Incremental 2-Edge-Connectivity In Directed Graphs

Loukas Georgiadis

University of Ioannina
Greece

Giuseppe F. Italiano

University of Rome
Tor Vergata
Italy

Nikos Parotsidis

University of Rome
Tor Vergata
Italy

ICALP 2016, Rome

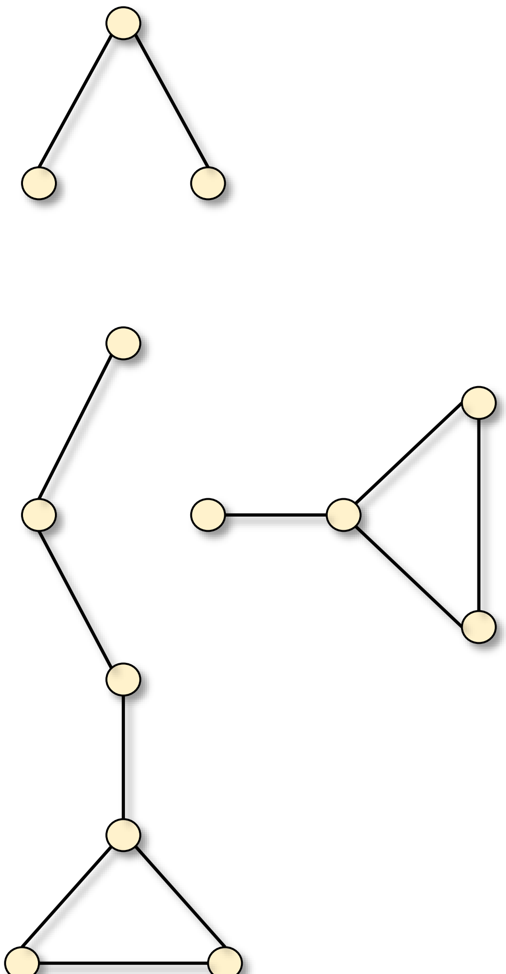
Outline

- Definitions
  - 2-edge-connectivity in undirected graphs
  - 2-edge-connectivity in directed graphs
  - Problem definition
  - Known algorithm and our result
- High-level idea
- Basic ingredients
  - Dominators
  - Auxiliary components
- Tools
- Conclusion

Outline

- Definitions
  - 2-edge-connectivity in undirected graphs
  - 2-edge-connectivity in directed graphs
  - Problem definition
  - Known algorithm and our result
- High-level idea
- Basic ingredients
  - Dominators
  - Auxiliary components
- Tools
- Conclusion

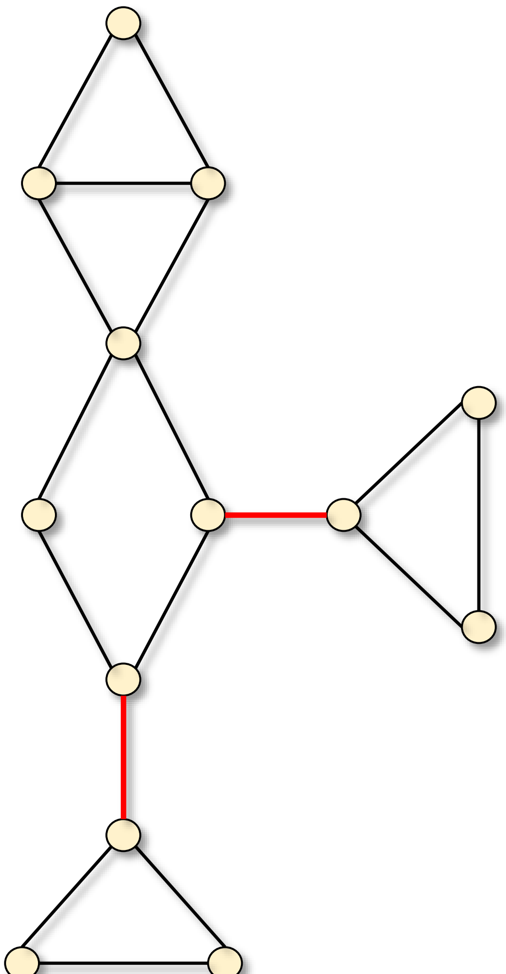
Undirected: Connected components



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

- G is **connected** if there is a path between any two vertices.
- The **connected components** of G are its maximal connected subgraphs.

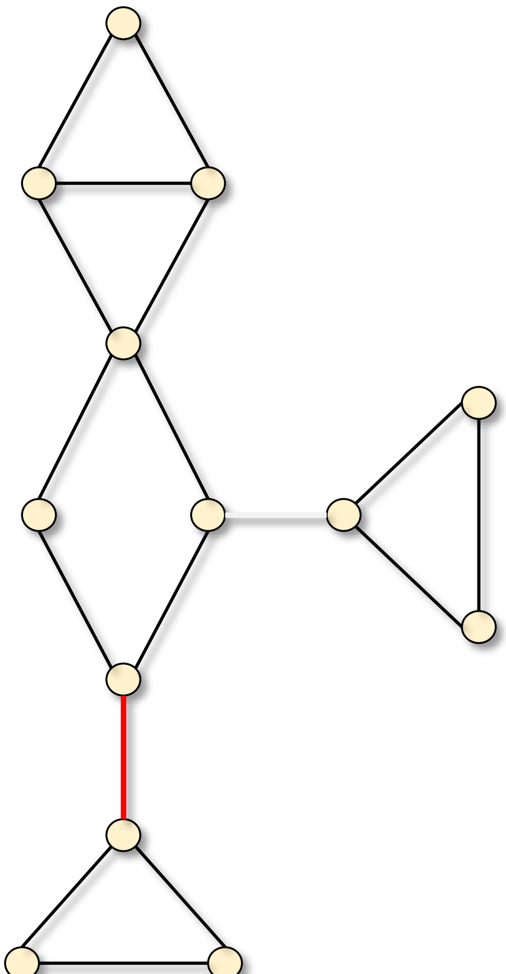
Undirected: Connected components



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

- An edge is a bridge, if its removal increases the number of connected components.

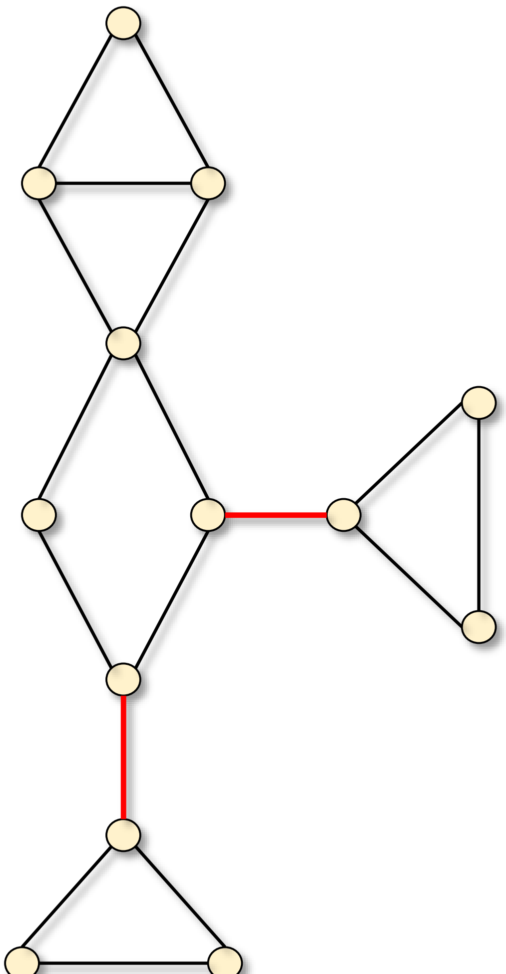
Undirected: Connected components



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

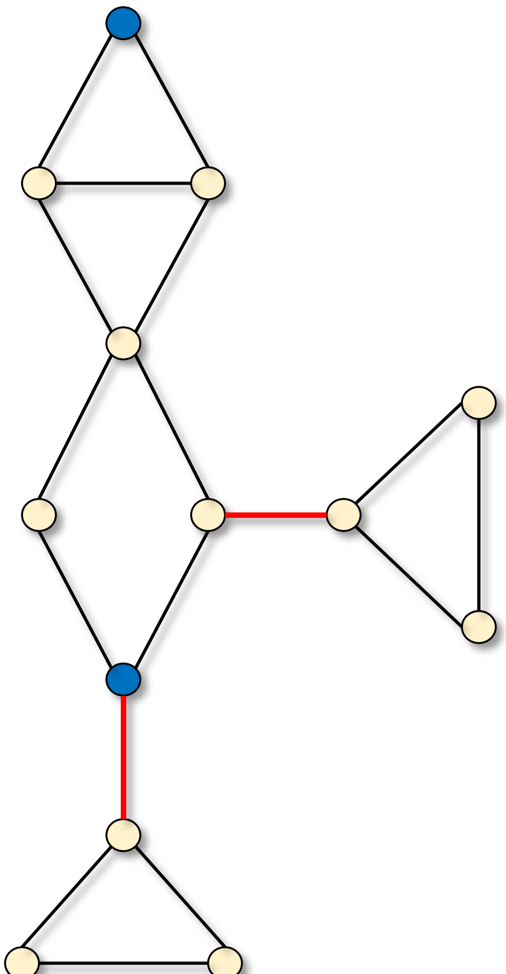
- An edge is a bridge, if its removal increases the number of connected components.

Undirected: Connected components



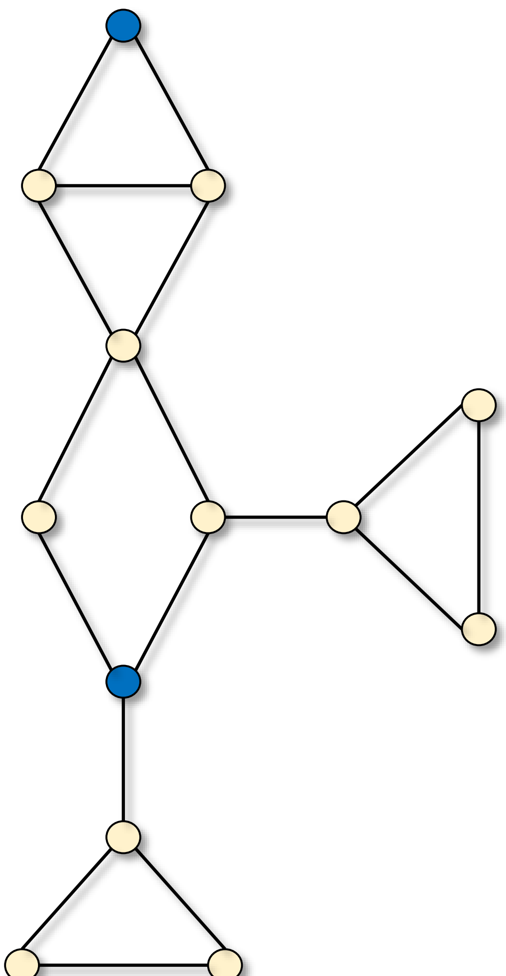
By Menger's theorem, two vertices are **2-edge-connected** iff the removal of any bridge leaves them in the same connected component.

Undirected: Connected components



By Menger's theorem, two vertices are **2-edge-connected** iff the removal of any bridge leaves them in the same connected component.

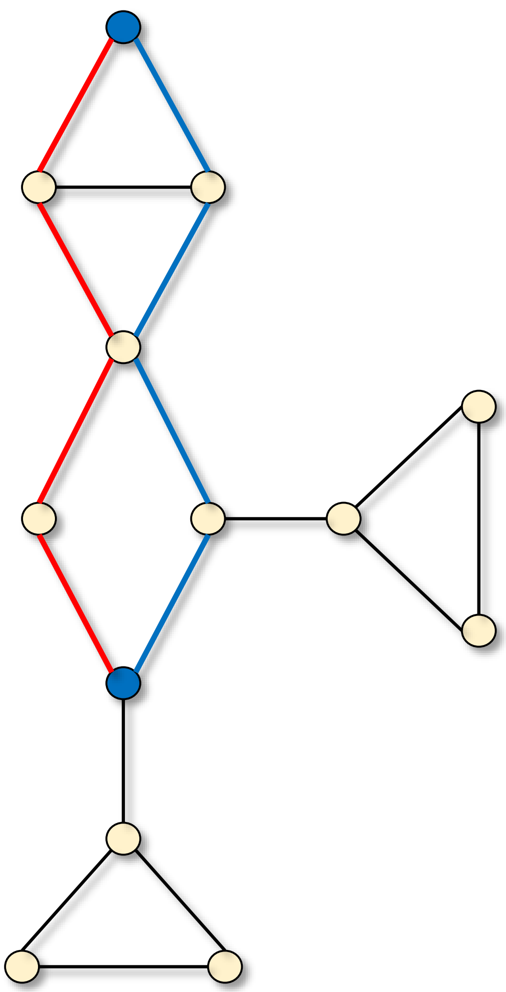
Undirected: Connected components



By Menger's theorem, two vertices are **2-edge-connected** iff the removal of any bridge leaves them in the same connected component.

Two vertices are **2-edge-connected** if there are **two edge-disjoint** paths between them.

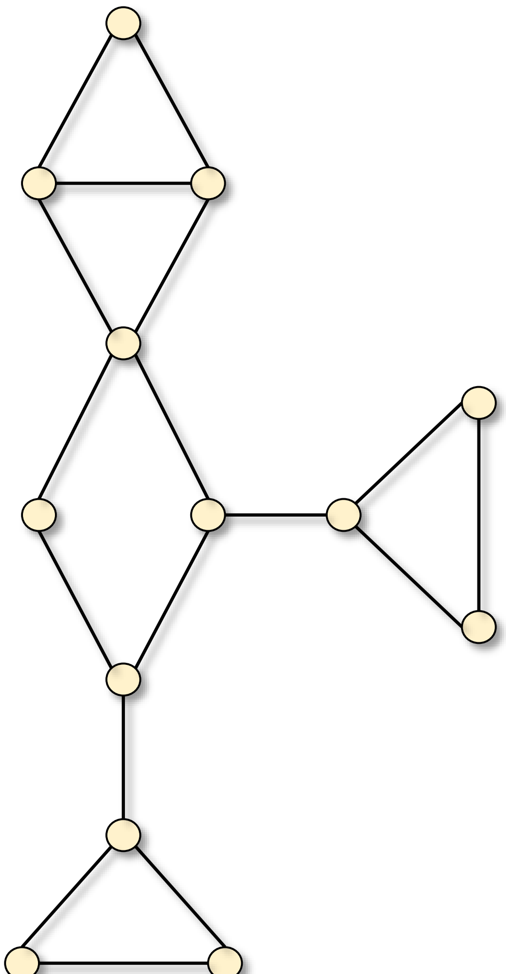
Undirected: Connected components



By Menger's theorem, two vertices are **2-edge-connected** iff the removal of any bridge leaves them in the same connected component.

Two vertices are **2-edge-connected** if there are **two edge-disjoint** paths between them.

