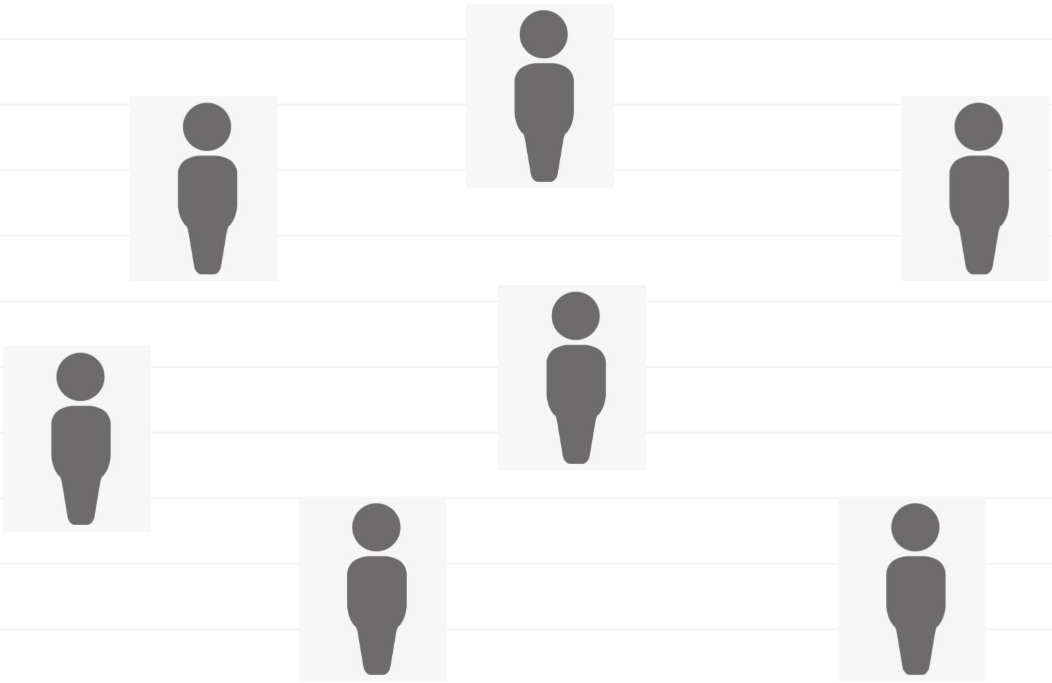
DSC 40B

Lecture 16 : Graphs

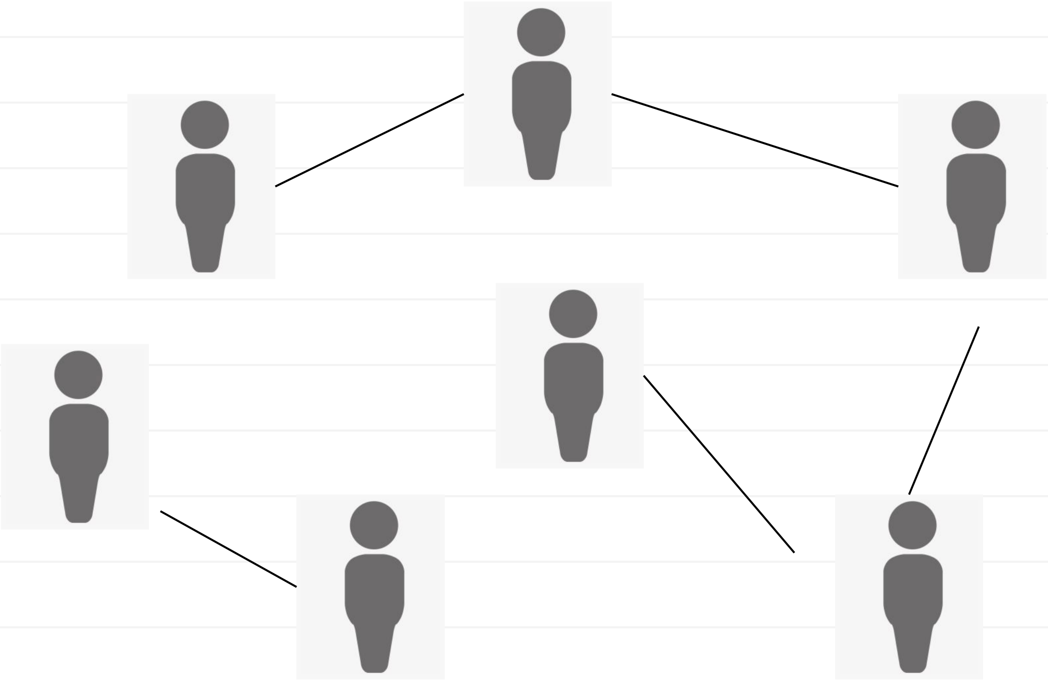
Data Types

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

Example: Facebook

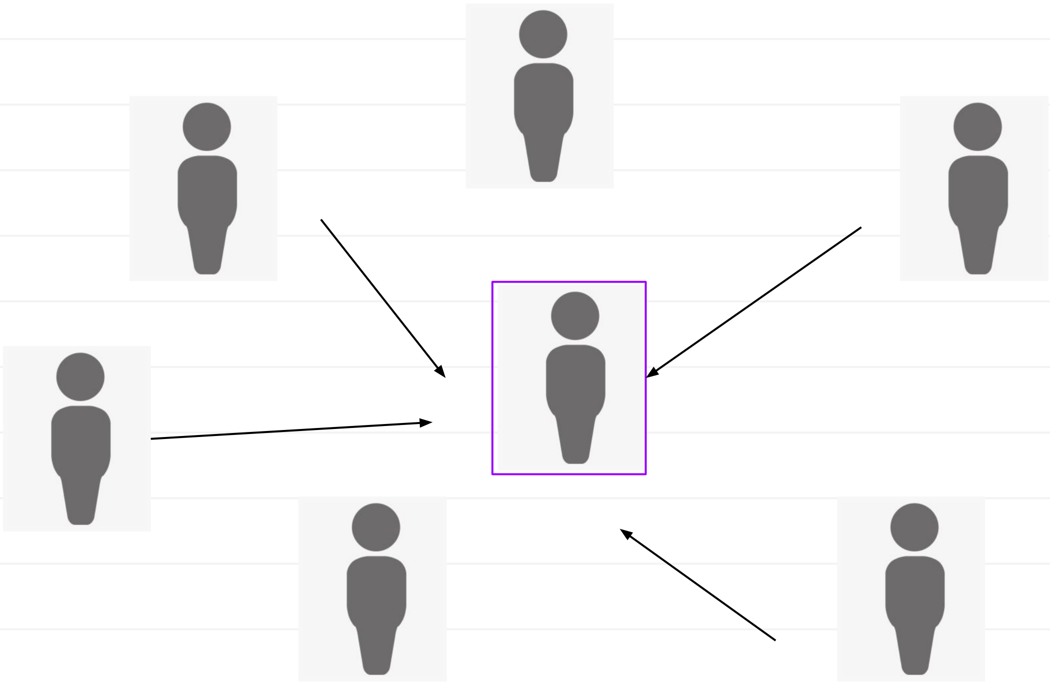


Example: Facebook

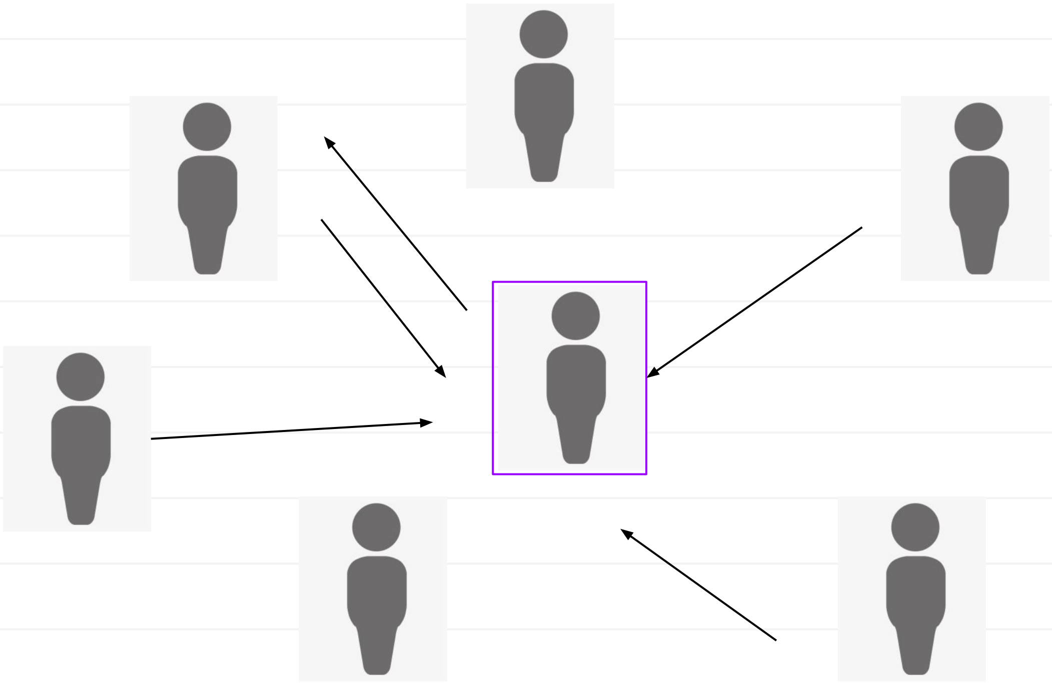


Example: Twitter

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

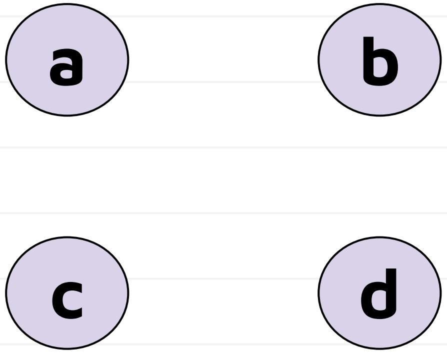
E = $\{(a, c), (a, b), (d, b), (b, d), (b, b)\}$

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).

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

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

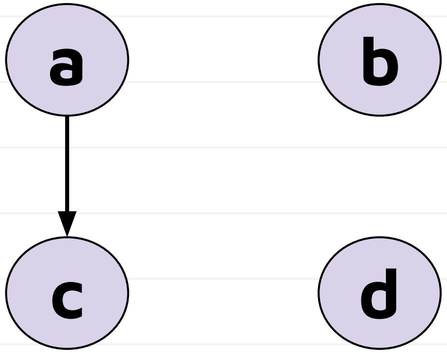


Definition

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).

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

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

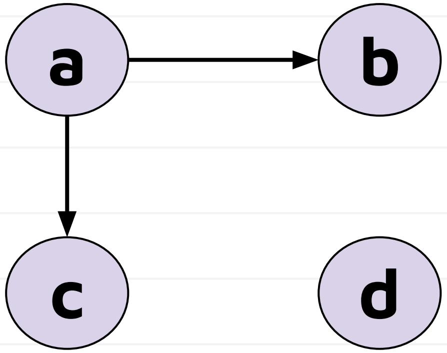


Definition

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).

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

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

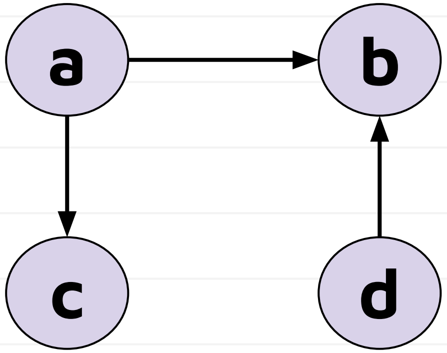


Definition

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).

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

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

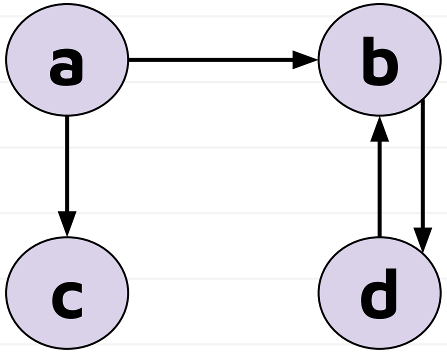


Definition

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).

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

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

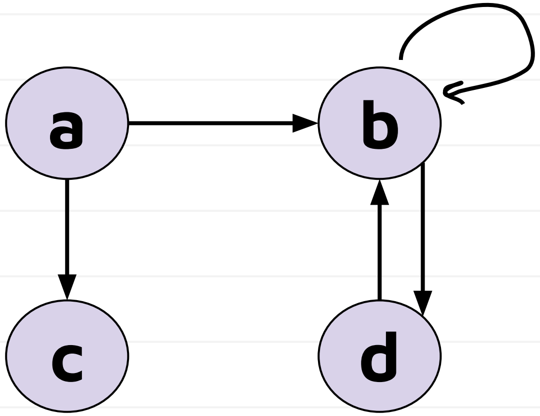


Definition

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).

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

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



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)**,

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

}

Directed Graphs (More Formally)

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

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

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

}

Directed Graphs (More Formally)

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

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

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

(c, 1), (c, 2)

}

Directed Graphs (More Formally)

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

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, $\boldsymbol{V} \times \boldsymbol{V}$.

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

$\{a, b, c\} \times \{a, b, c\}$ = (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 $V \times V$:

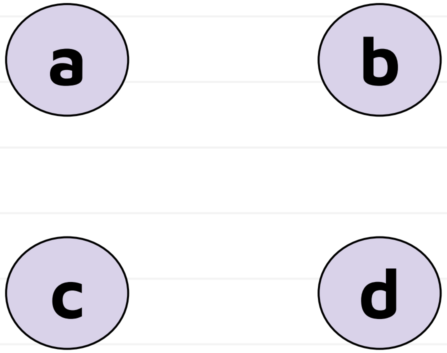
- the edges have **directions**.
  - e.g., (a, b) is "from a to b"
- can have "**opposite**" edges.
  - e.g., (a, b) and (b, a).
- can have "**self-loops**"
  - e.g., (a, a)

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).

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

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

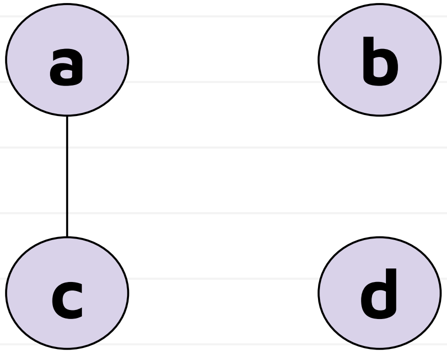


Definition

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).

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

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

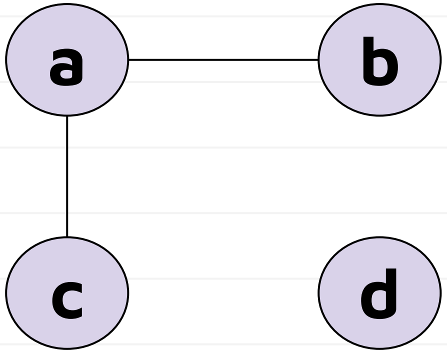


Definition

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).

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

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

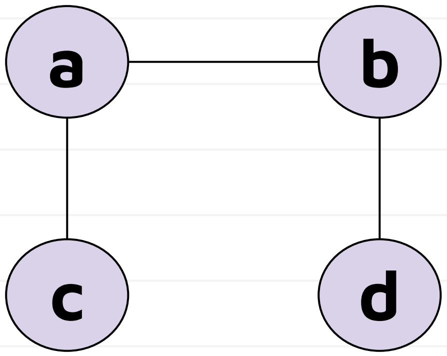


Definition

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).

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

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



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:

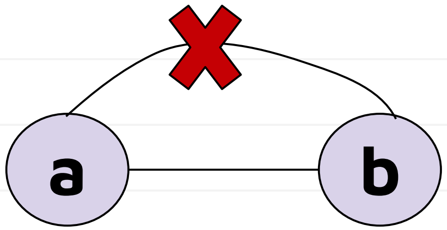
- **Edges have direction?**:
  - Directed: **Yes**
  - Undirected: **No**
- **Self-loops, (u, u)?**
  - Directed: **Yes**
  - Undirected: **No**
- **Opposite edges, (u, v) and (v, u)?**
  - Directed: **Yes**
  - Undirected: **No** (they are the same edge)

Note

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

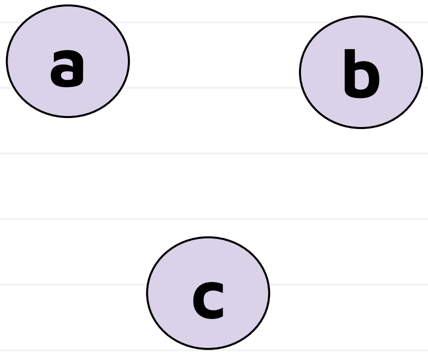
Note

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



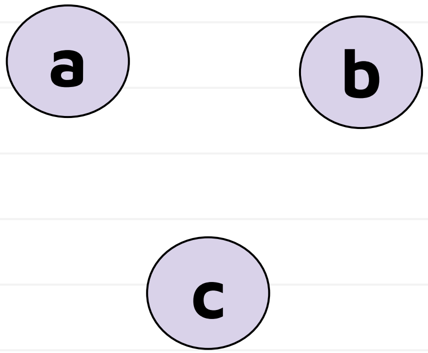
Note

Graphs don’t need to be “connected”



Note

Graphs don’t need to be “connected”

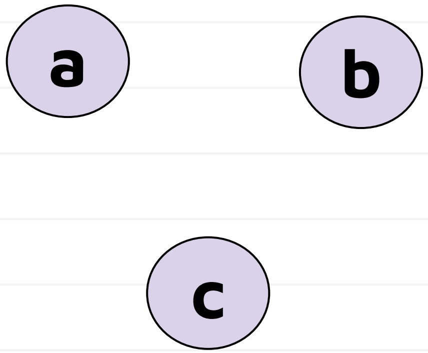


V = ?

E= ?

Note

Graphs don’t need to be “connected”

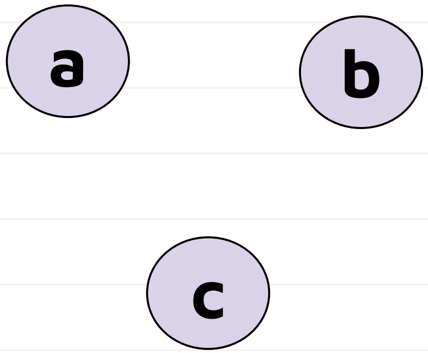


V = {a, b, c}

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: $\sim n^2/2$

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

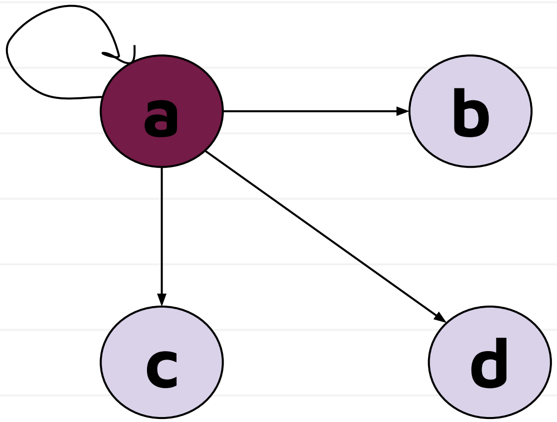
b

c

d

Counting Edges

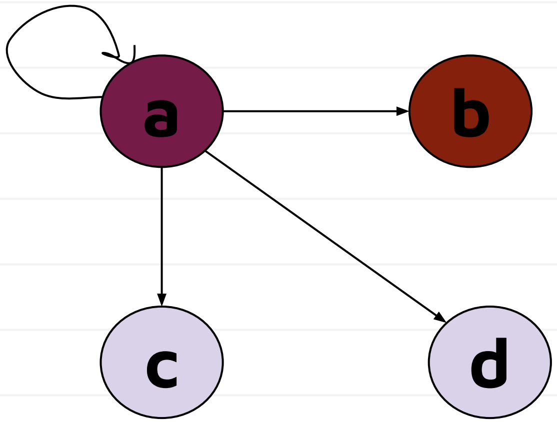
- What is the **greatest** number edges possible in a **directed** graph with *n* nodes?



4

Counting Edges

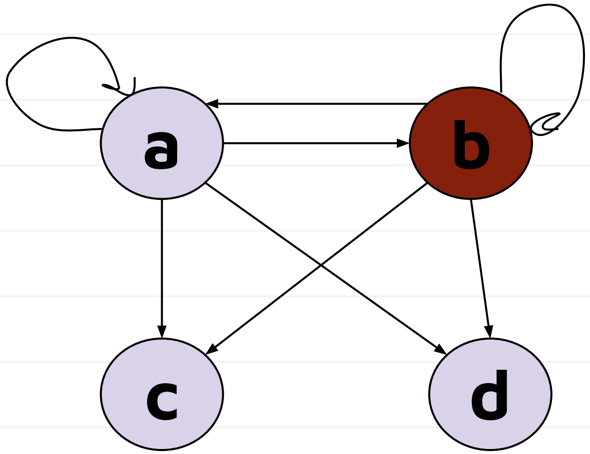
- What is the **greatest** number edges possible in a **directed** graph with *n* nodes?



4

Counting Edges

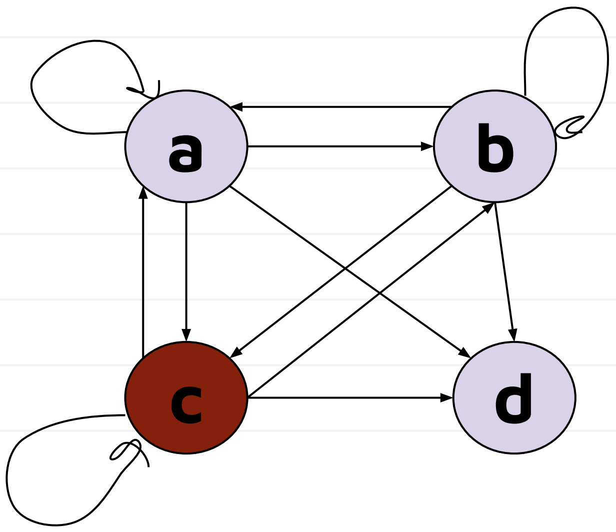
- What is the **greatest** number edges possible in a **directed** graph with *n* nodes?



4 + 4

Counting Edges

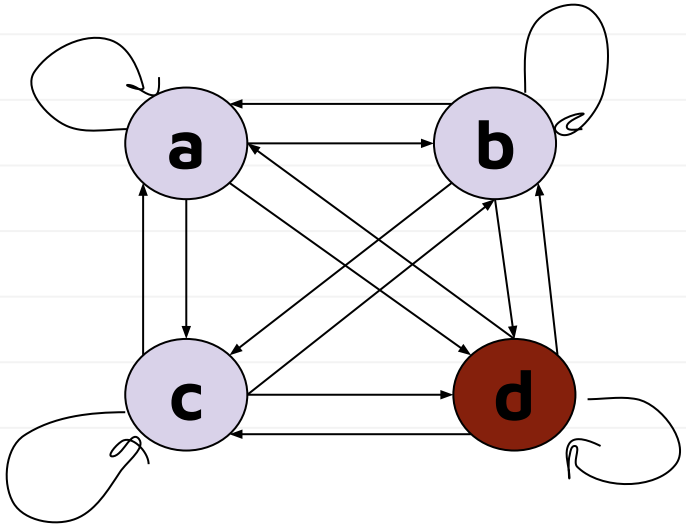
- What is the **greatest** number edges possible in a **directed** graph with *n* nodes?



