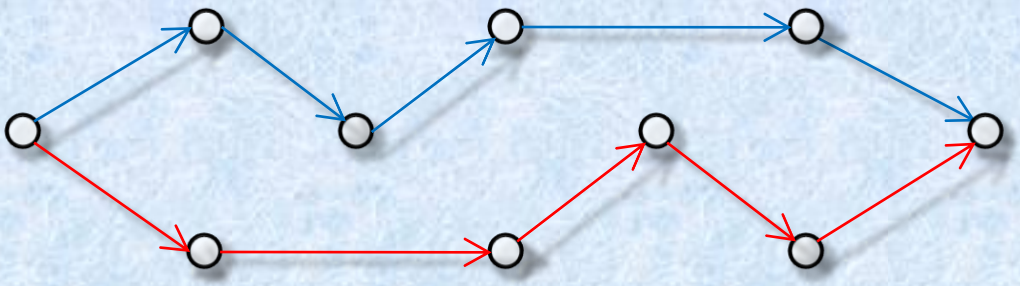
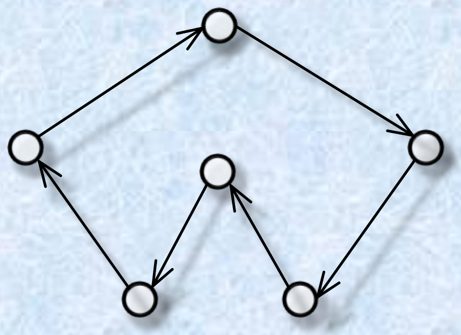
Testing 2-Vertex-Connectivity
and
Computing Pairs of Vertex-Disjoint Paths



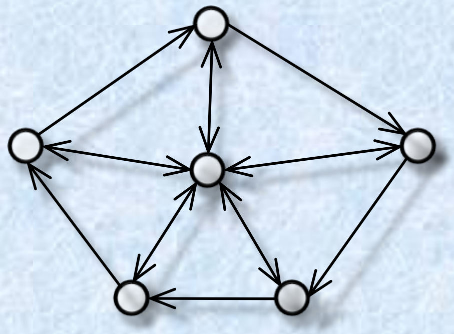
Loukas Georgiadis
University of Western Macedonia, Greece

Vertex Connectivity



Strongly connected digraph $G = (V, A)$

contains an s-t path for any pair $s, t \in V$



k-vertex connected digraph $G = (V, A)$

the removal of any subset $X \subseteq V, |X| \leq k-1$

leaves the graph strongly connected

Basic problems :

- Compute vertex connectivity (largest k such that G is k-vertex connected)
- Test if the given digraph is k-vertex connected

Vertex Connectivity

Basic problems :

- Compute vertex connectivity $\kappa =$ largest k such that G is k-vertex connected

  $O((n + \min\{\kappa^{5/2}, \kappa n^{3/4}\})m)$ [Gabow 2006]

- Test if the given digraph is k-vertex connected

  $O(\min\{k^3 + n, kn\}m)$
  $O(mn)$ with error probability $1/2$
  [Henzinger, Rao and Gabow 2000]

  $O((M(n) + nM(k)) \log n)$ with error probability $1/n$
  $O((M(n) + nM(k))k)$ expected
  [Cheriyan and Reif 1994]

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

$M(n)$ = matrix multiplication time $(= O(n^{2.376}))$

Vertex Connectivity

Undirected graphs: $O(m+n)$ algorithms for testing

$k=2$ [Tarjan 1972]

$k=3$ [Hopcroft and Tarjan 1973]

Directed graphs: $O(m+n)$ algorithm for testing $k=2$?

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

Results

$O(m+n)$-time algorithm for testing 2-vertex connectivity

$\Downarrow$

$O(n)$-space data structure :

compute two vertex-disjoint s-t paths in $O(\log^2 n)$ time

report the two paths, P and Q, in $O(|P|+|Q|)$ time

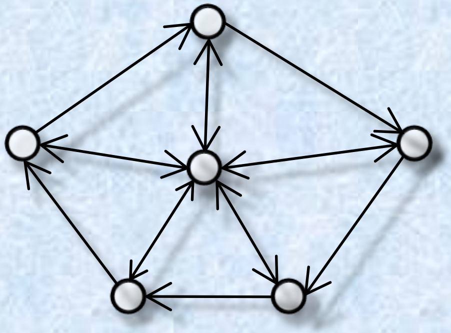
$n = |V|, m = |A|$

Vertex Connectivity

k-vertex connected digraph $G = (V, A)$

the removal of any subset $X \subseteq V, |X| \leq k - 1$

leaves the graph strongly connected



From Menger's theorem :

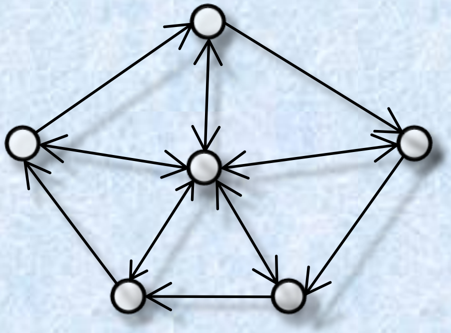
G is k-vertex connected $\Longleftrightarrow$ G contains k vertex-disjoint s-t paths for any $s, t \in V$

2-Vertex Connectivity

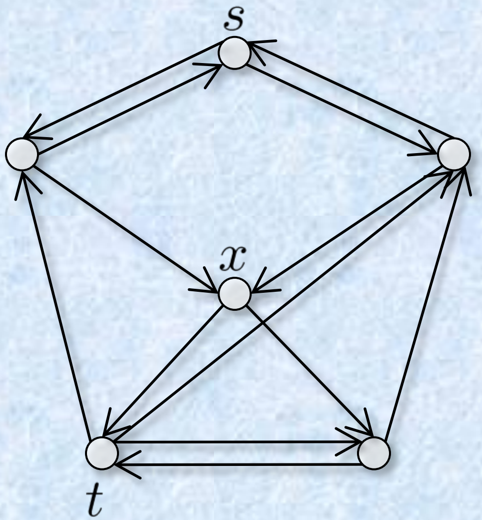
2-vertex connected digraph $G = (V, A)$

the removal of at most one vertex

leaves the graph strongly connected



If G is strongly connected but not 2-vertex connected :



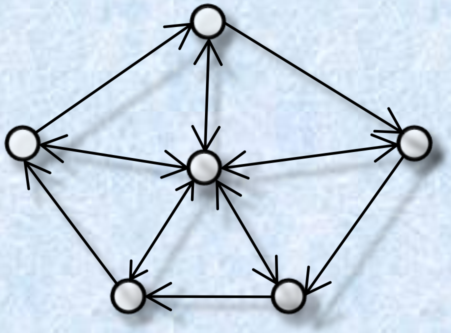
There are $s, t \in V$ such that all s-t paths contain a common vertex $x \neq s, t$

2-Vertex Connectivity

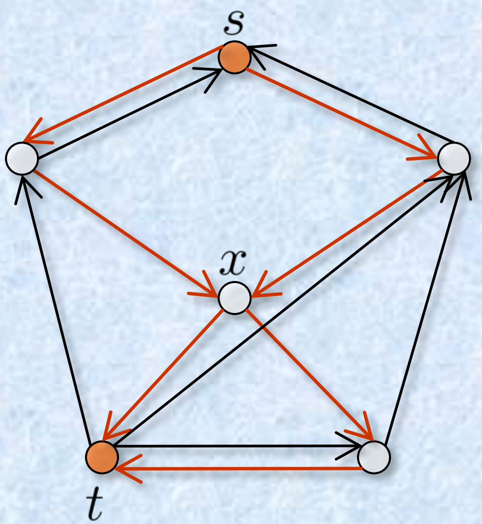
2-vertex connected digraph $G = (V, A)$

the removal of at most one vertex

leaves the graph strongly connected



If G is strongly connected but not 2-vertex connected :



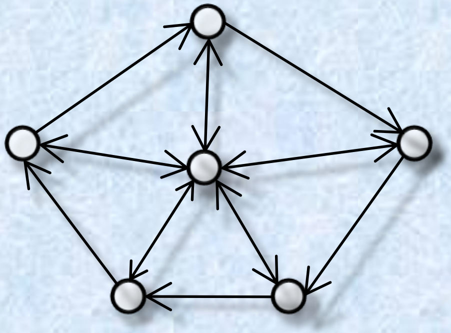
There are $s, t \in V$ such that all s-t paths contain a common vertex $x \neq s, t$

2-Vertex Connectivity

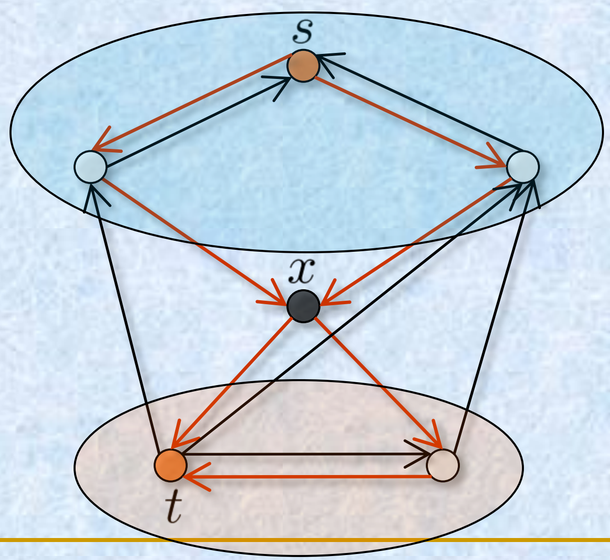
2-vertex connected digraph $G = (V, A)$

the removal of at most one vertex

leaves the graph strongly connected



If G is strongly connected but not 2-vertex connected :

