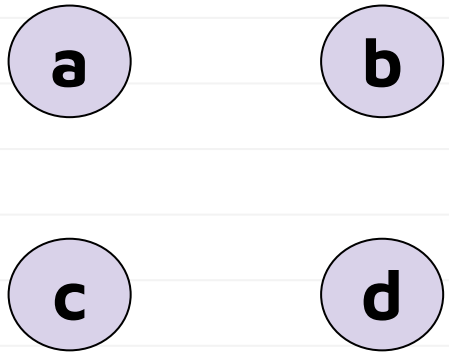


Definition

A directed graph (or digraph) G is a pair ($\boldsymbol{V}$, $\boldsymbol{E}$) where V is a finite set of nodes (or vertices) and E is a set of ordered pairs (the edges).

$\boldsymbol{V = \{a, b, c, d\}}$

$E = \{(a, c), (a, b), (d, b), (b, d), (b, b)\}$

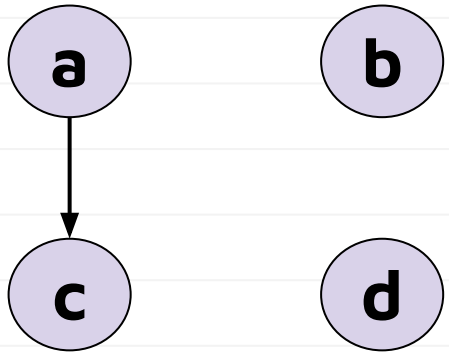


Definition

A directed graph (or digraph) G is a pair ($\boldsymbol{V}$, $\boldsymbol{E}$) where V is a finite set of nodes (or vertices) and E is a set of ***ordered*** pairs (the edges).

$V = \{a, b, c, d\}$

$E = \{\boldsymbol{(a, c)}, (a, b), (d, b), (b, d), (b, b)\}$

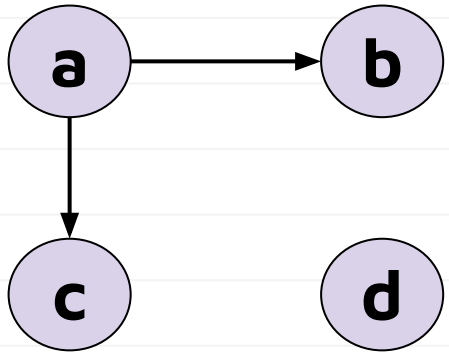


Definition

A directed graph (or digraph) G is a pair ($\boldsymbol{V}$, $\boldsymbol{E}$) where V is a finite set of nodes (or vertices) and E is a set of ***ordered*** pairs (the edges).

$V = \{a, b, c, d\}$

$E = \{(a, c), \mathbf{(a, b)}, (d, b), (b, d), (b, b)\}$

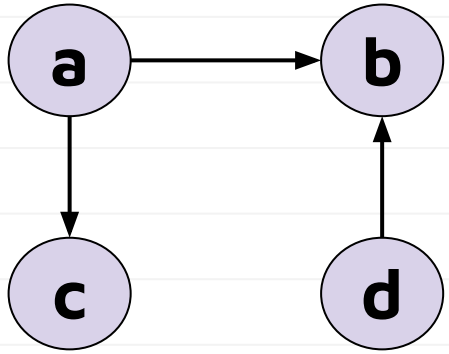


Definition

A directed graph (or digraph) G is a pair ($\boldsymbol{V}$, $\boldsymbol{E}$) where V is a finite set of nodes (or vertices) and E is a set of ***ordered*** pairs (the edges).

$V = \{a, b, c, d\}$

$E = \{(a, c), (a, b), \mathbf{(d, b)}, (b, d), (b, b)\}$

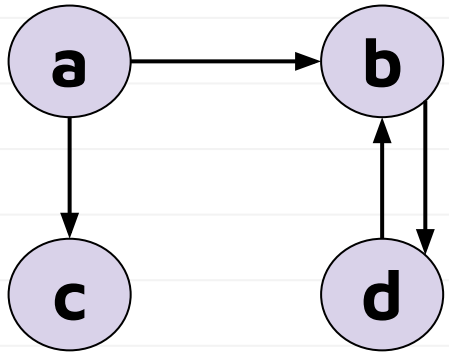


Definition

A directed graph (or digraph) G is a pair ($\boldsymbol{V}$, $\boldsymbol{E}$) where V is a finite set of nodes (or vertices) and E is a set of ***ordered*** pairs (the edges).

$V = \{a, b, c, d\}$

$E = \{(a, c), (a, b), (d, b), \mathbf{(b, d)}, (b, b)\}$

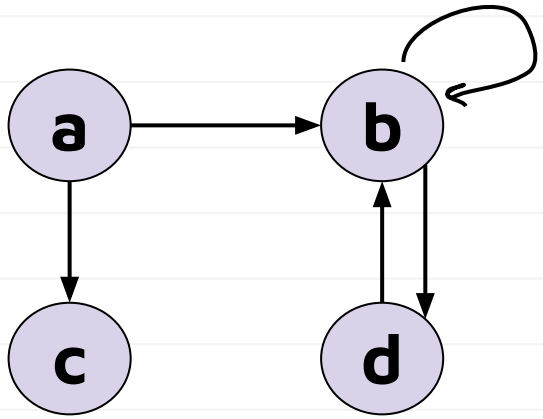


Definition

A directed graph (or digraph) G is a pair ($\boldsymbol{V}$, $\boldsymbol{E}$) where V is a finite set of nodes (or vertices) and E is a set of ***ordered*** pairs (the edges).

$V = \{a, b, c, d\}$

$E = \{(a, c), (a, b), (d, b), (b, d), \mathbf{(b, b)}\}$



Directed Graphs (More Formally)

- E is a subset of the Cartesian product, $V \times V$.

Example:

$\{a, b, c\} \times \{1, 2\} =$

Directed Graphs (More Formally)

- E is a subset of the Cartesian product, $V \times V$.

Example: { **(a,1)**, **(a,2)**,

$\{a, b, c\} \times \{1, 2\}$ =

}

Directed Graphs (More Formally)

- E is a subset of the Cartesian product, $V \times V$.

Example: $\{a, \boldsymbol{b}, c\} \times \{\mathbf{1}, \mathbf{2}\} =$ { (a,1), (a,2), **(b, 1), (b,2)**, }

Directed Graphs (More Formally)

- E is a subset of the Cartesian product, $V \times V$.

Example: $\{a, b, \boldsymbol{c}\} \times \{\mathbf{1, 2}\}$ = { (a,1), (a,2), (b, 1), (b,2), **(c, 1), (c, 2)** }

Directed Graphs (More Formally)

- E is a subset of the Cartesian product, $V \times V$.

Example:

$\{a, b, c\} \times \{a, b, c\} =$ { (a,a), (a,b), (a,c)
(b, a), (b,b), (b,c),
(c, a), (c, b), (c, c)
}

Directed Graphs (More Formally)

- E is a **subset** of the Cartesian product, $V \times V$.

Example: { **(a,a)**, (a,b), **(a,c)**

{a, b, c} × {a, b, c} = (b, a), (b,b), (b,c),

(c, a), **(c, b)**, (c, c)

}

Consequences

Because the edge set of a directed graph is allowed to be **any** subset of $V \times V$:

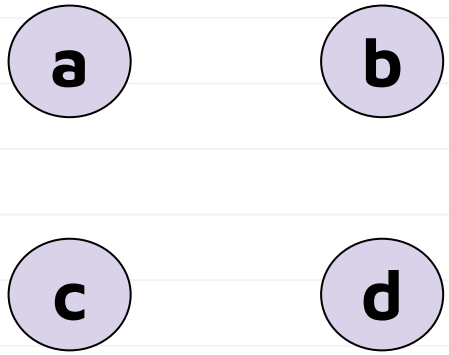
- the edges have **directions**.
  - e.g., (a, b) is "from a to b"
- can have "**opposite**" edges.
  - e.g., (a, b) and (b, a).
- can have "**self-loops**"
  - e.g., (a, a)

Definition

An undirected graph G is a pair $(\boldsymbol{V}, \boldsymbol{E})$ where V is a finite set of nodes (or vertices) and E is a set of **unordered** pairs (the edges).

$\boldsymbol{V} = \{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d}\}$

$E = \{(a, c), (a, b), (d, b)\}$

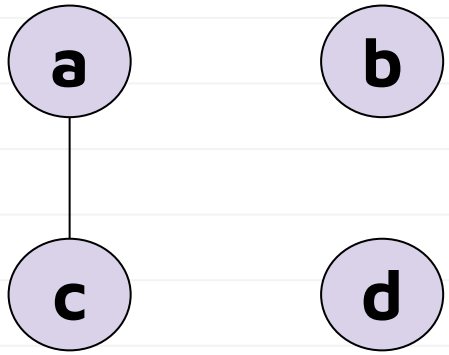


Definition

An undirected graph G is a pair $(\boldsymbol{V}, \boldsymbol{E})$ where V is a finite set of nodes (or vertices) and E is a set of **unordered** pairs (the edges).

$V = \{a, b, c, d\}$

$E = \{\boldsymbol{(a, c)}, (a, b), (d, b)\}$

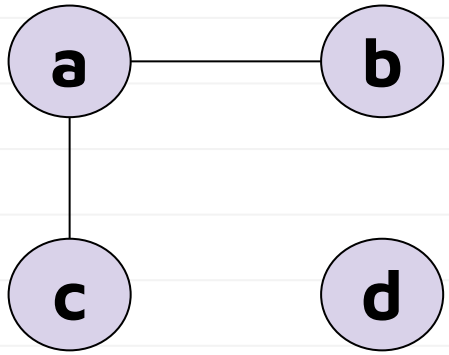


Definition

An undirected graph G is a pair $(\boldsymbol{V}, \boldsymbol{E})$ where V is a finite set of nodes (or vertices) and E is a set of **unordered** pairs (the edges).

$V = \{a, b, c, d\}$

$E = \{(a, c), \mathbf{(a, b)}, (d, b)\}$

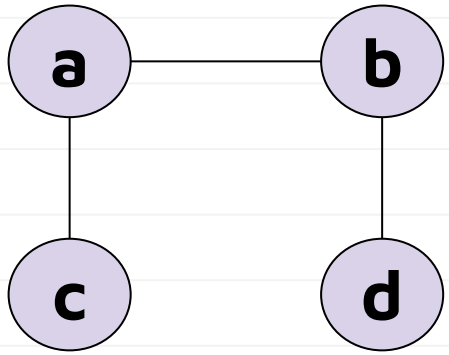


Definition

An undirected graph G is a pair $(\boldsymbol{V}, \boldsymbol{E})$ where V is a finite set of nodes (or vertices) and E is a set of **unordered** pairs (the edges).

$V = \{a, b, c, d\}$

$E = \{(a, c), \mathbf{(a, b)}, \mathbf{(d, b)}\}$



Undirected Graphs (More Formally)

An edge in an undirected graph is a set $\{u, v\}$ where $u \neq v$. This has consequences:

- the edges have **no direction**.
  - e.g., $\{a, b\}$ is not "from" a "to" b.
- cannot have "opposite" edges.
  - e.g., $\{a, b\}$ and $\{b, a\}$ **are the same**.
- **cannot have "self-loops"**
  - e.g., $\{a, a\}$ is not a valid edge

Notational Note

Although edges in undirected graphs are sets, we typically write them as pairs: (u, v) instead of $\{u, v\}$.

Summary

- **Edges have direction?**:
  - Directed: ?
  - Undirected: ?
- **Self-loops, (u, u)?**
  - Directed: ?
  - Undirected: ?
- **Opposite edges, (u, v) and (v, u)?**
  - Directed: ?
  - Undirected: ?

Summary

- **Edges have direction?**:
  - Directed: **Yes**
  - Undirected: ?
- **Self-loops, (u, u)?**
  - Directed: ?
  - Undirected: ?
- **Opposite edges, (u, v) and (v, u)?**
  - Directed: ?
  - Undirected: ?

Summary

- **Edges have direction?**:
  - Directed: **Yes**
  - Undirected: **No**
- **Self-loops, (u, u)?**
  - Directed: ?
  - Undirected: ?
- **Opposite edges, (u, v) and (v, u)?**
  - Directed: ?
  - Undirected: ?

Summary

- **Edges have direction?**:
  - Directed: **Yes**
  - Undirected: **No**
- **Self-loops, (u, u)?**
  - Directed: **Yes**
  - Undirected: ?
- **Opposite edges, (u, v) and (v, u)?**
  - Directed: ?
  - Undirected: ?

Summary

- **Edges have direction?**:
  - Directed: **Yes**
  - Undirected: **No**
- **Self-loops, (u, u)?**
  - Directed: **Yes**
  - Undirected: **No**
- **Opposite edges, (u, v) and (v, u)?**
  - Directed: ?
  - Undirected: ?

Summary:

- **Edges have direction?**:
  - Directed: **Yes**
  - Undirected: **No**
- **Self-loops, (u, u)?**
  - Directed: **Yes**
  - Undirected: **No**
- **Opposite edges, (u, v) and (v, u)?**
  - Directed: **Yes**
  - Undirected: ?

Summary:

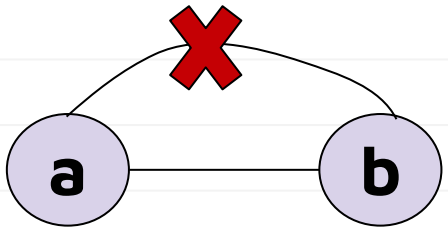
- **Edges have direction?**:
  - Directed: **Yes**
  - Undirected: **No**
- **Self-loops, (u, u)?**
  - Directed: **Yes**
  - Undirected: **No**
- **Opposite edges, (u, v) and (v, u)?**
  - Directed: **Yes**
  - Undirected: **No** (they are the same edge)

Note

- Neither directed nor undirected graphs can have **duplicate edges.**
  - There are other definitions which allow duplicate edges.

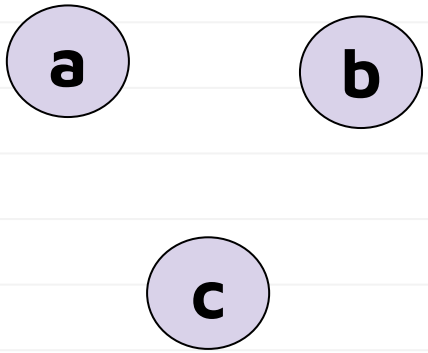
Note

- Neither directed nor undirected graphs can have **duplicate edges.**
  - There are other definitions which allow duplicate edges.



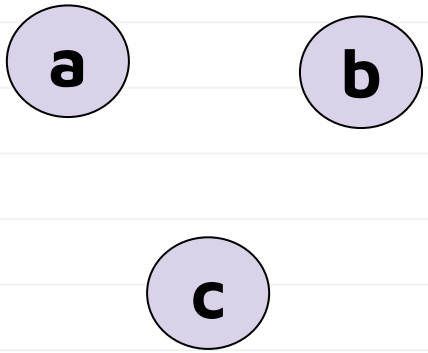
Note

Graphs don’t need to be “connected”



Note

Graphs don’t need to be “connected”

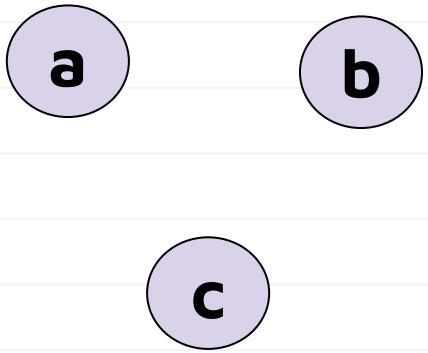


V = ?

E= ?

Note

Graphs don’t need to be “connected”

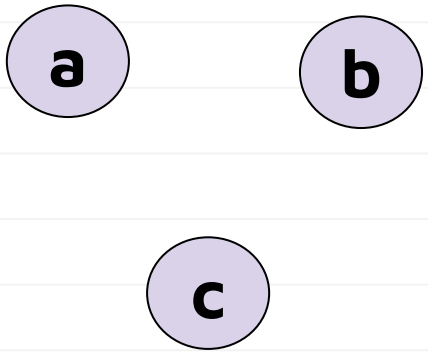


V = {a, b, c}

E= ?

Note

Graphs don’t need to be “connected”



V = {a, b, c}

E = { }

Exercise

- What is the **greatest** number edges possible in a **directed** graph with *n* nodes?

A: n

B: n^2

C: ~$n^2/2$

D: n choose 2

E: n^2 choose 2

Counting Edges

- What is the **greatest** number edges possible in a **directed** graph with *n* nodes?

a

b

c

d

Counting Edges

- What is the **greatest** number edges possible in a **directed** graph with *n* nodes?

a

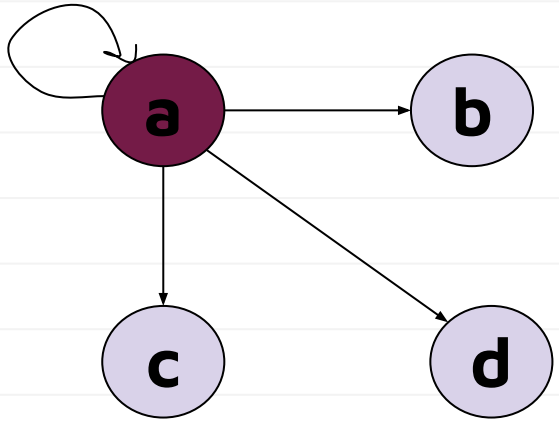
b

c

d

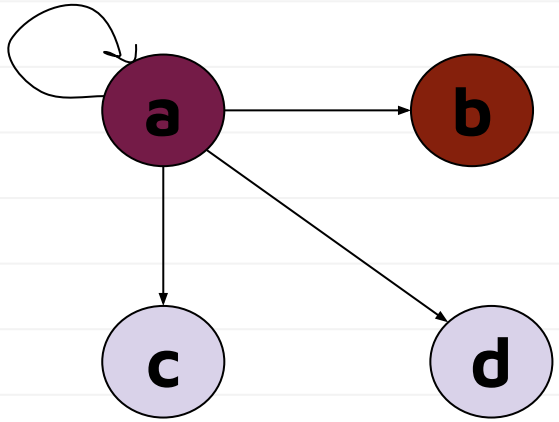
Counting Edges

- What is the **greatest** number edges possible in a **directed** graph with *n* nodes?



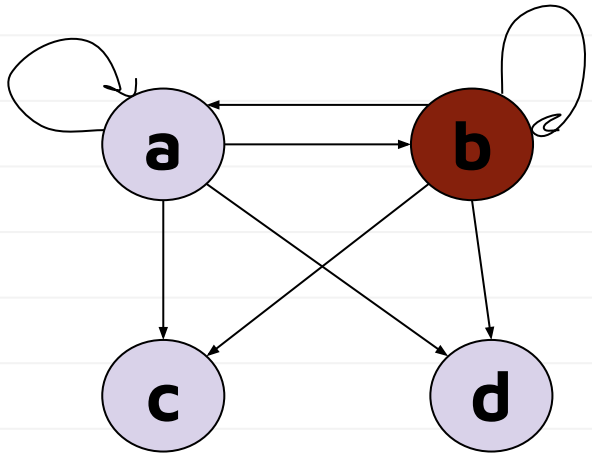
Counting Edges

- What is the **greatest** number edges possible in a **directed** graph with *n* nodes?



Counting Edges

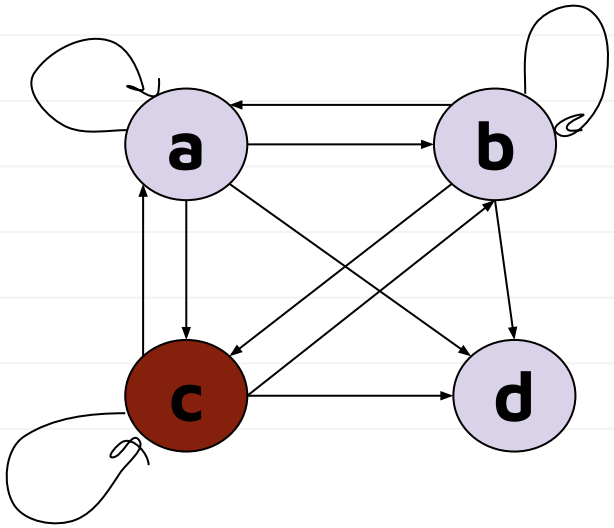
- What is the **greatest** number edges possible in a **directed** graph with *n* nodes?



4 + 4

Counting Edges

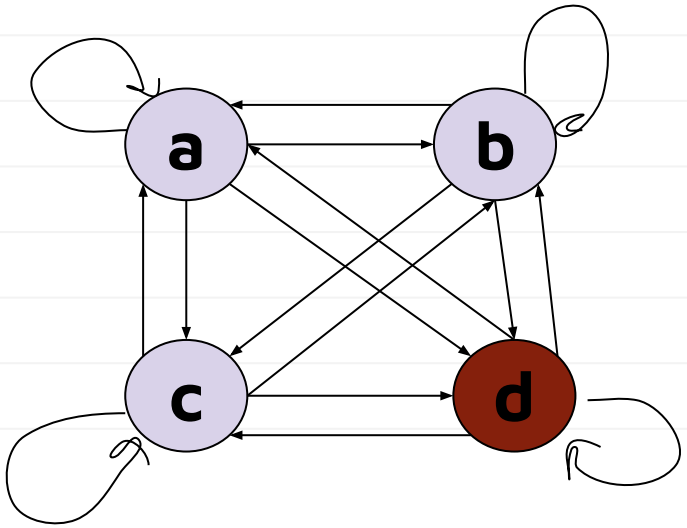
- What is the **greatest** number edges possible in a **directed** graph with *n* nodes?



4 + 4 + 4

Counting Edges

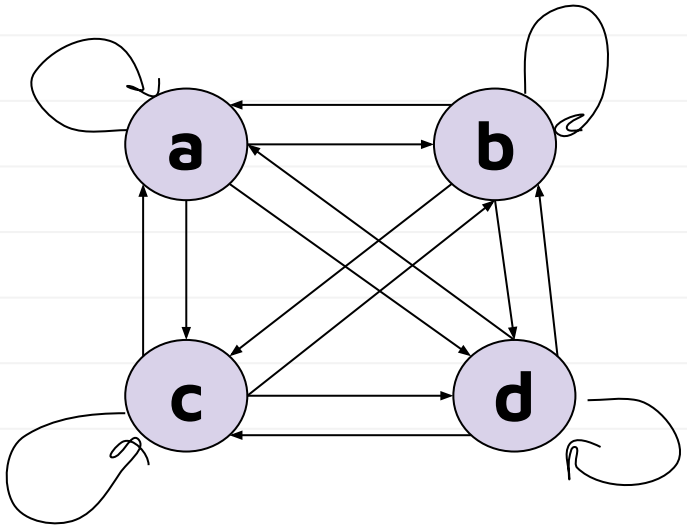
- What is the **greatest** number edges possible in a **directed** graph with *n* nodes?