4 + 4 + 4

Counting Edges

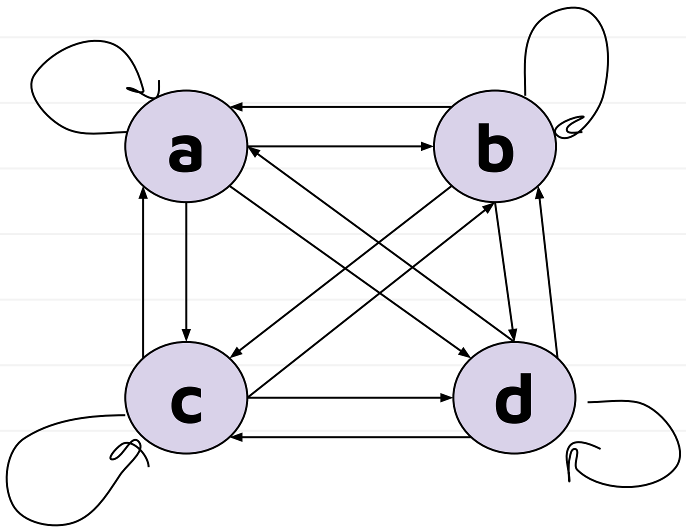
- What is the **greatest** number edges possible in a **directed** graph with *n* nodes?



4 + 4 + 4 + 4

Counting Edges

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

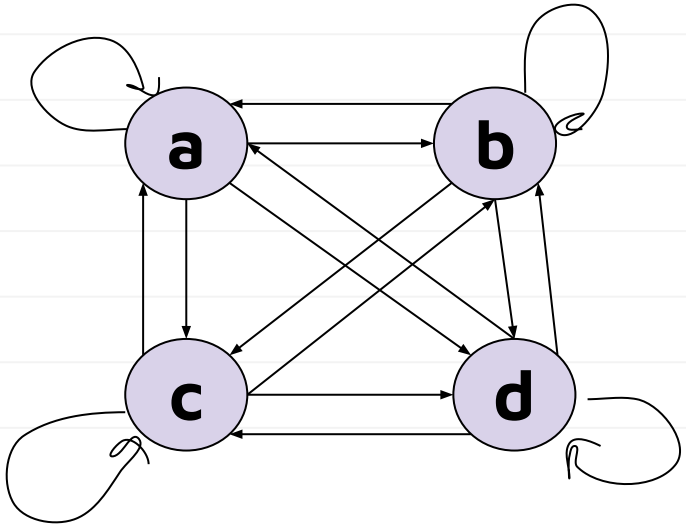


$4 + 4 + 4 + 4 = 16 = 4^2$

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

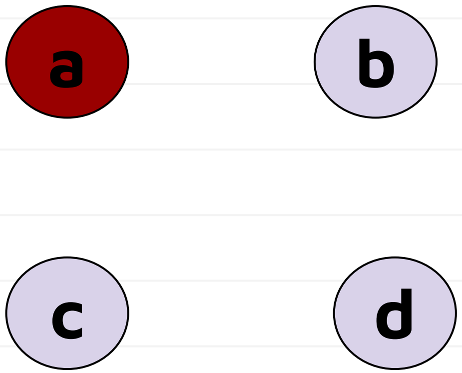
C: $\sim n^2/2$

D: n choose 2

E: n^2 choose 2

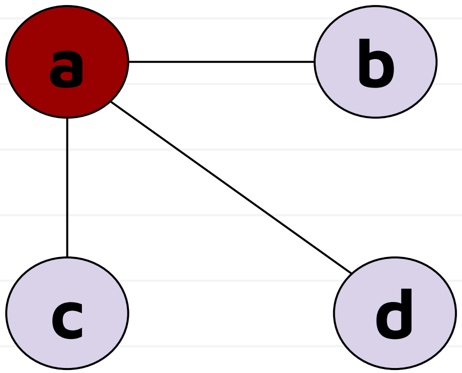
Counting Edges

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



Counting Edges

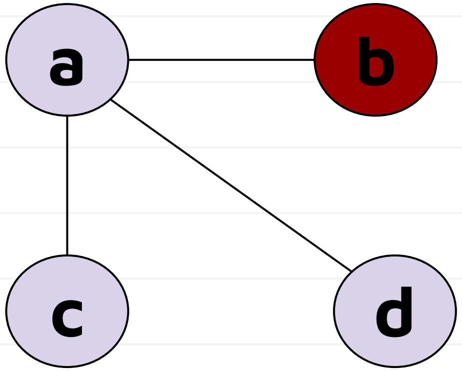
- What is the **greatest** number edges possible in an **undirected** graph with *n* nodes?



3

Counting Edges

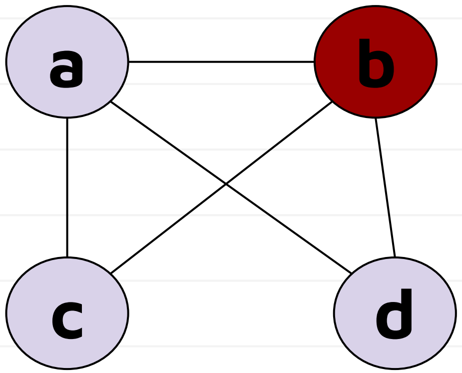
- What is the **greatest** number edges possible in an **undirected** graph with *n* nodes?



3

Counting Edges

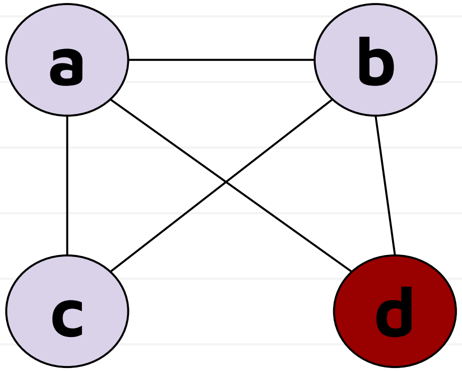
- What is the **greatest** number edges possible in an **undirected** graph with *n* nodes?



3 + 2

Counting Edges

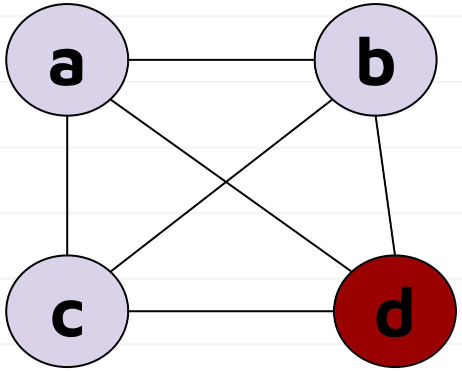
- What is the **greatest** number edges possible in an **undirected** graph with *n* nodes?



3 + 2

Counting Edges

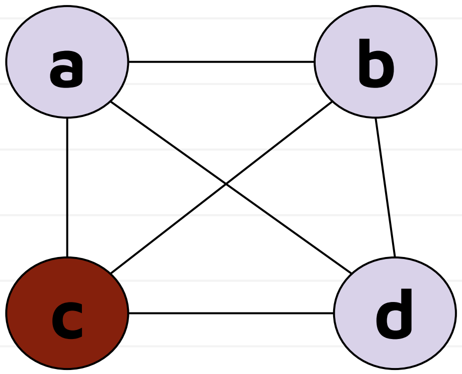
- What is the **greatest** number edges possible in an **undirected** graph with *n* nodes?



3 + 2 + 1

Counting Edges

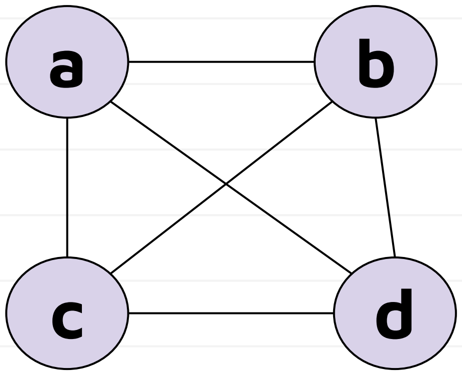
- What is the **greatest** number edges possible in an **undirected** graph with *n* nodes?



3 + 2 + 1 + 0

Counting Edges

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

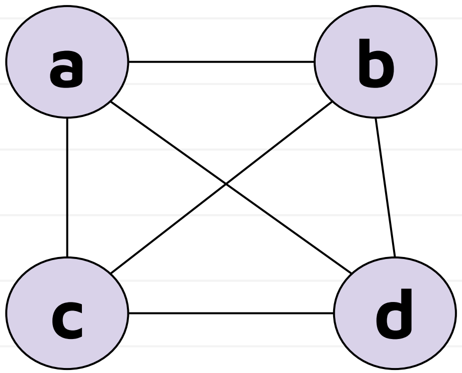


3 + 2 + 1 + 0 = 6

Counting Edges

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

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

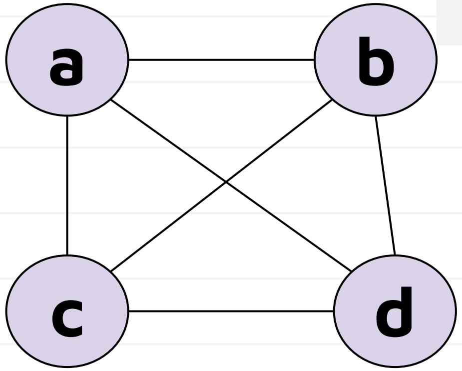


3 + 2 + 1 + 0 = 6

Counting Edges

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

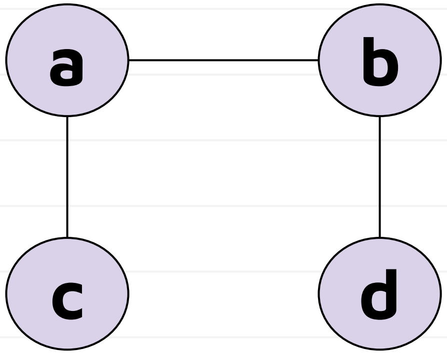
(n-1) + (n-2) + …. + 1 = **n(n-1)/2**



3 + 2 + 1 + 0 = 6

Degree

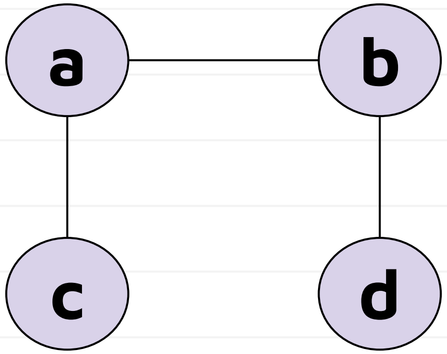
The degree of a node in an *undirected* graph is the **number of edges containing that node.**



Degree

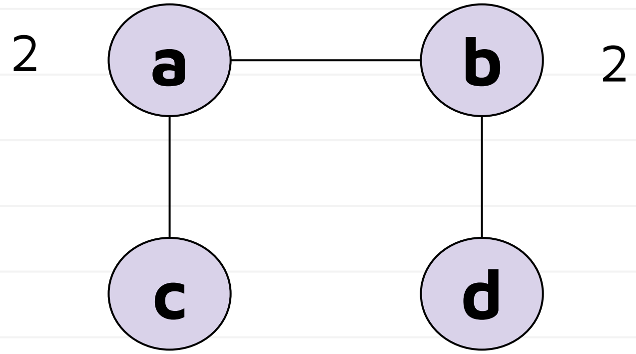
The degree of a node in an *undirected* graph is the **number of edges containing that node.**

2



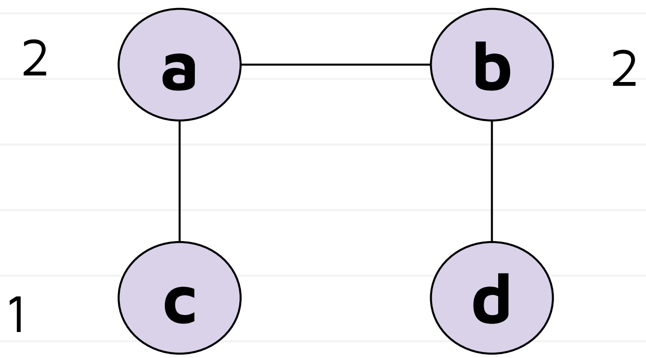
Degree

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



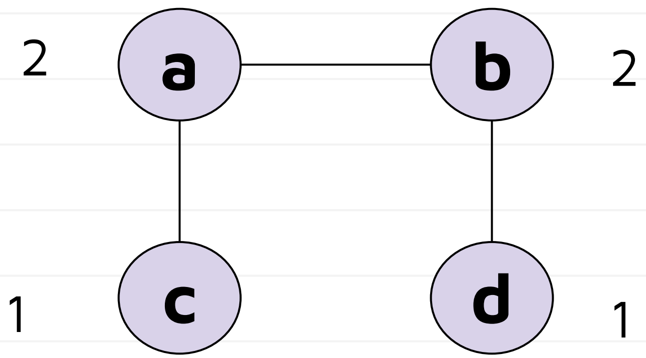
Degree

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

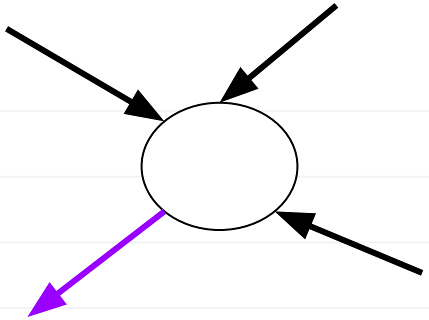


Degree

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

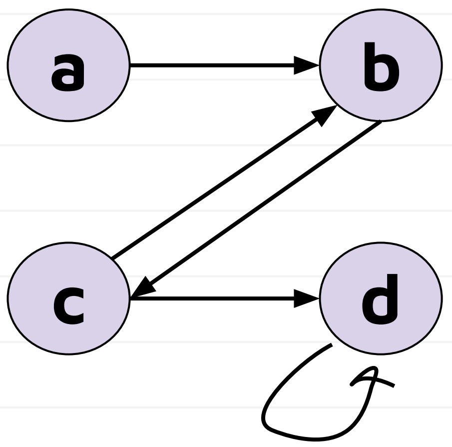


In-Degree/Out-Degree



- 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)

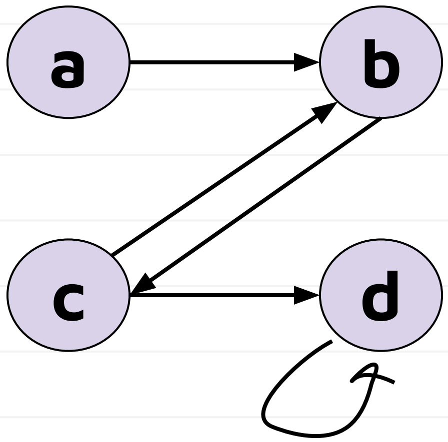
Examples



What are different degrees for node d?

	Out-degree,	in-degree,	degree
A:	**1**	**1**	**2**
B:	**3**	**1**	**2**
C:	**2**	**2**	**4**
D:	**1**	**2**	**2**
E:	**None of the above**		

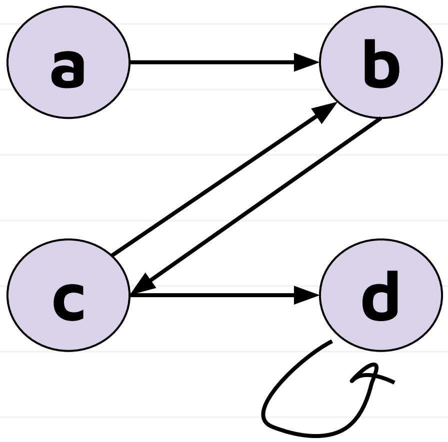
Examples



Out: 1
In: 2
D: 3

Examples

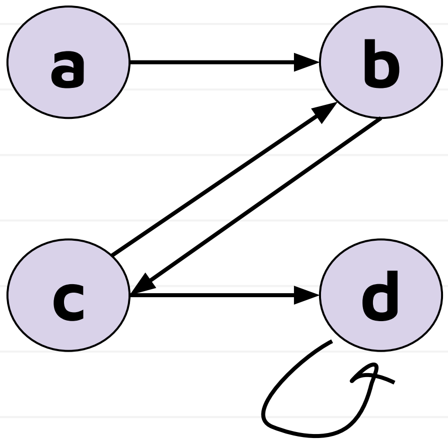
Out:
In:
D:



Out: 1
In: 2
D: 3

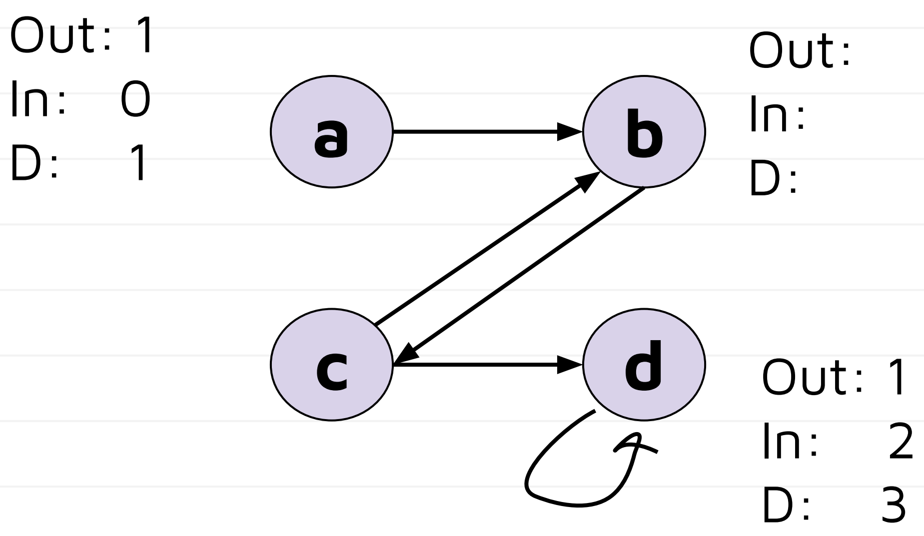
Examples

Out: 1
In: 0
D: 1

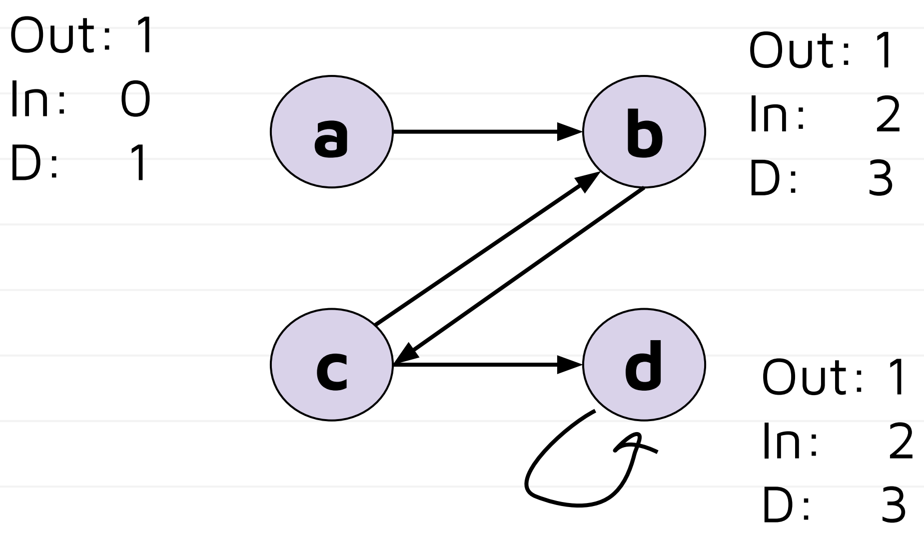


Out: 1
In: 2
D: 3

Examples

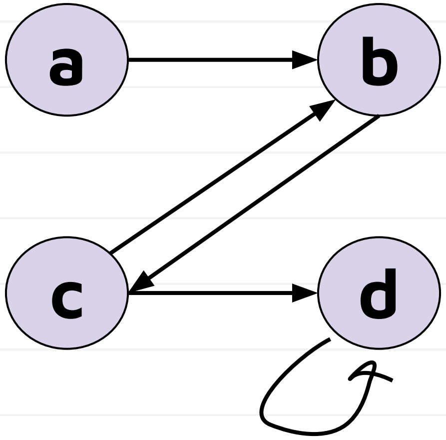


Examples



Examples

Out: 1
In: 0
D: 1



Out: 1
In: 2
D: 3

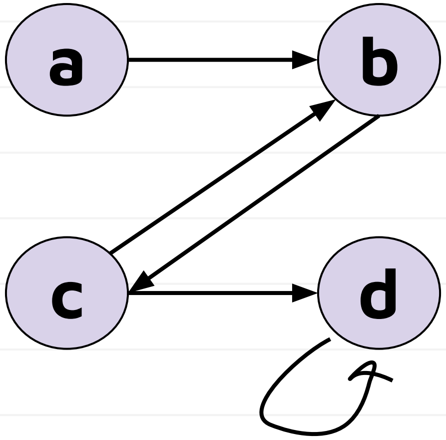
Out:
In:
D:

Out: 1
In: 2
D: 3

Examples

Out: 1
In: 0
D: 1

Out: 1
In: 2
D: 3

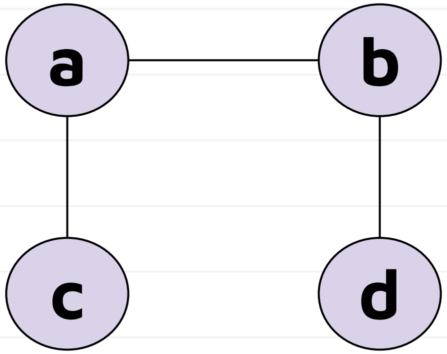


Out: 2
In: 1
D: 3

Out: 1
In: 2
D: 3

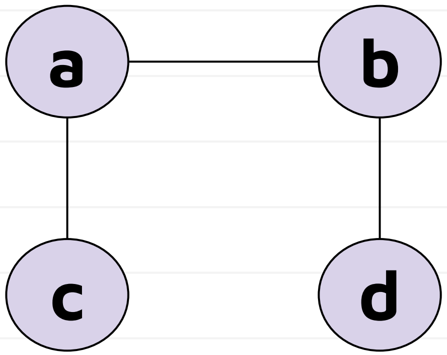
Neighbors

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

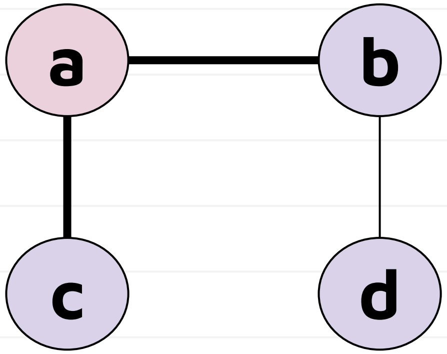
Definition: in an *undirected* graph, the set of neighbors of a node u is the set of all nodes which *share* an edge with u.



neighbors(a) = { }

Neighbors

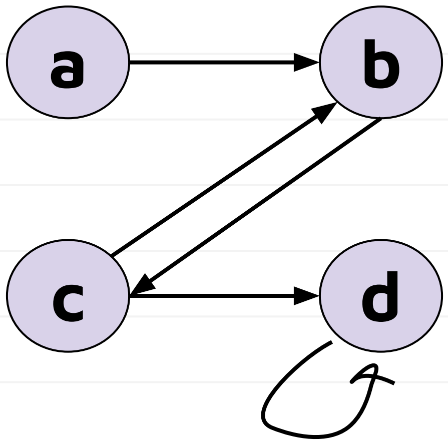
Definition: in an *undirected* graph, the set of neighbors of a node u is the set of all nodes which ***share*** an edge with u.



