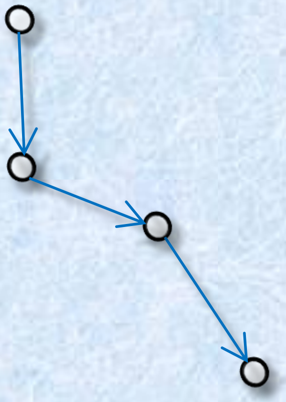
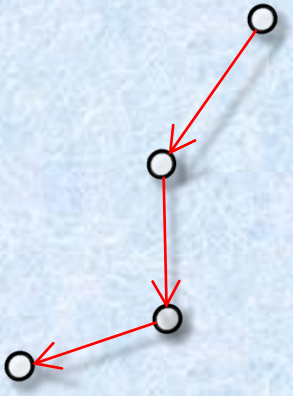


Join-Reachability Problems in Directed Graphs

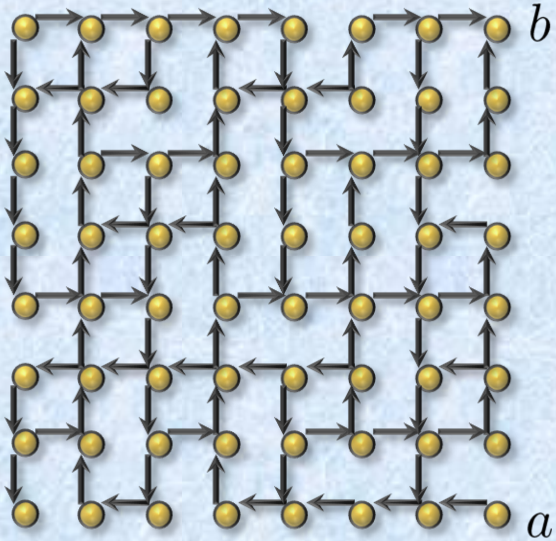


Loukas Georgiadis
University of Western Macedonia, Greece

Stavros Nikolopoulos, Leonidas Palios
University of Ioannina, Greece

Graph Reachability

G



(Directed) Graph $G = (V, A)$

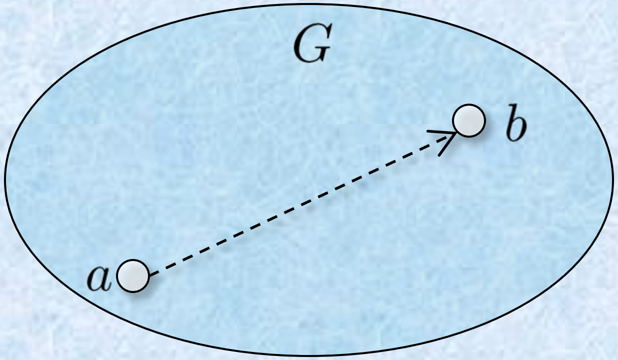
Reachability Query : $a \rightsquigarrow b$?

Is vertex b reachable from vertex a ?

(Is there a path in G from a to b ?)

Goal: Construct a Data Structure that answers reachability queries **efficiently**

Graph Reachability



Reachability Query : $a \rightsquigarrow b$?

Is vertex b reachable from vertex a ?

(Is there a path in G from a to b ?)

Goal: Construct a Data Structure that answers reachability queries **efficiently**

Efficiency of a Data Structure: $\langle s(n), q(n) \rangle$

$O(s(n))$ storage space

$O(q(n))$ query time

Easy : Efficiency $\langle n^2, 1 \rangle$ or $\langle m+n, m+n \rangle$

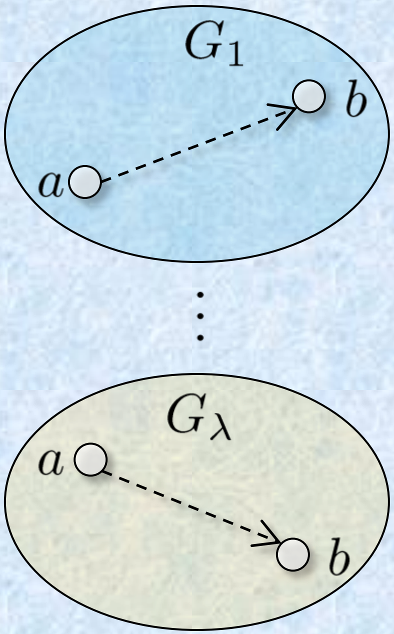
Hard : Efficiency close to $\langle m+n, 1 \rangle$

So far achieved only for restricted graph classes (e.g., trees, planar graphs [Thorup, JACM 2004])

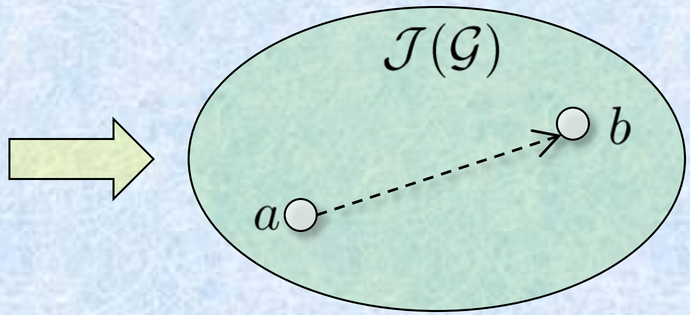
$n = |V|, m = |A|$

Join-Reachability

Collection of graphs $\mathcal{G} = \{G_1, G_2, \ldots, G_\lambda\}$



Join-Reachability Graph



$a \rightsquigarrow b$ in $\mathcal{J}(\mathcal{G})$

$$\Updownarrow$$

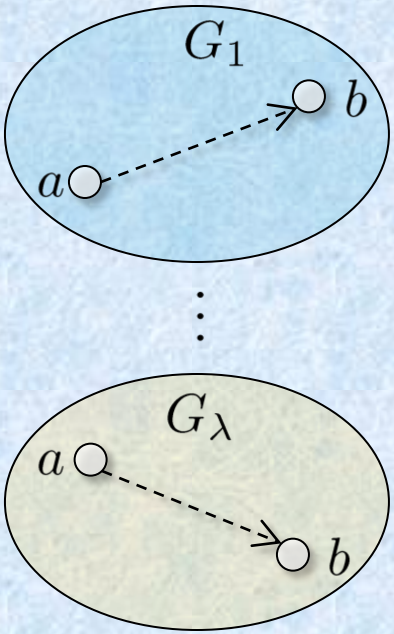
$a \rightsquigarrow b$ in all $G_i \in \mathcal{G}$

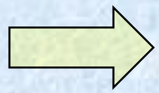
Join-Reachability Query :

Report all vertices that reach b in all graphs $G_i \in \mathcal{G}$

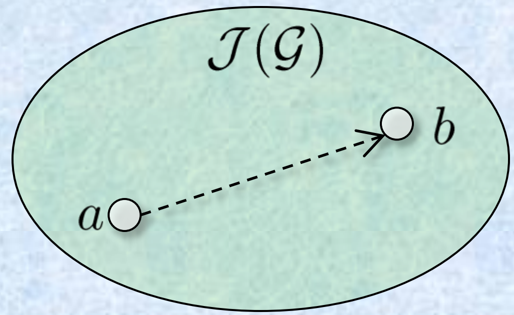
Join-Reachability

Collection of graphs $\mathcal{G} = \{G_1, G_2, \ldots, G_\lambda\}$





Join-Reachability Graph



$a \rightsquigarrow b$ in $\mathcal{J}(\mathcal{G})$

$\Updownarrow$

$a \rightsquigarrow b$ in all $G_i \in \mathcal{G}$

Join-Reachability Query :

Report all vertices that reach b in all graphs $G_i \in \mathcal{G}$

Combinatorial Problem :
- Compute a $\mathcal{J}(\mathcal{G})$ of small size $|V(\mathcal{J}(\mathcal{G}))| + |A(\mathcal{J}(\mathcal{G}))|$
- Compute/approximate smallest $\mathcal{J}(\mathcal{G})$

Data Structure Problem :
- Compute an efficient data structure for $\mathcal{J}(\mathcal{G})$
  - report all vertices a s.t. $a \rightsquigarrow b$ for a query vertex b
  - small space

Join-Reachability

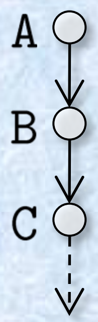
Collection of graphs $\mathcal{G} = \{G_1, G_2, \ldots, G_\lambda\}$

Join-Reachability Query :

Report all vertices that reach b in all graphs $G_i \in \mathcal{G}$

Applications: Graph Algorithms, Data Bases, Natural Language Processing,…

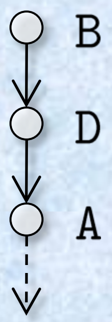
Example: Rank Aggregation




```
Ranking 1

1.  Item A
2.  Item B
3.  Item C
     ...
```

. . .

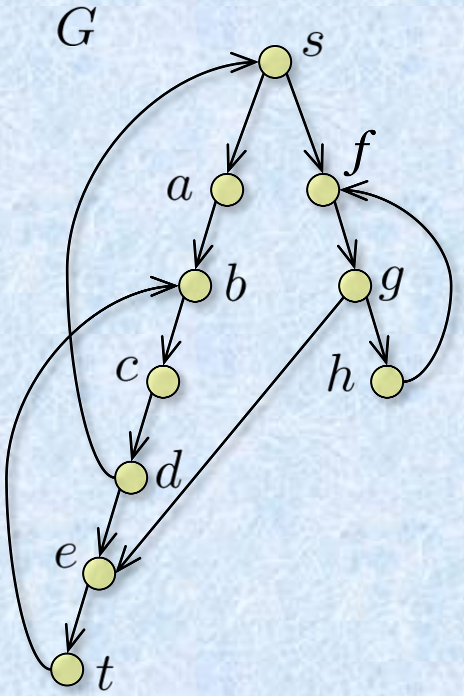
```
Ranking λ

1.  Item B
2.  Item D
3.  Item A
     ...
```

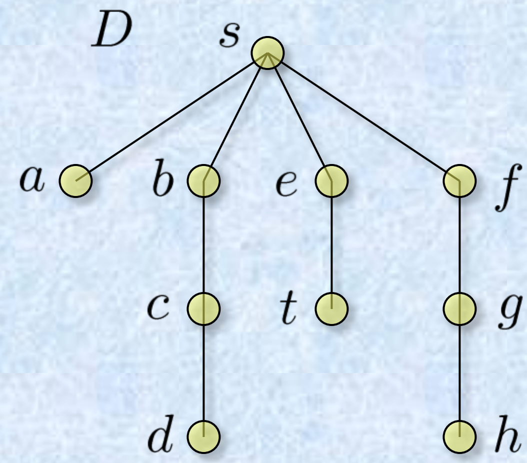


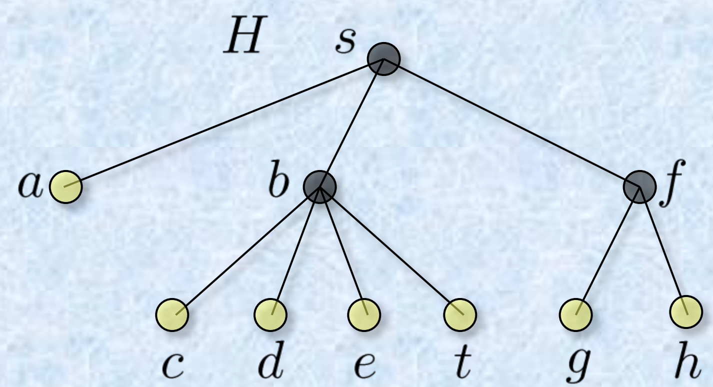
Given a collection of rankings of some items, we would like to report fast all items ranked higher than a query item in all rankings.

Motivation

Computing frequency dominators [Lee, Resnick, Bond, and McKinley ‘07, G. ‘08]







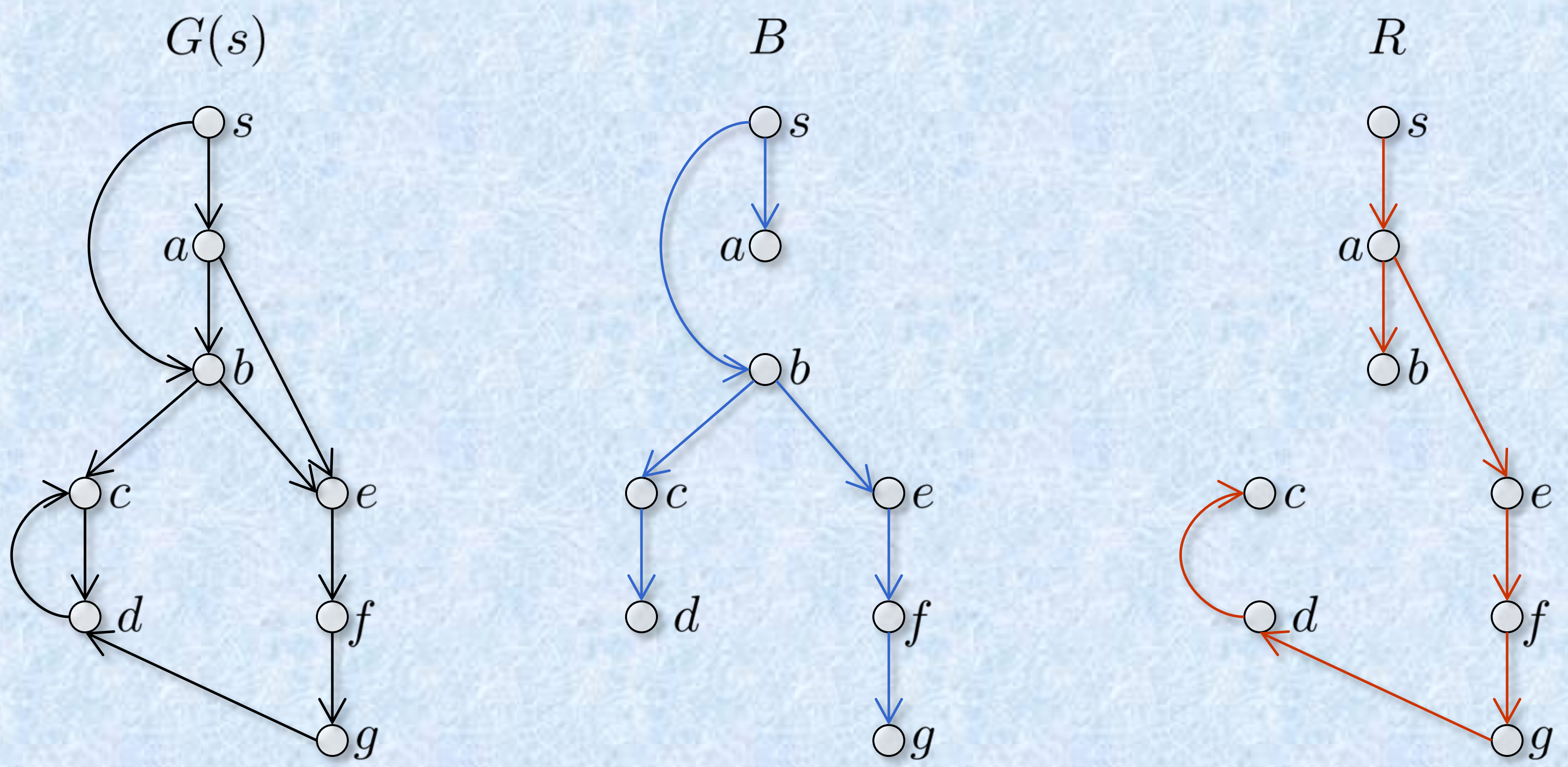
$$x \in H(\hat{h}(y)) \;\wedge\; x \in dom(y)$$

Motivation

Applications of Independent Spanning Trees [G. and Tarjan 2005, 2011]

Any flowgraph $G(s) = (V, A, s)$ has two spanning trees, B and R, such that for any $v \in V$

$$B[s, v] \cap R[s, v] = dom(v)$$



Motivation

Computing Pairs of Disjoint s-t Paths [G. 2010]

Data Structure : Given rooted trees S_1 and S_2 on the same nodes support the operations:

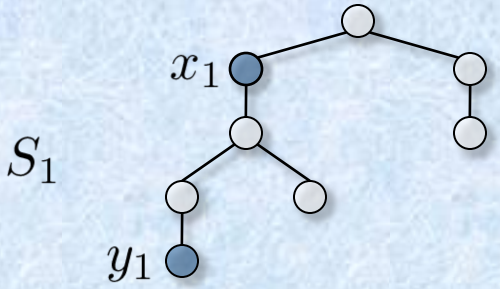
(i) Test if $S_1[x_1, y_1]$ contains x_2.

