A&W

Exercise Session 2

Connectivity

Nil Ozer

Outline

- Minitest

- Connectivity
- Articulation Points and Bridges

- (Cycles)

Minitest

A&W Overview

A&W

Connectivity

- Articulation Points
- Bridges
- Block - Decomposition
- Menger's Theorem

Cycles

- Closed Eulerian Walk
- Hamiltonian Cycle
- TSP

Matchings

- Definition
- Algorithms
- Hall's Theorem

Colorings

- Definition
- Algorithm
- Brooks's Theorem

Wahrscheinlichkeit

- Grundbegriffe und Notationen
- Bedingte Wahrscheinlichkeiten
- Unabhängigkeiten
- Zufallsvariablen
- Wichtige Diskrete Verteilungen
- Abschätzen von Wahrscheinlichkeiten

Randomized Algorithms

- Las-Vegas
- Monte-Carlo
- Primality Test
- Target - Shooting
- Finding Duplicates
- Longest Path Problem

Graph Algorithms

Flow

- Definition
- Maxflow - Mincut
- Ford - Fulkerson
- Matching w. Flow
- Edge-disjoint paths w. Flow

Minimum Cut

- Definition
- Cut(G) Algorithm
- Bootstrapping

Geometric Algorithms

Convex Hull

- Definition
- Jarvis Wrap
- Local optimization

Smallest Enclosing Circle

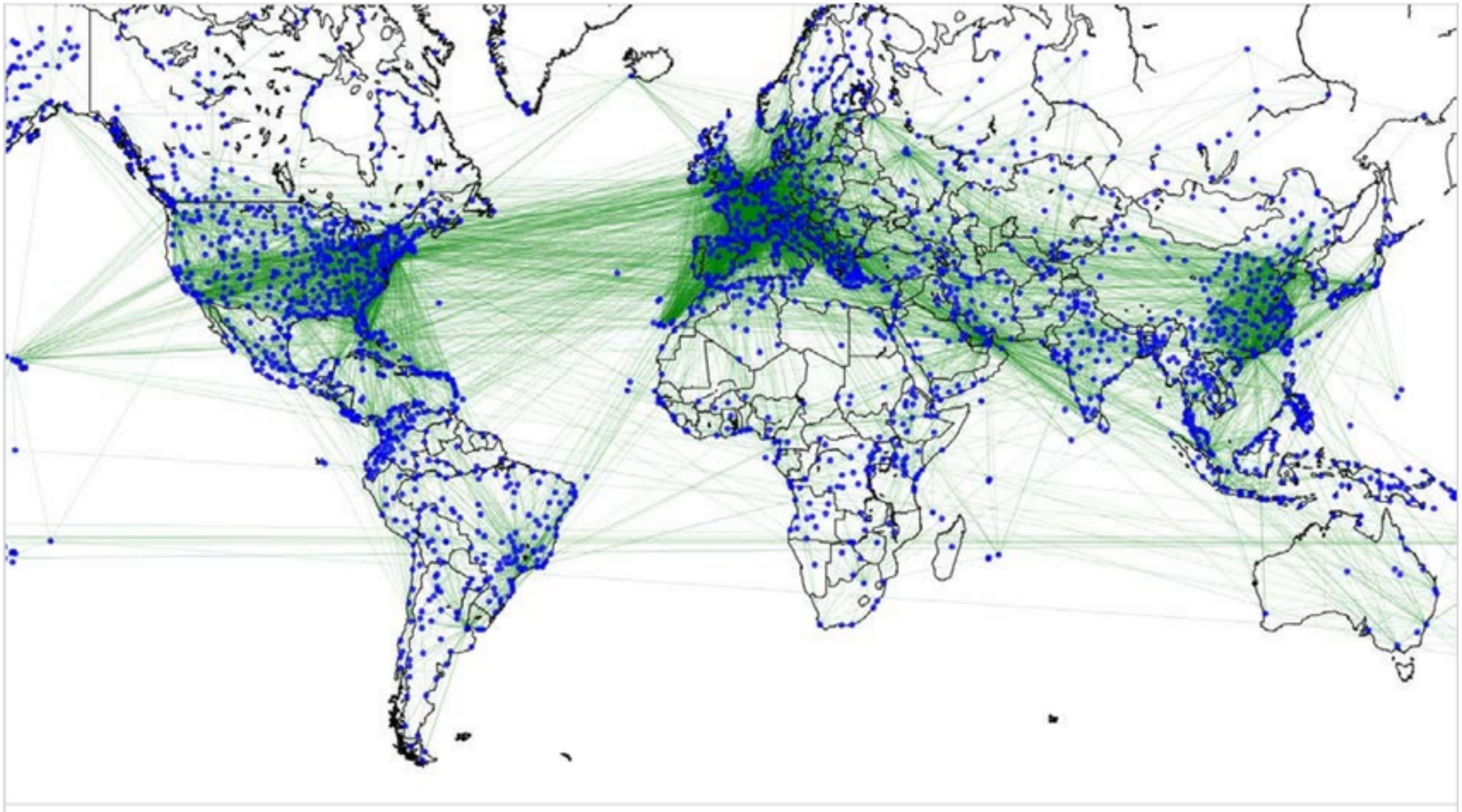
- Definition
- First Algorithm
- Final Algorithm

Connectivity

Connectivity
Intuition

measuring fault tolerance of a network !

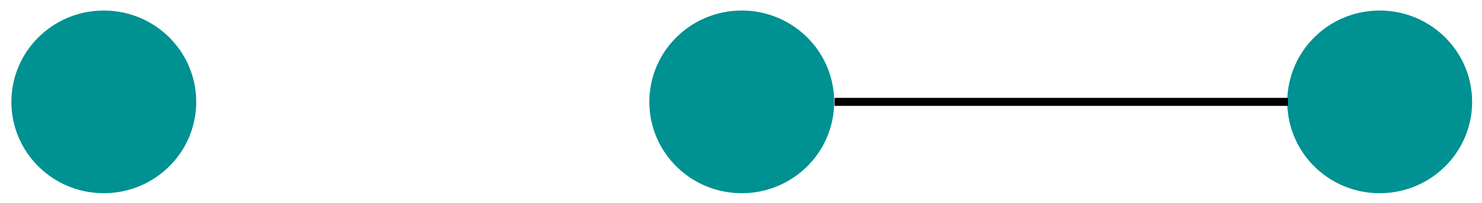
How many connections can fail without cutting off the communication ?



Connectivity
Definitions

- A $G = (V, E)$ is connected if

  for $\forall u, v \in V, u \neq v$ there exists an u-v path

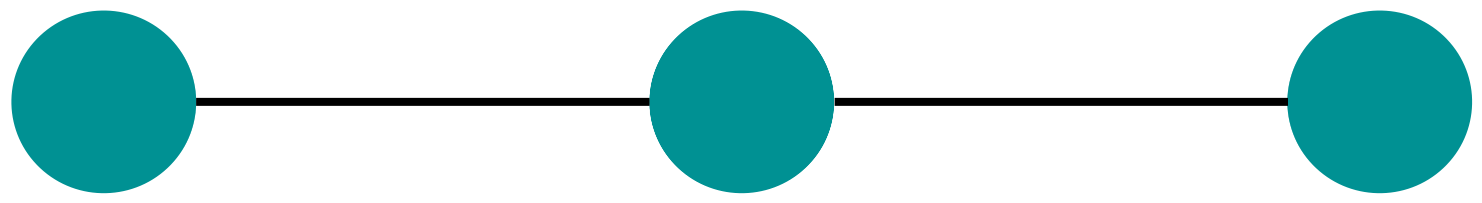


not connected

Connectivity
Definitions

- A $G = (V, E)$ is connected if

  for $\forall u, v \in V, u \neq v$ there exists an u-v path



connected

Connectivity

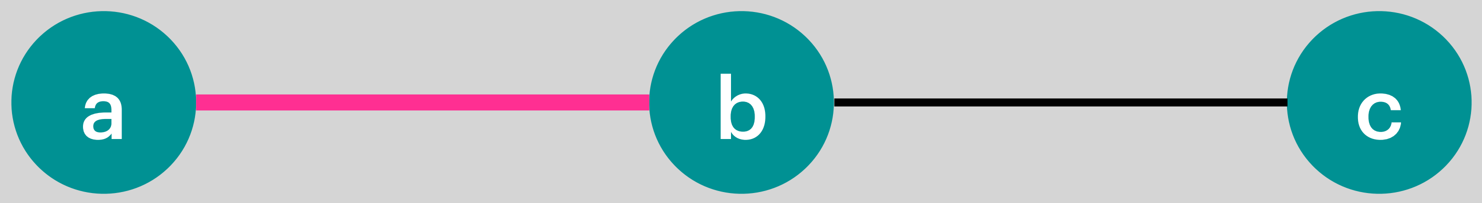
“Removing”

Remove the edge {a,b}

Connectivity

“Removing”

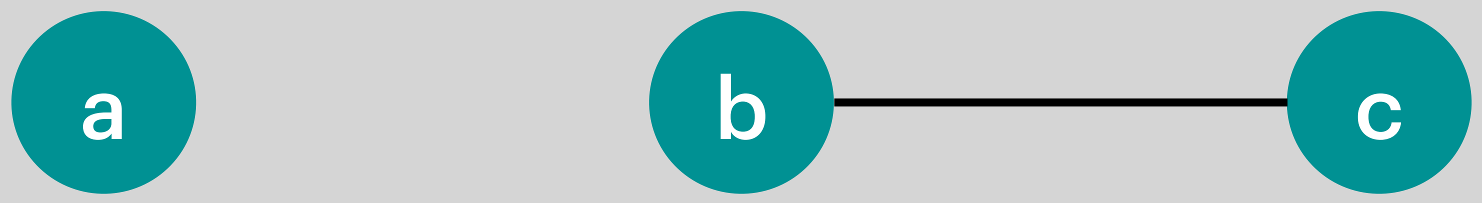
Remove the edge {a,b}



Connectivity

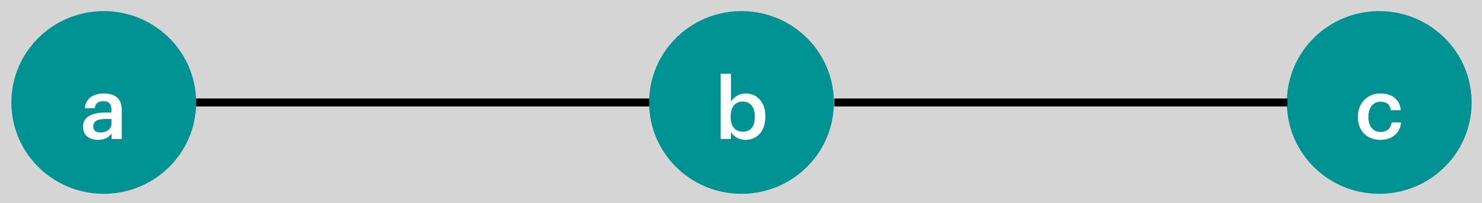
“Removing”

Remove the edge {a,b}



Connectivity
“Removing”

Remove the vertex b



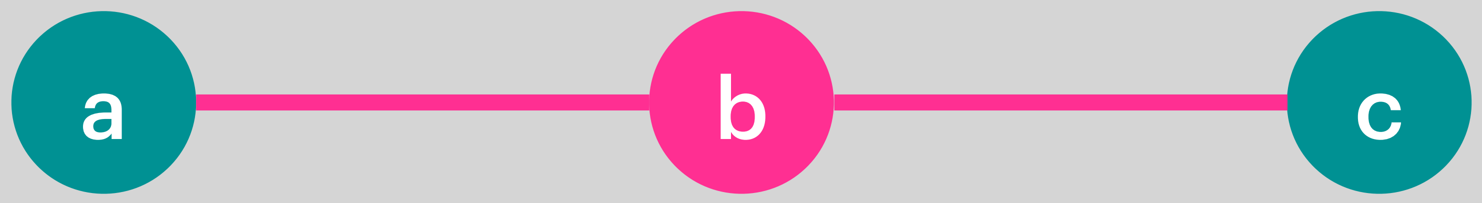
Connectivity

“Removing”

Remove the vertex b

Connectivity
“Removing”

Remove the vertex b



Connectivity
“Removing”

Remove the vertex b

Connectivity

Definitions

- A $G = (V, E)$ is k-edge connected if

$$\forall X \subseteq E \text{ with } |X| < k \ , G(V, E \setminus X) \text{ is connected}$$

" The G remains connected whenever fever than k edges are removed"

" At least k edges must be removed to make the G disconnected"

How to find the k-edge connectivity?

Start from 1-edge connected, increase one by one !

(Every connected G is 1-edge connected)

Connectivity

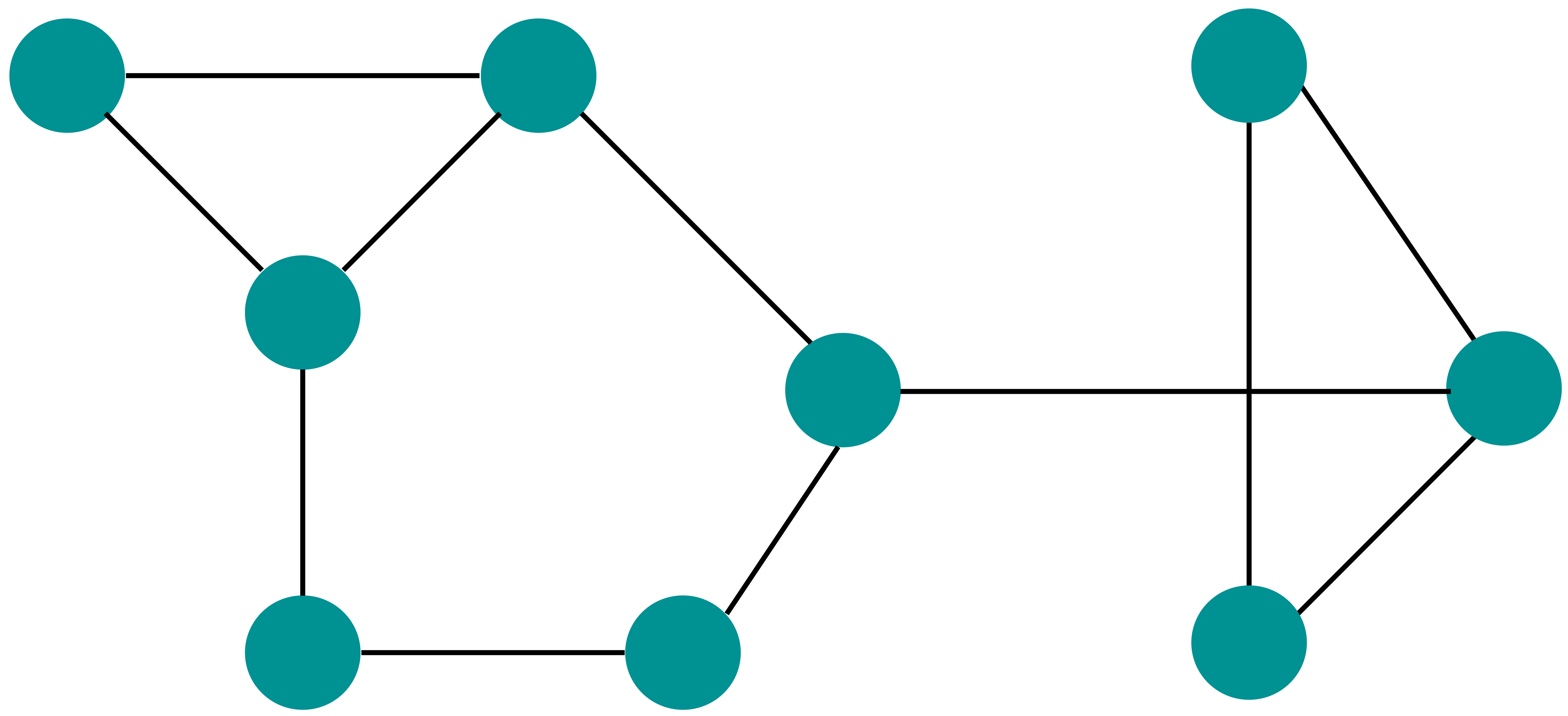
Example

- A $G = (V, E)$ is k-edge connected if

  $\forall X \subseteq E$ with $|X| < k$, $G(V, E \setminus X)$ is connected

" The G remains connected whenever fever than k edges are removed"

