Lempel, Even, and Cederbaum Planarity Method

John M. Boyer

(PureEdge Solutions Inc.)

Cristina G. Fernandes

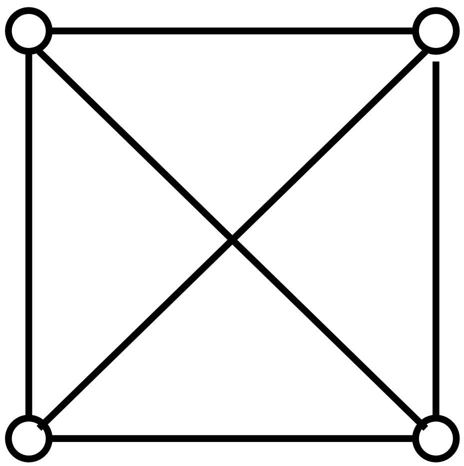
Alexandre Noma

José Coelho de Pina

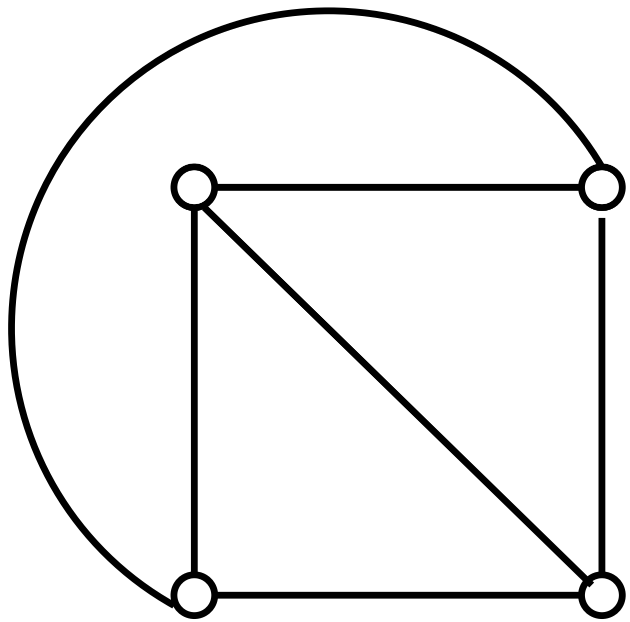
(Universidade de São Paulo)

Planar Graphs

A graph is planar if it can be embedded into the plane without edge crossings.



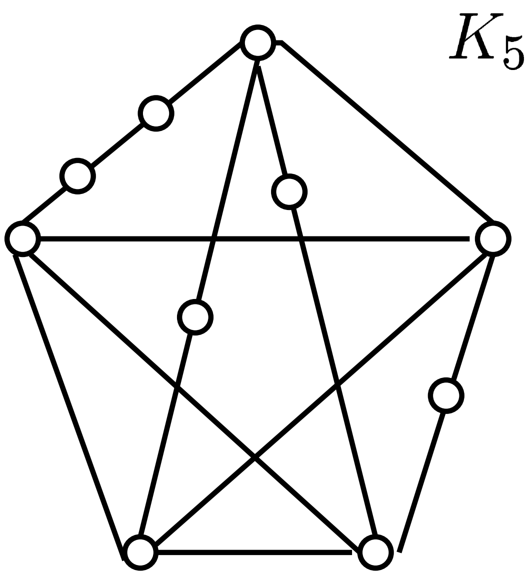
planar graph

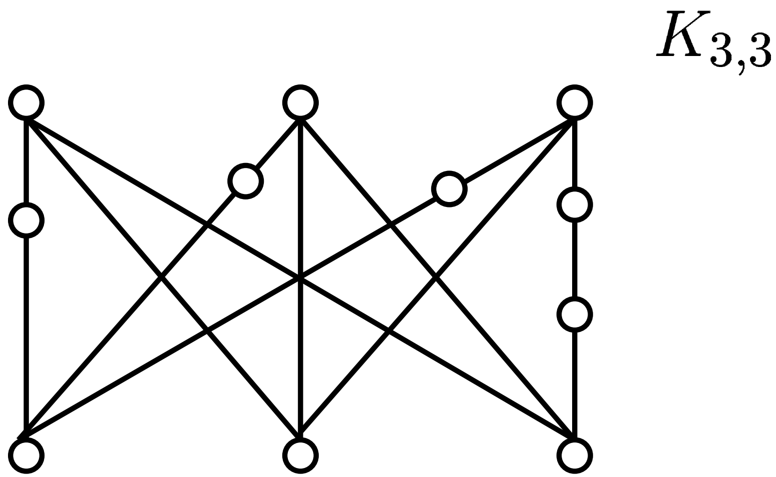


planar embedding

Planar Graphs

A graph is planar if it can be embedded into the plane without edge crossings.





nonplanar graphs

Kuratowski subgraphs

Planarity Test

Problem: Given a graph, decide whether it is planar or not.

Planarity Test

Problem: Given a graph, decide whether it is planar or not.

Method of Auslander and Parter '61 and Goldstein '63

- Hopcroft and Tarjan (HT) '74

Method of Lempel, Even and Cederbaum (LEC) '67

- Booth and Lueker (BL) '74
- Shih and Hsu (SH) '93
- Boyer and Myrvold (BM) '99

Planarity Test

Problem: Given a graph, decide whether it is planar or not.