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}

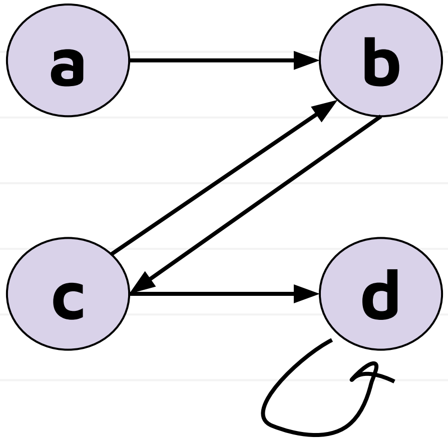
B: {a}

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}

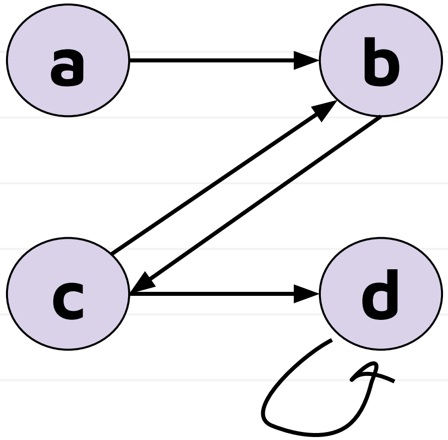
B: {a}

C: {a, c}

D: {a, b, c}

Predecessors

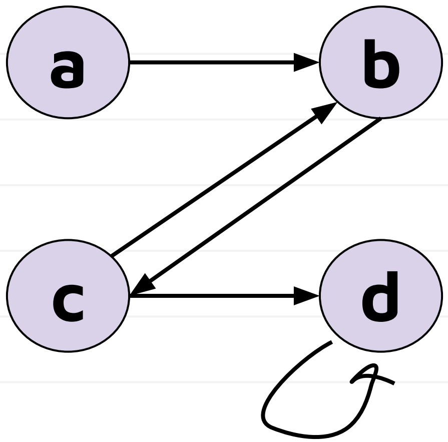
Definition: in an *directed* graph, the set of predecessors of a node u is the set of all nodes which are at the **start** of an edge **entering** u.



predecessors(a) = { }

Predecessors

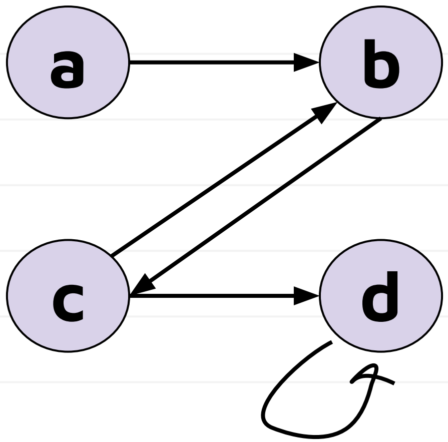
Definition: in an *directed* graph, the set of predecessors of a node u is the set of all nodes which are at the **start** of an edge **entering** u.



predecessors(d) = { ? }

Predecessors

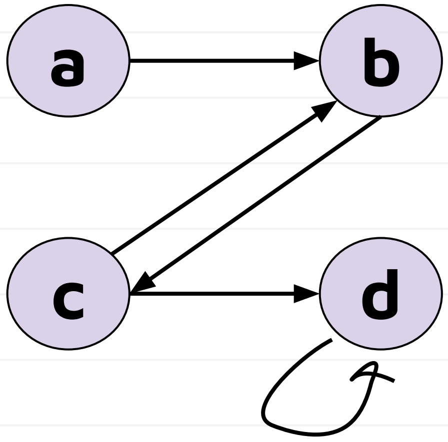
Definition: in an *directed* graph, the set of predecessors of a node u is the set of all nodes which are at the **start** of an edge **entering** u.



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}

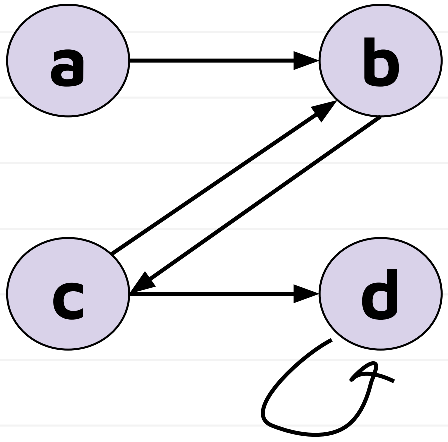
B: {c}

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}

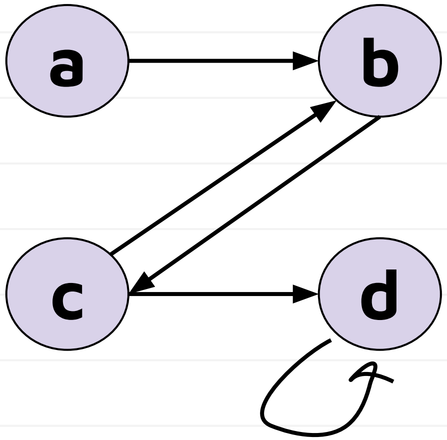
B: {c}

C: {b, d}

D: {a, b, d}

Successors

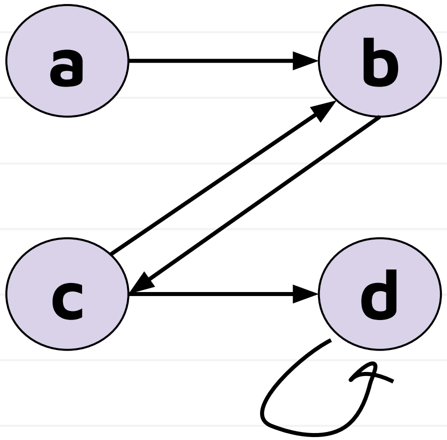
Definition: in an *directed* graph, the set of *successors* of a node u is the set of all nodes which are at the **end** of an edge **leaving** u.



successors **(a) = ?**

Successors

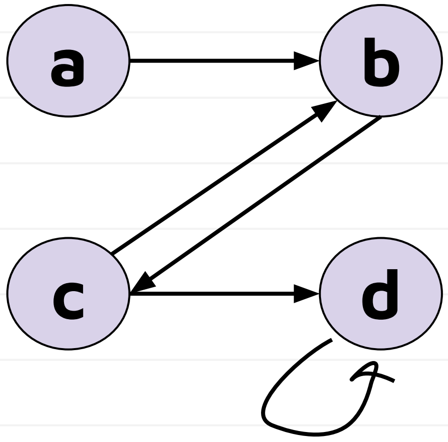
Definition: in an *directed* graph, the set of *successors* of a node u is the set of all nodes which are at the **end** of an edge **leaving** u.



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

Successors

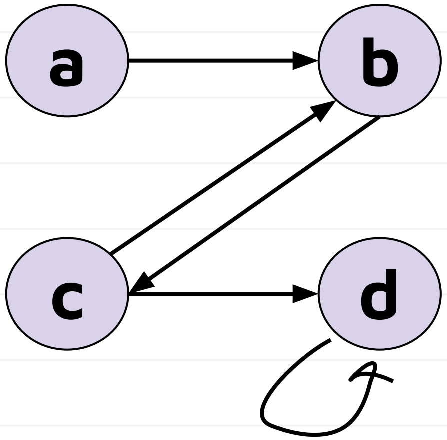
Definition: in an *directed* graph, the set of *successors* of a node u is the set of all nodes which are at the **end** of an edge **leaving** u.



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

Successors

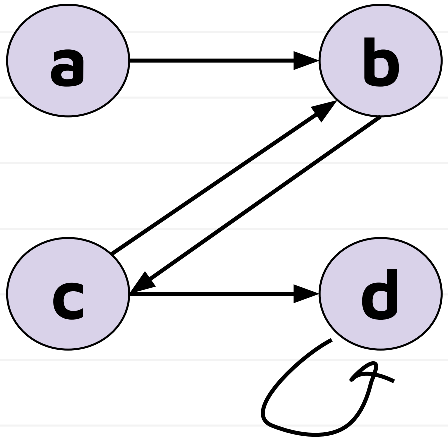
Definition: in an *directed* graph, the set of *successors* of a node u is the set of all nodes which are at the **end** of an edge **leaving** u.



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

Successors

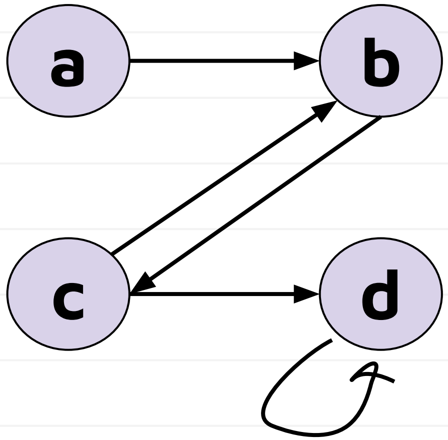
Definition: in an *directed* graph, the set of *successors* of a node u is the set of all nodes which are at the **end** of an edge **leaving** u.



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

Successors

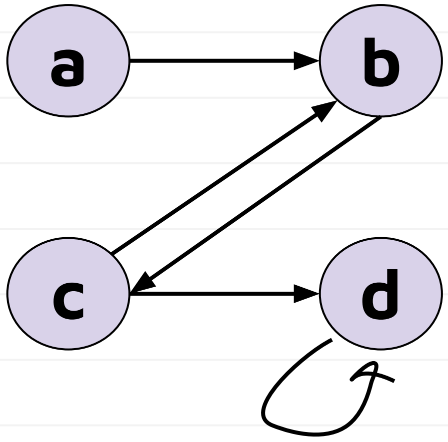
Definition: in an *directed* graph, the set of *successors* of a node u is the set of all nodes which are at the **end** of an edge **leaving** u.



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

A Convention

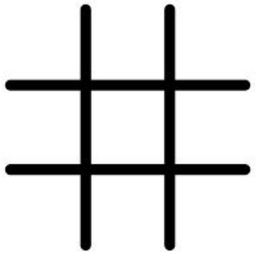
- In a **directed** graph, the **neighbors** of u are the **successors** of u.

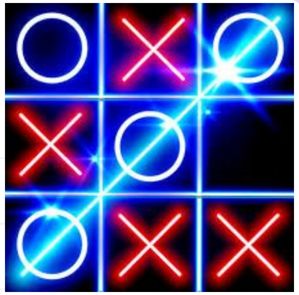


Other Graphs

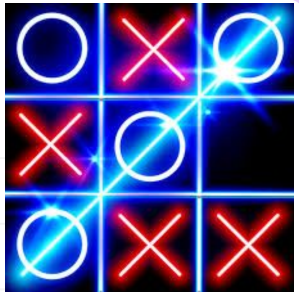
- Graphs can be used to represent states of a process, system, game, etc.
- They could (in principle) have infinitely-many nodes and edges.