" At least k edges must be removed to make the G disconnected"



Connectivity

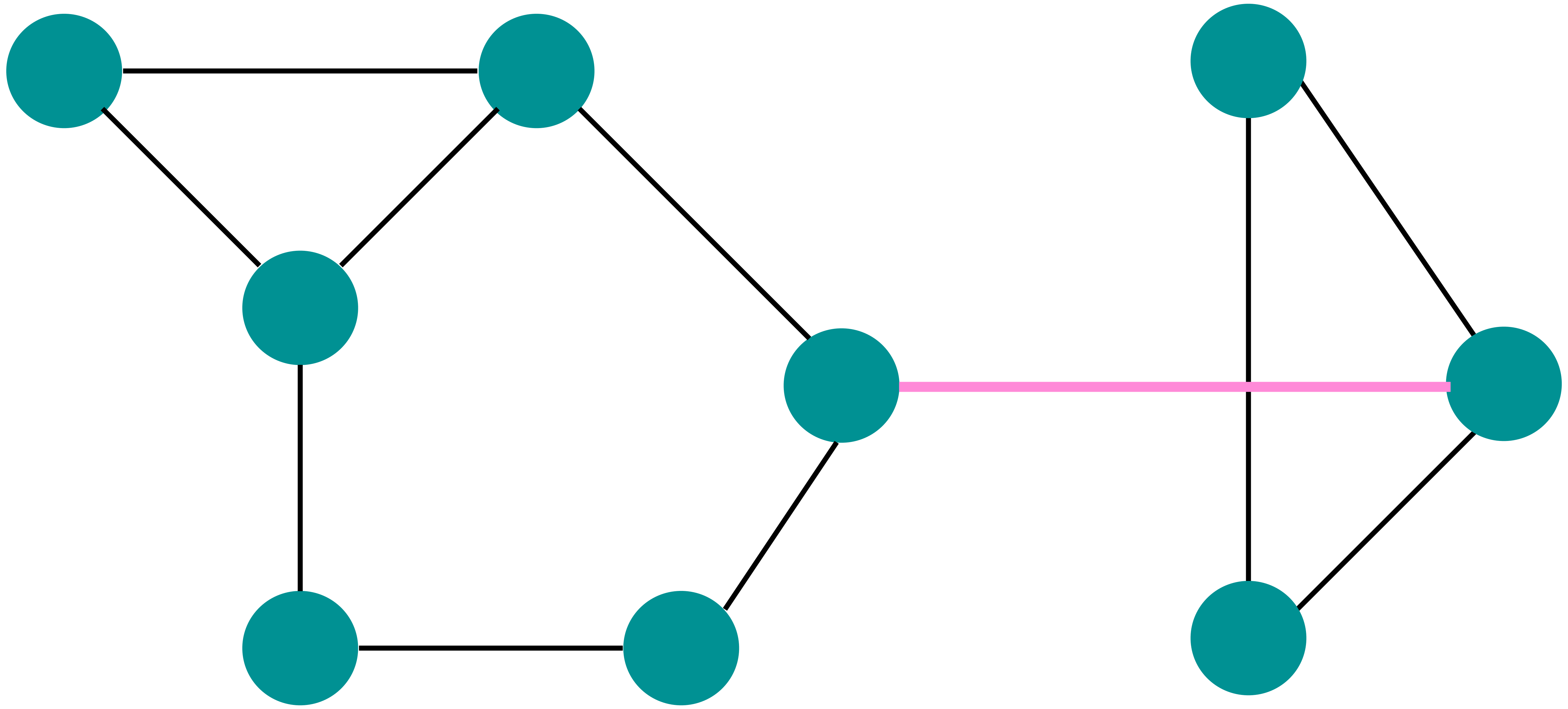
Example

- A $G = (V, E)$ is k-edge connected if

  $\forall X \subseteq E$ with $|X| < k$, $G(V, E \setminus X)$ is connected

" The G remains connected whenever fever than k edges are removed"

" At least k edges must be removed to make the G disconnected"



1-edge connected

Connectivity

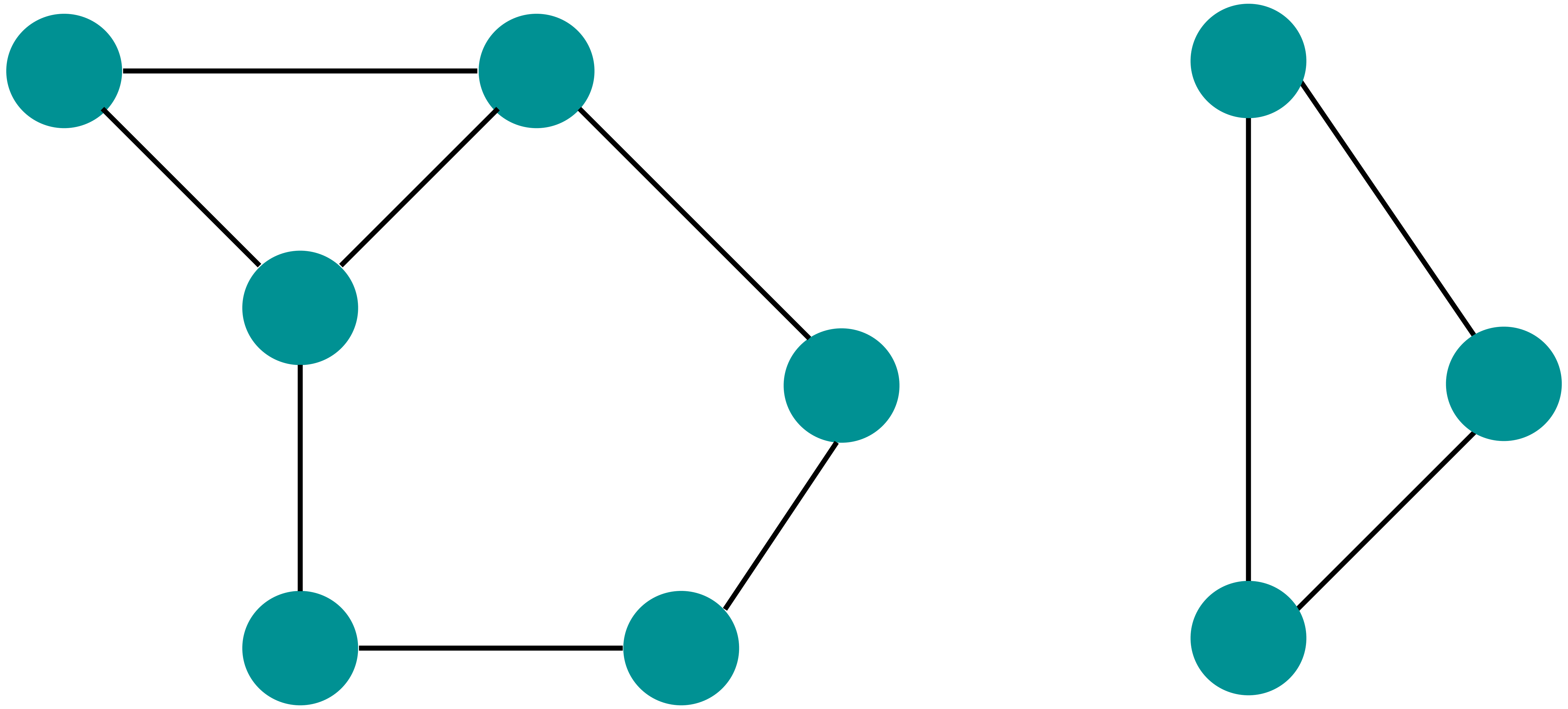
Example

- A $G = (V, E)$ is k-edge connected if

  $\forall X \subseteq E$ with $|X| < k$, $G(V, E \setminus X)$ is connected

" The G remains connected whenever fever than k edges are removed"

" At least k edges must be removed to make the G disconnected"



1-edge connected

becomes disconnected after removing 1 edge

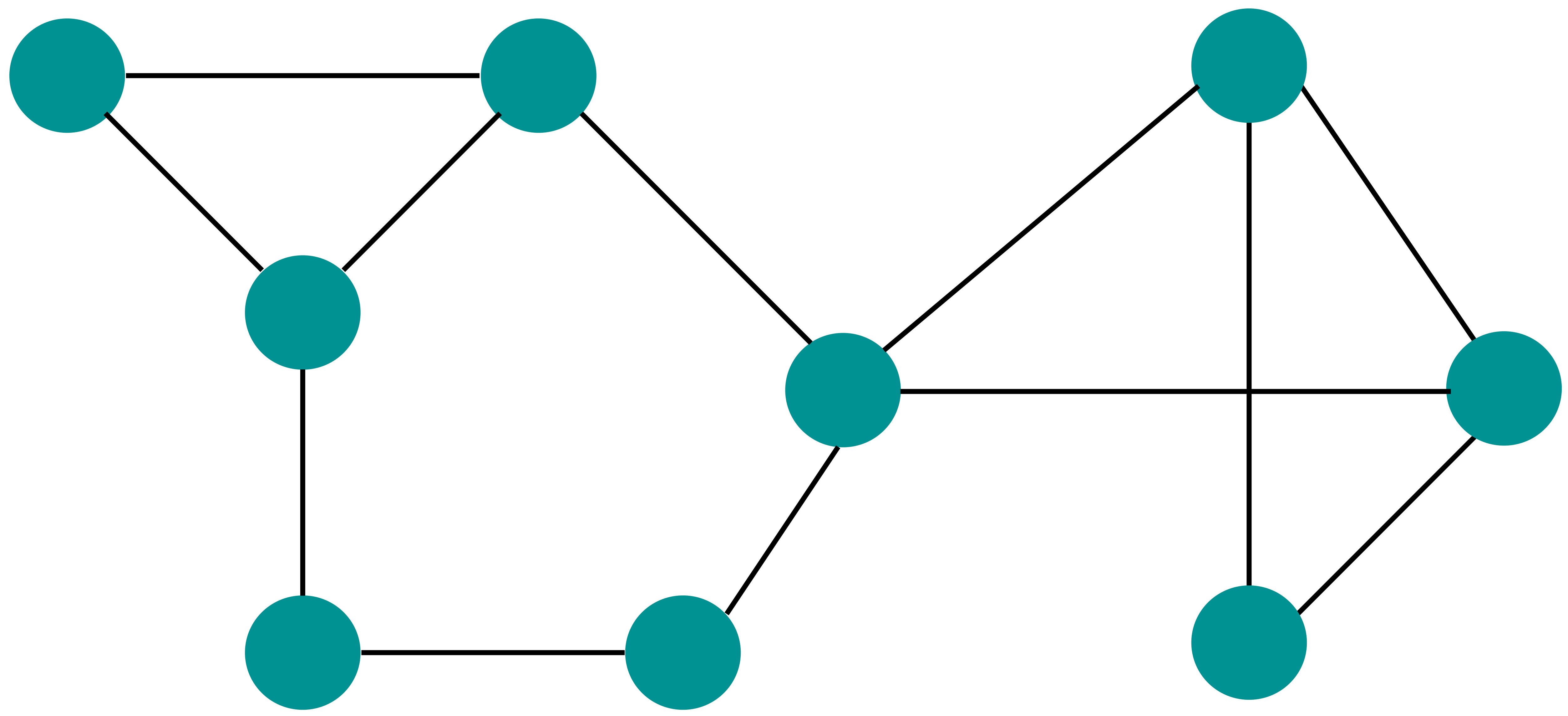
Connectivity
Example

- A $G = (V, E)$ is k-edge connected if

  $\forall X \subseteq E$ with $|X| < k$, $G(V, E \setminus X)$ is connected

" The G remains connected whenever fever than k edges are removed"

" At least k edges must be removed to make the G disconnected"



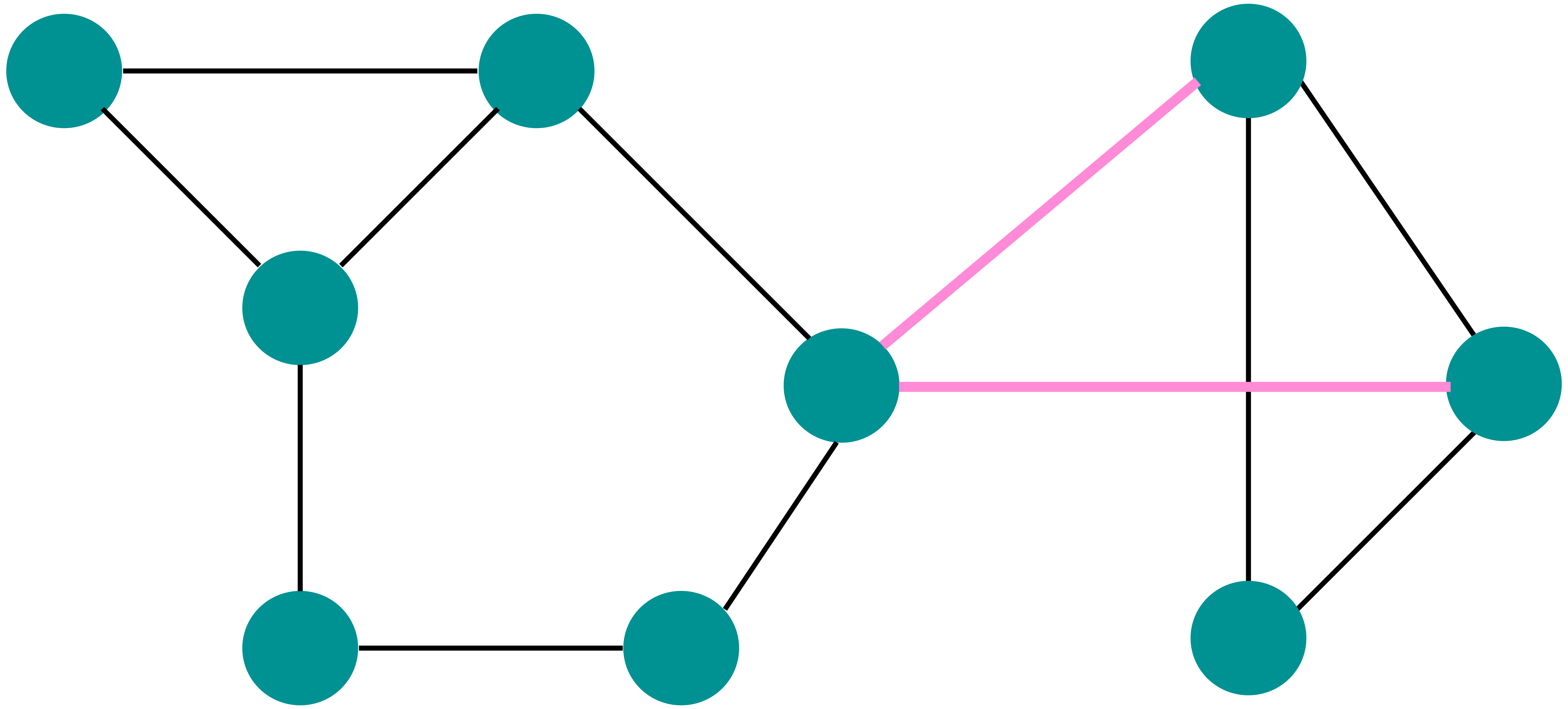
Connectivity
Example

- A $G = (V, E)$ is k-edge connected if

  $\forall X \subseteq E$ with $|X| < k$, $G(V, E \setminus X)$ is connected

" The G remains connected whenever fever than k edges are removed"