Undirected: Connected components

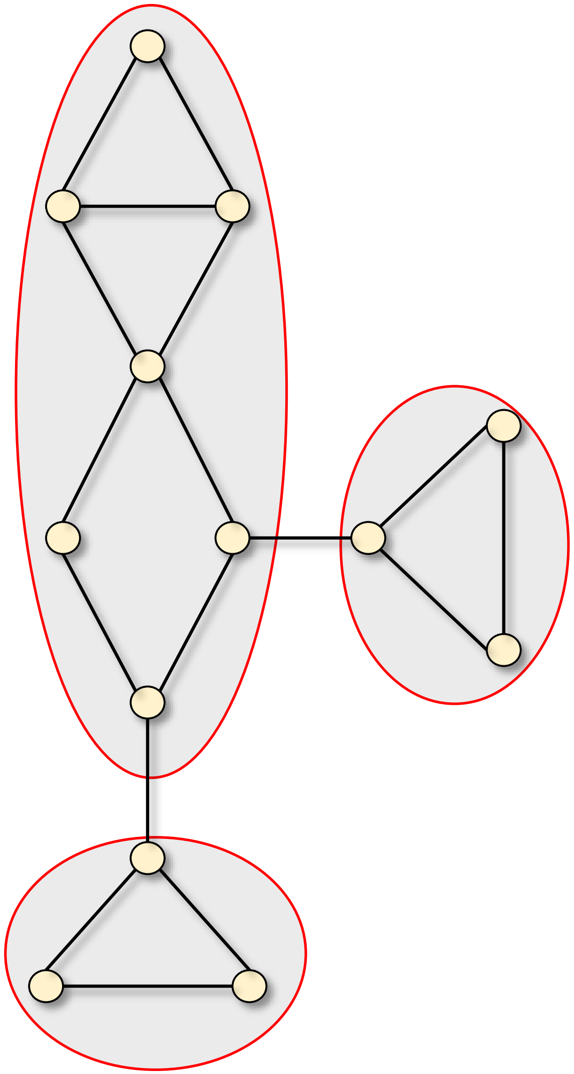


By Menger's theorem, two vertices are **2-edge-connected** iff the removal of any bridge leaves them in the same connected component.

Two vertices are **2-edge-connected** if there are **two edge-disjoint** paths between them.

The **2-edge-connected blocks** of G are its maximal subsets $B \subseteq V$ s.t. u and v are **2-edge-connected** $\forall u, v \in B$

Undirected: Connected components



By Menger's theorem, two vertices are **2-edge-connected** iff the removal of any bridge leaves them in the same connected component.

Two vertices are **2-edge-connected** if there are **two edge-disjoint** paths between them

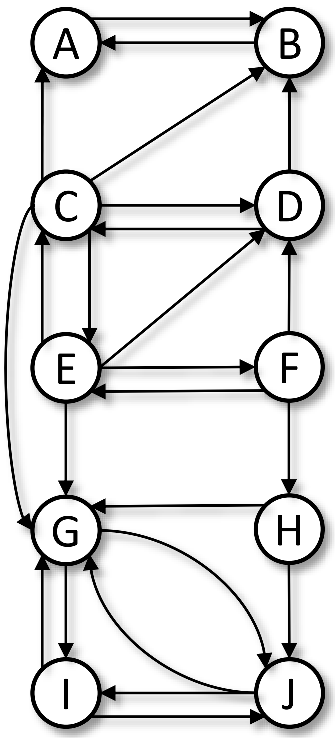
The **2-edge-connected blocks** of G are its maximal subsets $B \subseteq V$ s.t. u and v are **2-edge-connected** $\forall u, v \in B$

- $O(m+n)$ time algorithm [Tarjan 1972]

Outline

- Definitions
  - 2-edge-connectivity in undirected graphs
  - 2-edge-connectivity in directed graphs
  - Problem definition
  - Known algorithm and our result
- High-level idea
- Basic ingredients
  - Dominators
  - Auxiliary components
- Tools
- Conclusion

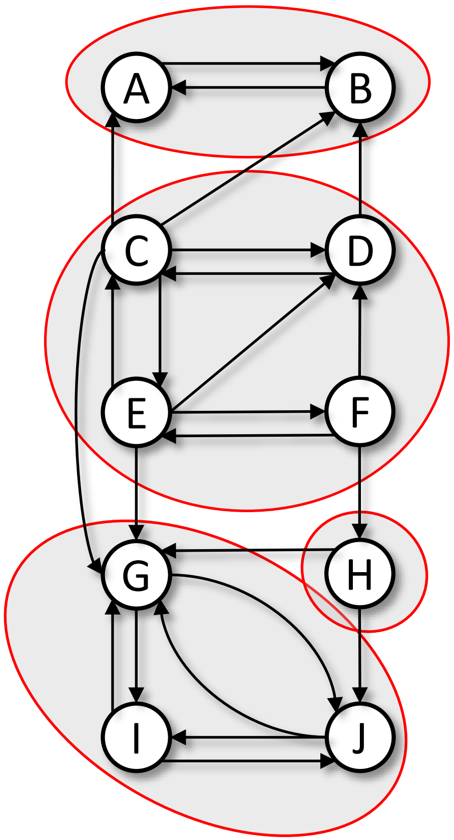
Directed: Strongly connected components



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

- G is **strongly connected** if there is a directed path from each vertex to every other vertex.
- The **strongly connected components** of G are its maximal strongly connected subgraphs.

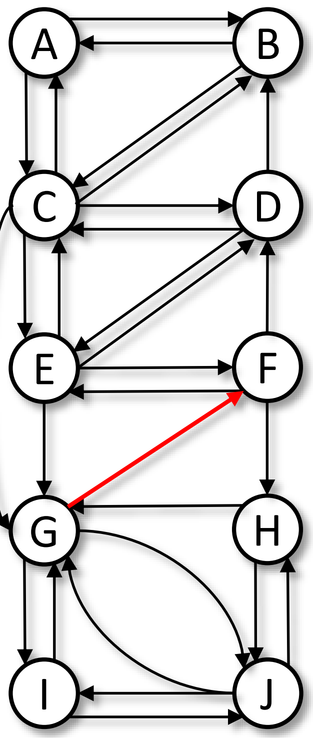
Directed: Strongly connected components



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

- G is **strongly connected** if there is a directed path from each vertex to every other vertex.
- The **strongly connected components** of G are its maximal strongly connected **subgraphs**.

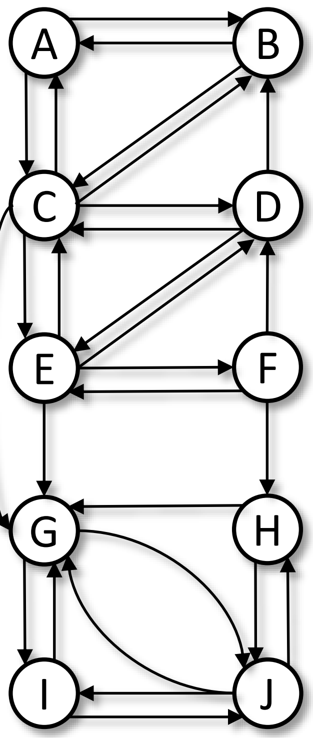
Directed: 2-edge-connectivity



Let $G = (V, E)$ be a **strongly connected directed** graph.

- An edge $e \in E$ is a **strong bridge** if its removal increases the strongly connected components of G.

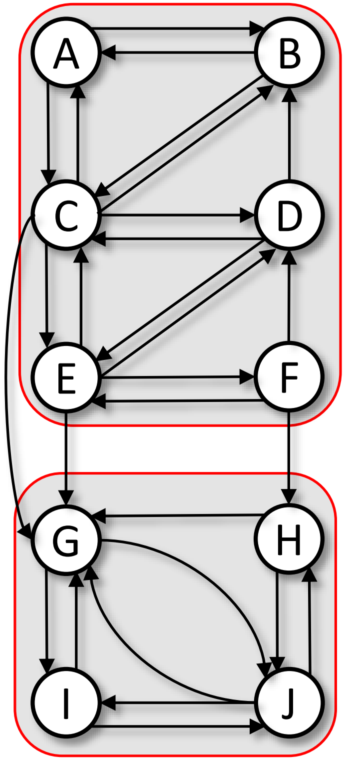
Directed: 2-edge-connectivity



Let $G = (V, E)$ be a **strongly connected directed** graph.

- An edge $e \in E$ is a **strong bridge** if its removal increases the strongly connected components of G.

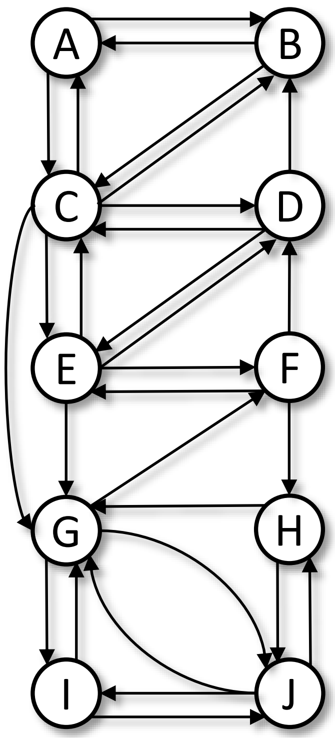
Directed: 2-edge-connectivity



Let $G = (V, E)$ be a **strongly connected directed** graph.

- An edge $e \in E$ is a **strong bridge** if its removal increases the strongly connected components of G.

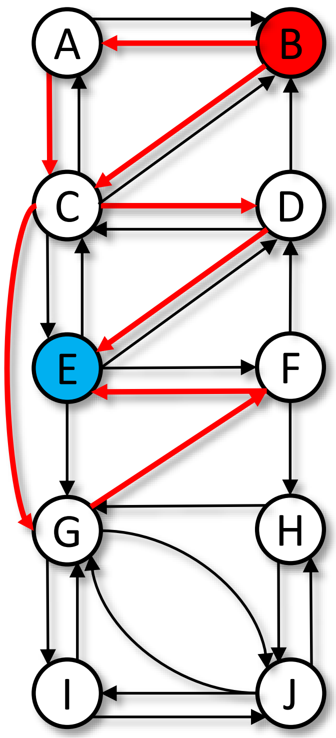
Directed: 2-edge-connectivity



By Menger's Theorem, vertices u and v are **2-edge connected** if and only if the removal of any strong bridge leaves them in same strongly connected component.

Vertices u and v are **2-edge connected** if there are two edge-disjoint paths from u to v and two edge-disjoint paths from v to u.

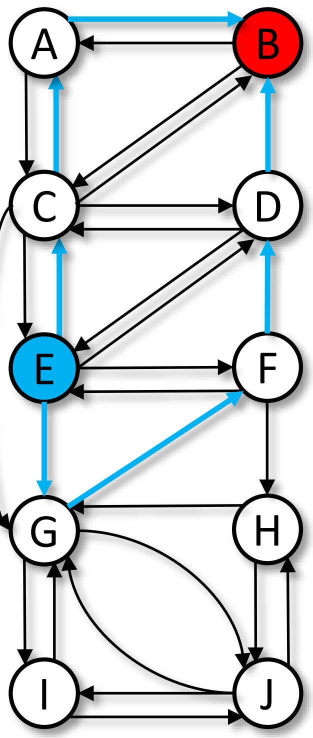
Directed: 2-edge-connectivity



By Menger's Theorem, vertices u and v are **2-edge connected** if and only if the removal of any strong bridge leaves them in same strongly connected component.

Vertices u and v are **2-edge connected** if there are two edge-disjoint paths from u to v and two edge-disjoint paths from v to u.

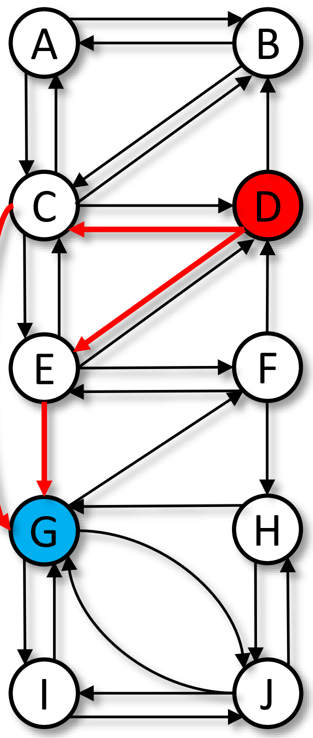
Directed: 2-edge-connectivity



By Menger's Theorem, vertices u and v are **2-edge connected** if and only if the removal of any strong bridge leaves them in same strongly connected component.

Vertices u and v are **2-edge connected** if there are two edge-disjoint paths from u to v and two edge-disjoint paths from v to u.

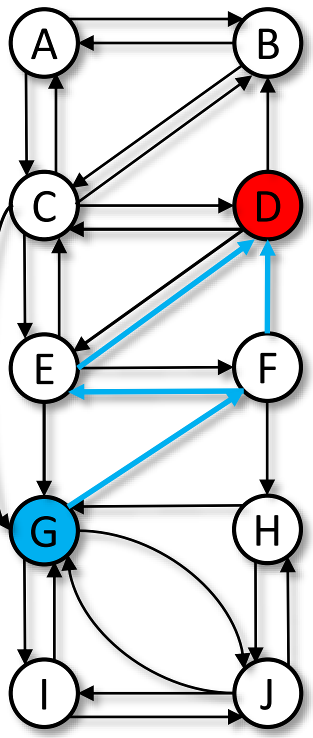
Directed: 2-edge-connectivity



By Menger's Theorem, vertices u and v are **2-edge connected** if and only if the removal of any strong bridge leaves them in same strongly connected component.

Vertices u and v are **2-edge connected** if there are two edge-disjoint paths from u to v and two edge-disjoint paths from v to u.

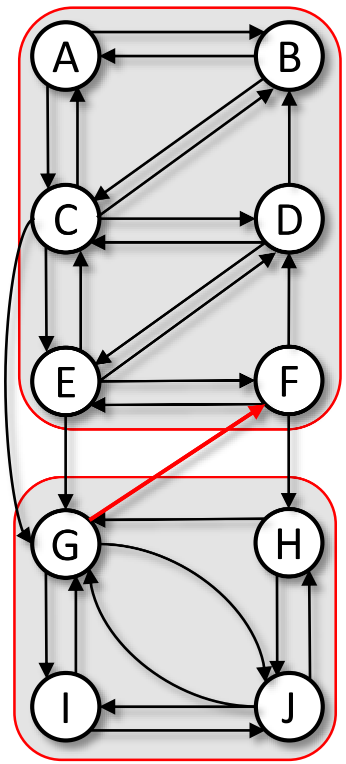
Directed: 2-edge-connectivity



By Menger's Theorem, vertices u and v are **2-edge connected** if and only if the removal of any strong bridge leaves them in same strongly connected component.

Vertices u and v are **2-edge connected** if there are two edge-disjoint paths from u to v and two edge-disjoint paths from v to u.

Directed: 2-edge-connectivity

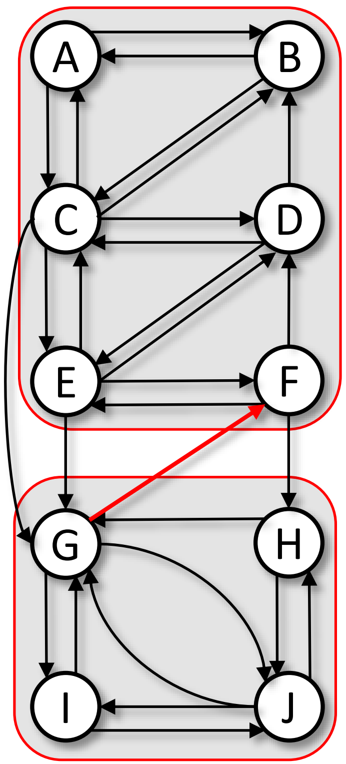


By Menger's Theorem, vertices u and v are **2-edge connected** if and only if the removal of any strong bridge leaves them in same strongly connected component.

Vertices u and v are **2-edge connected** if there are two edge-disjoint paths from u to v and two edge-disjoint paths from v to u.

A **2-edge-connected block** of G is a maximal subset $B \subseteq V$ s. t. u and v are **2-edge connected** for all $u, v \in B$.

Directed: 2-edge-connectivity



By Menger's Theorem, vertices u and v are **2-edge connected** if and only if the removal of any strong bridge leaves them in same strongly connected component.

Vertices u and v are **2-edge connected** if there are two edge-disjoint paths from u to v and two edge-disjoint paths from v to u.

A **2-edge-connected block** of G is a maximal subset $B \subseteq V$ s. t. u and v are **2-edge connected** for all $u, v \in B$.