Method of Auslander and Parter ’61 and Goldstein ’63

- Hopcroft and Tarjan (HT) ’74

Method of Lempel, Even and Cederbaum (LEC) ’67

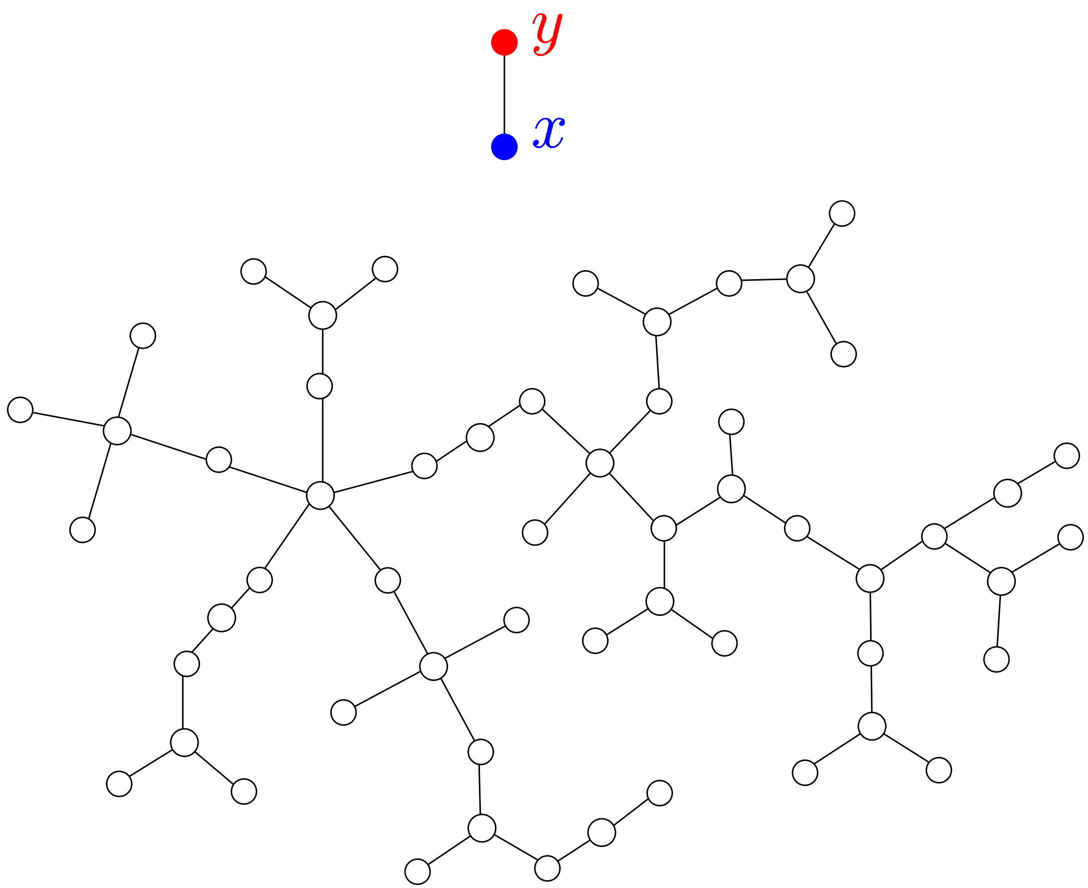
- Booth and Lueker (BL) ’74
- Shih and Hsu (SH) ’93
- Boyer and Myrvold (BM) ’99

This work

- Simple graph theoretical description of LEC
- Linear-time implementation of LEC (SH)
- Experimental study