" At least k edges must be removed to make the G disconnected"



2-edge connected

Connectivity

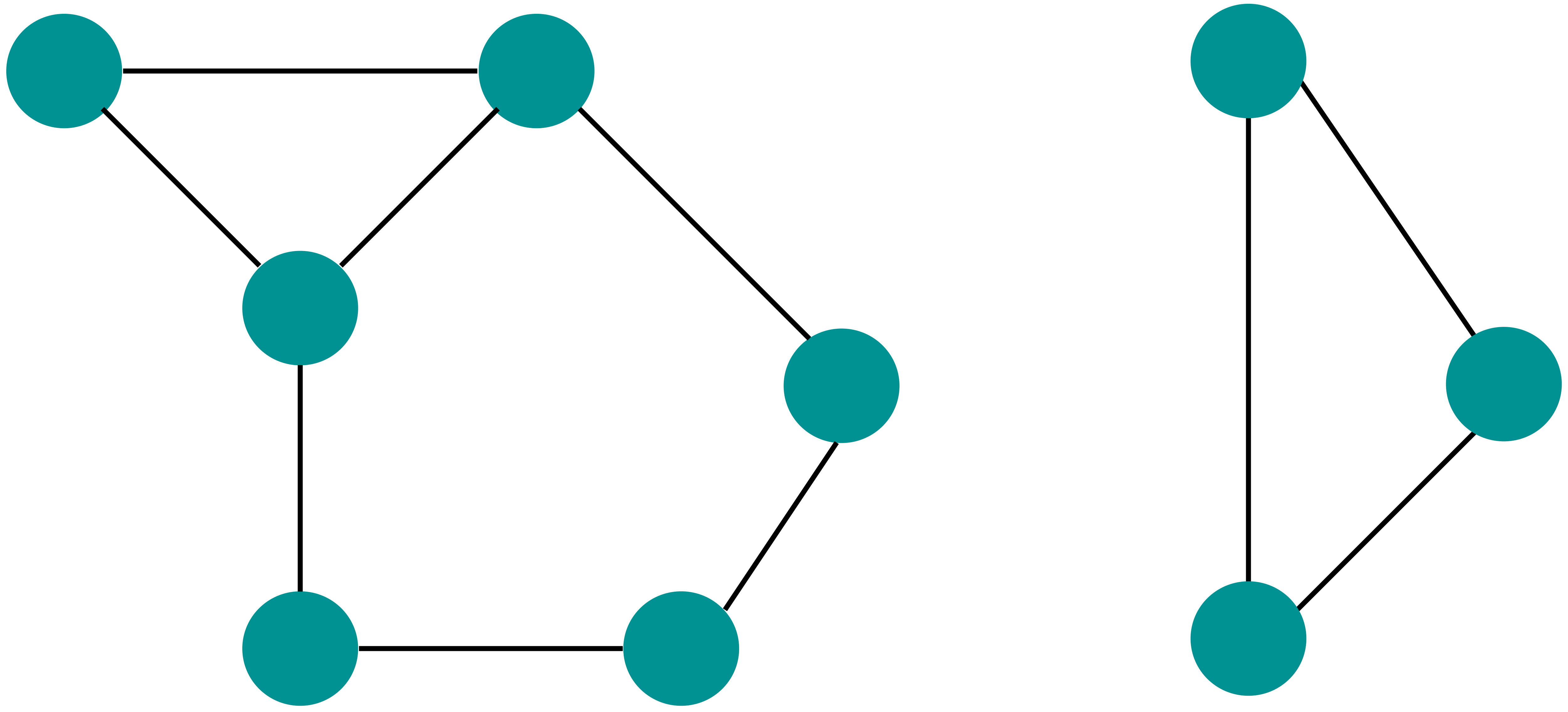
Example

- A $G = (V, E)$ is k-edge connected if

  $\forall X \subseteq E$ with $|X| < k$, $G(V, E \setminus X)$ is connected

" The G remains connected whenever fever than k edges are removed"

" At least k edges must be removed to make the G disconnected"



2-edge connected

becomes disconnected after removing 2 edges

Connectivity
Definitions

- A $G = (V, E)$ is k - (vertex) connected if
  - $|V| \geq k + 1$
  - $\forall X \subseteq V$ with $|X| < k$, $G[V \backslash X]$ is connected

" The G remains connected whenever fever than k vertices are removed"

" At least k vertices must be removed to make the G disconnected"

minitest 8,9

Connectivity

Lemma

k-vertex connectivity ≤ k-edge connectivity ≤ minimum degree

Connectivity

Menger's Theorem

G is k-edge connected $\iff$ For $\forall u, v \in V, u \neq v$ there exists k edge-disjoint u-v paths

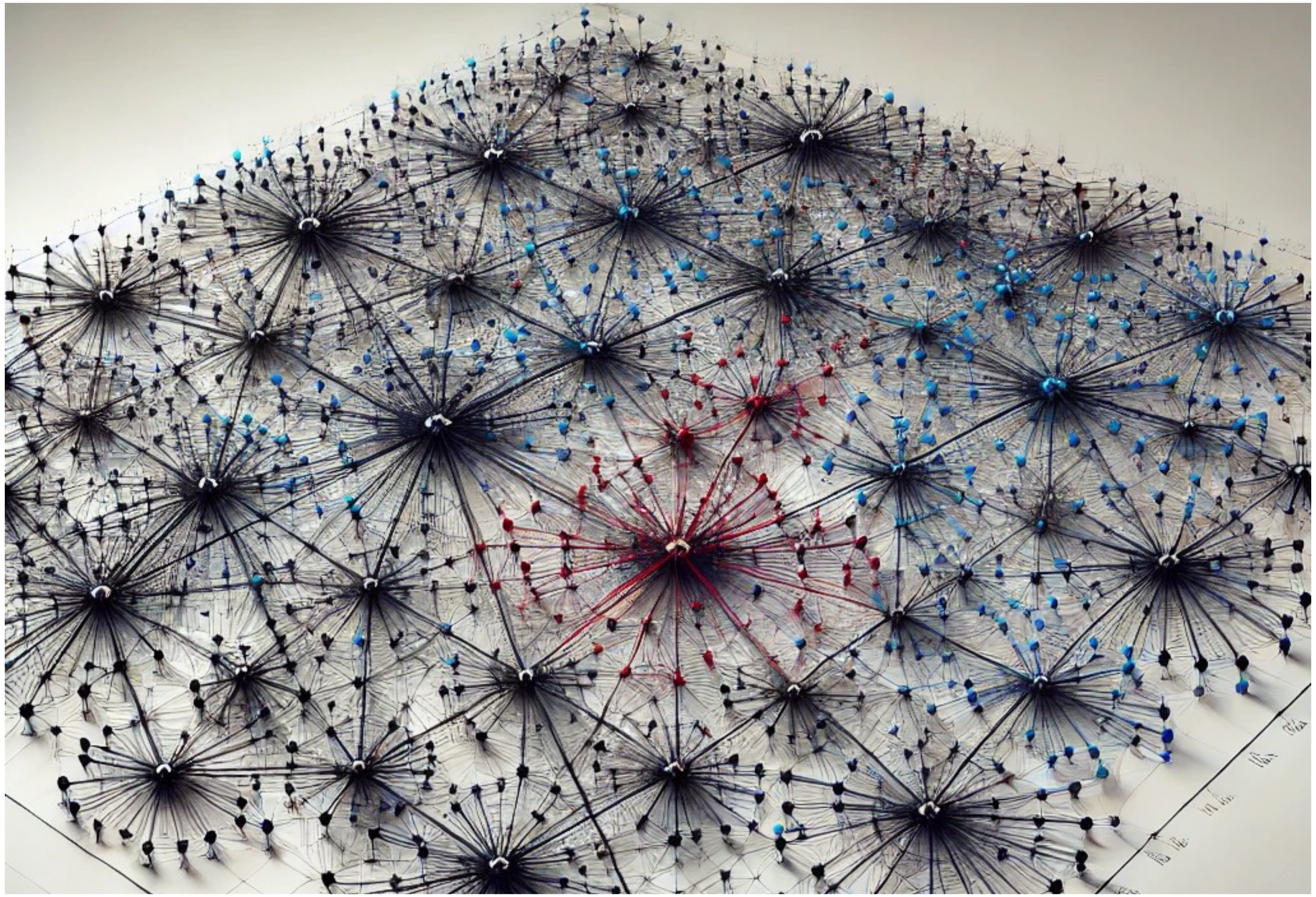
G is k-vertex connected $\iff$ For $\forall u, v \in V, u \neq v$ there exists k internally-vertex-disjoint u-v paths

they share the starting and ending vertex

Let's take a break

Articulation Points and Bridges
Intuition

single points whose failure would split the network into 2 or more components

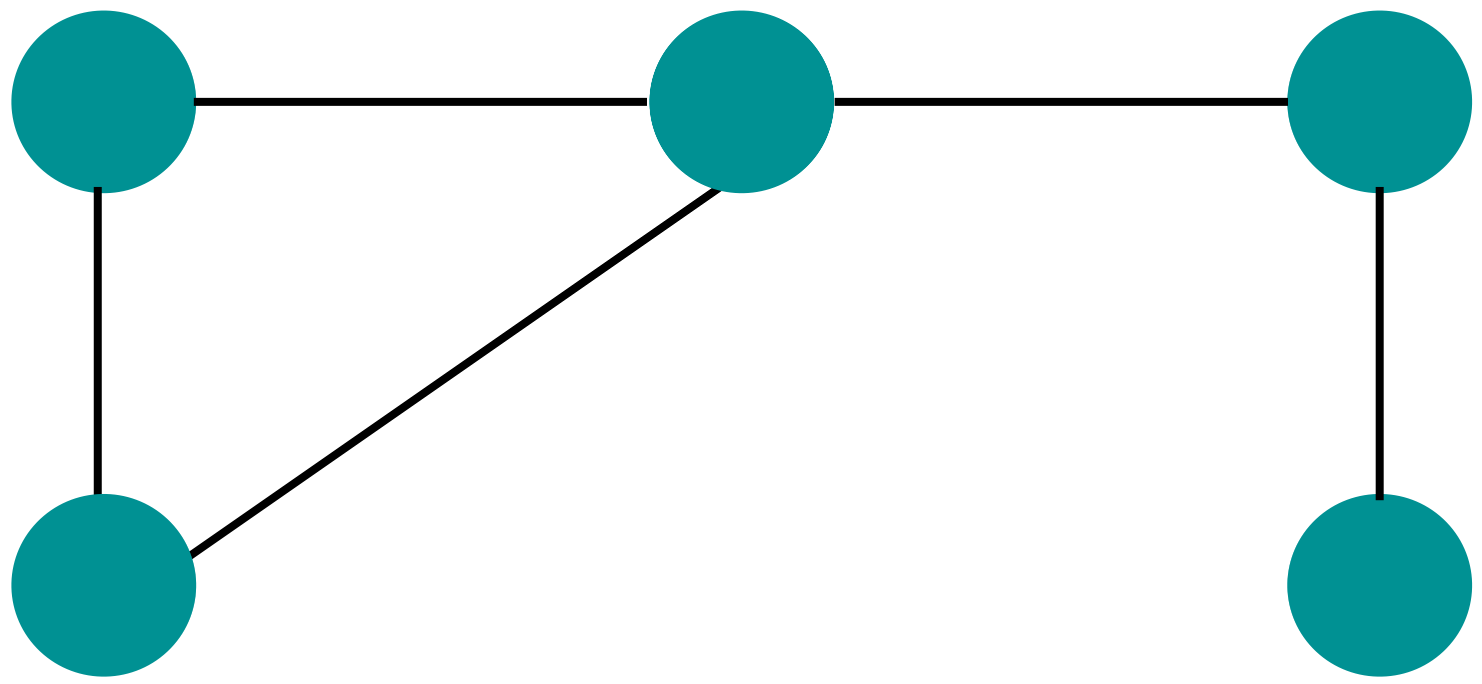


vulnerabilities in a network

Articulation Points and Bridges
Definitions

- Let $G = (V, E)$ be connected.

A vertex $v \in V$ is an articulation point (cut vertex) iff $G[V \setminus \{v\}]$ is not connected

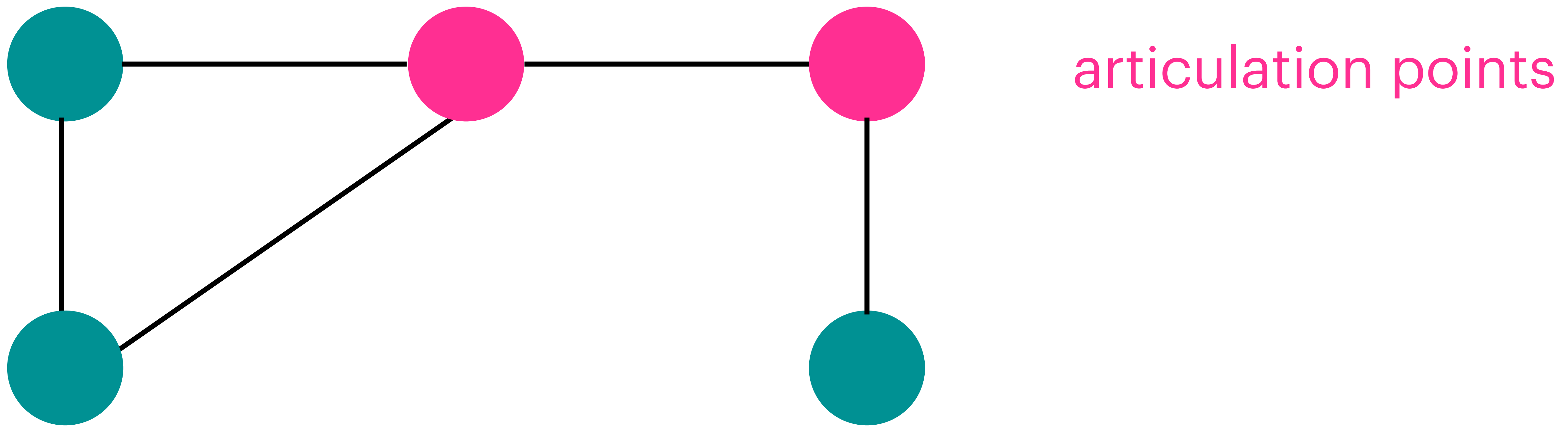


Articulation Points and Bridges

Definitions

- Let $G = (V, E)$ be connected.

  A vertex $v \in V$ is an articulation point (cut vertex) iff $G[V \setminus \{v\}]$ is not connected



Articulation Points and Bridges
Definitions

- Let $G = (V, E)$ be connected.