4 + 4 + 4 + 4

Counting Edges

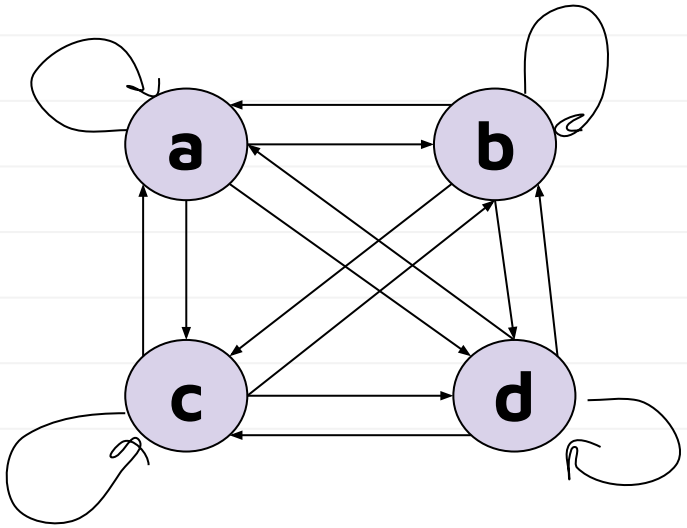
- What is the **greatest** number edges possible in a **directed** graph with *n* nodes?



$4 + 4 + 4 + 4 = 16 = 4^2$

Counting Edges

- What is the **greatest** number edges possible in a **directed** graph with *n* nodes?



n^2

$4 + 4 + 4 + 4 = 16 = 4^2$

Exercise

What is the greatest number edges possible in an **undirected** graph with n nodes?

A: n

B: n^2

C: ~$n^2/2$

D: n choose 2

E: n^2 choose 2

Counting Edges

- What is the **greatest** number edges possible in an **undirected** graph with *n* nodes?

a

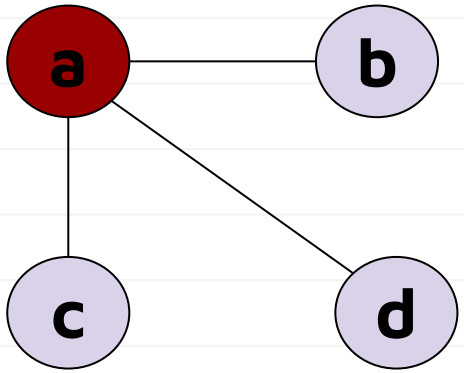
b

c

d

Counting Edges

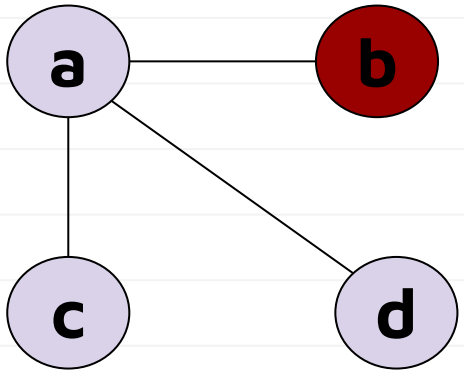
- What is the **greatest** number edges possible in an **undirected** graph with *n* nodes?



3

Counting Edges

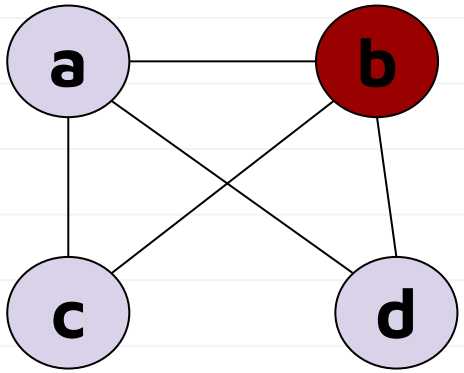
- What is the **greatest** number edges possible in an **undirected** graph with *n* nodes?



3

Counting Edges

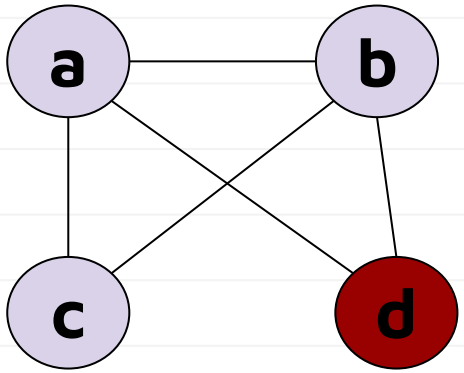
- What is the **greatest** number edges possible in an **undirected** graph with *n* nodes?



3 + 2

Counting Edges

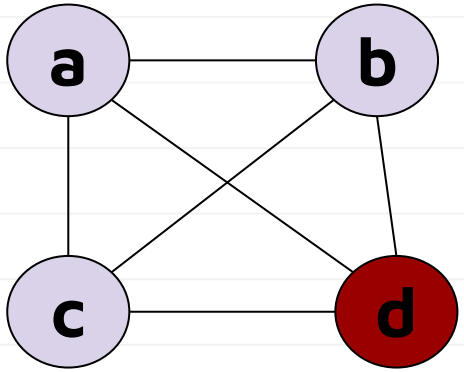
- What is the **greatest** number edges possible in an **undirected** graph with *n* nodes?



3 + 2

Counting Edges

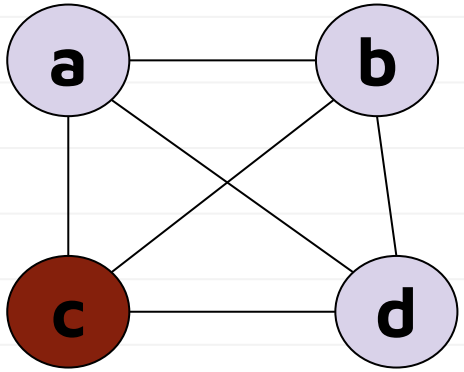
- What is the **greatest** number edges possible in an **undirected** graph with *n* nodes?



3 + 2 + 1

Counting Edges

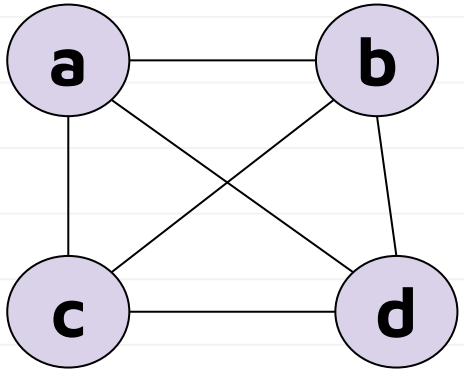
- What is the **greatest** number edges possible in an **undirected** graph with *n* nodes?



3 + 2 + 1 + 0

Counting Edges

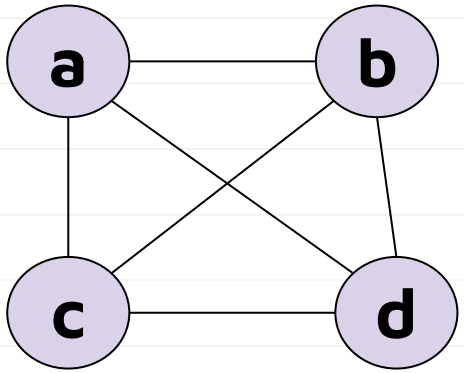
- What is the **greatest** number edges possible in an **undirected** graph with *n* nodes?



3 + 2 + 1 + 0 = 6

Counting Edges

- What is the **greatest** number edges possible in an **undirected** graph with *n* nodes?



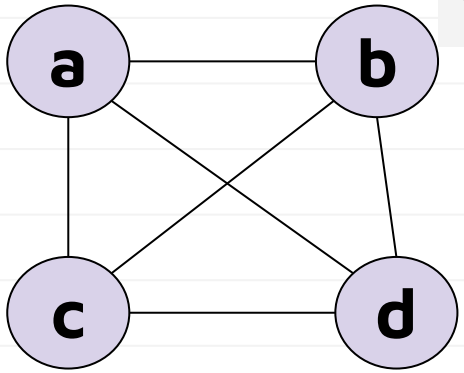
(n-1) + (n-2) + ... + 1 = ?

3 + 2 + 1 + 0 = 6

Counting Edges

- What is the **greatest** number edges possible in an **undirected** graph with *n* nodes?

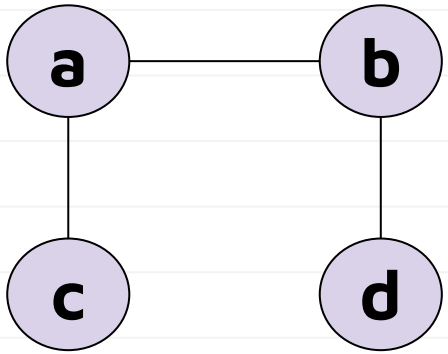
(n-1) + (n-2) + ... + 1 = **n(n-1)/2**



3 + 2 + 1 + 0 = 6

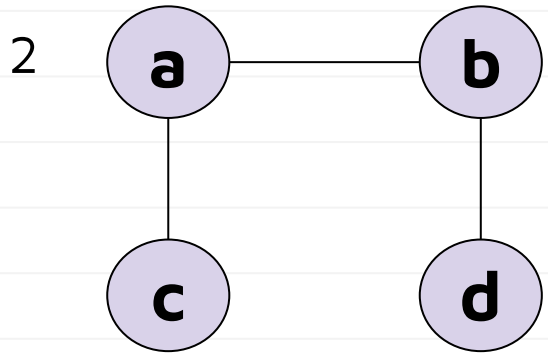
Degree

The degree of a node in an *undirected* graph is the **number of edges containing that node.**



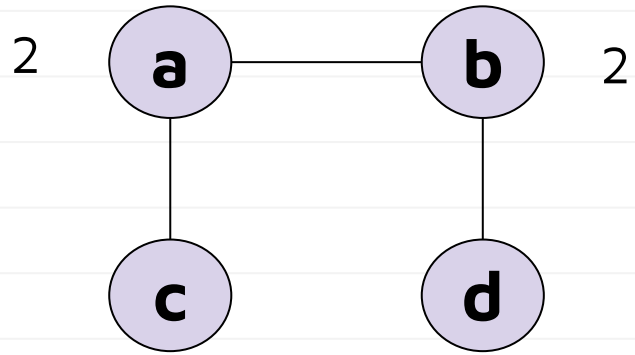
Degree

The degree of a node in an *undirected* graph is the **number of edges containing that node.**



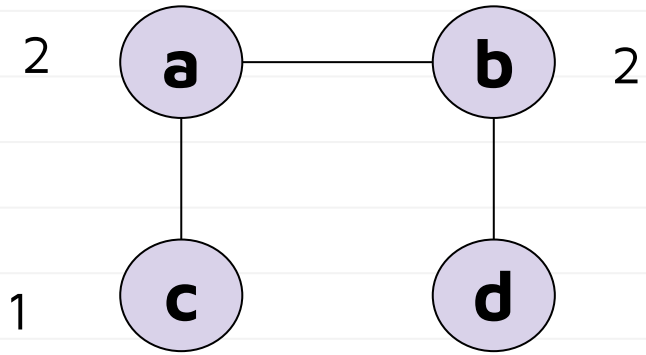
Degree

The degree of a node in an *undirected* graph is the **number of edges containing that node.**



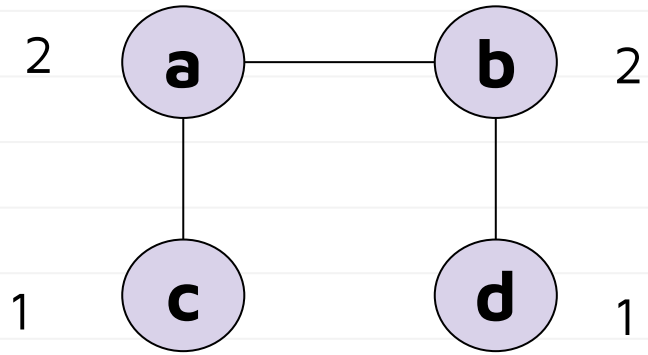
Degree

The degree of a node in an *undirected* graph is the **number of edges containing that node.**

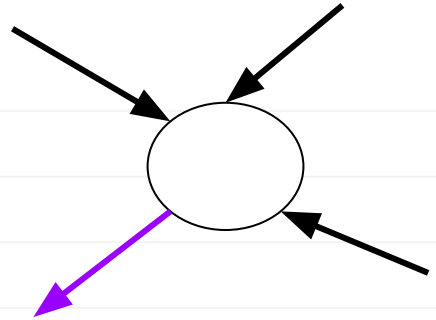


Degree

The degree of a node in an *undirected* graph is the **number of edges containing that node.**

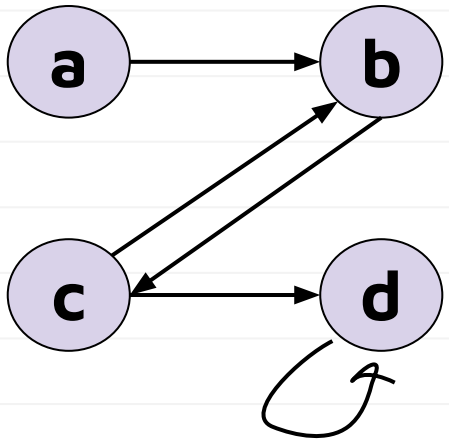


In-Degree/Out-Degree



- The in-degree of a node in an directed graph is the number of edges **entering** that node. (3 for the given node above)
- The out-degree of a node in an directed graph is the number of edges leaving that node. (1 for the given node above)
- The degree of a node in a directed graph is the *in-degree + out-degree.* (4 for the given node above)

Examples

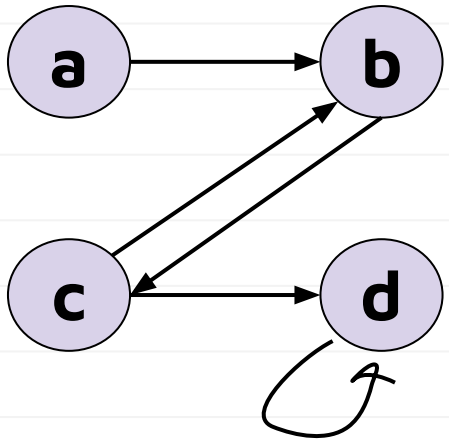


What are different degrees for node d?

	Out-degree,	in-degree,	degree
A:	**1**	**3**	**4**
B:	**3**	**1**	**2**
C:	**2**	**2**	**3**
D:	**1**	**2**	**2**

E: None of the above

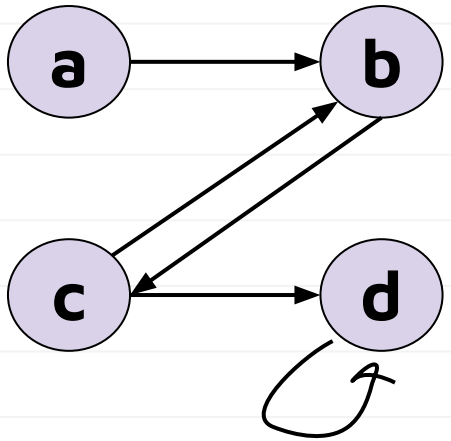
Examples



Out: 1
In: 2
D: 3

Examples

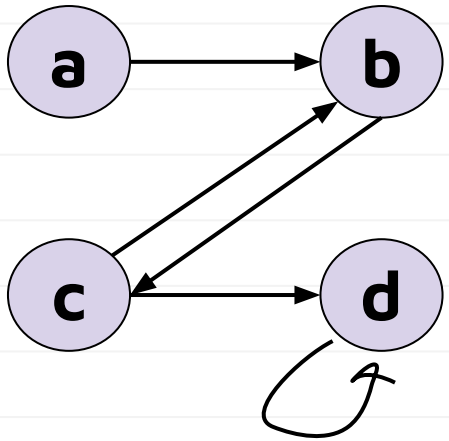
Out:
In:
D:



Out: 1
In: 2
D: 3

Examples

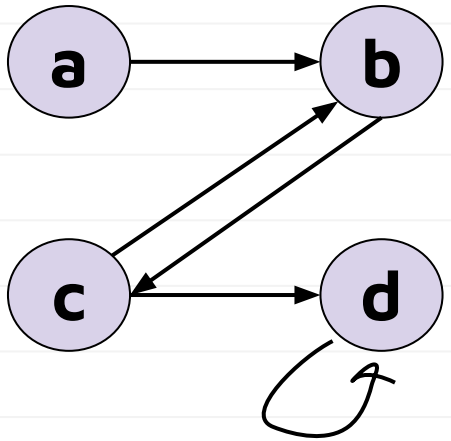
Out: 1
In: 0
D: 1



Out: 1
In: 2
D: 3

Examples

Out: 1
In: 0
D: 1

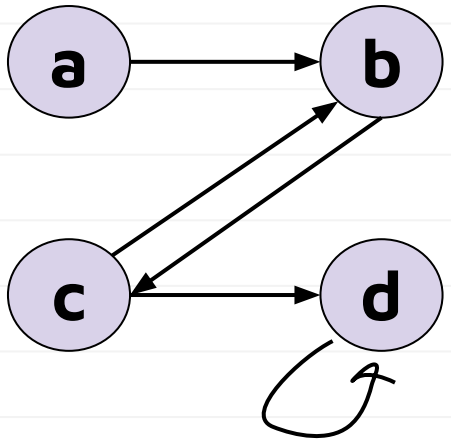


Out:
In:
D:

Out: 1
In: 2
D: 3

Examples

Out: 1
In: 0
D: 1

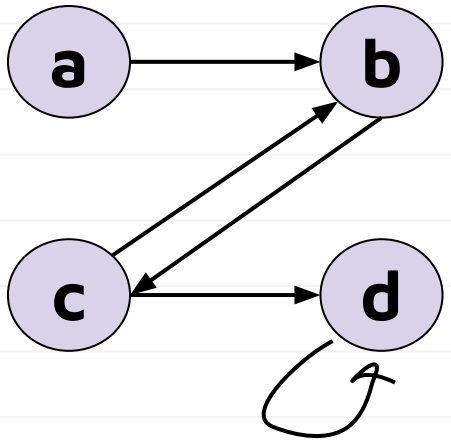


Out: 1
In: 2
D: 3

Out: 1
In: 2
D: 3

Examples

Out: 1
In: 0
D: 1



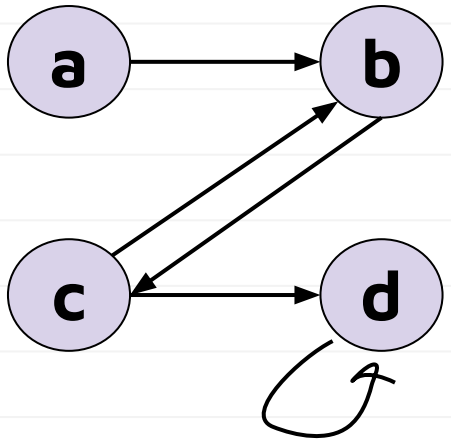
Out: 1
In: 2
D: 3

Out:
In:
D:

Out: 1
In: 2
D: 3

Examples

Out: 1
In: 0
D: 1



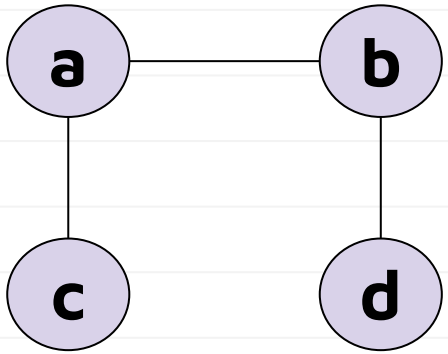
Out: 1
In: 2
D: 3

Out: 2
In: 1
D: 3

Out: 1
In: 2
D: 3

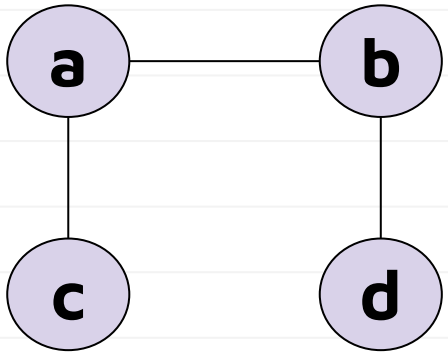
Neighbors

Definition: in an *undirected* graph, the set of neighbors of a node u is the set of all nodes which *share* an edge with u.



Neighbors

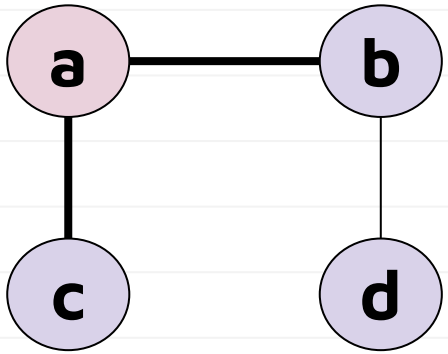
Definition: in an *undirected* graph, the set of neighbors of a node u is the set of all nodes which *share* an edge with u.



neighbors(a) = { }

Neighbors

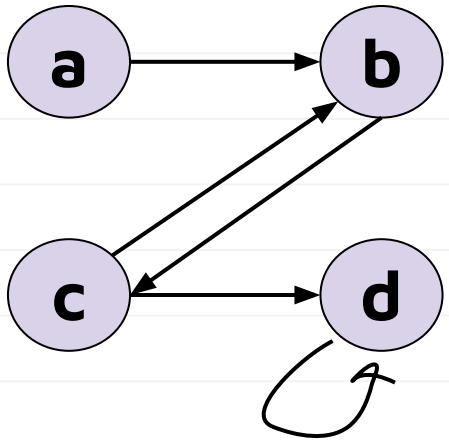
Definition: in an *undirected* graph, the set of neighbors of a node u is the set of all nodes which ***share*** an edge with u.



neighbors(a) = {b, c}

Predecessors

Definition: in an *directed* graph, the set of predecessors of a node u is the set of all nodes which are at the **start** of an edge **entering** u.



predecessors(b)

A: {c}

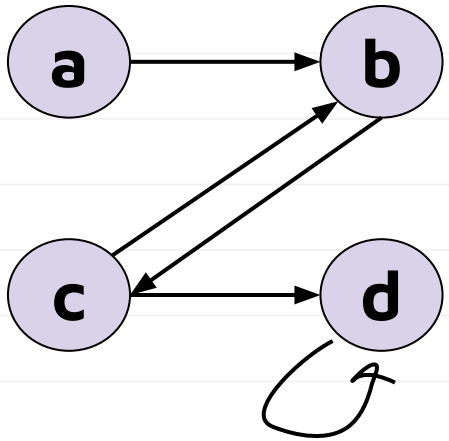
B: {a}

C: {a, c}

D: {a, b, c}

Predecessors

Definition: in an *directed* graph, the set of predecessors of a node u is the set of all nodes which are at the **start** of an edge **entering** u.



predecessors(b)

A: {c}

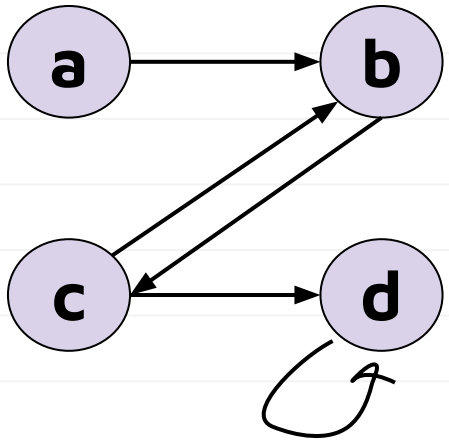
B: {a}

C: {a, c}

D: {a, b, c}

Predecessors

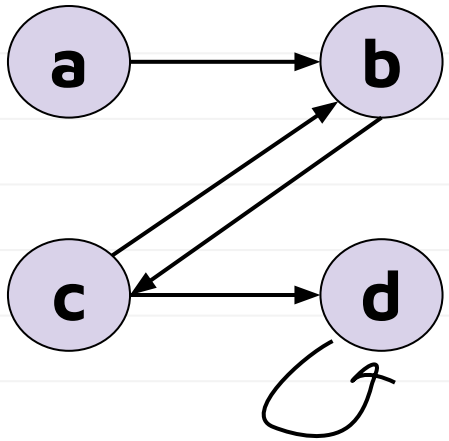
Definition: in an *directed* graph, the set of predecessors of a node u is the set of all nodes which are at the **start** of an edge **entering** u.



predecessors(a) = { }

Predecessors

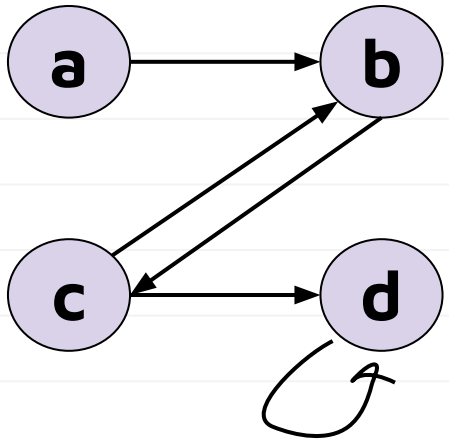
Definition: in an *directed* graph, the set of predecessors of a node u is the set of all nodes which are at the **start** of an edge **entering** u.



predecessors(d) = { ? }

Predecessors

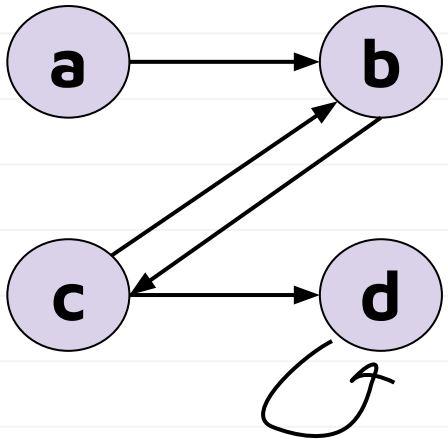
Definition: in an *directed* graph, the set of predecessors of a node u is the set of all nodes which are at the **start** of an edge **entering** u.



predecessors(d) = {c, d}

Successors

Definition: in an *directed* graph, the set of *successors* of a node u is the set of all nodes which are at the **end** of an edge **leaving** u.