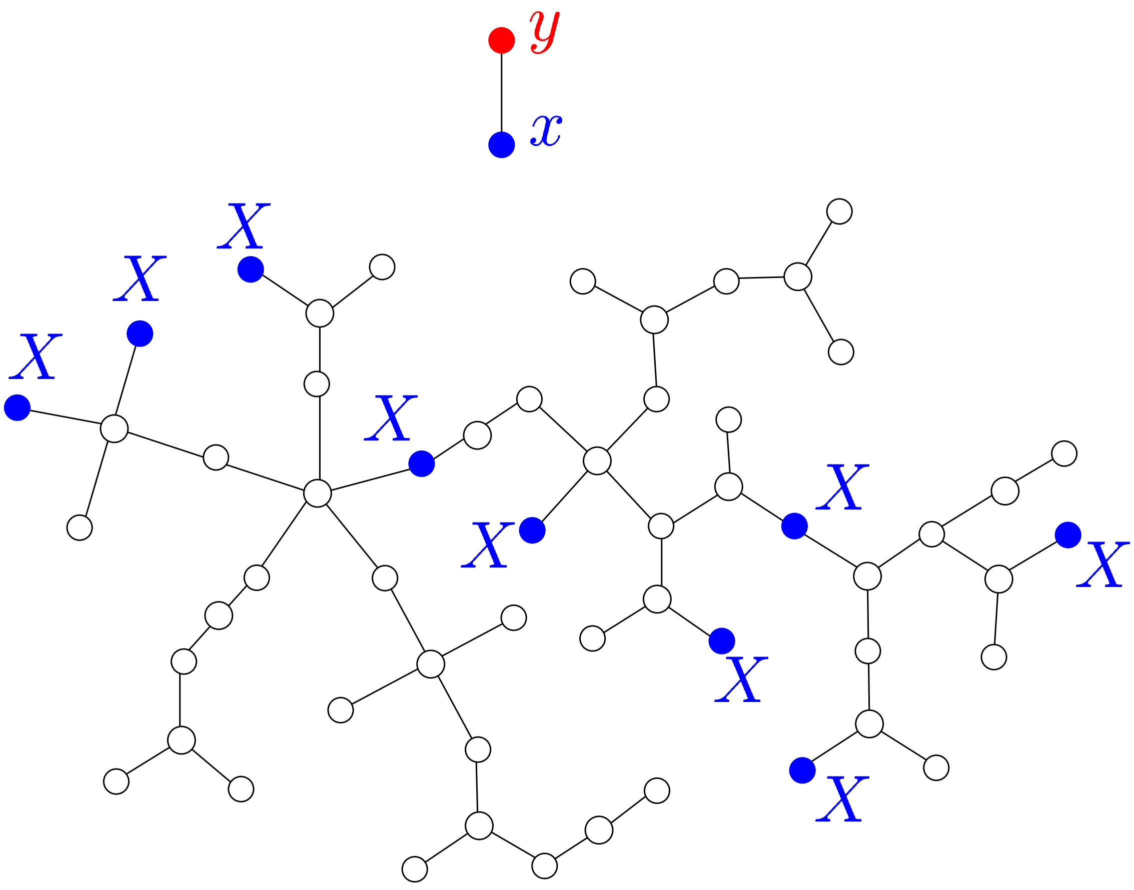
Key Simple Problem

G graph on vertex set $V_T \cup \{x, y\}$
T: V_T induces a tree T
xy is an edge



Key Simple Problem

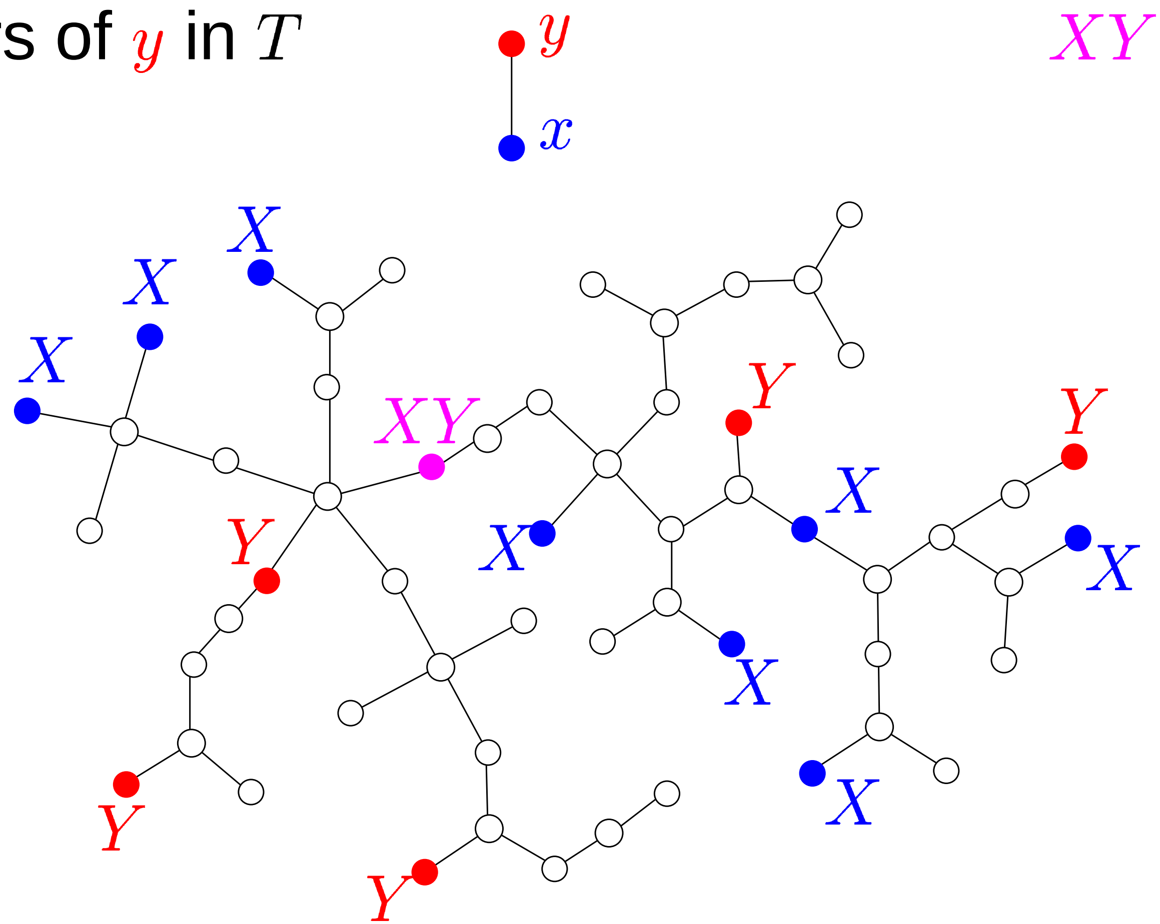
G graph on vertex set $V_T \cup \{x, y\}$
T: V_T induces a tree T
xy is an edge
X neighbors of x in T



Key Simple Problem

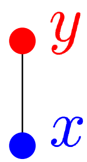
G graph on vertex set $V_T \cup \{x, y\}$
T: V_T induces a tree T
xy is an edge
X neighbors of x in T
Y neighbors of y in T