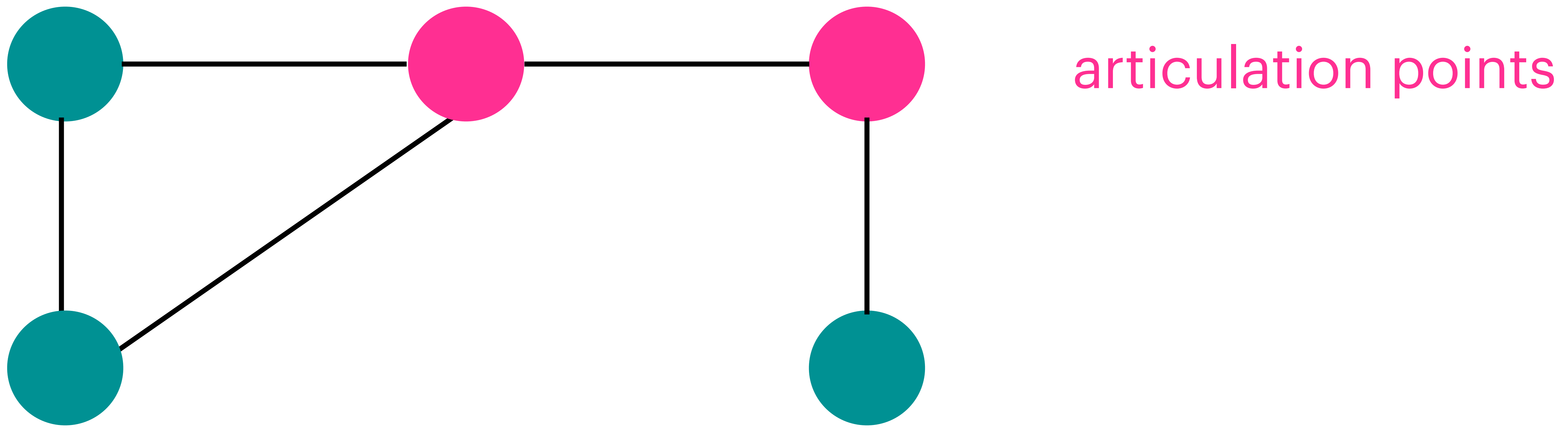
  A vertex $v \in V$ is an articulation point (cut vertex) iff $G[V \setminus \{v\}]$ is not connected

Articulation Points and Bridges

Definitions

- Let $G = (V, E)$ be connected.

  A vertex $v \in V$ is an articulation point (cut vertex) iff $G[V \setminus \{v\}]$ is not connected

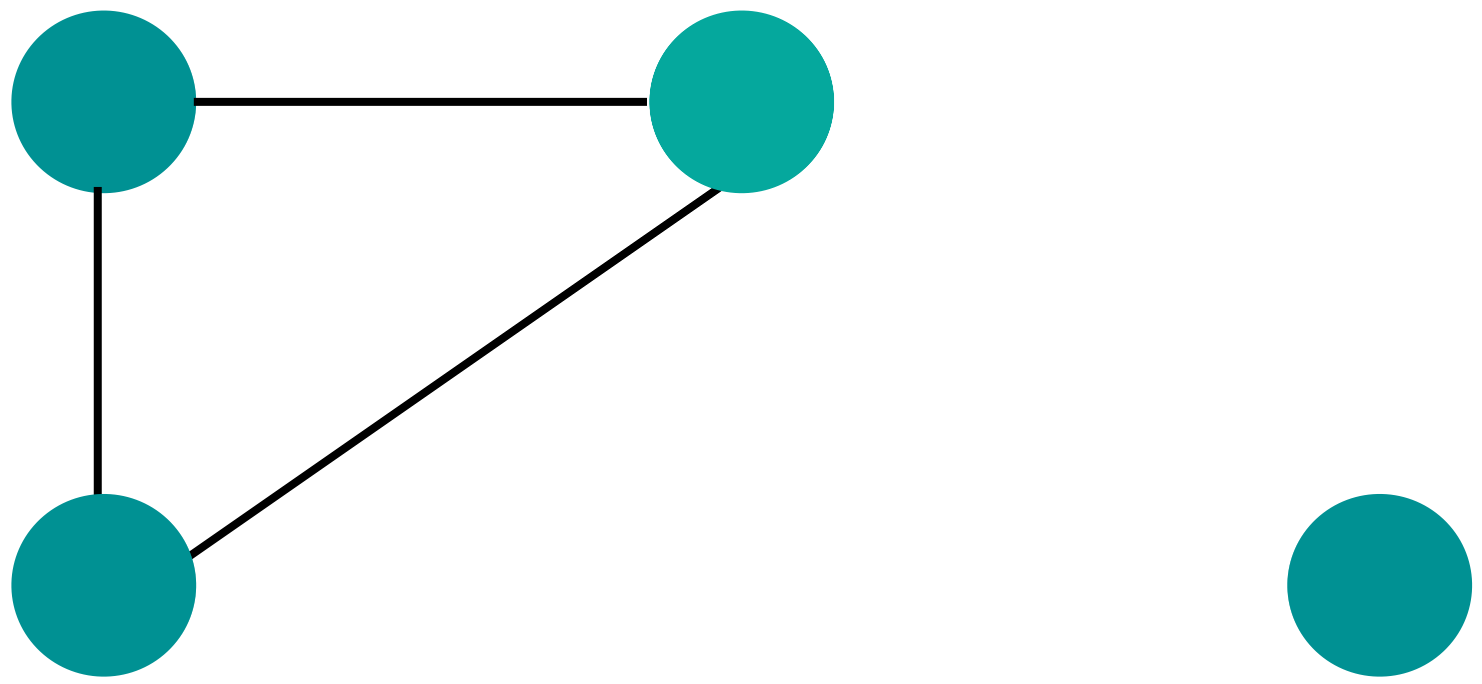


Articulation Points and Bridges

Definitions

- Let $G = (V, E)$ be connected.

A vertex $v \in V$ is an articulation point (cut vertex) iff $G[V \setminus \{v\}]$ is not connected

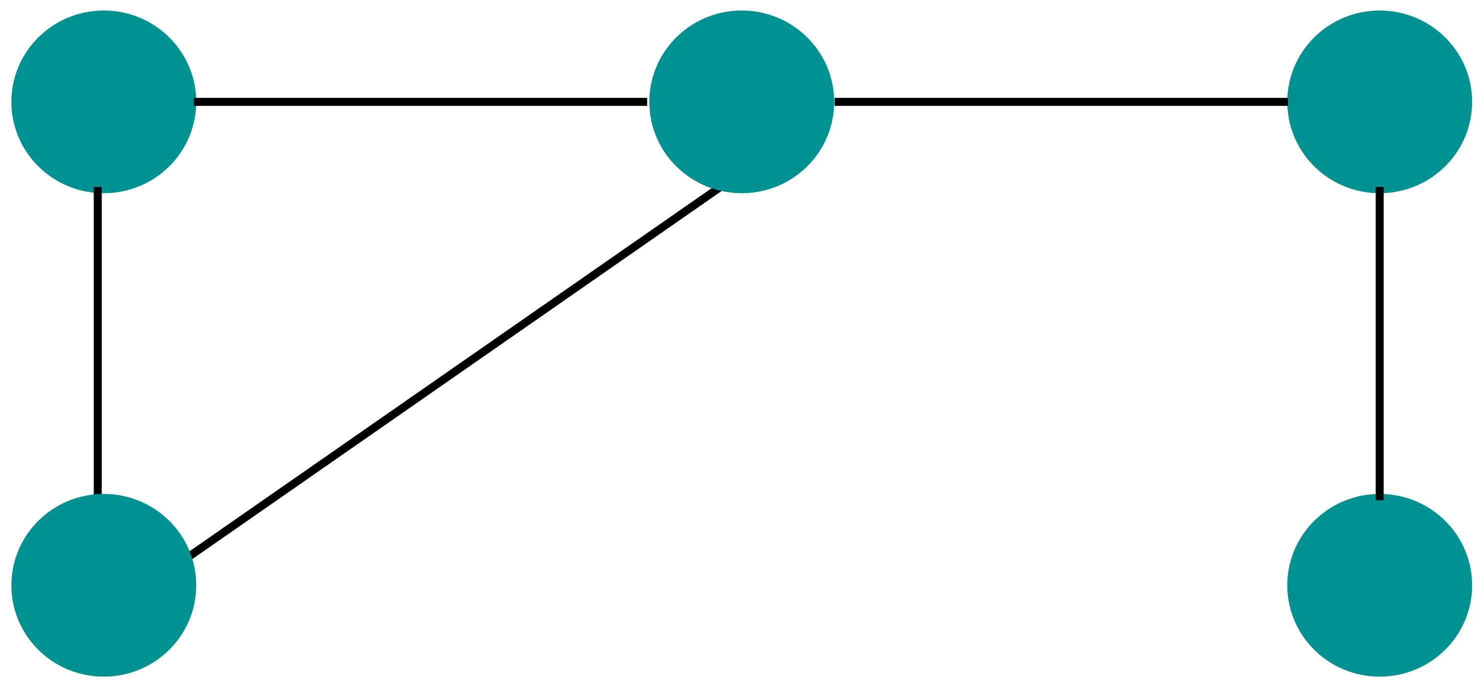


Articulation Points and Bridges

Definitions

- Let $G = (V, E)$ be connected.

An edge $e \in E$ is a bridge (cut edge) iff $G - e$ is not connected

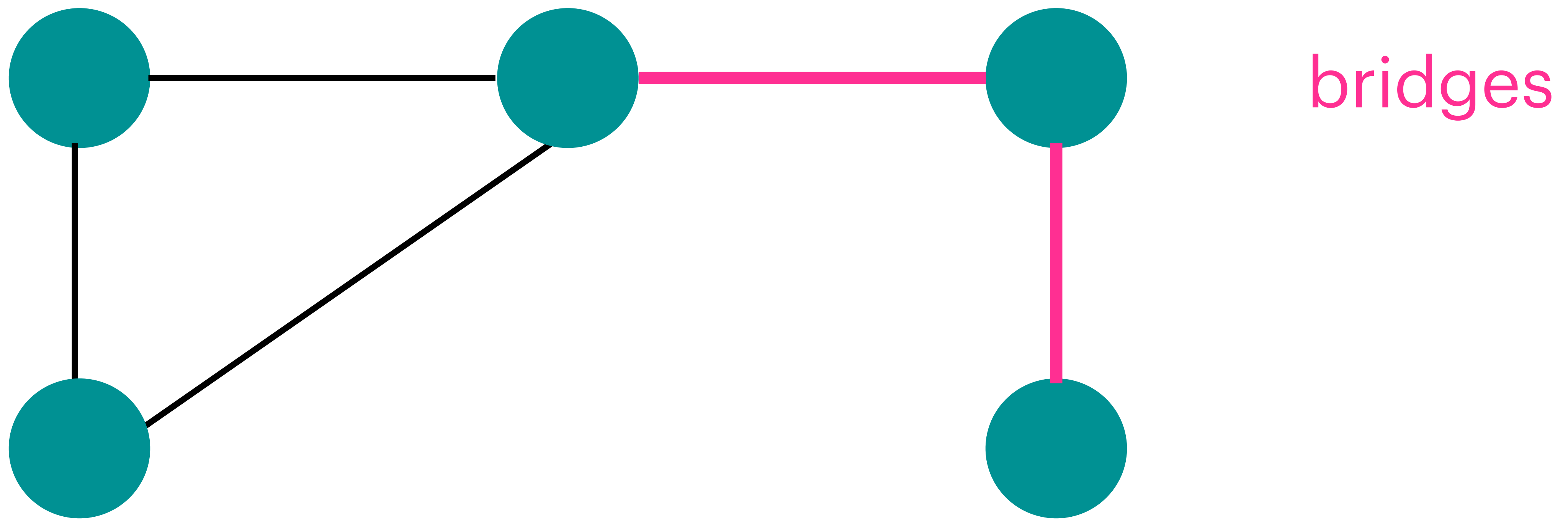


Articulation Points and Bridges

Definitions

- Let $G = (V, E)$ be connected.

A vertex $e \in E$ is a bridge (cut edge) iff $G - e$ is not connected

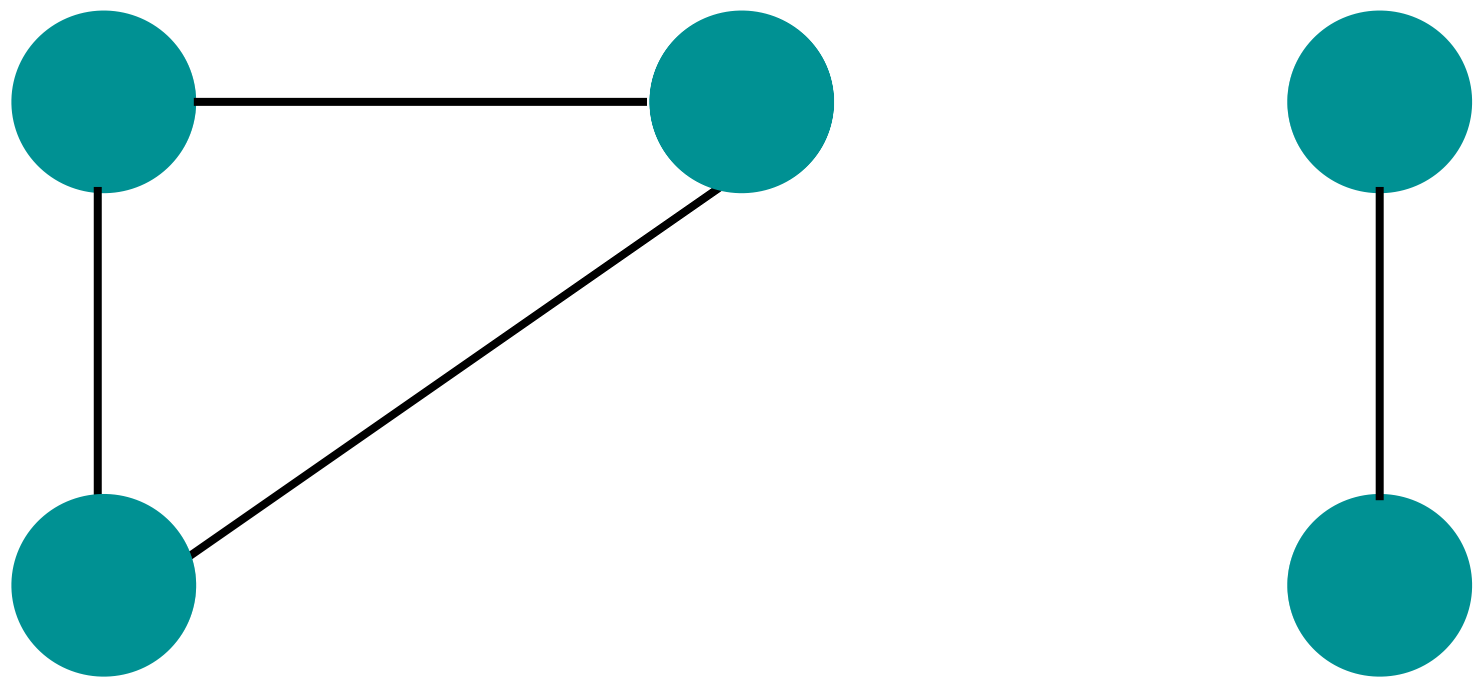


Articulation Points and Bridges

Definitions

- Let $G = (V, E)$ be connected.