- $O(m + n)$ time algorithm
  [Georgiadis, Italiano, Laura, P. 2015]

Outline

- Definitions
  - 2-edge-connectivity in undirected graphs
  - 2-edge-connectivity in directed graphs
  - Problem definition
  - Known algorithm and our result
- High-level idea
- Basic ingredients
  - Dominators
  - Auxiliary components
- Tools
- Conclusion

Problem definition

Graph + data structure

Problem definition

Are u and v 2-edge-connected

Graph + data structure

Problem definition

Are u and v 2-edge-connected

Yes/No

Graph + data structure

Problem definition

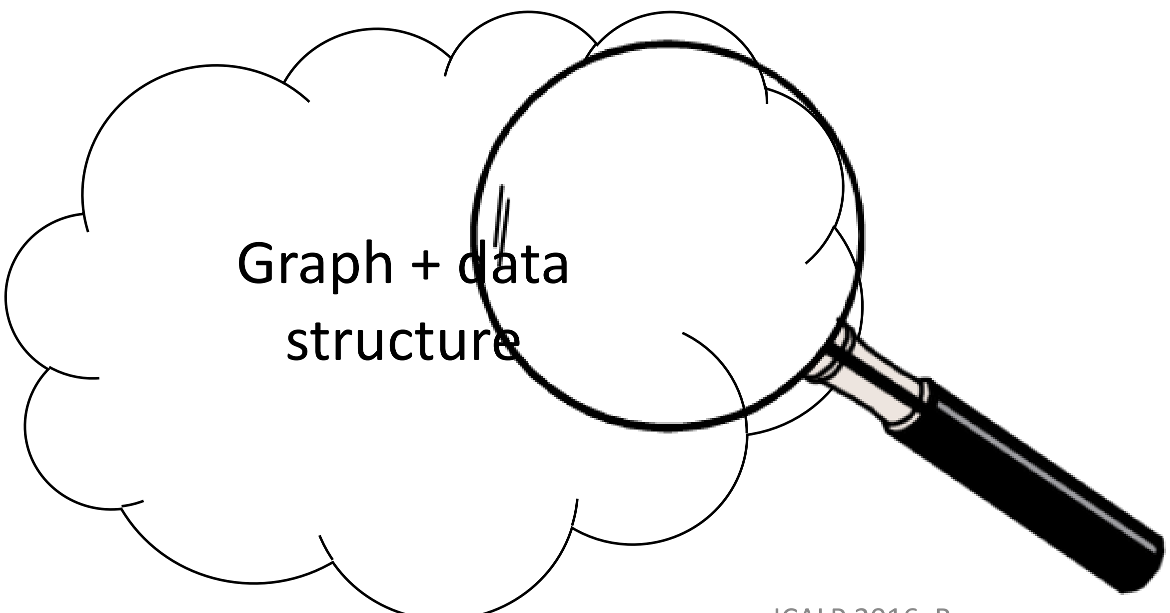
What are the 2-edge-connected blocks in G

Graph + data structure

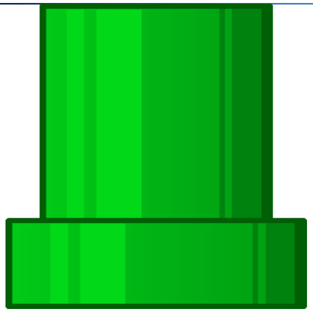
Problem definition

What are the 2-edge-connected blocks in G

$\{a, b, d\}, \{c, e\}, \{f\}$

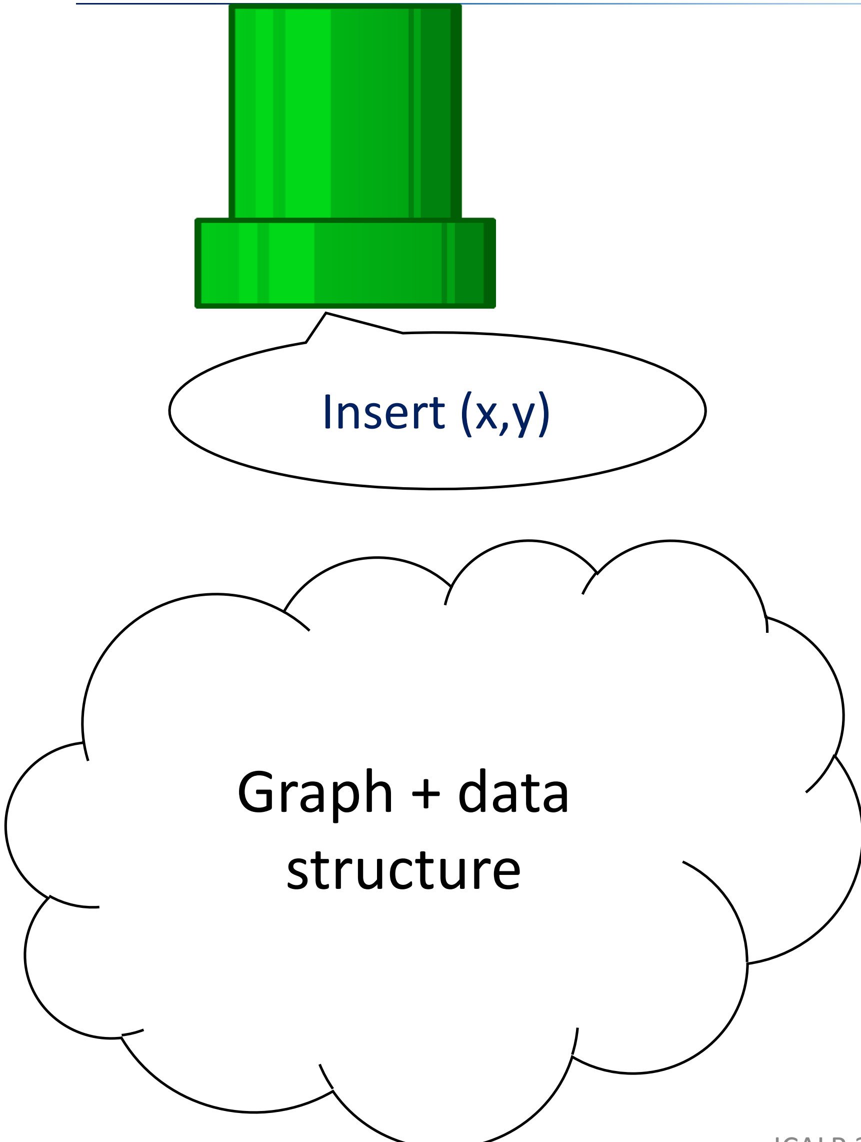


Problem definition

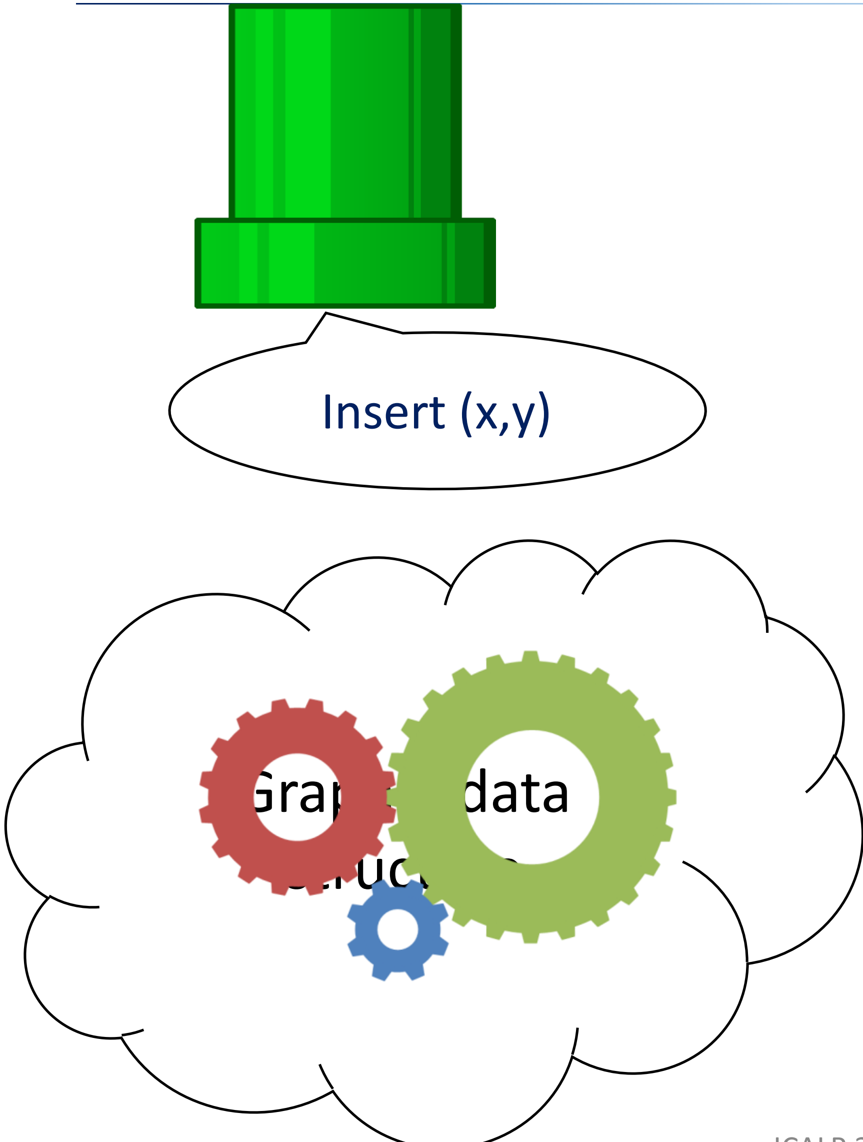


Graph + data structure

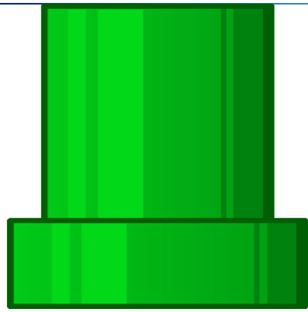
Problem definition



Problem definition



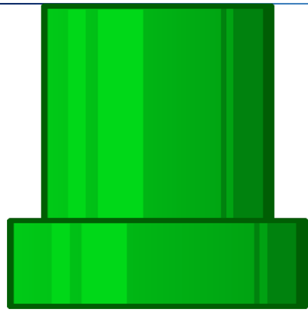
Problem definition



Are u and v 2-edge-connected

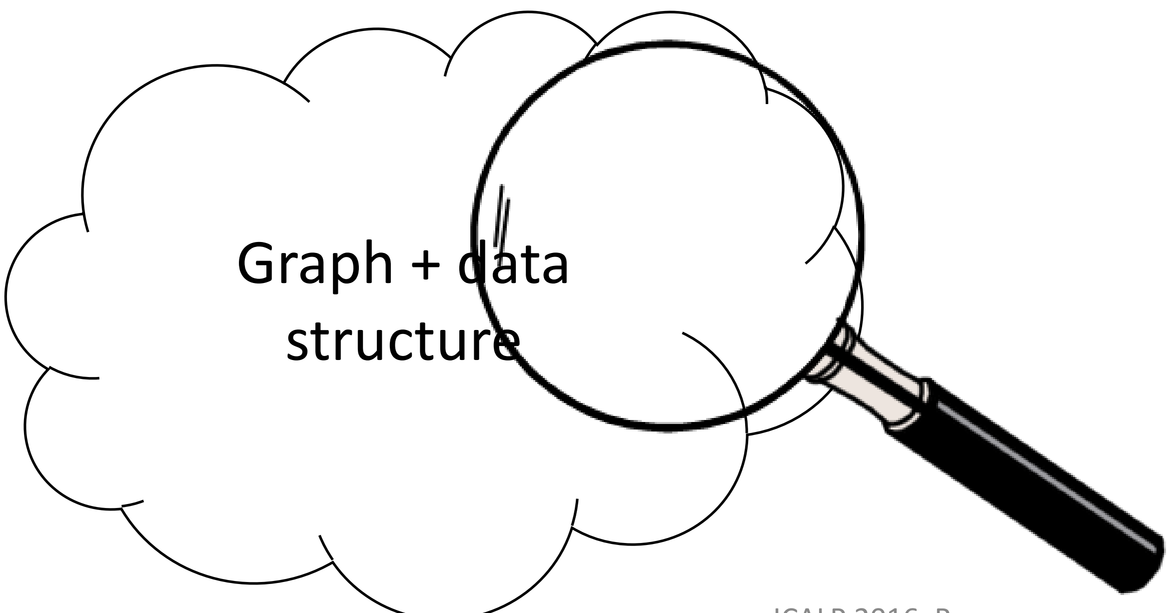
Graph + data structure

Problem definition

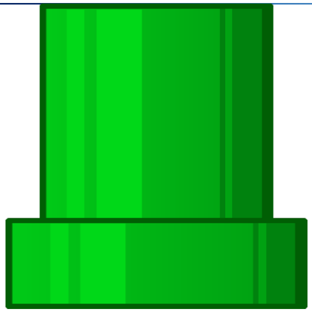


Are u and v 2-edge-connected

Yes/No



Problem definition



Goal:

Update time faster that recomputing

Fast **query time**

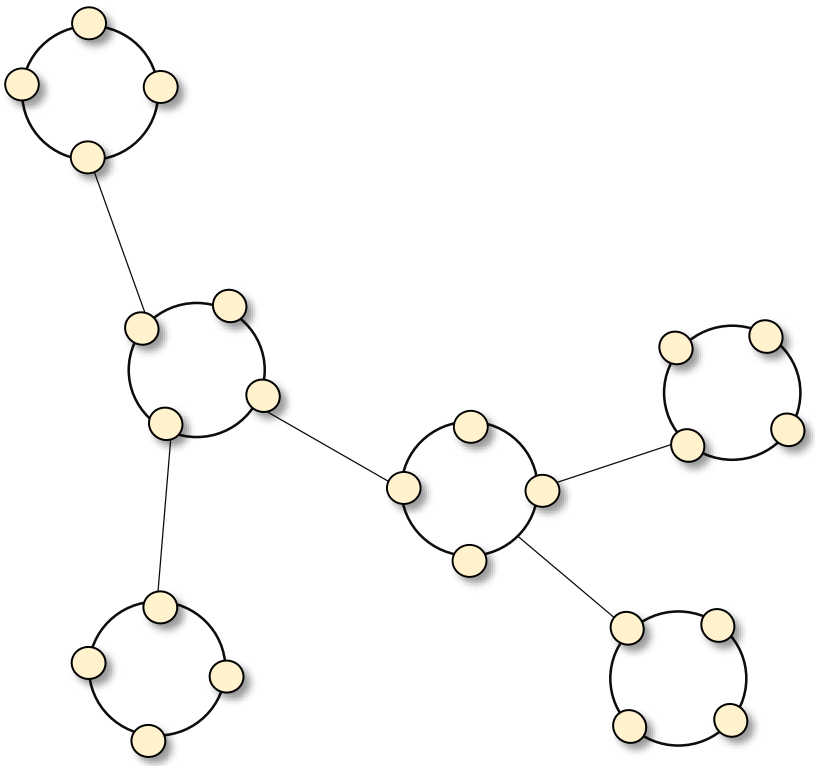
Graph + data structure

Dynamic graph algorithms

Problem	Undirected graphs	Directed graphs
Connectivity/ Transitive closure	Yes	Yes
Connected components/ Strongly connected components	Yes	Yes
APSP	Yes	Yes
DFS tree	Yes	(only on DAGs)
MST	Yes	?
2-edge-connectivity	Yes	?
2-vertex-connectivity	Yes	?