***successors*(c)**

A: {a, b, c}

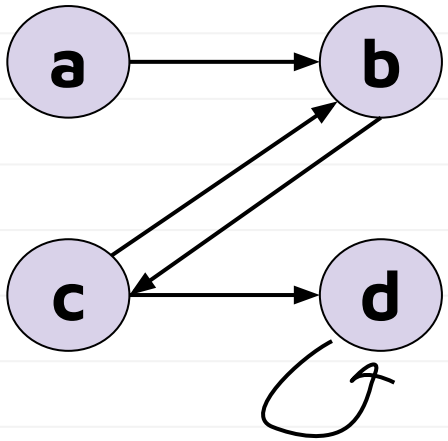
B: {c}

C: {b, d}

D: {a, b, d}

Successors

Definition: in an *directed* graph, the set of *successors* of a node u is the set of all nodes which are at the **end** of an edge **leaving** u.



***successors*(c)**

A: {a, b, c}

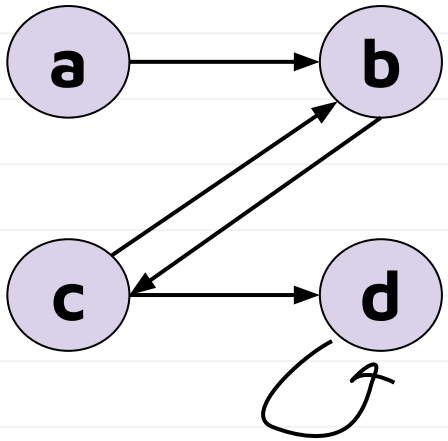
B: {c}

C: {b, d}

D: {a, b, d}

Successors

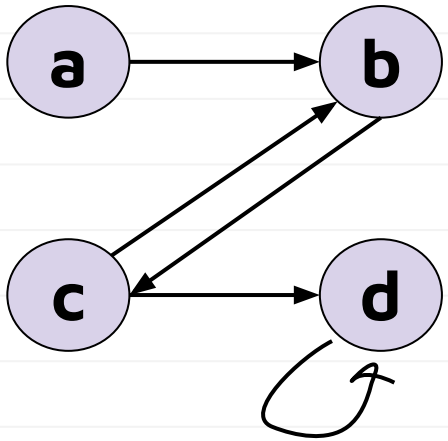
Definition: in an *directed* graph, the set of *successors* of a node u is the set of all nodes which are at the **end** of an edge **leaving** u.



***successors*(a) = ?**

Successors

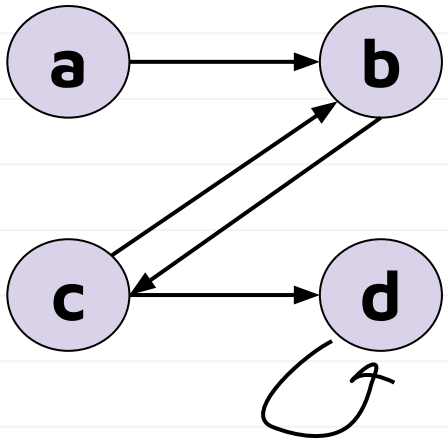
Definition: in an *directed* graph, the set of *successors* of a node u is the set of all nodes which are at the **end** of an edge **leaving** u.



***successors*(a) = {b}**

Successors

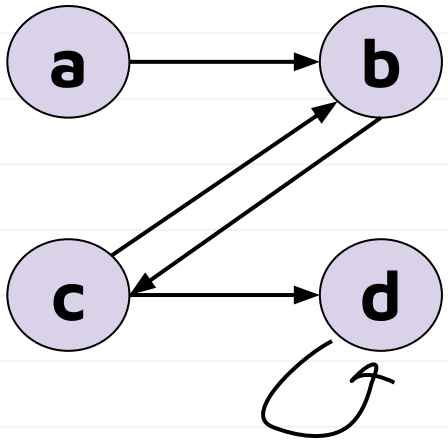
Definition: in an *directed* graph, the set of *successors* of a node u is the set of all nodes which are at the **end** of an edge **leaving** u.



***successors*(b) = {?}**

Successors

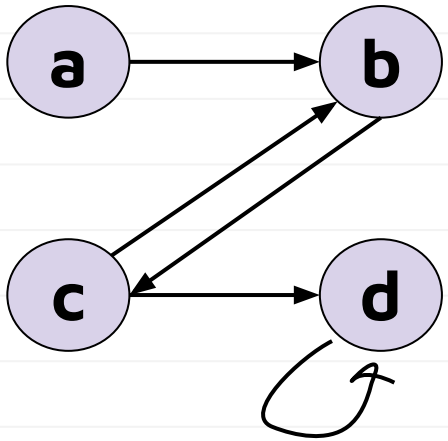
Definition: in an *directed* graph, the set of *successors* of a node u is the set of all nodes which are at the **end** of an edge **leaving** u.



***successors*(b) = {c}**

Successors

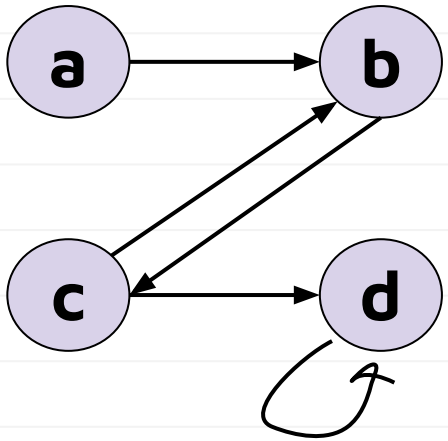
Definition: in an *directed* graph, the set of *successors* of a node u is the set of all nodes which are at the **end** of an edge **leaving** u.



***successors*(d) = {?}**

Successors

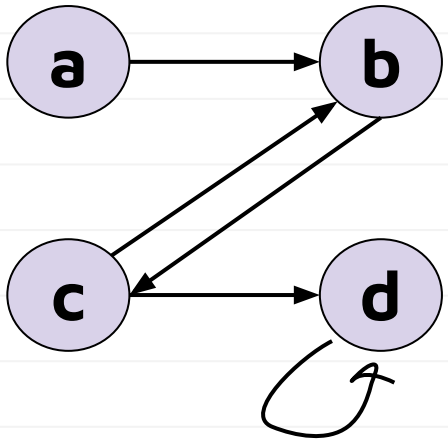
Definition: in an *directed* graph, the set of *successors* of a node u is the set of all nodes which are at the **end** of an edge **leaving** u.



***successors*(d) = {d}**

A Convention

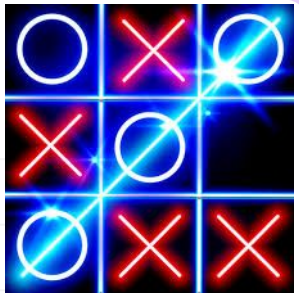
- In a **directed** graph, the **neighbors** of u are the **successors** of u.

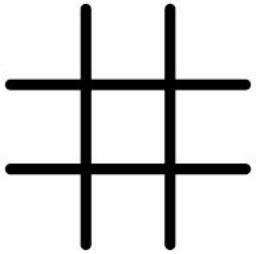


Other Graphs

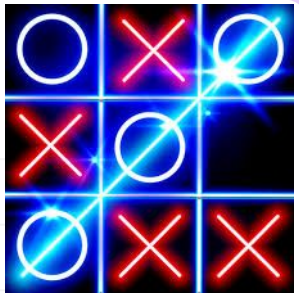
- Graphs can be used to represent states of a process, system, game, etc.
- They could (in principle) have infinitely-many nodes and edges.