  A vertex $e \in E$ is a bridge (cut edge) iff $G - e$ is not connected

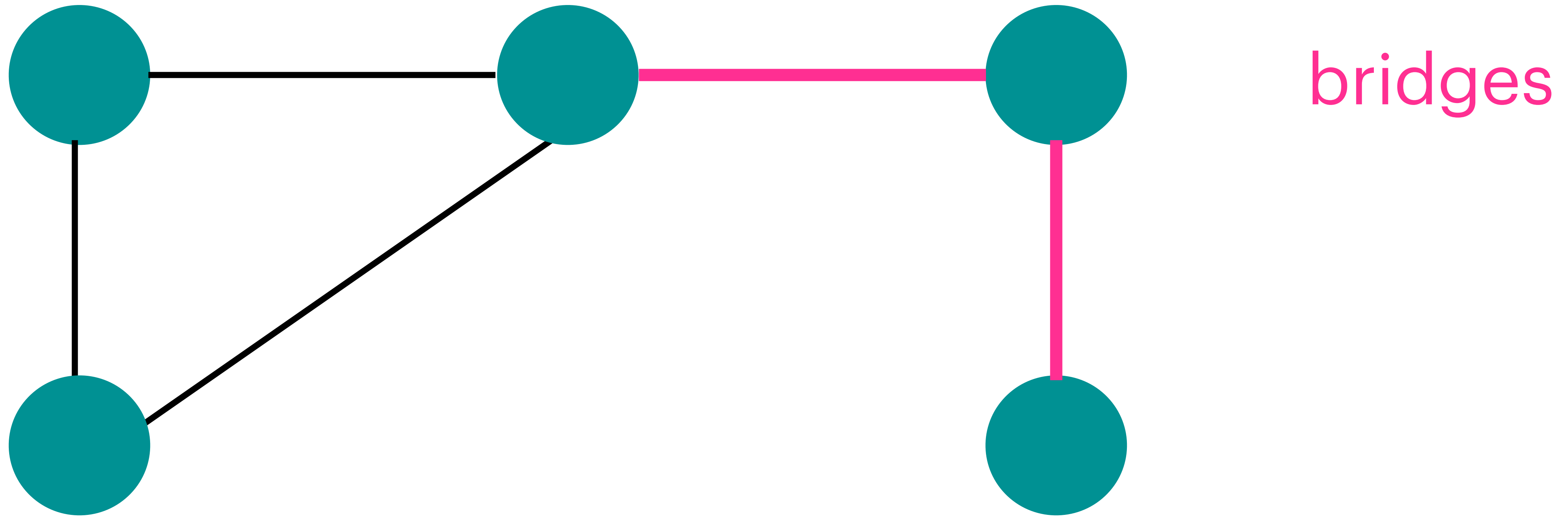


Articulation Points and Bridges

Definitions

- Let $G = (V, E)$ be connected.

A vertex $e \in E$ is a bridge (cut edge) iff $G - e$ is not connected

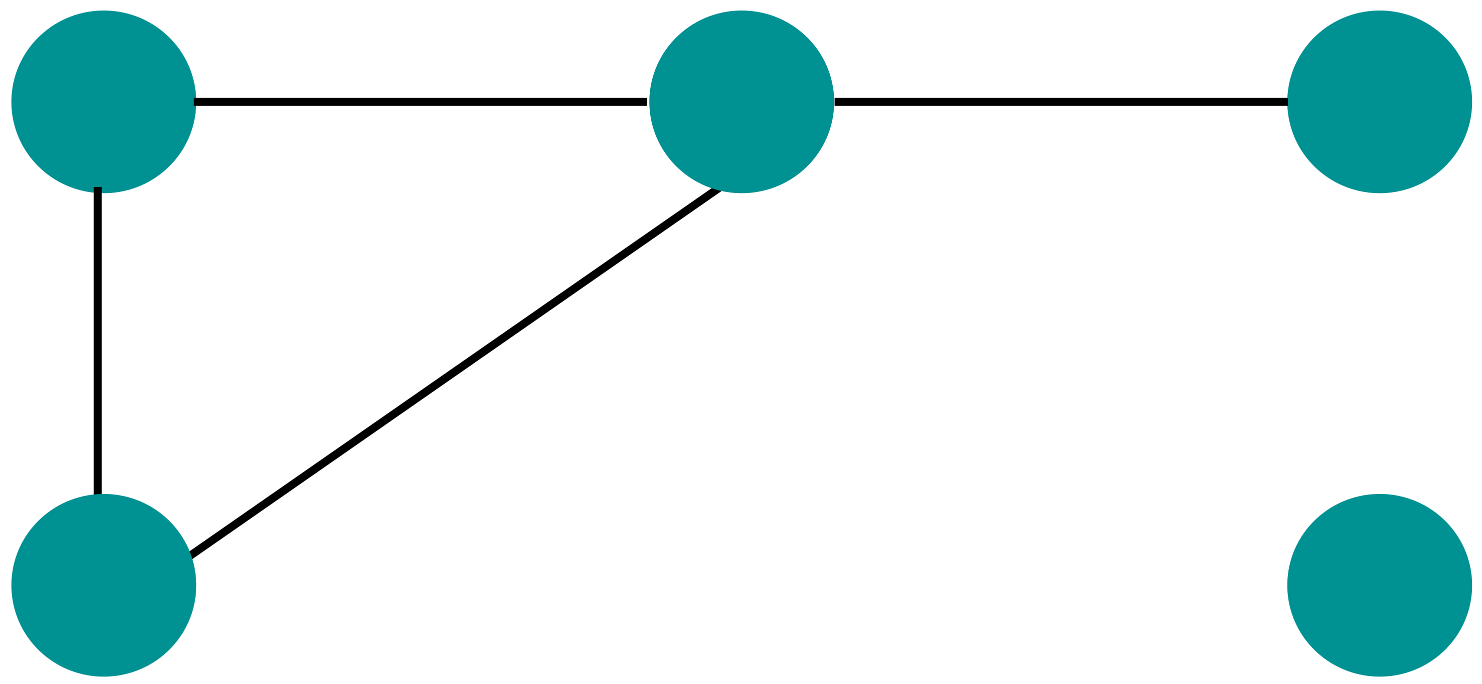


Articulation Points and Bridges

Definitions

- Let $G = (V, E)$ be connected.

A vertex $e \in E$ is a bridge (cut edge) iff $G - e$ is not connected



Articulation Points and Bridges

Lemma

- Let $G = (V, E)$ be a connected graph.

$\{x, y\} \in E$ is a bridge $\Rightarrow$ deg(x) = 1 or x is an articulation point

$\not\Leftarrow$

minitest 2

Articulation Points and Bridges

Definition

- Let $G = (V, E)$ be a graph.

  The equivalence relation $\sim$ on E is defined as :

$$e \sim f \quad := \begin{cases} e = f & \text{or} \\ e \text{ and } f \text{ are on a common cycle} \end{cases}$$

Articulation Points and Bridges

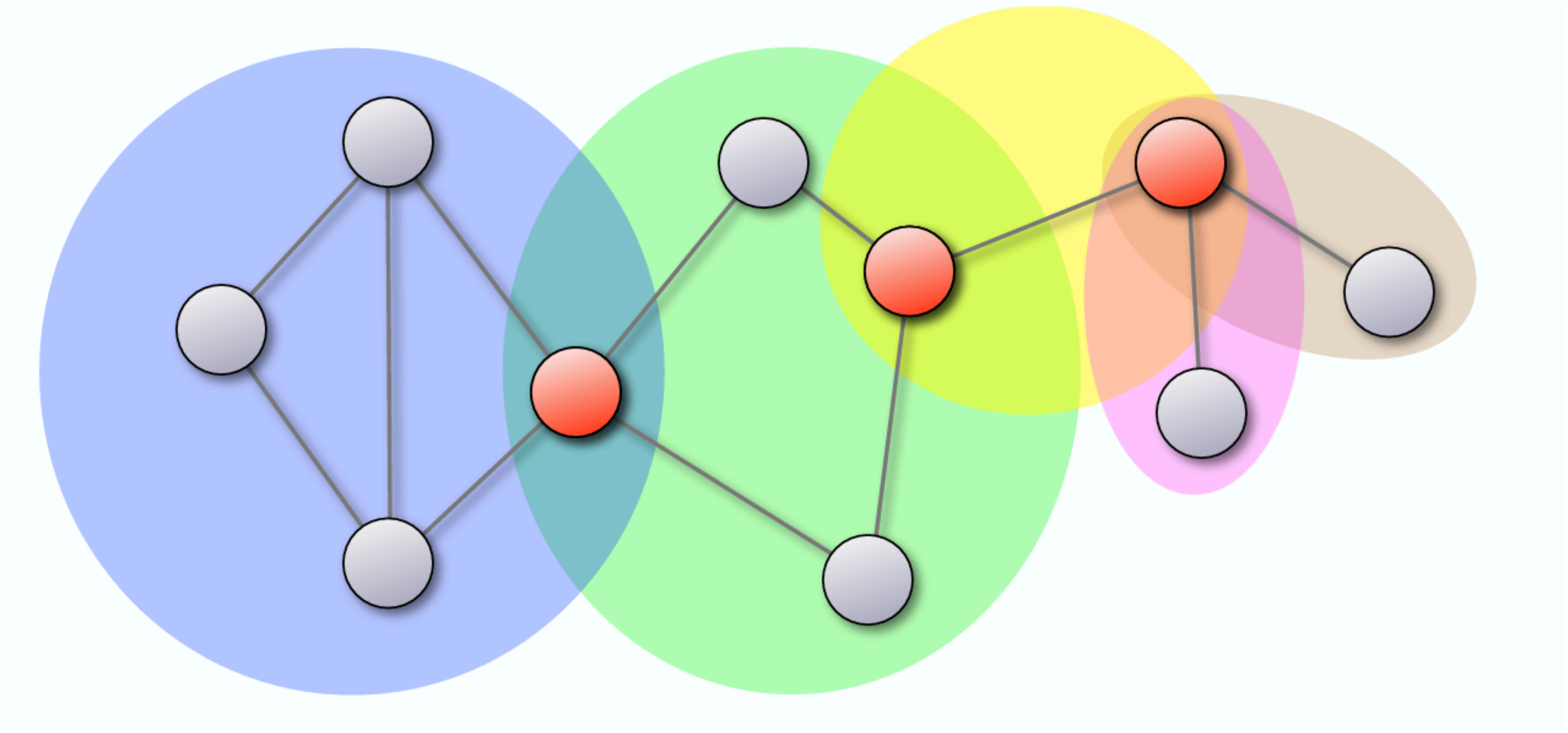
Definition

- Let $G = (V, E)$ be a graph.

  The equivalence relation $\sim$ on E is defined as :

$$e \sim f \quad := \begin{cases} e = f & \text{or} \\ e \text{ and } f \text{ are on a common cycle} \end{cases}$$

- The equivalence classes are named as Blocks



Lemma :

2 blocks always intersect at an articulation point.

Articulation point is the critical point that holds blocks together. If a graph has an articulation point, it serves as the **only connection** between two or more blocks.

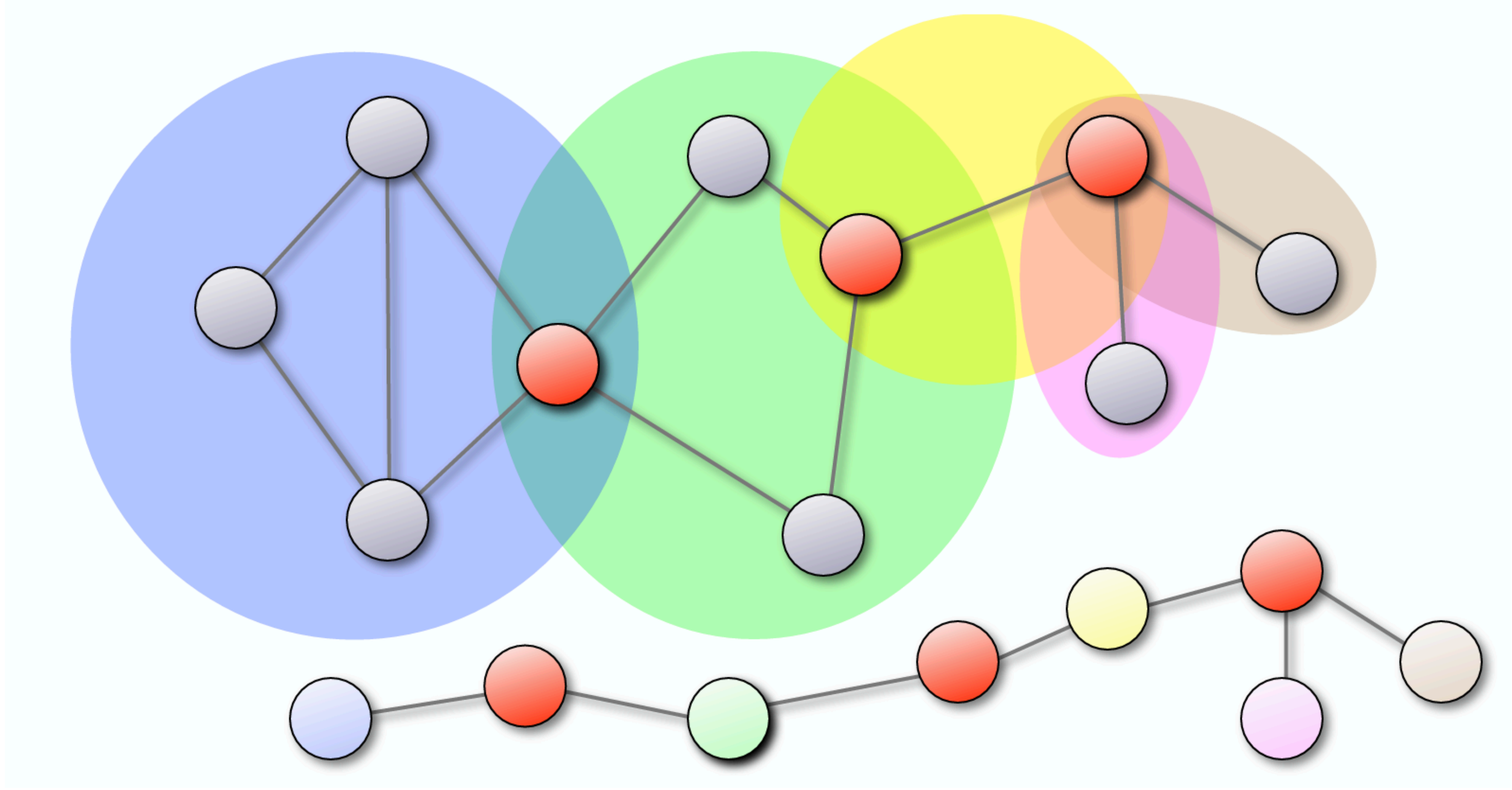
Articulation Points and Bridges

Definition

- Let $G = (V, E)$ be connected

  The Block-Graph of G is the bipartite Graph $T = \left(A \uplus B, E_T\right)$ with

  - A = {Articulation points of G}
  - B = {Block of G }
  - $\forall a \in A, b \in B : \{a, b\} \in E_T \iff a$ is incident to an edge in b