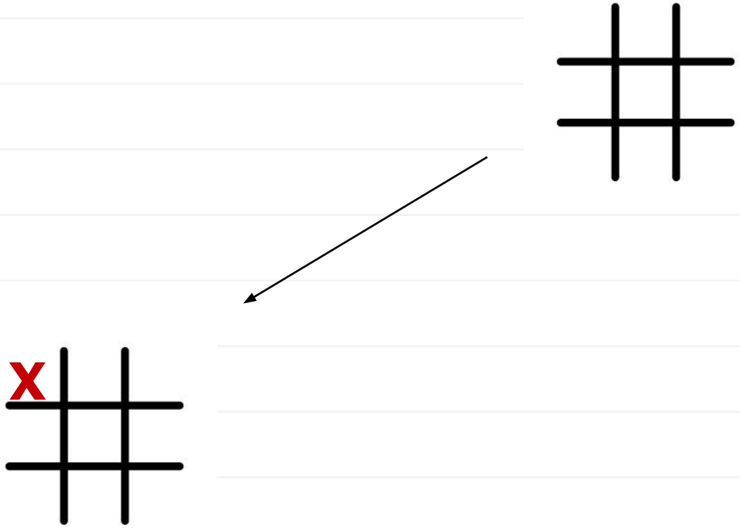
Example: Tic-Tac-Toe



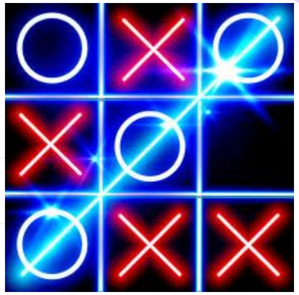


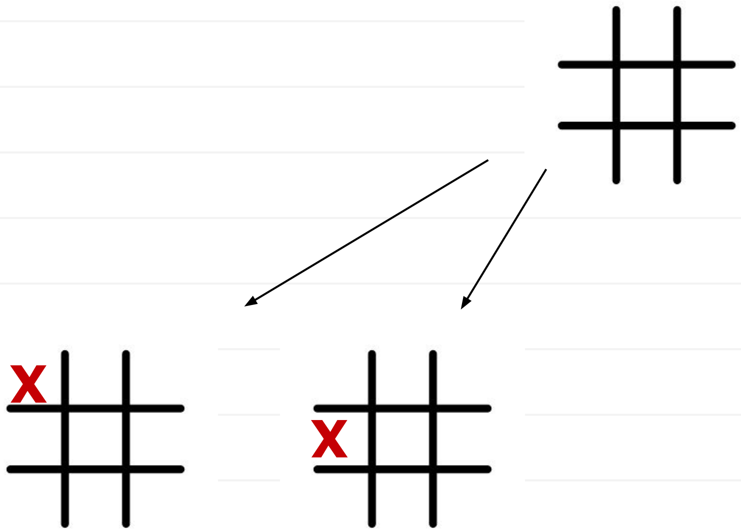
Example: Tic-Tac-Toe



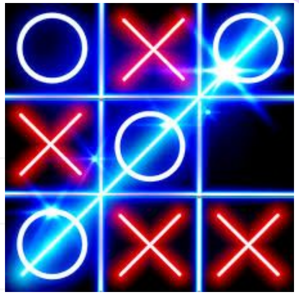


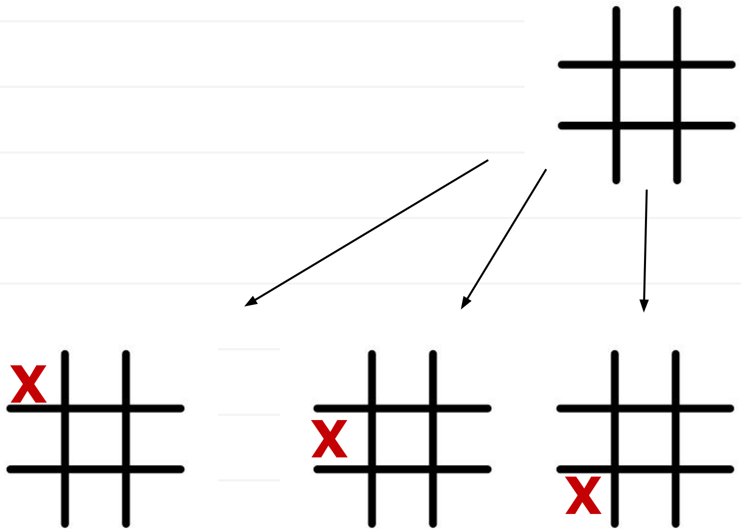
Example: Tic-Tac-Toe



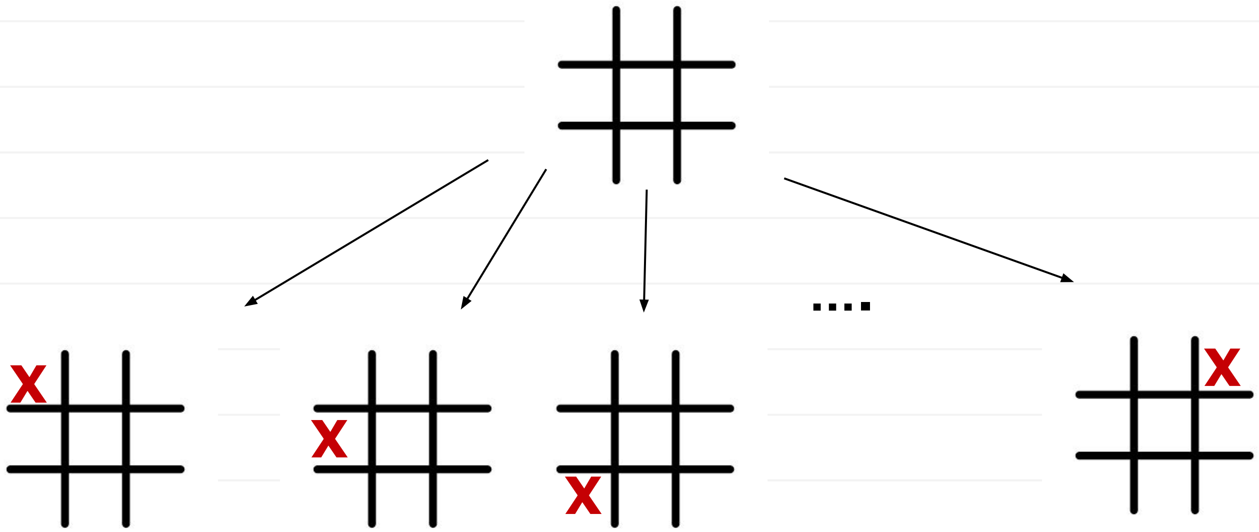


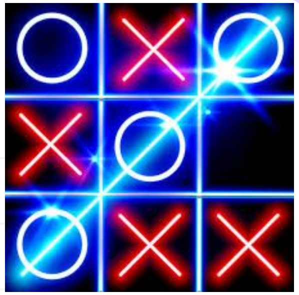
Example: Tic-Tac-Toe



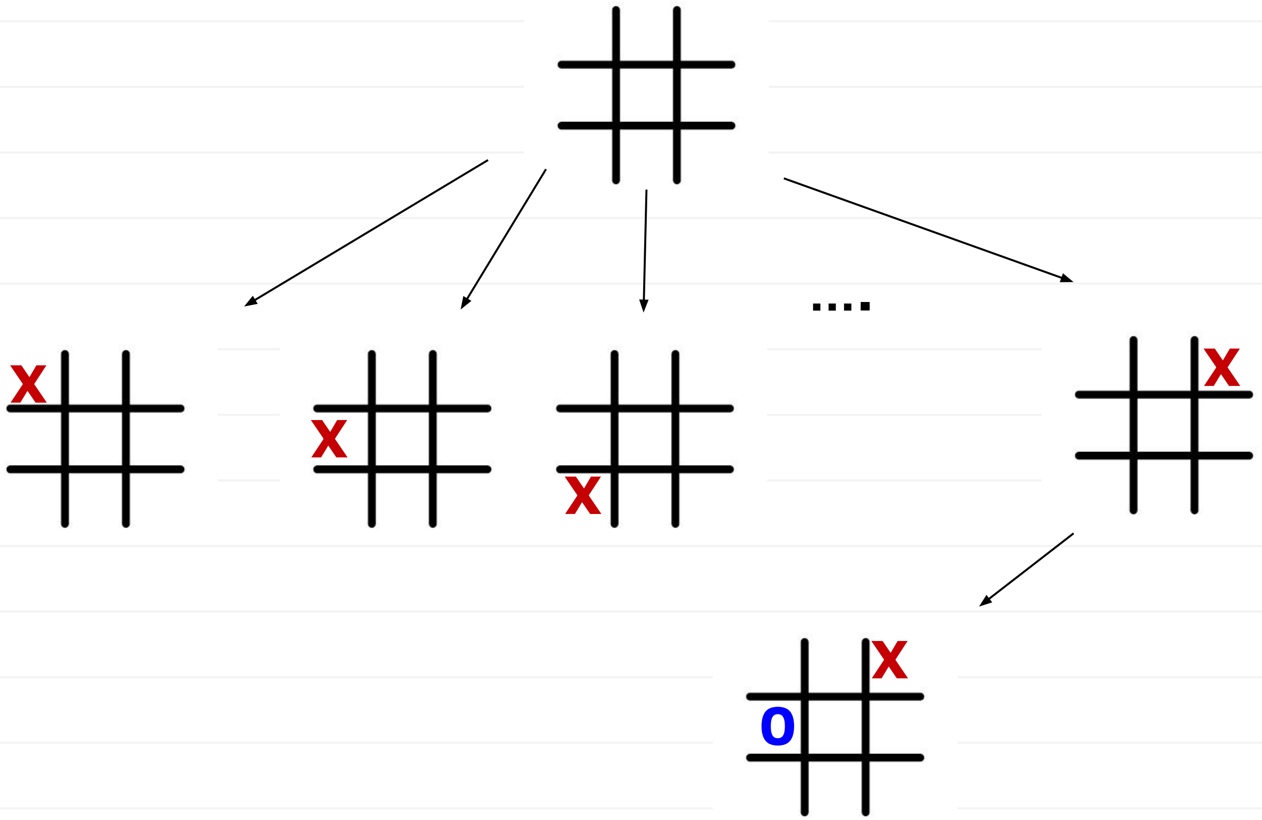


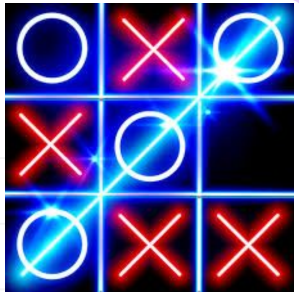
Example: Tic-Tac-Toe



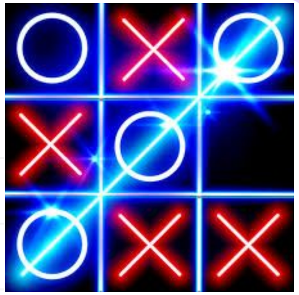


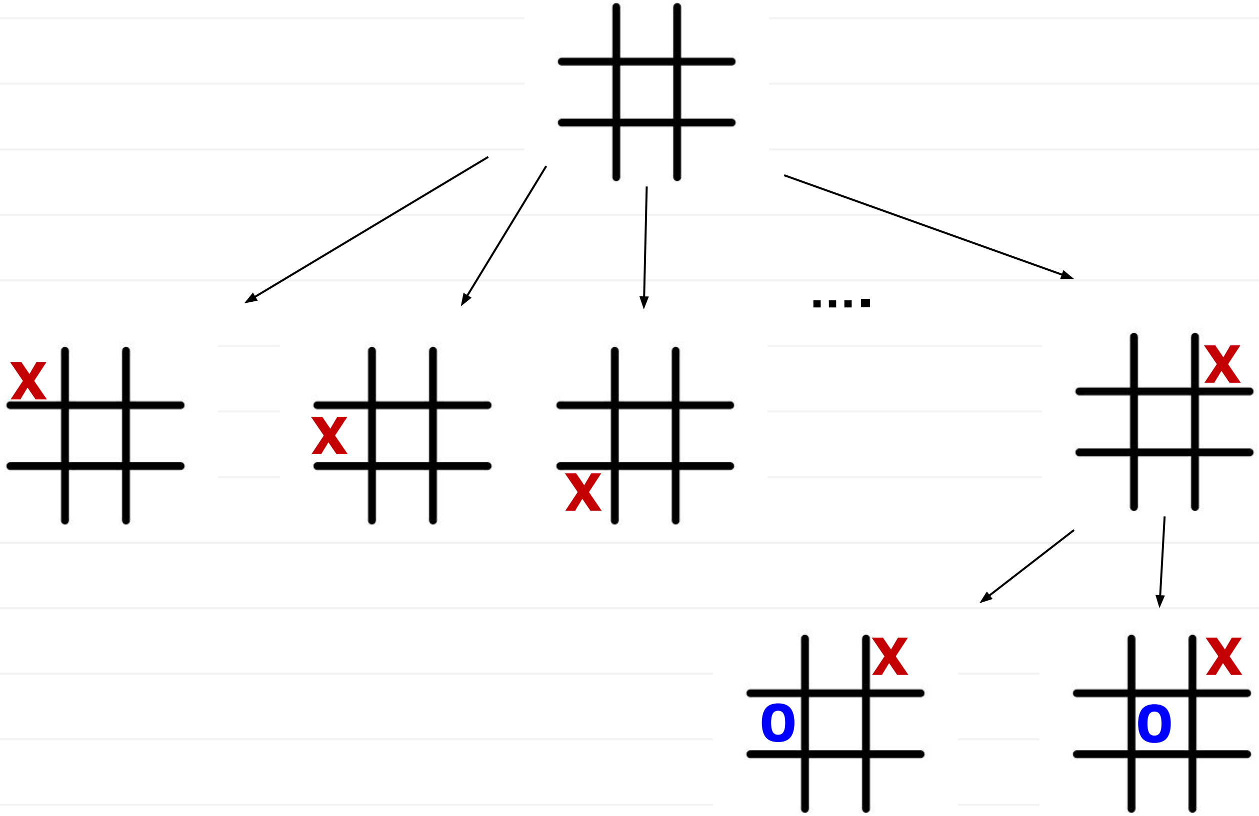
Example: Tic-Tac-Toe



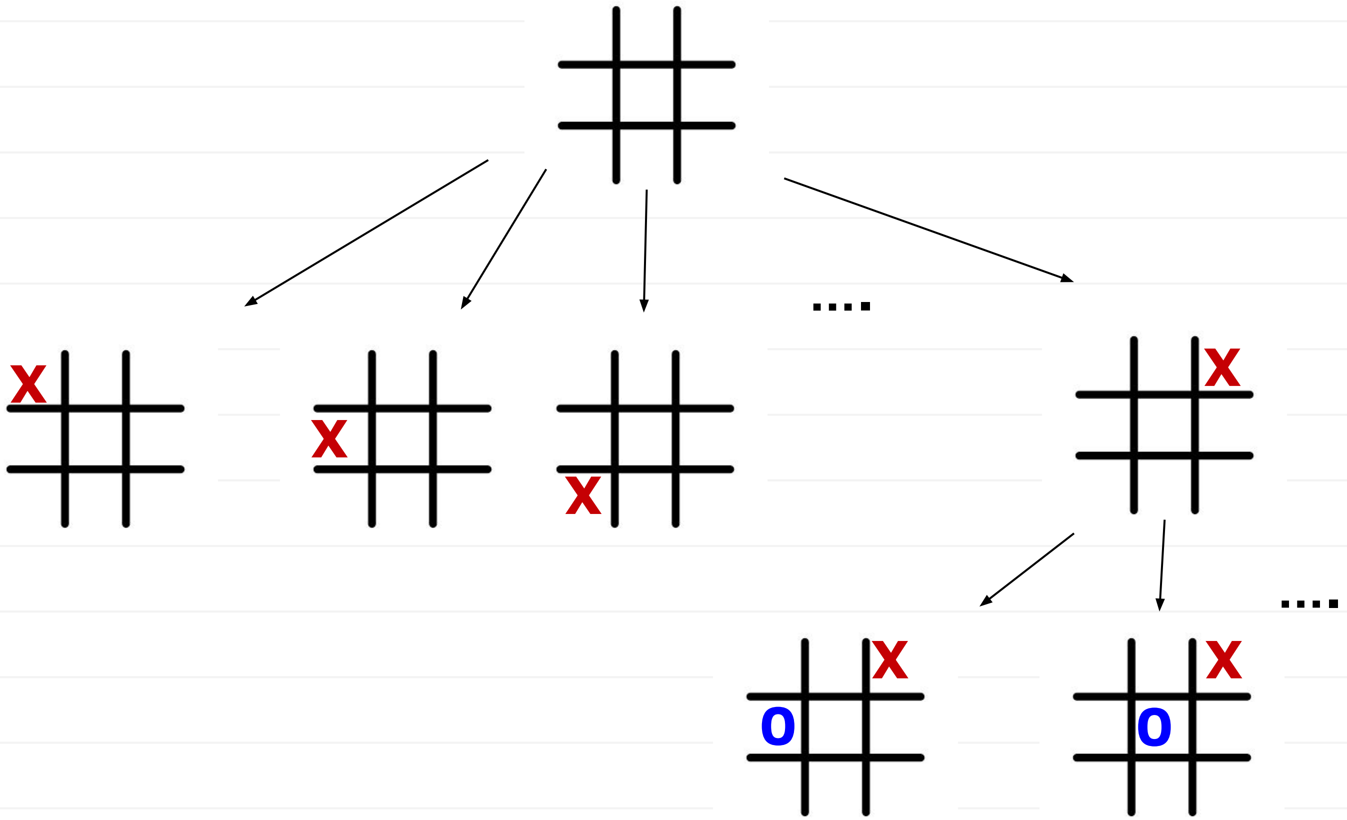


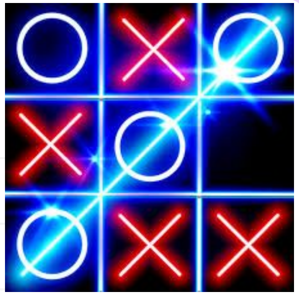
Example: Tic-Tac-Toe



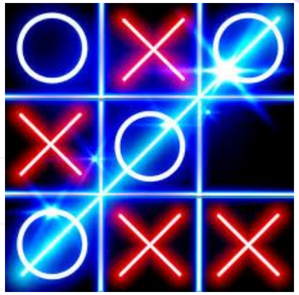


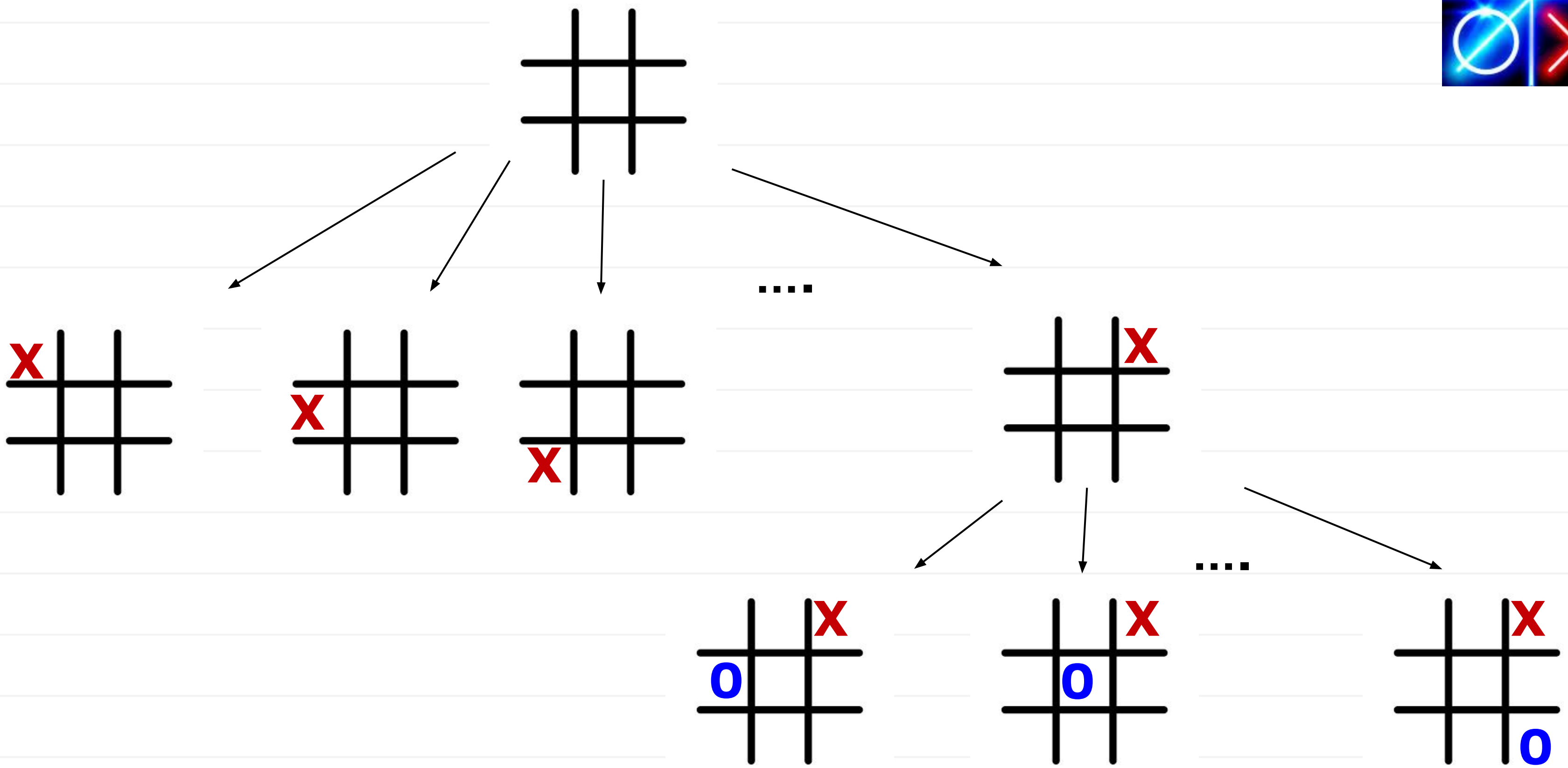
Example: Tic-Tac-Toe



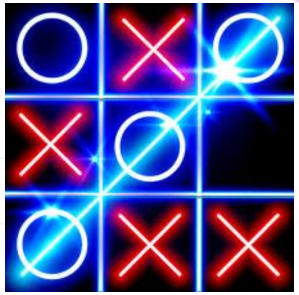


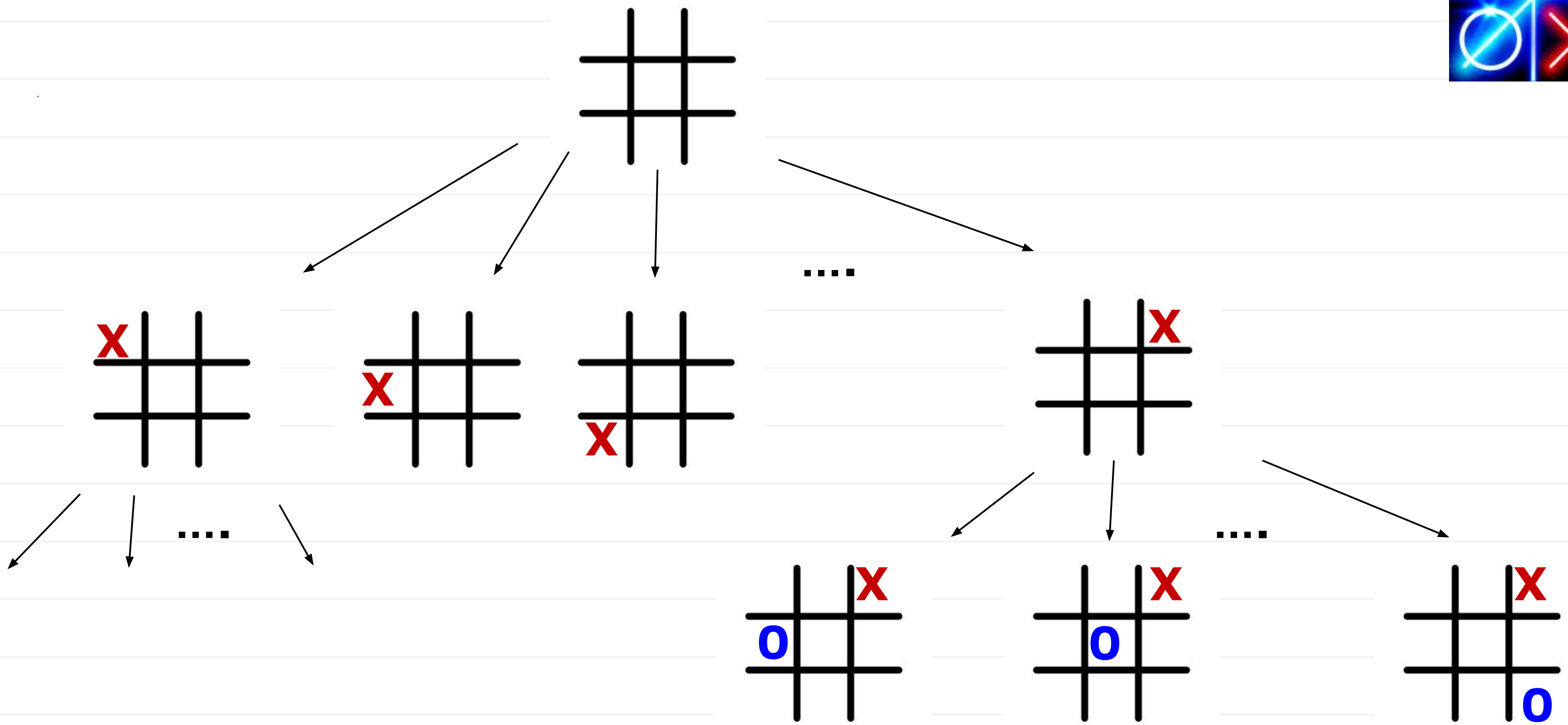
Example: Tic-Tac-Toe



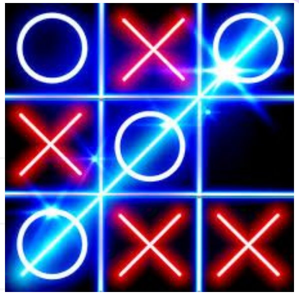


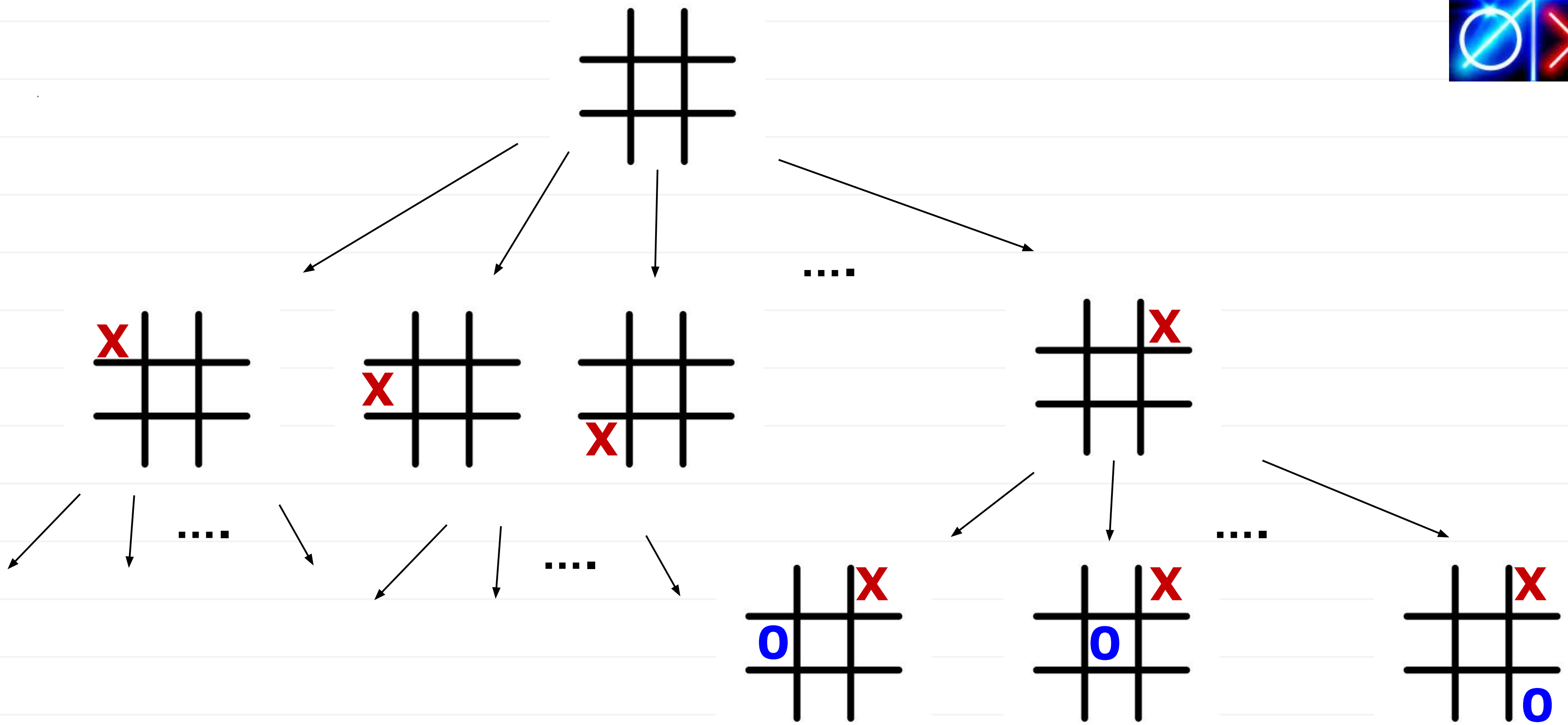
Example: Tic-Tac-Toe



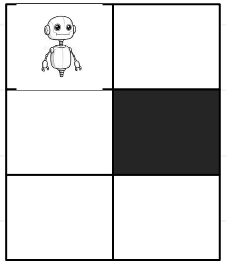


Example: Tic-Tac-Toe

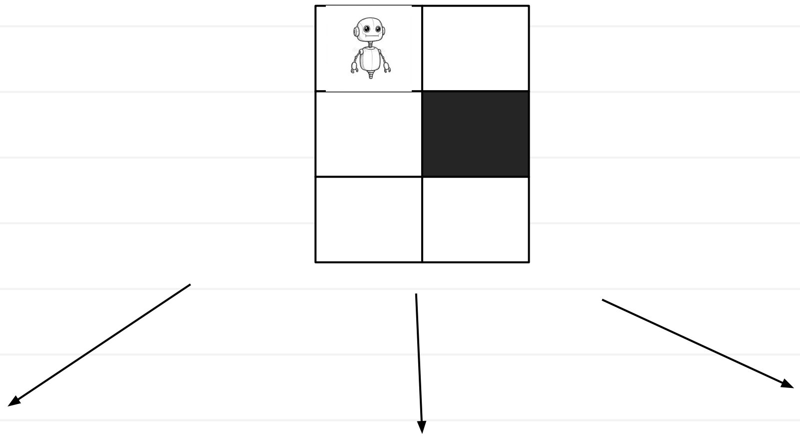




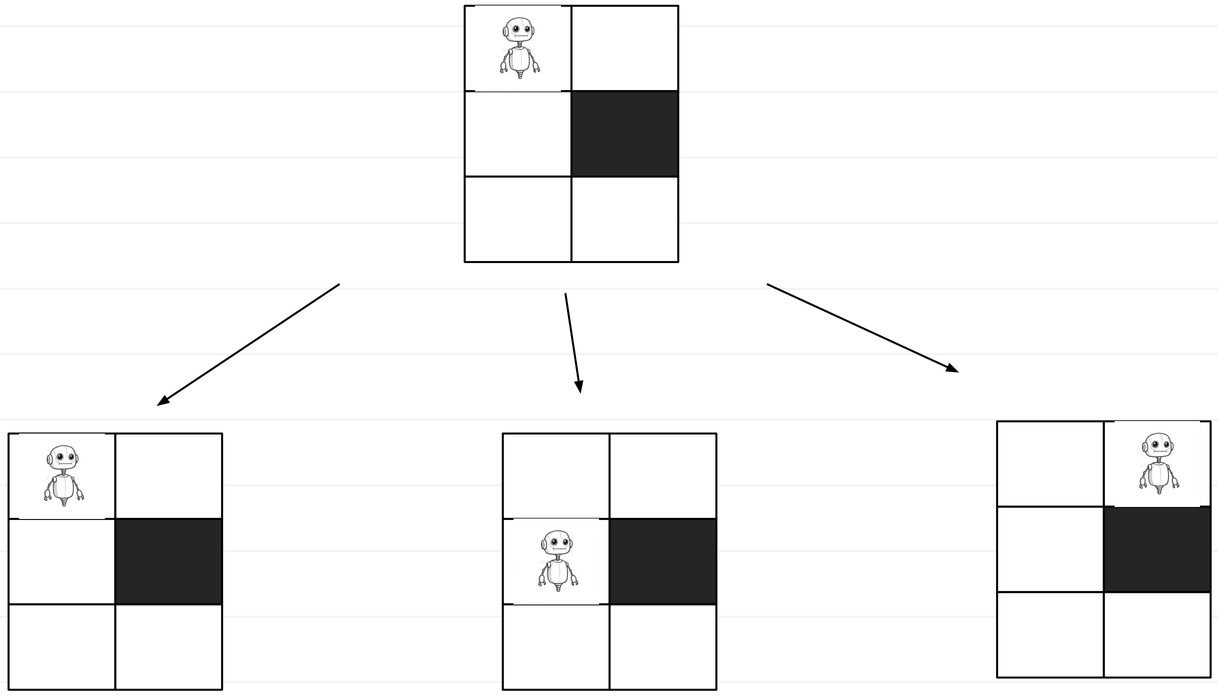
Example: Robot Navigation



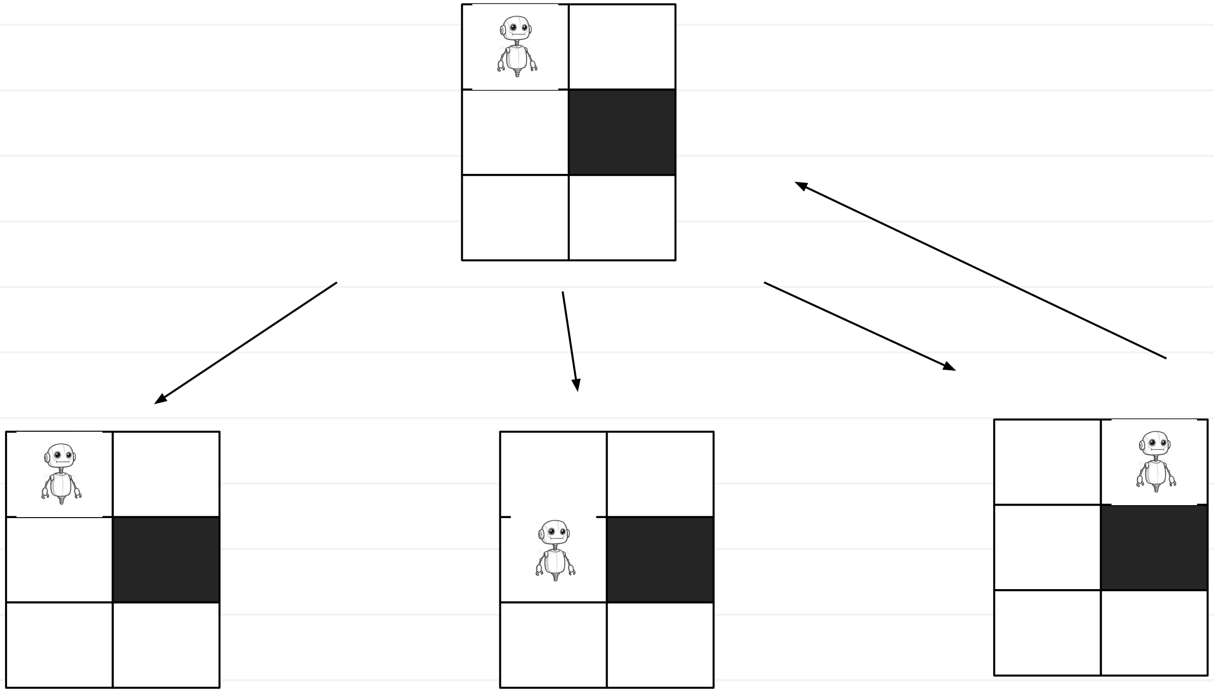
Example: Robot Navigation



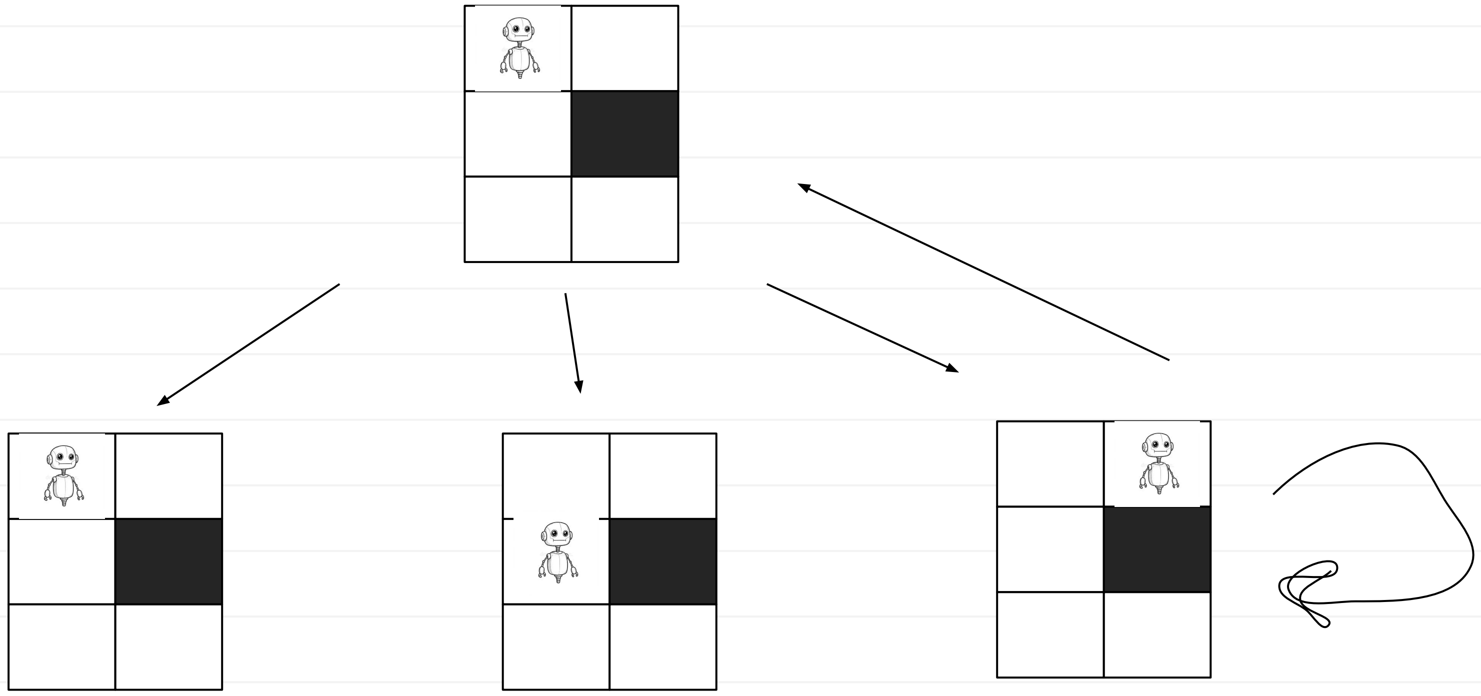
Example: Robot Navigation



Example: Robot Navigation



Example: Robot Navigation

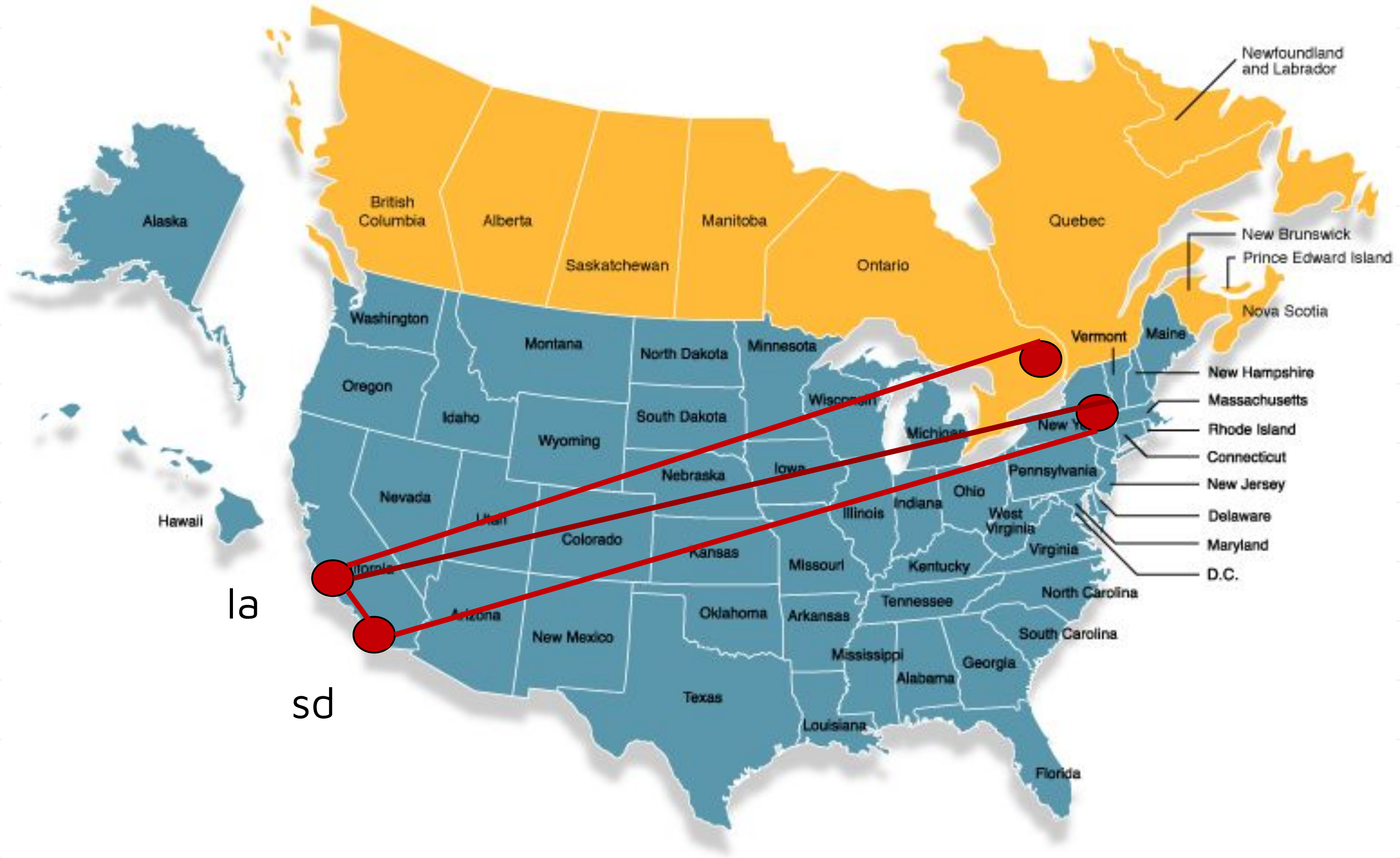


Paths

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



Alaska
Hawaii
British Columbia
Alberta
Saskatchewan
Manitoba
Ontario
Quebec
Newfoundland and Labrador
New Brunswick
Prince Edward Island
Nova Scotia
Washington
Oregon
Idaho
Montana
North Dakota
South Dakota
Wyoming
Nevada
Colorado
Nebraska
Kansas
Minnesota
Iowa
Missouri
Illinois
Indiana
Ohio
Pennsylvania
West Virginia
Virginia
Kentucky
Tennessee
North Carolina
South Carolina
Georgia
Alabama
Mississippi
Arkansas
Oklahoma
New Mexico
Texas
Louisiana
Florida
Vermont
Maine
New Hampshire