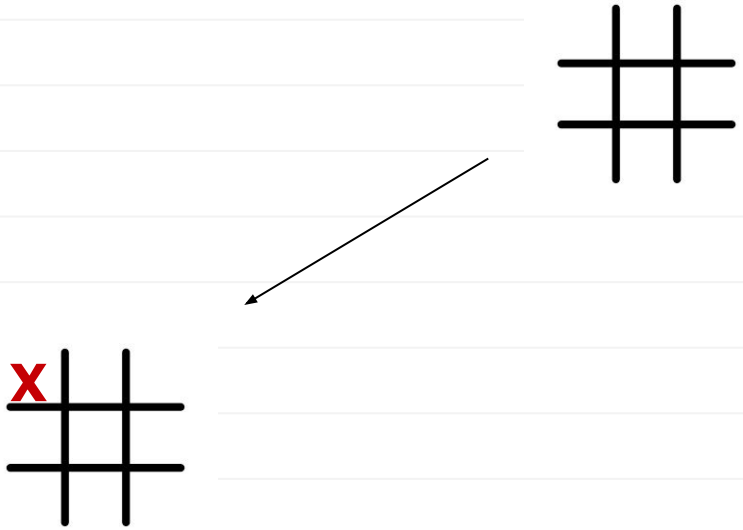
Example: Tic-Tac-Toe



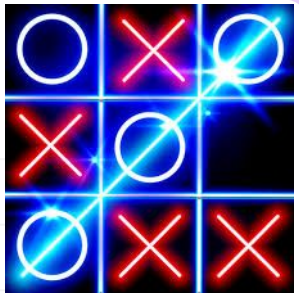


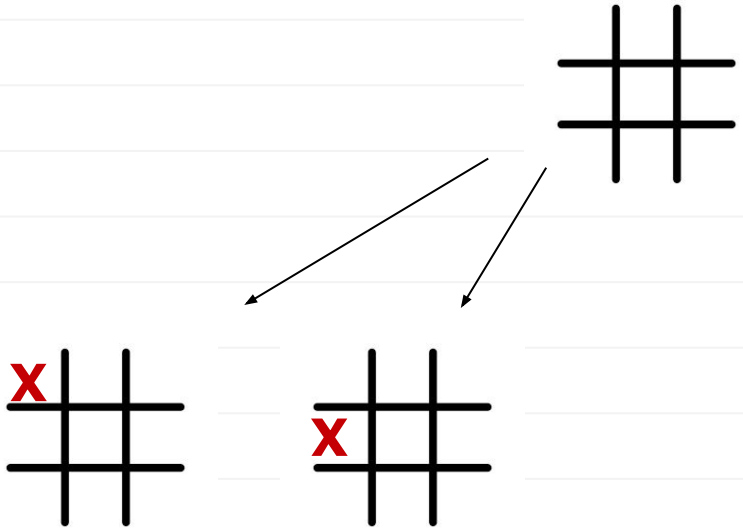
Example: Tic-Tac-Toe



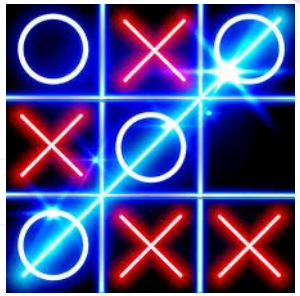


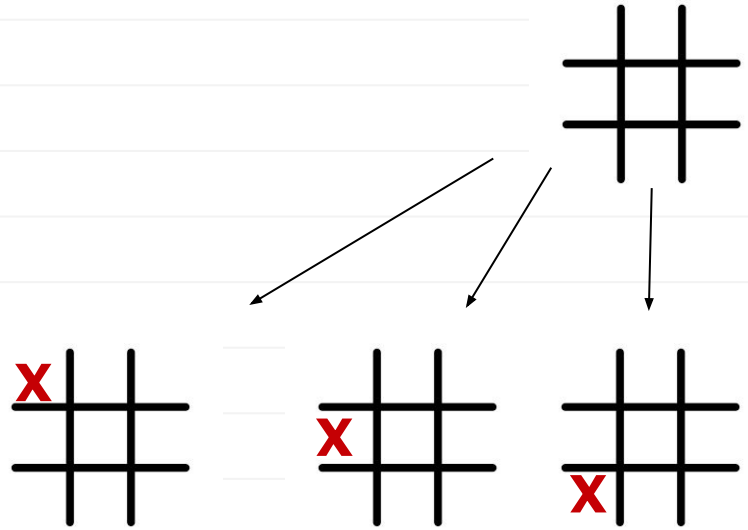
Example: Tic-Tac-Toe



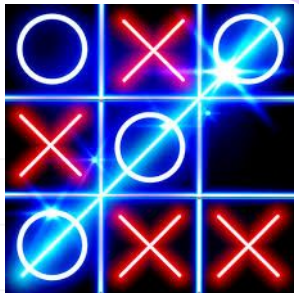


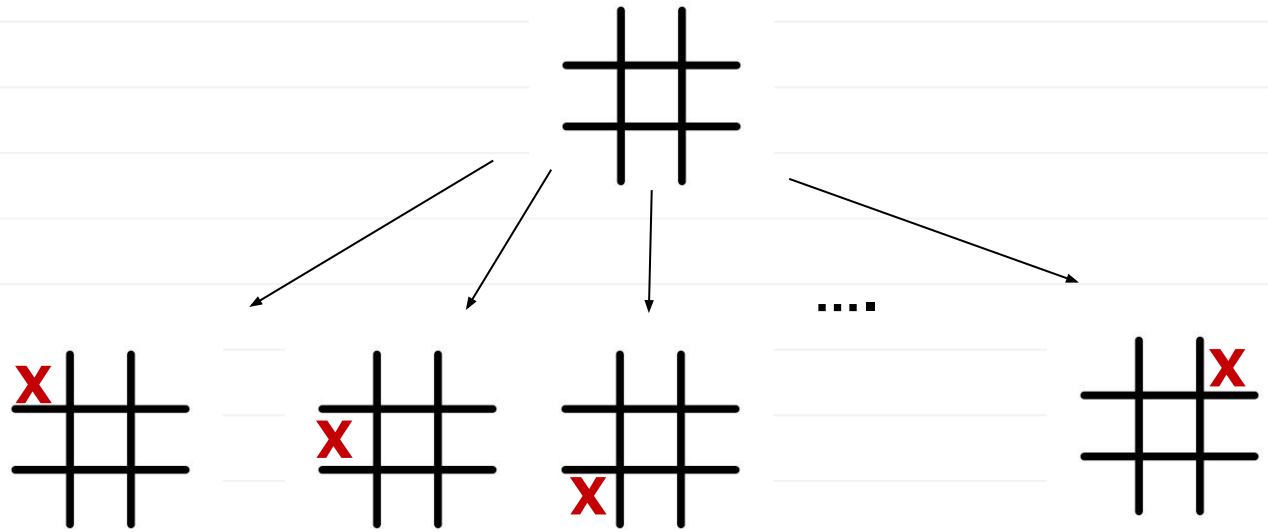
Example: Tic-Tac-Toe



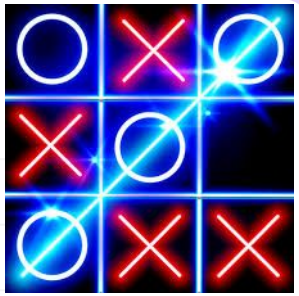


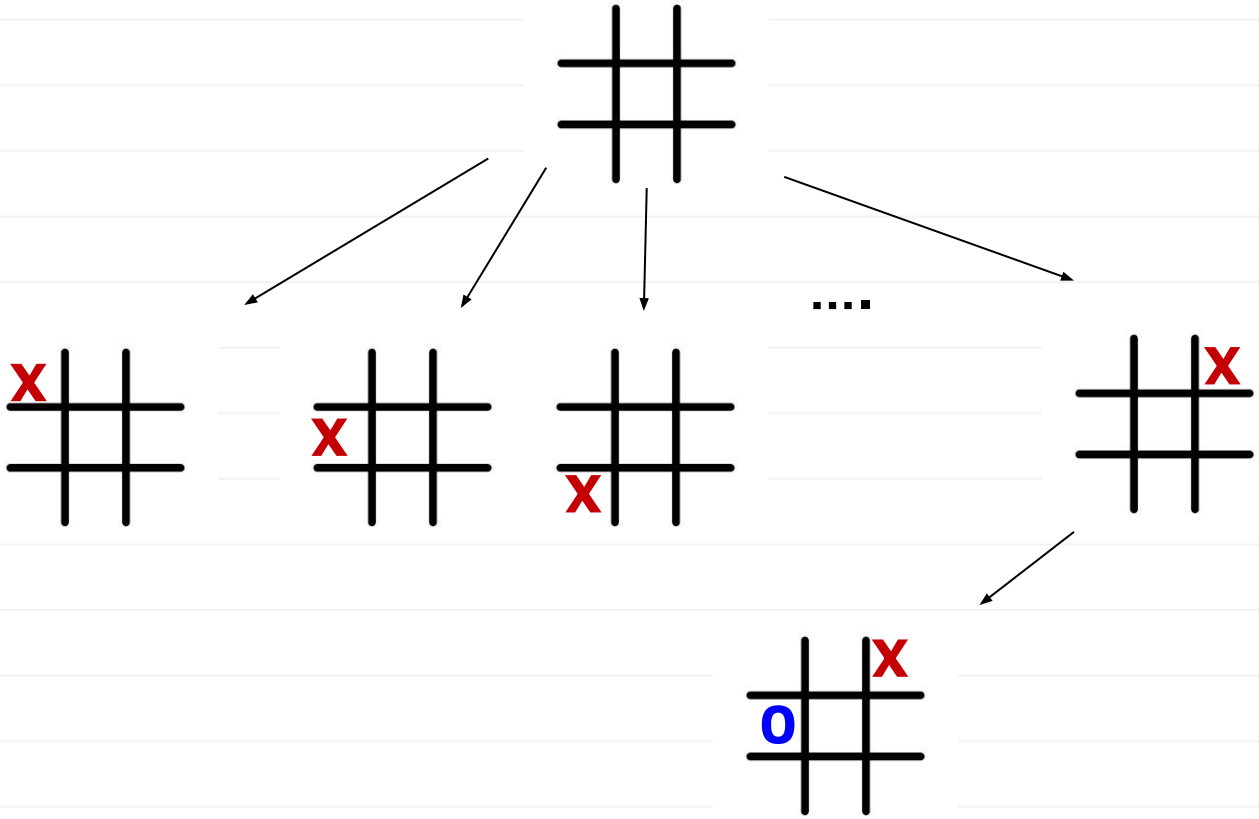
Example: Tic-Tac-Toe



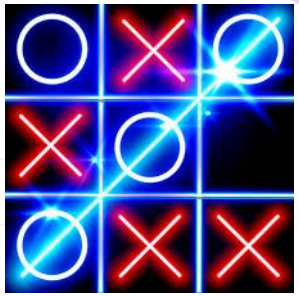


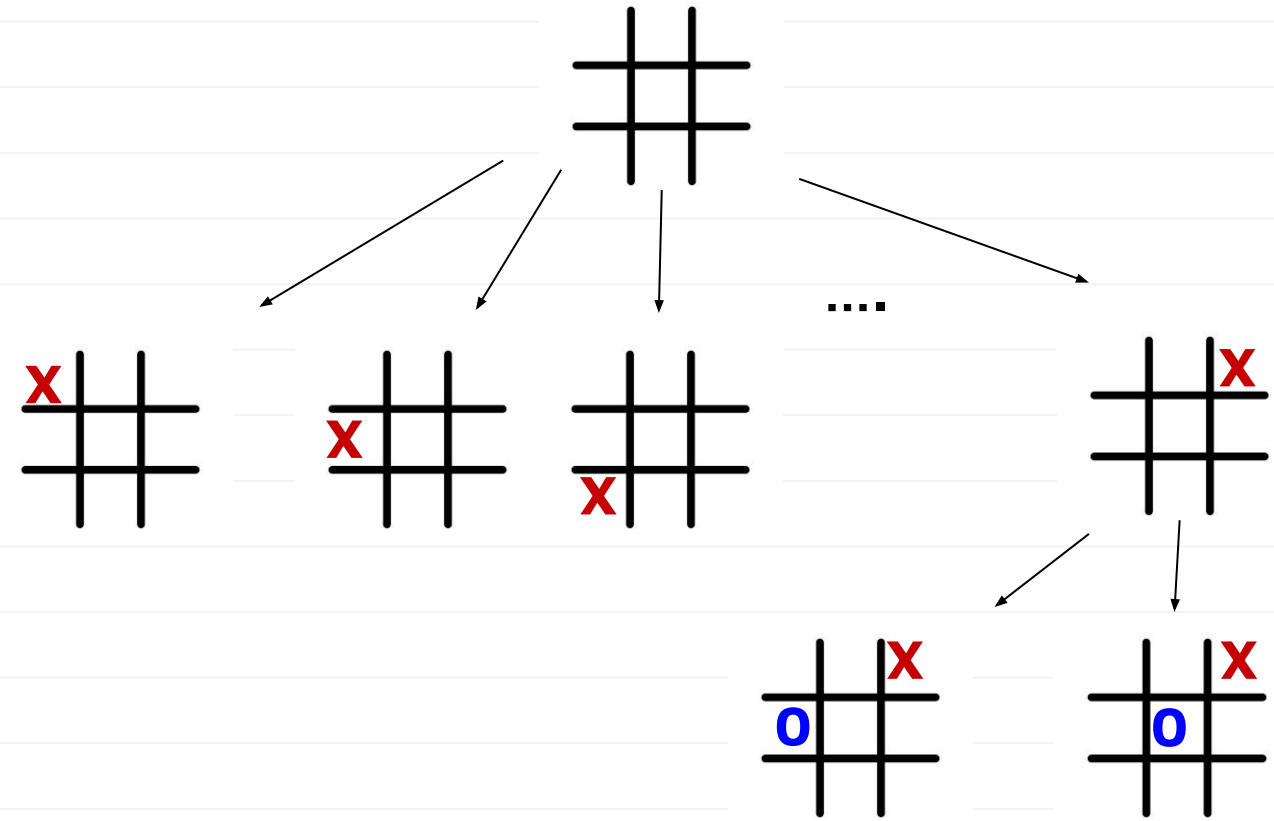
Example: Tic-Tac-Toe



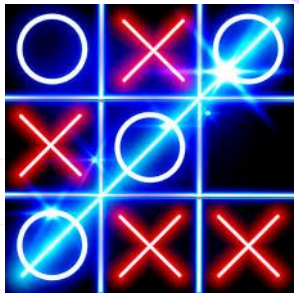


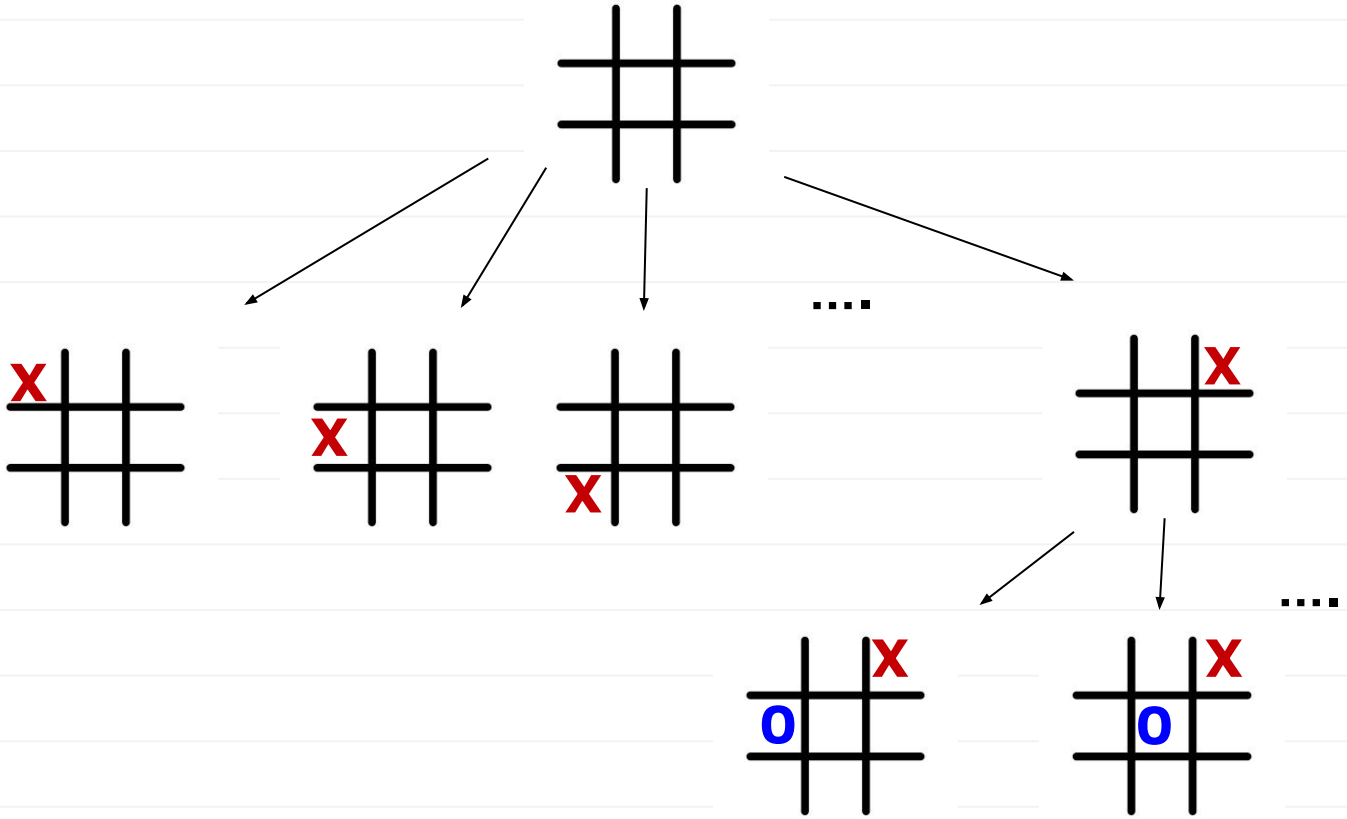
Example: Tic-Tac-Toe



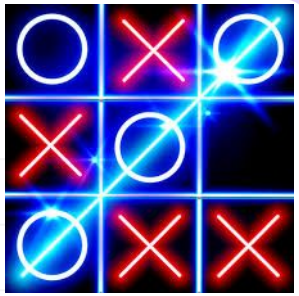


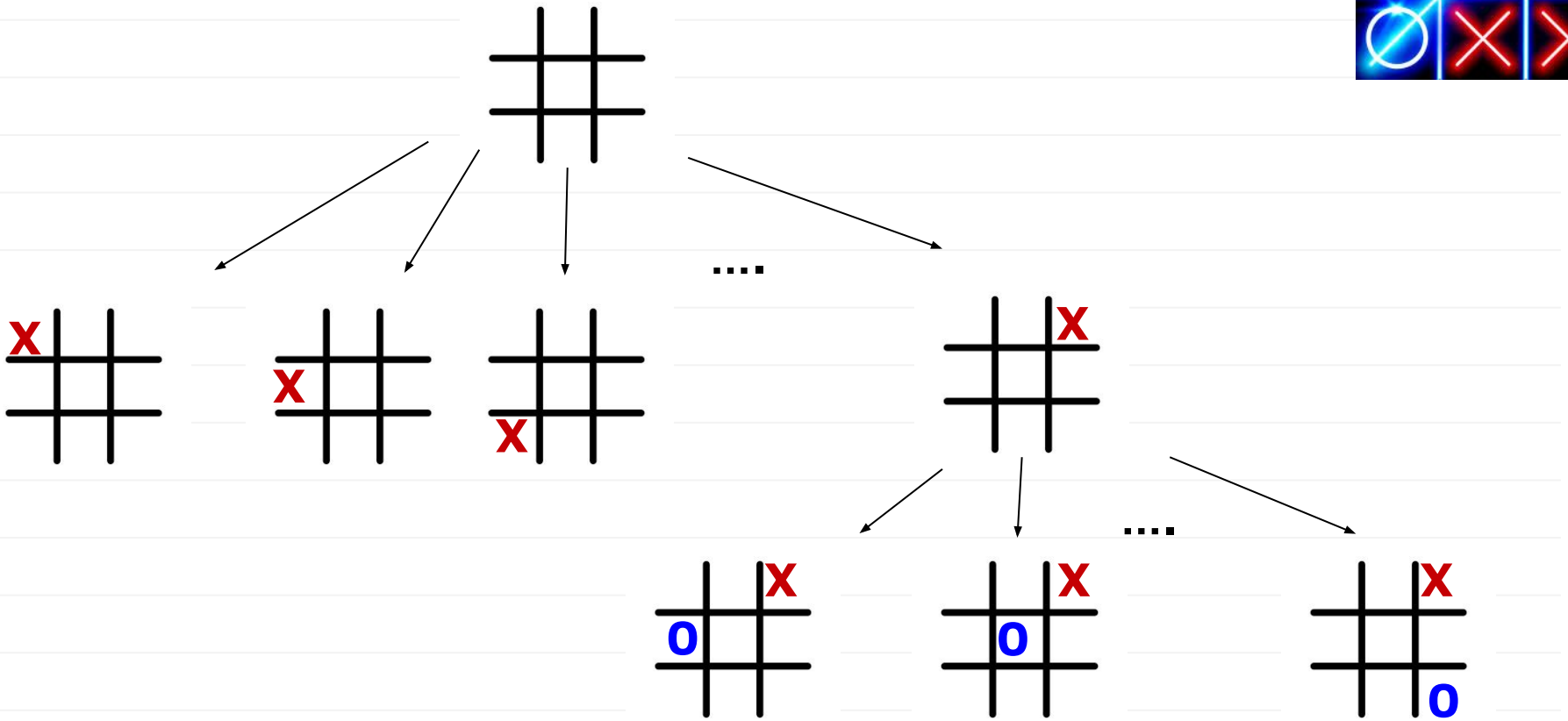
Example: Tic-Tac-Toe



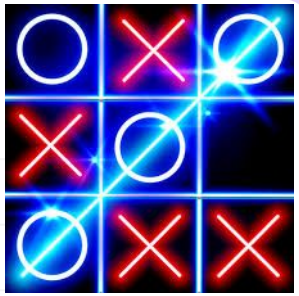


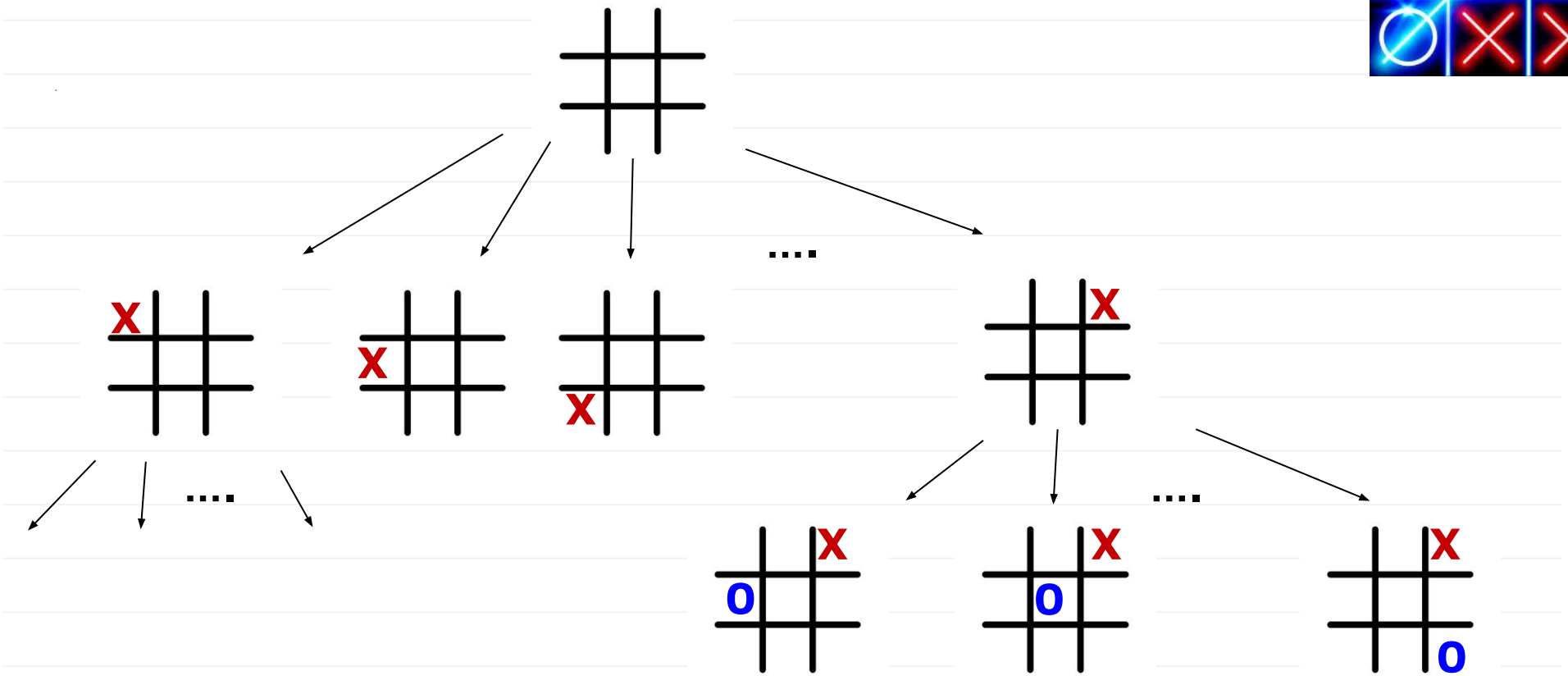
Example: Tic-Tac-Toe



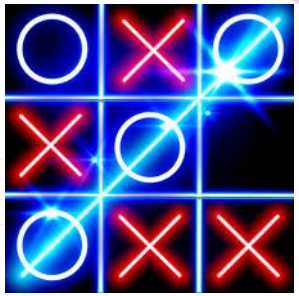


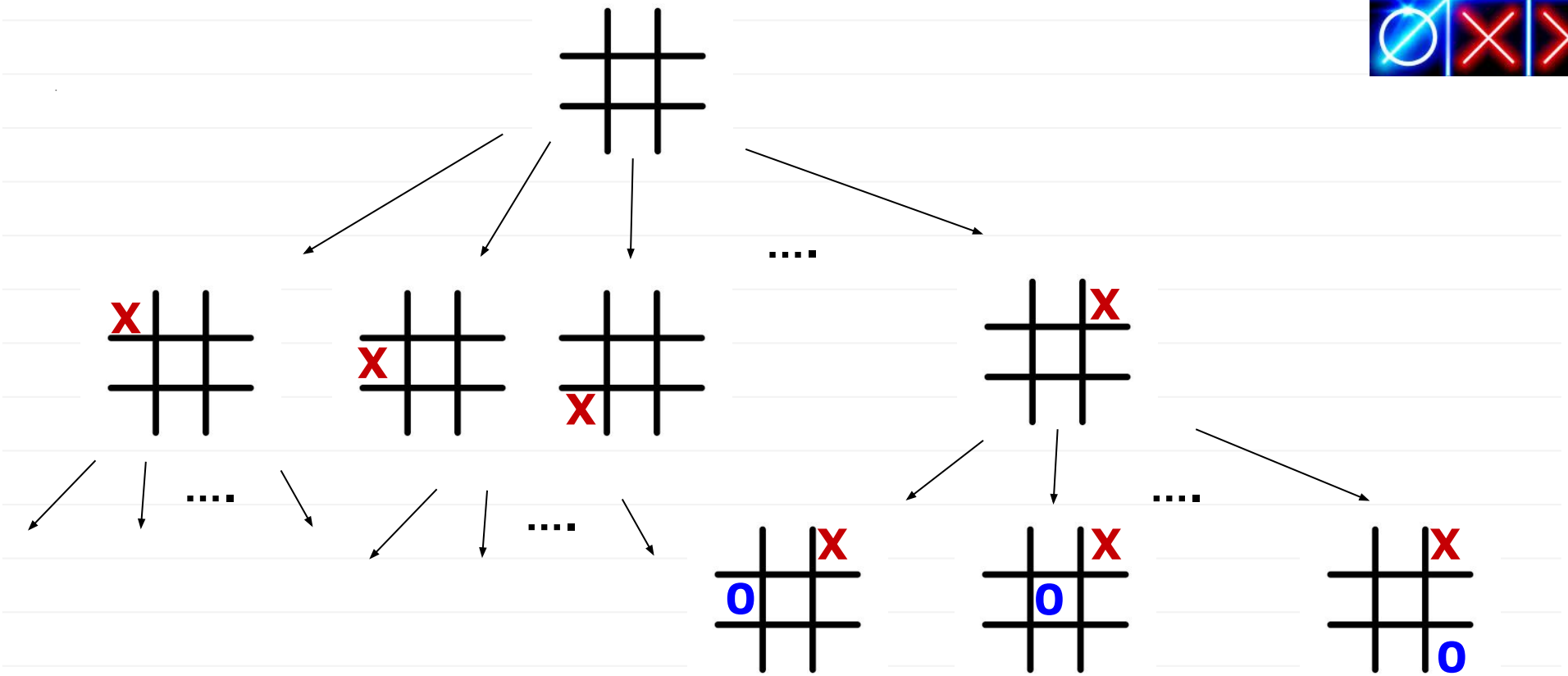
Example: Tic-Tac-Toe



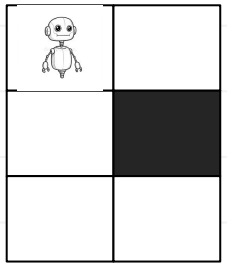


Example: Tic-Tac-Toe

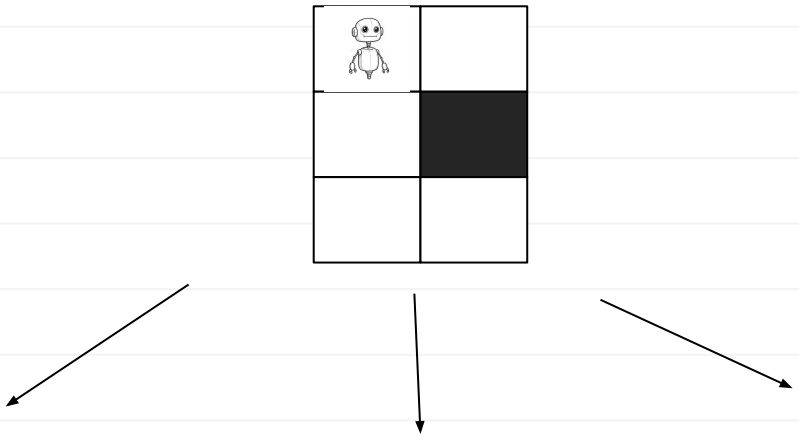




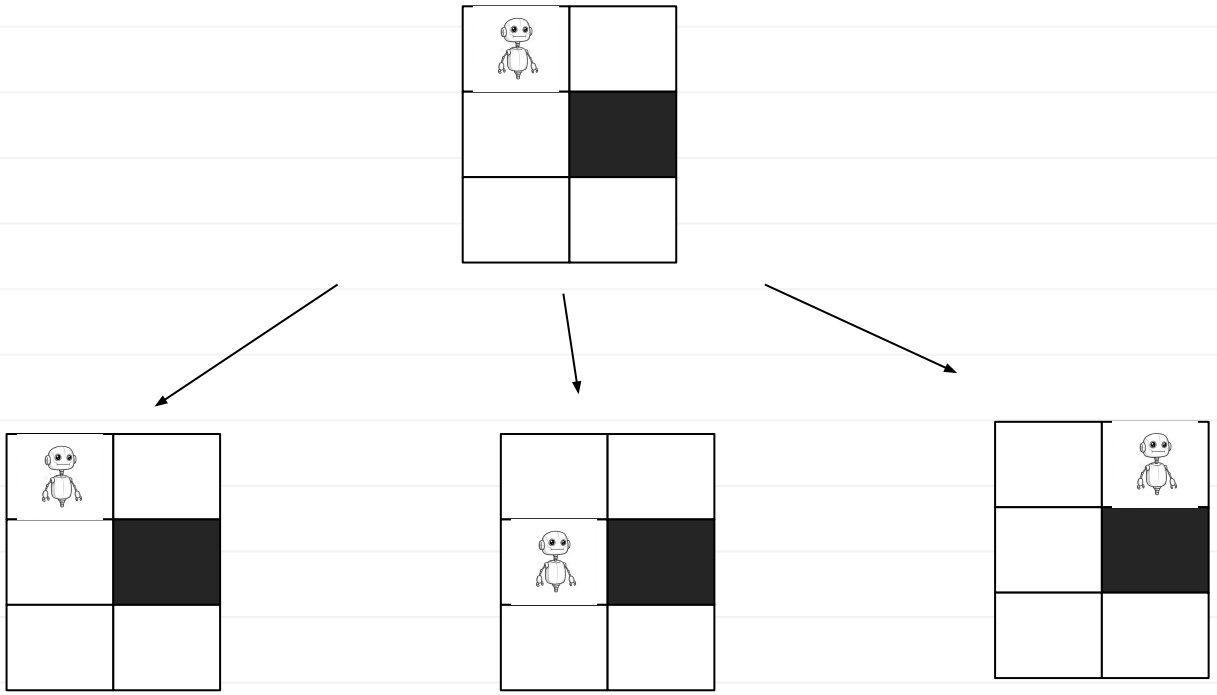
Example: Robot Navigation



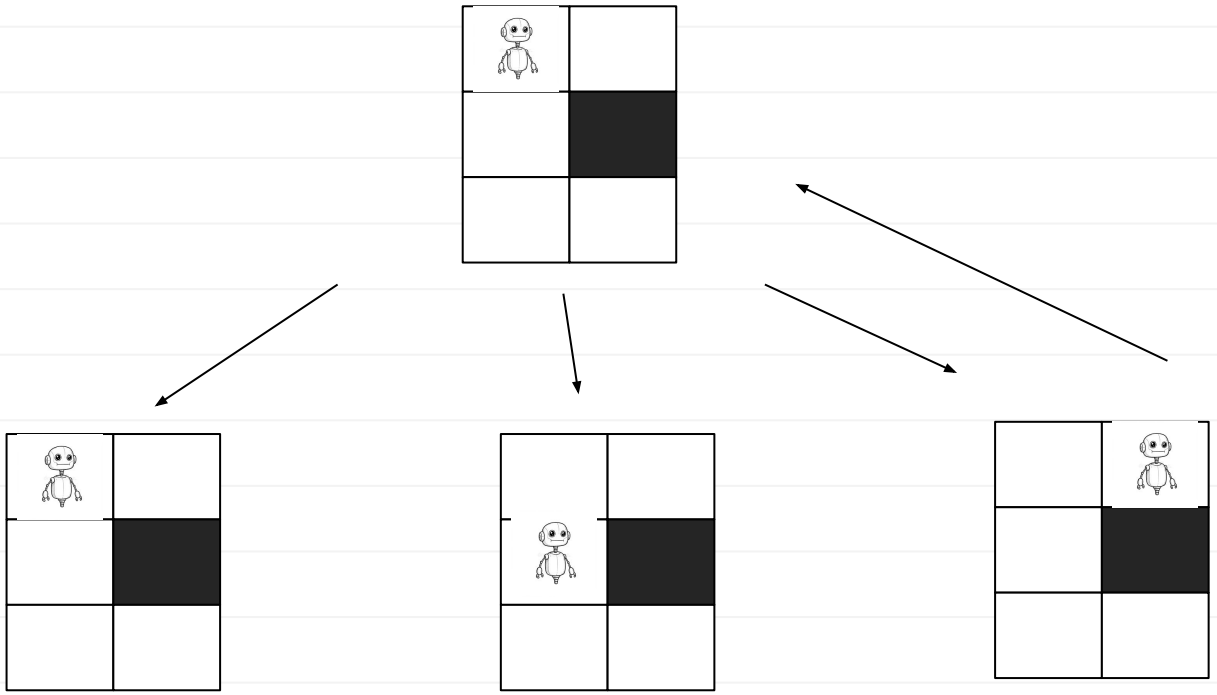
Example: Robot Navigation



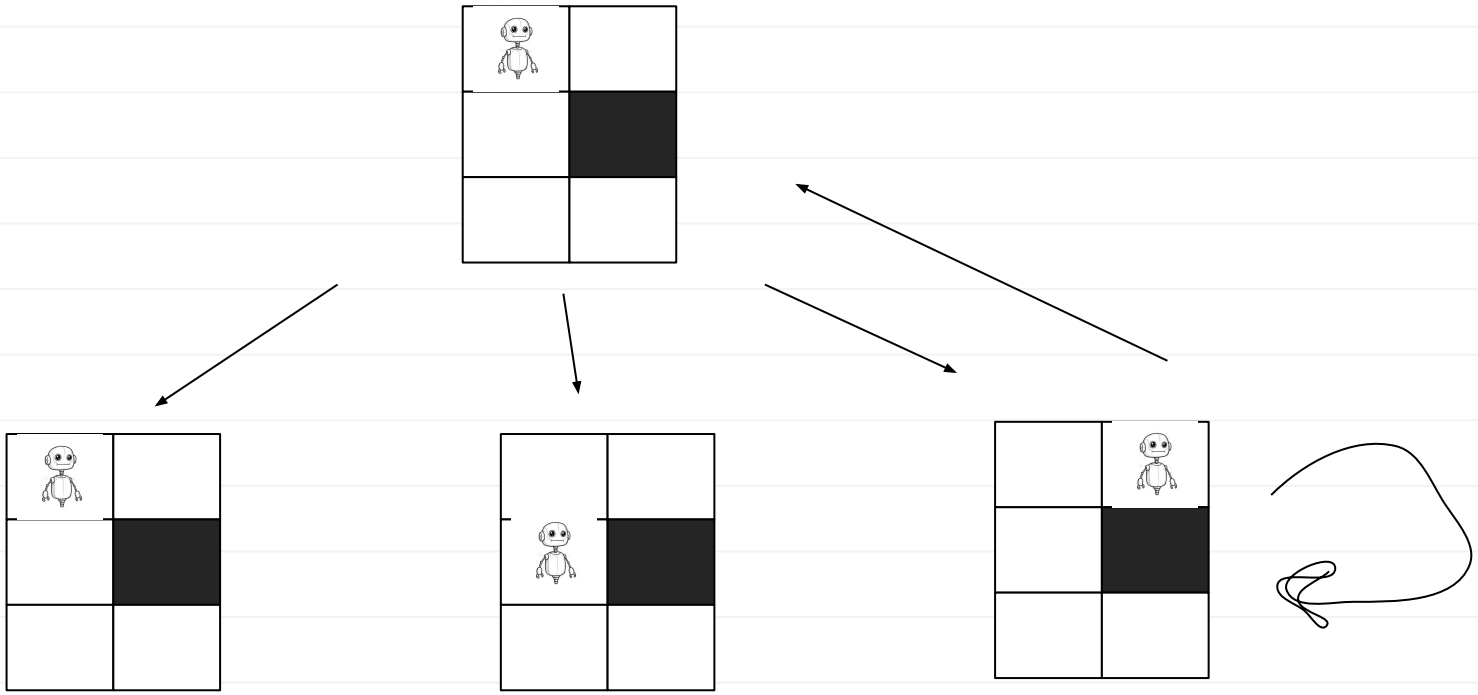
Example: Robot Navigation



Example: Robot Navigation



Example: Robot Navigation

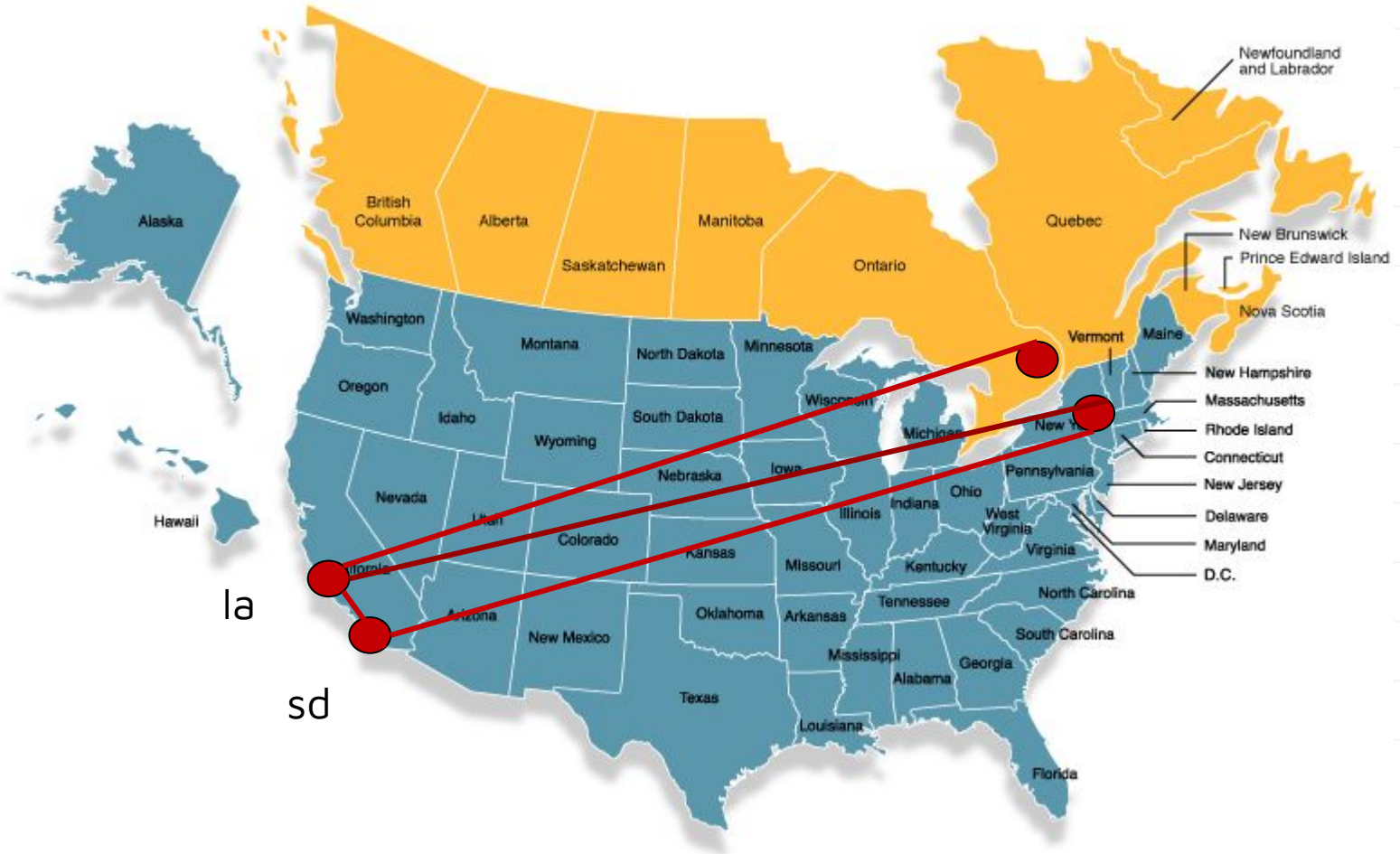


Paths

Example

- Consider a graph of direct flights.
- Each node is an airport.
- Each edge is a direct flight.
- Should the graph be **directed** or **undirected**?

Example



la

sd

Example

- Can we get from San Diego to Ottawa?
- Not with a single edge.
- But with a path.

Definition

A path from u to u' in a (directed or undirected) graph $G = (V, E)$ is a sequence of one or more nodes $u = v_0, v_1, \ldots, v_k = u'$ such that there is an edge between each consecutive pair of nodes in the sequence.

Path Length

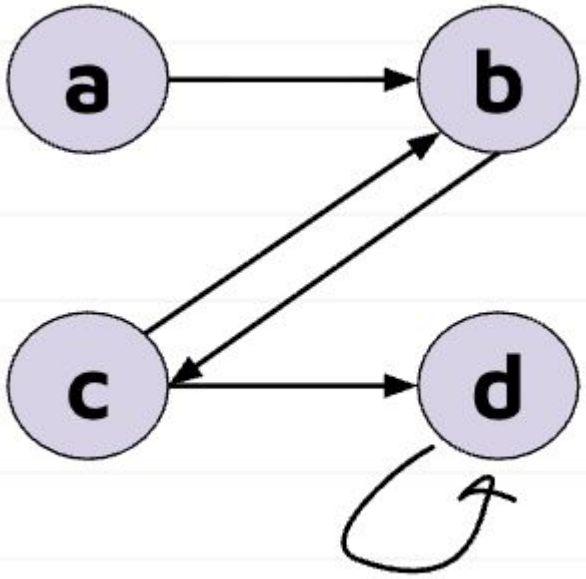