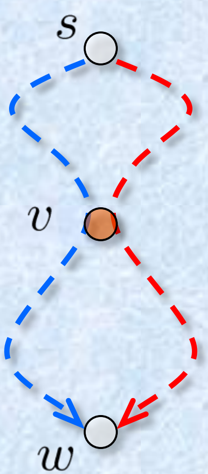


There are $s, t \in V$ such that all s-t paths contain a common vertex $x \neq s, t$

Flowgraphs and Dominators

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

v dominates w if every path from s to w includes v



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

Trivial dominators : $s, w \in dom(w)$

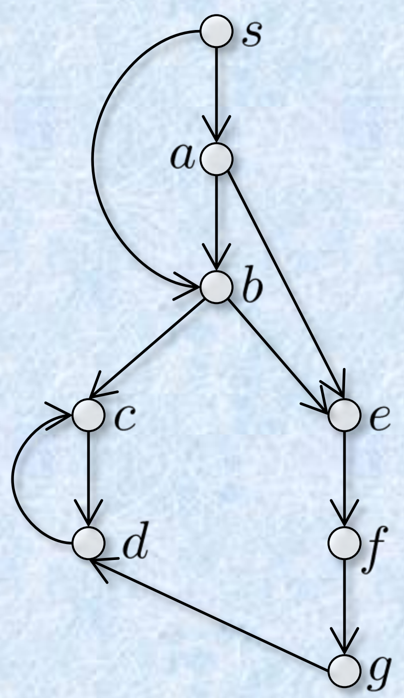
Application areas : Program optimization, VLSI testing, theoretical biology, distributed systems, constraint programming

Flowgraphs and Dominators

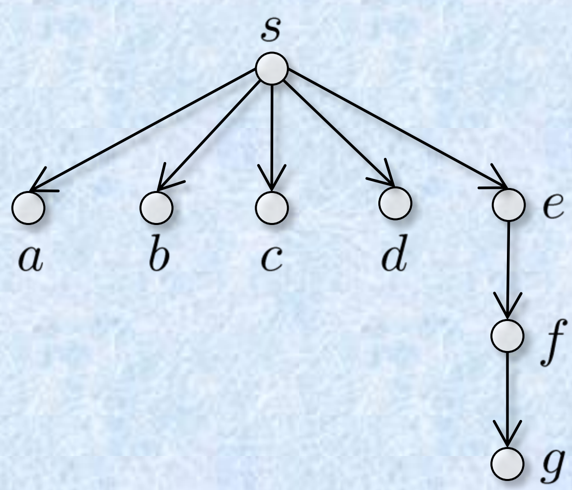
Flowgraph $G(s) = (V, A, s)$: all vertices are reachable from start vertex s

v dominates w if every path from s to w includes v

$G(s)$



dominator tree of $G(s)$



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

$O(m+n)$ algorithms:

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

[Buchsbaum, Kaplan, Rogers, and Westbrook '04]

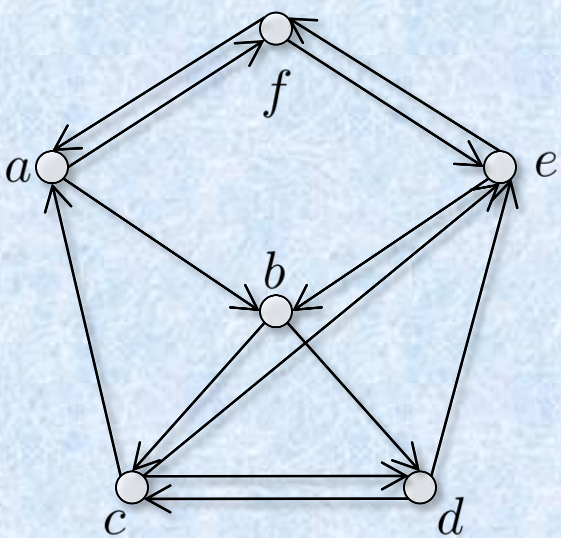
[G., and Tarjan '04]

2-Vertex Connectivity

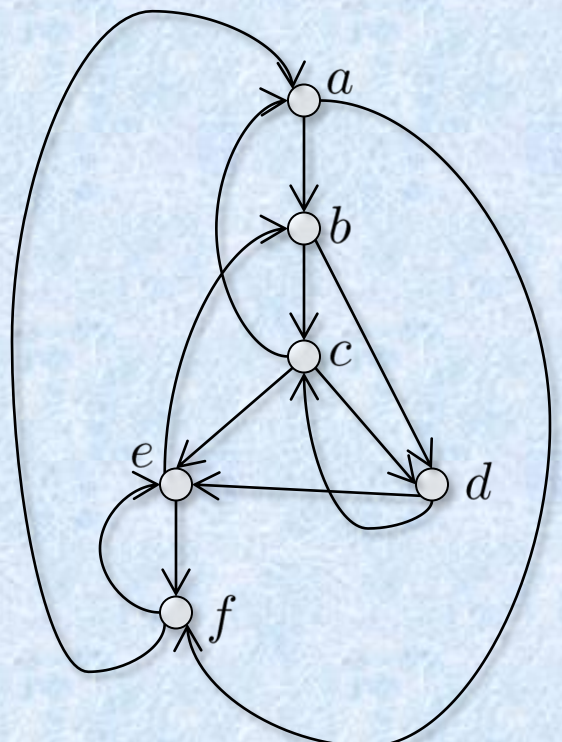
Main Idea : Compute dominators in $G(s)$ and $G^r(s)$ for arbitrary $s \in V$

G^r : has reversed arcs

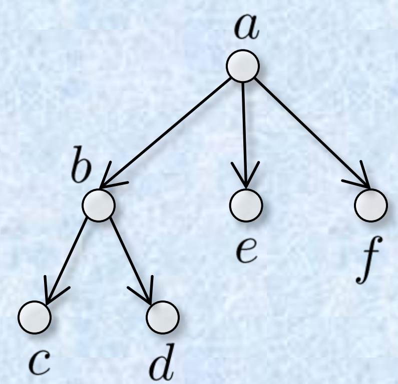
G



$G(a)$



dominator tree of $G(a)$

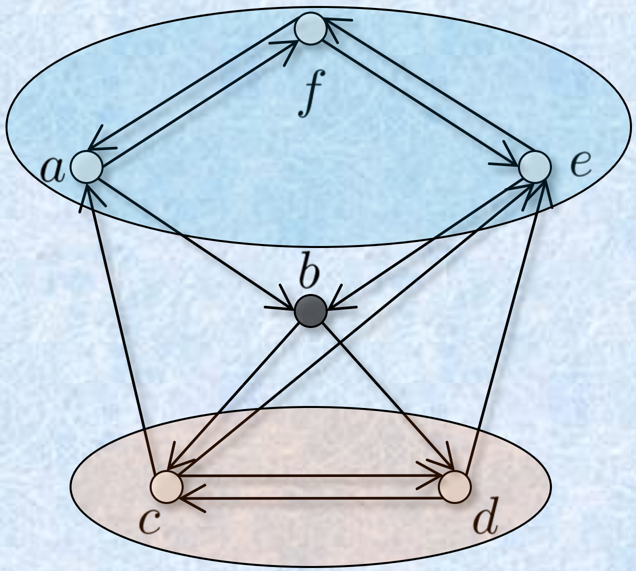


2-Vertex Connectivity

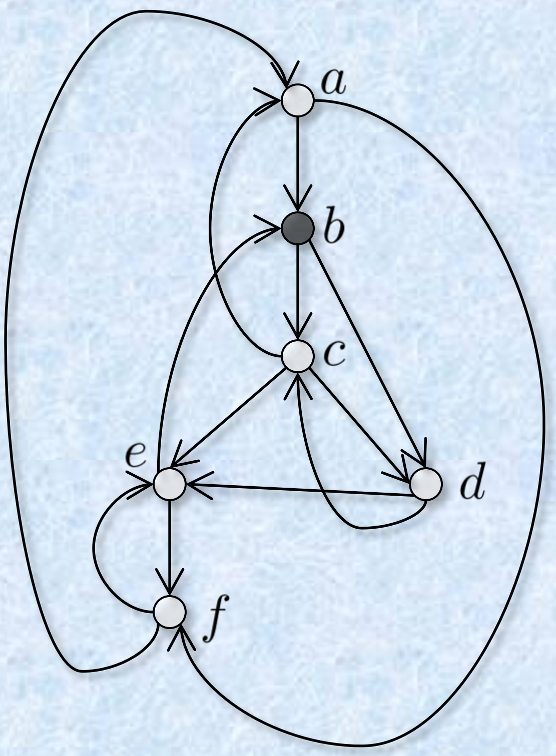
Main Idea : Compute dominators in $G(s)$ and $G^r(s)$ for arbitrary $s \in V$