Massachusetts
Rhode Island
Connecticut
New Jersey
Delaware
Maryland
D.C.
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

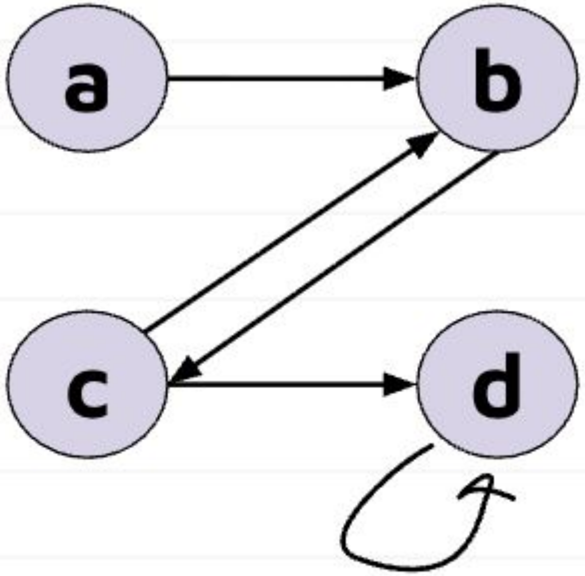
Definition: The length of a path is the number of nodes in the sequence, minus one. Paths of length zero are possible.

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.

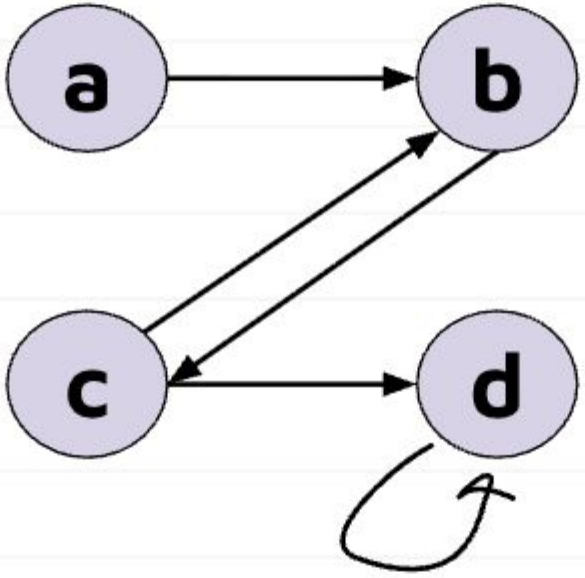
Examples



Paths	Not Paths

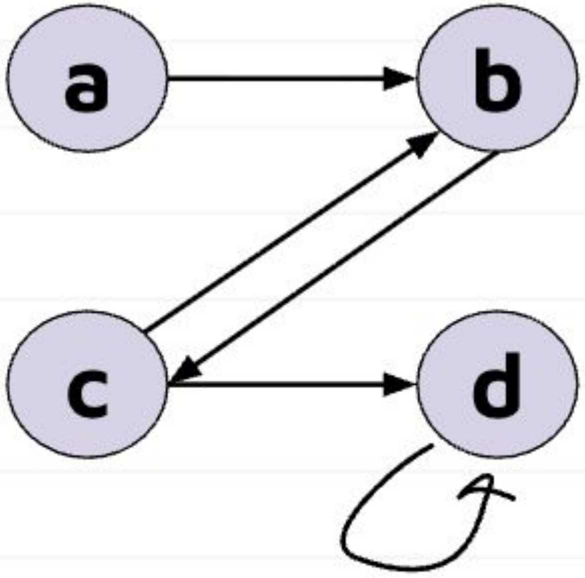
(a, b, c)

Examples



Paths	Not Paths
(a, b, c)	

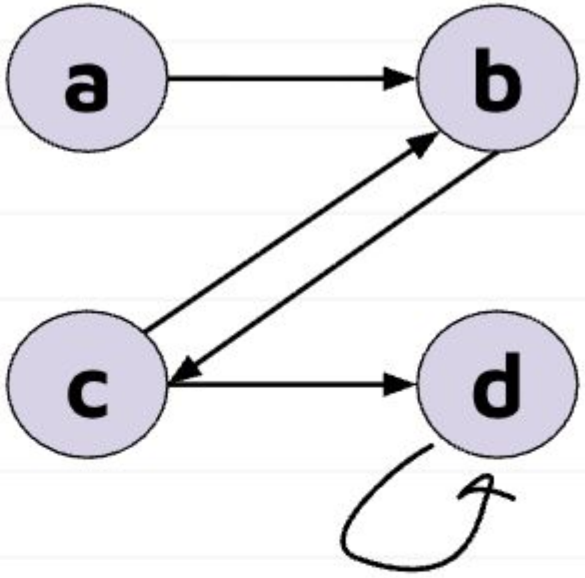
Examples



Paths	Not Paths
(a, b, c)	

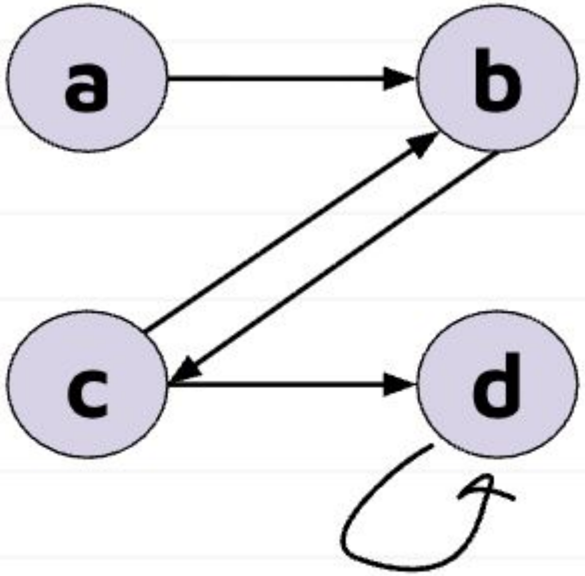
(a, b, c, b, c)

Examples



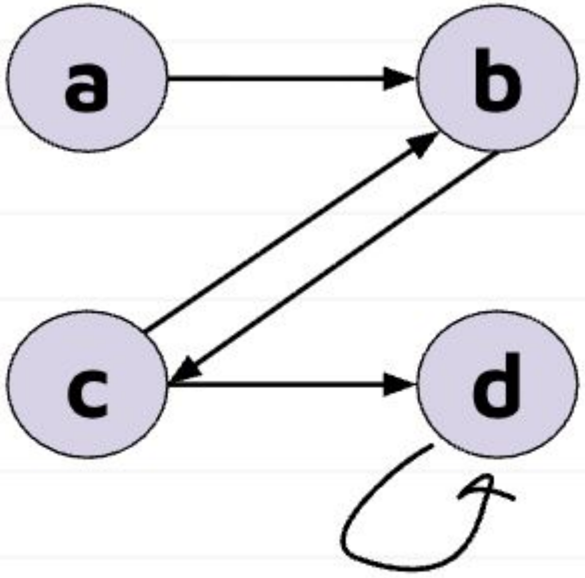
Paths	Not Paths
(a, b, c)	
(a, b, c, b, c)	

Examples



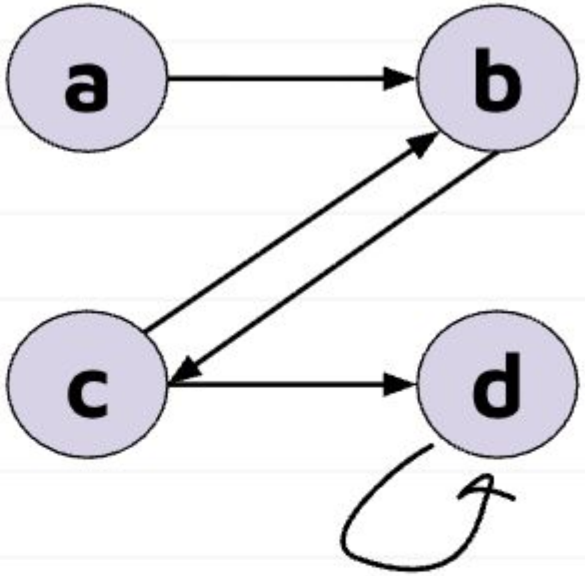
Paths	Not Paths
(a, b, c)	
(a, b, c, b, c)	
(a, b, c, b, c, b, c, b, c)	

Examples



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

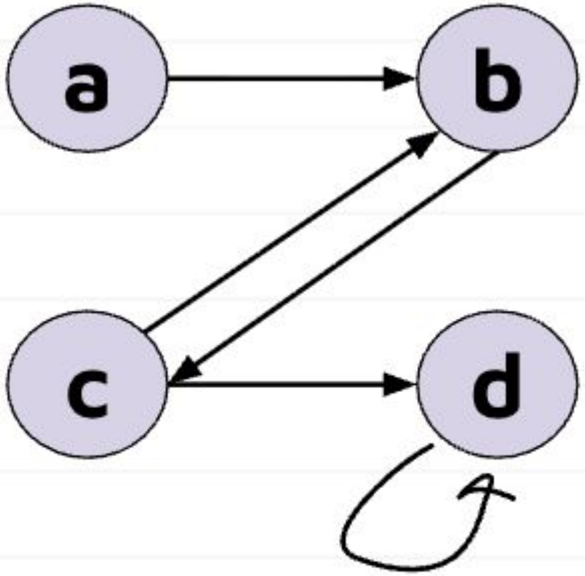
Examples



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

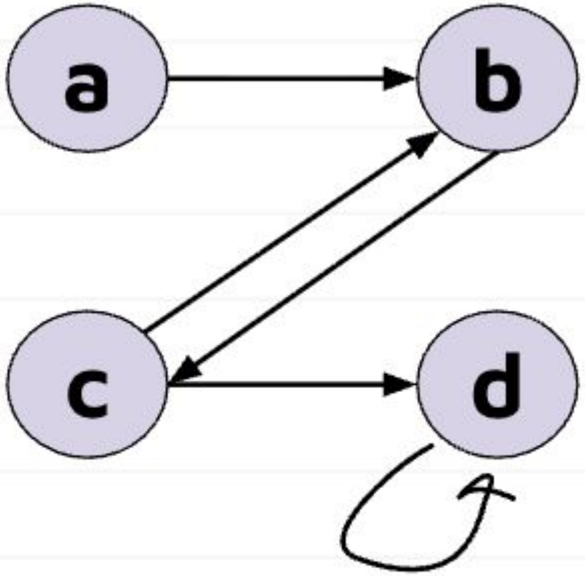
(a, b, d)

Examples



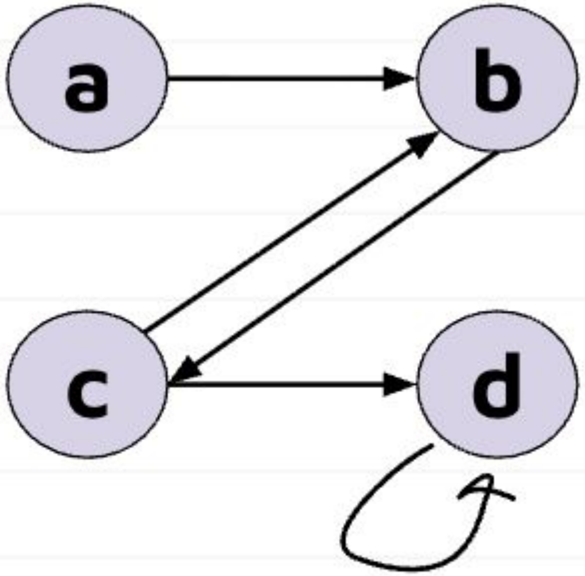
Paths	Not Paths
(a, b, c)	(a, b, d)
(a, b, c, b, c)	
(a, b, c, b, c, b, c, b, c)	

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)	

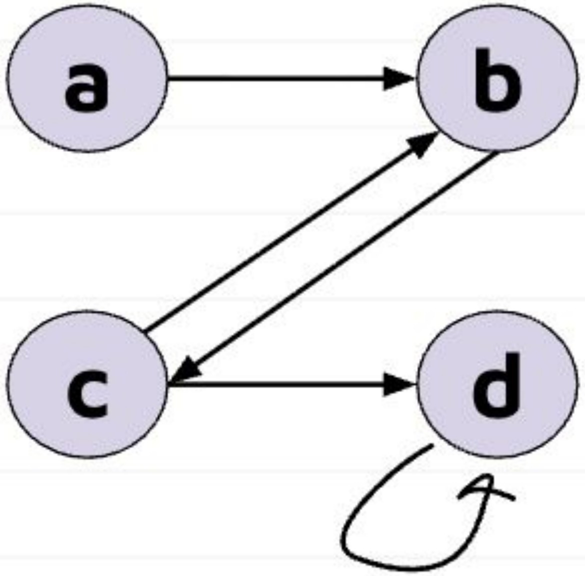
Examples



(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)	

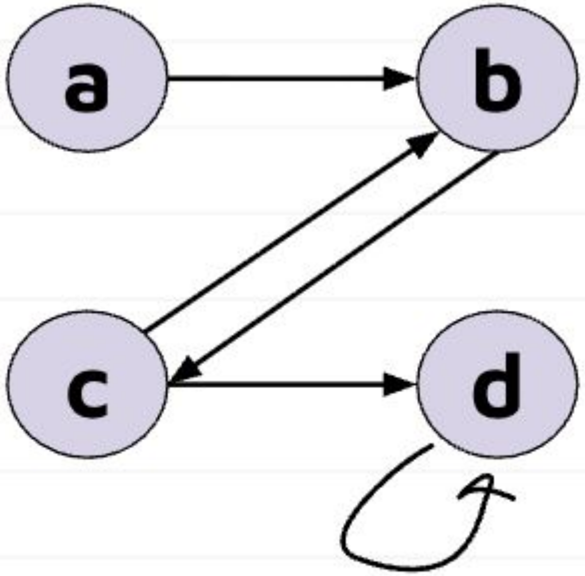
Examples



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)	

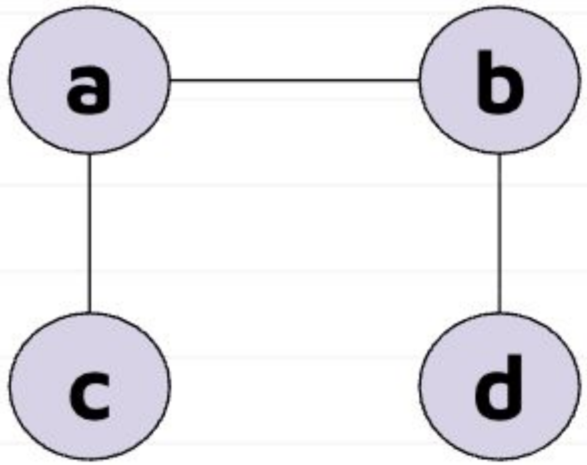
(a, a)

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)	

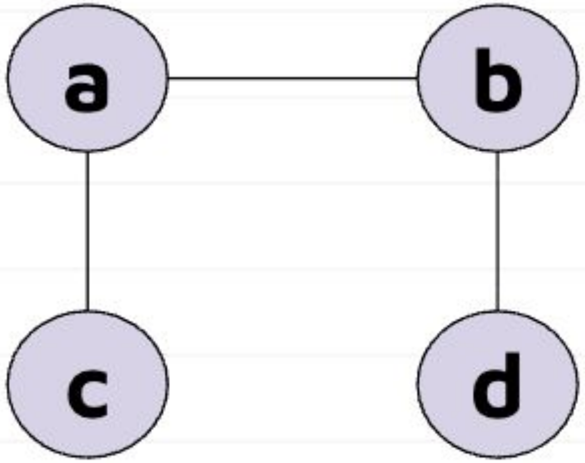
Examples



Note

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

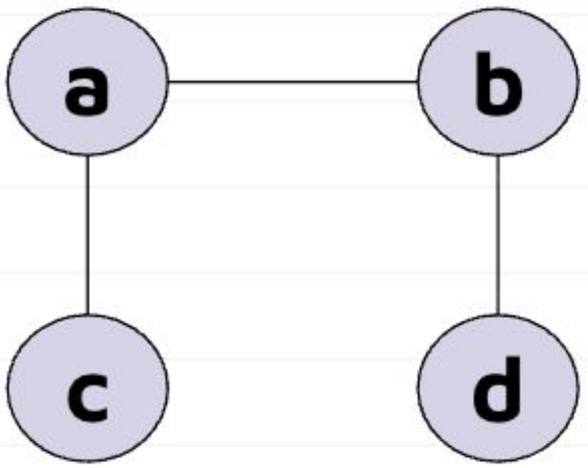
(c, **a**, **b**, **a**, **b**, d)



Simple Paths

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

(c, a, b, d)

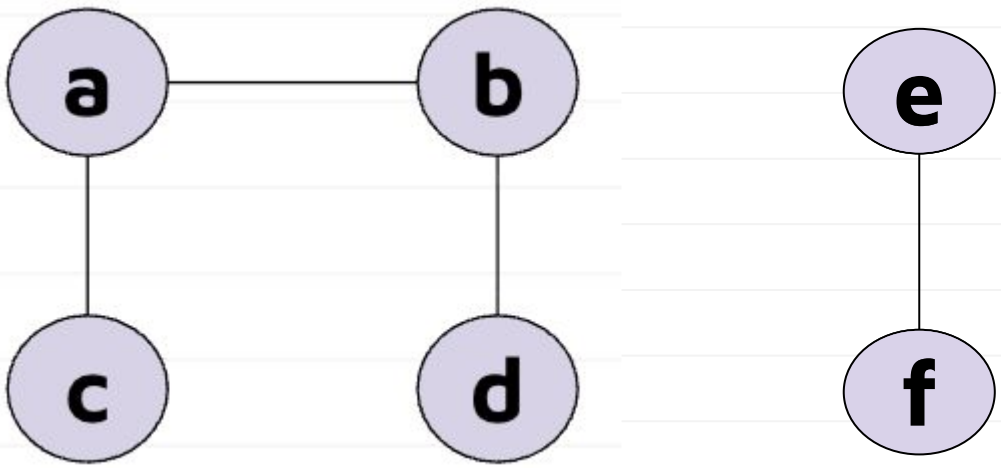


Reachability

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.