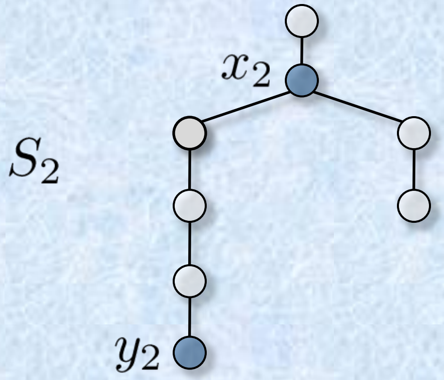
(ii) Return the topmost vertex in $S_1(x_1, y_1]$.

(iii) Test if $S_1[x_1, y_1]$ and $S_2[x_2, y_2]$ contain a common vertex.

(iv) Find the lowest ancestor of y_2 in $S_2[x_2, y_2]$ that is contained in $S_1[x_1, y_1]$.

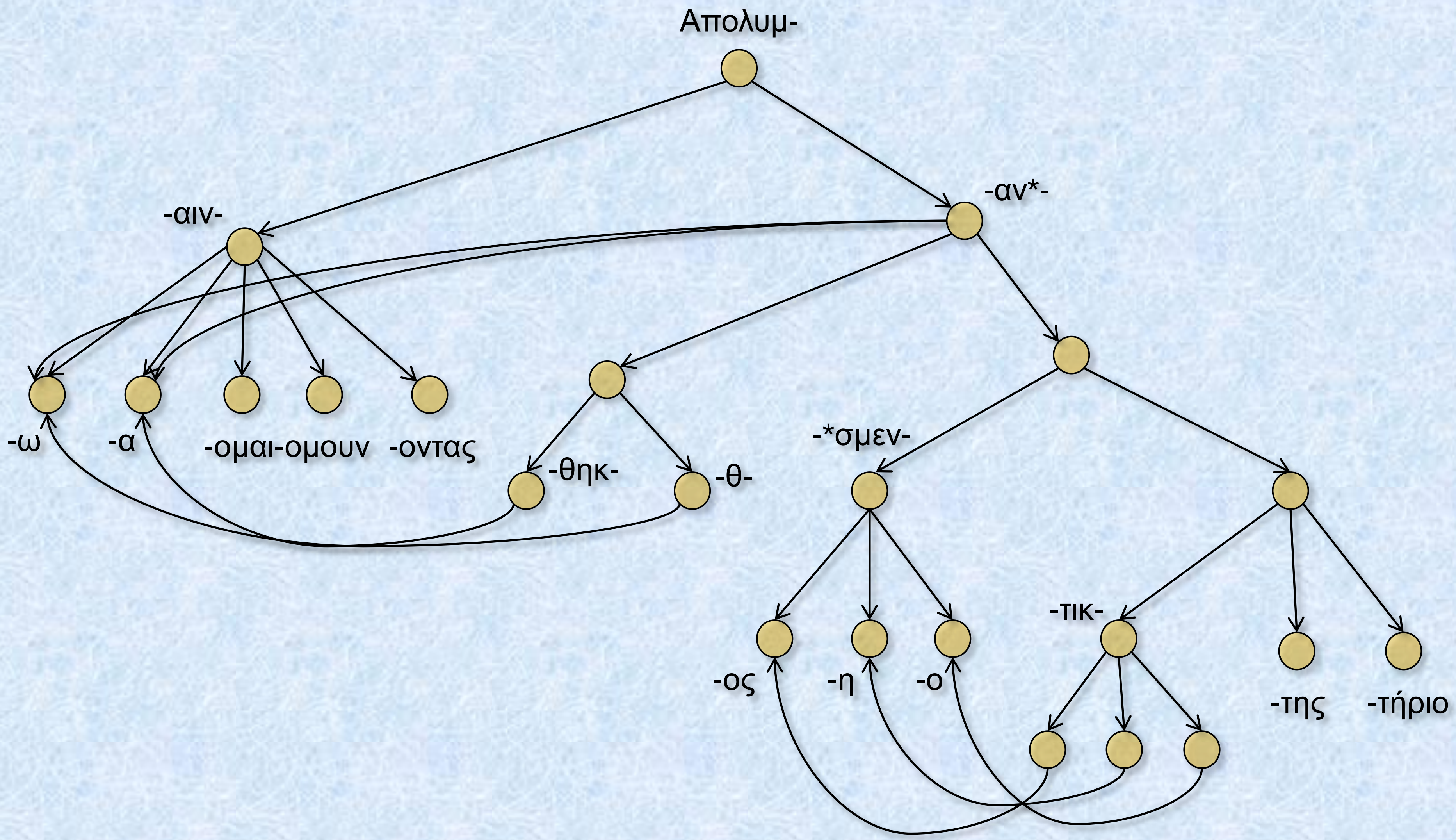
(v) Find the highest ancestor of y_2 in $S_2[x_2, y_2]$ that is contained in $S_1[x_1, y_1]$.





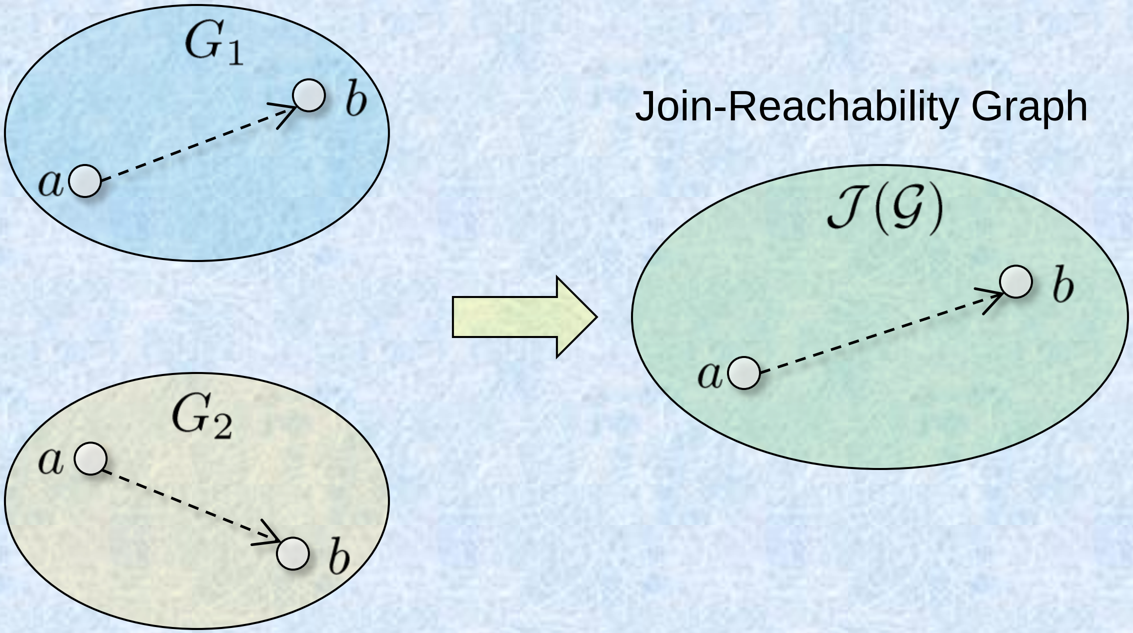
Computational Morphological Analysis

Morphological patterns as graph reachability problems



Join-Reachability

We consider the case of two digraphs: $\mathcal{G} = \{G_1, G_2\}$



$a \rightsquigarrow b$ in $\mathcal{J}(\mathcal{G})$

$\Updownarrow$

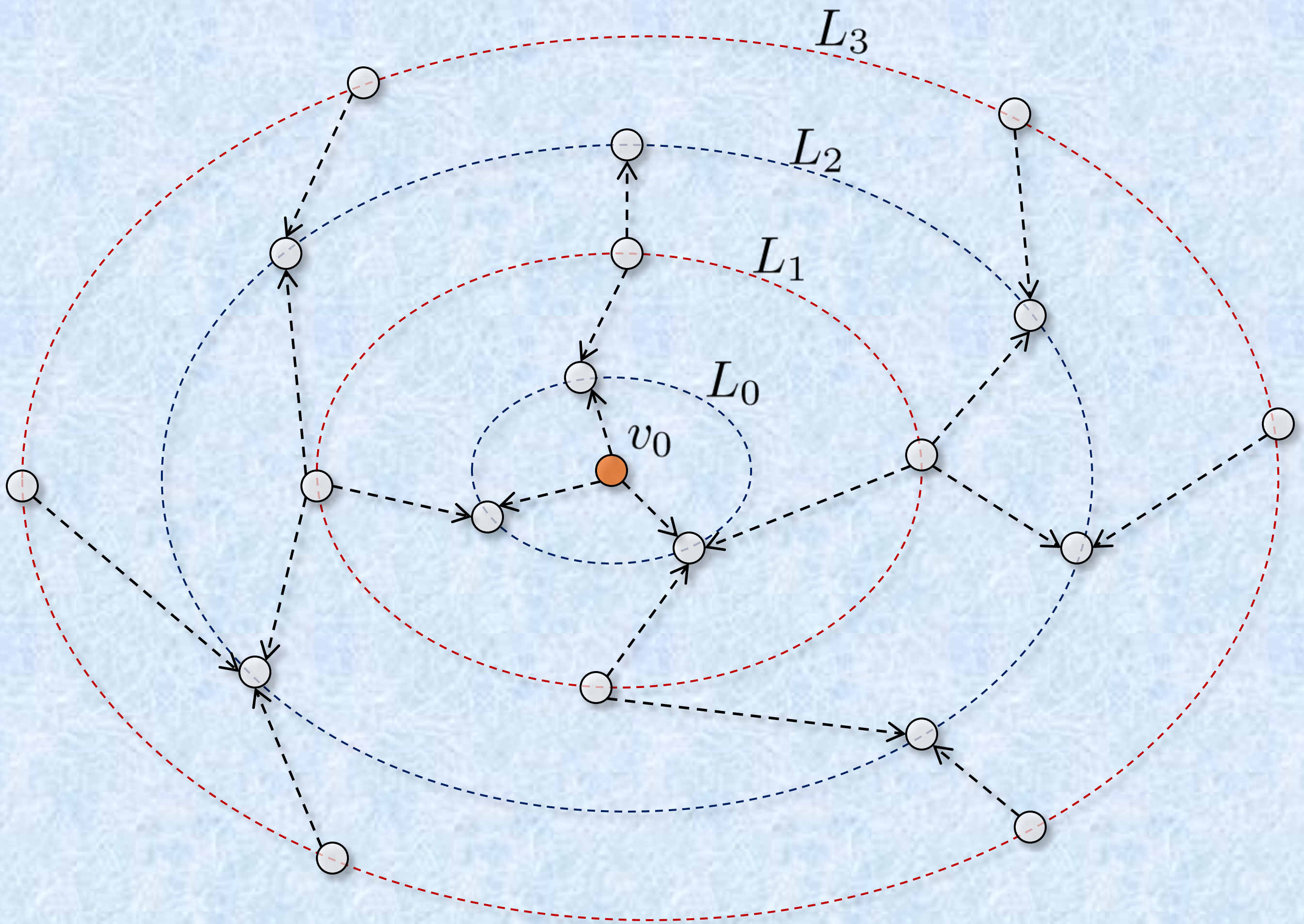
$a \rightsquigarrow b$ in G_1 and G_2

Outline

- Graph Reachability
- Join-Reachability Problems
  - Motivation
  - **Preprocessing**
    - Layer Decomposition
    - Removing Cycles
  - Join-Reachability Graph
    - Computational Complexity
    - Combinatorial Complexity
  - Join-Reachability Data Structures
- Concluding Remarks