G^r : has reversed arcs

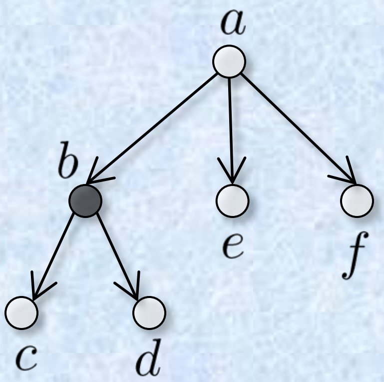
G



$G(a)$



dominator tree of $G(a)$

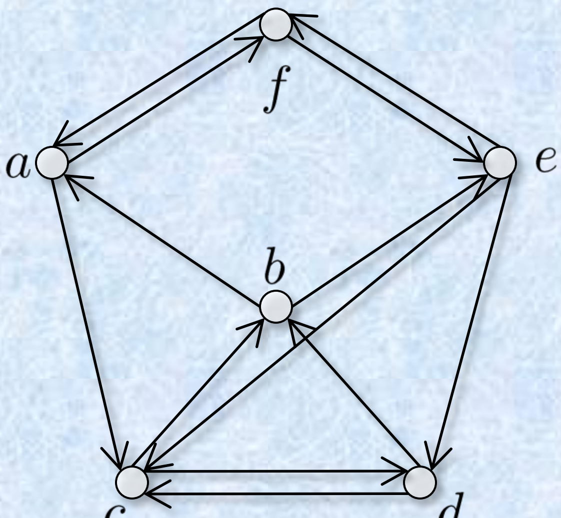


2-Vertex Connectivity

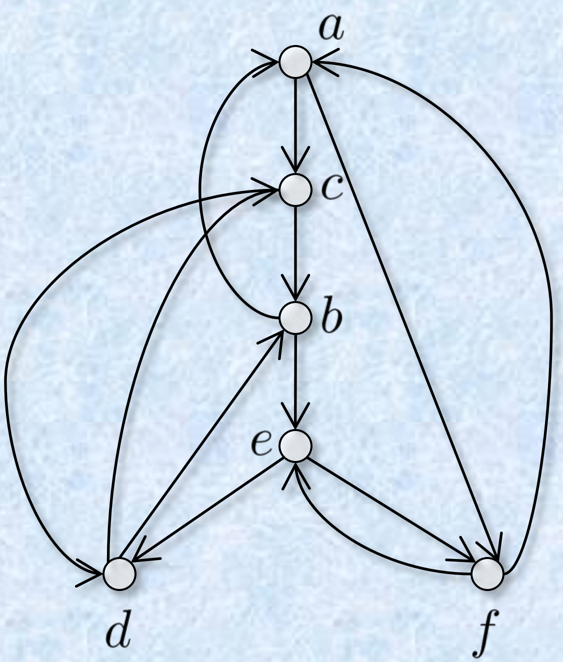
Main Idea : Compute dominators in $G(s)$ and $G^r(s)$ for arbitrary $s \in V$

G^r : has reversed arcs

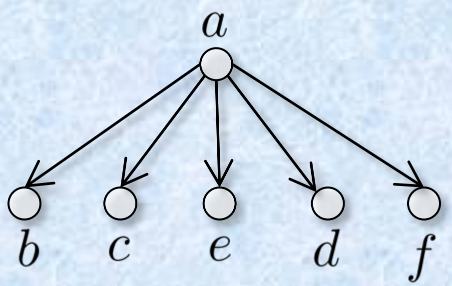
G



$G(a)$



dominator tree of $G(a)$

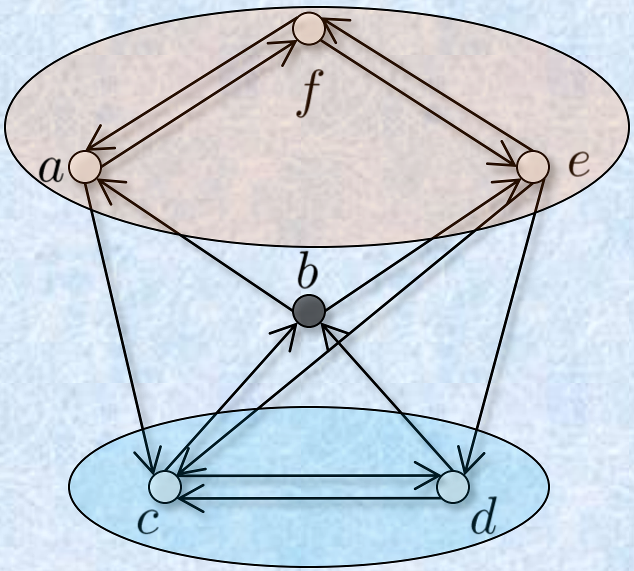


2-Vertex Connectivity

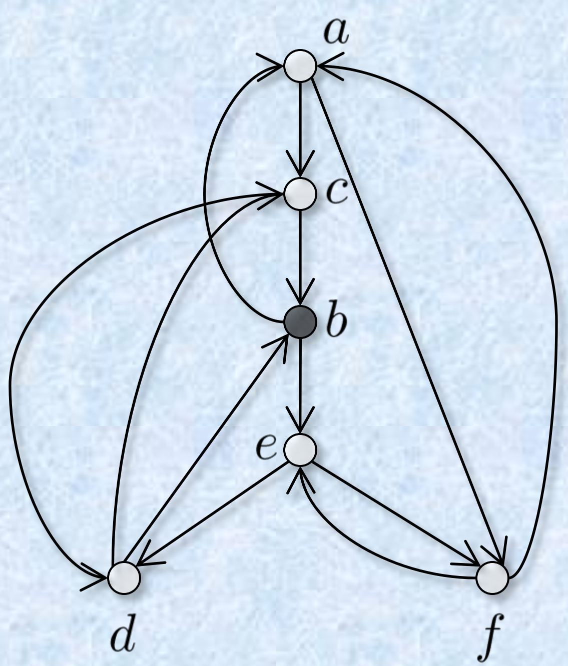
Main Idea : Compute dominators in $G(s)$ and $G^r(s)$ for arbitrary $s \in V$

G^r : has reversed arcs

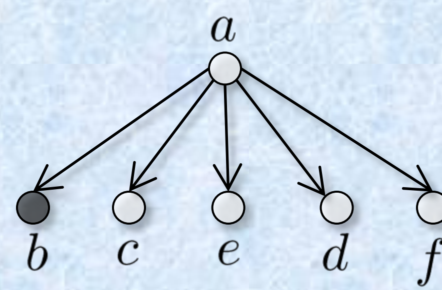
G



$G(a)$



dominator tree of $G(a)$

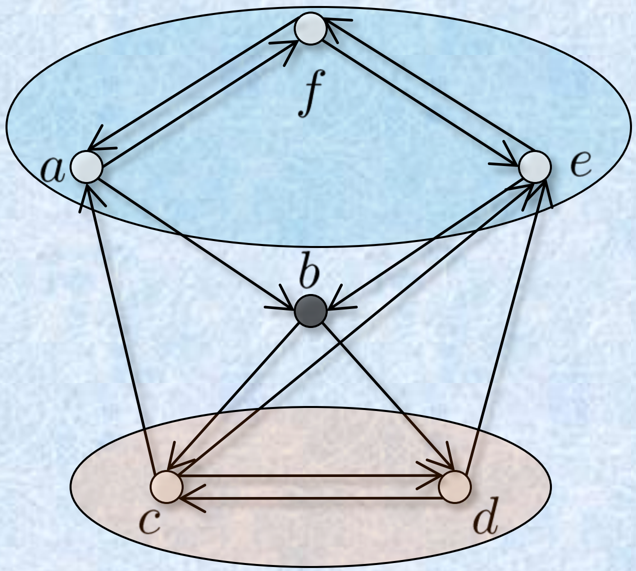


2-Vertex Connectivity

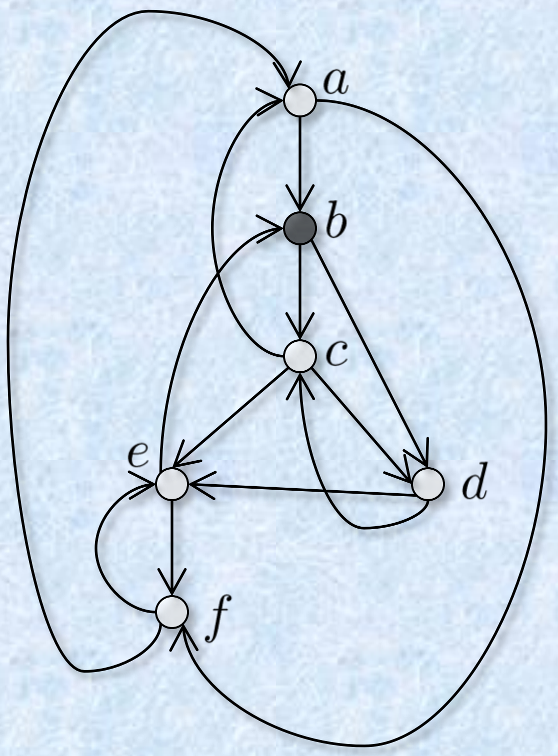
Main Idea : Compute dominators in $G(s)$ and $G^r(s)$ for arbitrary $s \in V$