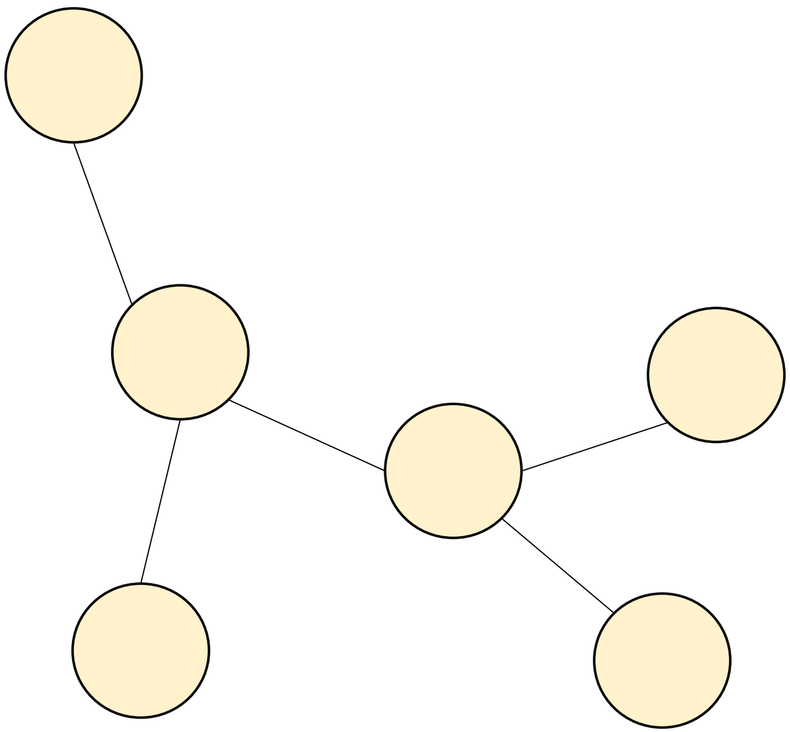
Incremental 2-edge-connectivity Undirected VS Directed

Block tree structure



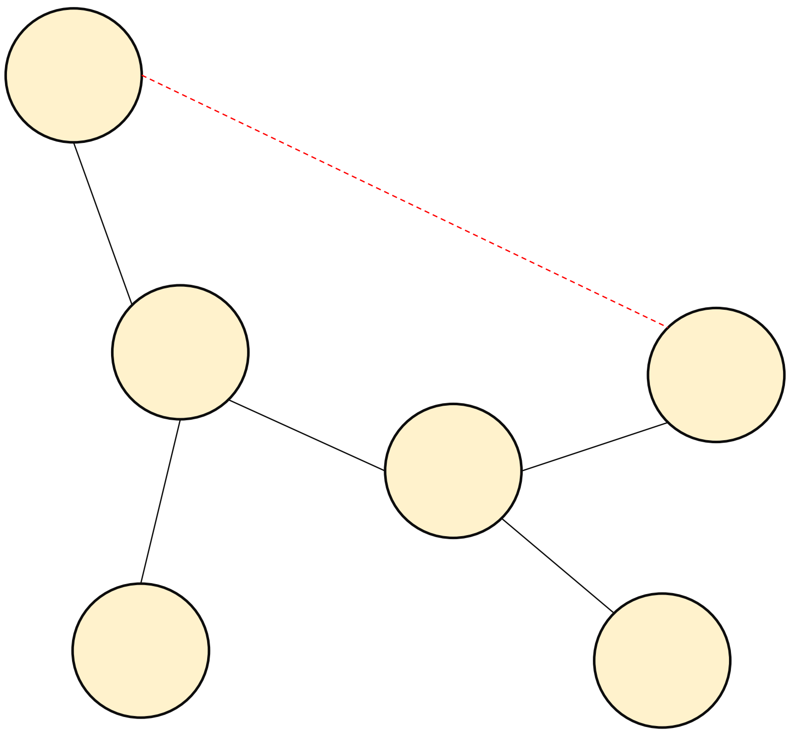
Incremental 2-edge-connectivity Undirected VS Directed

Block tree structure



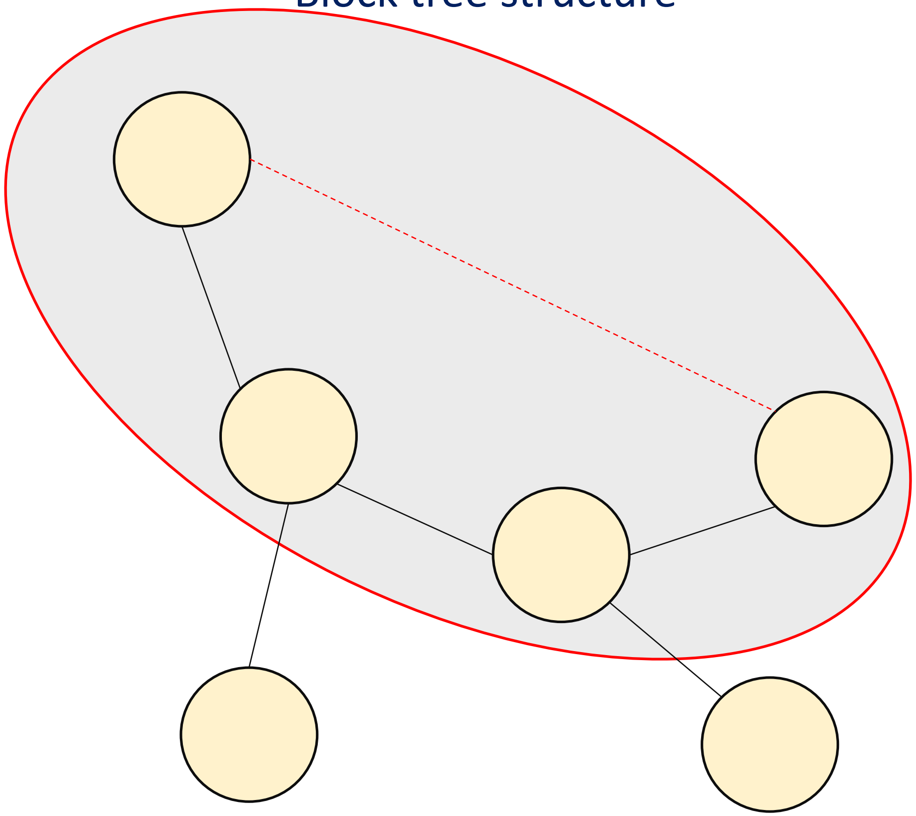
Incremental 2-edge-connectivity Undirected VS Directed

Block tree structure

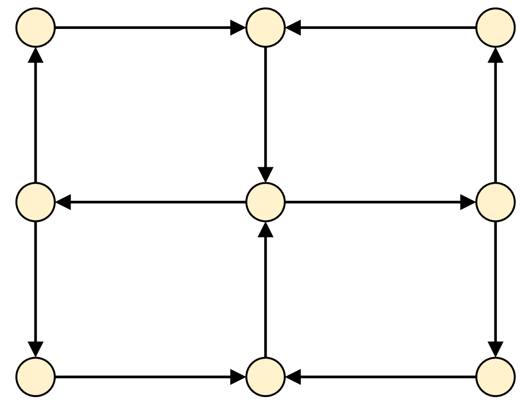


Incremental 2-edge-connectivity Undirected VS Directed

Block tree structure

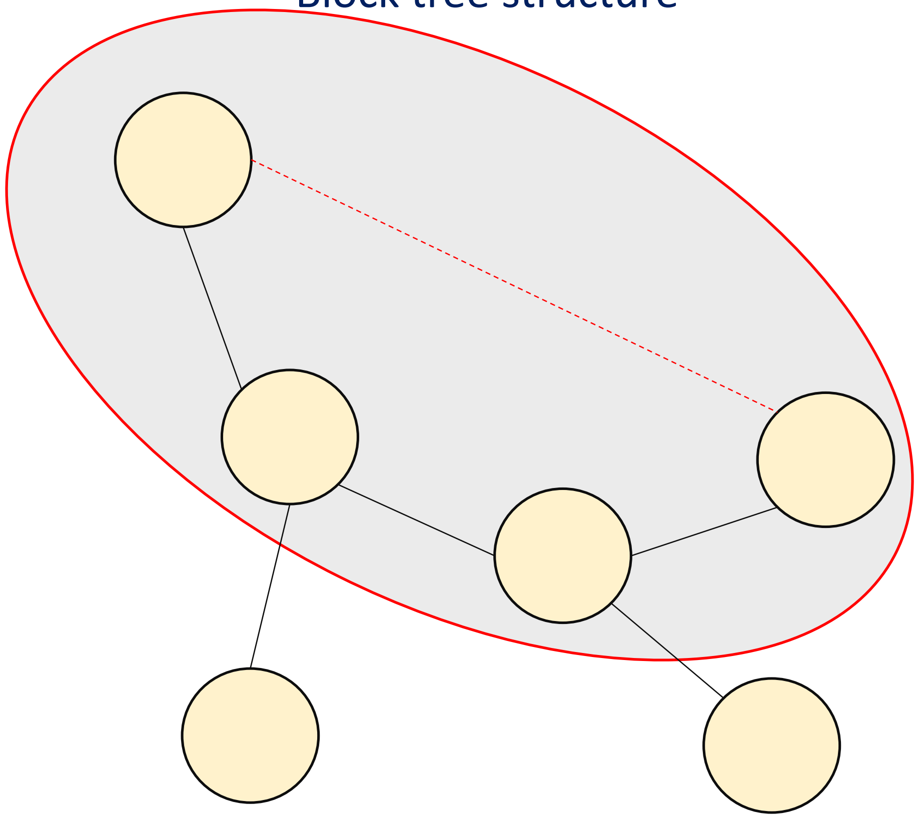


No tree structure is possible

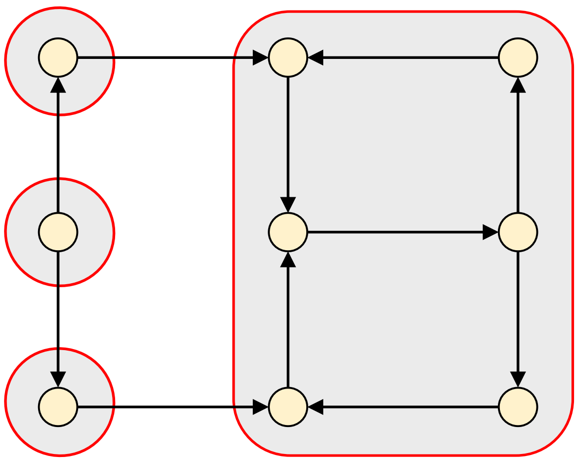


Incremental 2-edge-connectivity Undirected VS Directed

Block tree structure

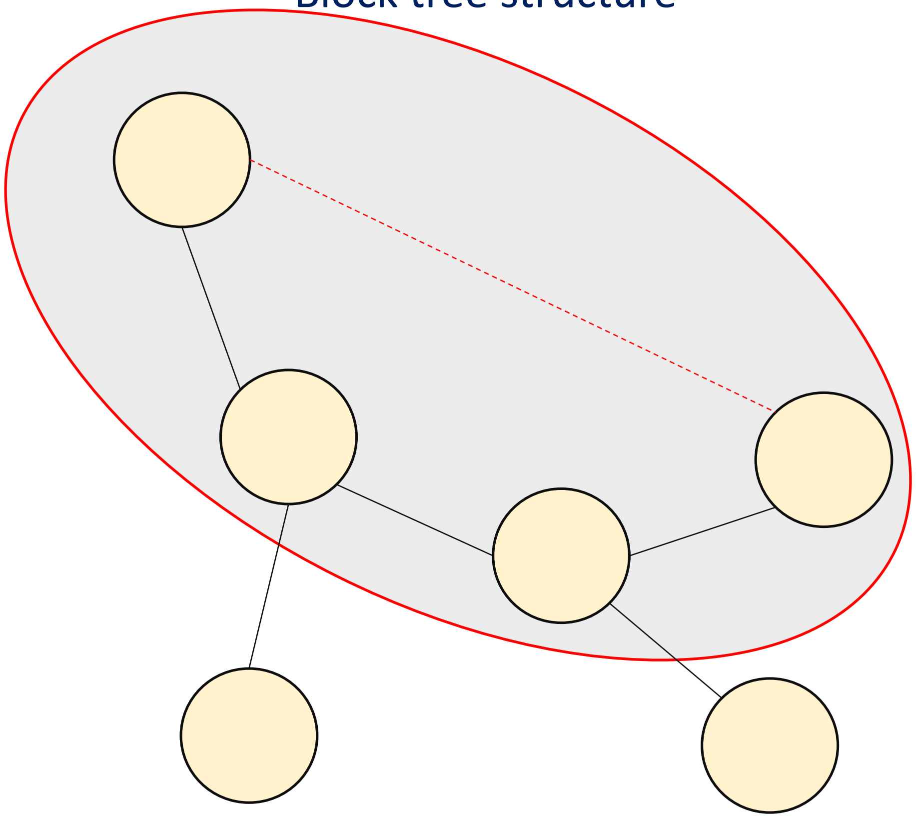


No tree structure is possible

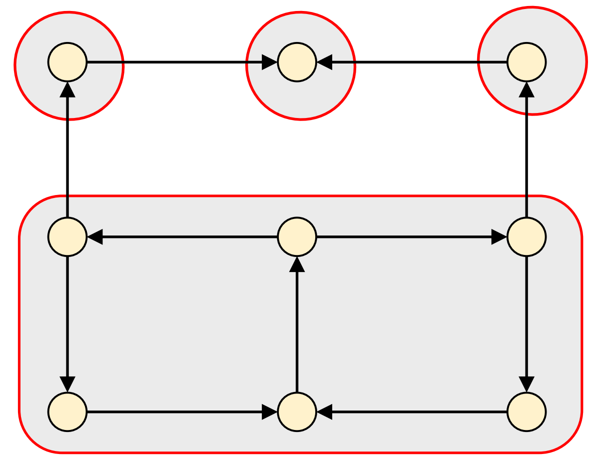


Incremental 2-edge-connectivity
Undirected VS Directed

Block tree structure



No tree structure is possible



Outline

- Definitions
  - 2-edge-connectivity in undirected graphs
  - 2-edge-connectivity in directed graphs
  - Problems definition
  - Known algorithm and our result
- High-level idea
- Basic ingredients
  - Dominators
  - Auxiliary components
- Tools
- Conclusion

Simple-minded solutions

	Update time	Query time
Never update	$O(1)$ per insertion	$O(m+n)$
Always update	$O(m+n)$ per insertion	$O(1)$