Thorup's Layer Decomposition

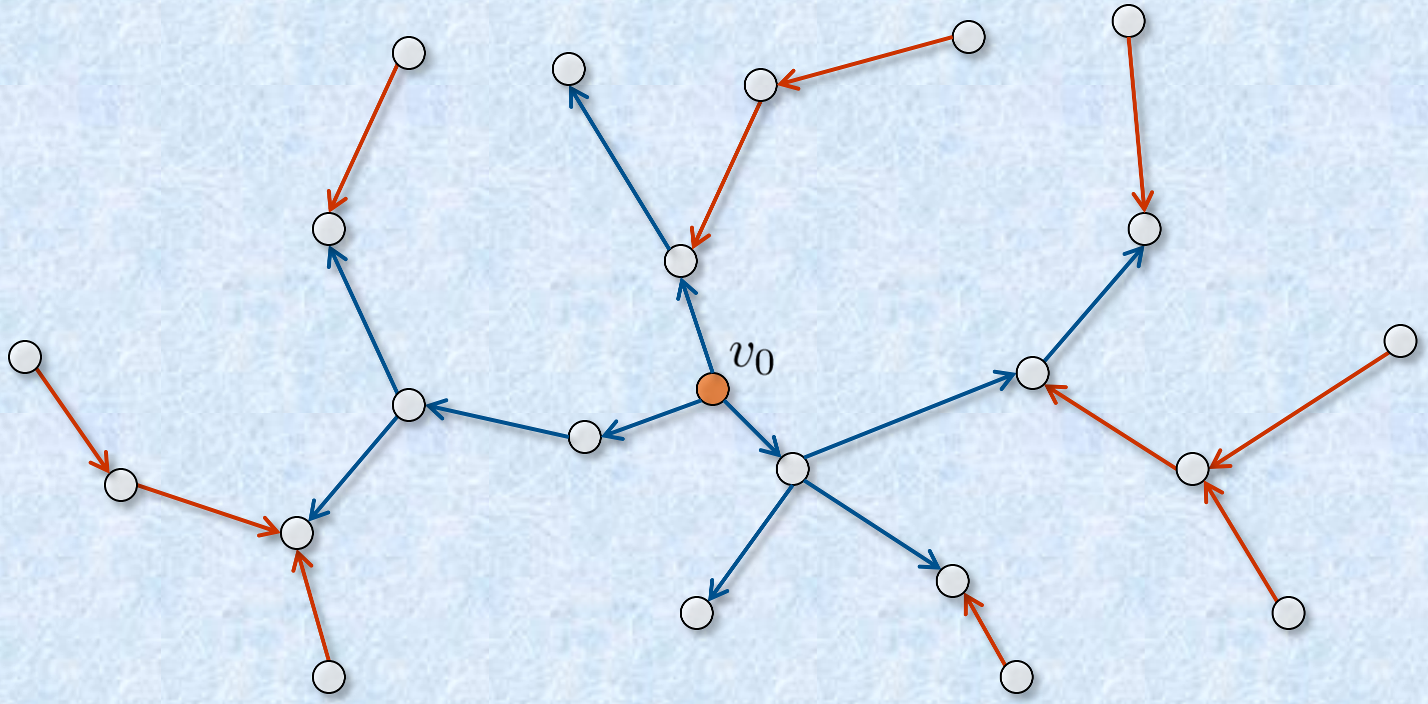
Reduces digraph reachabiliy to reachability in **2-layered** digraphs $G^0, G^1, G^2, \ldots$



Thorup's Layer Decomposition

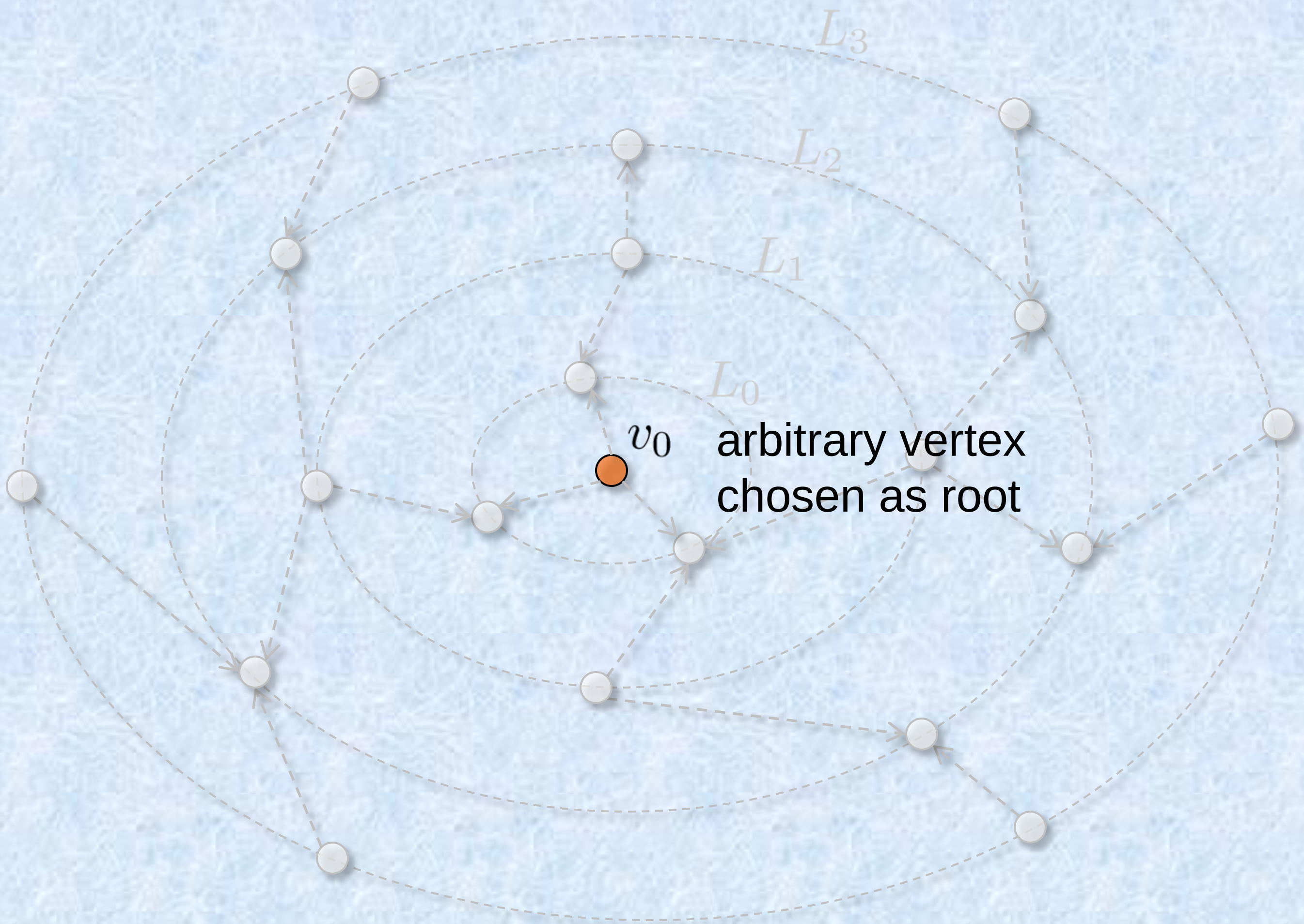
Reduces digraph reachabiliy to reachability in **2-layered** digraphs $G^0, G^1, G^2, \ldots$

2-layered digraph : Has a 2-layered spanning tree, i.e. every undirected root-to-leaf path consists of 2 directed paths



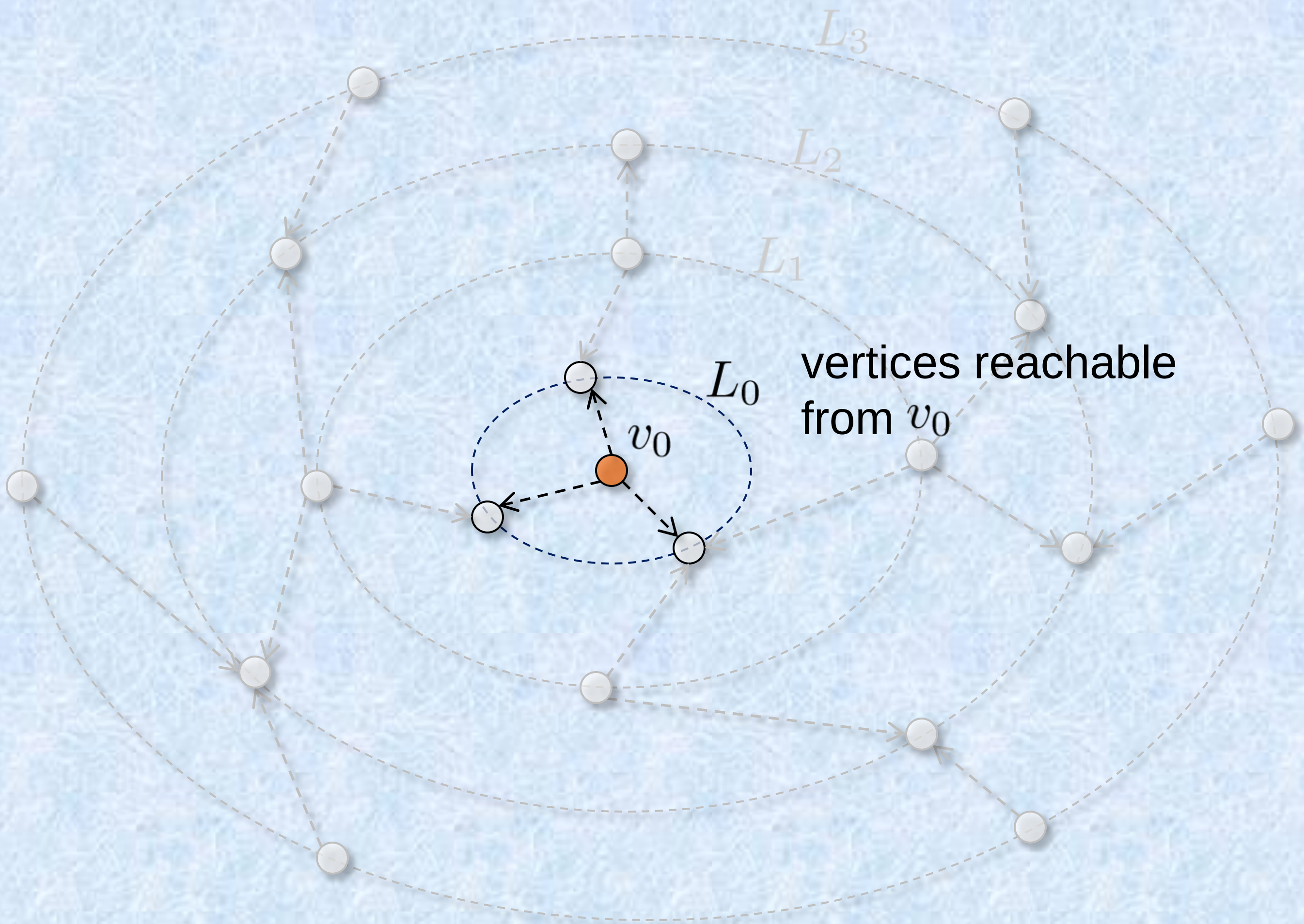
Thorup's Layer Decomposition

Reduces digraph reachabiliy to reachability in **2-layered** digraphs $G^0, G^1, G^2, \ldots$



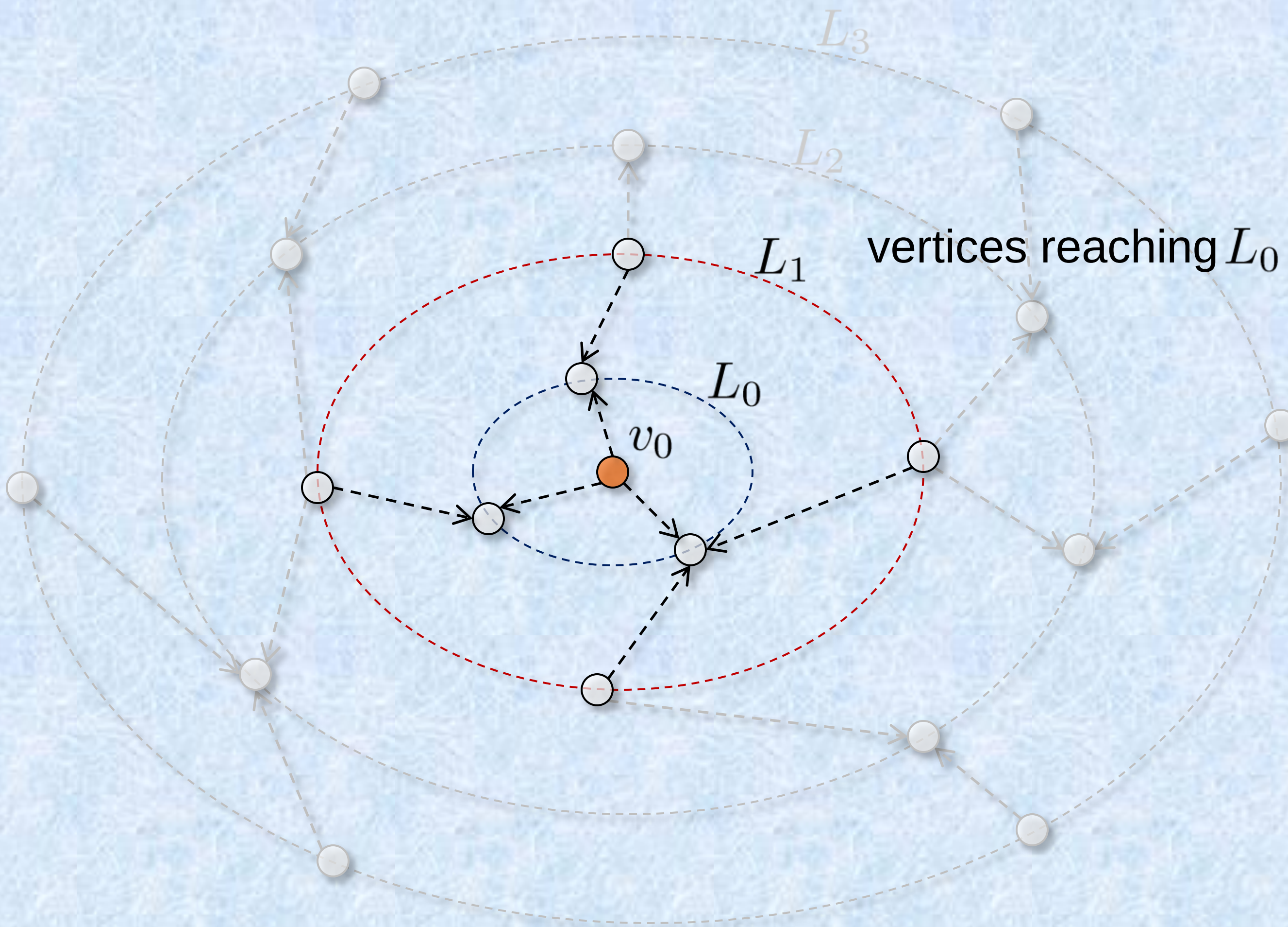
Thorup's Layer Decomposition

Reduces digraph reachabiliy to reachability in **2-layered** digraphs $G^0, G^1, G^2, \ldots$