G^r : has reversed arcs

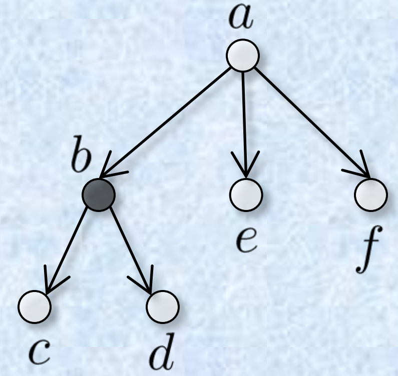
G^r



$G^r(a)$



dominator tree of $G^r(a)$



2-Vertex Connectivity

Algorithm

pick any vertex $a \in V$ of the input digraph $G = (V, A)$

if $G(a)$ has nontrivial dominators then return FALSE

if $G^r(a)$ has nontrivial dominators then return FALSE

if $G - a$ is strongly connected then return TRUE

2-Vertex Connectivity

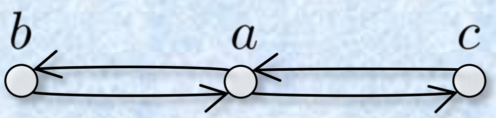
Algorithm

pick any vertex $a \in V$ of the input digraph $G = (V, A)$

if $G(a)$ has nontrivial dominators then return FALSE

if $G^r(a)$ has nontrivial dominators then return FALSE

if $G - a$ is strongly connected then return TRUE



2-Vertex Connectivity

Algorithm

pick any vertex $a \in V$ of the input digraph $G = (V, A)$

if $G(a)$ has nontrivial dominators then return FALSE

if $G^r(a)$ has nontrivial dominators then return FALSE

if $G - a$ is strongly connected then return TRUE

Verification of trivial dominators: $O(m + n)$ time [G. and Tarjan 2005]

Total running time: $O(m + n)$

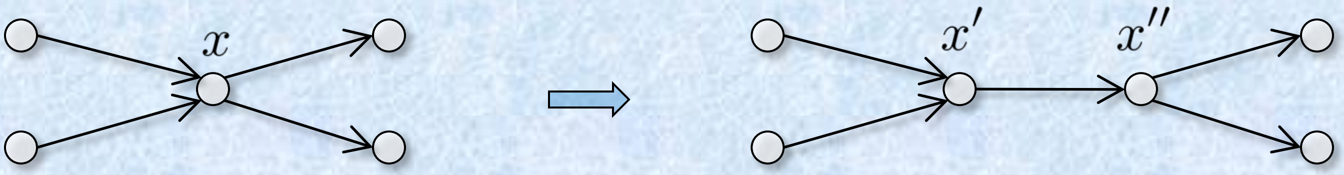
Vertex-Disjoint s-t Paths

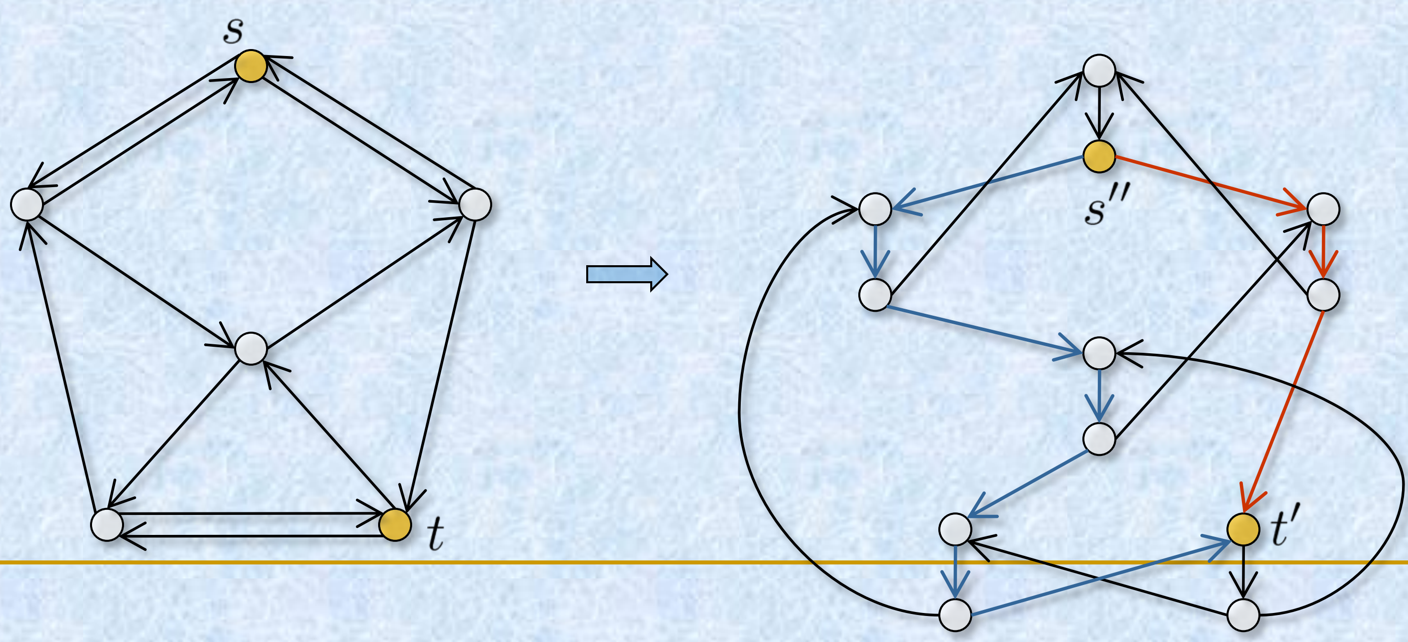
Given a digraph $G = (V, A)$ how fast can we compute a pair of vertex-disjoint s-t paths?

Vertex-Disjoint s-t Paths

Given a digraph $G = (V, A)$ how fast can we compute a pair of vertex-disjoint s-t paths?

$O(m + n)$ time : “vertex-splitting” + “flow augmentation”





Vertex-Disjoint s-t Paths

Given a digraph $G = (V, A)$ how fast can we compute a pair of vertex-disjoint s-t paths?

$O(m + n)$ time : “vertex-splitting” + “flow augmentation”

We can get a more efficient solution when G is 2-vertex connected

- Use a 2-vertex connected spanning subgraph of G with $O(n)$ arcs

  [Cheriyan and Thurimella 2000] : $1 + 1/k$ approximation of the minimum k-vertex connected spanning subgraph in $O(km^2)$ time

Vertex-Disjoint s-t Paths

Given a digraph $G = (V, A)$ how fast can we compute a pair of vertex-disjoint s-t paths?

$O(m + n)$ time : “vertex-splitting” + “flow augmentation”

We can get a more efficient solution when G is 2-vertex connected

- Use a 2-vertex connected spanning subgraph of G with $O(n)$ arcs

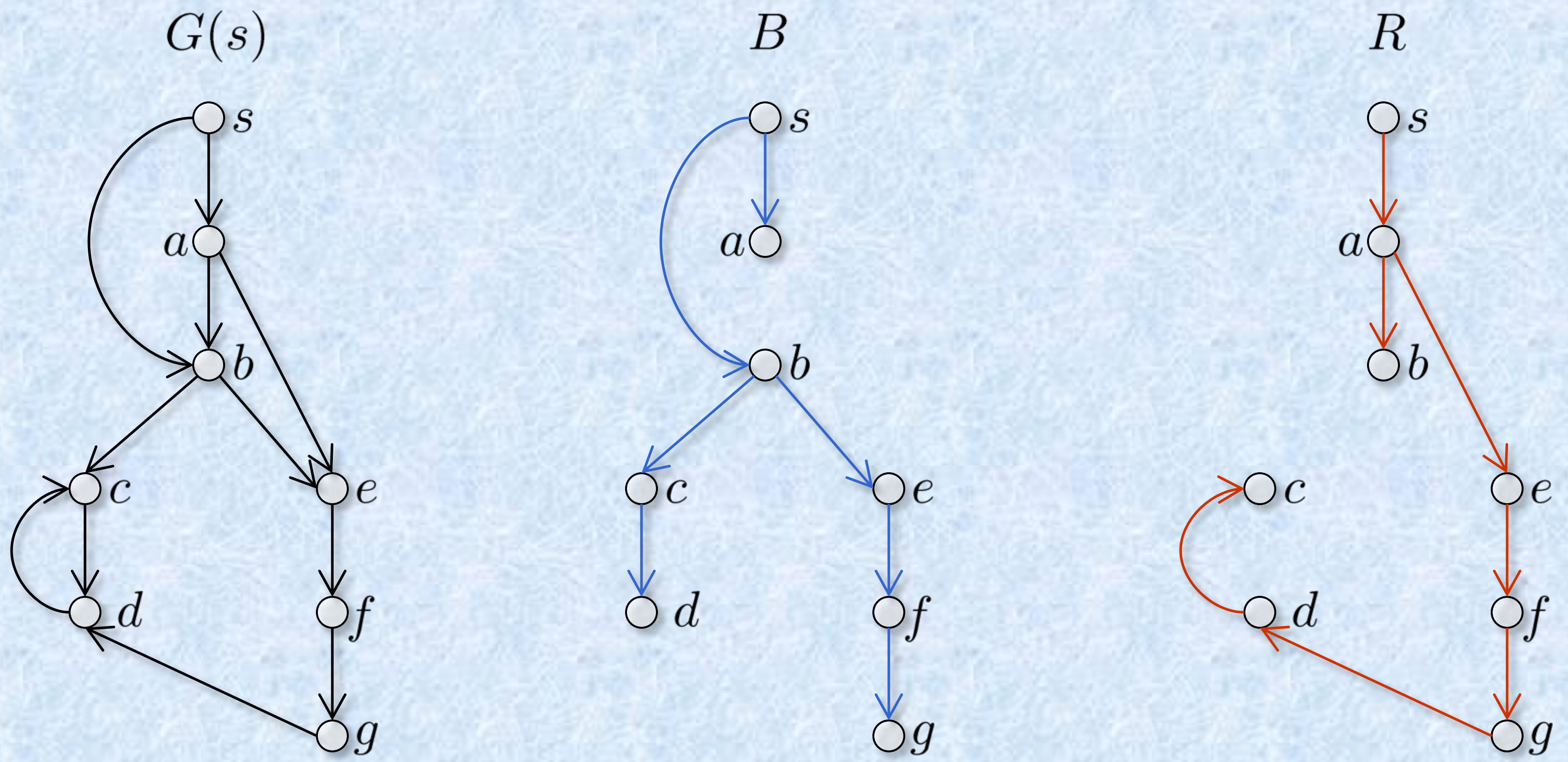
  [Cheriyan and Thurimella 2000] : $1 + 1/k$ approximation of the minimum k-vertex connected spanning subgraph in $O(km^2)$ time