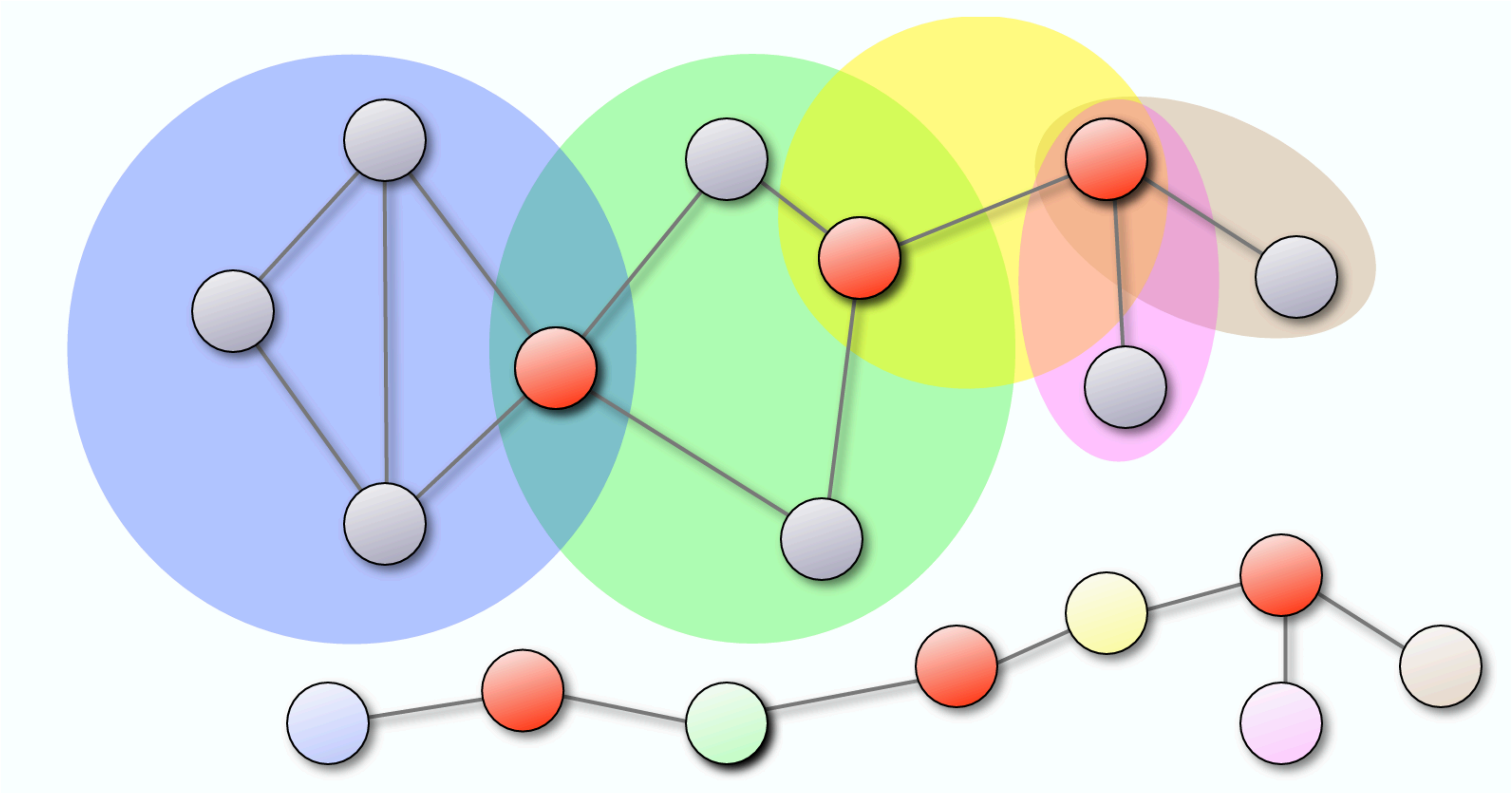


Articulation Points and Bridges
Lemma

- Let $G = (V, E)$ be connected

  The Block-Graph of G is the bipartite Graph $T = \left(A \uplus B, E_T\right)$ with

  - A = {Articulation points of G}
  - B = {Block of G }
  - $\forall a \in A, b \in B : \{a, b\} \in E_T \iff a$ is incident to an edge in b



If G is connected , then the Block-Graph of G is a tree

Articulation Points and Bridges

Finding articulation points

Queue:
First-in-first-out

Stack:
Last-in-first-out

BFS-Visit-Iterative(G, v)

```
Q ← ∅
Markiere v als aktiv
Enqueue(Q, v)
while Q ≠ ∅ do
    w ← Dequeue(Q)
    Markiere w als besucht
    for each (w, x) ∈ E do
        if x nicht aktiv und x noch nicht besucht then
            Markiere x als aktiv
            Enqueue(Q, x)
```

shortest paths

DFS-Visit-Iterative(G, v)

```
S ← ∅
Push(S, v)
while S ≠ ∅ do
    w ← Pop(S)
    if w noch nicht besucht then
        Markiere w als besucht
        for each (w, x) ∈ E in reverse order do
            if x noch nicht besucht then
                Push(S, x)
```

articulation points

Articulation Points and Bridges

Finding articulation points

```
DFS-VISIT-ITERATIVE(G, v)

S ← ∅
PUSH(S, v)
while S ≠ ∅ do
    w ← POP(S)
    if w noch nicht besucht then
        Markiere w als besucht
        for each (w, x) ∈ E in reverse order do
            if x noch nicht besucht then
                PUSH(S, x)
```

\+

Calculate low[v]

Articulation Points and Bridges

Finding articulation points

```
DFS-Visit-Iterative(G, v)

S ← ∅
Push(S, v)
while S ≠ ∅ do
    w ← Pop(S)
    if w noch nicht besucht then
        Markiere w als besucht
        for each (w, x) ∈ E in reverse order do
            if x noch nicht besucht then
                Push(S, x)
```

+

Calculate low[v]

low[v] := the smallest dfs-number that one can reach from v with a directed path consisting of (any number of) tree edges and maximum one remaining edge .

Articulation Points and Bridges

Finding articulation points

DFS-Visit-Iterative(G, v)

```
S ← ∅
Push(S, v)
while S ≠ ∅ do
    w ← Pop(S)
    if w noch nicht besucht then
        Markiere w als besucht
        for each (w, x) ∈ E in reverse order do
            if x noch nicht besucht then
                Push(S, x)
```

+

Calculate low[v]

low[v] := the smallest dfs-number that one can reach from v with a directed path consisting of (any number of) tree edges and maximum one remaining edge .

Articulation Points and Bridges

Finding articulation points

```
DFS-Visit-Iterative(G, v)

S ← ∅
Push(S, v)
while S ≠ ∅ do
    w ← Pop(S)
    if w noch nicht besucht then
        Markiere w als besucht
        for each (w, x) ∈ E in reverse order do
            if x noch nicht besucht then
                Push(S, x)
```

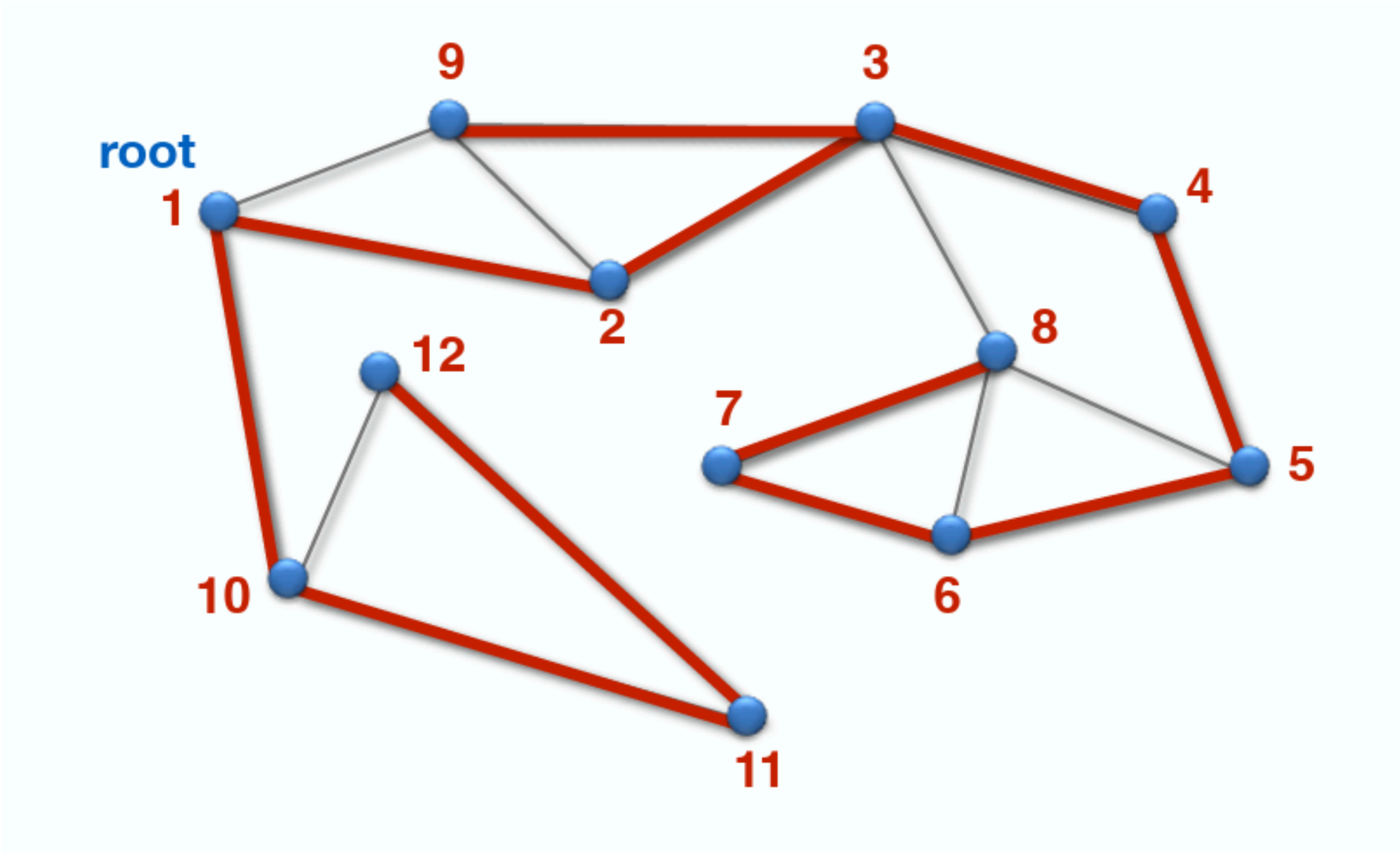
+

Calculate low[v]

low[v] := the smallest dfs-number that one can reach from v with a directed path consisting of (any number of) tree edges and maximum one remaining edge .

Articulation Points and Bridges

Finding articulation points

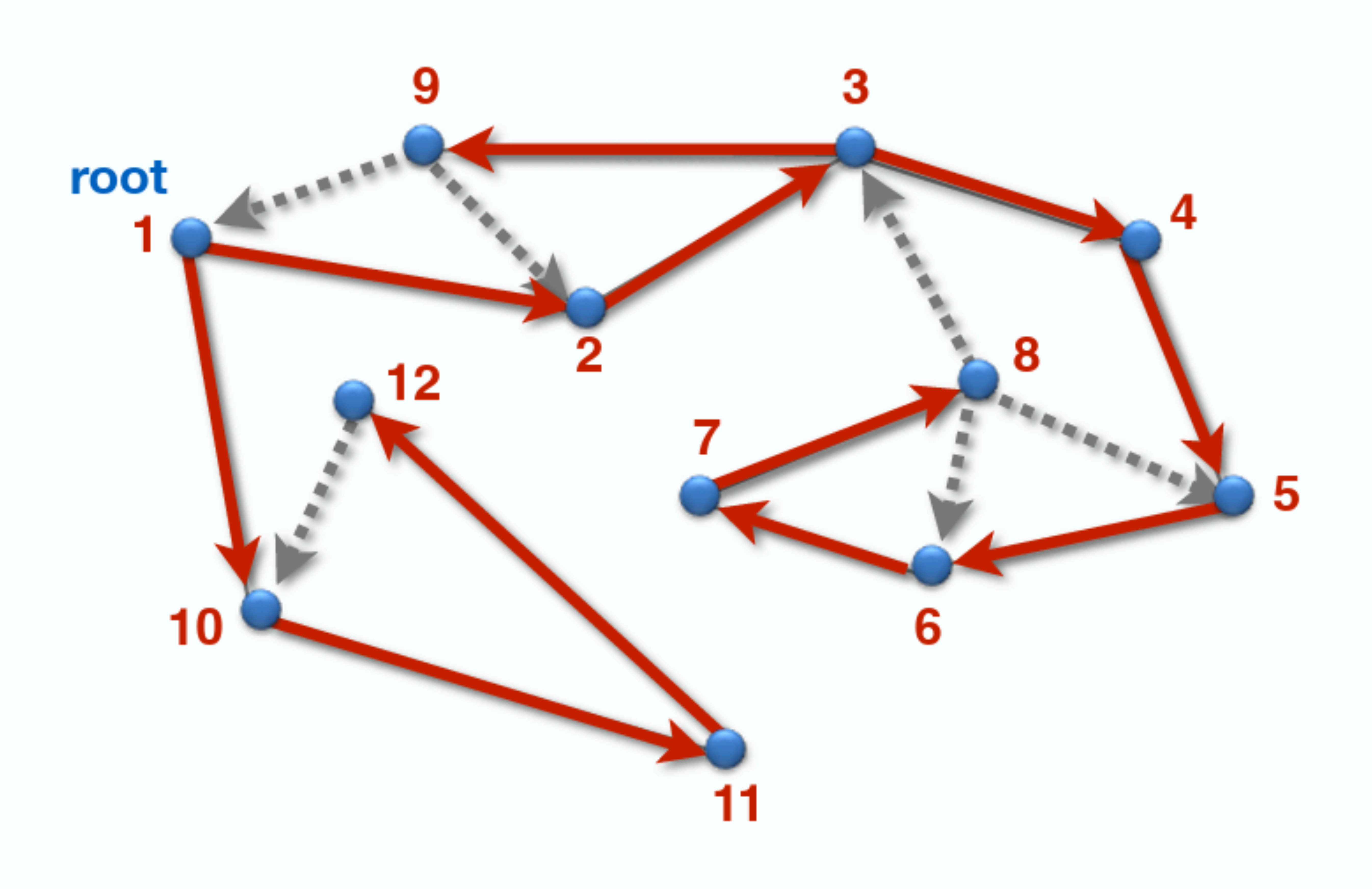


1) Find DFS numbers

- DFS from A&D
- pre-number from A&D is now the DFS number
- back edge/forward edge/ cross edge are now all remaining edges