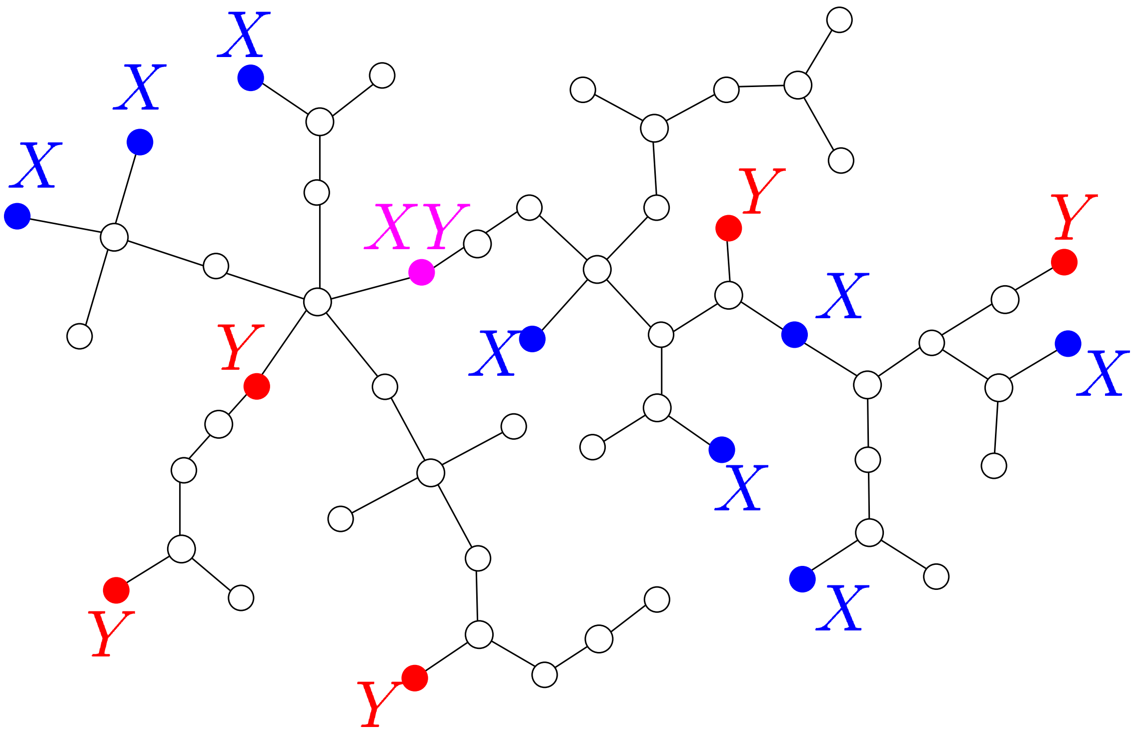
$XY = X \cap Y$



Key Simple Problem

Give an algorithm that receives T, xy, X, Y and decides whether there is an embedding of $T + x$ leaving all Y vertices on the 'frame'.