- Use pairs of independent trees

Vertex-Disjoint s-t Paths

Theorem [G. and Tarjan 2005] : Any flowgraph $G(s) = (V, A, s)$ has two spanning trees, B and R, such that for any $v \in V$

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

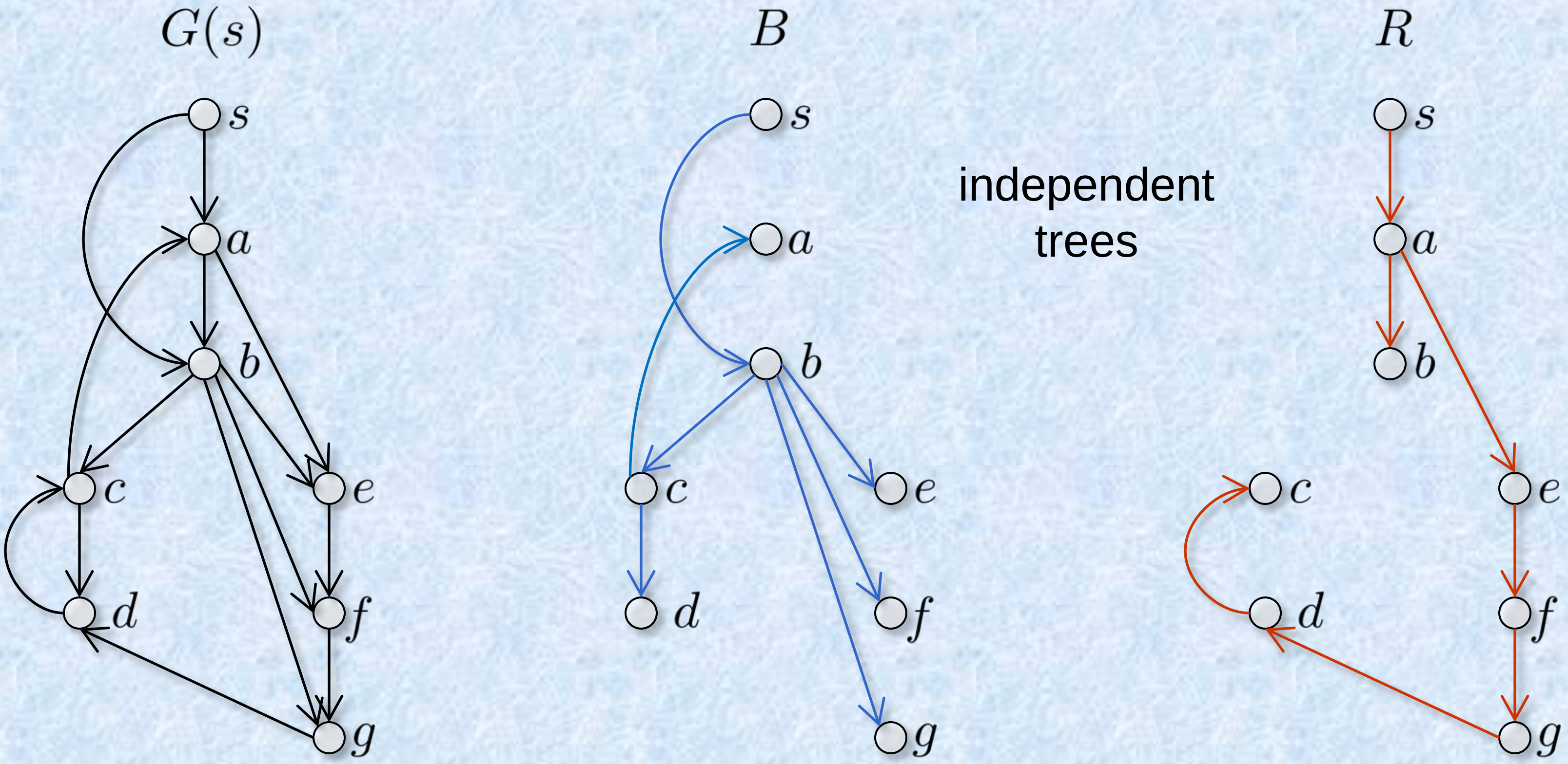


the two trees can be computed in linear time

Vertex-Disjoint s-t Paths

Corollary : If $G(s)$ has trivial dominators only then for any $v \in V$

$$B(s,v) \cap R(s,v) = \emptyset$$



independent trees

the two trees can be computed in linear time

Vertex-Disjoint s-t Paths

2-Vertex Connectivity Algorithm

pick any vertex $a \in V$ of the input digraph $G = (V, A)$

if $G(a)$ has nontrivial dominators then return FALSE

if $G^r(a)$ has nontrivial dominators then return FALSE

if $G - a$ is strongly connected then return TRUE

Implies a linear-time algorithm for computing a 2-vertex connected spanning subgraph with $\leq 6n$ arcs $\Rightarrow$ 3-approximation

Vertex-Disjoint s-t Paths

2-Vertex Connectivity Algorithm

pick any vertex $a \in V$ of the input digraph $G = (V, A)$

if $G(a)$ has nontrivial dominators then return FALSE

if $G^r(a)$ has nontrivial dominators then return FALSE

if $G - a$ is strongly connected then return TRUE

Corollary : A digraph $G = (V, A)$ is 2-vertex connected if and only if for two arbitrary vertices $a, b \in V$ $(a \neq b)$ the flowgraphs $G(a), G^r(a), G(b)$ and $G^r(b)$ have trivial dominators only.

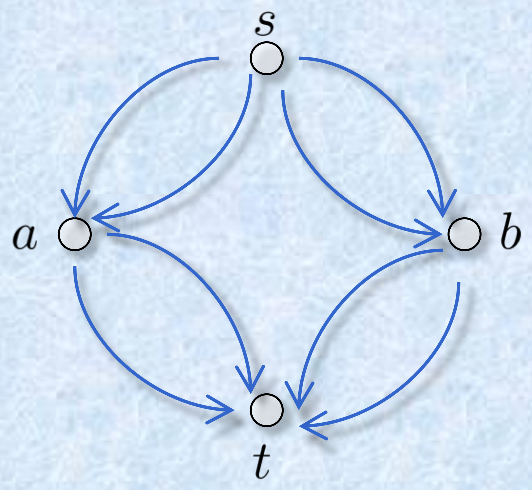
Main Idea : Use pairs of independent trees rooted at a and b .

Vertex-Disjoint s-t Paths

Corollary : A digraph $G = (V, A)$ is 2-vertex connected if and only if for two arbitrary vertices $a, b \in V$ $(a \neq b)$ the flowgraphs $G(a), G^r(a), G(b)$ and $G^r(b)$ have trivial dominators only.