Articulation Points and Bridges

Finding articulation points



tree edges

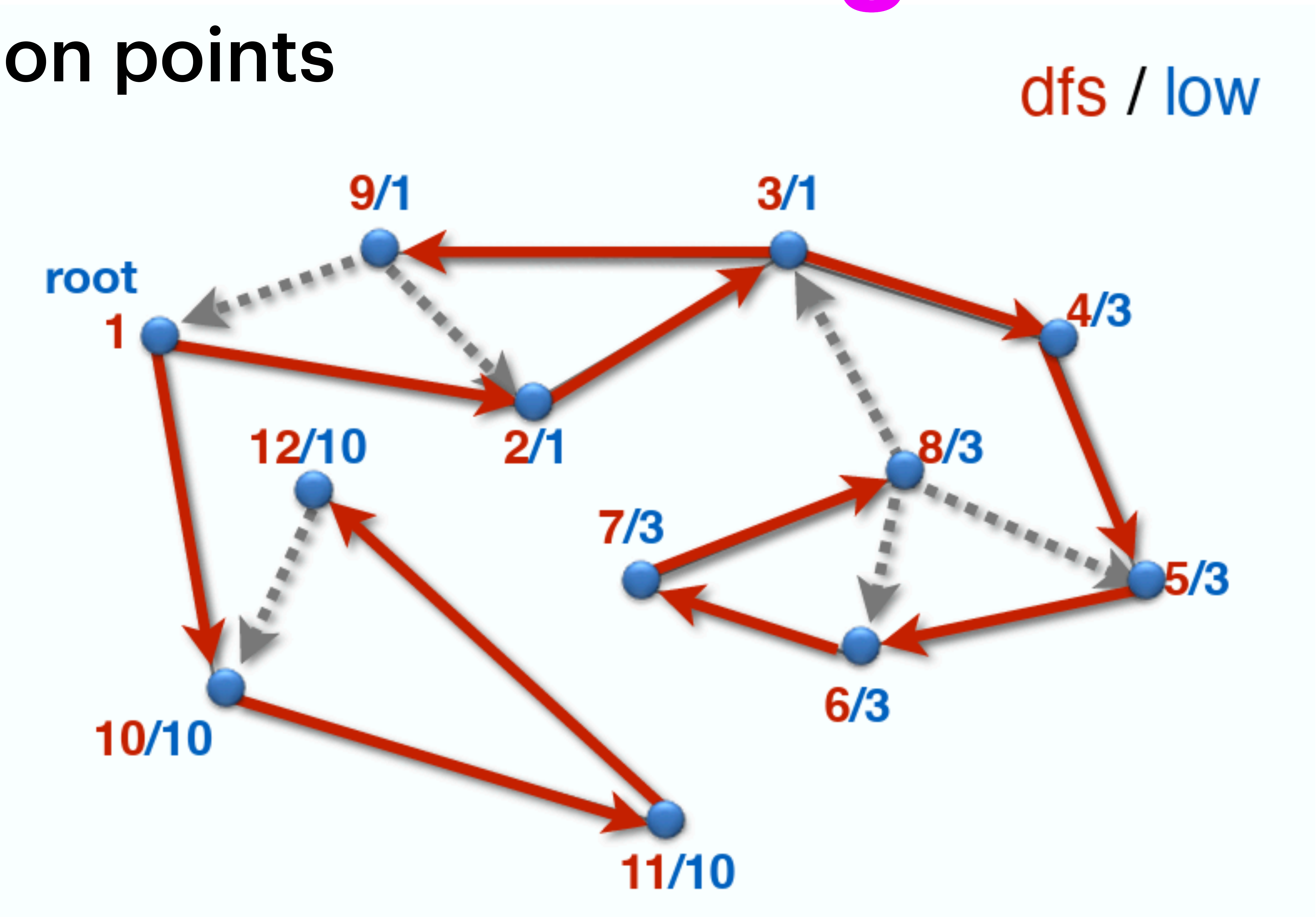
remaining edges

2) Orientation

- tree edges in the increasing dfs-number direction
- remaining edges in decreasing dfs-number direction

Articulation Points and Bridges

Finding articulation points

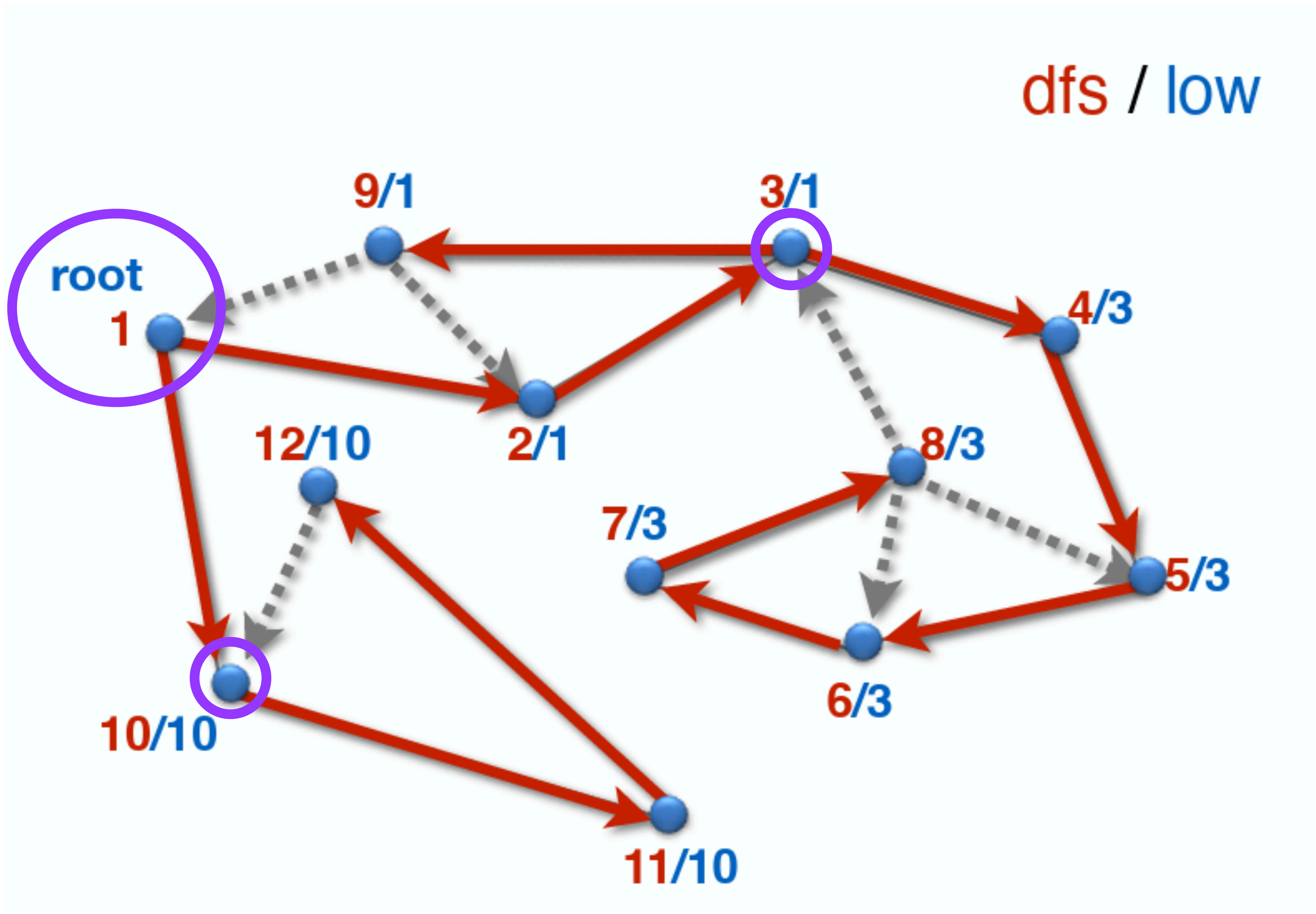


3) Calculate low[v]

$$\text{low}[v] = \min\left(\text{dfs}[v], \quad \min_{(v,w)\in E}\begin{cases}\text{dfs}[w], & \text{if (v,w) is a remaining edge}\\ \text{low}[w], & \text{if (v,w) is a tree edge}\end{cases}\right)$$

Articulation Points and Bridges

Finding articulation points



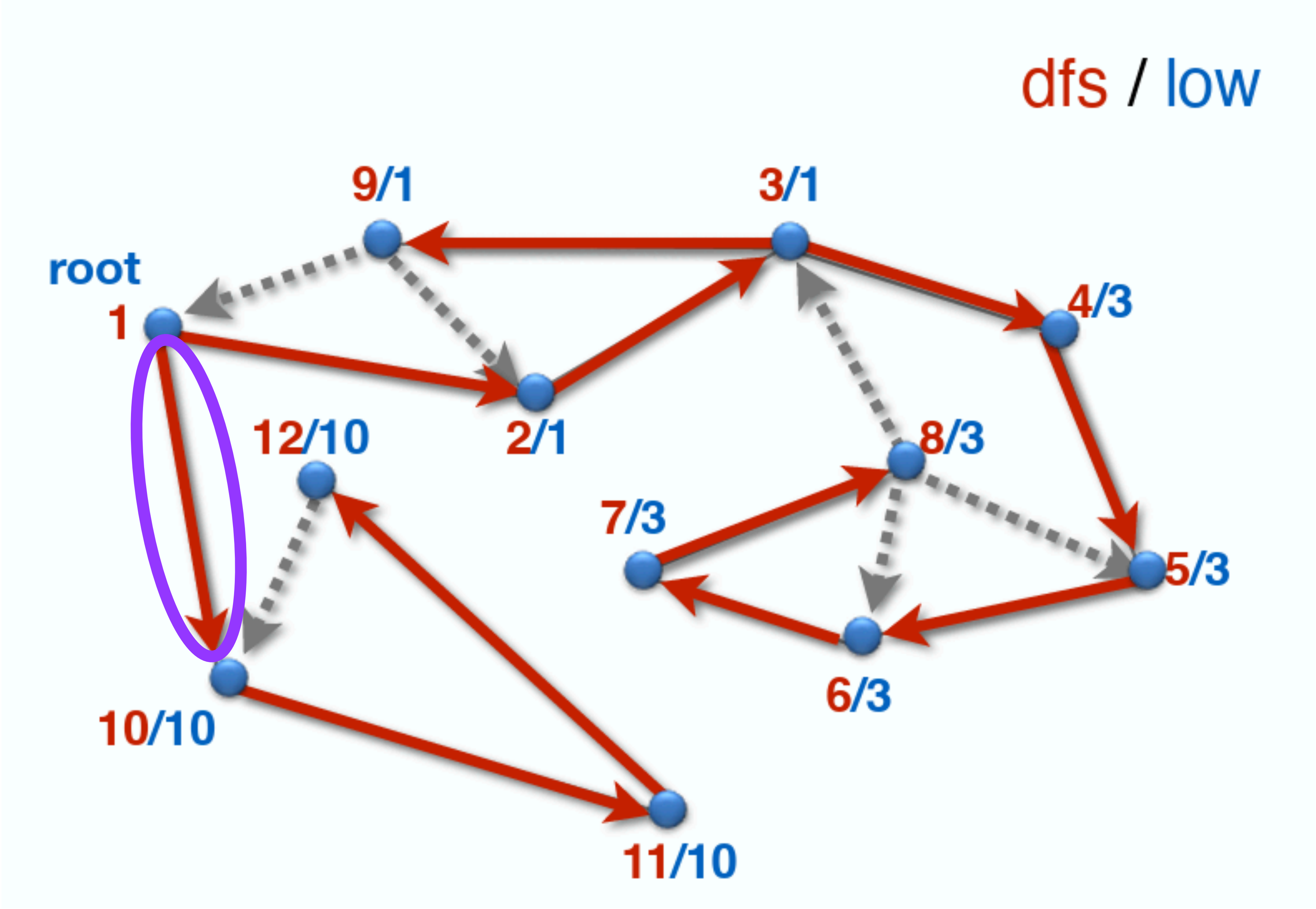
A vertex v is an **articulation point** iff

1) v ≠ root and v has a child u in DFS-Tree with low[u] ≥ dfs[v] or

2) v = root and v has at least 2 children in DFS-Tree

Articulation Points and Bridges

Finding articulation points



A tree edge e = (v,w) is a **bridge** iff

low[w] > dfs[v]

Remaining edges can never be a bridge

Cycles

Cycles
Definitions

Closed walk

- A sequence of vertices $(v_0, v_1, \ldots, v_k)$ is a **closed walk** (german "Zyklus") if it is a walk, $k \geq 2$ and $v_0 = v_k$.

Cycle

- A sequence of vertices $(v_0, v_1, \ldots, v_k)$ is a **cycle** (german "Kreis") if it is a closed walk, $k \geq 3$ and all vertices (except v_0 and v_k) are distinct.

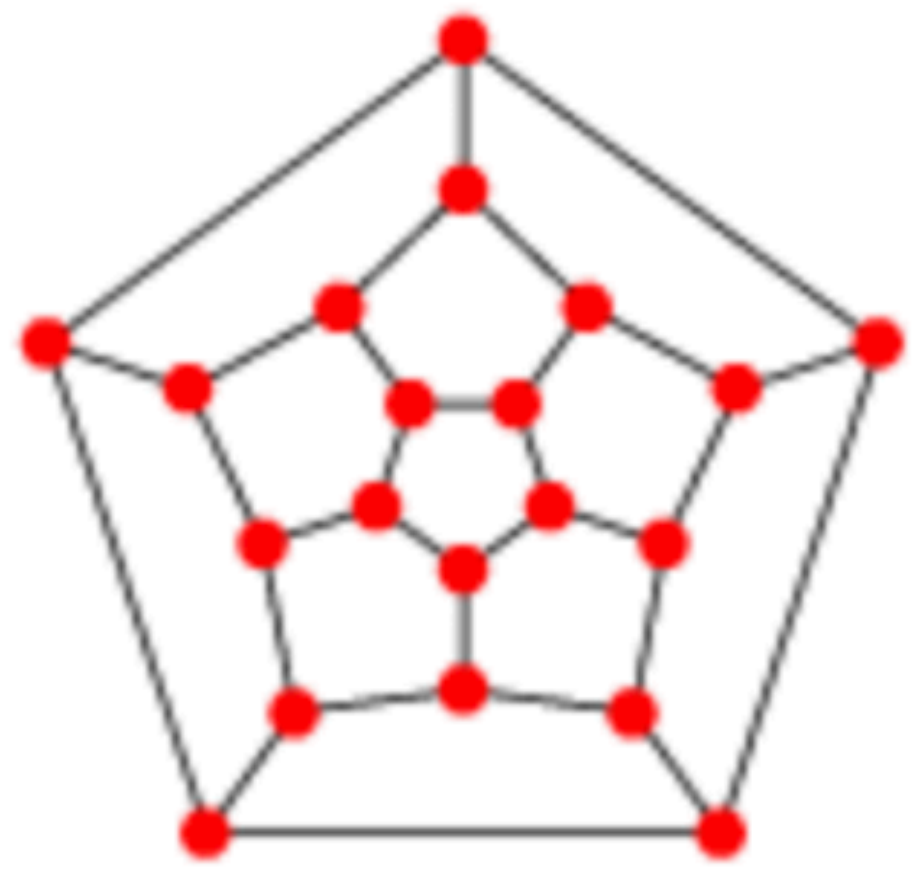
- Hamlitonian Cycle
  - A cycle in G that contains every vertex exactly once