Definition: The length of a path is the number of nodes in the sequence, minus one. Paths of length zero are possible.

Path Length

Definition: The length of a path is the number of nodes in the sequence, minus one. Paths of length zero are possible.

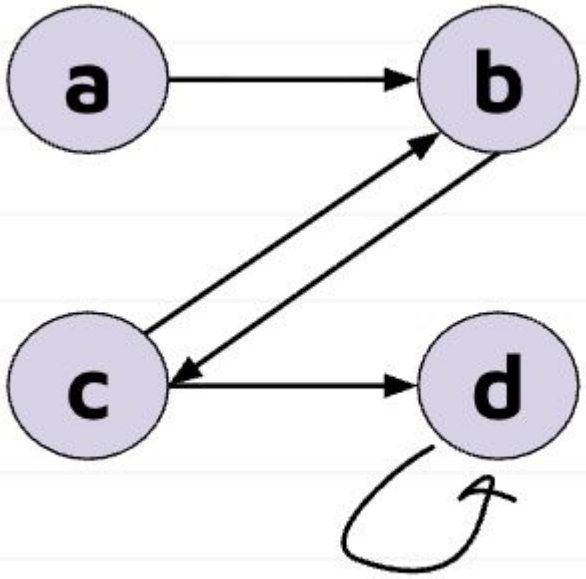
Path: (a, b, c, d).
4 nodes - 1 = 3.

Examples



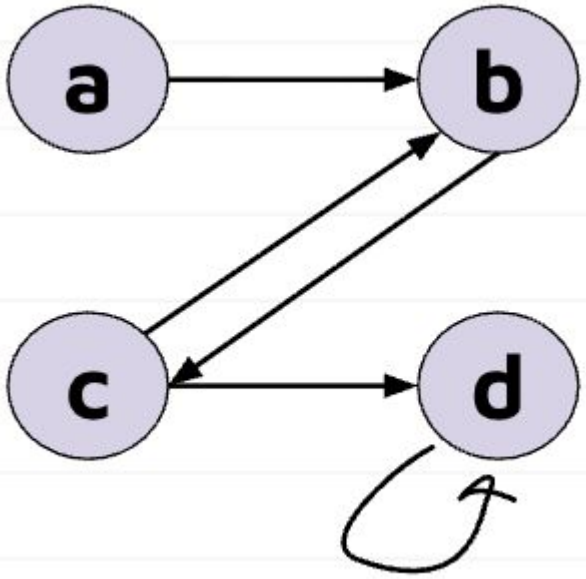
Paths	Not Paths
(a, b, c)	

Examples



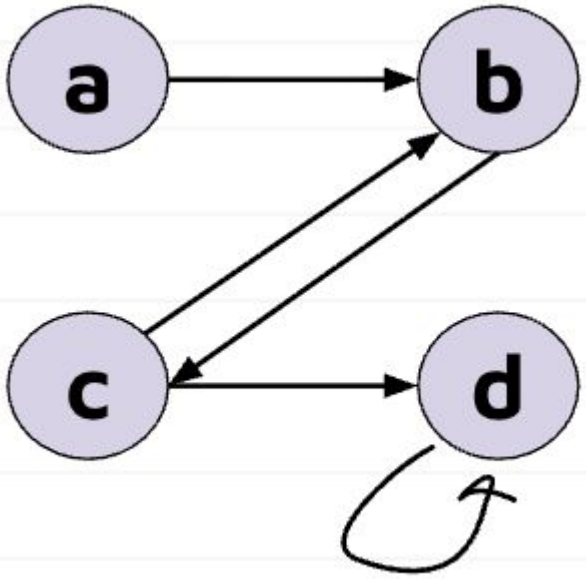
Paths	Not Paths
(a, b, c)	

Examples



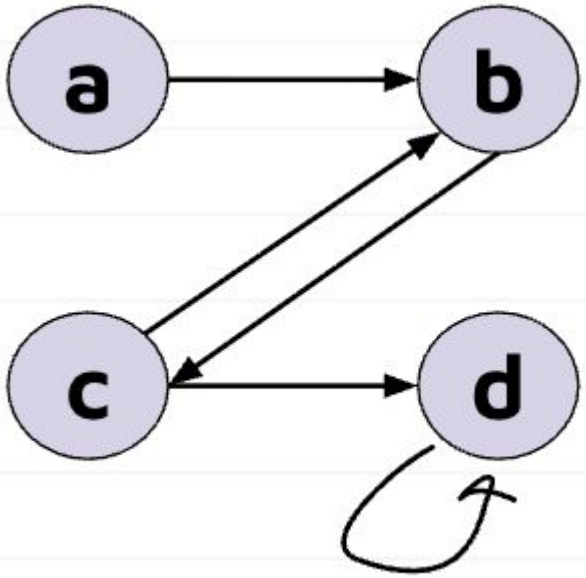
Paths	Not Paths
(a, b, c)	
(a, b, c, b, c)	

Examples



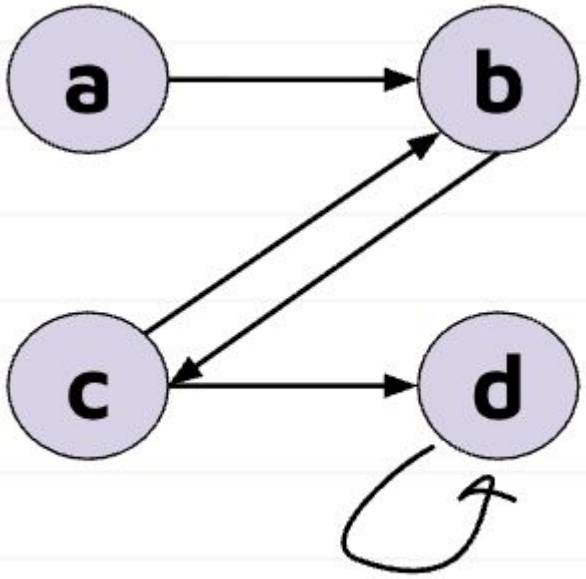
Paths	Not Paths
(a, b, c)	
(a, b, c, b, c)	

Examples



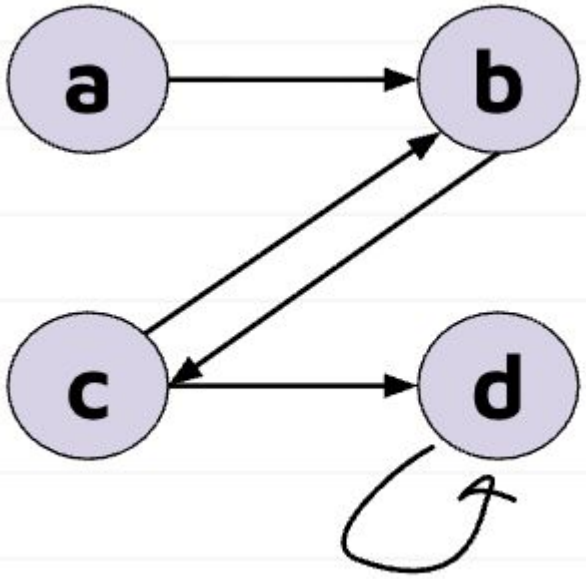
Paths	Not Paths
(a, b, c)	
(a, b, c, b, c)	
(a, b, c, b, c, b, c, b, c)	

Examples



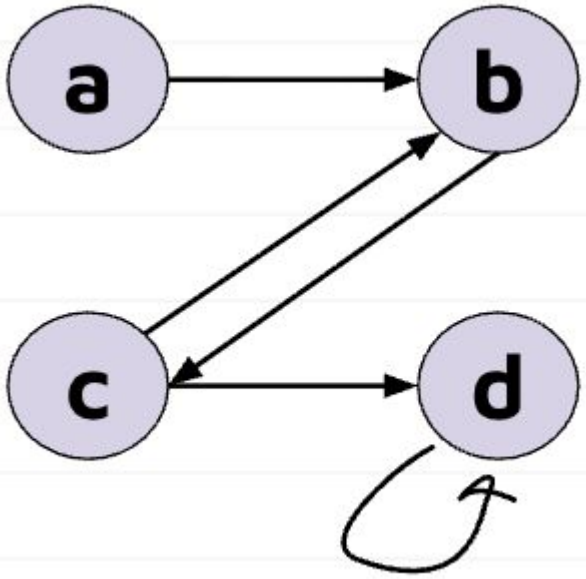
Paths	Not Paths
(a, b, c)	
(a, b, c, b, c)	
(a, b, c, b, c, b, c, b, c)	

Examples



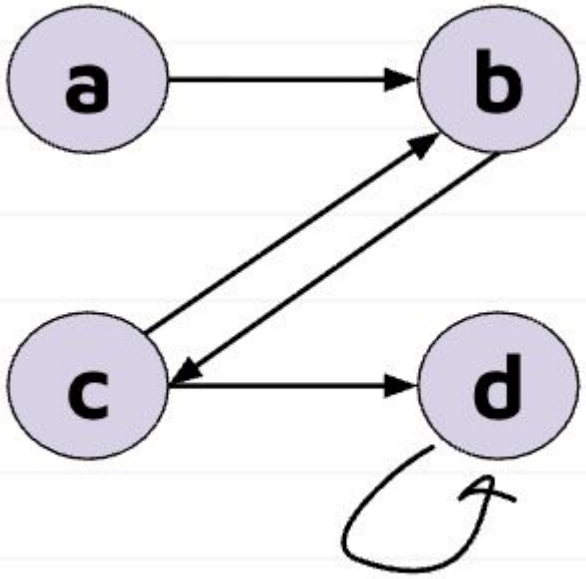
Paths	Not Paths
(a, b, c)	
(a, b, c, b, c)	
(a, b, c, b, c, b, c, b, c)	
(a, b, d)	

Examples



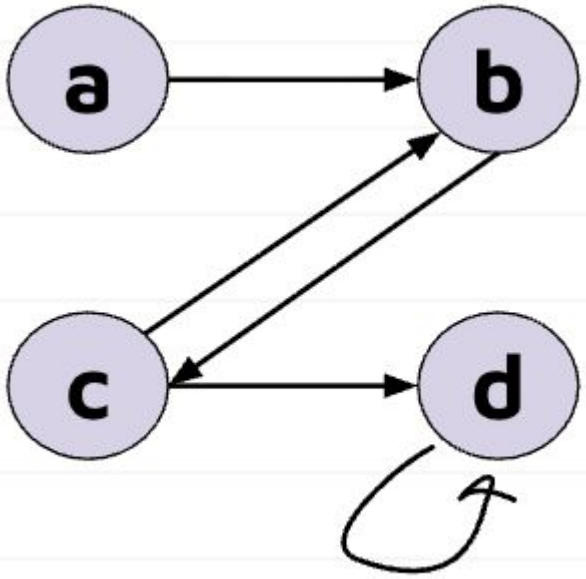
Paths	Not Paths
(a, b, c)	(a, b, d)
(a, b, c, b, c)	
(a, b, c, b, c, b, c, b, c)	

Examples



Paths	Not Paths
(a, b, c)	(a, b, d)
(a, b, c, b, c)	
(a, b, c, b, c, b, c, b, c)	
(d, d, d, d)	

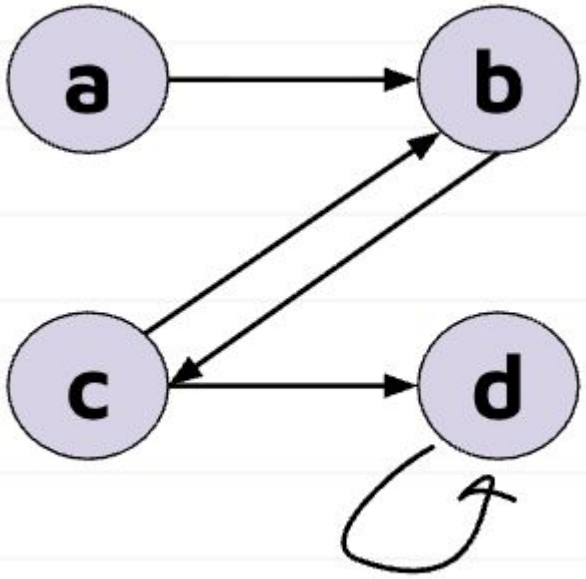
Examples



(a)

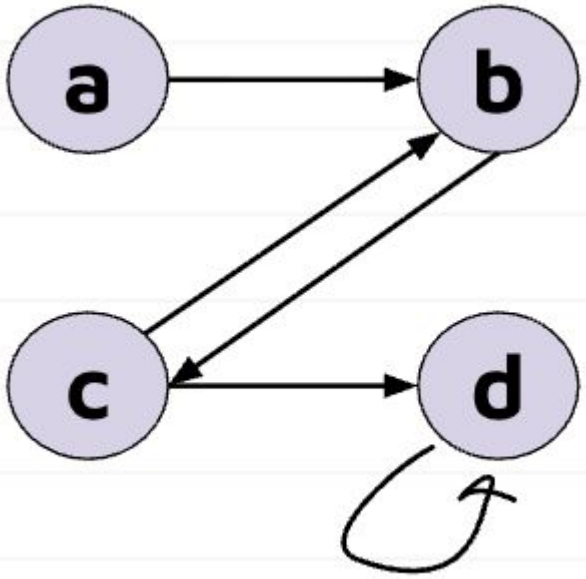
Paths	Not Paths
(a, b, c)	(a, b, d)
(a, b, c, b, c)	
(a, b, c, b, c, b, c, b, c)	
(d, d, d, d)	

Examples



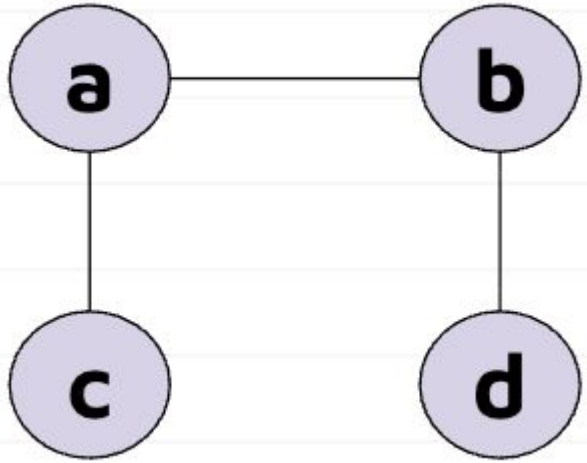
Paths	Not Paths
(a, b, c)	(a, b, d)
(a, b, c, b, c)	
(a, b, c, b, c, b, c, b, c)	
(d, d, d, d)	
(a)	
(a, a)	

Examples



Paths	Not Paths
(a, b, c)	(a, b, d)
(a, b, c, b, c)	(a, a)
(a, b, c, b, c, b, c, b, c)	
(d, d, d, d)	
(a)	

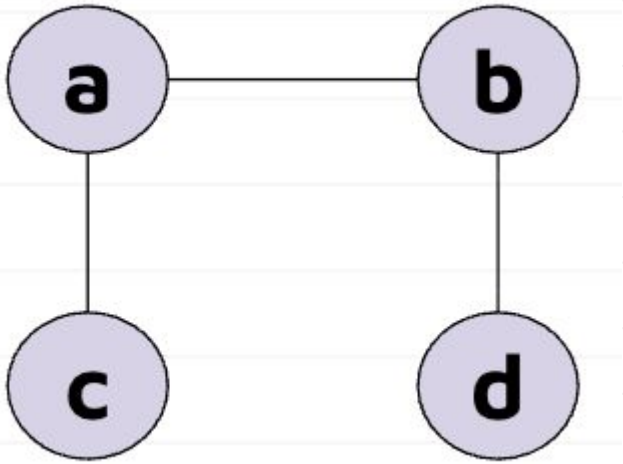
Examples



Note

Paths **can** go through the same node more than once!

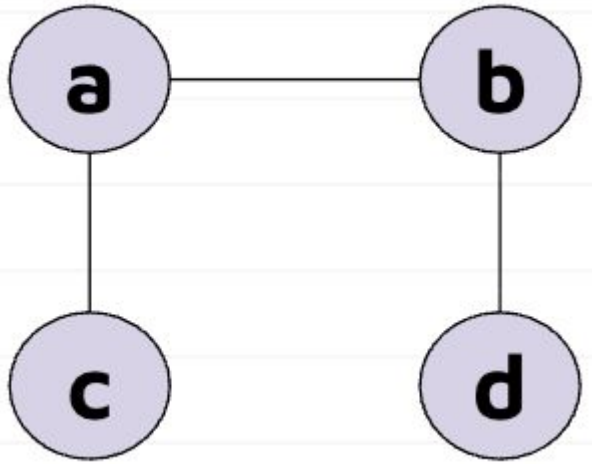
(c, **a**, **b**, **a**, **b**, d)



Simple Paths

Definition: A **simple path** is a path in which every node is *unique*.