We use a pair of independent spanning trees for each of the flowgraphs

$$G(a), G^r(a), G(b), G^r(b)$$



2-Vertex Connectivity

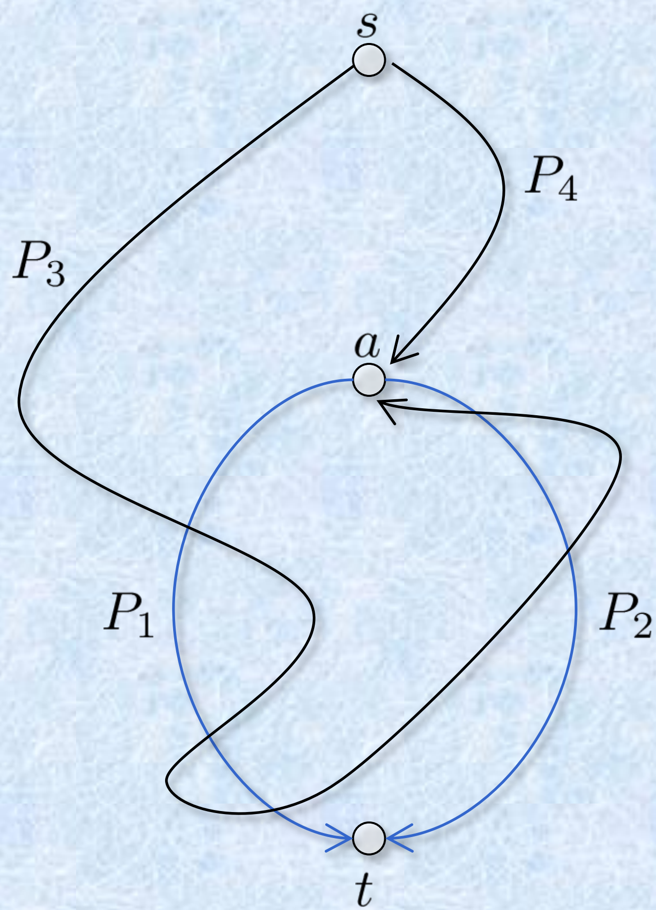
P_1, P_2 : vertex-disjoint a-t paths

P_3, P_4 : vertex-disjoint s-a paths

Suppose

$$P_3[s,a) \cap \big(P_1(a,t] \cup P_2(a,t]\big) \neq \emptyset$$

$$P_4(s,a) \cap \big(P_1(a,t) \cup P_2(a,t)\big) = \emptyset$$



2-Vertex Connectivity

P_1, P_2 : vertex-disjoint a-t paths

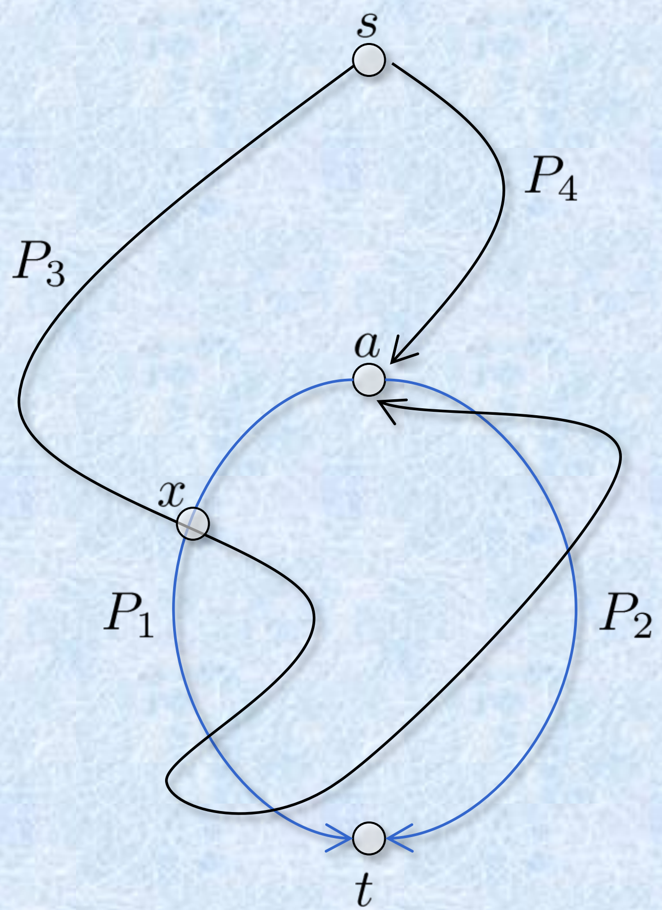
P_3, P_4 : vertex-disjoint s-a paths

Suppose

$$P_3[s,a) \cap \big(P_1(a,t] \cup P_2(a,t]\big) \neq \emptyset$$

$$P_4(s,a) \cap \big(P_1(a,t) \cup P_2(a,t)\big) = \emptyset$$

Let x be the first vertex on $P_3[s,a]$ such that $x \in \big(P_1(a,t] \cup P_2(a,t]\big)$.



2-Vertex Connectivity

P_1, P_2 : vertex-disjoint a-t paths

P_3, P_4 : vertex-disjoint s-a paths

Suppose

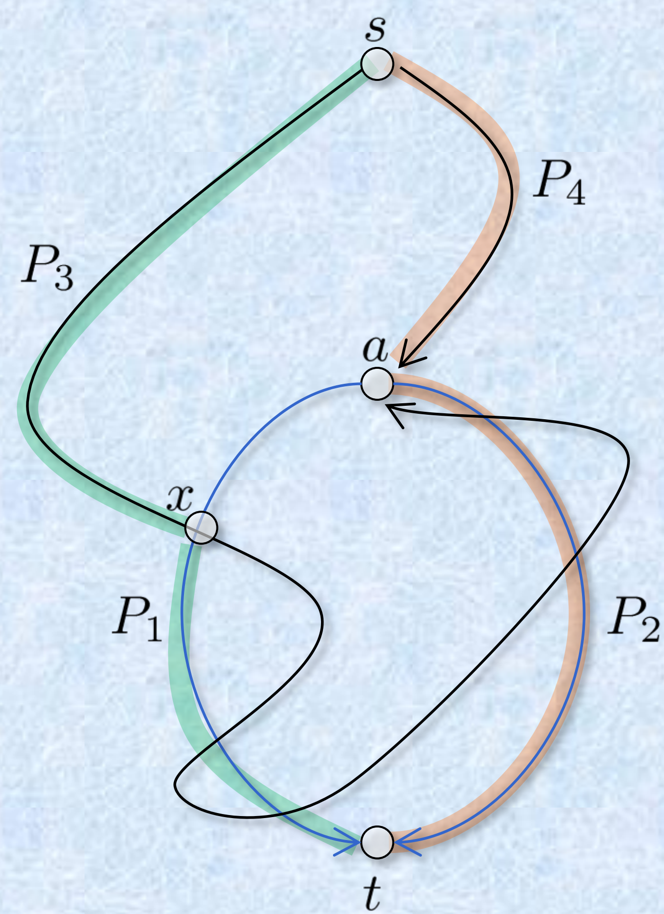
$$P_3[s,a) \cap \big(P_1(a,t] \cup P_2(a,t]\big) \neq \emptyset$$

$$P_4(s,a) \cap \big(P_1(a,t) \cup P_2(a,t)\big) = \emptyset$$

Let x be the first vertex on $P_3[s,a]$ such that $x \in \big(P_1(a,t] \cup P_2(a,t]\big)$.

Consider $x \in P_1(a,t] \Rightarrow$

$P_3[s,x] \cdot P_1[x,t]$ and $P_4[s,a] \cdot P_2[a,t]$ are vertex-disjoint s-t paths



2-Vertex Connectivity

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

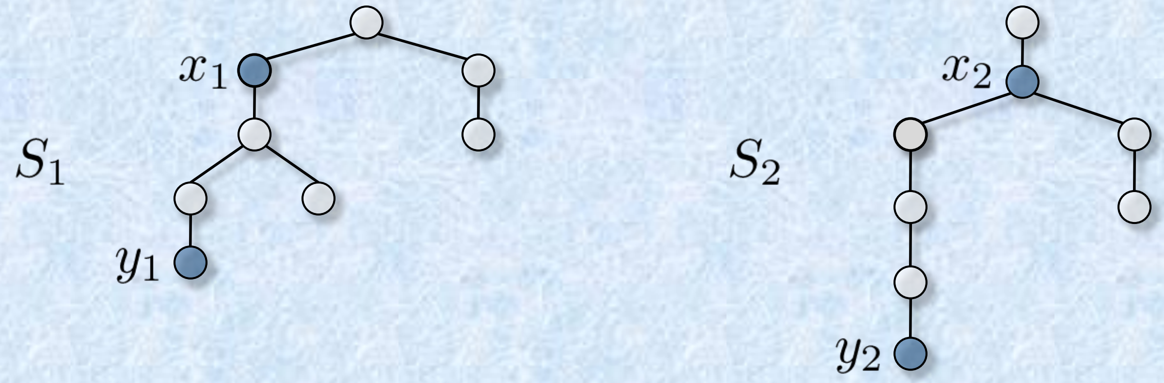
(i) Test if $S_1[x_1, y_1]$ contains x_2.

(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 $S_1[x_1, y_1]$.



2-Vertex Connectivity

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 $S_1(x_1, y_1]$.

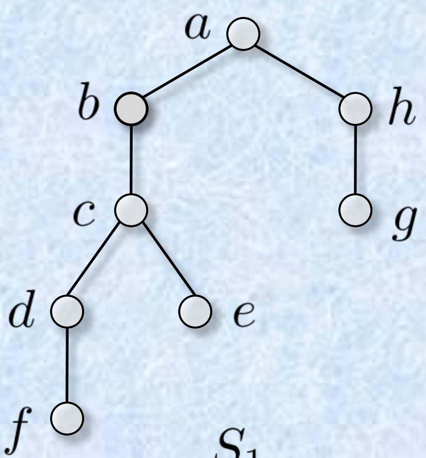
(iii) Test if $S_1[x_1, y_1]$ and $S_2[x_2, y_2]$ contain a common vertex.

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

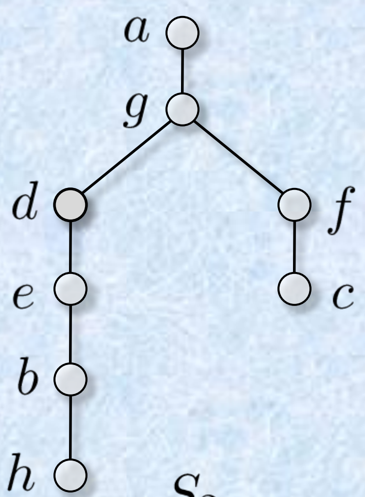
(v) Find the highest ancestor of y_2 in $S_2[x_2, y_2]$ that is contained in $S_1[x_1, y_1]$.

- A query uses a constant number of these operations.
- We give an $O(n)$- space data structure with $O(\log^2 n)$ time per operation.