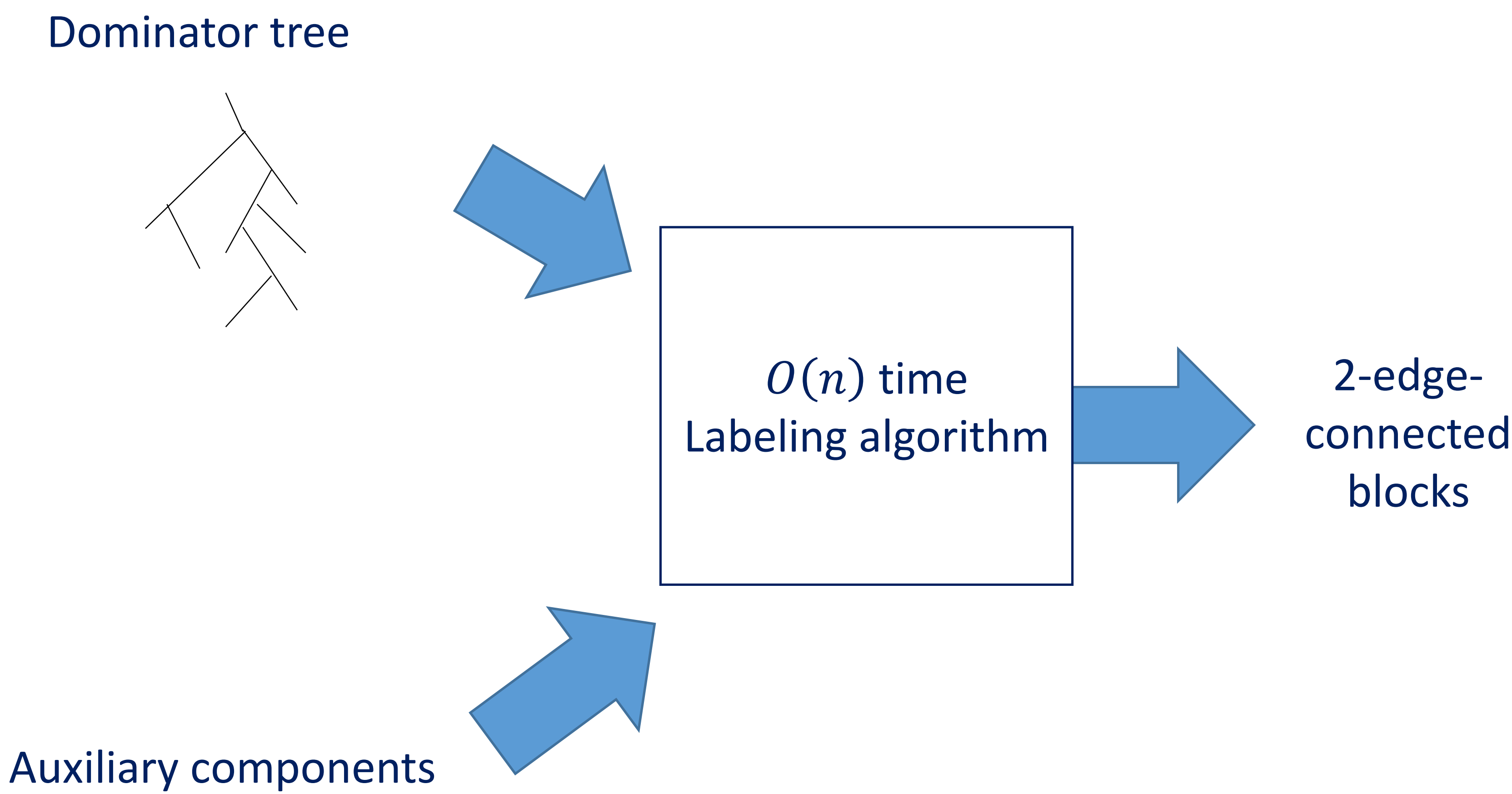
Our algorithm

	Update time	Query time
Never update	$O(1)$ per insertion	$O(m+n)$
Always update	$O(m+n)$ per insertion	$O(1)$
Our algorithm	$O(mn)$ total time	$O(1)$

Outline

- Definitions
  - 2-edge-connectivity in undirected graphs
  - 2-edge-connectivity in directed graphs
  - Problems definition
  - Known algorithm and our result
- High-level idea
- Basic ingredients
  - Dominators
  - Auxiliary components
- Tools
- Conclusion