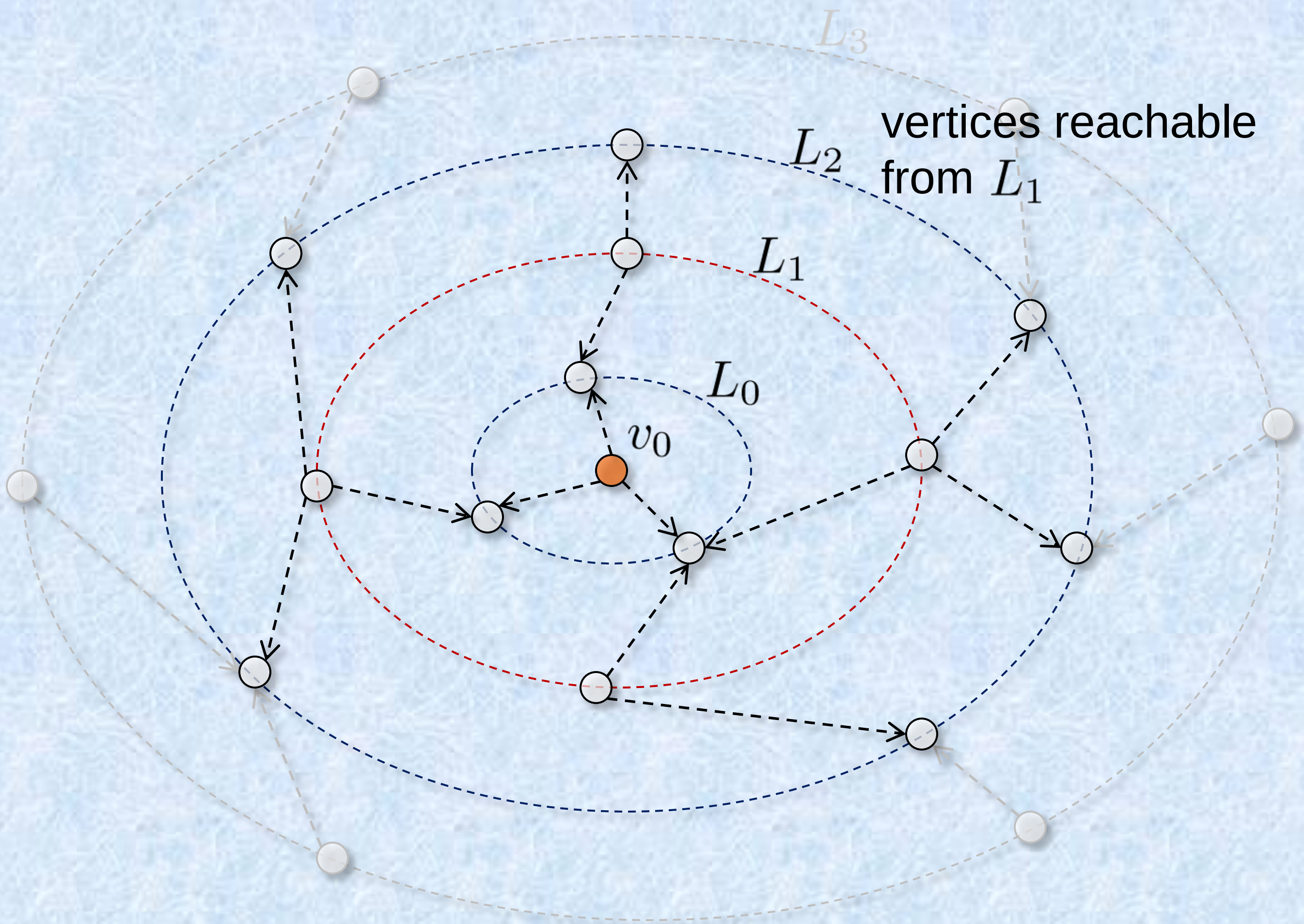
Thorup's Layer Decomposition

Reduces digraph reachabiliy to reachability in **2-layered** digraphs $G^0, G^1, G^2, \ldots$



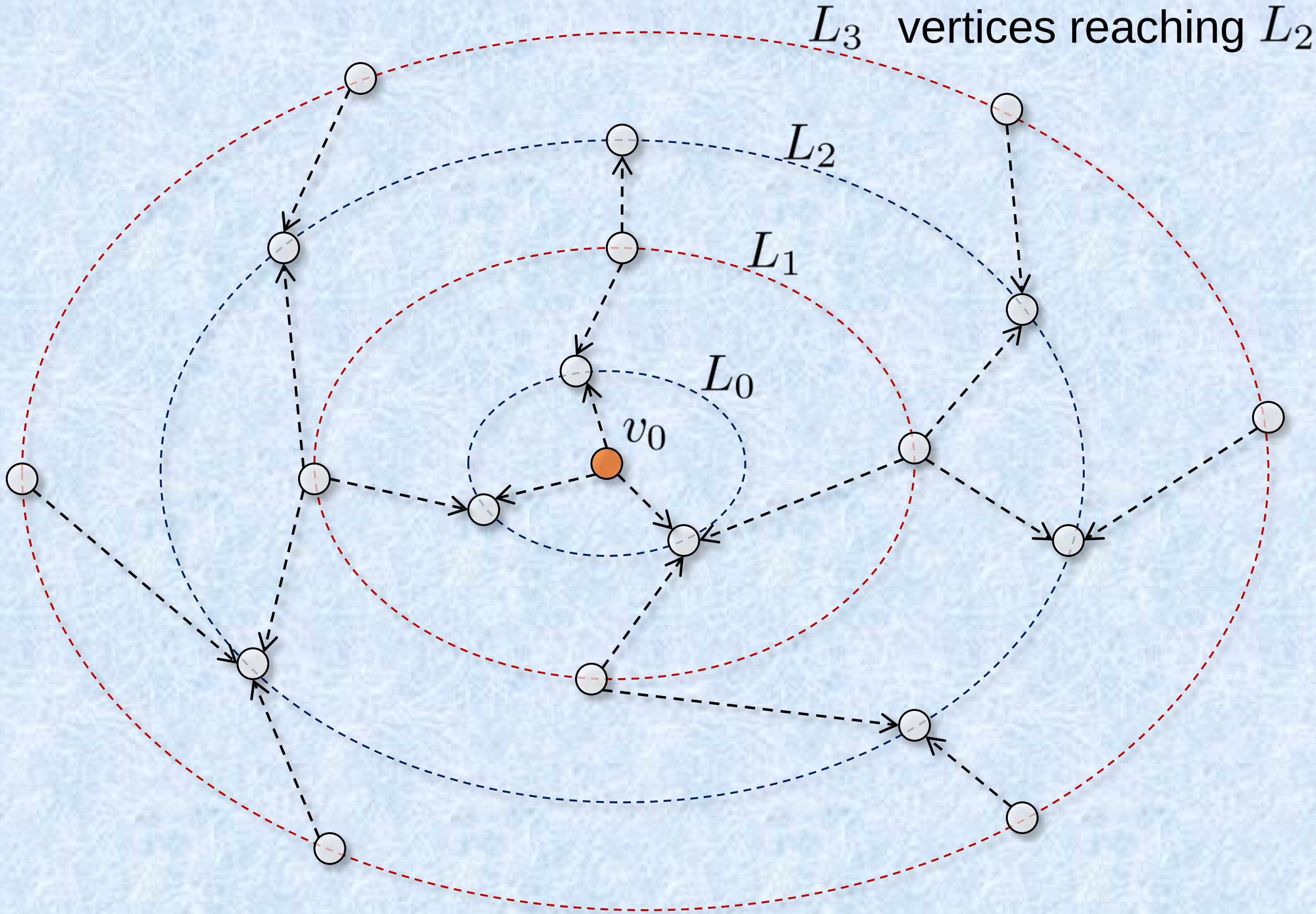
Thorup's Layer Decomposition

Reduces digraph reachabiliy to reachability in **2-layered** digraphs $G^0, G^1, G^2, \ldots$



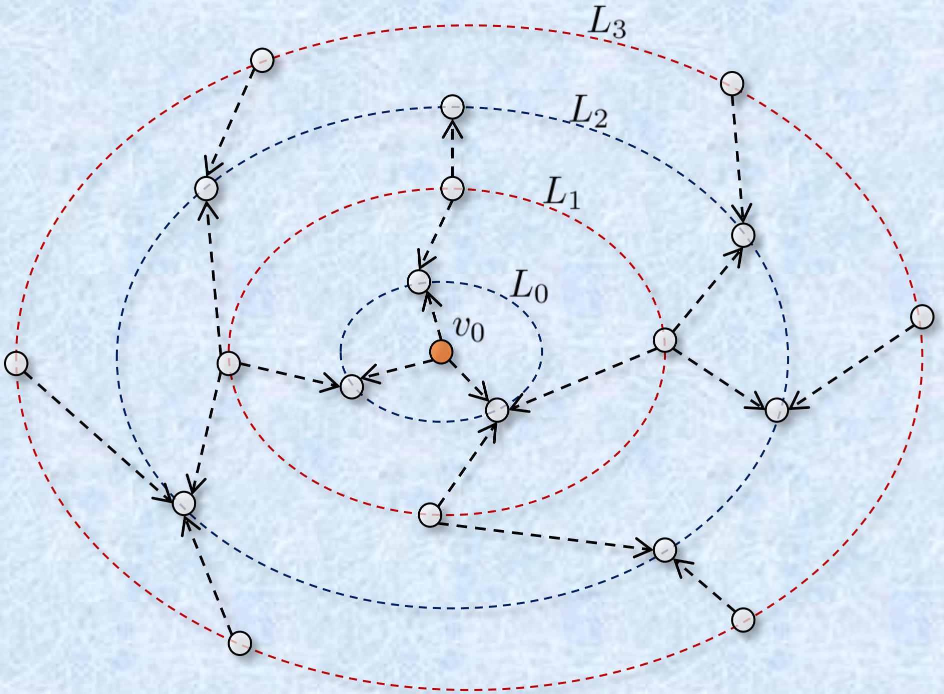
Thorup's Layer Decomposition

Reduces digraph reachabiliy to reachability in **2-layered** digraphs $G^0, G^1, G^2, \ldots$



Thorup's Layer Decomposition

Reduces digraph reachabiliy to reachability in **2-layered** digraphs $G^0, G^1, G^2, \ldots$



G^i is induced by L_i and L_{i+1}

$\iota(v) =$ index of layer containing v

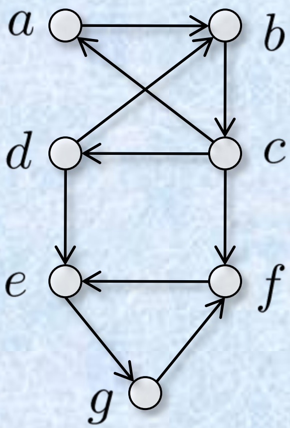
$$u \rightsquigarrow_G v$$
$$\Rightarrow \quad u \in \left(L_{\iota(v)-1} \cup L_{\iota(v)} \cup L_{\iota(v)+1}\right)$$
$$\Rightarrow \quad u \rightsquigarrow_{G^{\iota(v)-1}} v \text{ or } u \rightsquigarrow_{G^{\iota(v)}} v$$

We can use this method to reduce general join-reachabiliy to join-reachability in 2-layered digraphs