(c, a, b, d)

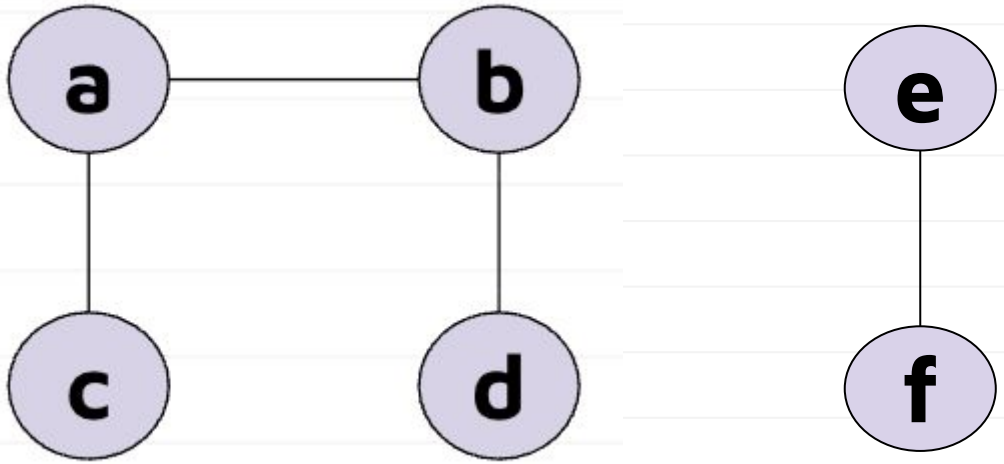


Reachability

Definition: node v is reachable from node u if there is a path from u to v.

Reachability

Definition: node v is reachable from node u if there is a path from u to v.



Reachability

Definition: node v is reachable from node u if there is a path from u to v.

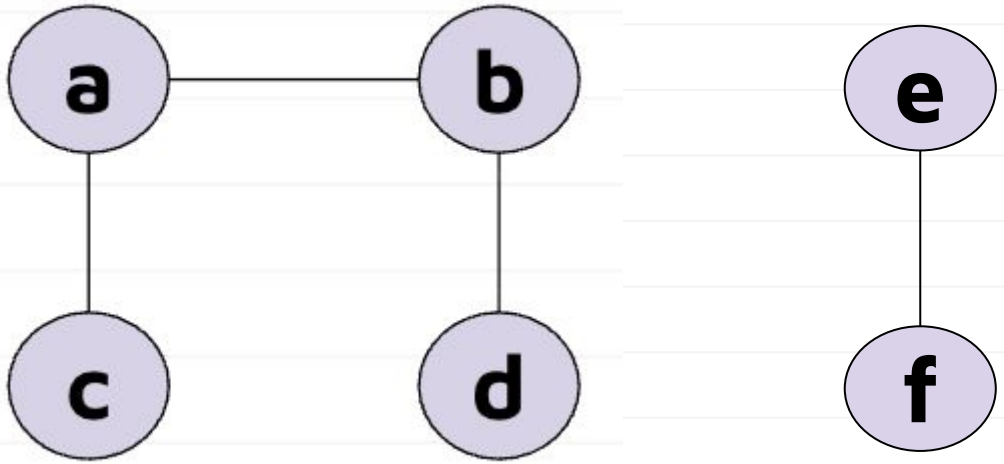
How many nodes are reachable from a?

A: 1

B: 2

C: 3

D: 4



Reachability

Definition: node v is reachable from node u if there is a path from u to v.

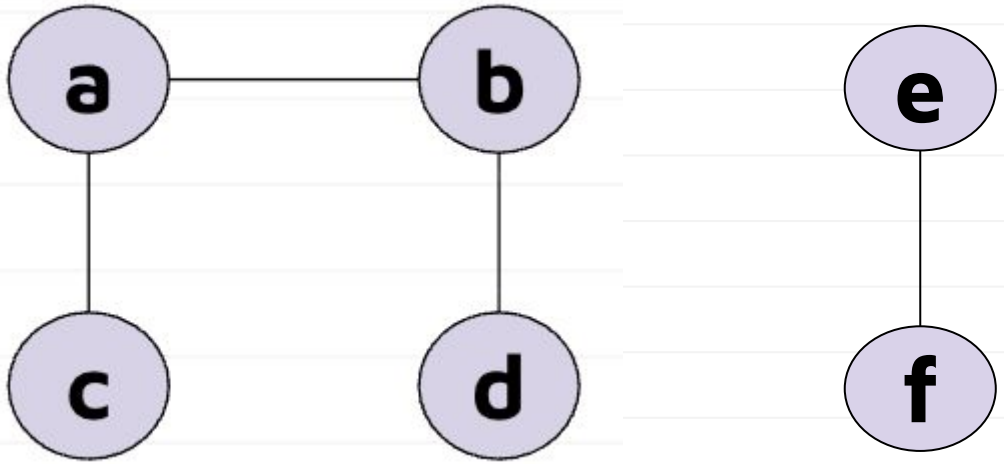
How many nodes are reachable from a?

A: 1

B: 2

C: 3

D: 4



Reachability and Directedness

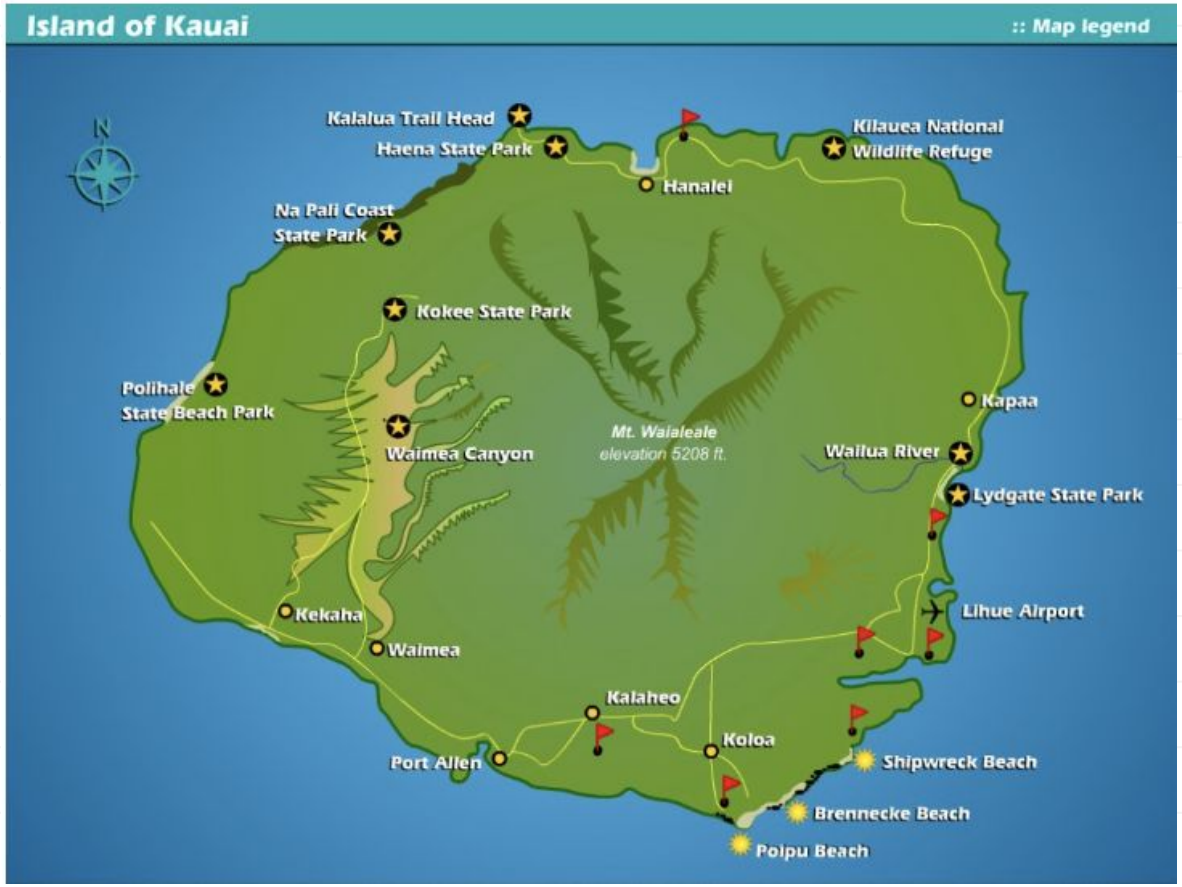
- If G is **undirected**, reachability is **symmetric**.
  - If u reachable from v, then v reachable from u.
- If G is **directed**, reachability is **not symmetric**.
  - If u reachable from v, then v may/may not be reachable from u.

Important Trivia

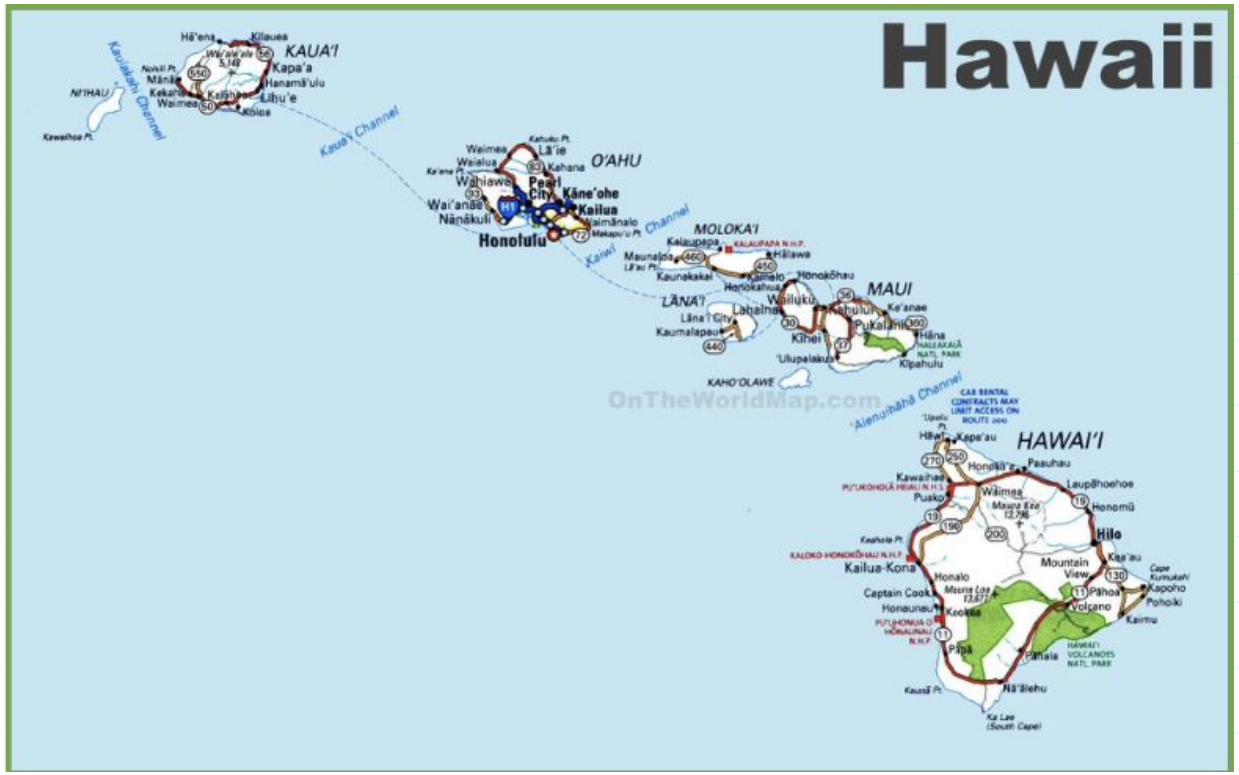
- In any graph, any node is **reachable** from **itself**.

Connected Components

Example



Example



Connectedness

A graph is connected if every node u is reachable from every other node v. Otherwise, it is disconnected.

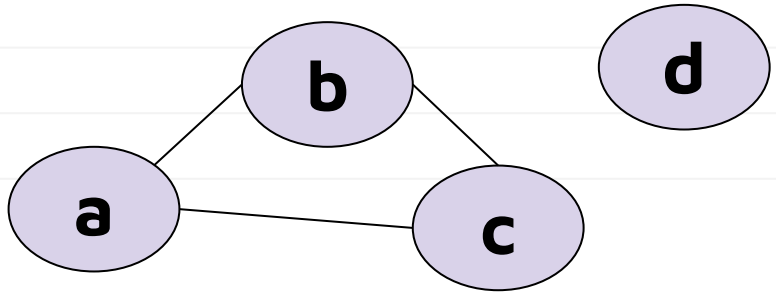
Equivalent: there is a path between every pair of nodes.

Connected Components

- A connected component is a **maximally-connected** set of nodes.
- I.e., if $G = (V, E)$ is an undirected graph, a connected component is a set $C \subseteq V$ such that
  - any pair $u, u' \in C$ are **reachable** from one another; and
  - if $u \in C$ and $z \notin C$ then u and z are **not** reachable from one another.

Connected Components

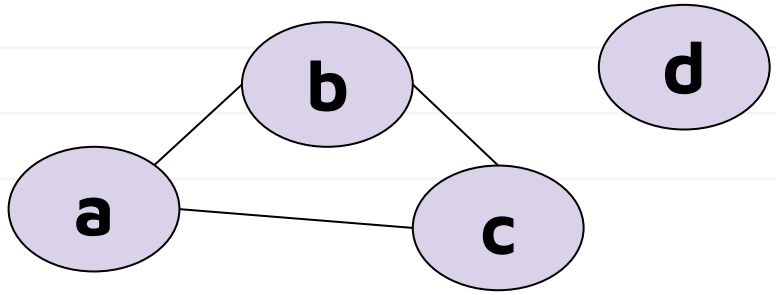
- A connected component is a **maximally-connected** set of nodes.
- I.e., if $G = (V, E)$ is an undirected graph, a connected component is a set $C \subset V$ such that
  - any pair $u, u' \in C$ are **reachable** from one another; and
  - if $u \in C$ and $z \notin C$ then u and z are **not** reachable from one another.



{a, b, c} - max. connected?

Connected Components (back)

- A connected component is a **maximally-connected** set of nodes.
- I.e., if $G = (V, E)$ is an undirected graph, a connected component is a set $C \subset V$ such that
  - any pair $u, u' \in C$ are **reachable** from one another; and
  - if $u \in C$ and $z \notin C$ then u and z are **not** reachable from one another.



{b, c} - max. connected?

Exercise mic

What are the connected components?

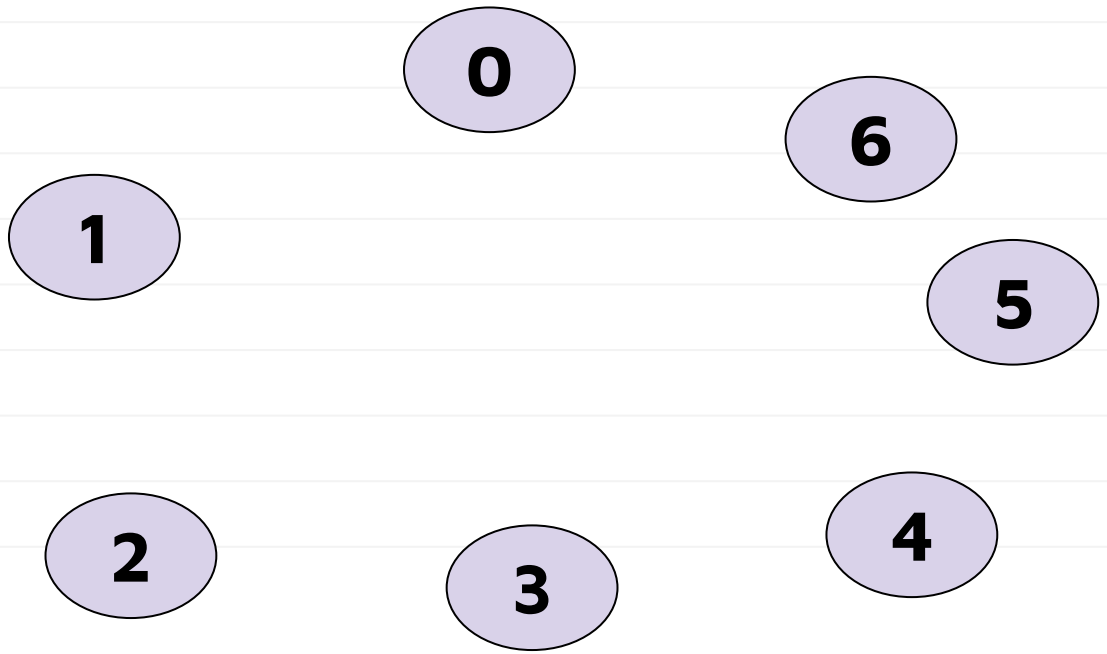
$V = \{0, 1, 2, 3, 4, 5, 6\}$

$E = \{(0, 2), (1, 5), (3, 1), (2, 4), (0, 4), (5, 3)\}$

What are the connected components?

V = {0, 1, 2, 3, 4, 5, 6}

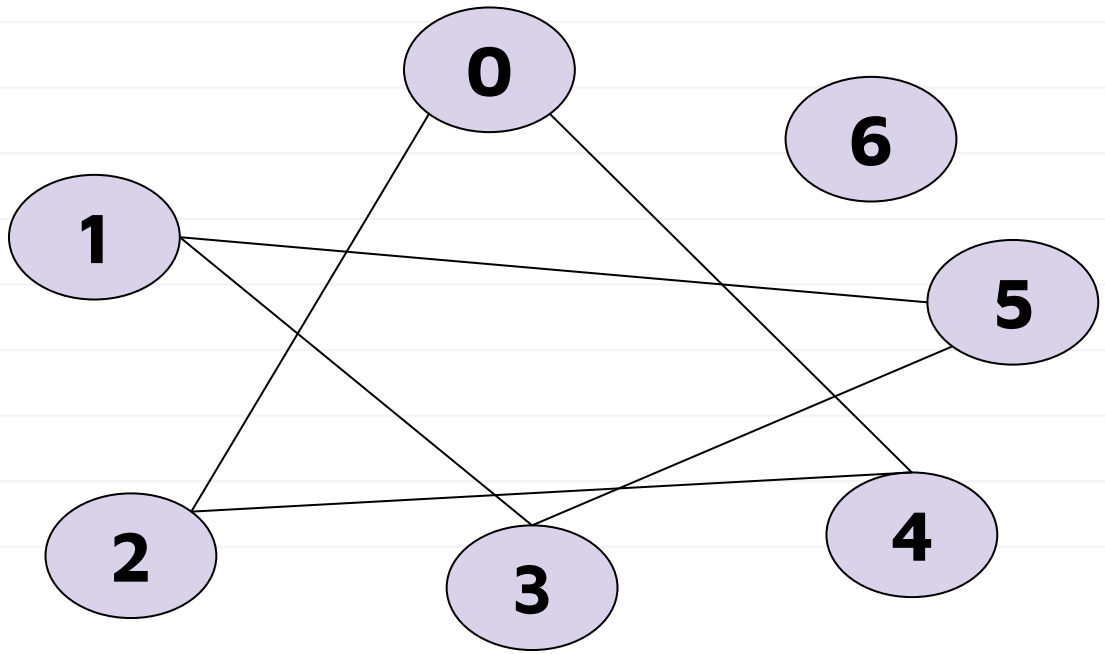
E = {(0, 2), (1, 5), (3, 1), (2, 4), (0, 4), (5, 3)}



What are the connected components?

V = {0, 1, 2, 3, 4, 5, 6}

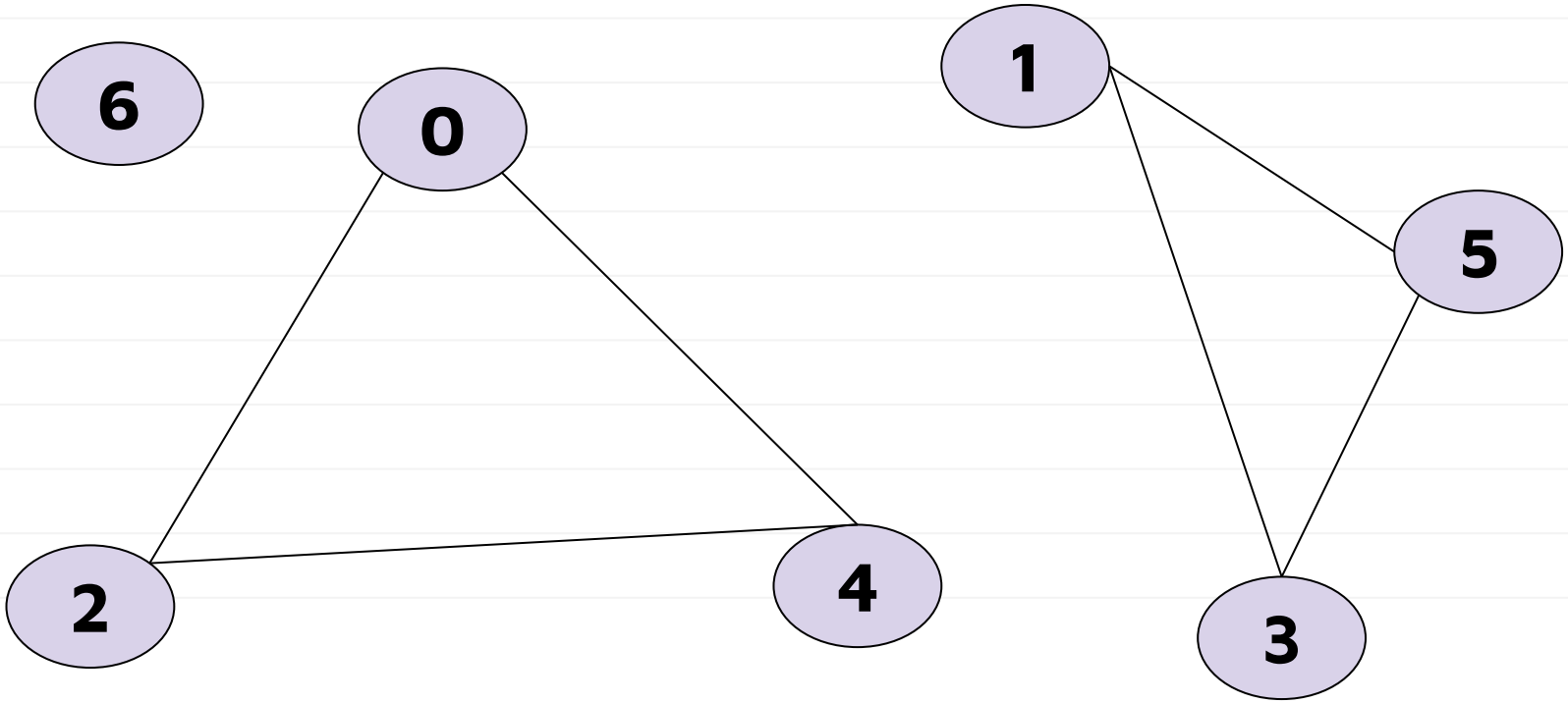
E = {(0, 2), (1, 5), (3, 1), (2, 4), (0, 4), (5, 3)}



What are the connected components?