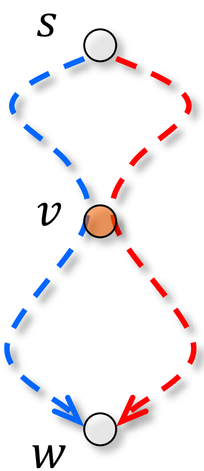
High-level idea



Dominators

Flow graph $G(s) = (V, A, s)$: all vertices are reachable from start vertex s

v dominates w if all paths from s to w contain v

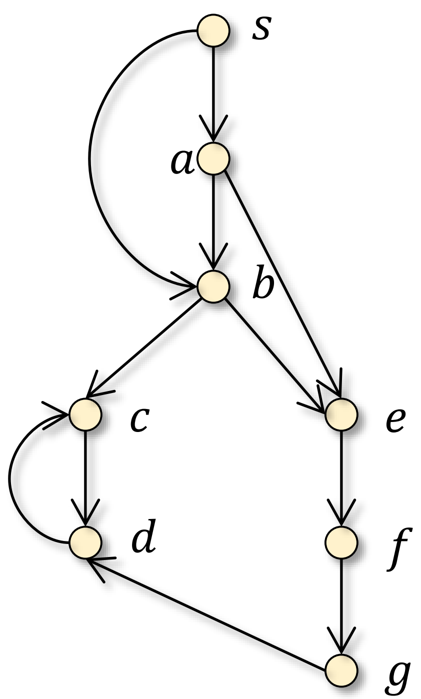


$dom(w) =$ set of vertices that dominate w

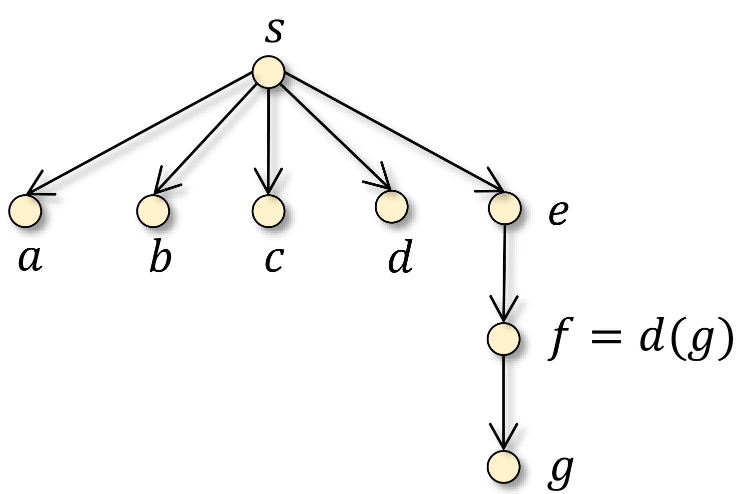
Dominators

v dominates w if all paths from s to w contain v

$G(s)$



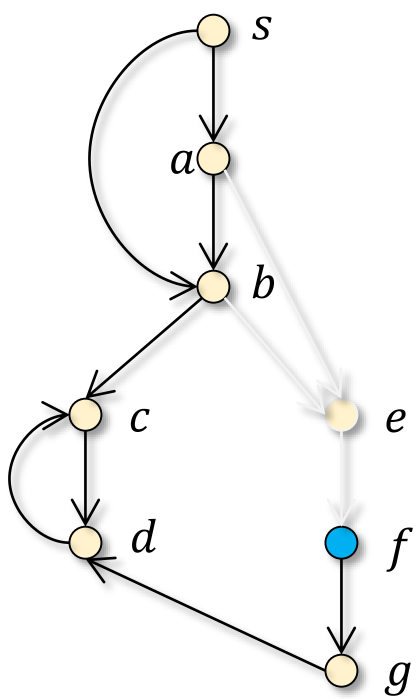
$D(s)$ = dominator tree of $G(s)$



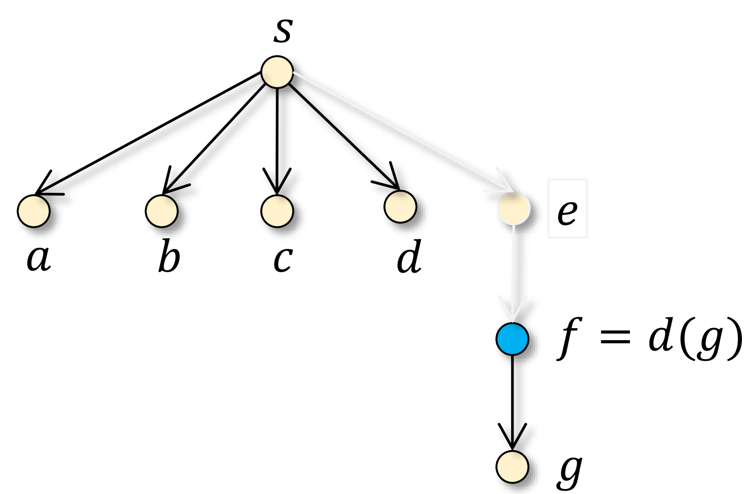
Dominators

v dominates w if all paths from s to w contain v

$G(s)$



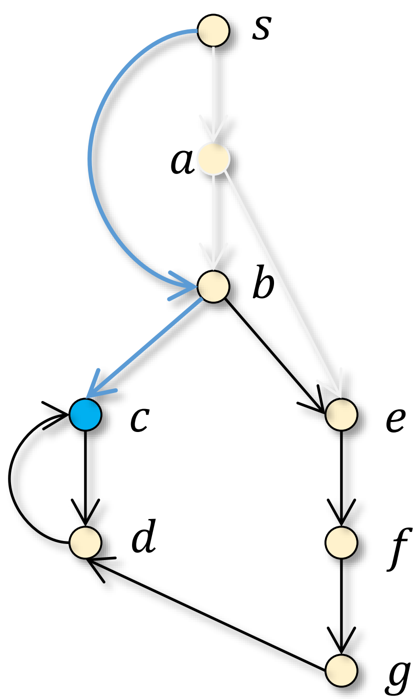
$D(s)$ = dominator tree of $G(s)$



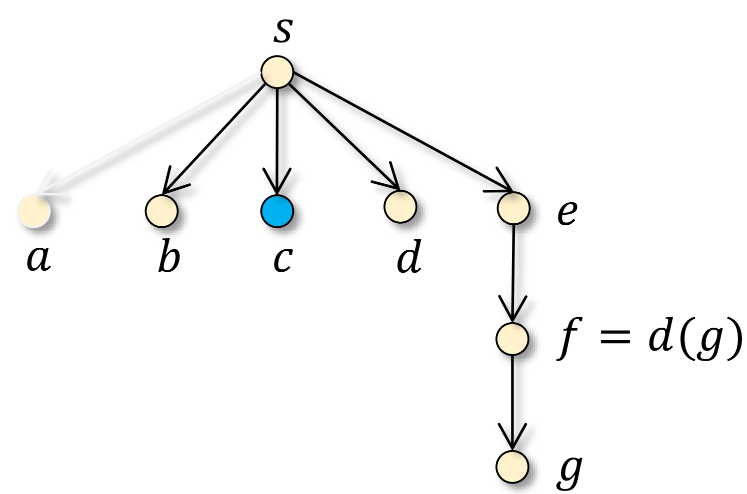
Dominators

v dominates w if all paths from s to w contain v

$G(s)$



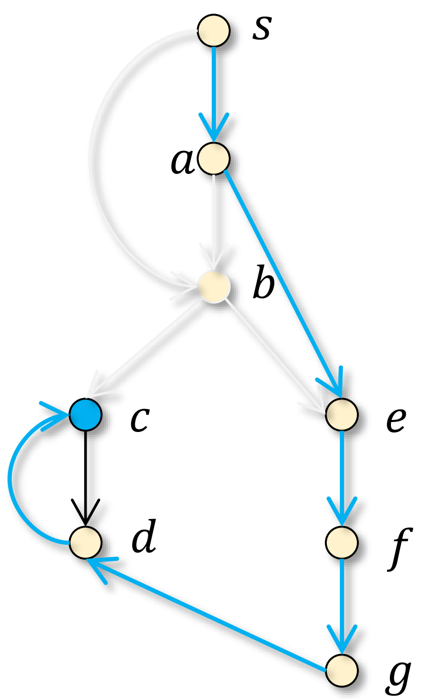
$D(s)$ = dominator tree of $G(s)$



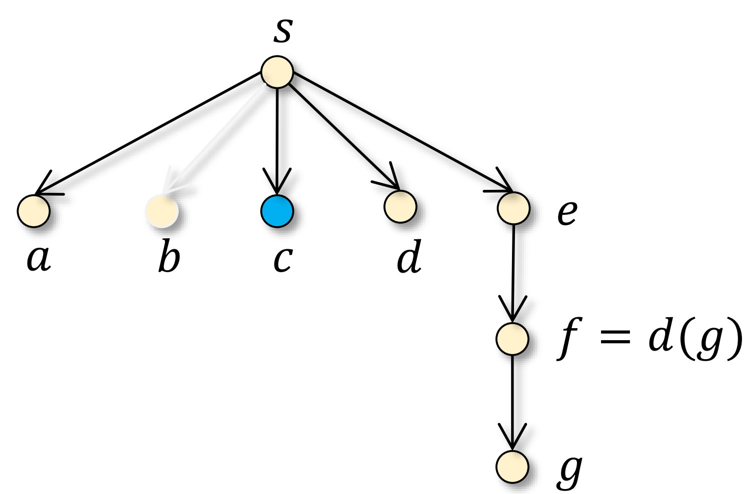
Dominators

v dominates w if all paths from s to w contain v

$G(s)$



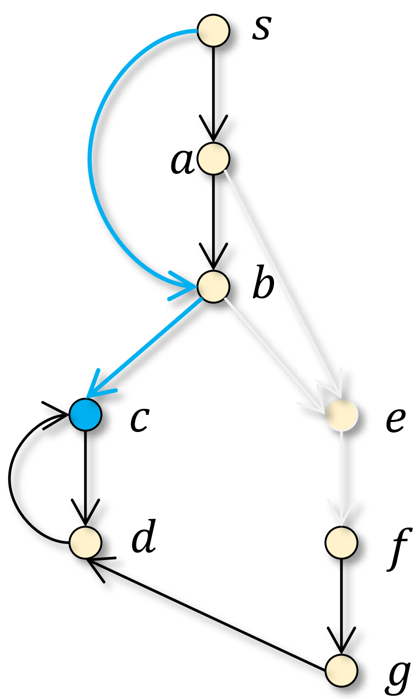
$D(s)$ = dominator tree of $G(s)$



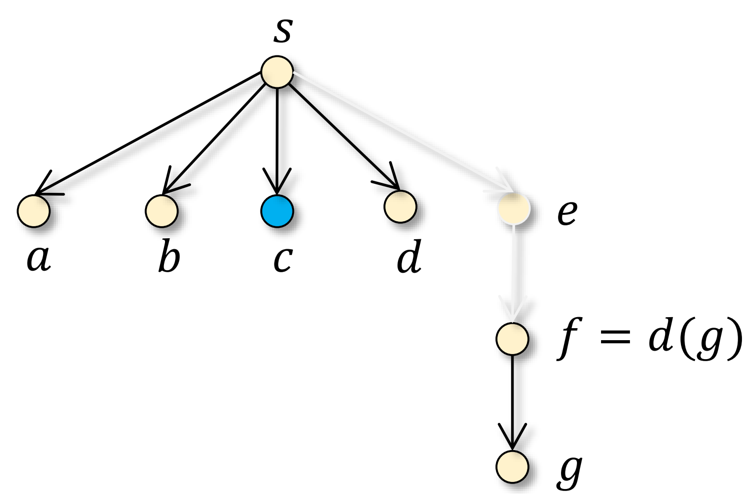
Dominators

v dominates w if all paths from s to w contain v

$G(s)$



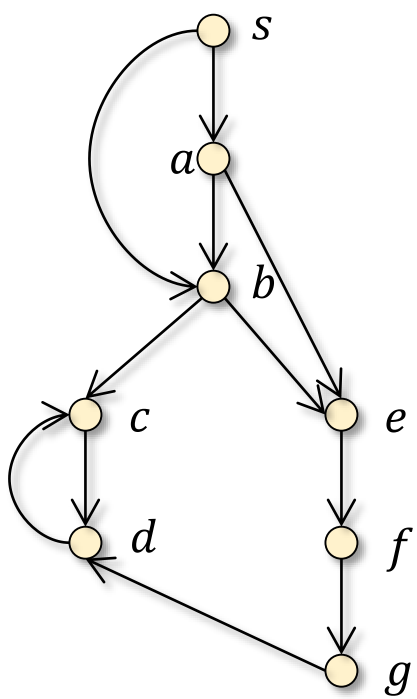
$D(s) \; = \;$ dominator tree of $G(s)$



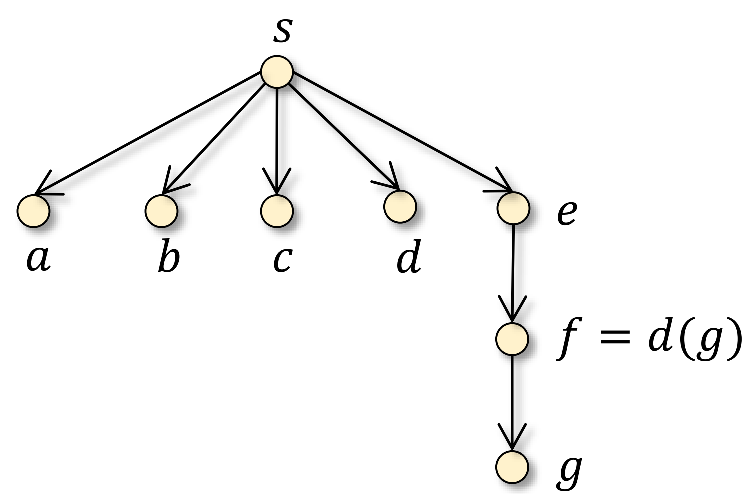
Dominators

v dominates w if all paths from s to w contain v

$G(s)$



$D(s)$ = dominator tree of $G(s)$

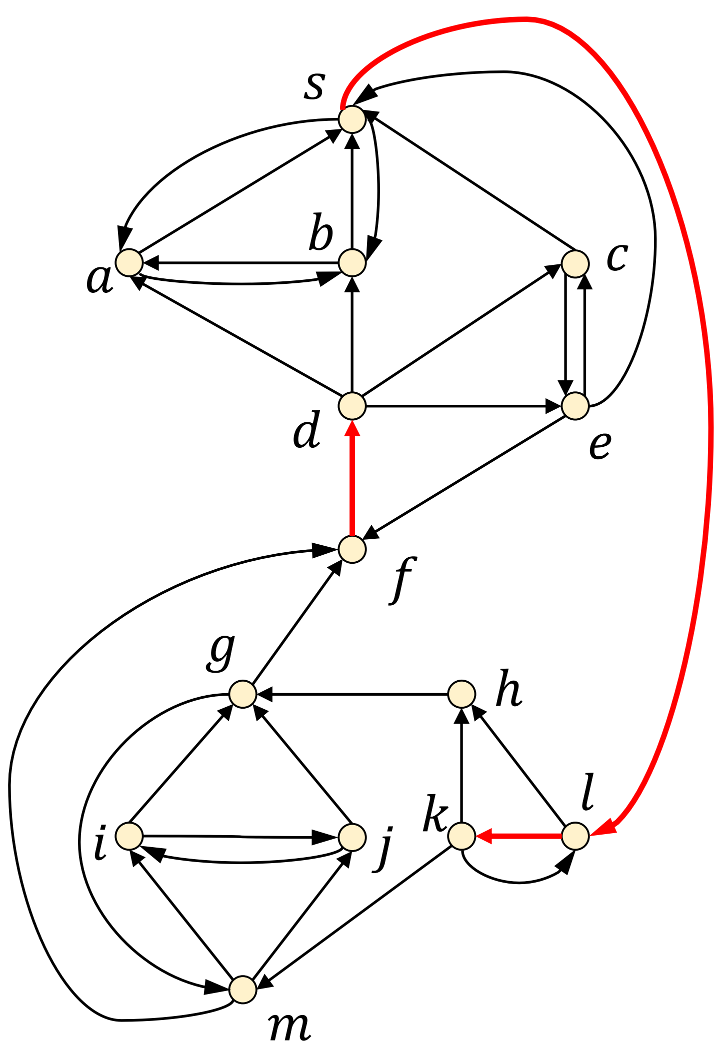


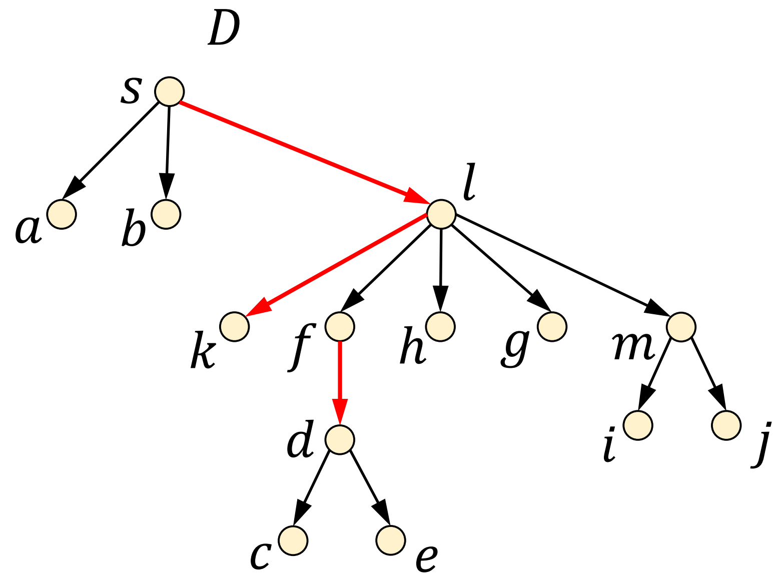
$O(m\alpha(m,n))$-time algorithm: [Lengauer and Tarjan '79]

$O(m+n)$-time algorithms:

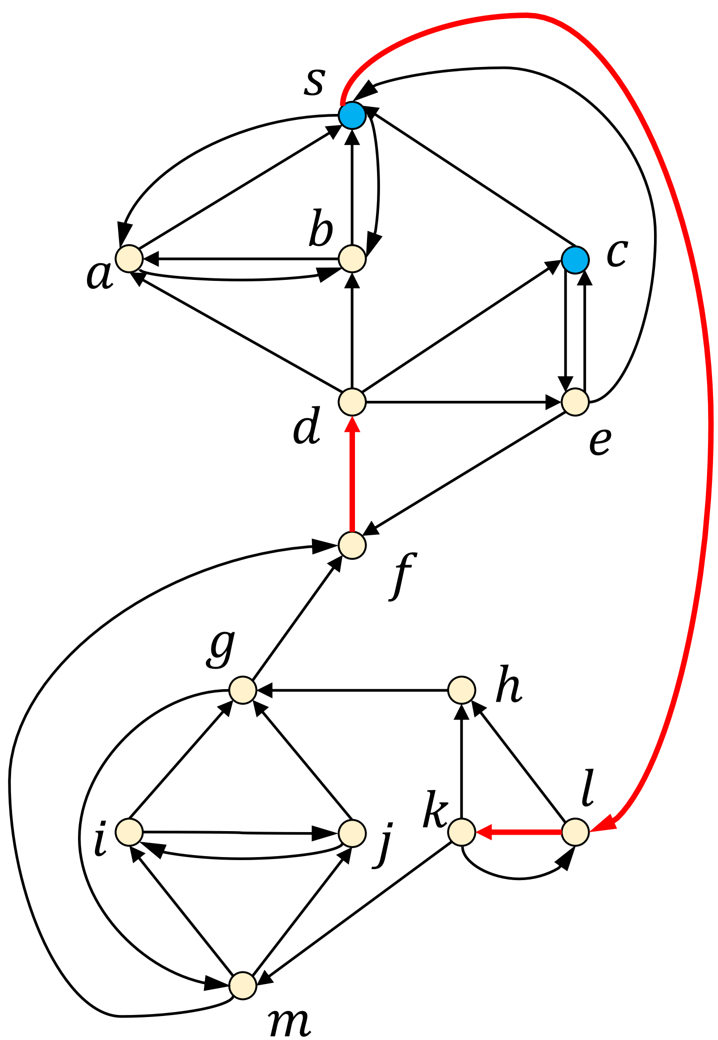
[Alstrup, Harel, Lauridsen, and Thorup '97]

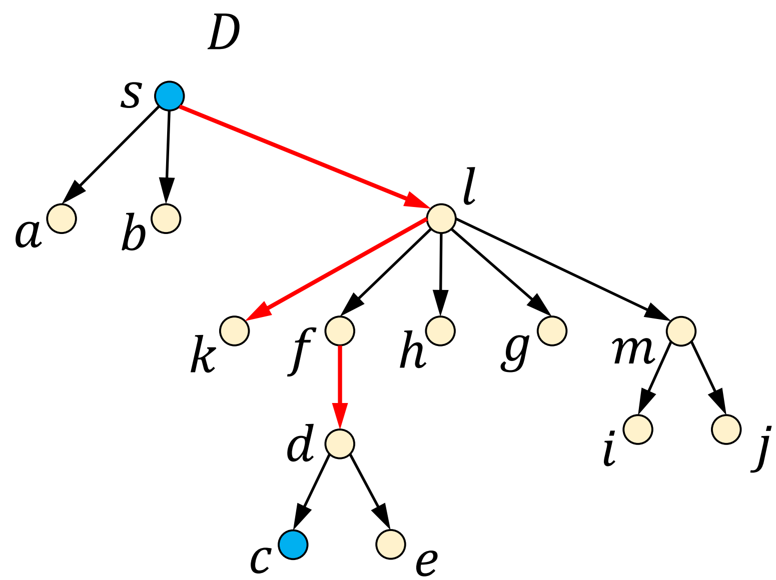
Exploiting dominator tree





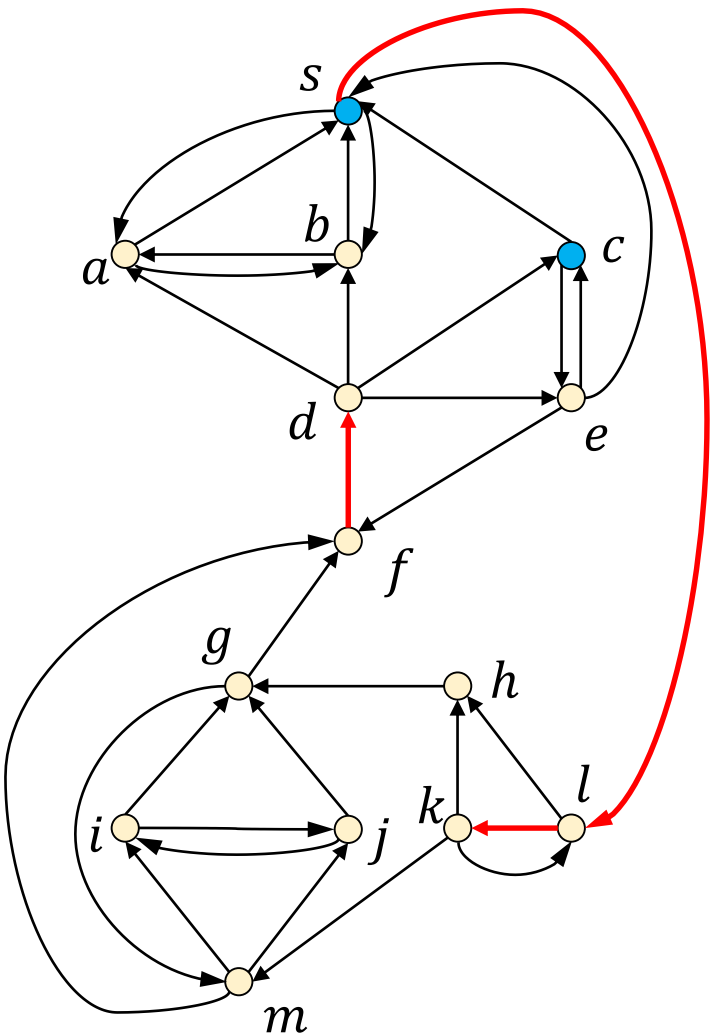
Exploiting dominator tree

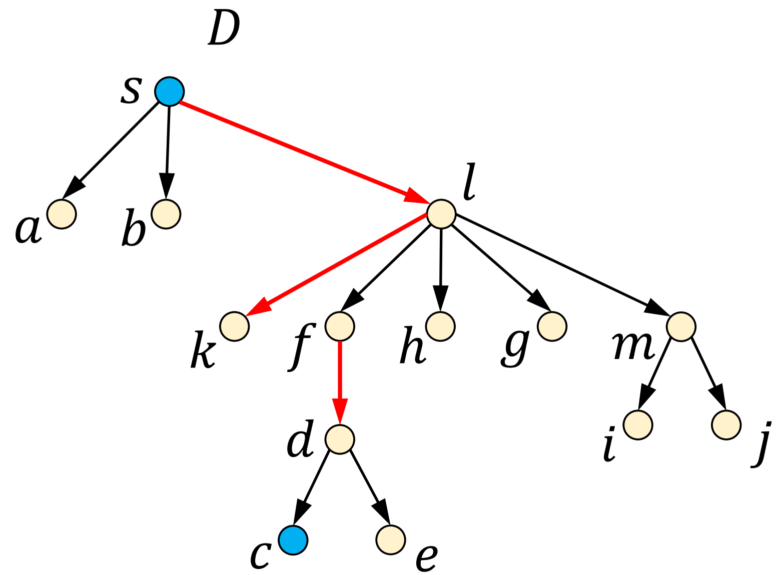




- **All paths** from s to c contain l, f, d

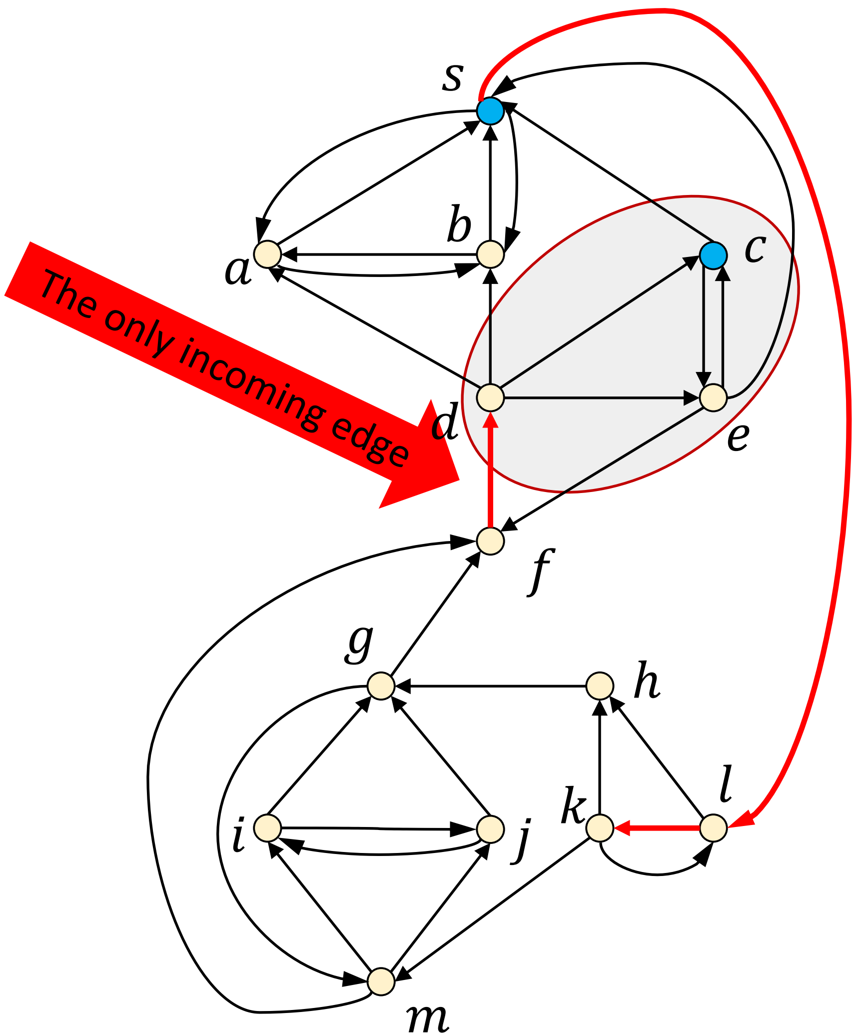
Exploiting dominator tree

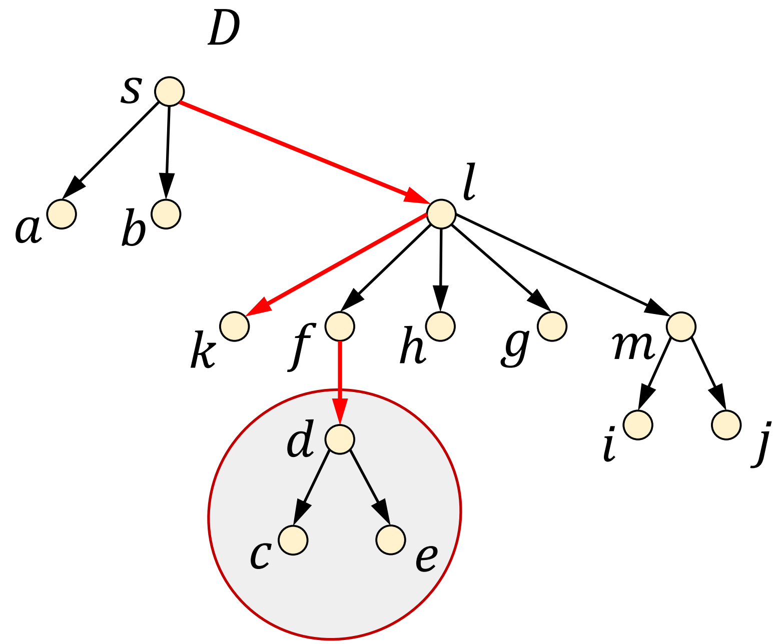




- **All paths** from s to c contain l, f, d
- **All paths** from s to c contain the strong bridges $(s, l), (f, d)$