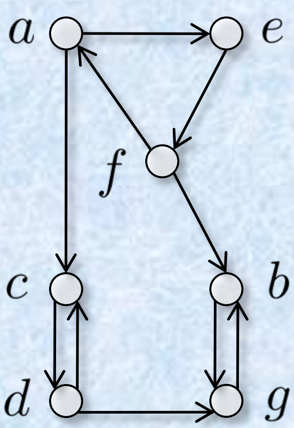
Removing Cycles

Reduces digraph (join-)reachabiliy to (join-)reachability in acyclic digraphs

G_1

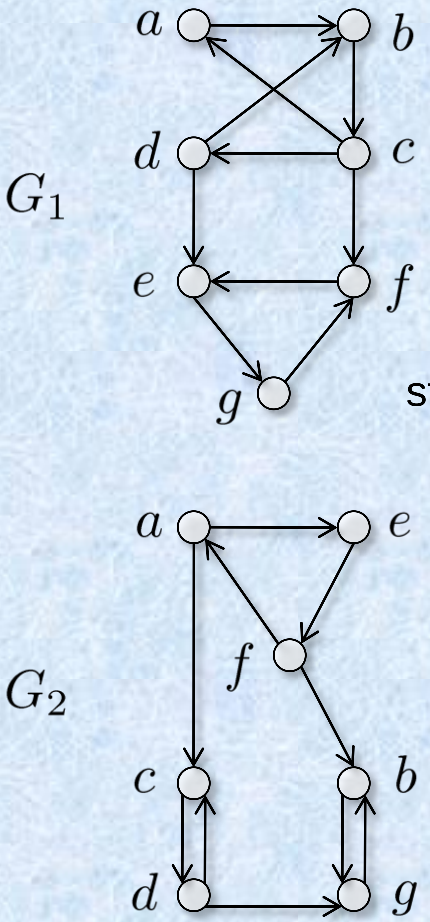


G_2

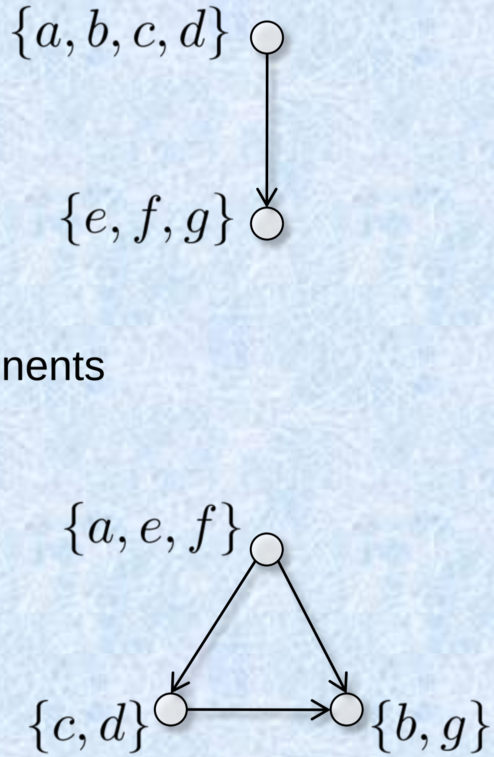


Removing Cycles

Reduces digraph (join-)reachabiliy to (join-)reachability in acyclic digraphs

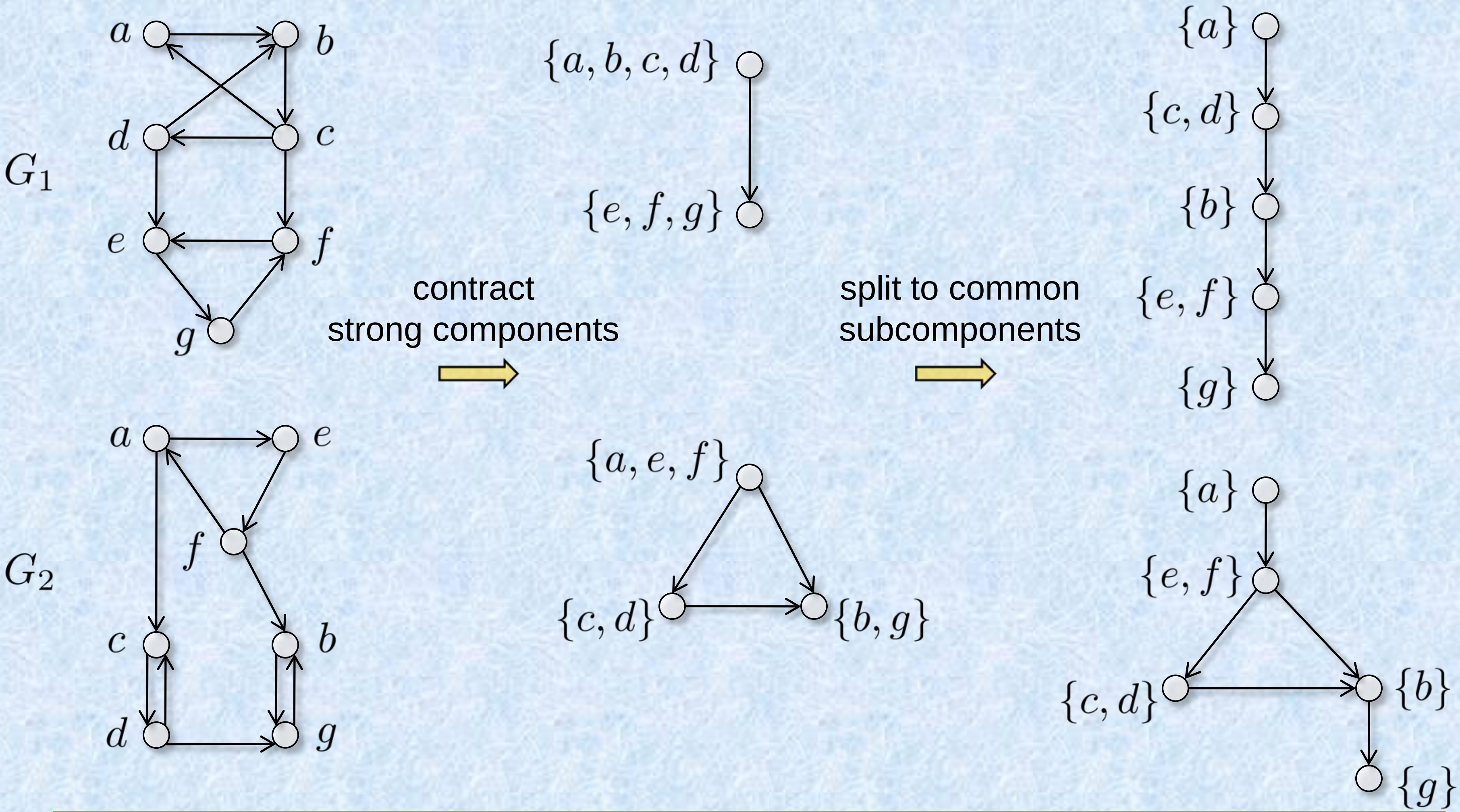


contract
strong components



Removing Cycles

Reduces digraph (join-)reachabiliy to (join-)reachability in acyclic digraphs

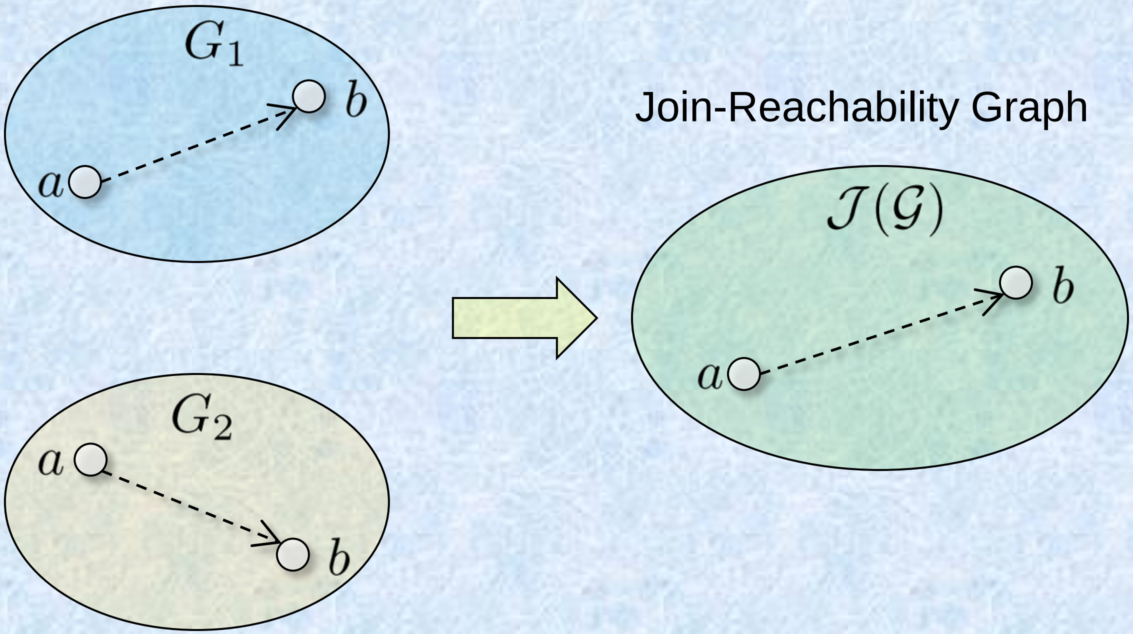


Outline

- Graph Reachability
- Join-Reachability Problems
  - Motivation
  - Preprocessing
    - Layer Decomposition
    - Removing Cycles
  - **Join-Reachability Graph**
    - Computational Complexity
    - Combinatorial Complexity
  - Join-Reachability Data Structures
- Concluding Remarks

Join-Reachability Graph

Collection of graphs $\mathcal{G} = \{G_1, G_2\}$



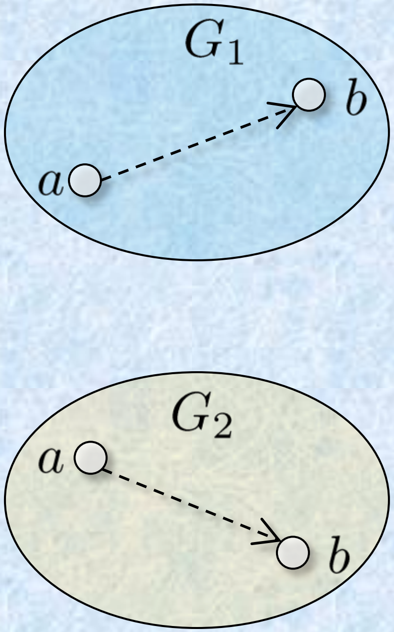
$a \rightsquigarrow b$ in $\mathcal{J}(\mathcal{G})$

$$\Updownarrow$$

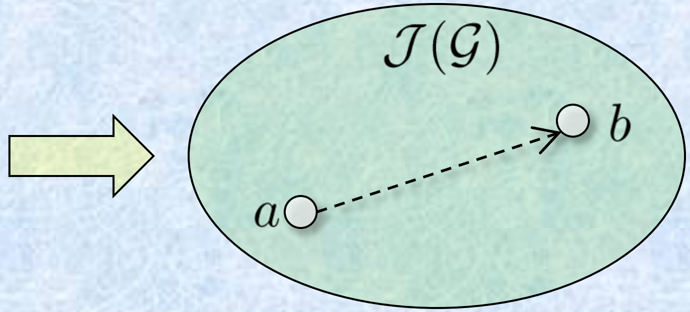
$a \rightsquigarrow b$ in G_1 and G_2

Join-Reachability Graph: Computational Complexity

Collection of graphs $\mathcal{G} = \{G_1, G_2\}$



Join-Reachability Graph



$a \rightsquigarrow b$ in $\mathcal{J}(\mathcal{G})$

$\Updownarrow$

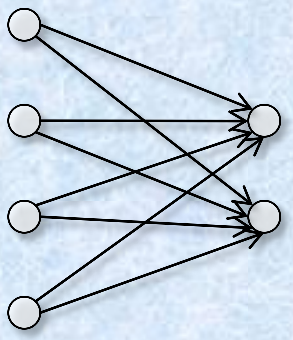
$a \rightsquigarrow b$ in G_1 and G_2

Two cases:

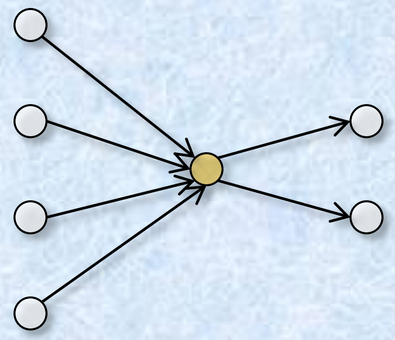
- $V(\mathcal{J}(\mathcal{G})) = V$ (Steiner vertices not allowed)
  smallest $\mathcal{J}(\mathcal{G})$ is polynomial-time computable
- $V(\mathcal{J}(\mathcal{G})) \supseteq V$ (Steiner vertices allowed)
  smallest $\mathcal{J}(\mathcal{G})$ is NP-hard to compute

Join-Reachability Graph

Steiner vertices $V(\mathcal{J}) \setminus V$ can significantly reduce the size of $\mathcal{J}$



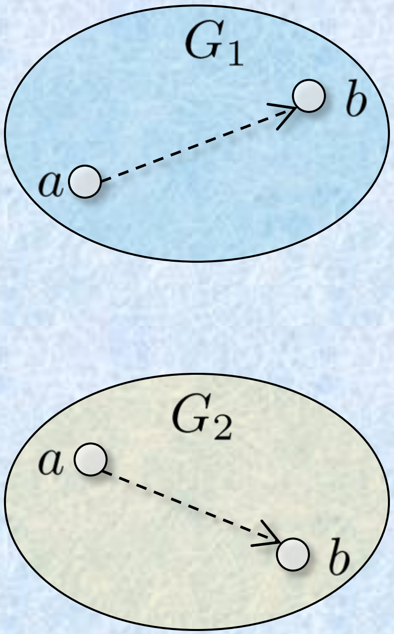
$V(\mathcal{J}) = V$



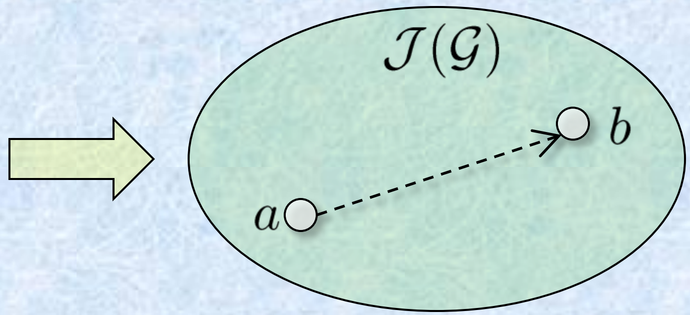
$V(\mathcal{J}) \supset V$

Join-Reachability Graph: Combinatorial Complexity

Collection of graphs $\mathcal{G} = \{G_1, G_2\}$



Join-Reachability Graph



$a \rightsquigarrow b$ in $\mathcal{J}(\mathcal{G})$

$\Updownarrow$

$a \rightsquigarrow b$ in G_1 and G_2

We bound the size of $\mathcal{J}(\mathcal{G})$ when Steiner vertices are allowed

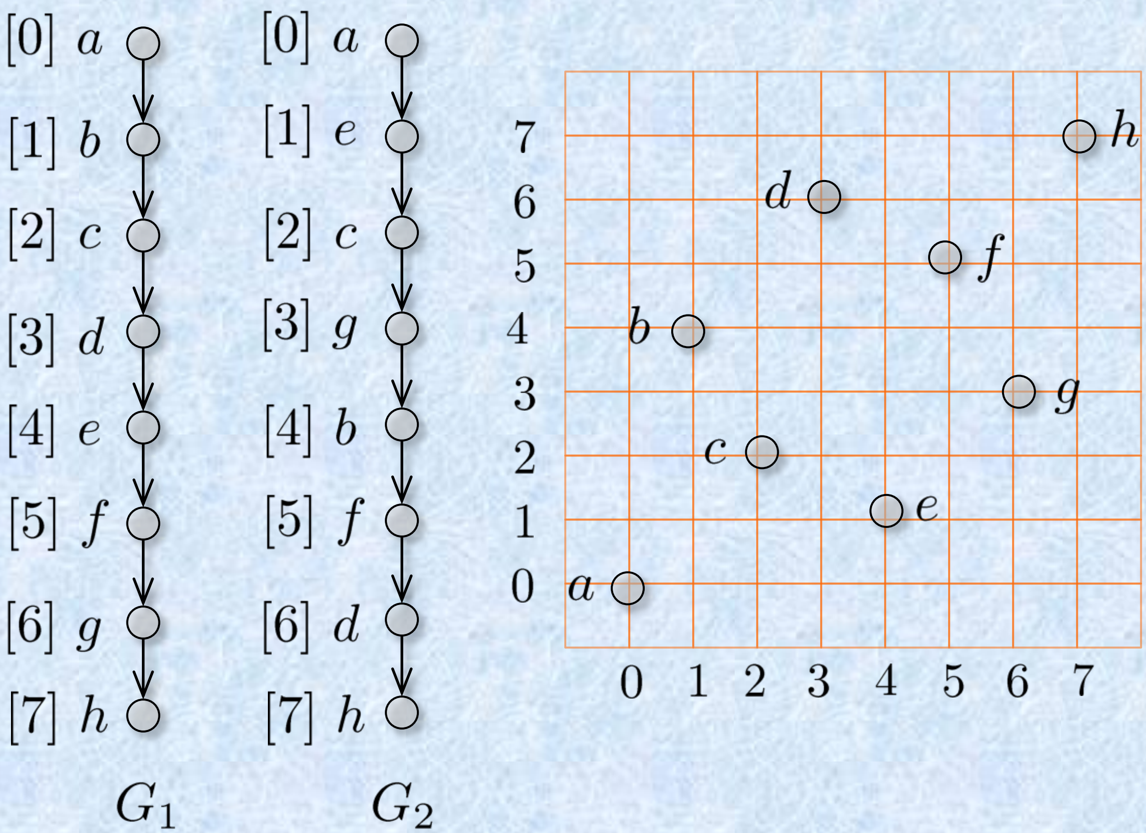
Main Idea: Geometric representation of join-reachability for paths and trees

Join-Reachability Graph: Combinatorial Complexity

Construction of a compact join-reachability graph $\mathcal{J}(\mathcal{G})$ for paths