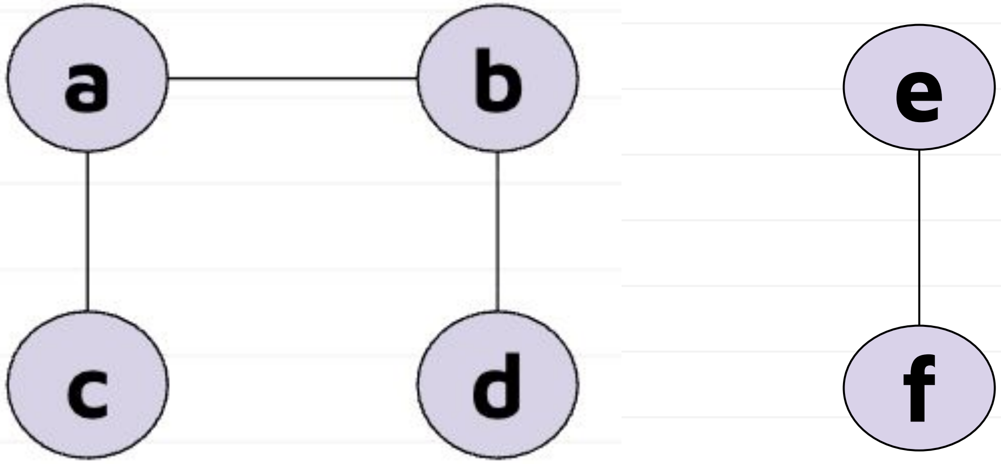
How many nodes are reachable from a?

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.

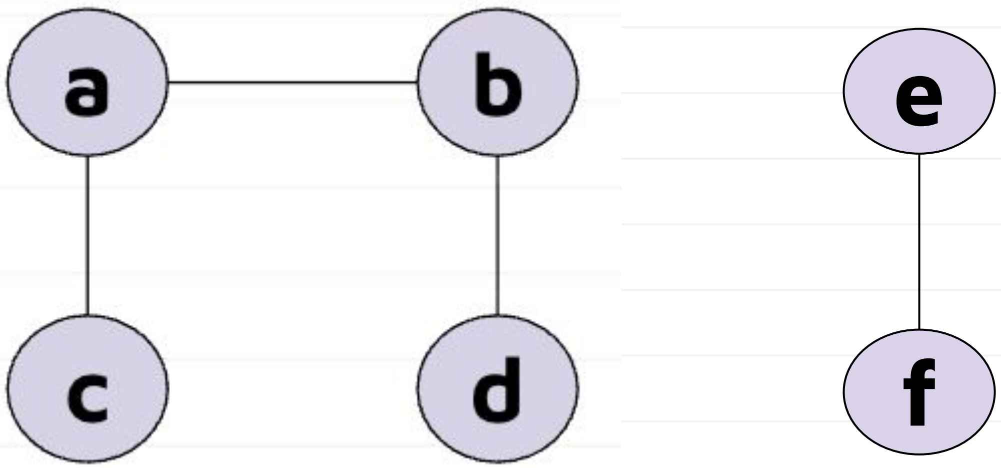
How many nodes are reachable from a?

A: 1

B: 2

C: 3

D: 4



Reachability and Directedness

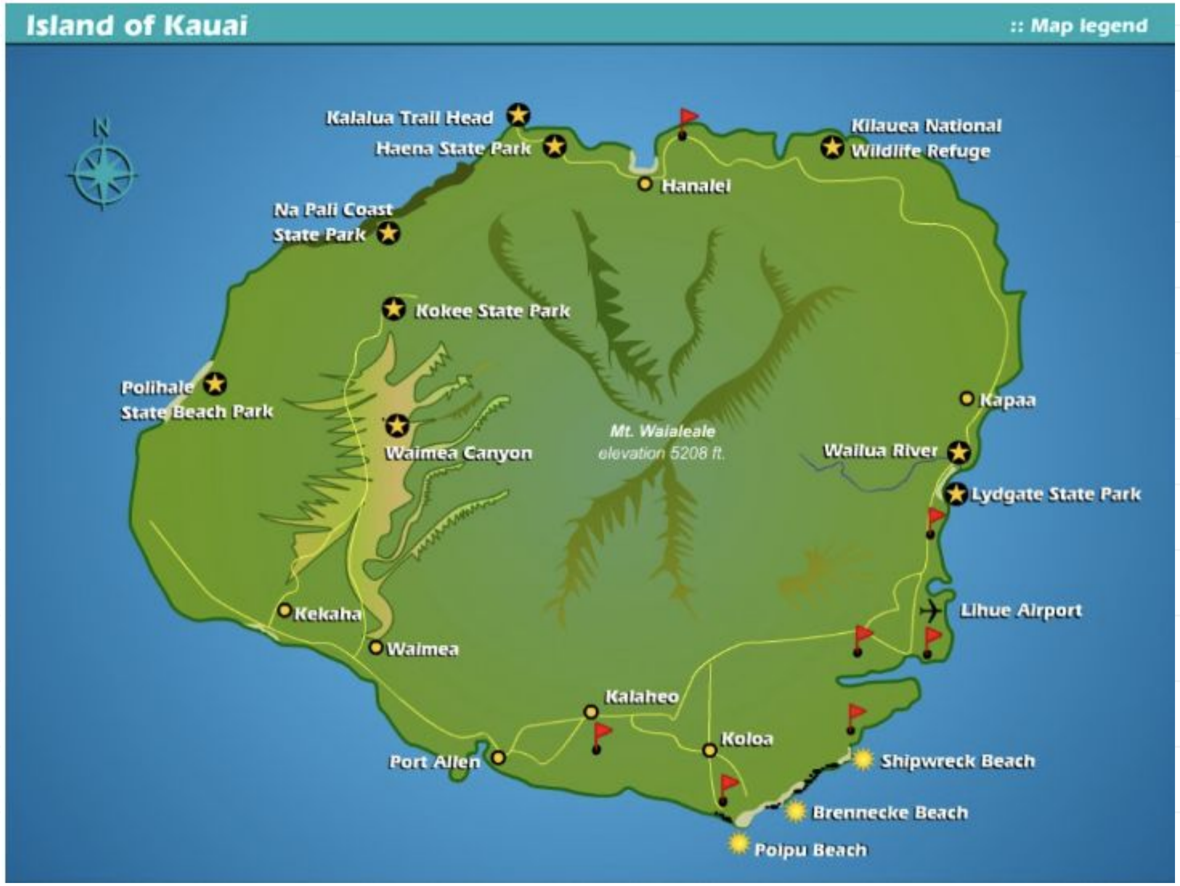
- If G is **undirected**, reachability is **symmetric**.
  - If u reachable from v, then v reachable from u.
- If G is **directed**, reachability is **not symmetric**.
  - If u reachable from v, then v may/may not be reachable from u.

Important Trivia

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

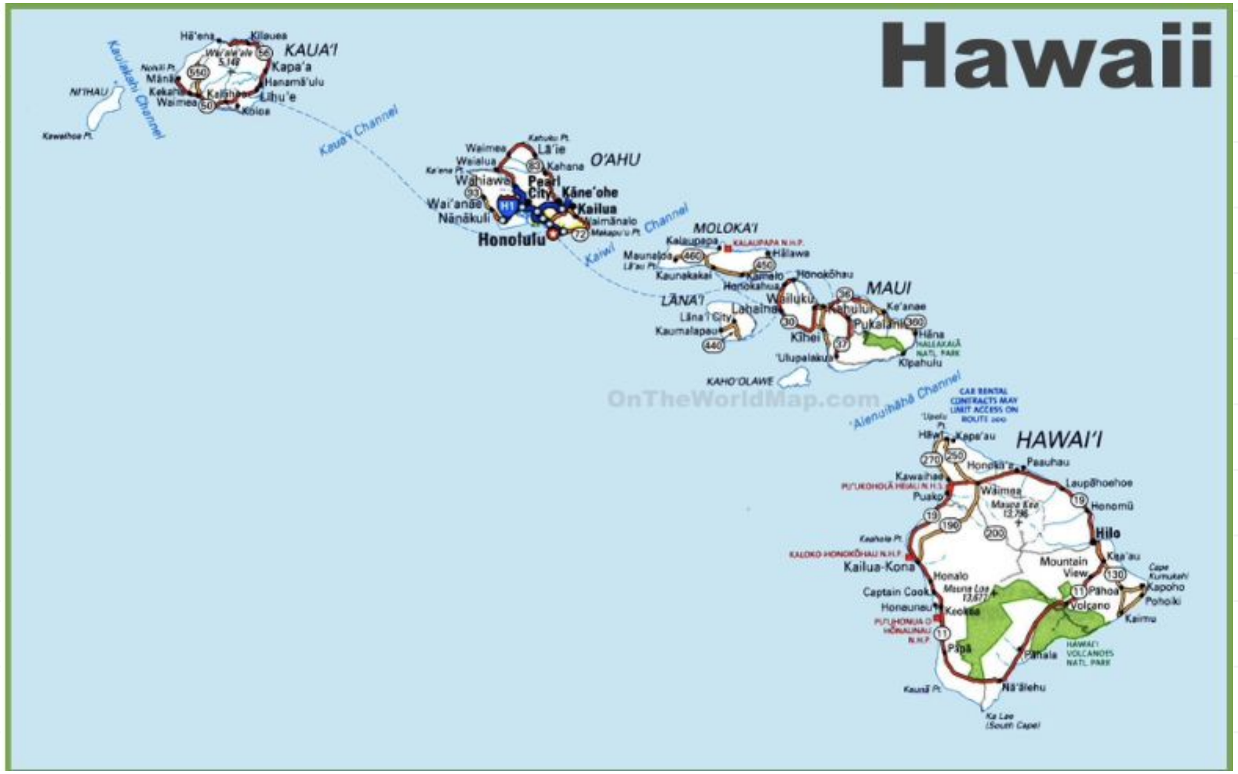
Connected Components

Example



Island of Kauai
:: Map legend
N
Kalalua Trail Head
Haena State Park
Hanalei
Kilauea National
Wildlife Refuge
Na Pali Coast
State Park
Kokee State Park
Polihale
State Beach Park
Waimea Canyon
Mt. Waialeale
elevation 5208 ft.
Kapaa
Wailua River
Lydgate State Park
Lihue Airport
Kekaha
Waimea
Kalaheo
Koloa
Port Allen
Shipwreck Beach
Brennecke Beach
Poipu Beach

Example



Hawaii
KAUA'I
Hā'ena
Kilauea
Kapa'a
Hanamā'ulu
Līhu'e
Kōloa
Waimea
Kekaha
Mānā
NI'IHAU
Kaulakahi Channel
Kaua'i Channel
O'AHU
Lā'ie
Kahana
Waimea
Waialua
Wahiawa
Pearl City
Kāne'ohe
Kailua
Wai'anae
Nānākuli
Honolulu
Waimānalo
Kaiwi Channel
MOLOKA'I
Kalaupapa
KALAUPAPA N.H.P.
Hālawa
Maunaloa
Kaunakakai
Kamalo
LĀNA'I
Lāna'i City
Kaumalapau
MAUI
Honokōhau
Honokahua
Wailuku
Lahaina
Kahului
Ke'anae
Pukalani
Hāna
HALEAKALĀ NATL. PARK
Kīhei
'Ulupalakua
Kīpahulu
KAHO'OLAWE
OnTheWorldMap.com
'Alenuihāhā Channel
CAR RENTAL CONTRACTS MAY LIMIT ACCESS ON ROUTE 200
HAWAI'I
Kapa'au
Paauhau
Kawaihae
Waimea
Laupāhoehoe
Honomū
Puako
Hilo
Mauna Kea 13,796
KALOKO-HONOKŌHAU N.H.P.
Kailua-Kona
Mountain View
Honalo
Captain Cook
Mauna Loa 13,677
Pāhoa
Kapoho
Volcano
Pohoiki
Kaimu
Keokea
Pāhala
HAWAI'I VOLCANOES NATL. PARK
Nā'ālehu
Ka Lae (South Cape)

Connectedness

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

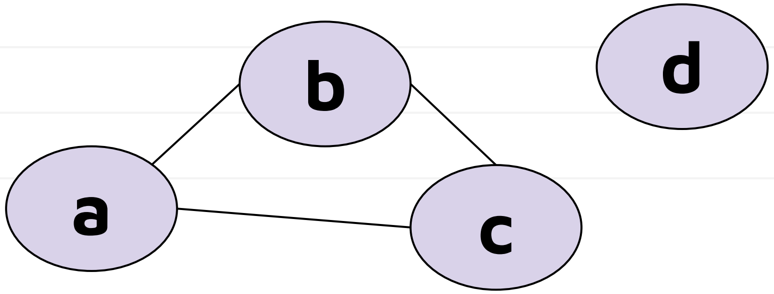
Equivalent: there is a path between every pair of nodes.

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 $C \subset V$ 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.

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.

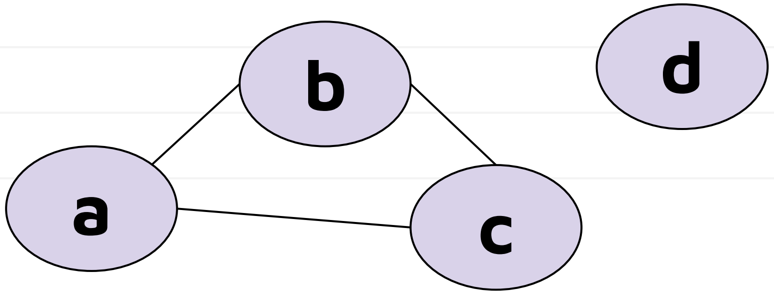


{a, b, c} - max. connected?

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?

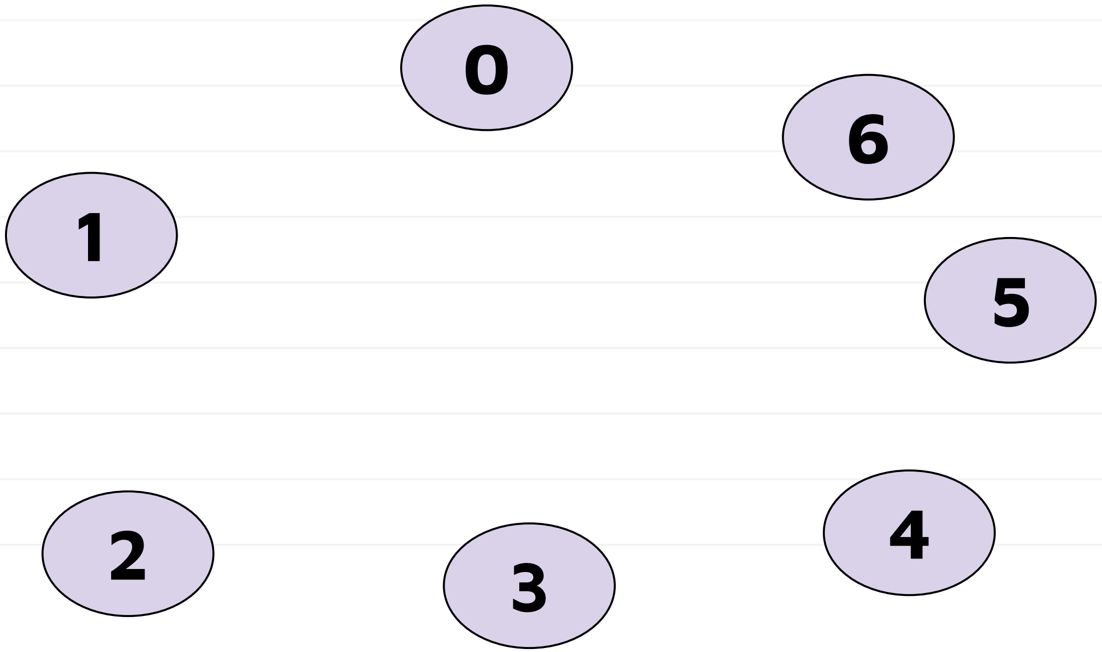
V = {0, 1, 2, 3, 4, 5, 6}

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}

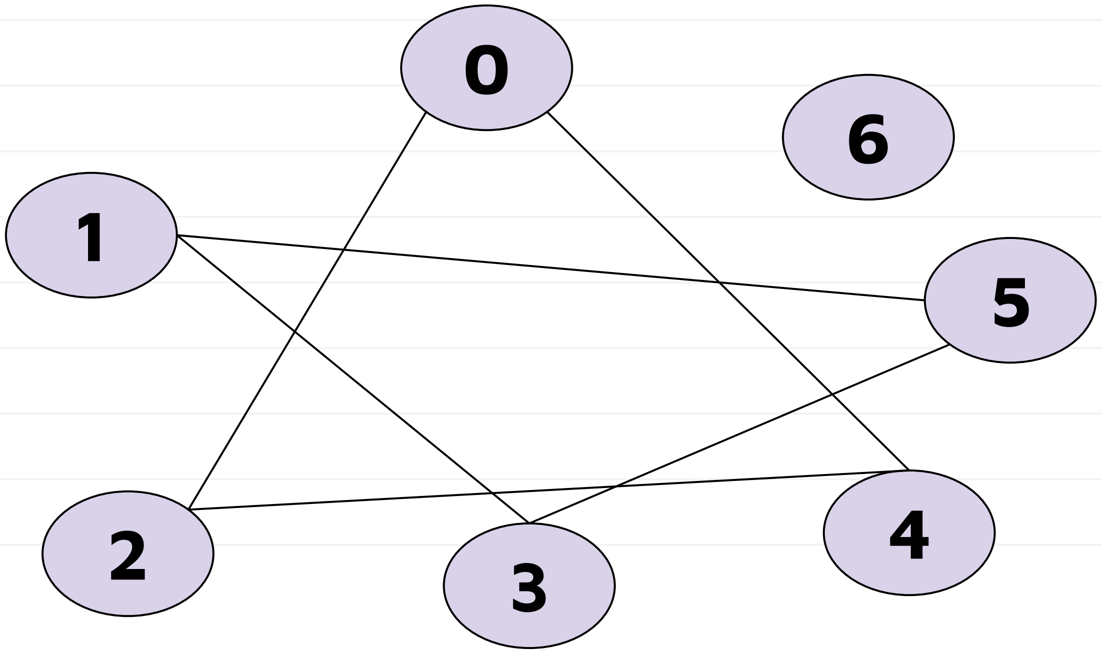
E = {(0, 2), (1, 5), (3, 1), (2, 4), (0, 4), (5, 3)}



What are the connected components?

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

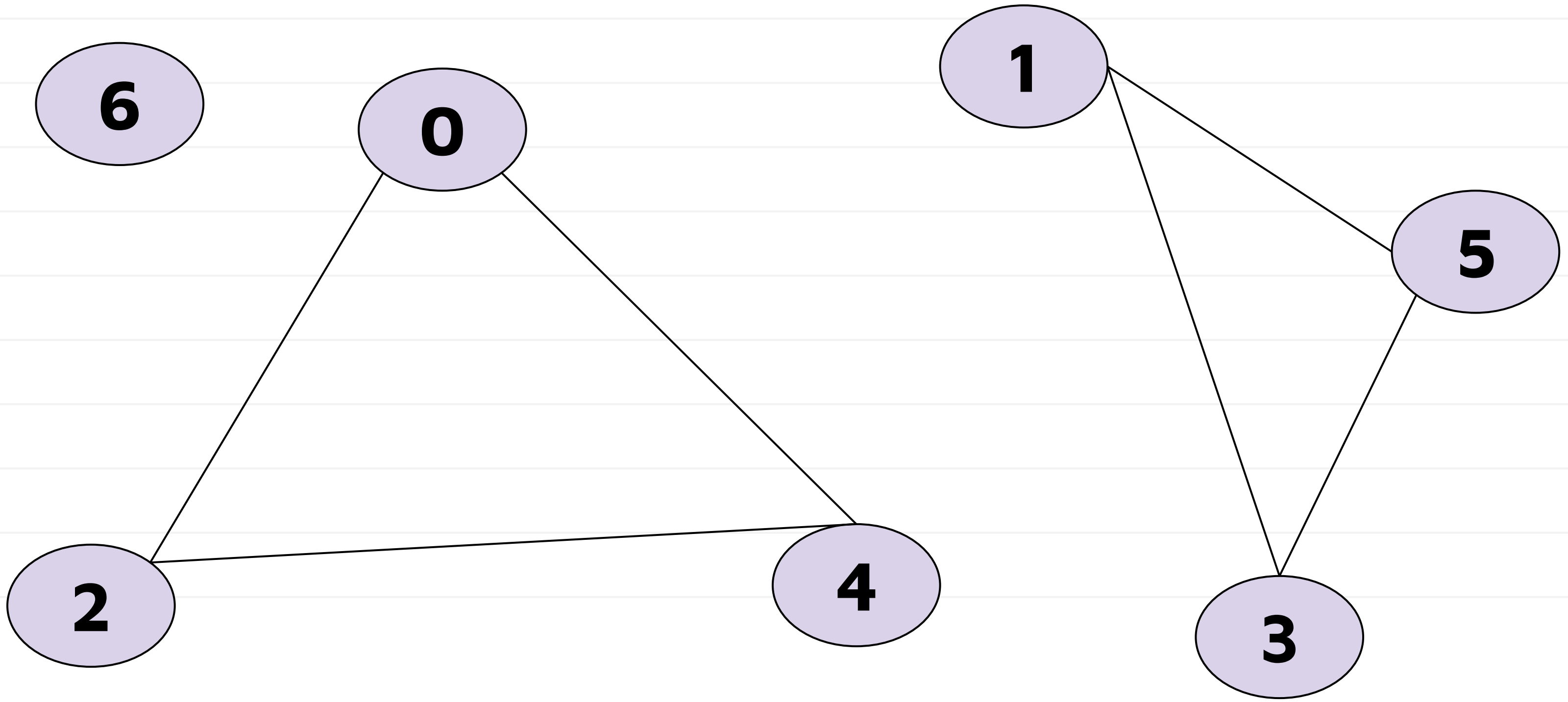
E = {(0, 2), (1, 5), (3, 1), (2, 4), (0, 4), (5, 3)}



What are the connected components?