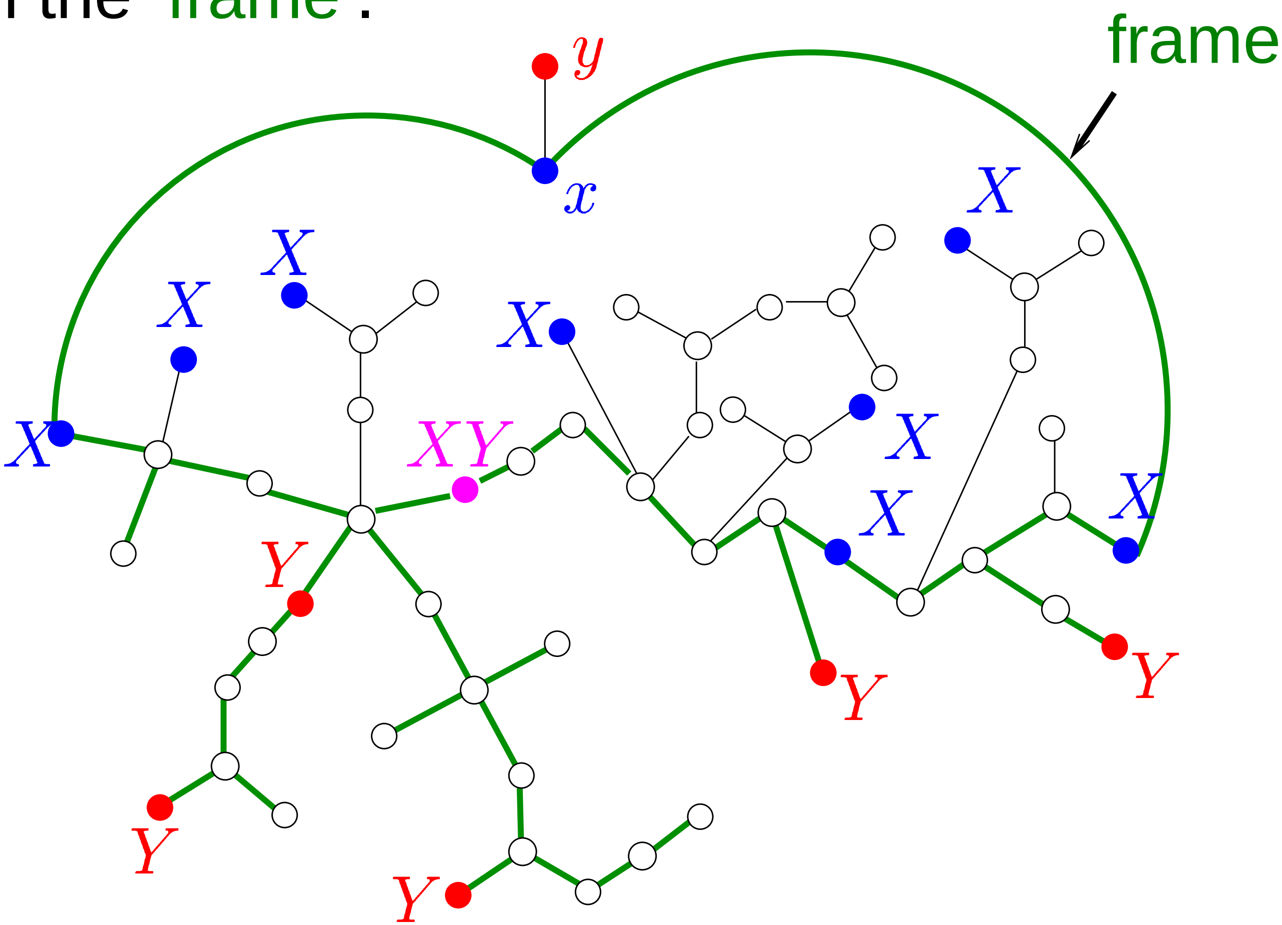




YES $\Leftrightarrow$ G is planar

Key Simple Problem

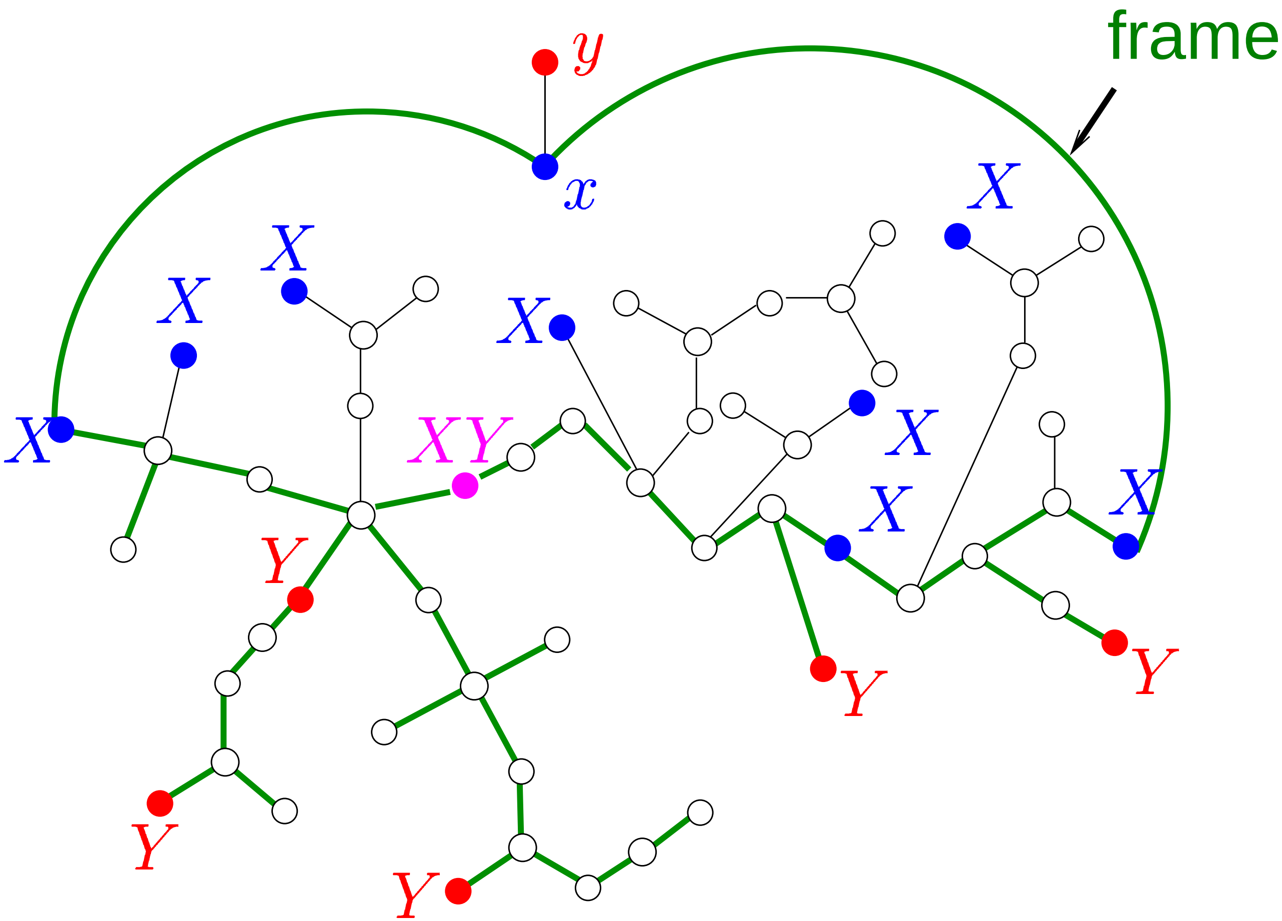
Give an algorithm that receives T, xy, X, Y and decides whether there is an embedding of $T + x$ leaving all Y vertices on the 'frame'.