Each vertex $v \in V$ is mapped to a point $(x_1(v), x_2(v))$

$x_i(v) =$ number of vertices reachable from v in G_i



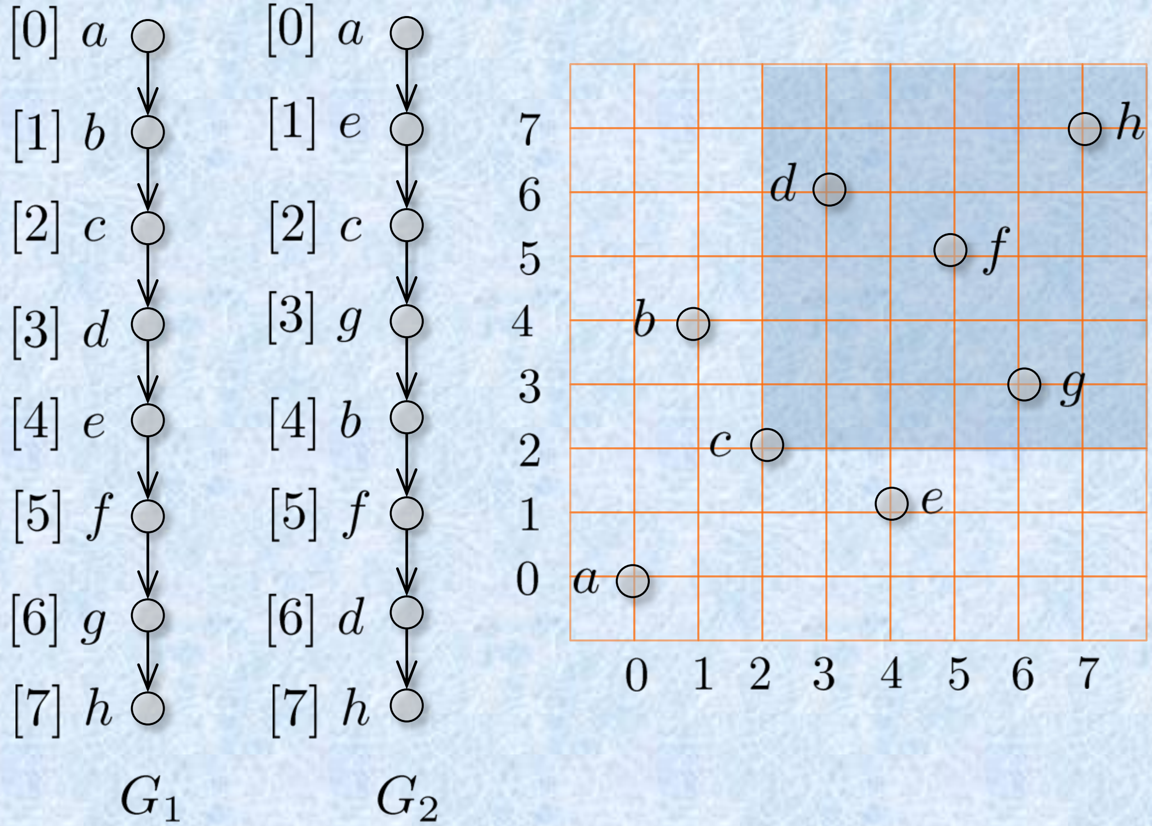
Join-Reachability Graph: Combinatorial Complexity

Construction of a compact join-reachability graph $\mathcal{J}(\mathcal{G})$ for paths

Each vertex $v \in V$ is mapped to a point $(x_1(v), x_2(v))$

$x_i(v) =$ number of vertices reachable from v in G_i

$$u \rightsquigarrow_{\mathcal{J}} v \Leftrightarrow (x_1(u), x_2(u)) \leq (x_1(v), x_2(v))$$



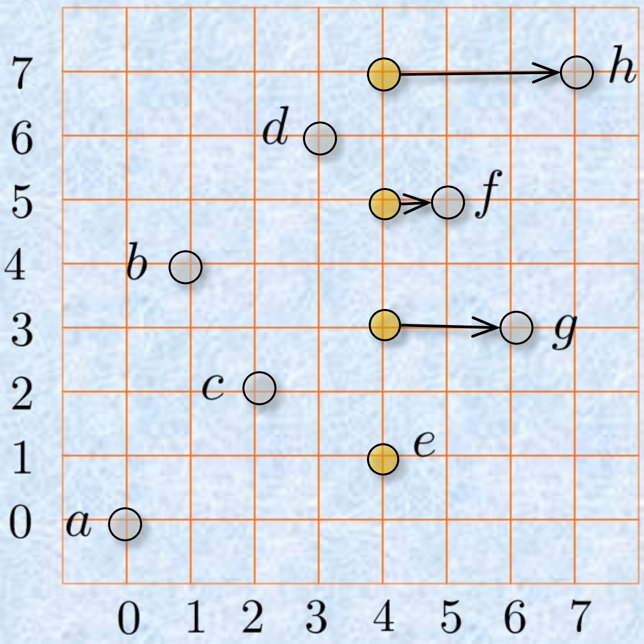
Join-Reachability Graph: Combinatorial Complexity

Construction of a compact join-reachability graph $\mathcal{J}(\mathcal{G})$ for paths

Each vertex $v \in V$ is mapped to a point $(x_1(v), x_2(v))$

$x_i(v) =$ number of vertices reachable from v in G_i

$$u \rightsquigarrow_{\mathcal{J}} v \Leftrightarrow (x_1(u), x_2(u)) \leq (x_1(v), x_2(v))$$



For each $v \in V$, such that $x_1(v) \geq \lfloor n/2 \rfloor$ add Steiner vertex s with coordinates $(\lfloor n/2 \rfloor, x_2(v))$ and arc (s, v)

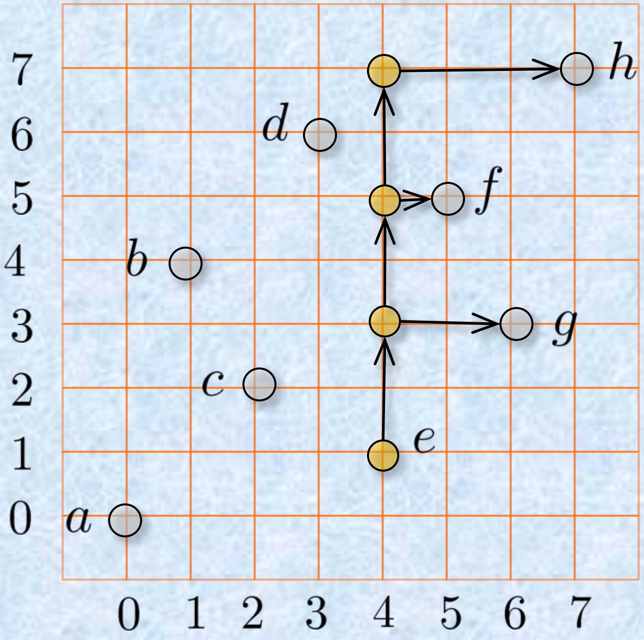
Join-Reachability Graph: Combinatorial Complexity

Construction of a compact join-reachability graph $\mathcal{J}(\mathcal{G})$ for paths

Each vertex $v \in V$ is mapped to a point $(x_1(v), x_2(v))$

$x_i(v) =$ number of vertices reachable from v in G_i

$$u \rightsquigarrow_{\mathcal{J}} v \Leftrightarrow (x_1(u), x_2(u)) \leq (x_1(v), x_2(v))$$



For each $v \in V$, such that $x_1(v) \geq \lfloor n/2 \rfloor$ add Steiner vertex s with coordinates $(\lfloor n/2 \rfloor, x_2(v))$ and arc (s, v)

Connect Steiner vertices in a bottom-up path

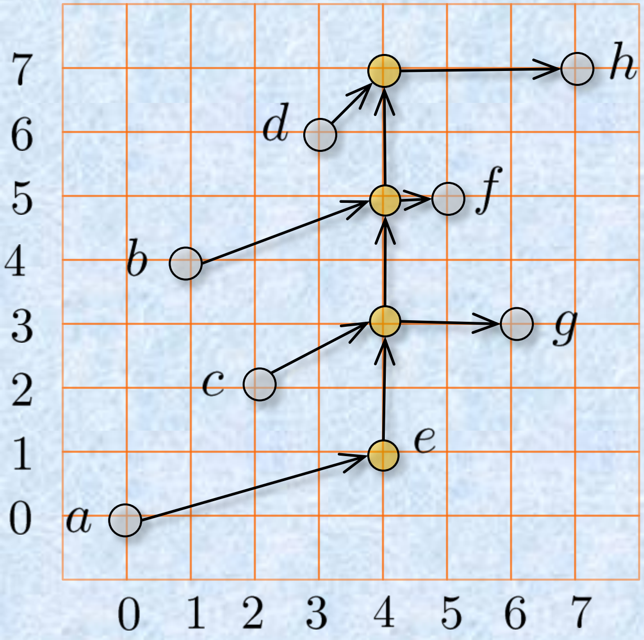
Join-Reachability Graph: Combinatorial Complexity

Construction of a compact join-reachability graph $\mathcal{J}(\mathcal{G})$ for paths

Each vertex $v \in V$ is mapped to a point $(x_1(v), x_2(v))$

$x_i(v) =$ number of vertices reachable from v in G_i

$$u \rightsquigarrow_{\mathcal{J}} v \Leftrightarrow (x_1(u), x_2(u)) \leq (x_1(v), x_2(v))$$



For each $v \in V$, such that $x_1(v) \geq \lfloor n/2 \rfloor$ add Steiner vertex s with coordinates $(\lfloor n/2 \rfloor, x_2(v))$ and arc (s, v)

Connect Steiner vertices in a bottom-up path

For each $v \in V$, such that $x_1(v) < \lfloor n/2 \rfloor$ add arc (v, s)

$s =$ nearest Steiner neighbor with $x_2(s) \leq x_2(v)$

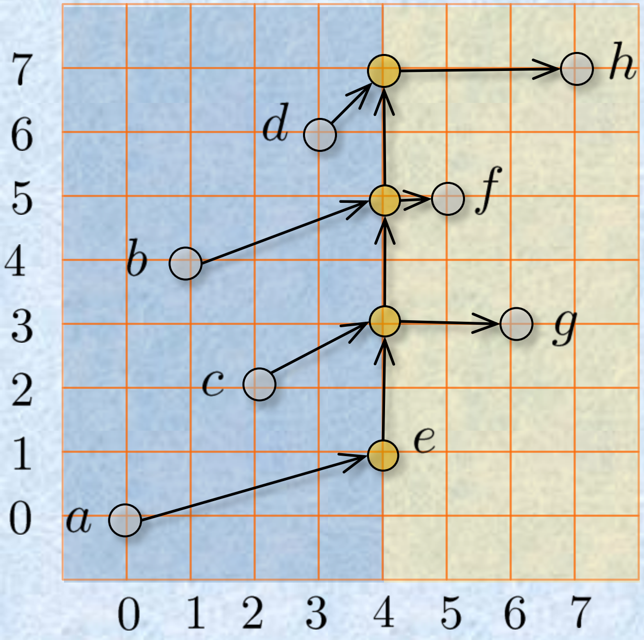
Join-Reachability Graph: Combinatorial Complexity

Construction of a compact join-reachability graph $\mathcal{J}(\mathcal{G})$ for paths

Each vertex $v \in V$ is mapped to a point $(x_1(v), x_2(v))$

$x_i(v) =$ number of vertices reachable from v in G_i

$$u \rightsquigarrow_{\mathcal{J}} v \Leftrightarrow (x_1(u), x_2(u)) \leq (x_1(v), x_2(v))$$



For each $v \in V$, such that $x_1(v) \geq \lfloor n/2 \rfloor$ add Steiner vertex s with coordinates $(\lfloor n/2 \rfloor, x_2(v))$ and arc (s, v)

Connect Steiner vertices in a bottom-up path

For each $v \in V$, such that $x_1(v) < \lfloor n/2 \rfloor$ add arc (v, s)

$s =$ nearest Steiner neighbor with $x_2(s) \leq x_2(v)$

Use recursion for the sets $L = \{v \in V \mid x_1(v) < n/2\}$
and $R = \{v \in V \mid x_1(v) > n/2\}$

Upper Bound: $O(n)$ arcs + vertices per recursion level $\Rightarrow$ $|\mathcal{J}(G)| = O(n \log n)$

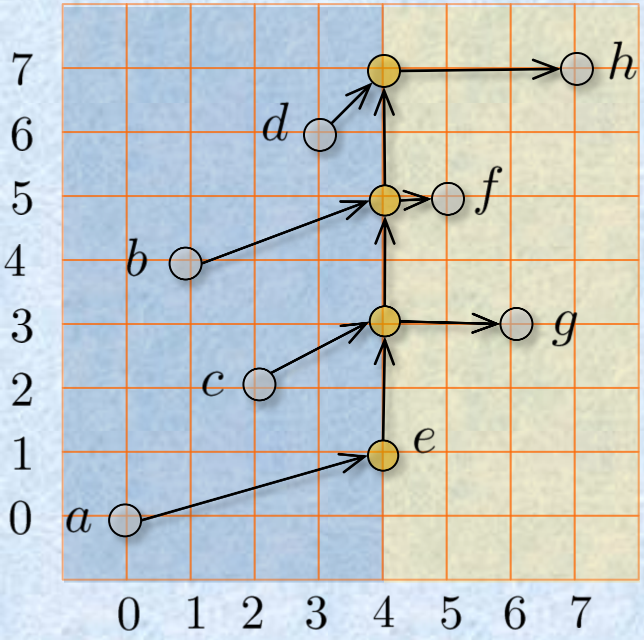
Join-Reachability Graph: Combinatorial Complexity

Construction of a compact join-reachability graph $\mathcal{J}(\mathcal{G})$ for paths

Each vertex $v \in V$ is mapped to a point $(x_1(v), x_2(v))$

$x_i(v) =$ number of vertices reachable from v in G_i

$$u \leadsto_{\mathcal{J}} v \Leftrightarrow (x_1(u), x_2(u)) \le (x_1(v), x_2(v))$$



For each $v \in V$, such that $x_1(v) \ge \lfloor n/2 \rfloor$ add Steiner vertex s with coordinates $(\lfloor n/2 \rfloor, x_2(v))$ and arc (s, v)

Connect Steiner vertices in a bottom-up path

For each $v \in V$, such that $x_1(v) < \lfloor n/2 \rfloor$ add arc (v, s)

$s =$ nearest Steiner neighbor with $x_2(s) \le x_2(v)$

Use recursion for the sets $L = \{v \in V \mid x_1(v) < n/2\}$
and $R = \{v \in V \mid x_1(v) > n/2\}$

Upper Bound: $O(n)$ arcs + vertices per recursion level $\Rightarrow$ $|\mathcal{J}(G)| = O(n \log n)$

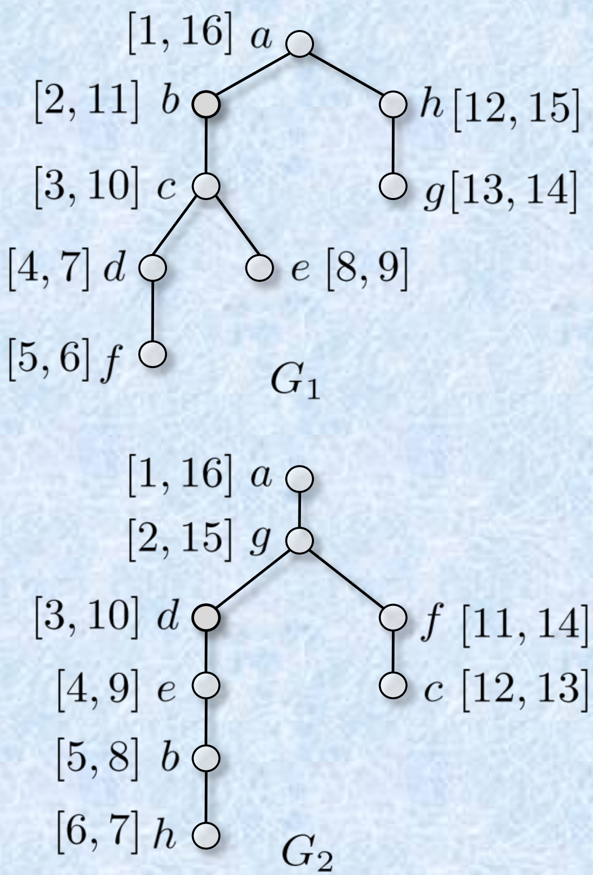
Lower Bound : There are instances for which $|\mathcal{J}(G)| = \Omega(n \log n)$