2-Vertex Connectivity

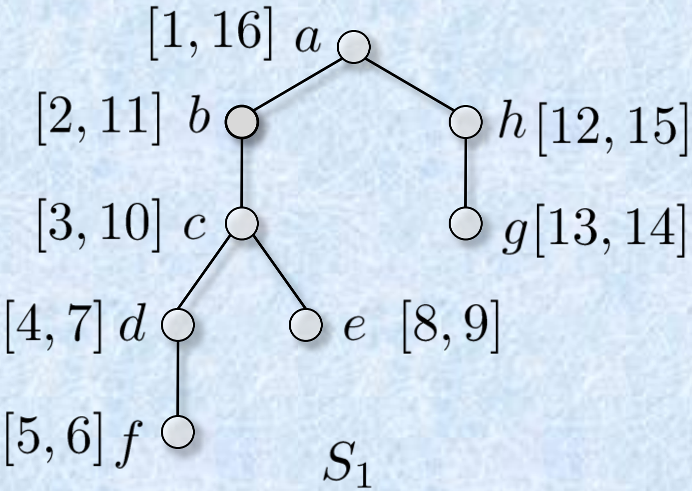


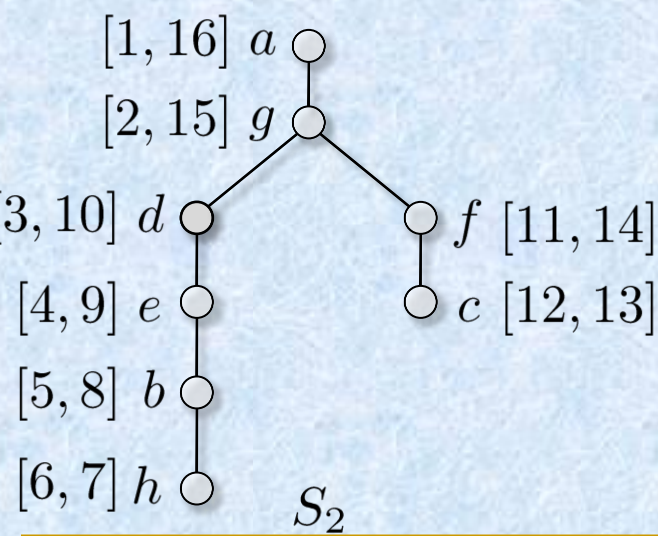
S_1



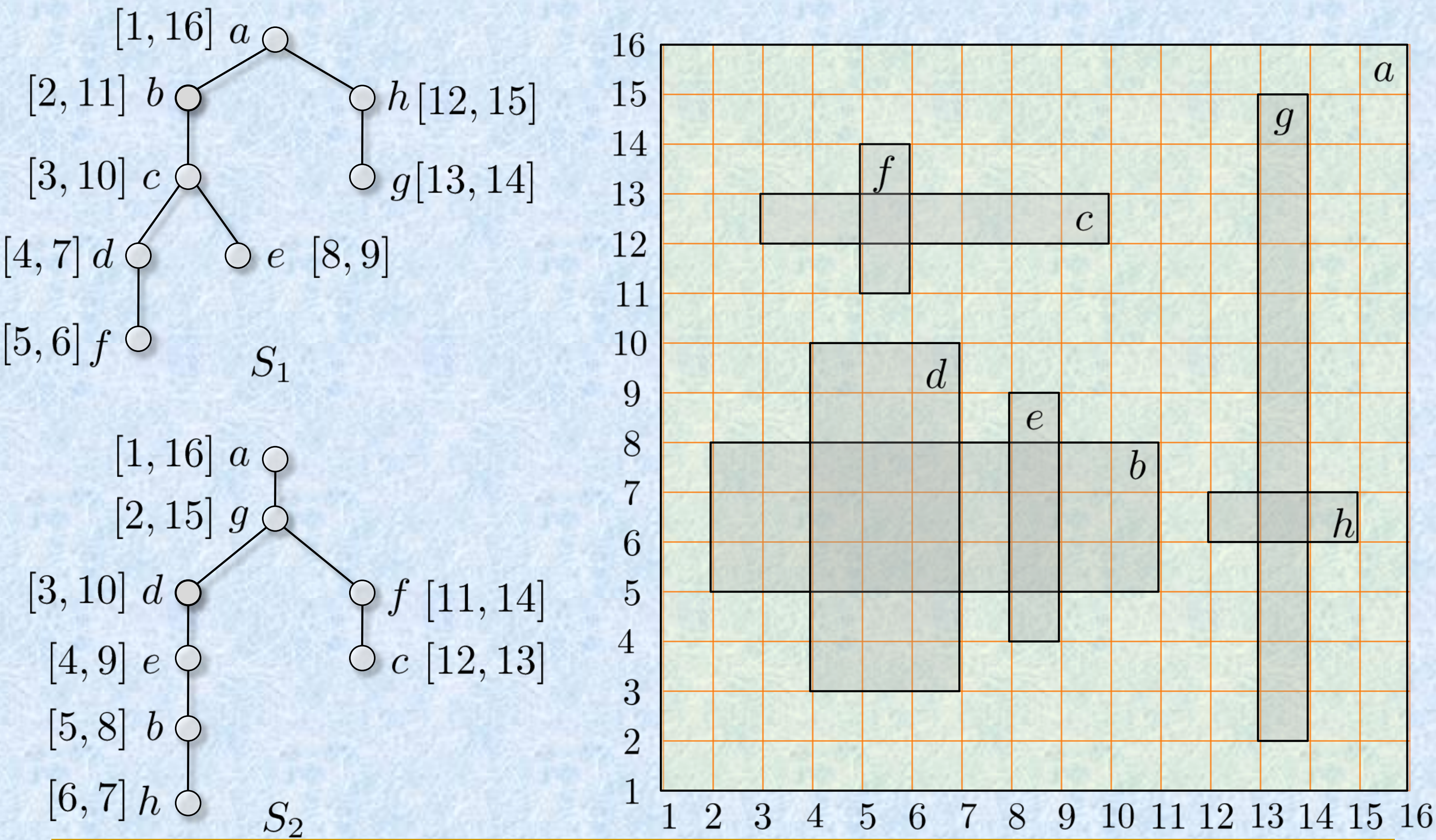
S_2

2-Vertex Connectivity

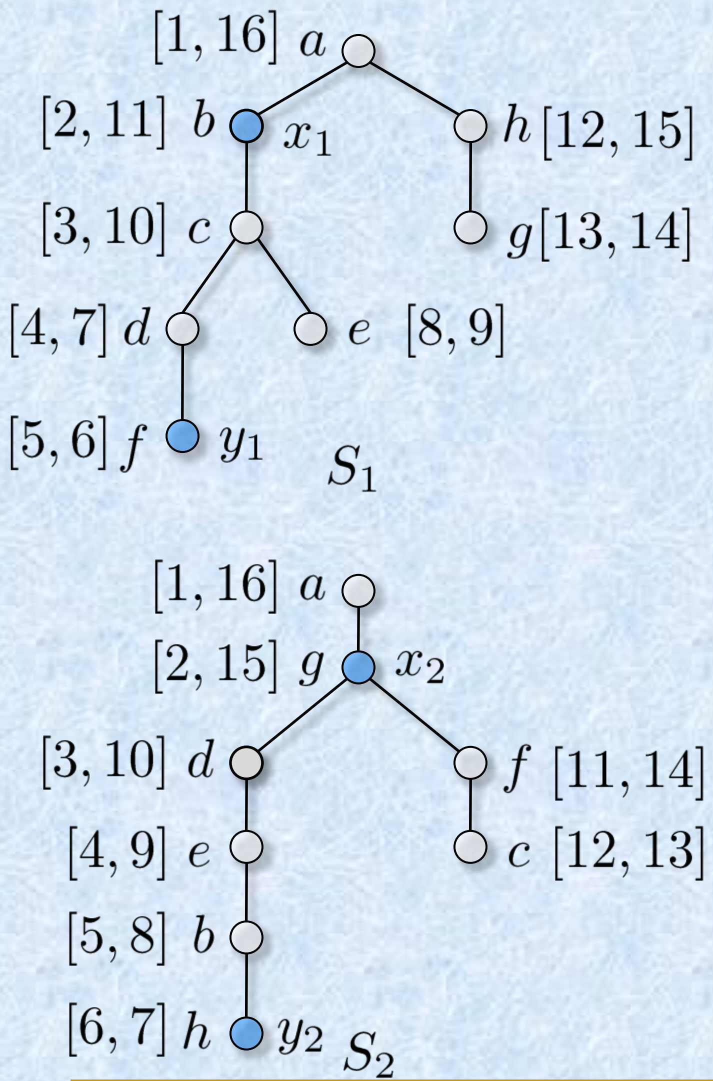




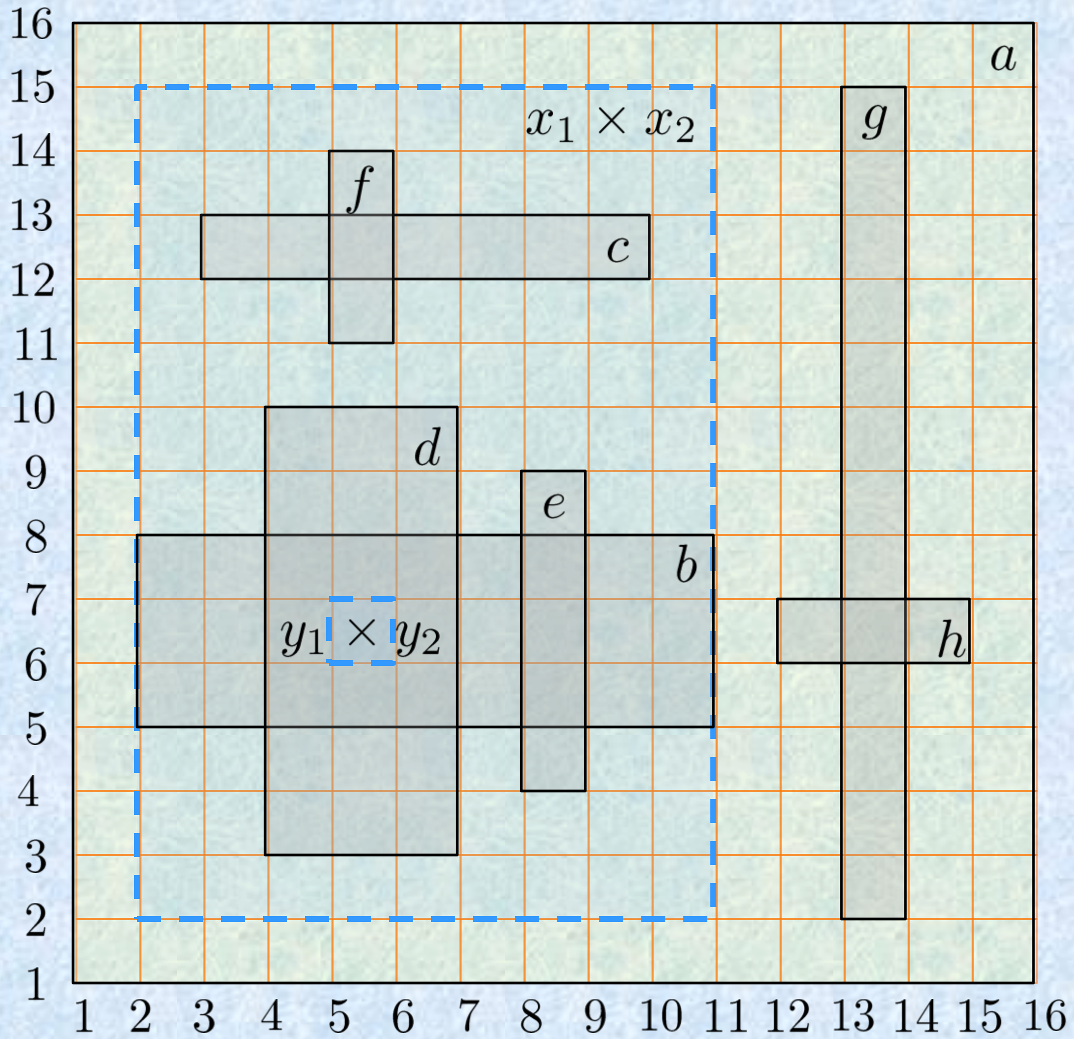
2-Vertex Connectivity



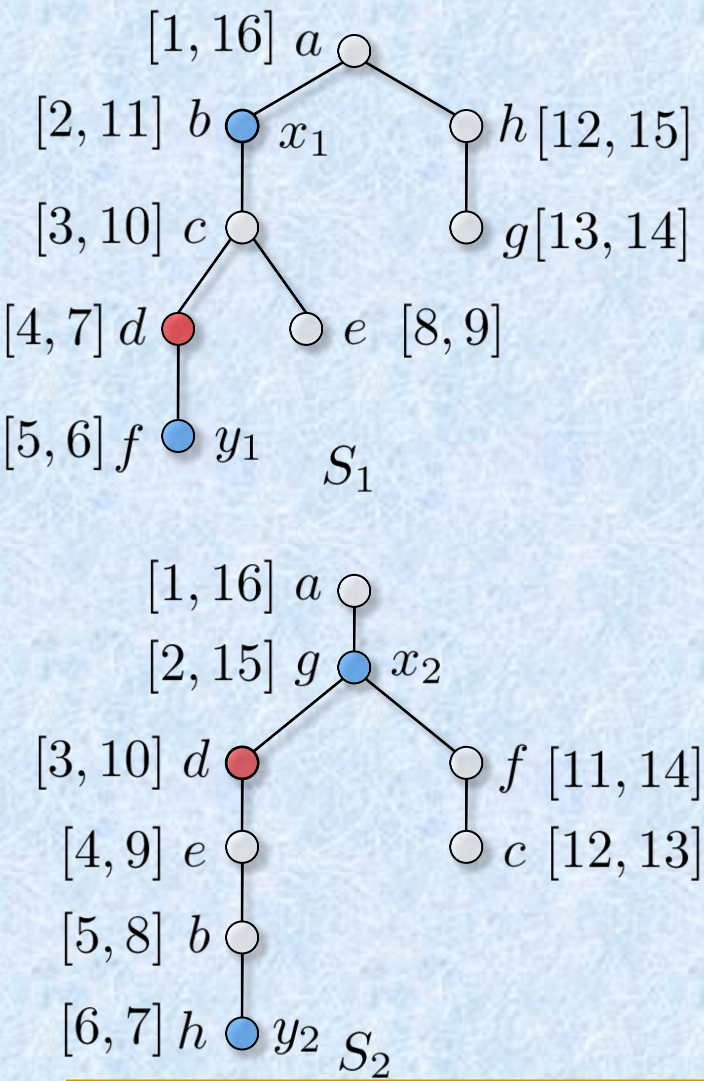
2-Vertex Connectivity