YES $\Leftrightarrow$ G is planar

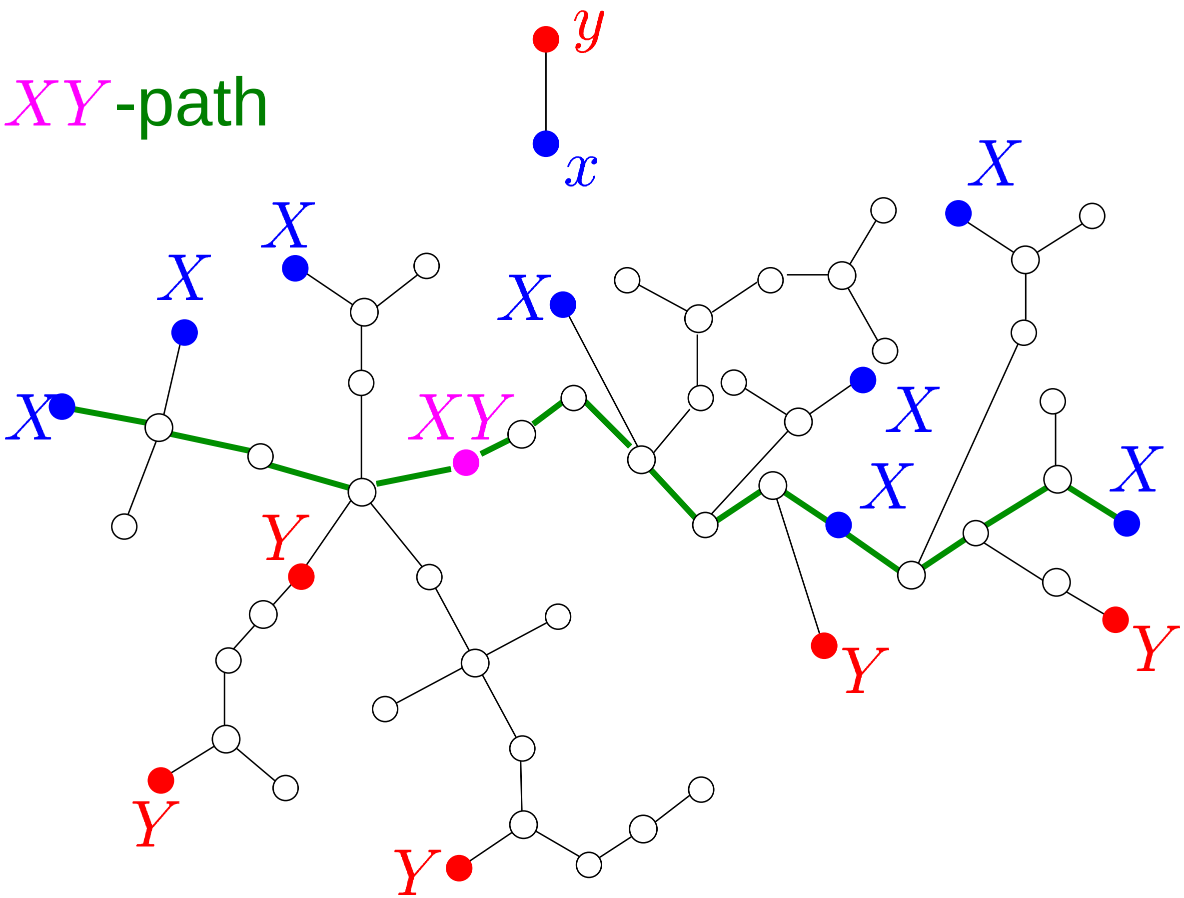
XY-path

Want a path in T connecting 2 vertices in X and 'splitting' X and Y.



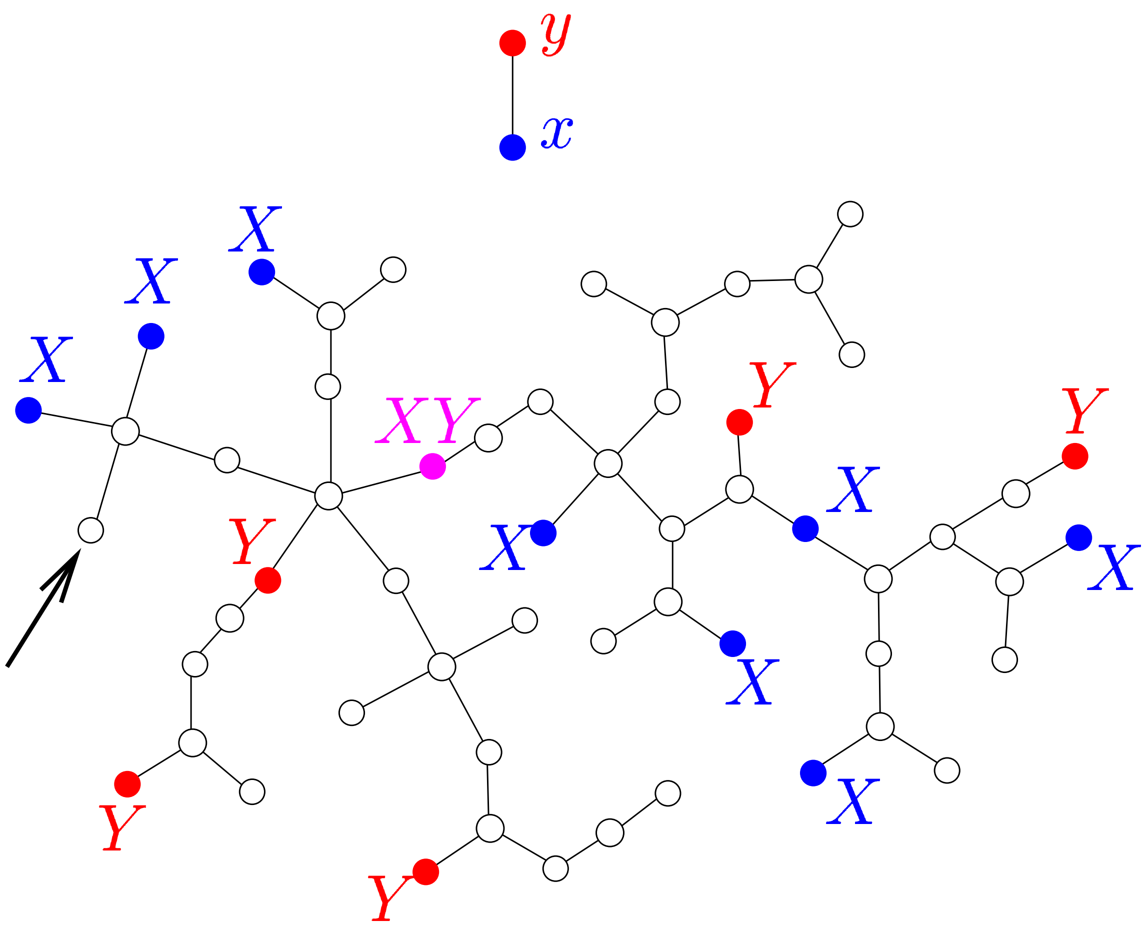
XY-path

Want a path in T connecting 2 vertices in X and 'splitting' X and Y.