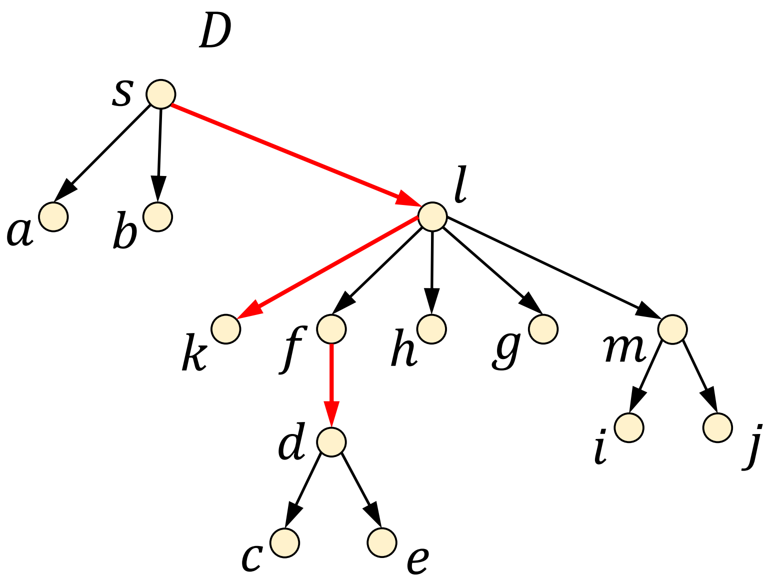
Exploiting dominator tree





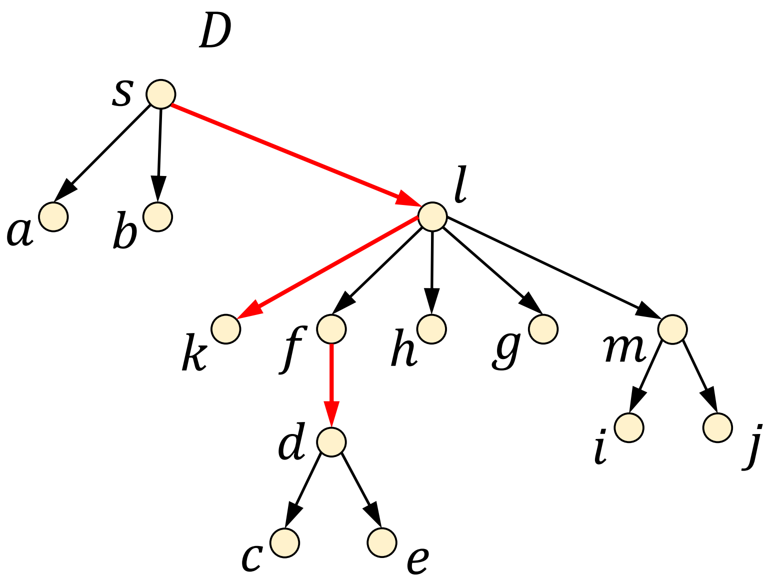
- **All paths** from s to c contain l, f, d
- **All paths** from s to c contain the strong bridges $(s, l), (f, d)$
- A **strong bridge** is the only incoming edge to the vertices of its subtree

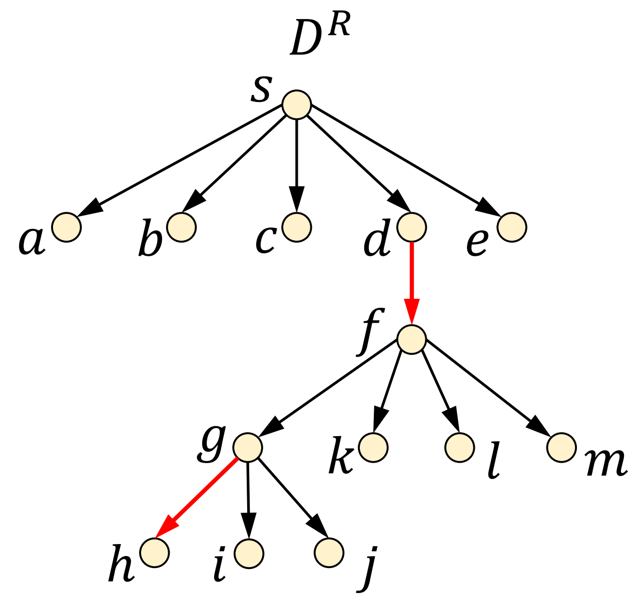
Exploiting dominator tree



The **dominator tree** of the graph provides only partial information.

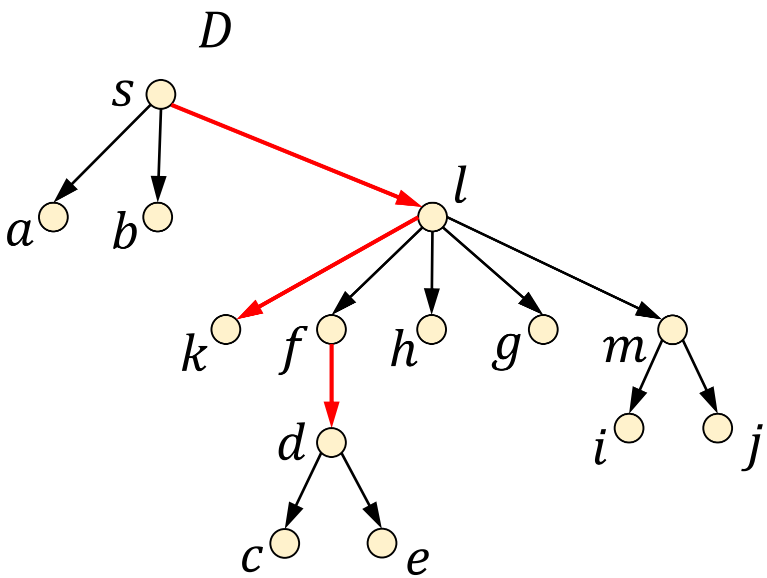
Exploiting dominator tree

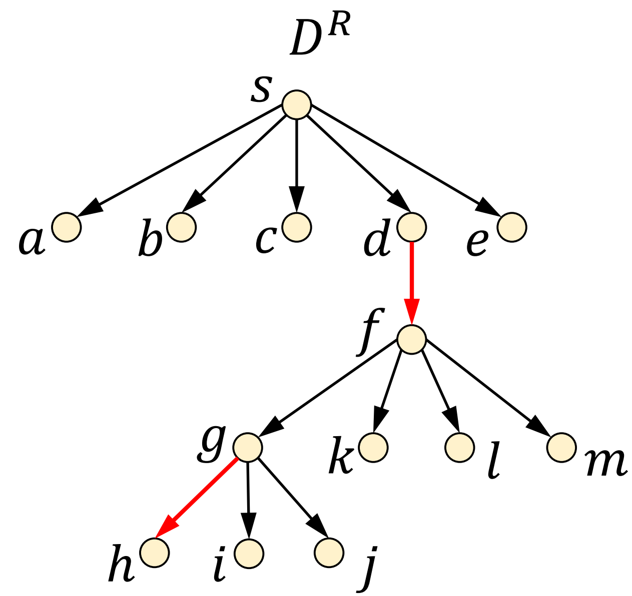




The **dominator tree** of the graph provides only partial information.
The **dominator tree of the reverse graph** provides **other** partial information.

Exploiting dominator tree

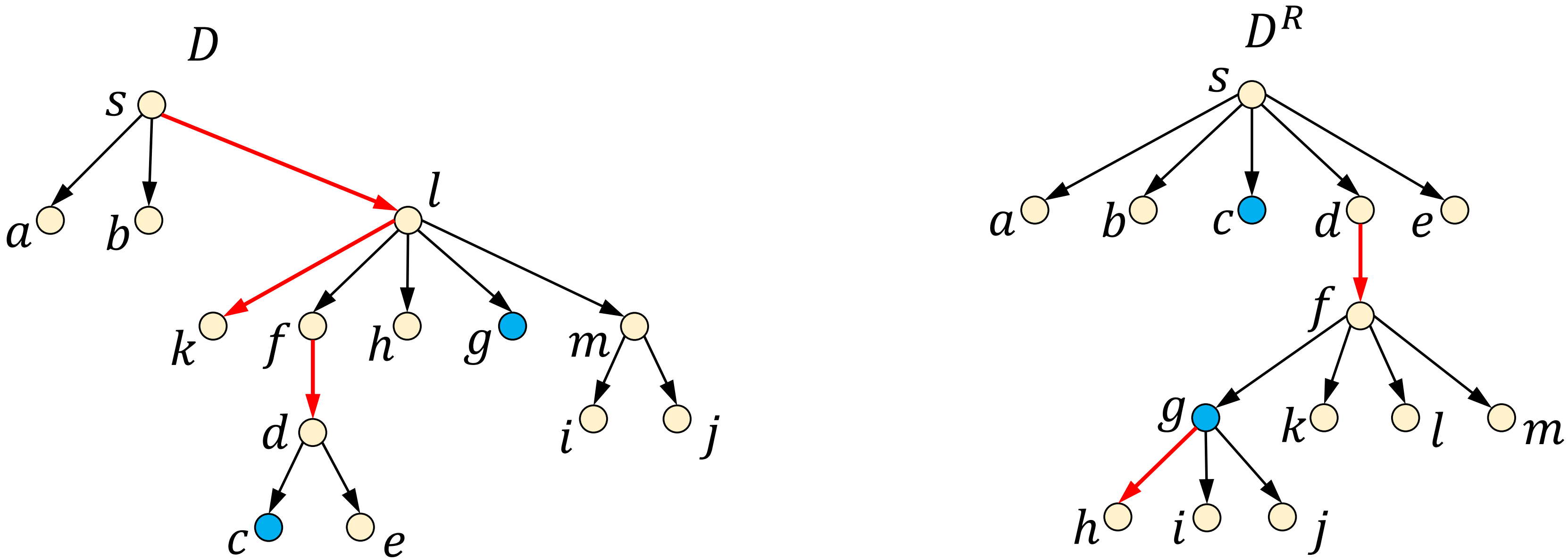




Lemma [Georgiadis, Italiano, P.]: Two vertices u and v are 2-edge-connected **iff**

- Their **nearest bridge** e in D is common and they are not separated in $G \setminus e$
- Their **nearest bridge** e in D^R is common and they are not separated in $G \setminus e$

Exploiting dominator tree

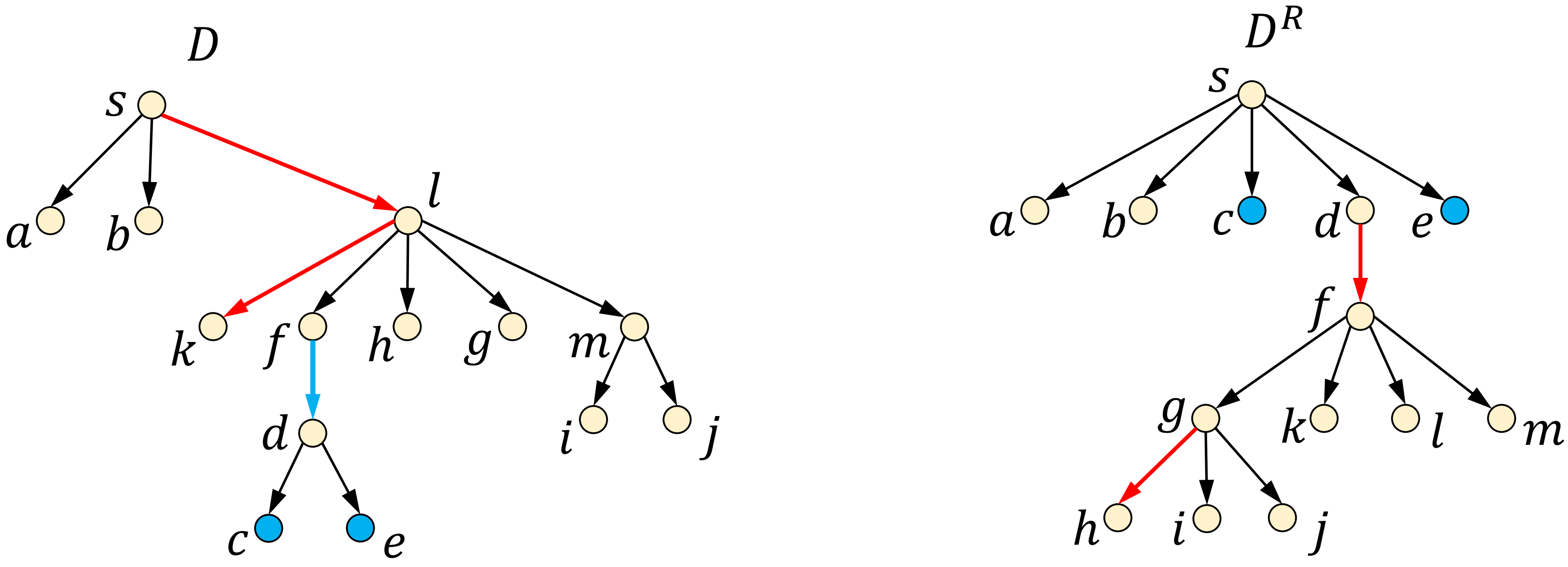


c and g are not 2-edge-connected since they have distinct nearest bridges

Lemma [Georgiadis, Italiano, P.]: Two vertices u and v are 2-edge-connected **iff**

- Their **nearest bridge** e in D is common and they are not separated in $G \setminus e$
- Their **nearest bridge** e in D^R is common and they are not separated in $G \setminus e$