V = {0, 1, 2, 3, 4, 5, 6}

E = {(0, 2), (1, 5), (3, 1), (2, 4), (0, 4), (5, 3)}

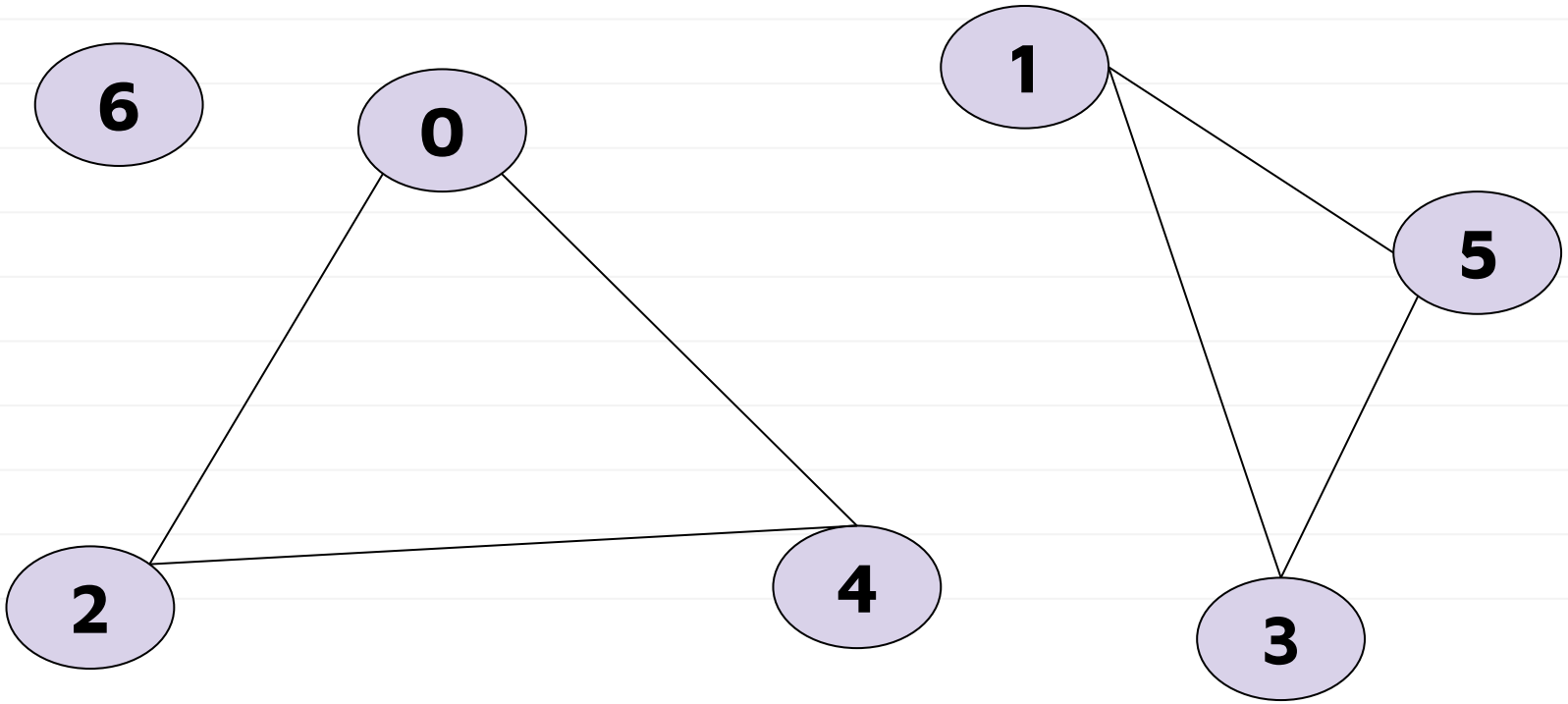


What are the connected components?

3

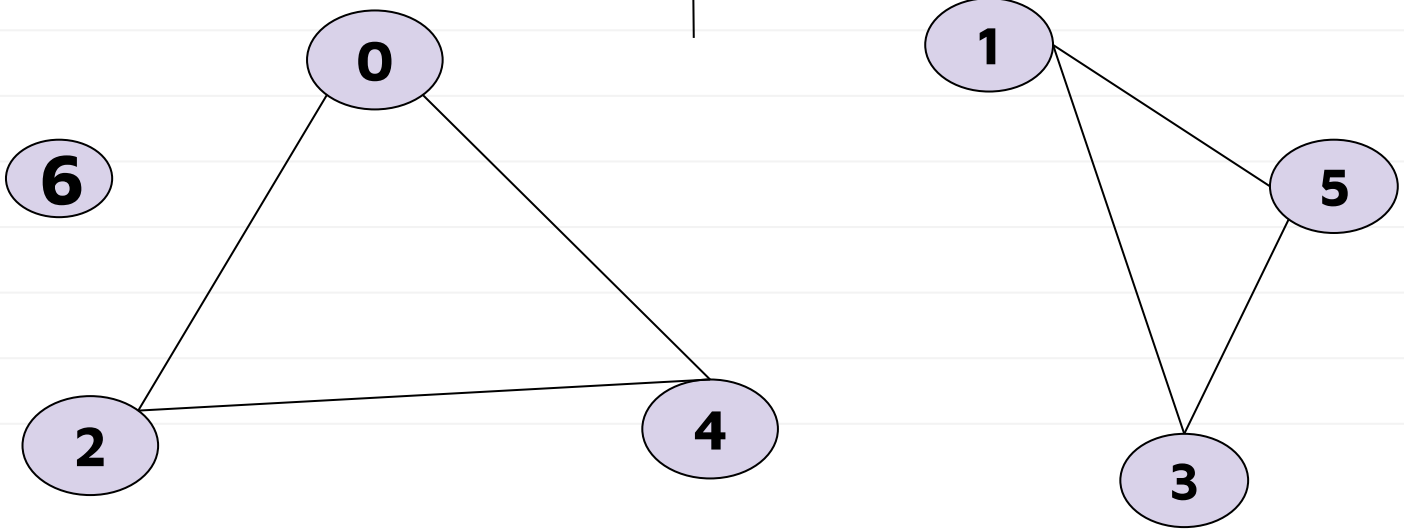
V = {0, 1, 2, 3, 4, 5, 6}

E = {(0, 2), (1, 5), (3, 1), (2, 4), (0, 4), (5, 3)}

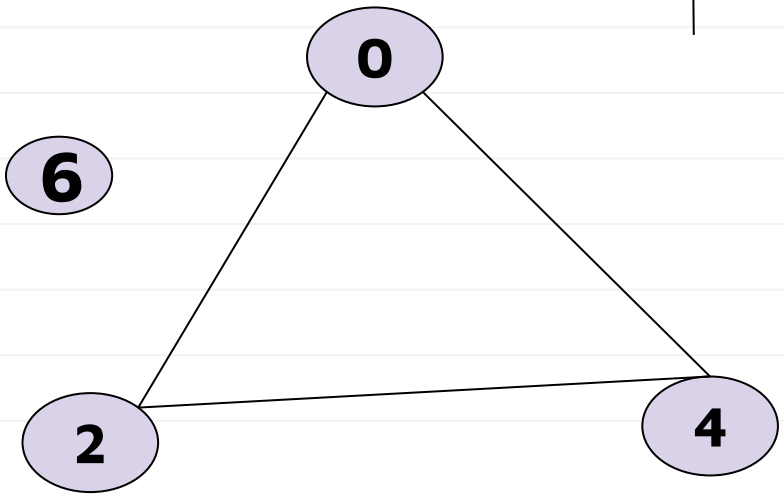


maximally-connected

not maximally-connected

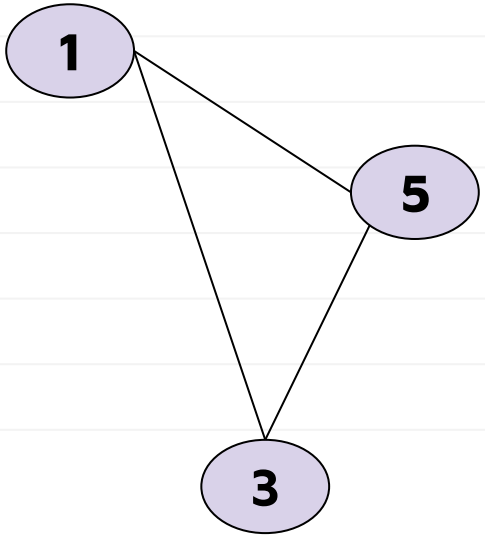


maximally-connected



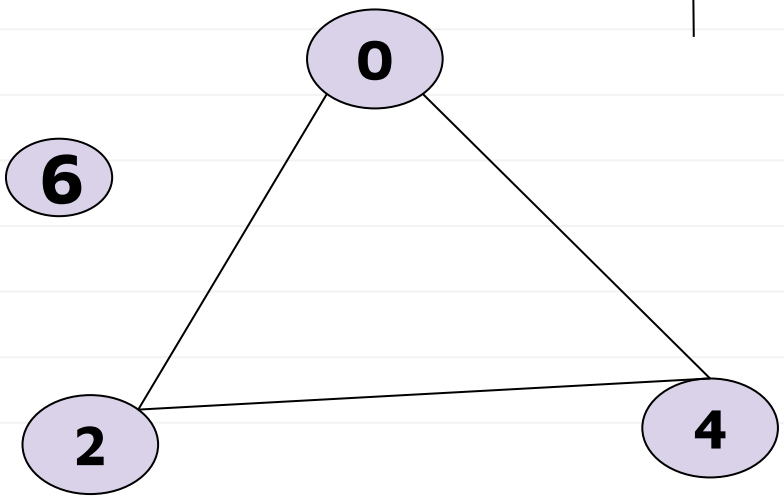
not maximally-connected

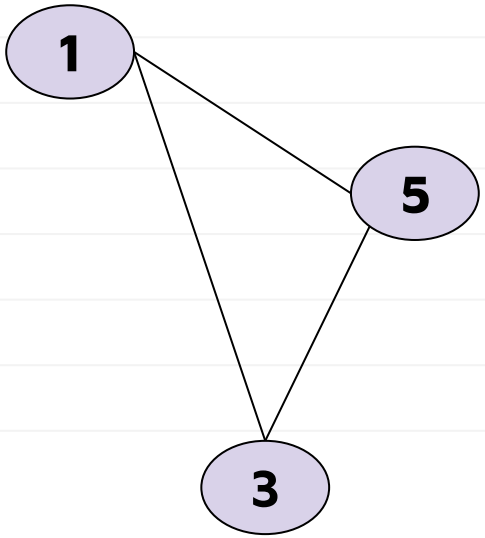
{2, 6}



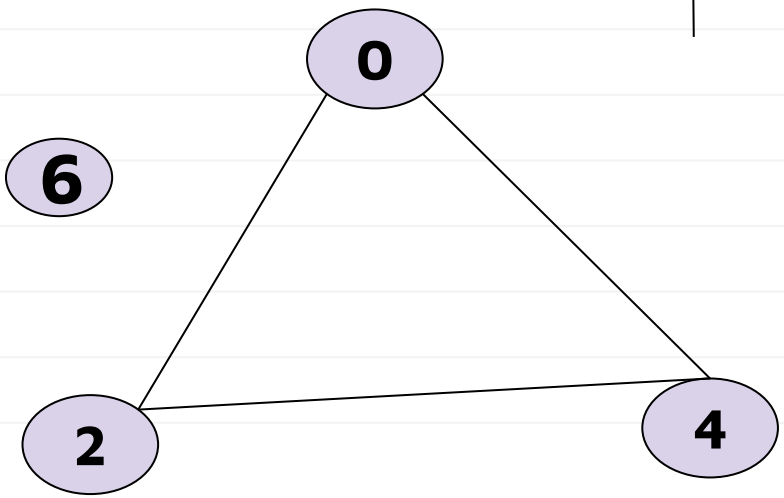
maximally-connected	not maximally-connected
	{2, 6}

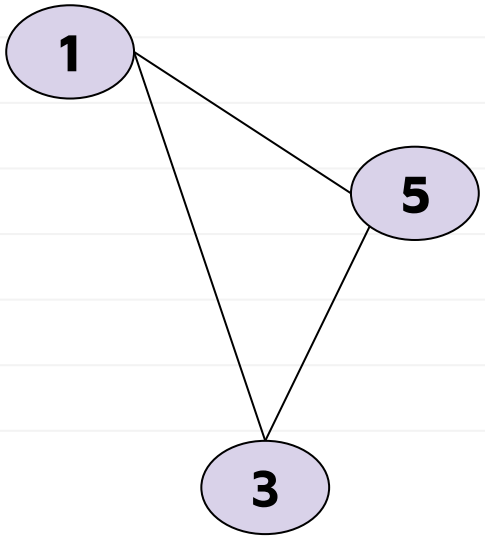
2 and 6 are **not** connected





maximally-connected	not maximally-connected
	{2, 6}
	{2, 0, 4, 6}



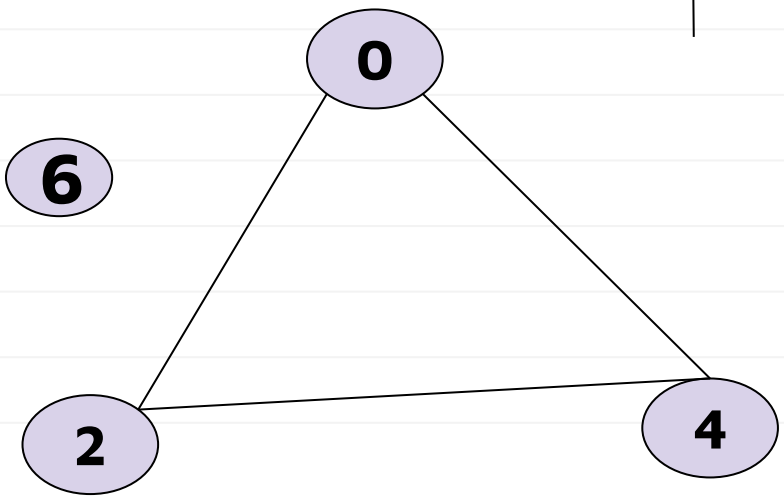


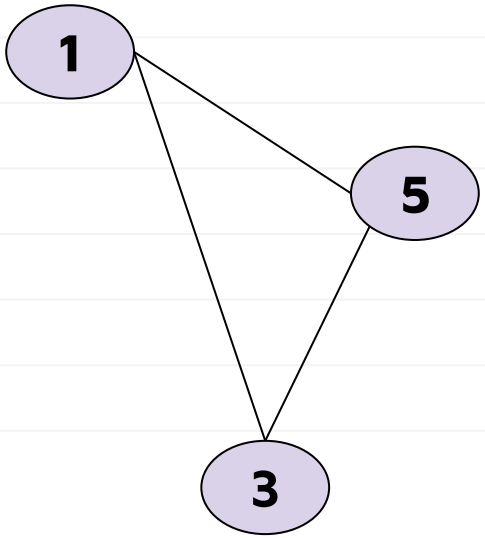
maximally-connected	not maximally-connected

{2, 6}

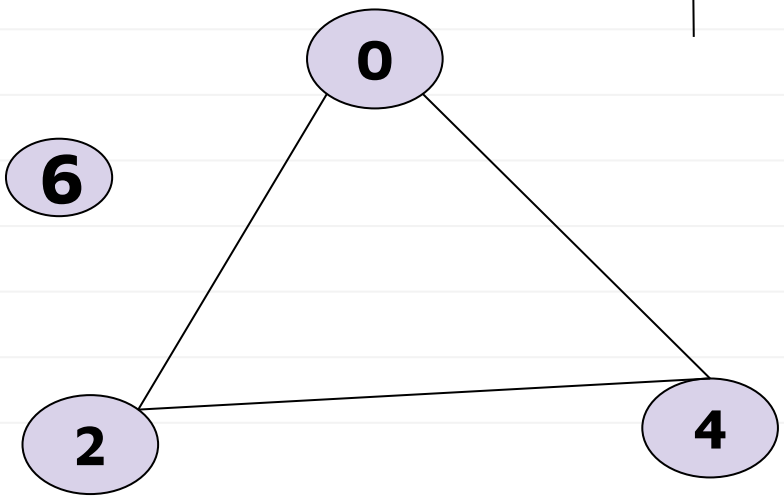
{2, 0, 4, 6}

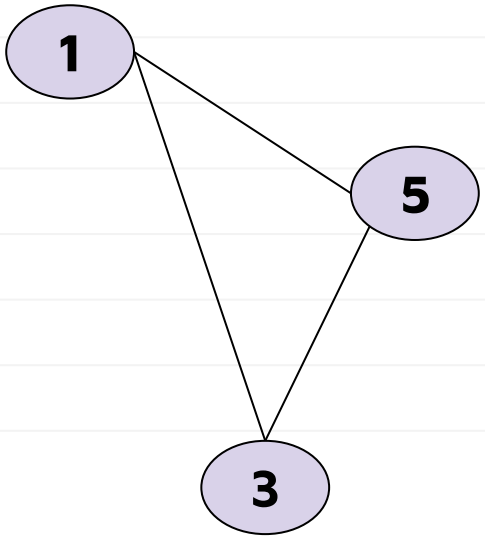
2 and 6 are **not** connected



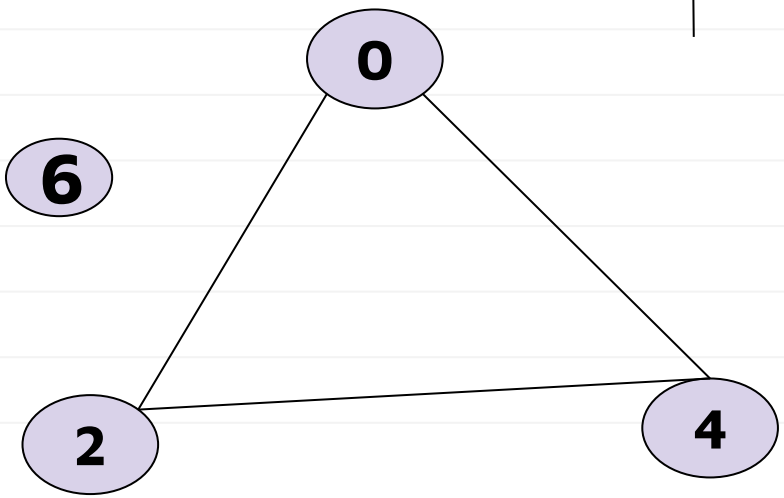


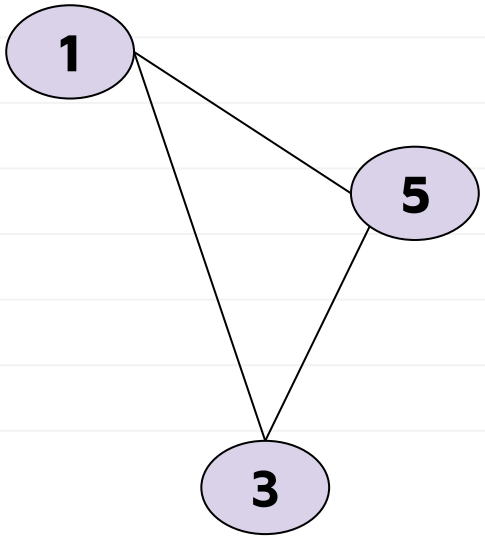
maximally-connected	not maximally-connected
	{2, 6}
	{2, 0, 4, 6}
	{1, 3, 5}



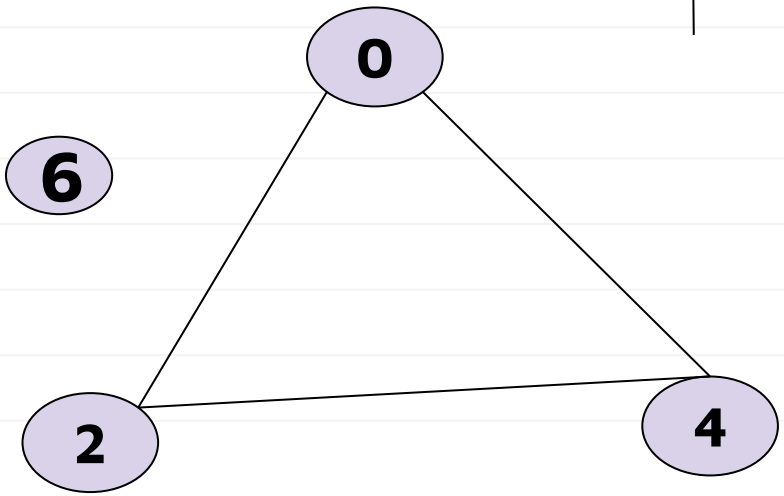


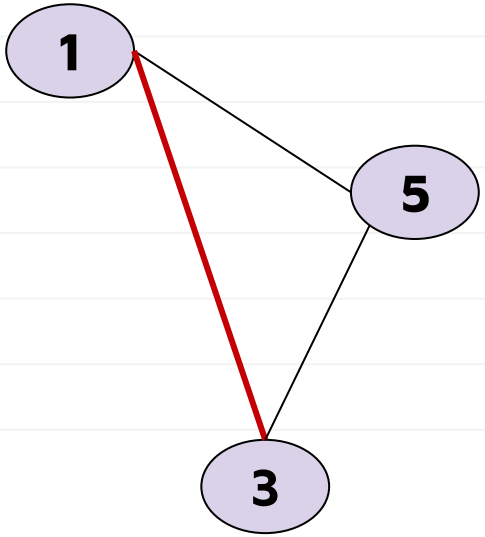
maximally-connected	not maximally-connected
{1, 3, 5}	{2, 6} {2, 0, 4, 6}



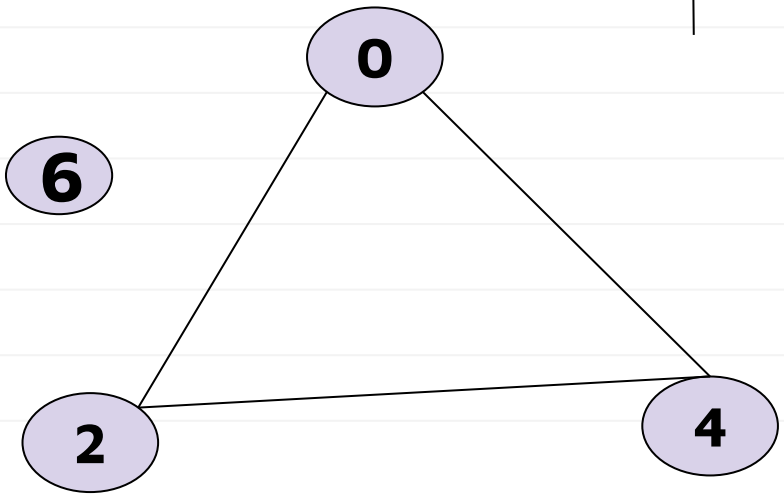


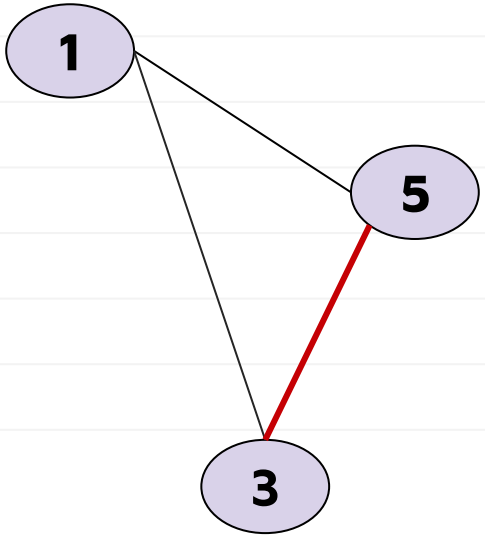
maximally-connected	not maximally-connected
{1, 3, 5}	{2, 6} {2, 0, 4, 6}