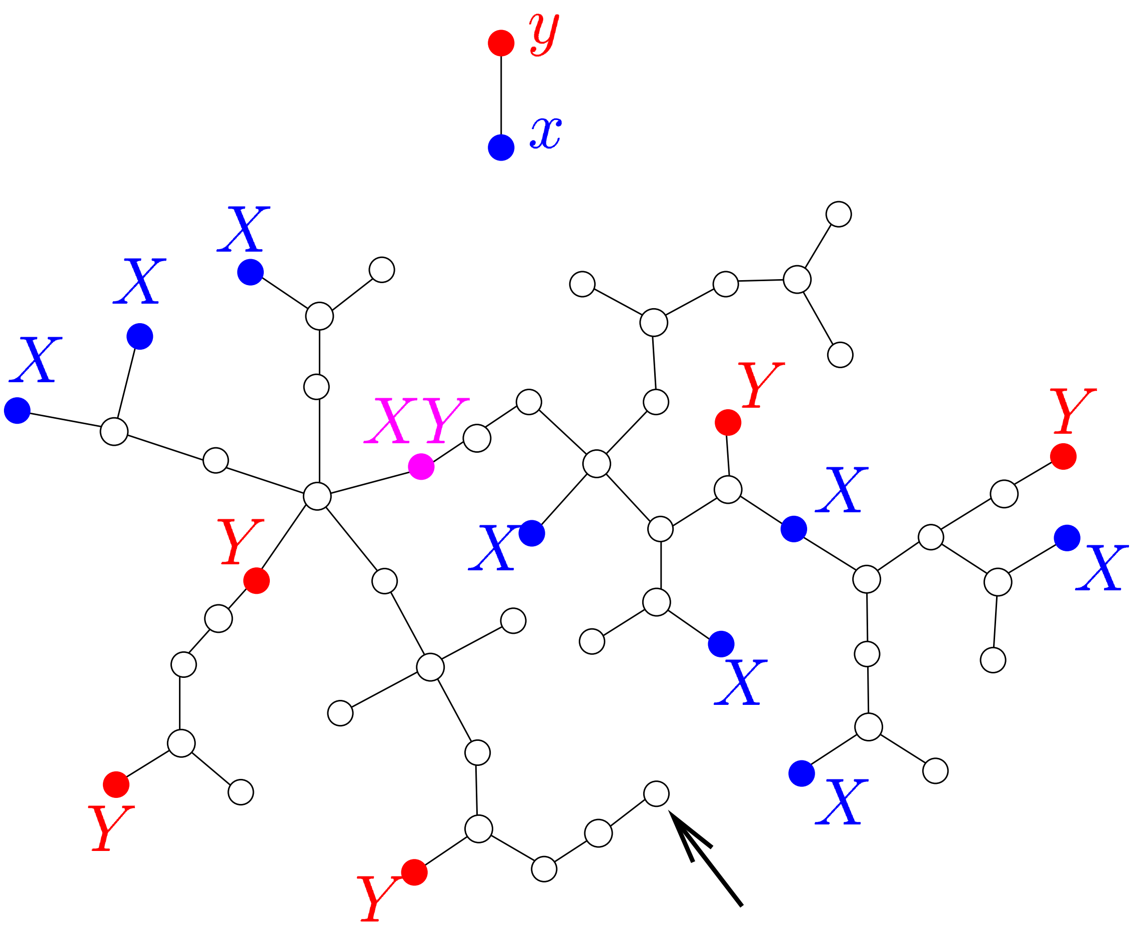
Reduction

Leaves not in $X \cup Y$ do not matter.