- Eulerian Cycle
  - A closed walk in G that contains every edge exactly once

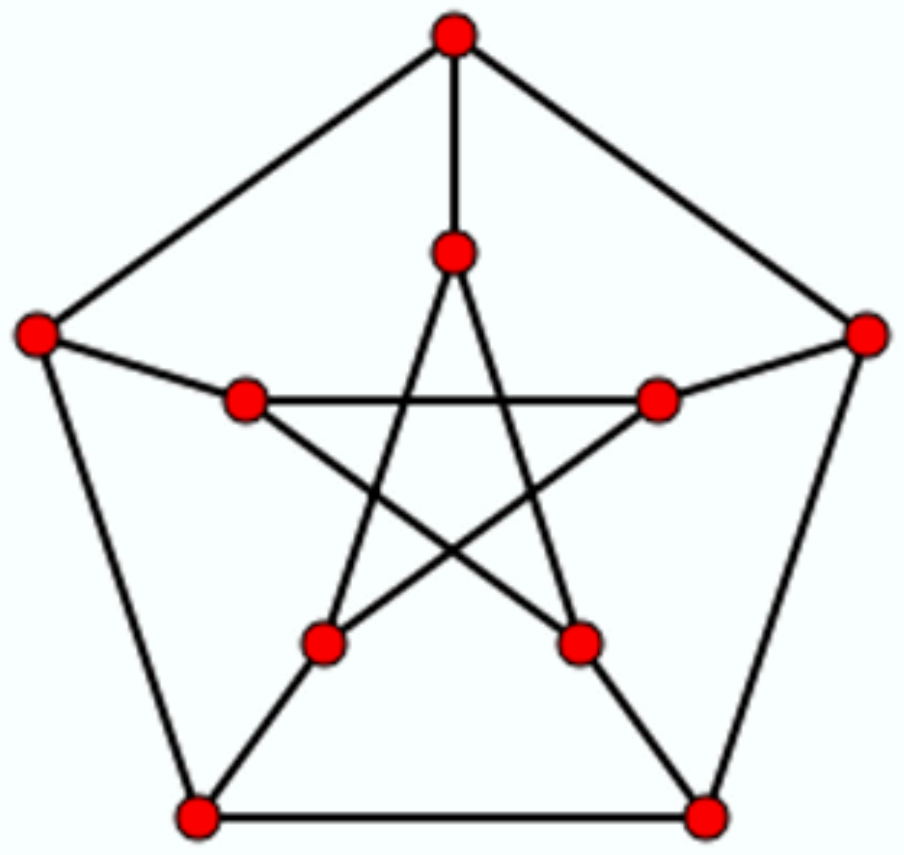
minitest 4,6,7

Cycles
Hamiltonian Cycle Examples

- Hamlitonian Cycle
  - A cycle in G that contains every vertex exactly once



Ikosaeder

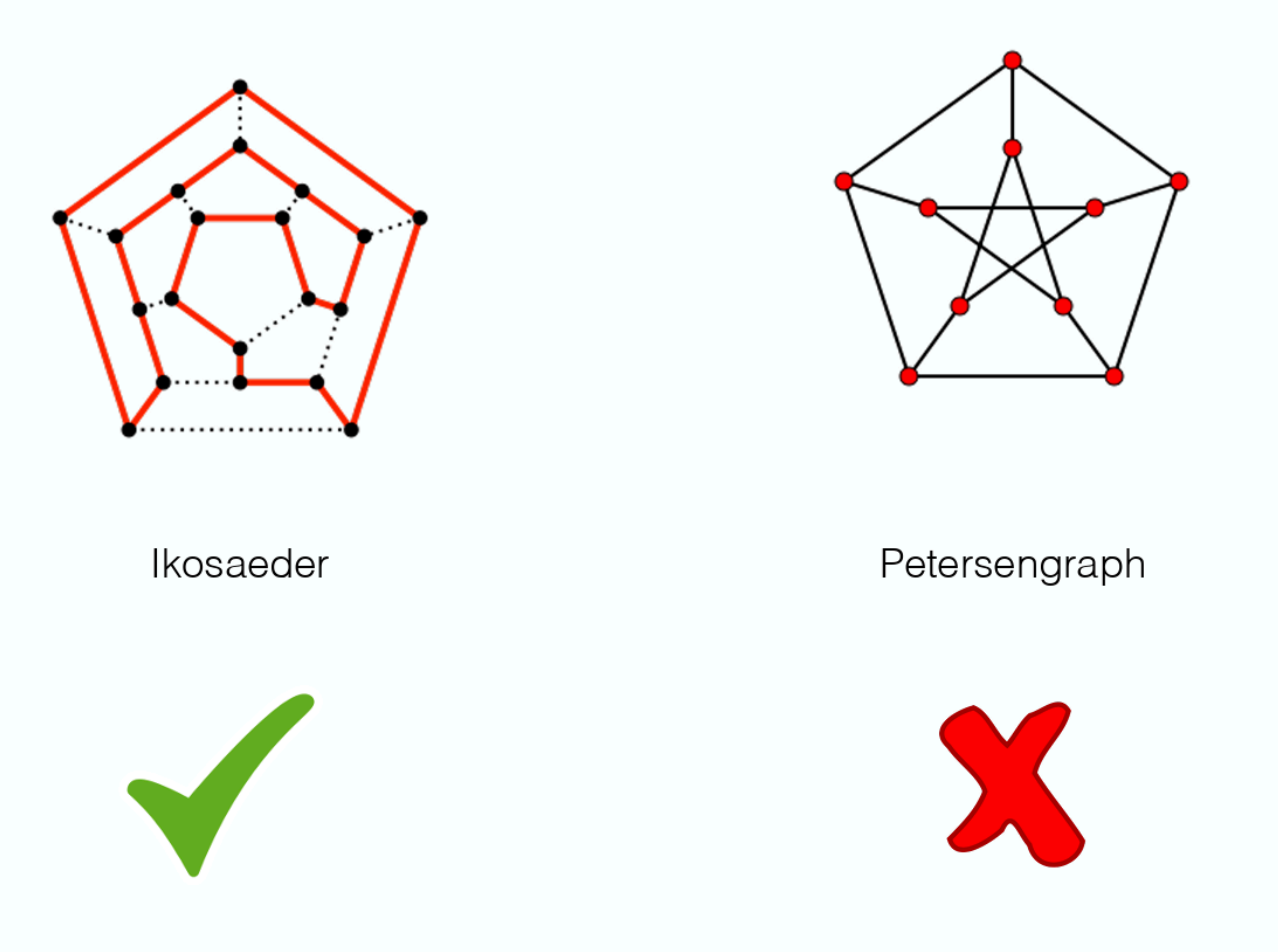


Petersengraph

Cycles

Hamiltonian Cycle Examples

- Hamlitonian Cycle
  - A cycle in G that contains every vertex exactly once

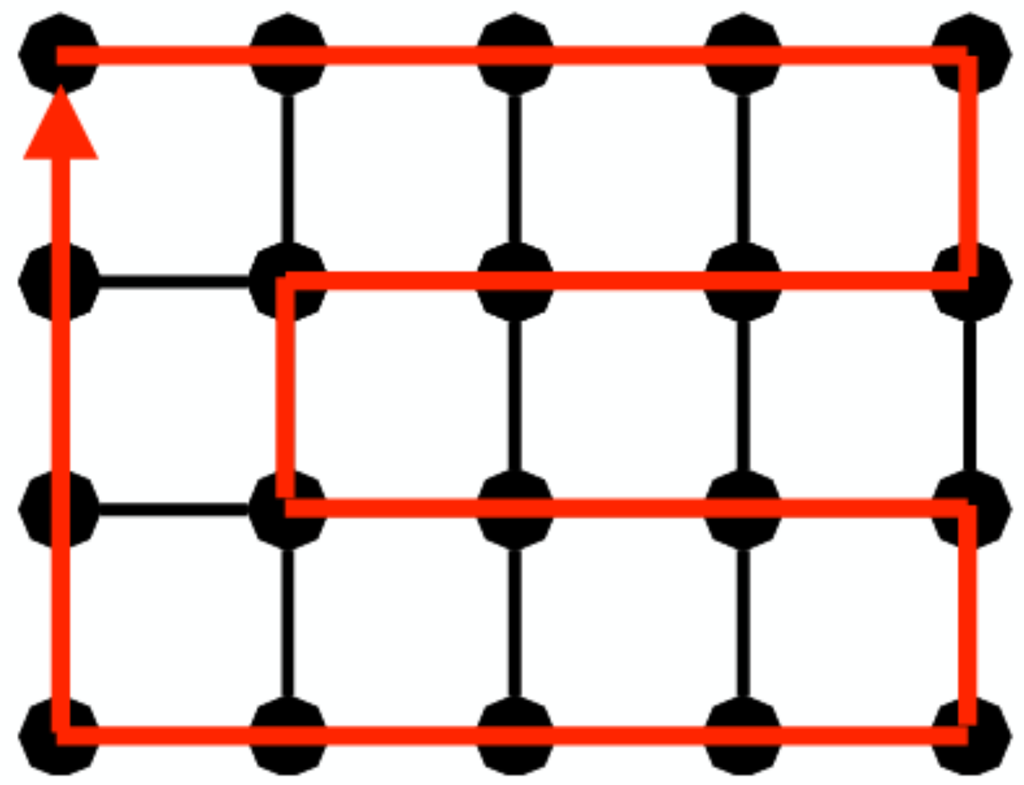


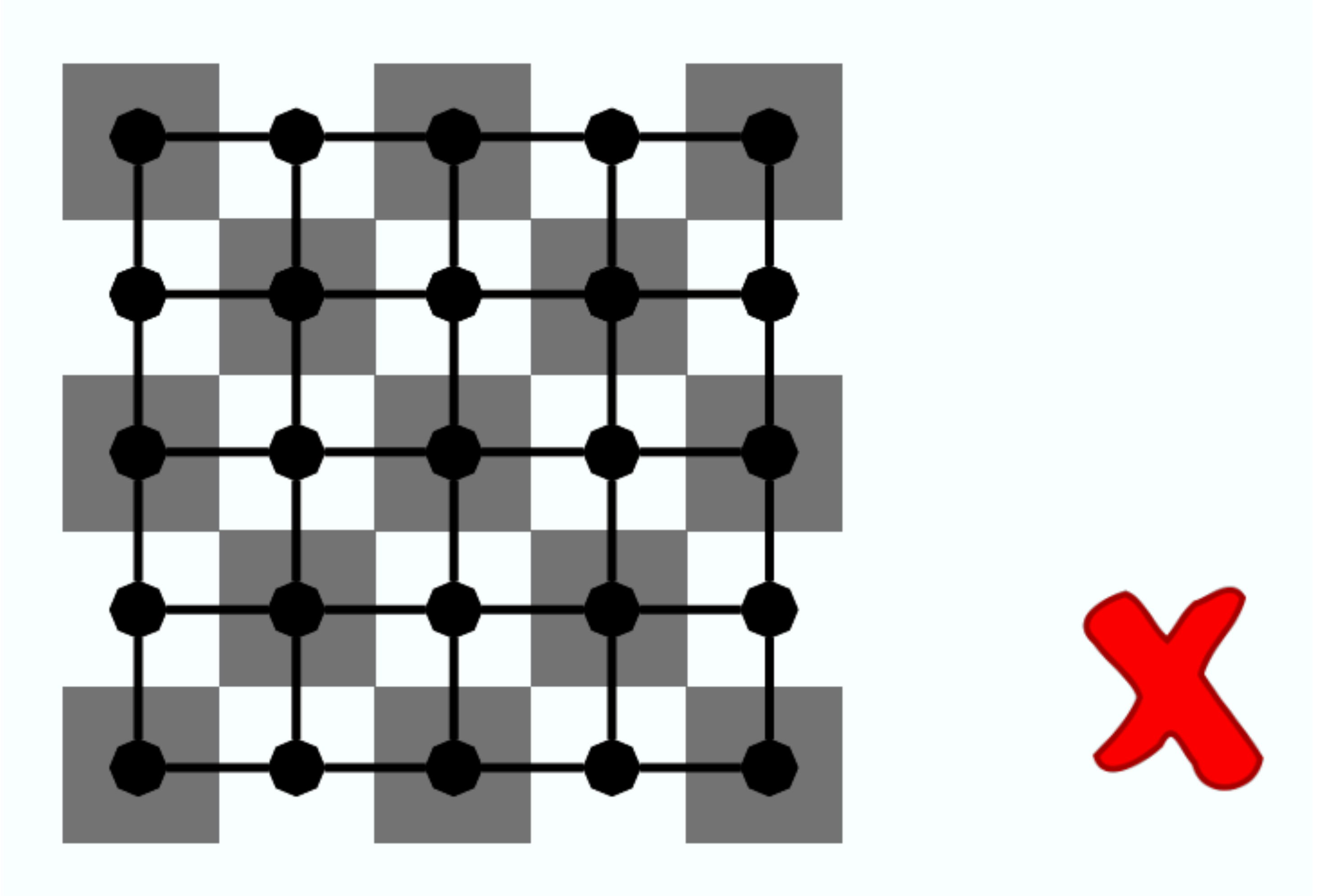
Cycles

Hamiltonian Cycle Examples

- Hamlitonian Cycle
  - A cycle in G that contains every vertex exactly once

Grid Graph





Let $m, n \geq 2$

A n x m Grid has a hamiltonian cycle iff n x m is even

Cycles

Hamiltonian Cycle Examples

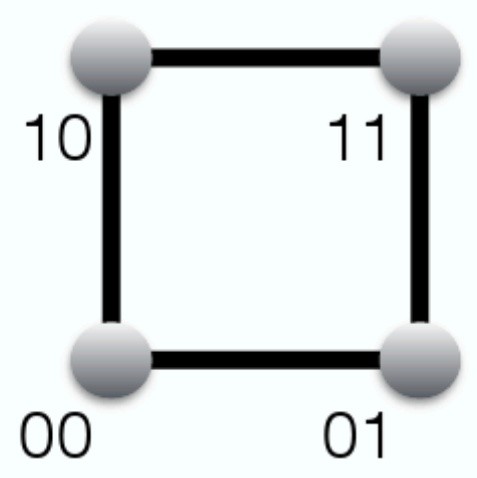
- Hamlitonian Cycle
  - A cycle in G that contains every vertex exactly once

d-dimensional Hypercube H_d

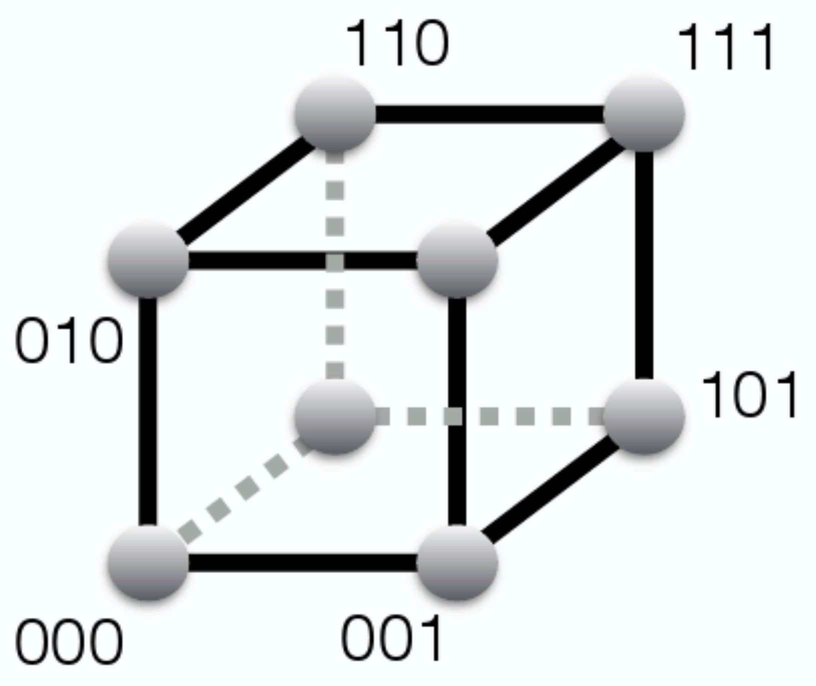
V := $\{0,1\}^d$

E := "All vertex pairs that differ in only one coordinate"

d=2:



d=3:



Has a hamiltonian cycle for all $d \geq 2$

Questions

Feedbacks , Recommendations

Nil Ozer