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

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

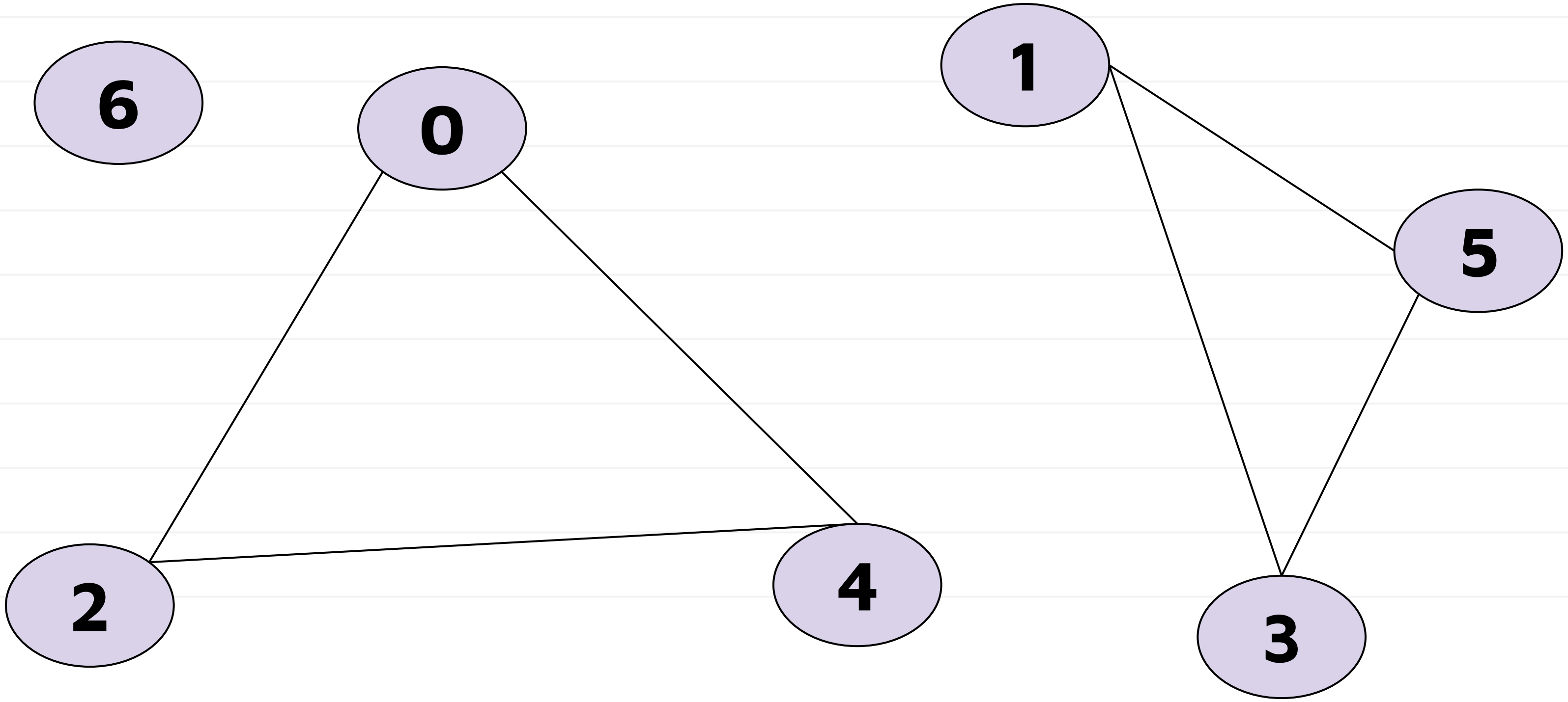


What are the connected components?

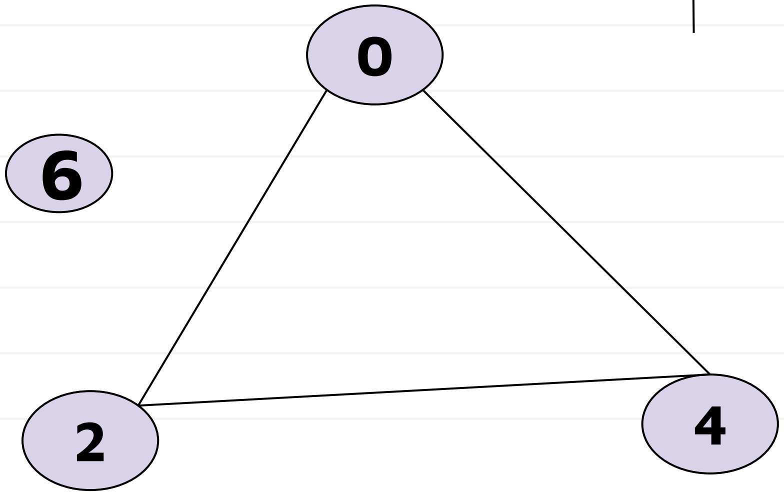
3

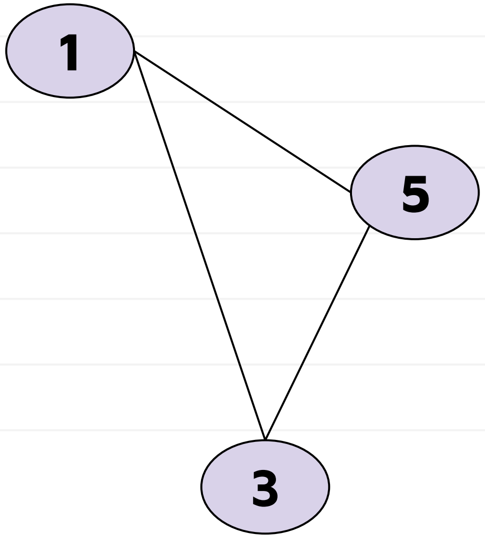
V = {0, 1, 2, 3, 4, 5, 6}

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



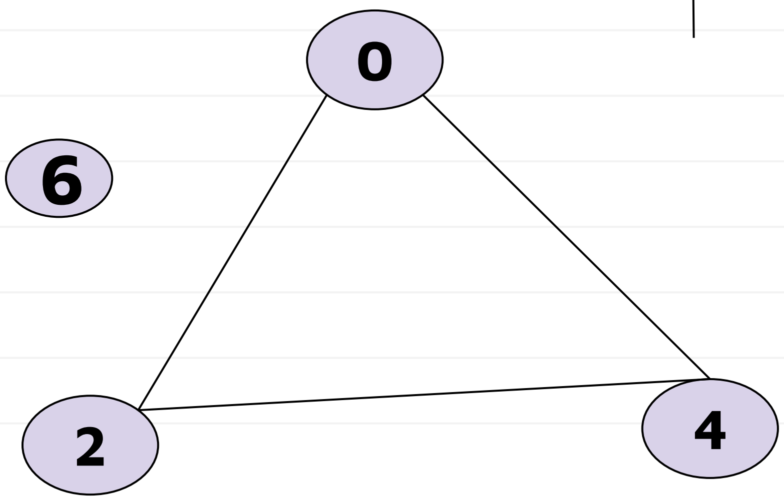
maximally-connected | not maximally-connected

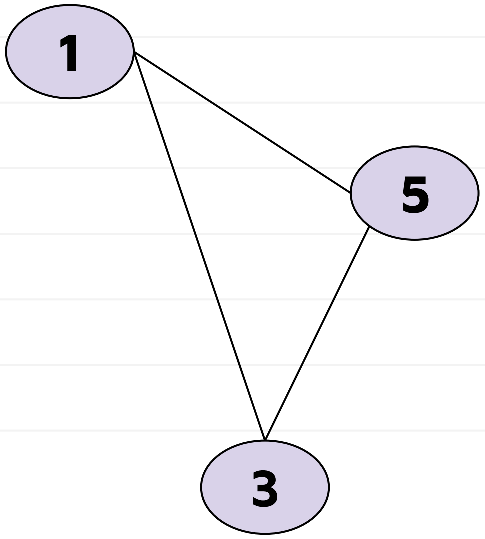




maximally-connected | not maximally-connected

{2, 6}

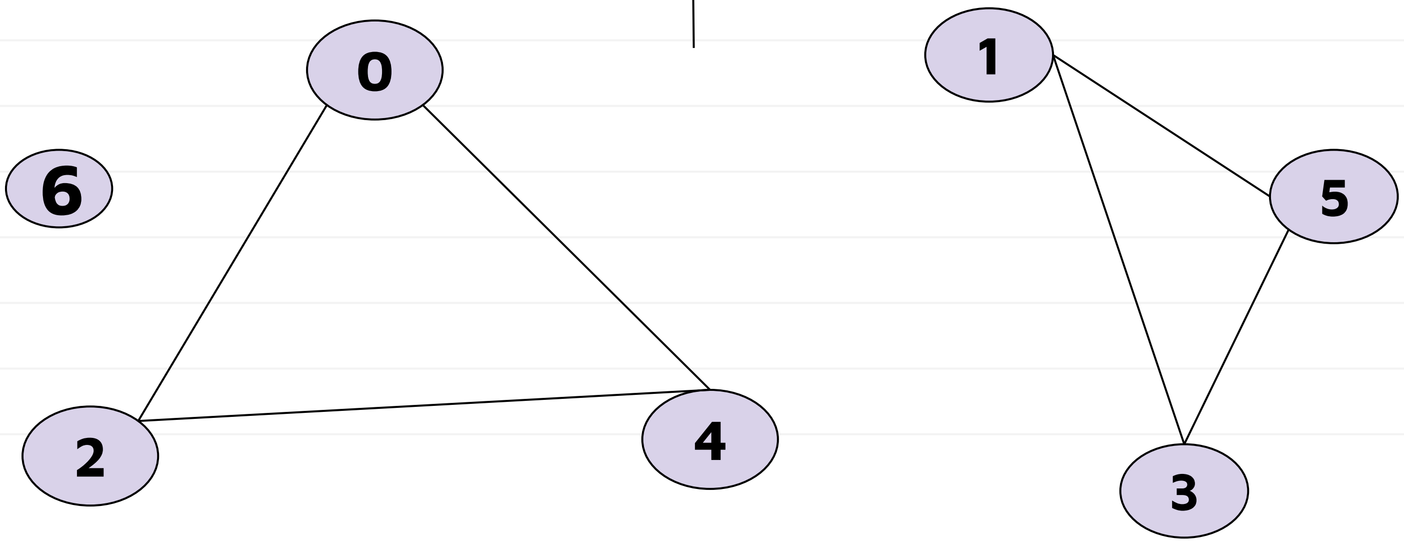




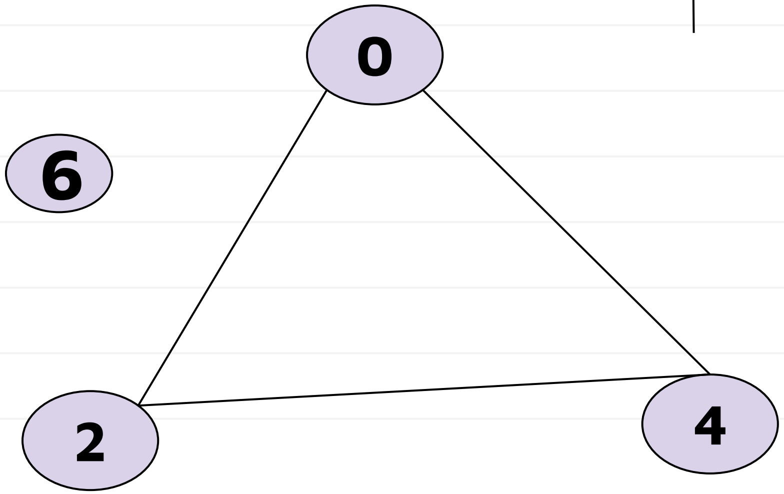
maximally-connected | **not maximally-connected**

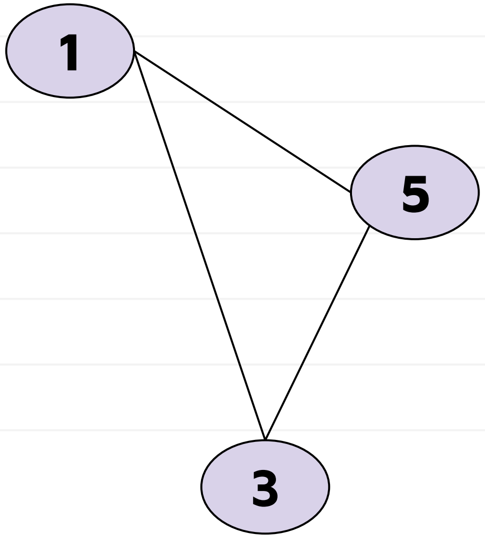
{2, 6}

2 and 6 are **not** connected



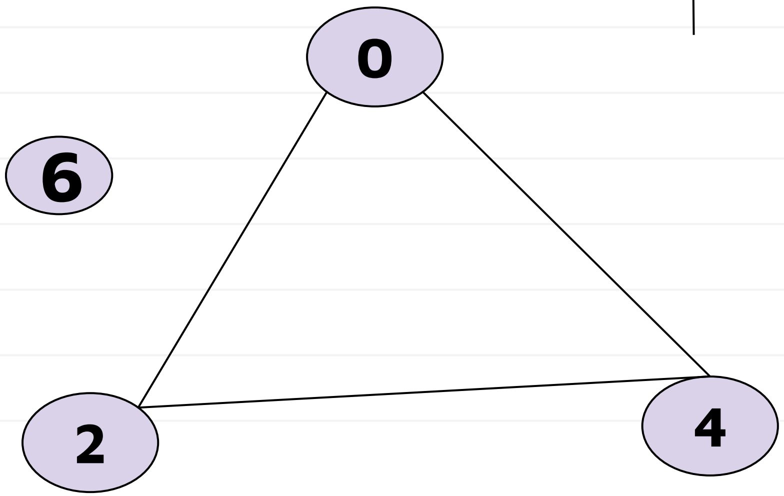
maximally-connected	not maximally-connected
	{2, 6}
	{2, 0, 4, 6}

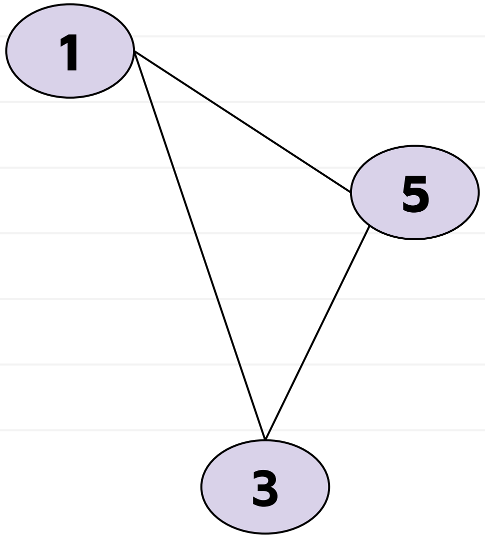




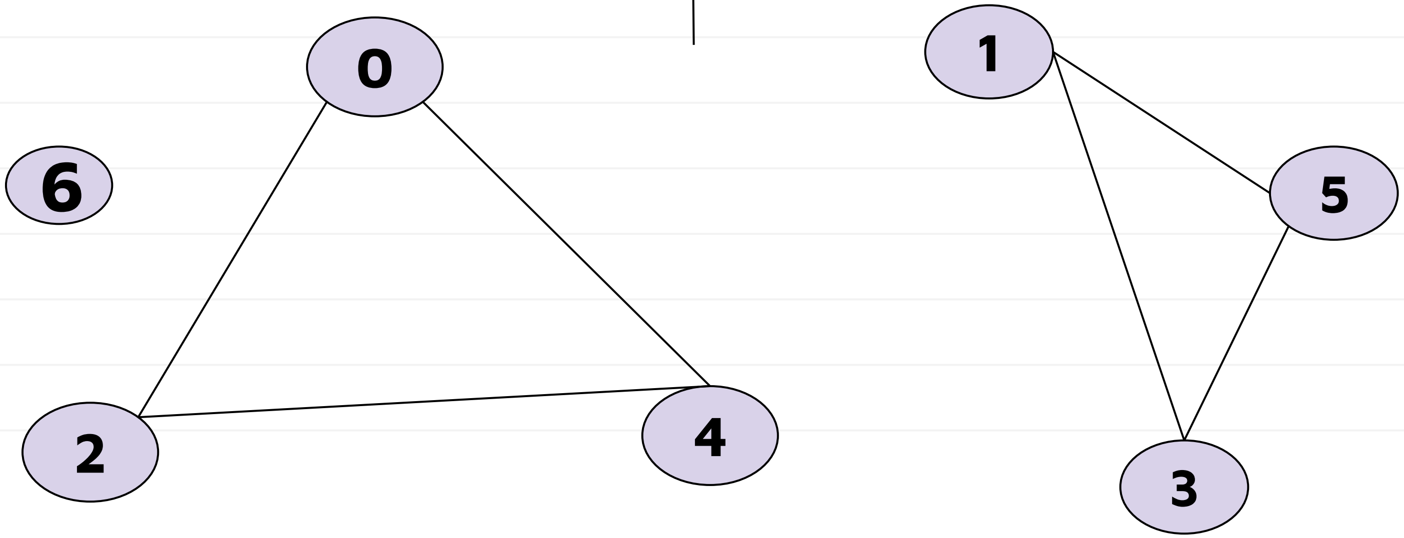
maximally-connected	not maximally-connected
	{2, 6}
	{2, 0, 4, 6}

2 and 6 are **not** connected

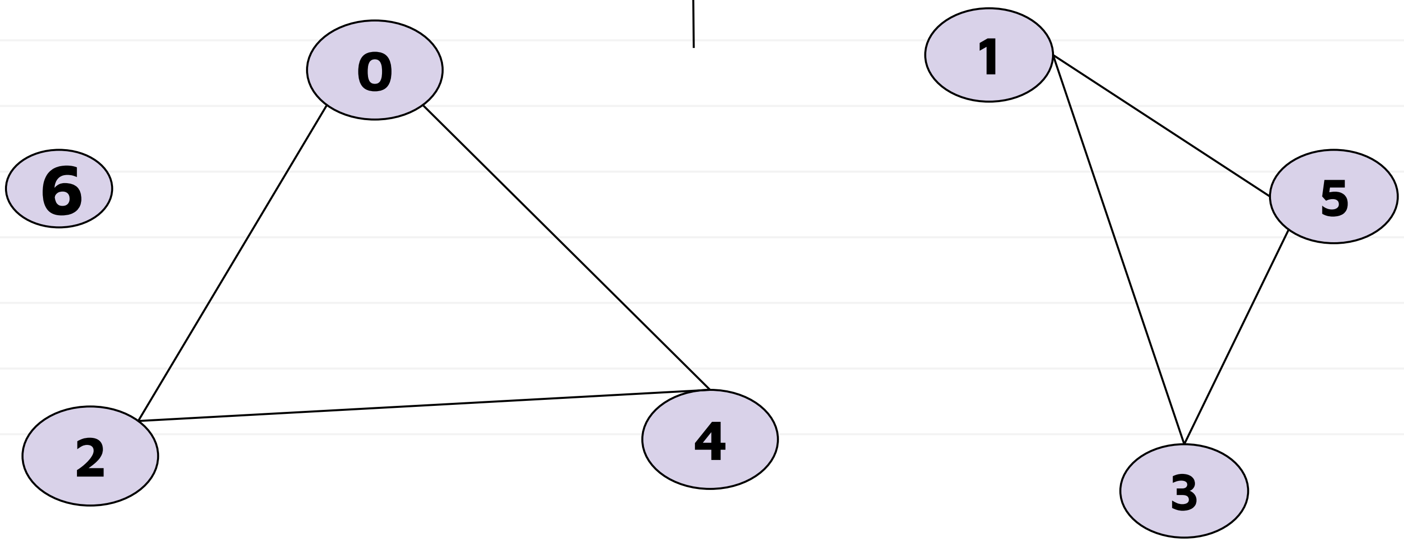




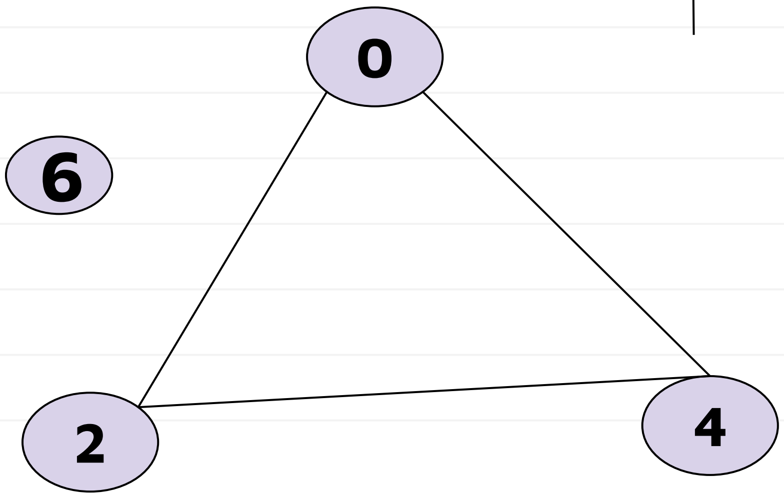
maximally-connected	not maximally-connected
	{2, 6}
	{2, 0, 4, 6}
	{1, 3, 5}

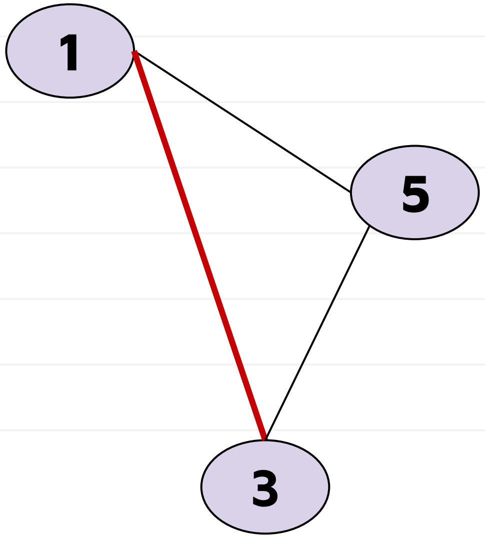


maximally-connected	not maximally-connected
{1, 3, 5}	{2, 6}
	{2, 0, 4, 6}

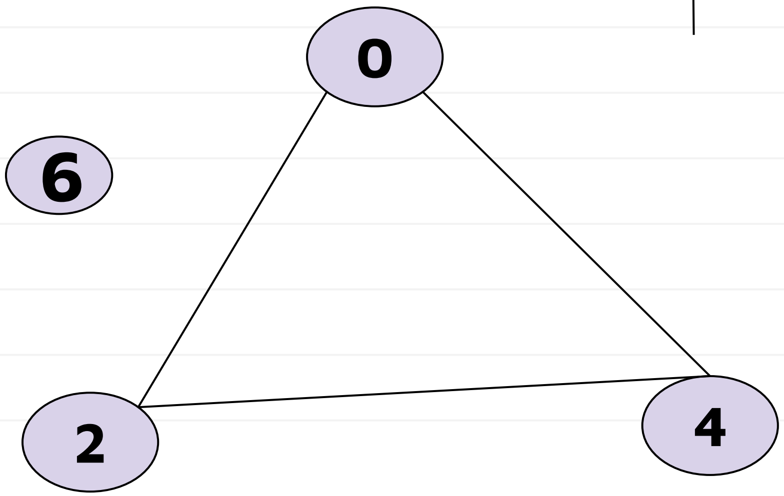


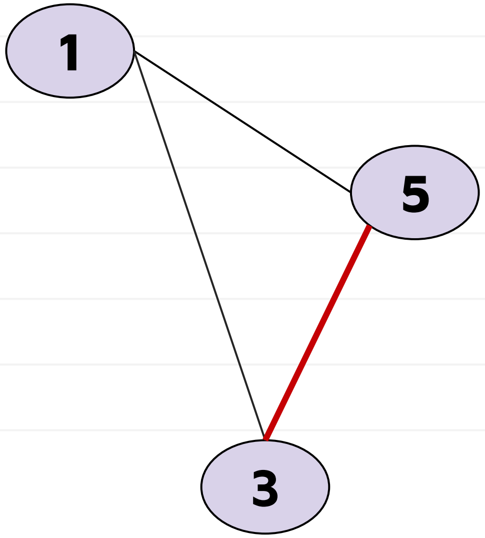
maximally-connected	not maximally-connected
{1, 3, 5}	{2, 6}
	{2, 0, 4, 6}