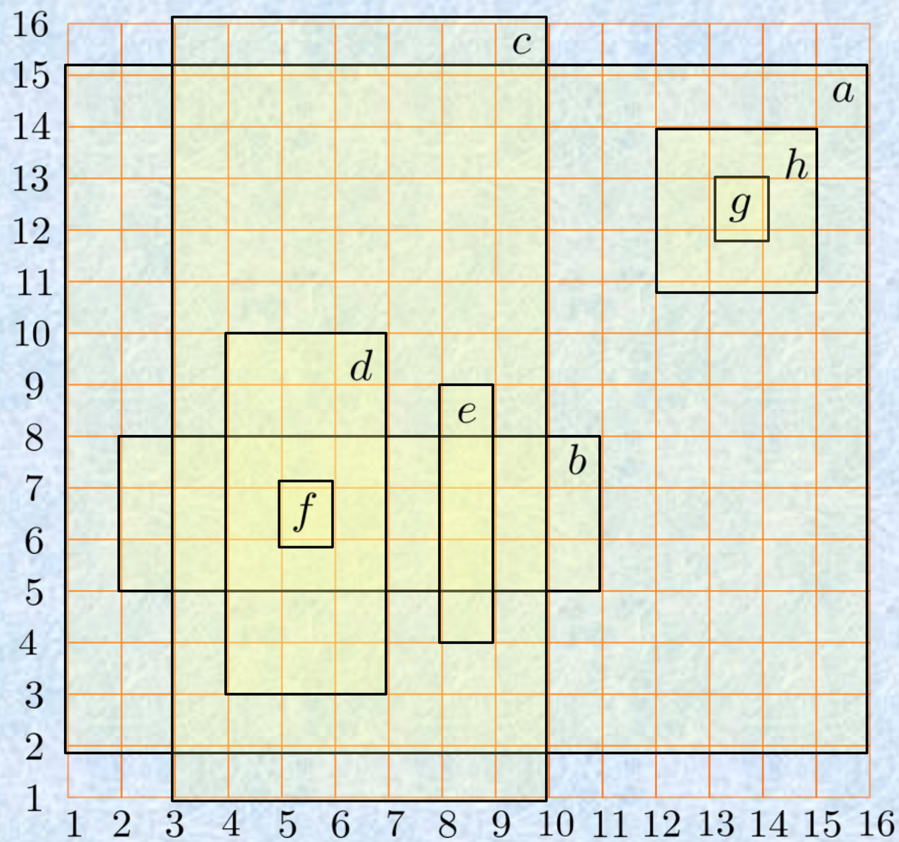
Join-Reachability Graph: Combinatorial Complexity

Construction of a compact join-reachability graph $\mathcal{J}(\mathcal{G})$ for trees

Each vertex $v \in V$ is mapped to a rectangle $R(v) = I_1(v) \times I_2(v)$

$I_i(v) = [s_i(v), t_i(v)] =$ depth-first search interval of v in G_i





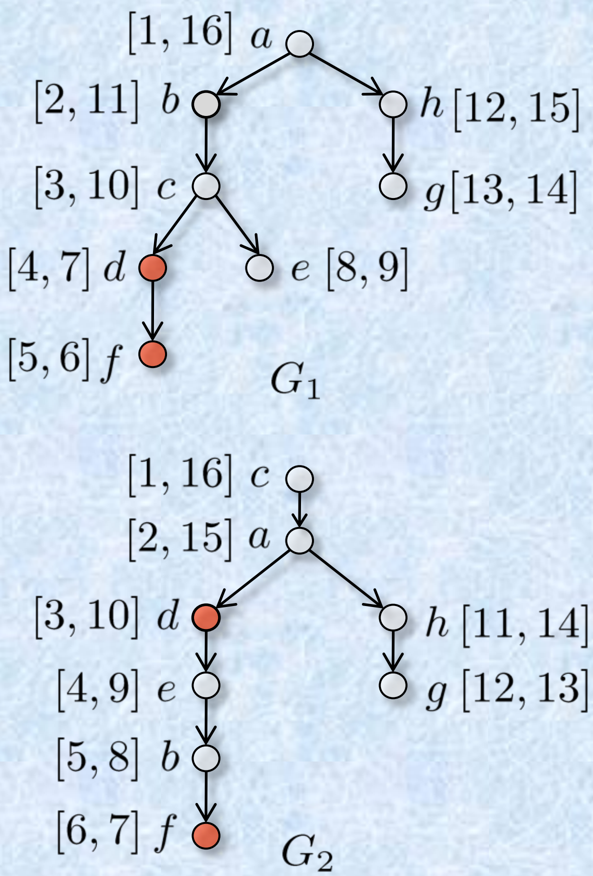
Join-Reachability Graph: Combinatorial Complexity

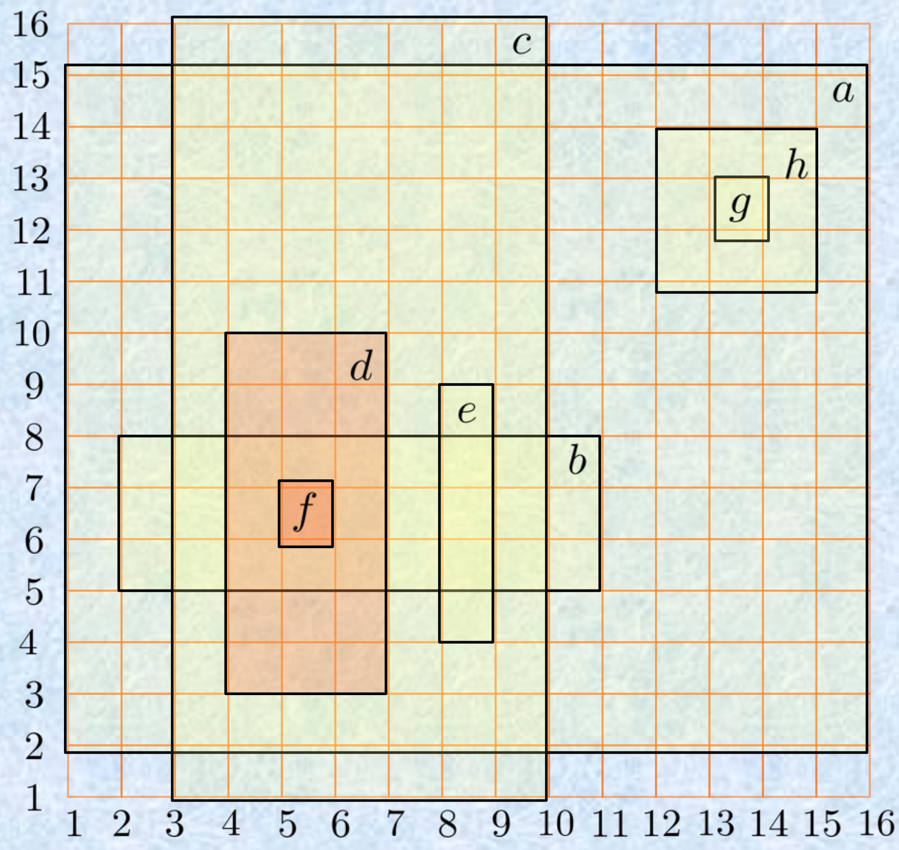
Construction of a compact join-reachability graph $\mathcal{J}(\mathcal{G})$ for trees

Each vertex $v \in V$ is mapped to a rectangle $R(v) = I_1(v) \times I_2(v)$

$I_i(v) = [s_i(v), t_i(v)] =$ depth-first search interval of v in G_i

$G_1 =$ out-tree, $G_2 =$ out-tree : $u \rightsquigarrow_{\mathcal{J}} v \Leftrightarrow R(u) \supseteq R(v)$





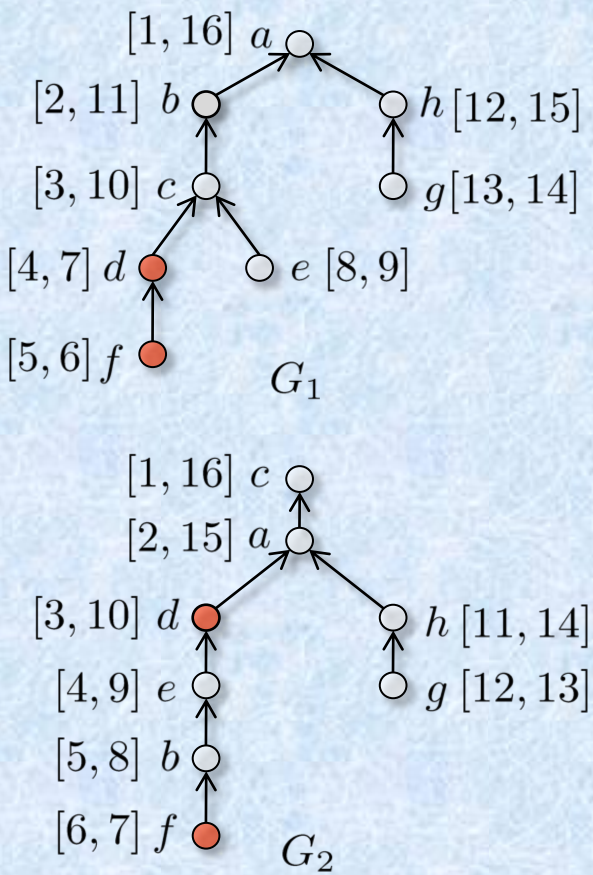
Join-Reachability Graph: Combinatorial Complexity

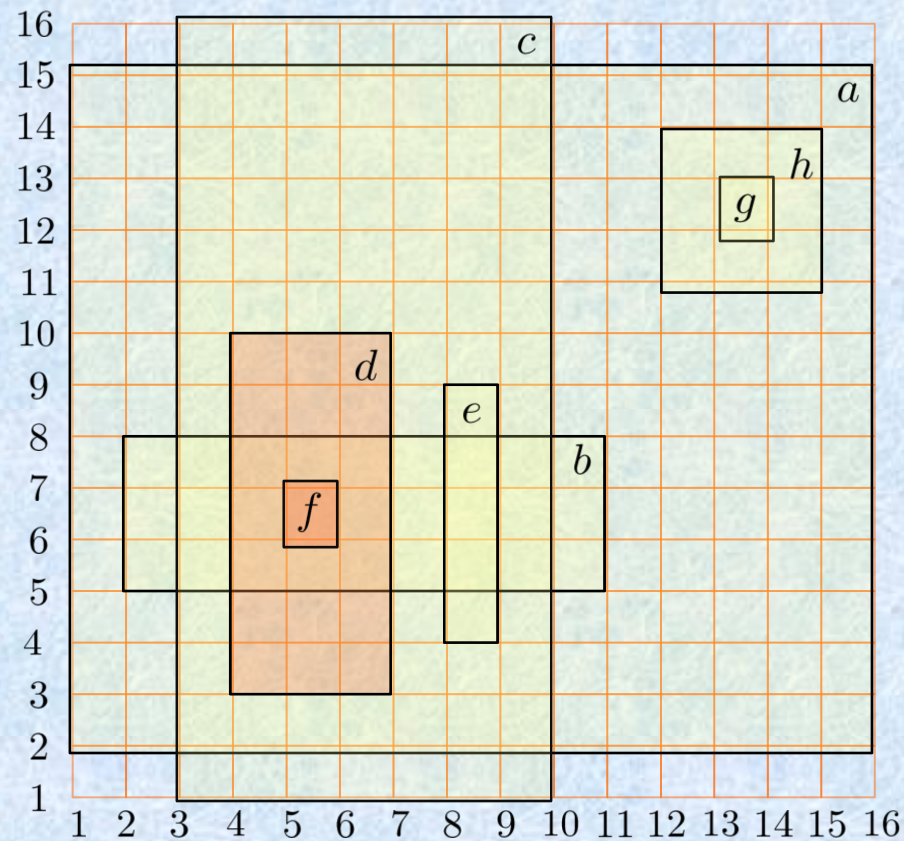
Construction of a compact join-reachability graph $\mathcal{J}(\mathcal{G})$ for trees

Each vertex $v \in V$ is mapped to a rectangle $R(v) = I_1(v) \times I_2(v)$

$I_i(v) = [s_i(v), t_i(v)] =$ depth-first search interval of v in G_i

$G_1 =$ in-tree, $G_2 =$ in-tree : $u \rightsquigarrow_{\mathcal{J}} v \Leftrightarrow R(u) \subseteq R(v)$