Exploiting dominator tree

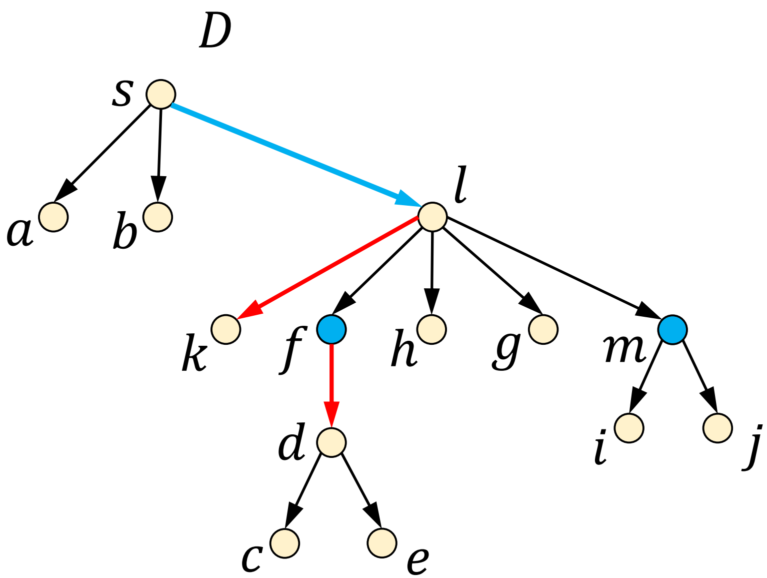


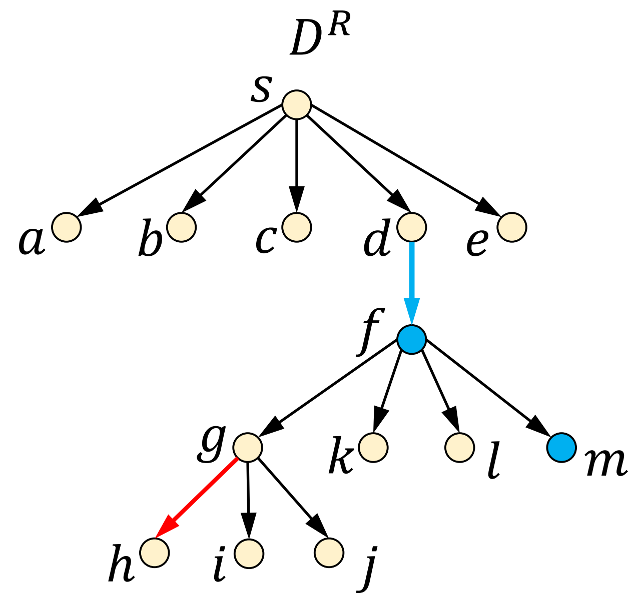
c and e are 2-edge-connected **iff** they are strongly connected in $G \setminus (f, d)$

Lemma [Georgiadis, Italiano, P.]: Two vertices u and v are 2-edge-connected **iff**

- Their **nearest bridge** e in D is common and they are not separated in $G \setminus e$
- Their **nearest bridge** e in D^R is common and they are not separated in $G \setminus e$

Exploiting dominator tree





f and m are 2-edge-connected **iff** they are strongly connected in $G \setminus (s, l)$ and in $G \setminus (f, d)$

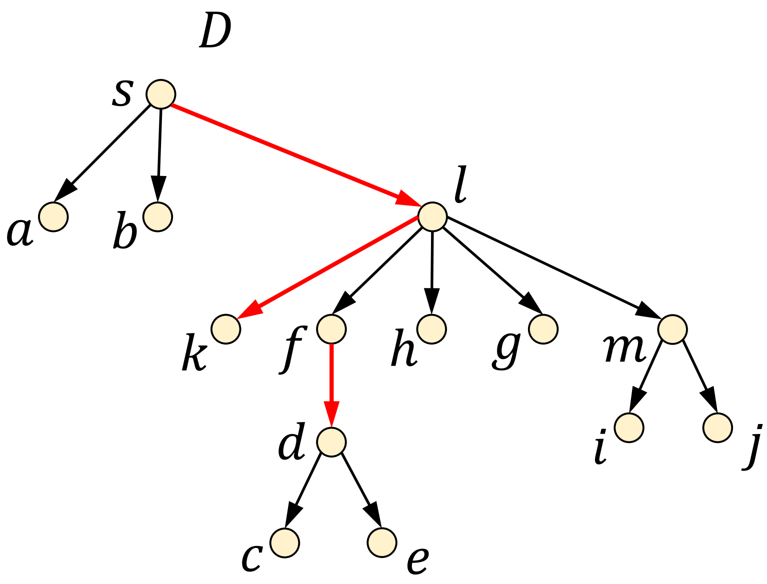
Lemma [Georgiadis, Italiano, P.]: Two vertices u and v are 2-edge-connected **iff**

- Their **nearest bridge** e in D is common and they are not separated in $G \setminus e$
- Their **nearest bridge** e in D^R is common and they are not separated in $G \setminus e$

Outline

- Definitions
  - 2-edge-connectivity in undirected graphs
  - 2-edge-connectivity in directed graphs
  - Problems definition
  - Known algorithm and our result
- High-level idea
- Basic ingredients
  - Dominators
  - Auxiliary components
- Tools
- Conclusion

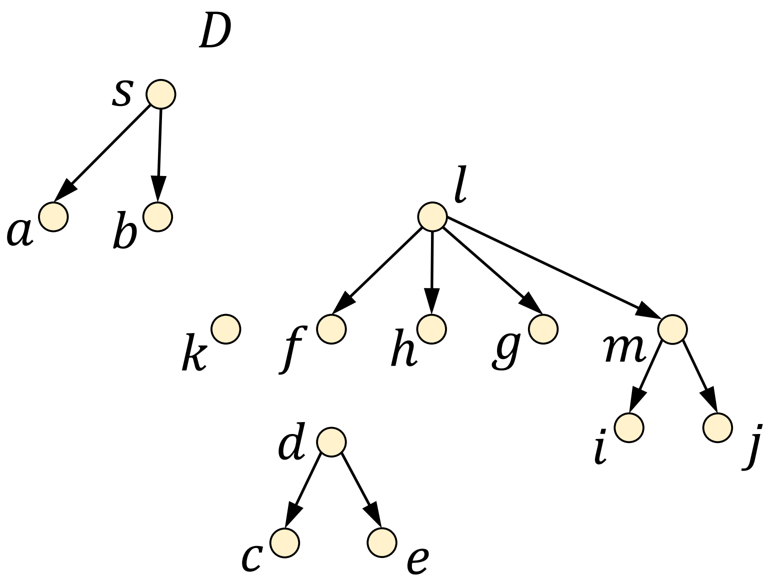
Auxiliary components



Bridge decomposition: The forest obtained by removing the strong bridges from the dominator tree

Lemma: two vertices are 2-edge-connected **only if** they are in the same tree of the bridge decomposition

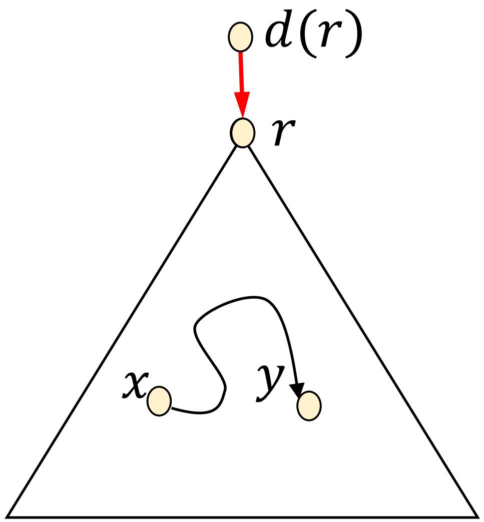
Auxiliary components



Bridge decomposition: The forest obtained by removing the strong bridges from the dominator tree

Lemma: two vertices are 2-edge-connected **only if** they are in the same tree of the bridge decomposition

Auxiliary components

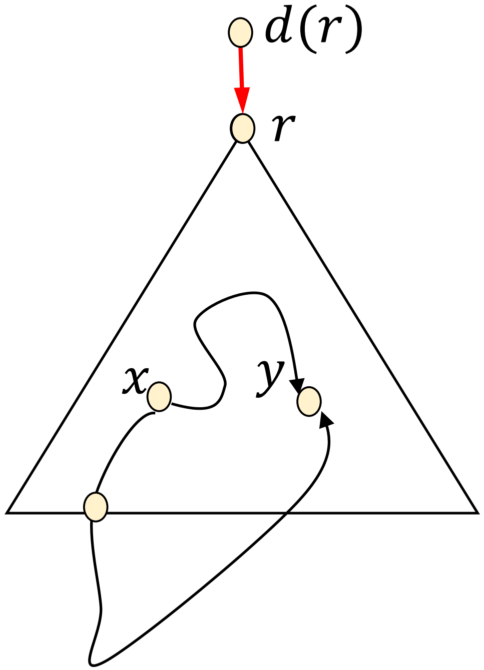


Idea: Encode all the paths that do not use the incoming strong bridge between **vertices in the same tree** of the bridge decomposition with an **auxiliary graph**.

Construction:

- **Keep the paths** using only vertices of the tree
- **For every path** using vertices outside the tree, replace the subpath outside with **a shortcut edge**

Auxiliary components

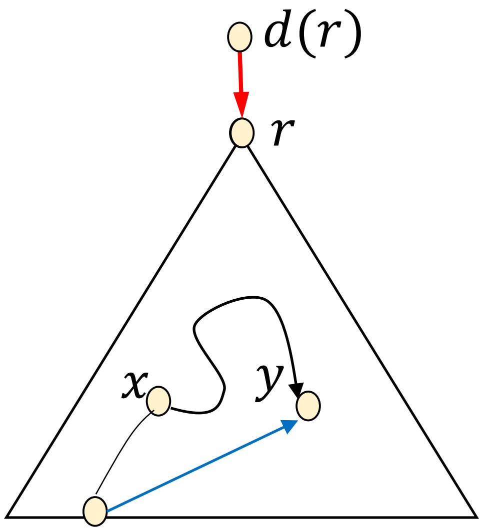


Idea: Encode all the paths that do not use the incoming strong bridge between **vertices in the same tree** of the bridge decomposition with an **auxiliary graph**.

Construction:

- **Keep the paths** using only vertices of the tree
- **For every path** using vertices outside the tree, replace the subpath outside with **a shortcut edge**

Auxiliary components

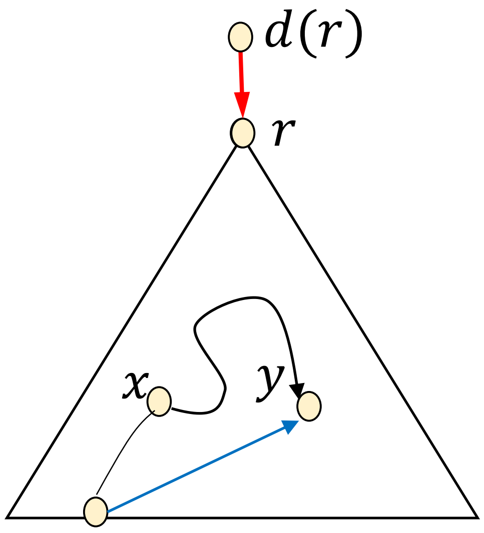


Idea: Encode all the paths that do not use the incoming strong bridge between **vertices in the same tree** of the bridge decomposition with an **auxiliary graph**.

Construction:

- **Keep the paths** using only vertices of the tree
- **For every path** using vertices outside the tree, replace the subpath outside with **a shortcut edge**

Auxiliary components



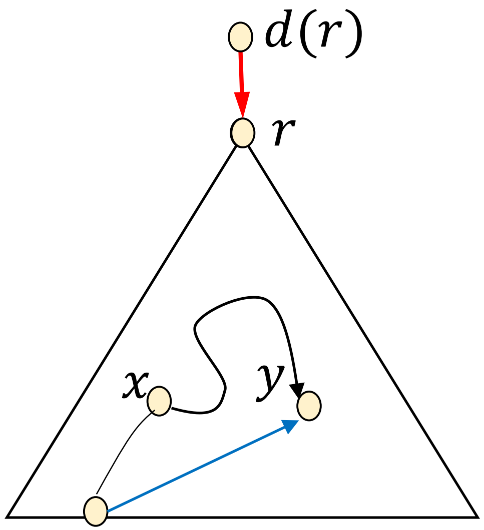
Idea: Encode all the paths that do not use the incoming strong bridge between **vertices in the same tree** of the bridge decomposition with an **auxiliary graph**.

Construction:

- **Keep the paths** using only vertices of the tree
- **For every path** using vertices outside the tree, replace the subpath outside with **a shortcut edge**

Lemma: Two vertices in the same tree (rooted at r) are **disconnected** by $(d(r), r)$ **iff** they **are not** strongly connected in the auxiliary graph of their tree (in the same auxiliary component).

Auxiliary components



Idea: Encode all the paths that do not use the incoming strong bridge between **vertices in the same tree** of the bridge decomposition with an **auxiliary graph**.

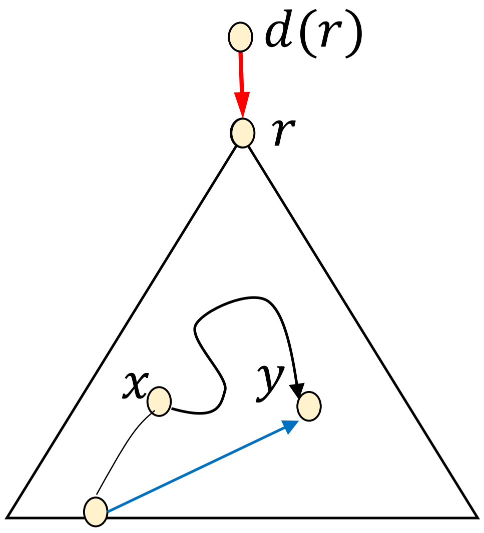
Construction:

- **Keep the paths** using only vertices of the tree
- **For every path** using vertices outside the tree, replace the subpath outside with **a shortcut edge**

Lemma: Two vertices in the same tree (rooted at r) are **disconnected** by $(d(r), r)$ **iff** they **are not** strongly connected in the auxiliary graph of their tree (in the same auxiliary component).

Algorithm: Two vertices u and v are 2-edge-connected iff they are in the same auxiliary component in G and G^R.

Auxiliary components



Idea: Encode all the paths that do not use the incoming strong bridge between **vertices in the same tree** of the bridge decomposition with an **auxiliary graph**.

Construction:

- **Keep the paths** using only vertices of the tree
- **For every path** using vertices outside the tree, replace the subpath outside with **a shortcut edge**

Lemma: Two vertices in the same tree (rooted at r) are **disconnected** by $(d(r), r)$ **iff** they **are not** strongly connected in the auxiliary graph of their tree (in the same auxiliary component).

GOAL: Incrementally maintain the bridge decomposition and the auxiliary components.

Outline

- Definitions
  - 2-edge-connectivity in undirected graphs
  - 2-edge-connectivity in directed graphs
  - Problems definition
  - Known algorithm and our result
- High-level idea
- Basic ingredients
  - Dominators
  - Auxiliary components
- Tools
- Conclusion

Tools

➢ Incremental dominator tree

- **2012** – Georgiadis, Italiano, Laura, Santaroni
  - $O(m \min\{n, k\} + kn)$

➢ Incremental SCCs in each auxiliary graph

- **2009 & 2016** – Bender, Fineman, Gilbert, Tarjan:
  - $O\left(m \min\{\sqrt{m}, n^{2/3}\}\right)$

Combining things…

- **Many instances** of the Incremental SCCs algorithm
- **Vertices can move** across auxiliary graphs
- Auxiliary graphs **can merge**
- …

Concluding remarks

Results:

- Incremental $O(mn)$ algorithm for maintaining the pairwise **2-edge-connectivity** in directed graphs.
- **Answer queries in $O(1)$ time,** whether two vertices are 2-edge-connected. If the two vertices are not 2-edge-connected, we return an edge that separates them.

Open problems:

- Can we maintain incrementally the 2-vertex-connected blocks?
- Decremental? Fully dynamic?

Concluding remarks

Results:

- Incremental $O(mn)$ algorithm for maintaining the pairwise **2-edge-connectivity** in directed graphs.
- **Answer queries in $\boldsymbol{O(1)}$ time,** whether two vertices are 2-edge-connected. If the two vertices are not 2-edge-connected, we return an edge that separates them.

Open problems:

- Can we maintain incrementally the 2-vertex-connected blocks?
- Decremental? Fully dynamic?

Thank you!