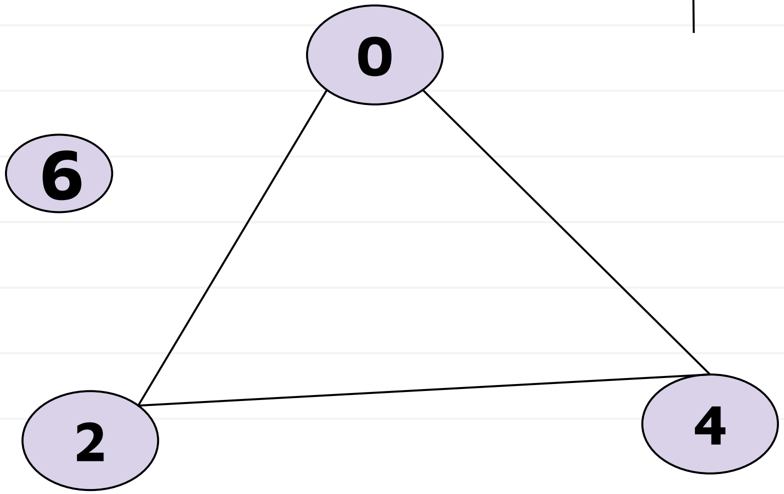


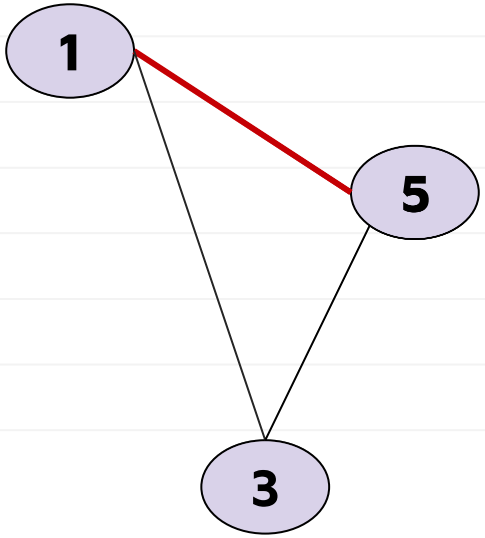
maximally-connected	not maximally-connected
{1, 3, 5}	{2, 6}
	{2, 0, 4, 6}





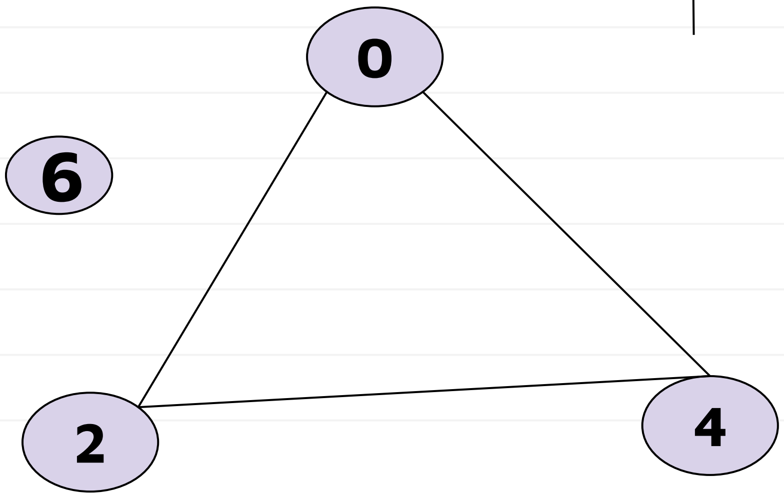
maximally-connected	not maximally-connected
{1, 3, 5}	{2, 6}
	{2, 0, 4, 6}

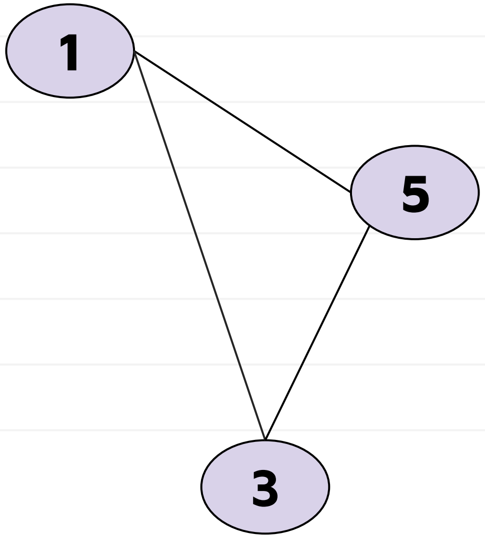




maximally-connected	not maximally-connected
{1, 3, 5}	{2, 6}
	{2, 0, 4, 6}

{2, 0}





maximally-connected

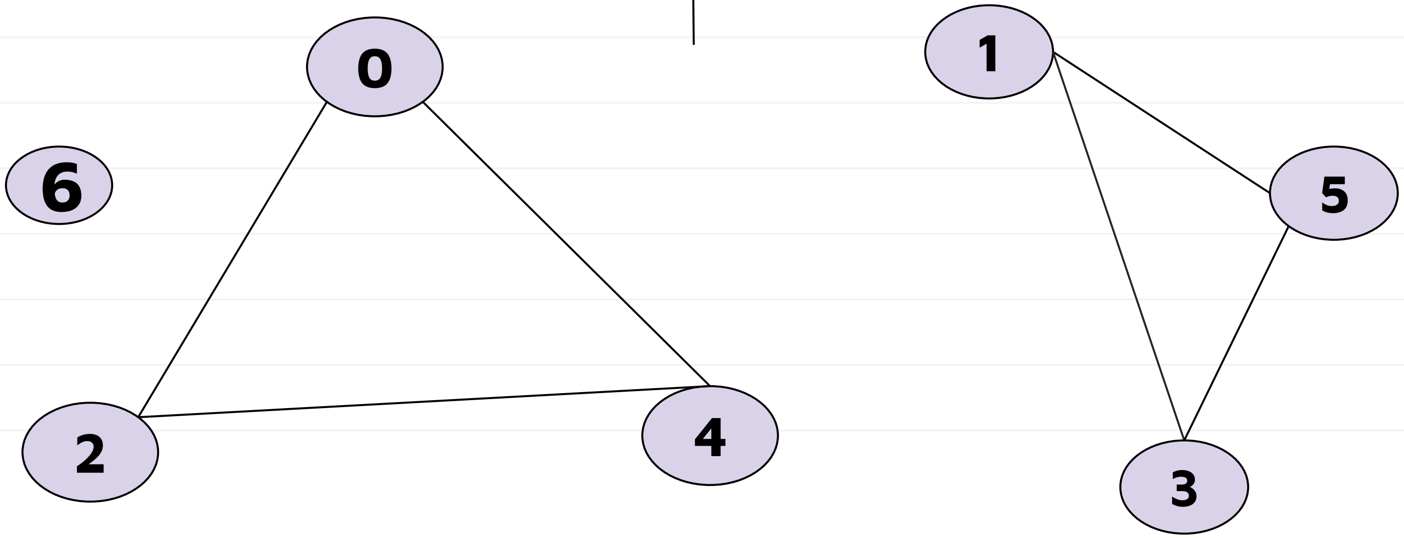
{1, 3, 5}

not maximally-connected

{2, 6}

{2, 0, 4, 6}

{2, 0}



maximally-connected

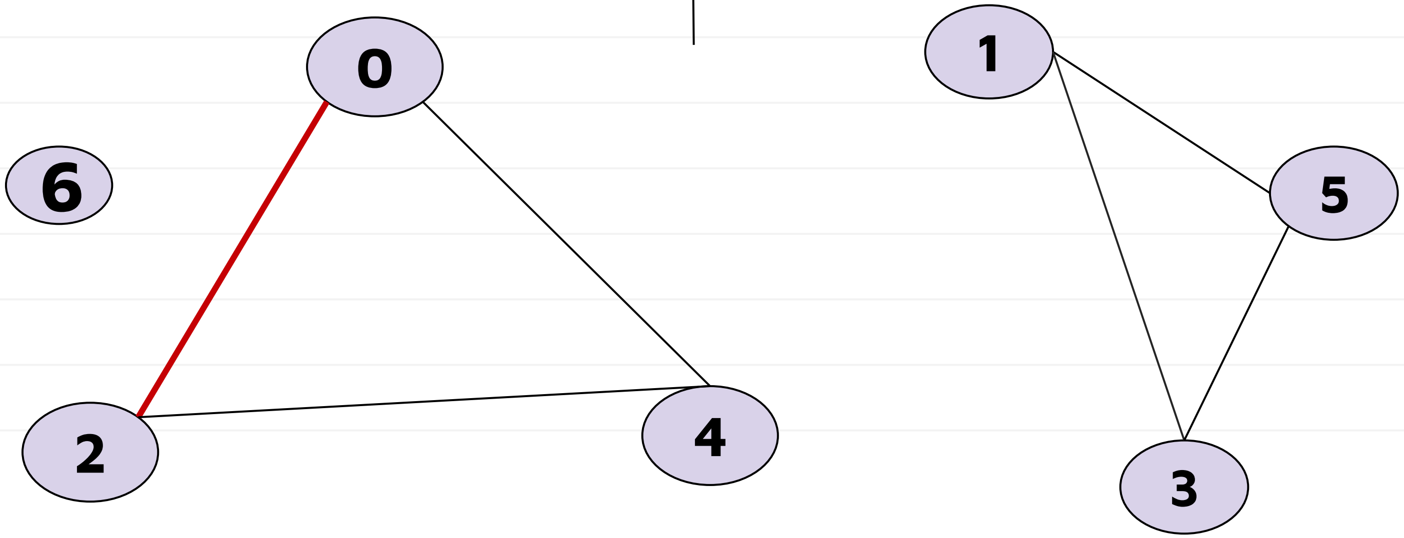
{1, 3, 5}

not maximally-connected

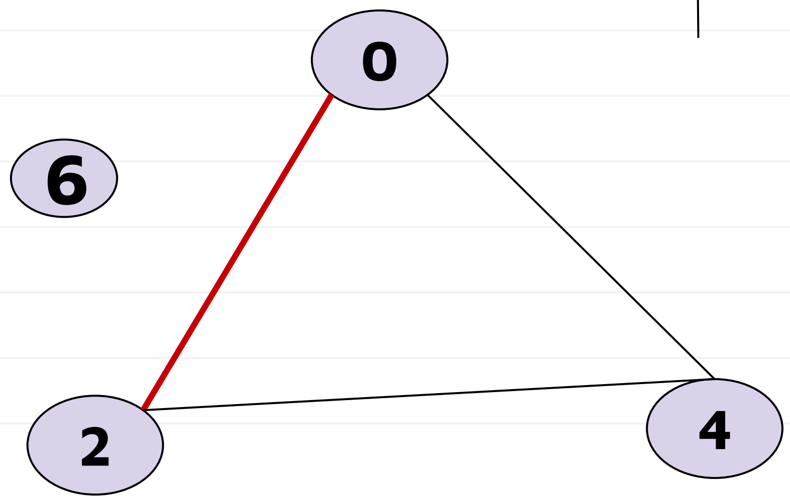
{2, 6}

{2, 0, 4, 6}

{2, 0}

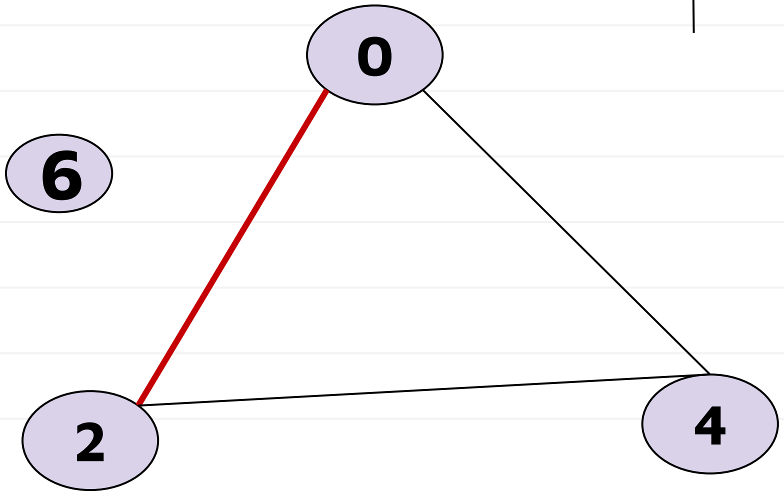


maximally-connected	not maximally-connected
{1, 3, 5}	{2, 6}
	{2, 0, 4, 6}
	{2, 0}



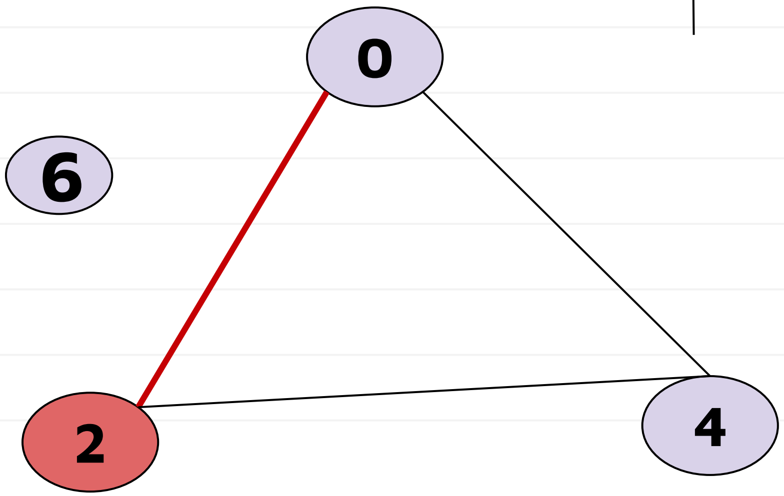
if $u \in C$ and $z \notin C$ then u and z are **not** reachable from one another.

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.

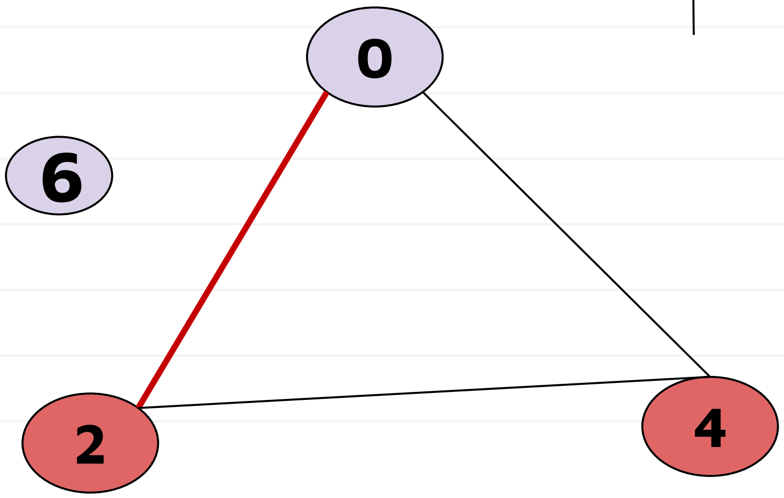
maximally-connected	not maximally-connected
{1, 3, 5}	{2, 6}
	{2, 0, 4, 6}
	{2, 0} = C



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

u is 2

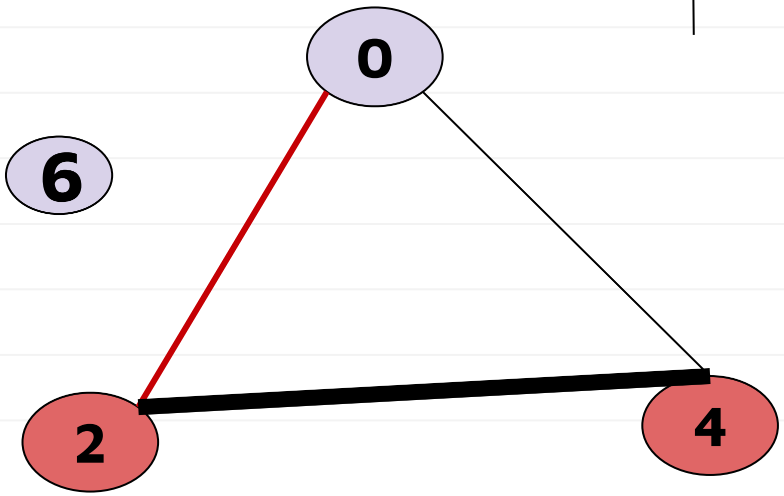
maximally-connected	not maximally-connected
{1, 3, 5}	{2, 6}
	{2, 0, 4, 6}
	{2, 0} = C



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

u is 2, z = 4

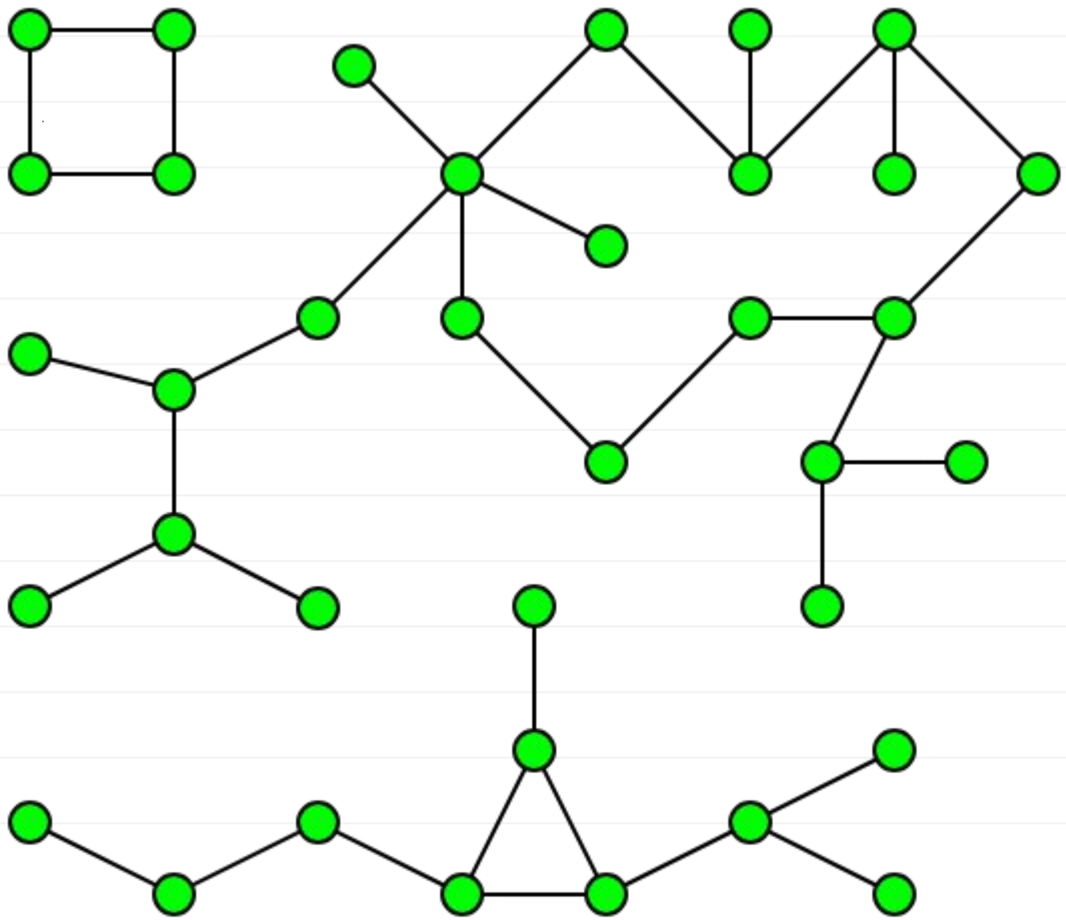
maximally-connected	**not maximally-connected**
{1, 3, 5}	{2, 6}
	{2, 0, 4, 6}
	{2, 0} = C



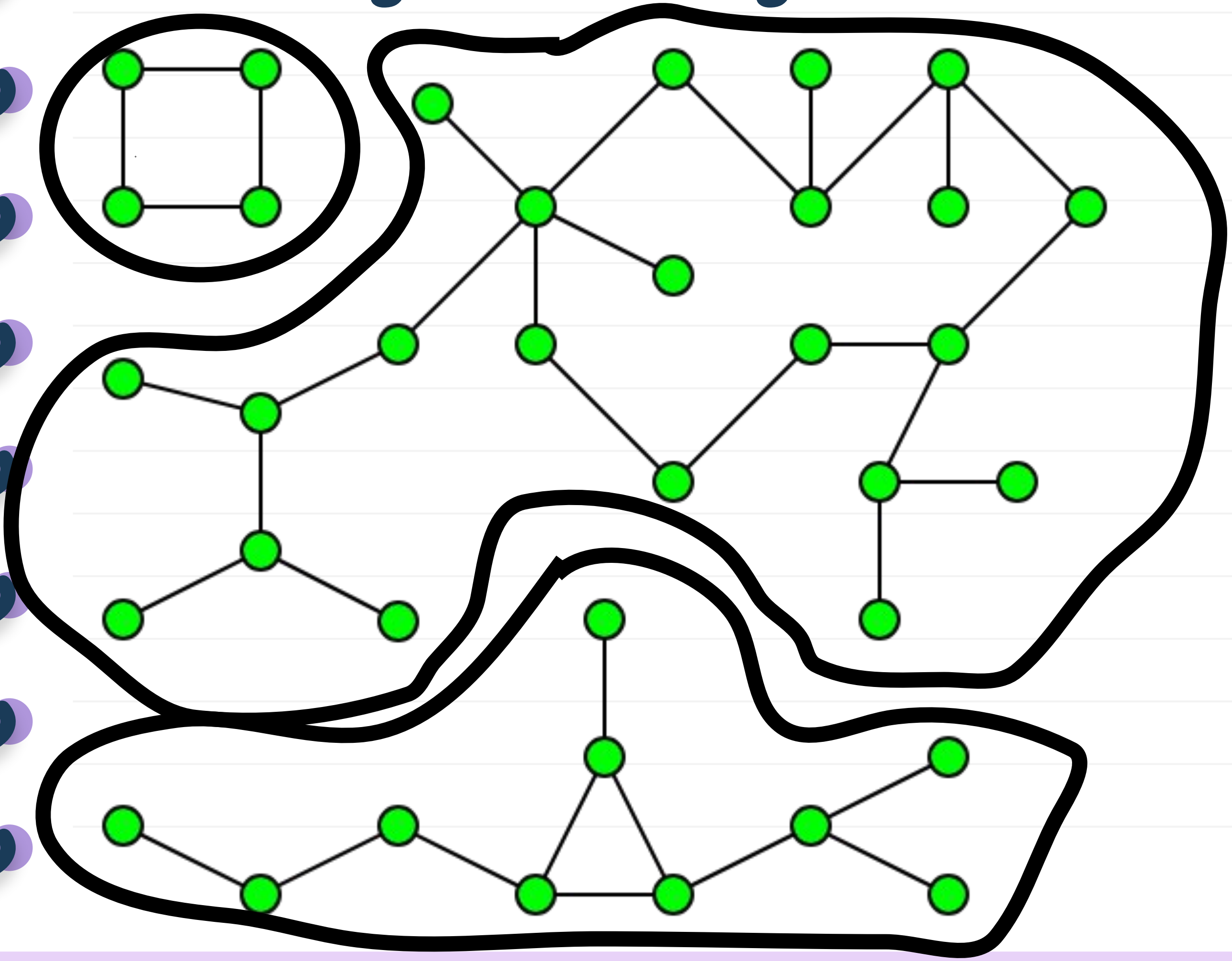
if $u \in C$ and $z \notin C$ then u and z **are not reachable from one another**.

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!