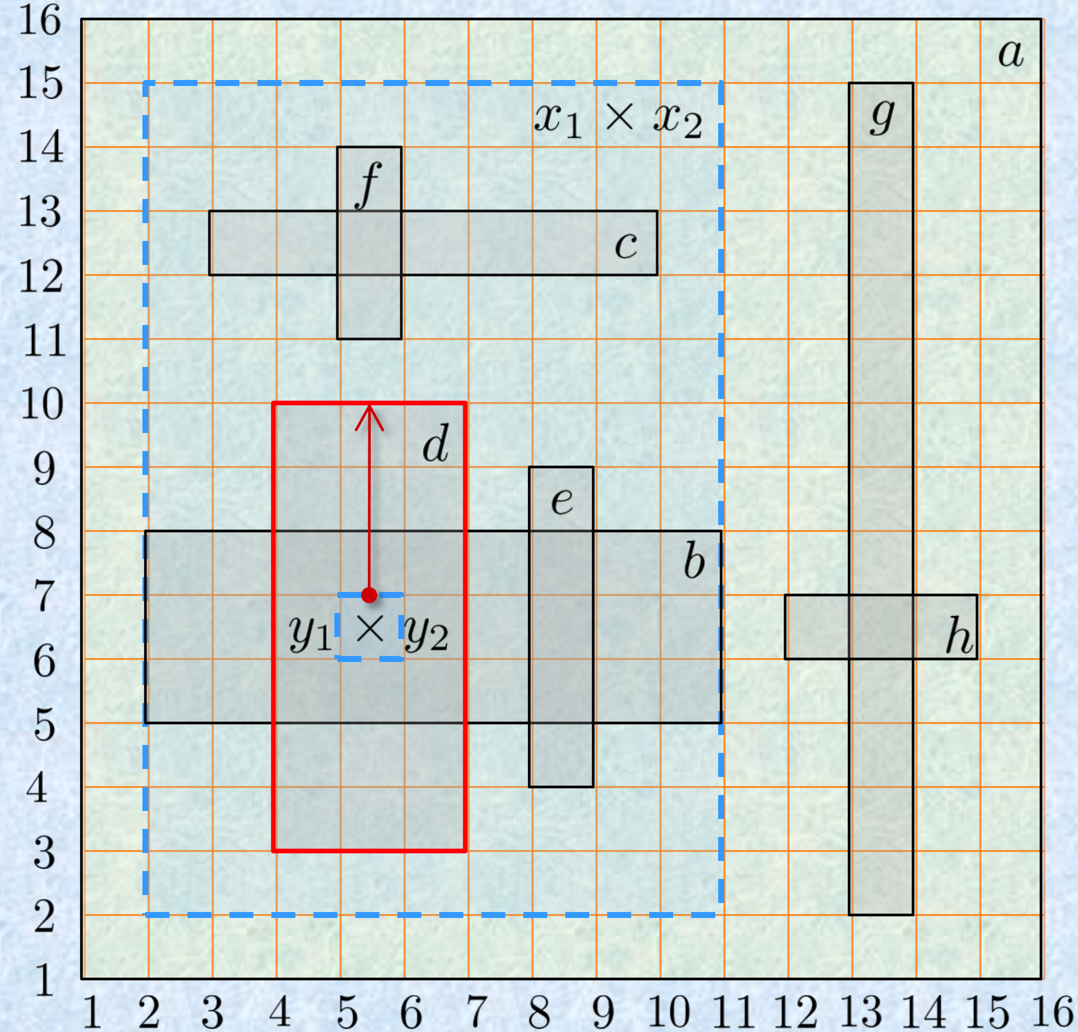
point enclosure problem



2-Vertex Connectivity



point enclosure problem



Operation (v) : find the rectangle R such that $y_1 \times y_2 \subseteq R \subseteq x_1 \times x_2$ and is the farthest from $y_1 \times y_2$ in the vertical direction

Extensions

- Data structure for edge-disjoint paths
- Compute more than 2 disjoint paths