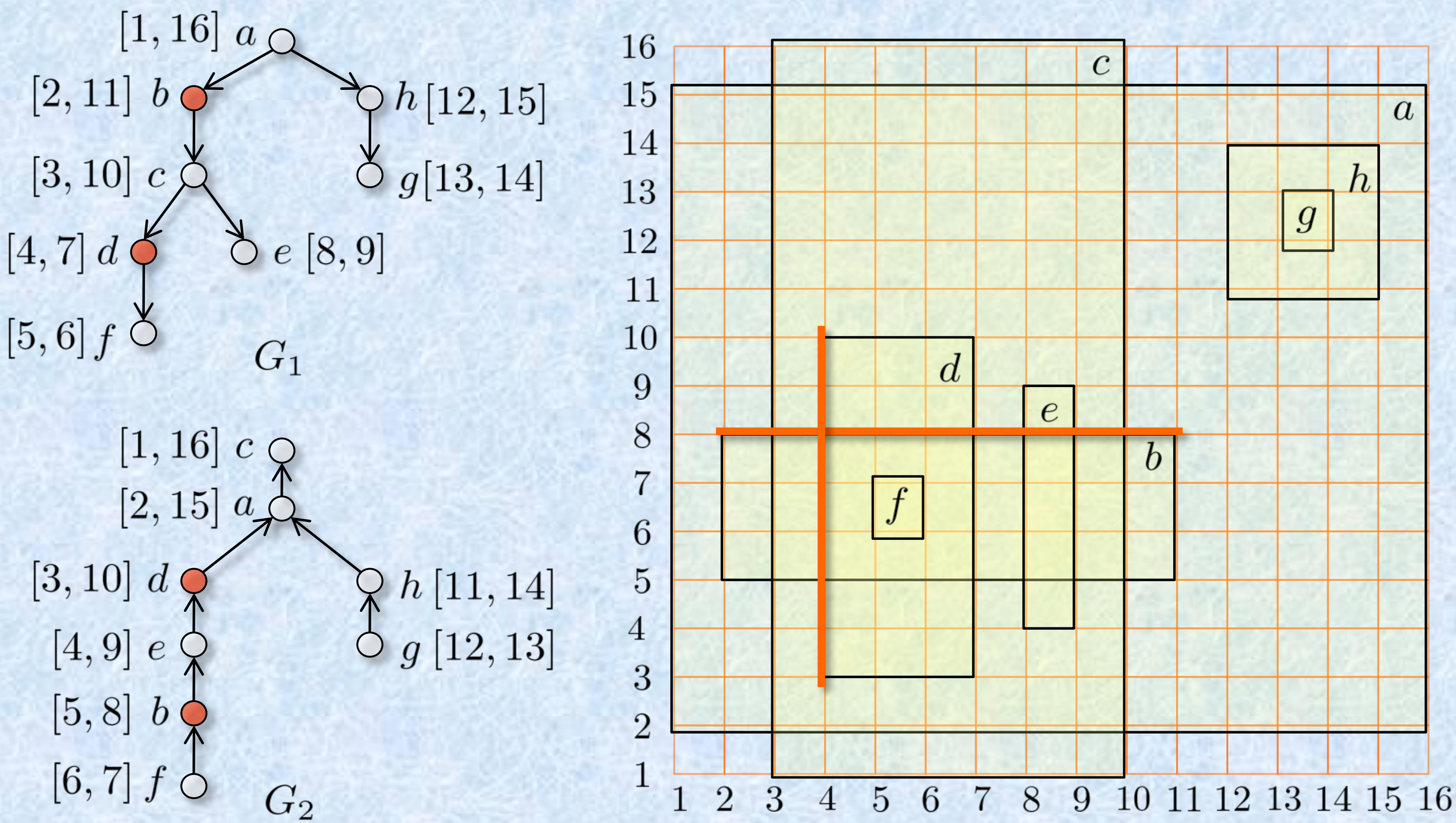
Join-Reachability Graph: Combinatorial Complexity

Construction of a compact join-reachability graph $\mathcal{J}(\mathcal{G})$ for trees

Each vertex $v \in V$ is mapped to a rectangle $R(v) = I_1(v) \times I_2(v)$

$I_i(v) = [s_i(v), t_i(v)] =$ depth-first search interval of v in G_i

$G_1 =$ out-tree, $G_2 =$ in-tree : $u \rightsquigarrow_{\mathcal{J}} v \Leftrightarrow (I_1(u) \times t_1(u)) \cap (s_1(v) \times I_2(v)) \neq \emptyset$



Join-Reachability Graph: Combinatorial Complexity

Given two digraphs G_1 and G_2 with n vertices, the following bounds on the size of the join-reachability graph $\mathcal{J}(\{G_1, G_2\})$ hold:

(a) $\Theta(n \log n)$ in the worst case when G_1 is an unoriented tree and G_2 is an unoriented dipath.

(b) $O(n \log^2 n)$ when both G_1 and G_2 are unoriented trees.

(c) $O(n \log^2 n)$ when G_1 is a planar digraph and G_2 is an unoriented dipath.

(d) $O(n \log^3 n)$ when both G_1 and G_2 are planar digraphs.

(e) $O(\kappa_1 n \log n)$ when G_1 is a digraph that can be covered with κ_1 vertex-disjoint dipaths and G_2 is an unoriented dipath.

(f) $O(\kappa_1 n \log^2 n)$ when G_1 is a digraph that can be covered with κ_1 vertex-disjoint dipaths and G_2 is a planar graph.

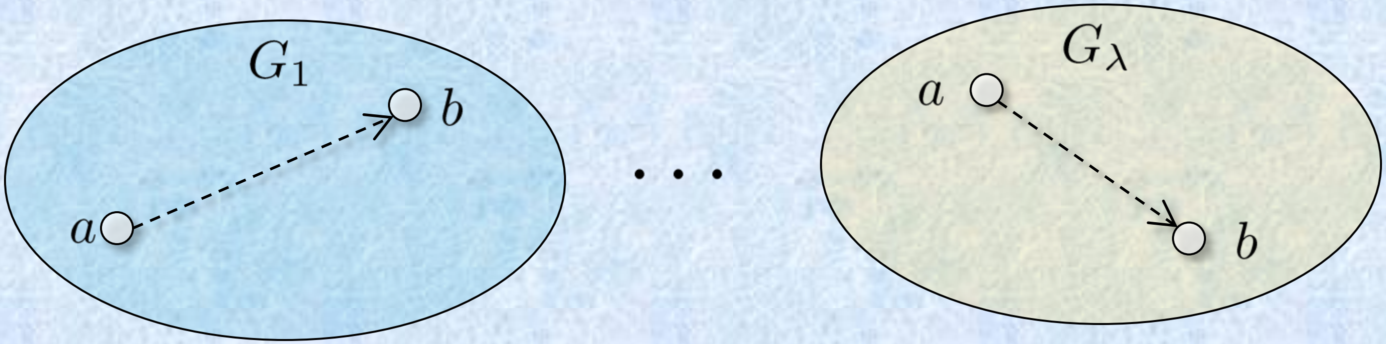
(g) $O(\kappa_1 \kappa_2 n \log n)$ when each G_i, $i = 1, 2$, is a digraph that can be covered with κ_i vertex-disjoint dipaths.

Outline

- Graph Reachability
- Join-Reachability Problems
  - Motivation
  - Preprocessing
    - Layer Decomposition
    - Removing Cycles
  - Join-Reachability Graph
    - Computational Complexity
    - Combinatorial Complexity
  - **Join-Reachability Data Structures**
- Concluding Remarks

Join-Reachability Data Structures

Collection of graphs $\mathcal{G} = \{G_1, G_2, \ldots, G_\lambda\}$



Join-Reachability Query :

Report all vertices that reach b in all graphs $G_i \in \mathcal{G}$

(Vertices a such that there is a $a \leadsto b$ path in all $G_i \in \mathcal{G}$)

Efficiency of a Data Structure: $\langle s(n), q(n,k) \rangle$

$s(n)$ storage space

$q(n,k)$ time to report k vertices

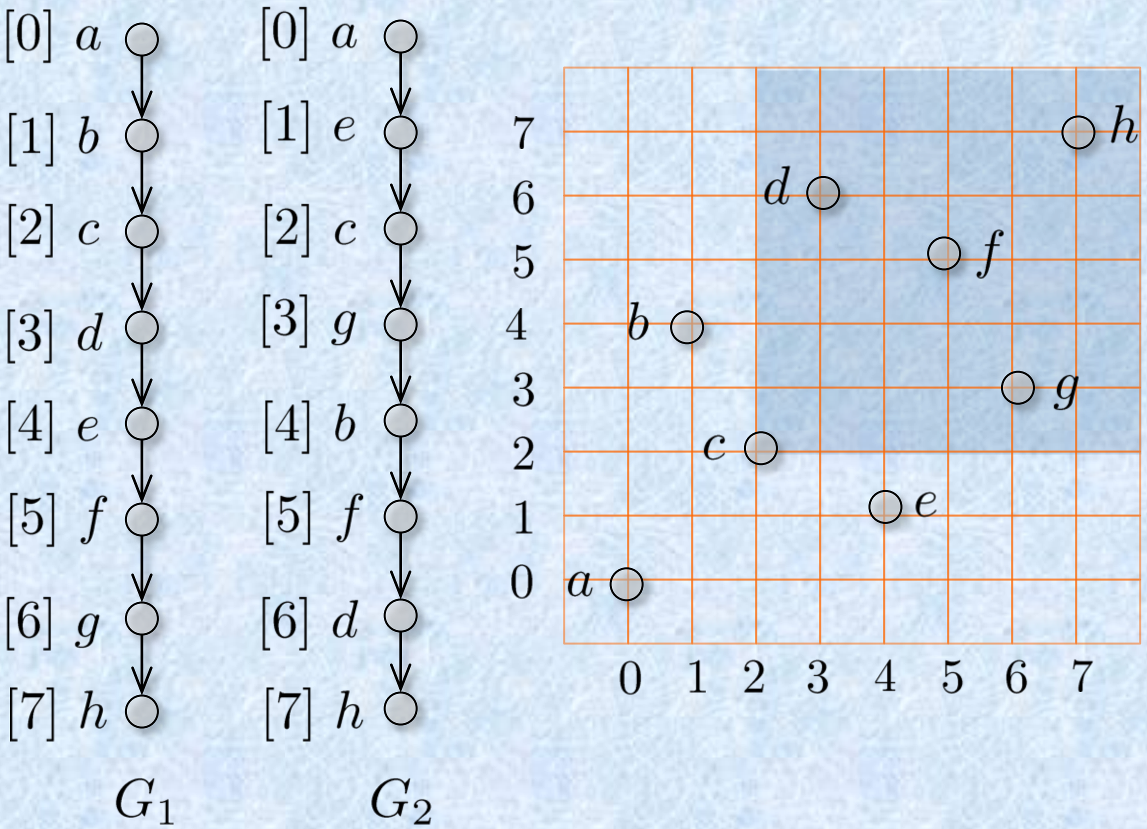
Join-Reachability Data Structures

Construction for paths

Each vertex $v \in V$ is mapped to a point $(x_1(v), x_2(v))$

$x_i(v) =$ number of vertices reachable from v in G_i

$$u \rightsquigarrow_{\mathcal{J}} v \Leftrightarrow (x_1(u), x_2(u)) \leq (x_1(v), x_2(v))$$



point-dominance problem

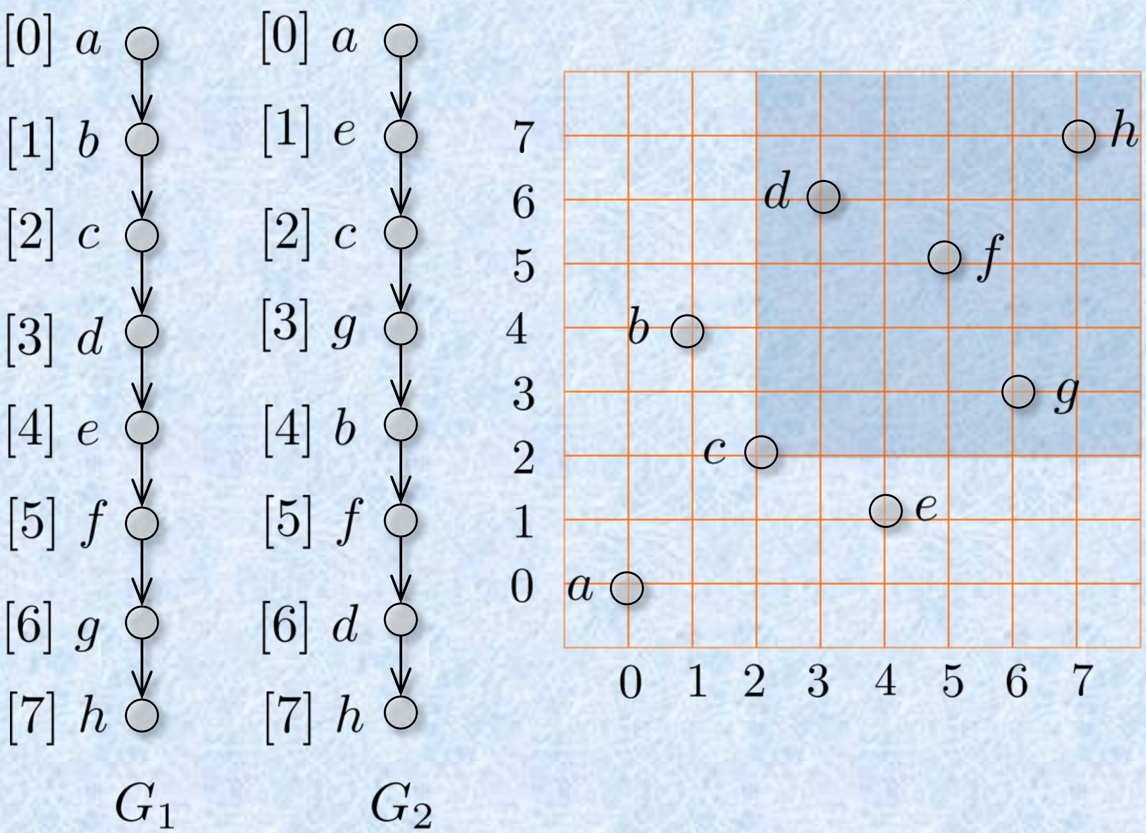
Join-Reachability Data Structures

Construction for paths

Each vertex $v \in V$ is mapped to a point $(x_1(v), x_2(v))$

$x_i(v) =$ number of vertices reachable from v in G_i

$$u \leadsto_{\mathcal{J}} v \Leftrightarrow (x_1(u), x_2(u)) \leq (x_1(v), x_2(v))$$



point-dominance problem

$\Rightarrow \langle n, k \rangle$ structure

(e.g., Cartesian trees [Gabow, Bentley and Tarjan '84])

Join-Reachability Data Structures

Given two digraphs G_1 and G_2 with n vertices we can construct join-reachability data structures with the following efficiency:

(a) $\langle n, k \rangle$ when G_1 is an unoriented tree and G_2 is an unoriented dipath.

(b) $\langle n, \log n + k \rangle$ when G_1 is an out-tree and G_2 is an unoriented tree.

(c) $\langle n \log^{\varepsilon} n, \log\log n + k \rangle$ (for any constant $\varepsilon > 0$), when G_1 and G_2 are unoriented trees.

(d) $\langle n \log n, k \log n \rangle$ when G_1 is planar digraph and G_2 is an unoriented tree.

(e) $\langle n \log^2 n, k \log^2 n \rangle$ when both G_1 and G_2 are planar digraphs.

(f) $\langle n\kappa_1, k \rangle$ when G_1 is a general digraph that can be covered with κ_1 vertex-disjoint dipaths and G_2 is an unoriented tree.

(g) $\langle n(\kappa_1 + \log n), k\kappa_1 \log n \rangle$ or $\langle n\kappa_1 \log n, k \log n \rangle$ when G_1 is a general digraph that can be covered with κ_1 vertex-disjoint dipaths and G_2 is planar digraph.

(h) $\langle n(\kappa_1 + \kappa_2), \kappa_1\kappa_2 + k \rangle$ or $\langle n\kappa_1\kappa_2, k \rangle$ when each G_i, $i = 1, 2$, is a digraph that can be covered with κ_i vertex-disjoint dipaths.

Concluding Remarks

More Problems:

- Complexity of computing smallest $\mathcal{J}(\mathcal{G})$ for simple graph classes
- Approximate smallest $\mathcal{J}(\mathcal{G})$ for simple graph classes
- Data structures supporting more general join-reachability queries

  e.g., report all a such that $a \rightsquigarrow_{G_1} b$ and $a \rightsquigarrow_{G_2} c$

Thank You!