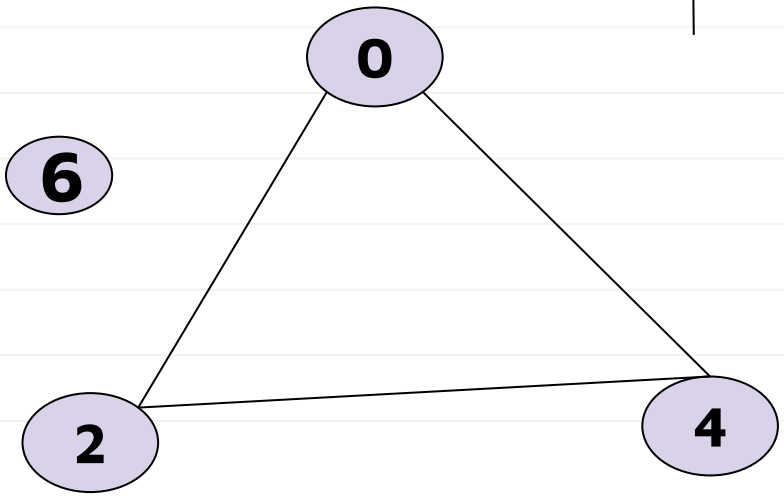
maximally-connected	not maximally-connected
{1, 3, 5}	{2, 6} {2, 0, 4, 6}





maximally-connected

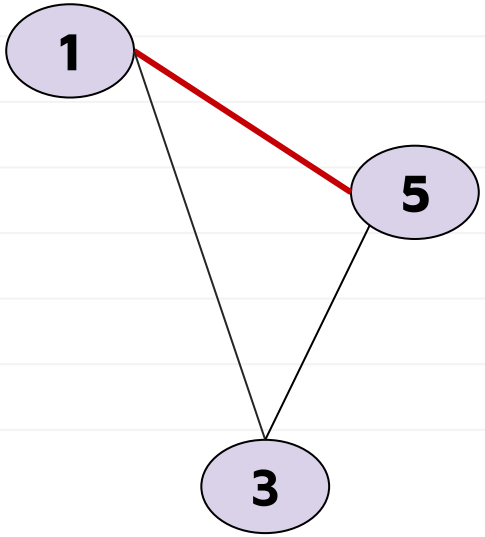
{1, 3, 5}



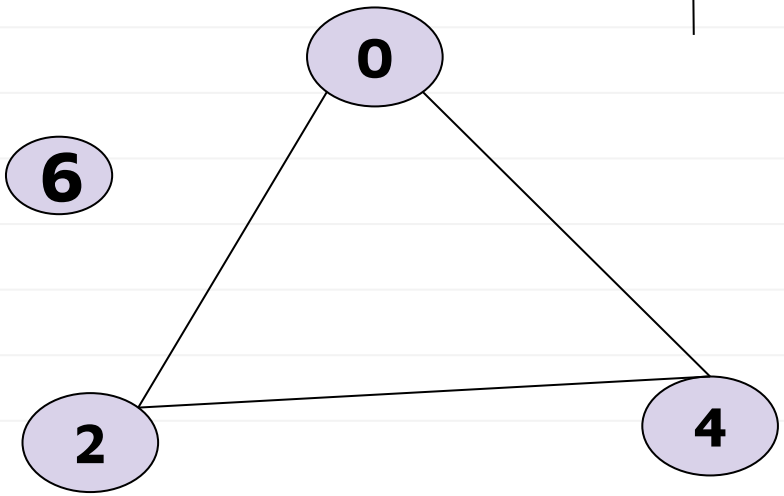
not maximally-connected

{2, 6}

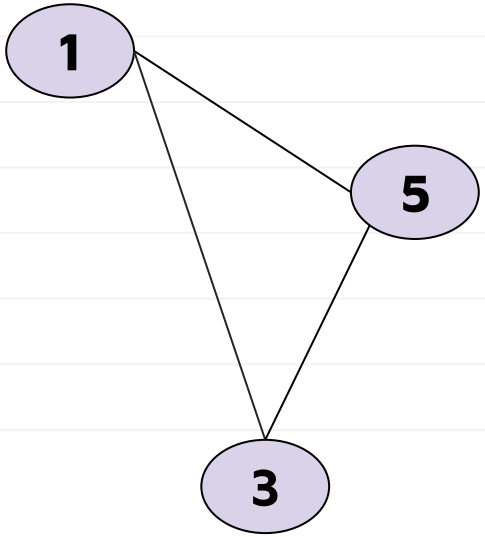
{2, 0, 4, 6}



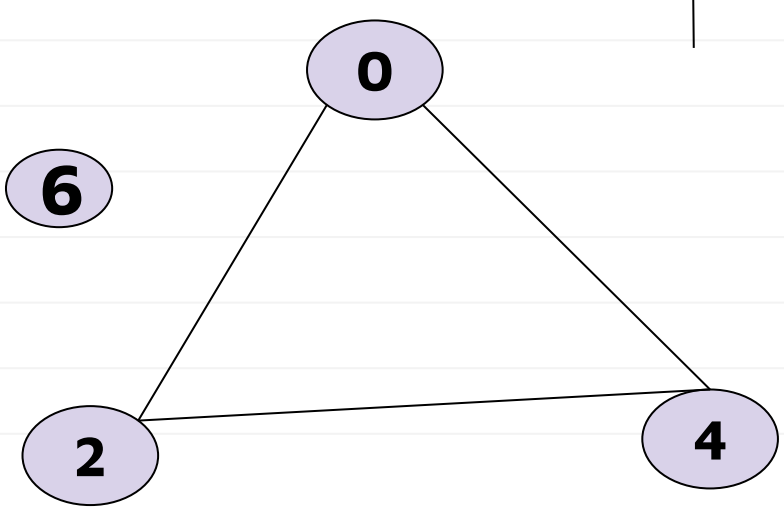
maximally-connected	not maximally-connected
{1, 3, 5}	{2, 6} {2, 0, 4, 6}

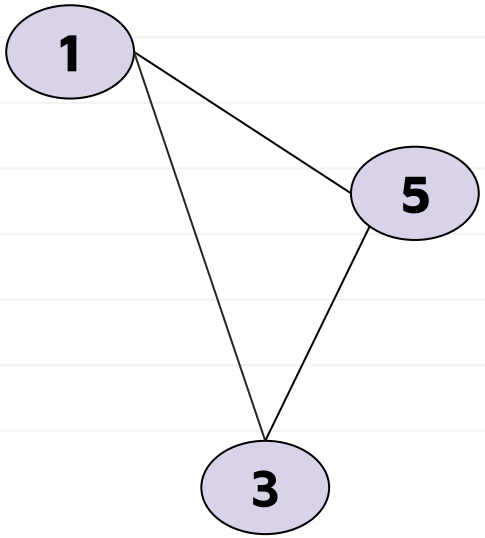


{2, 0}

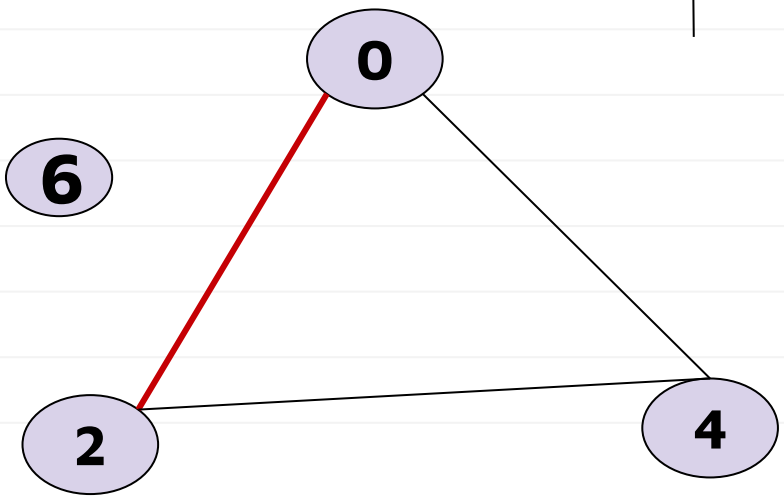


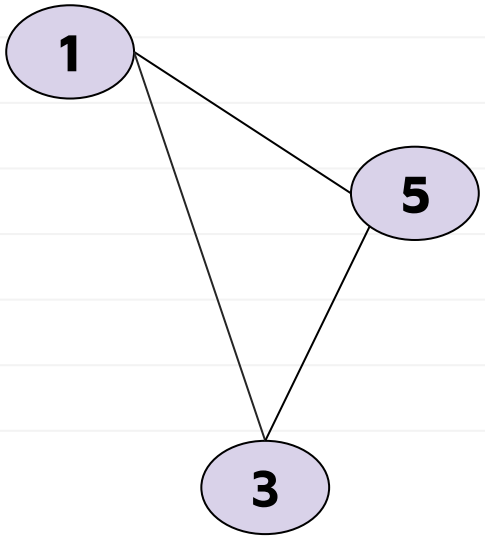
maximally-connected	not maximally-connected
{1, 3, 5}	{2, 6} {2, 0, 4, 6} {2, 0}



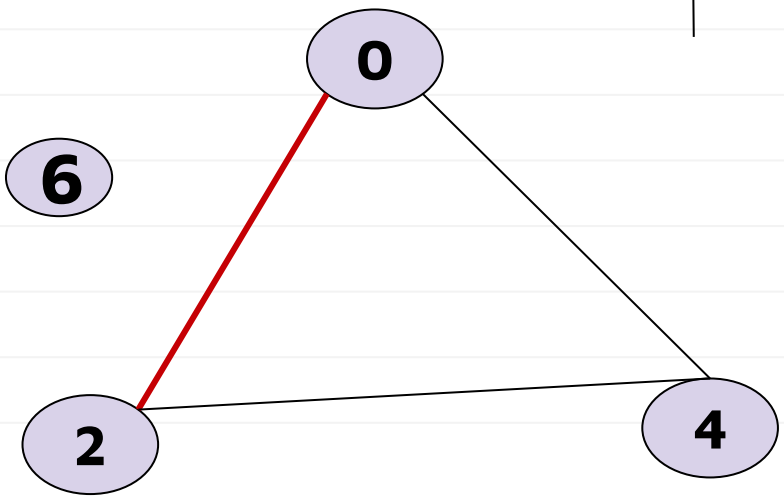


maximally-connected	not maximally-connected
{1, 3, 5}	{2, 6} {2, 0, 4, 6} {2, 0}



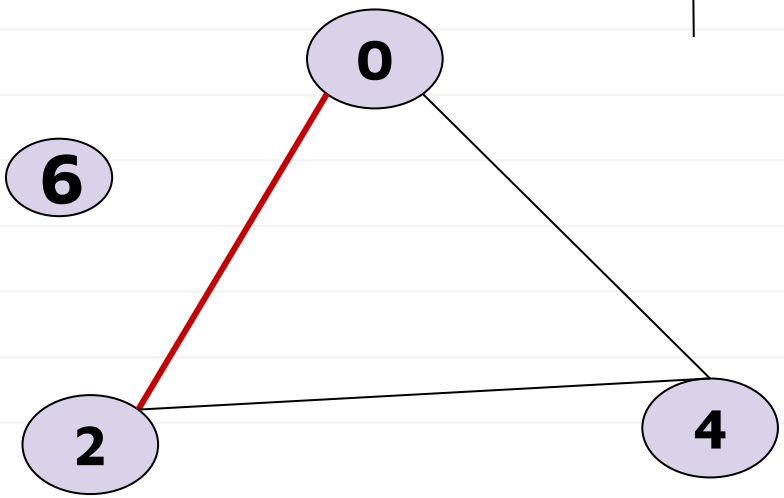


maximally-connected	not maximally-connected
{1, 3, 5}	{2, 6} {2, 0, 4, 6} {2, 0}



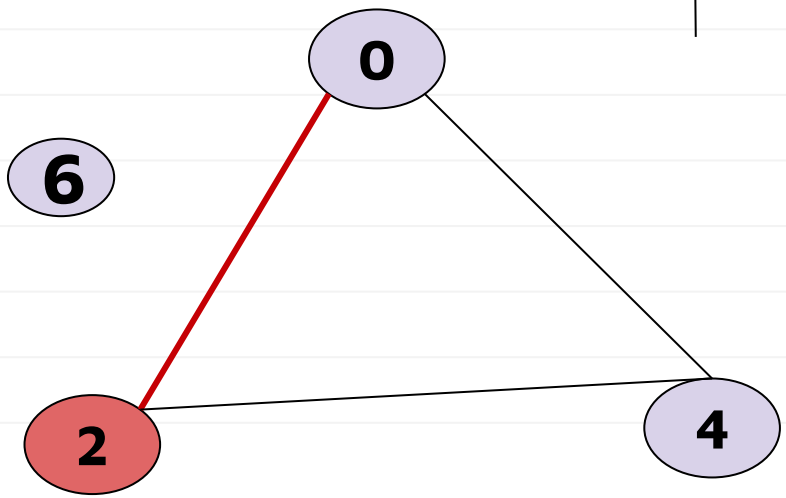
if $u \in C$ and $z \notin C$ then u and z are **not** reachable from one another.

maximally-connected	not maximally-connected
{1, 3, 5}	{2, 6} {2, 0, 4, 6} {2, 0} = C



if $u \in C$ and $z \notin C$ then u and z are **not** reachable from one another.

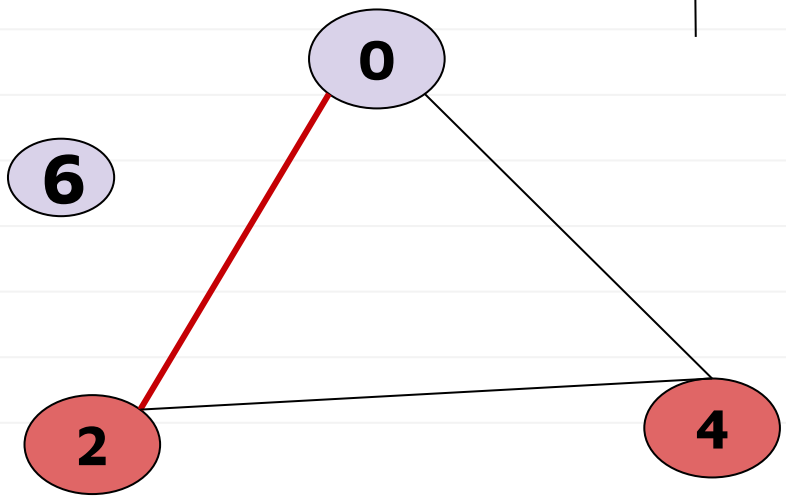
maximally-connected	not maximally-connected
{1, 3, 5}	{2, 6} {2, 0, 4, 6} {2, 0} = C



if $u \in C$ and $z \notin C$ then u and z are **not** reachable from one another.

u is 2

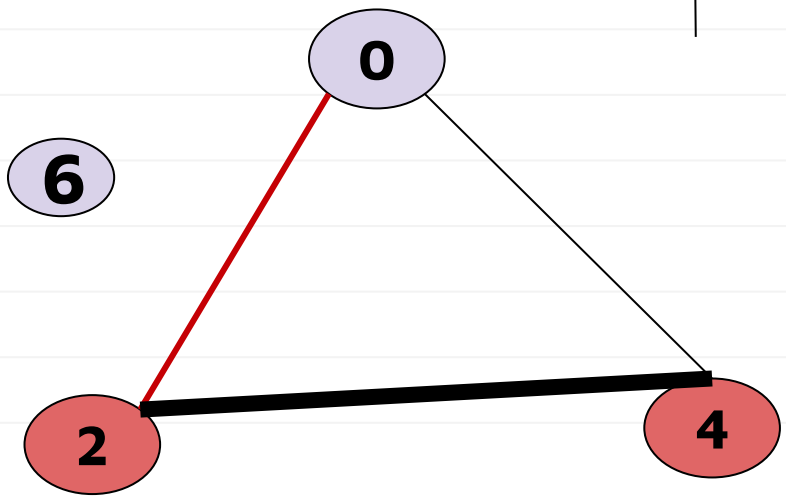
maximally-connected	not maximally-connected
{1, 3, 5}	{2, 6} {2, 0, 4, 6} {2, 0} = C



if $u \in C$ and $z \notin C$ then u and z are **not** reachable from one another.

u is 2, z = 4

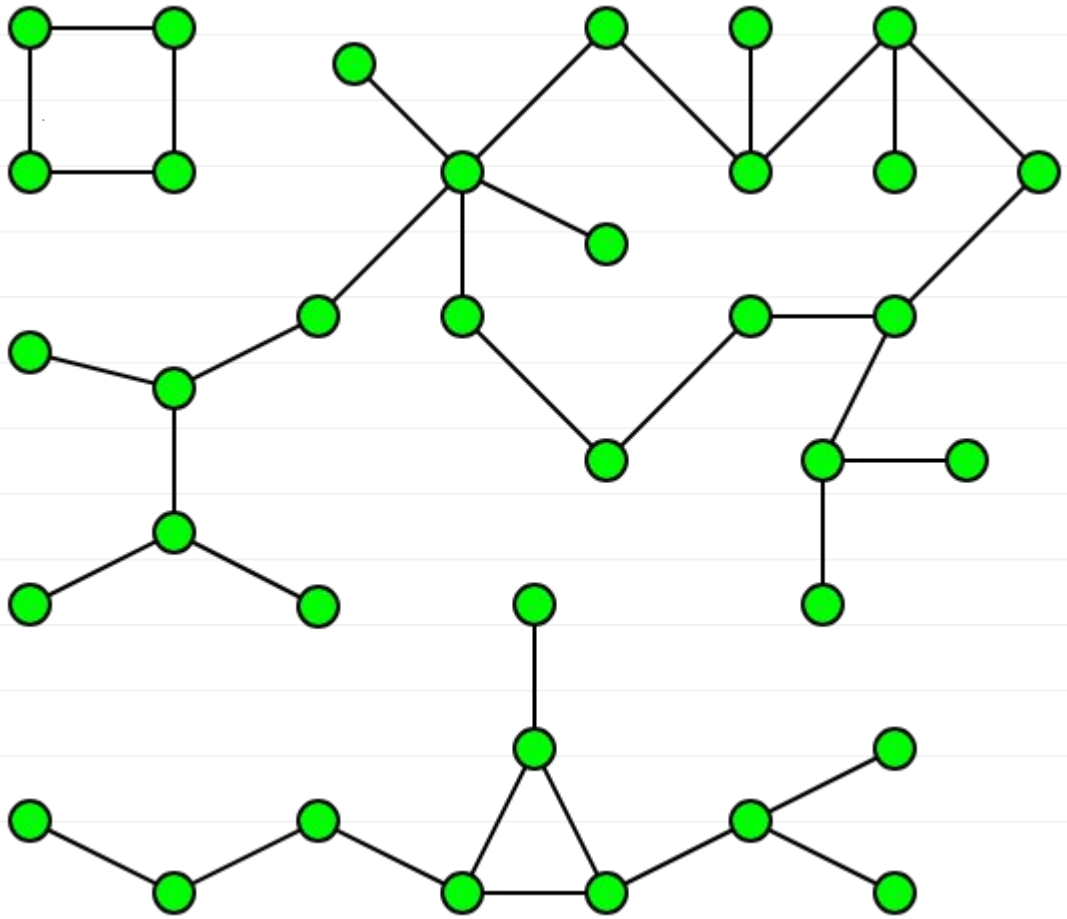
maximally-connected	not maximally-connected
{1, 3, 5}	{2, 6} {2, 0, 4, 6} {2, 0} = C



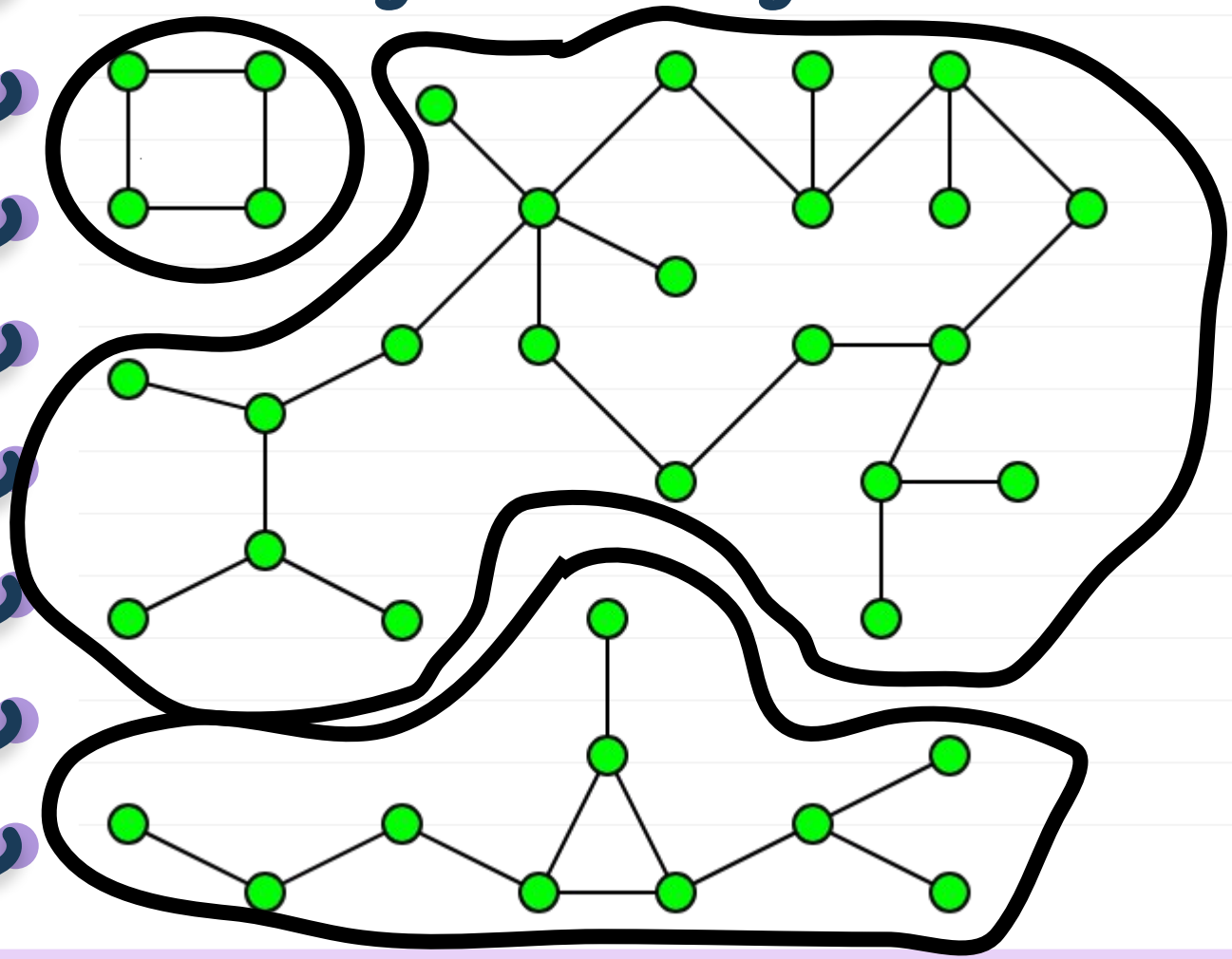
if $u \in C$ and $z \notin C$ then u and z **are not reachable from one another**.

u is 2, z = 4. But they are reachable! **So, C can be larger!**

How many maximally-connected components?



How many maximally-connected components?



Thank you!

Do you have any questions?

CampusWire!