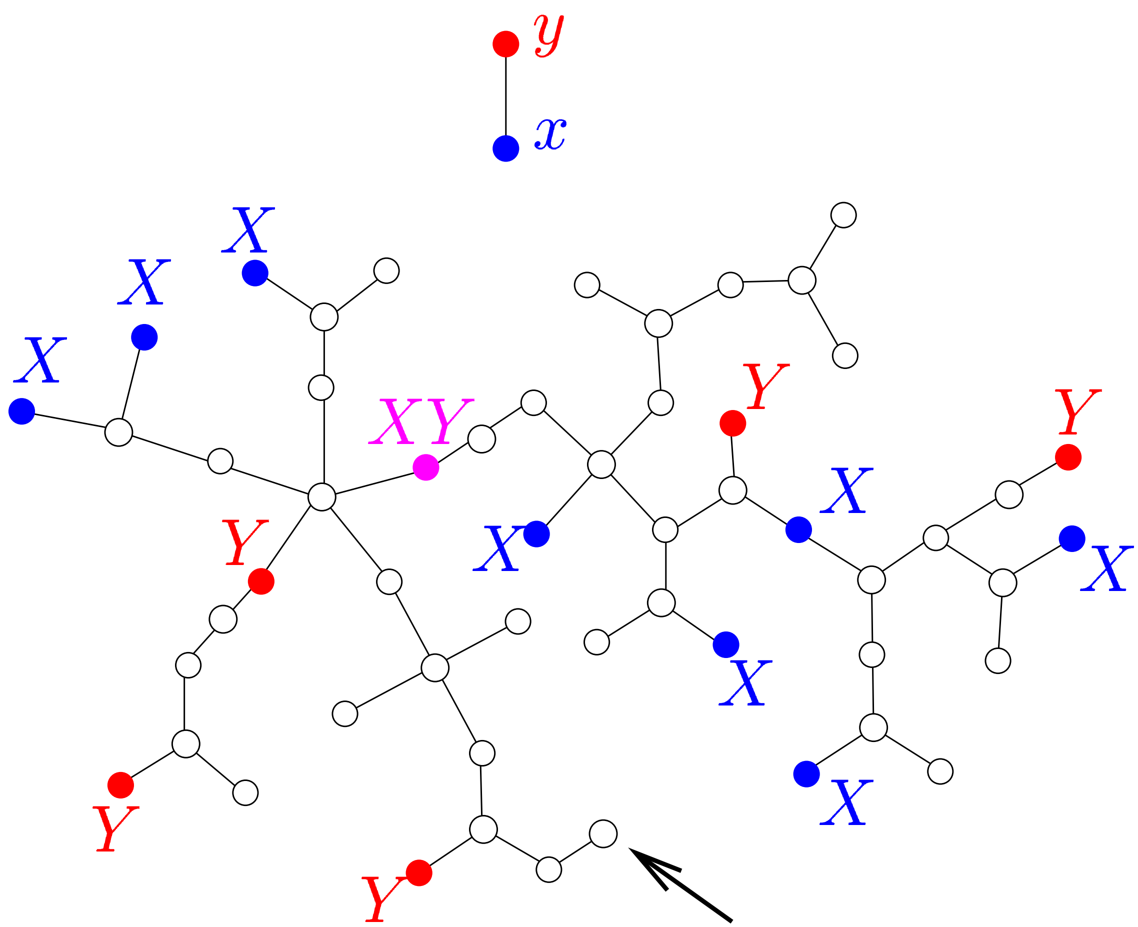
Reduction

Leaves not in $X \cup Y$ do not matter.



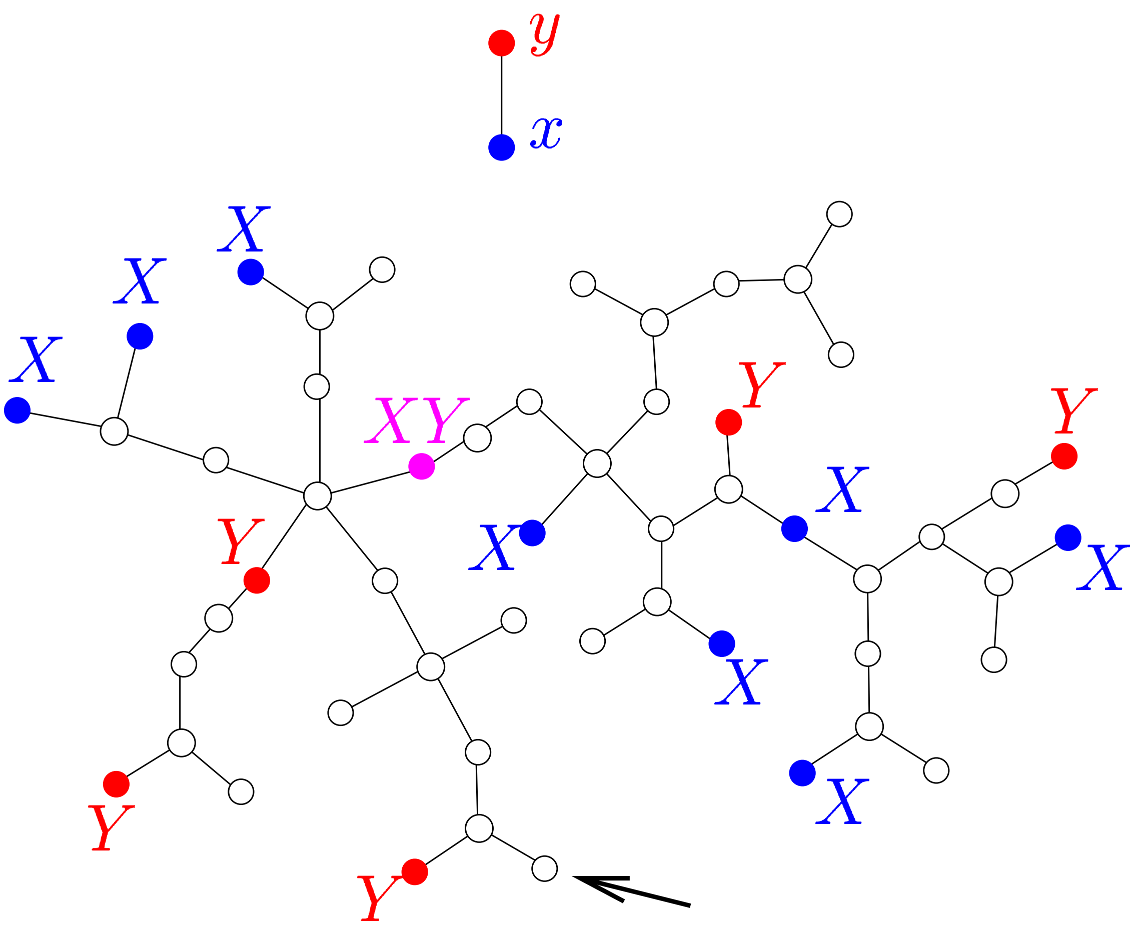
Reduction

Leaves not in $X \cup Y$ do not matter.



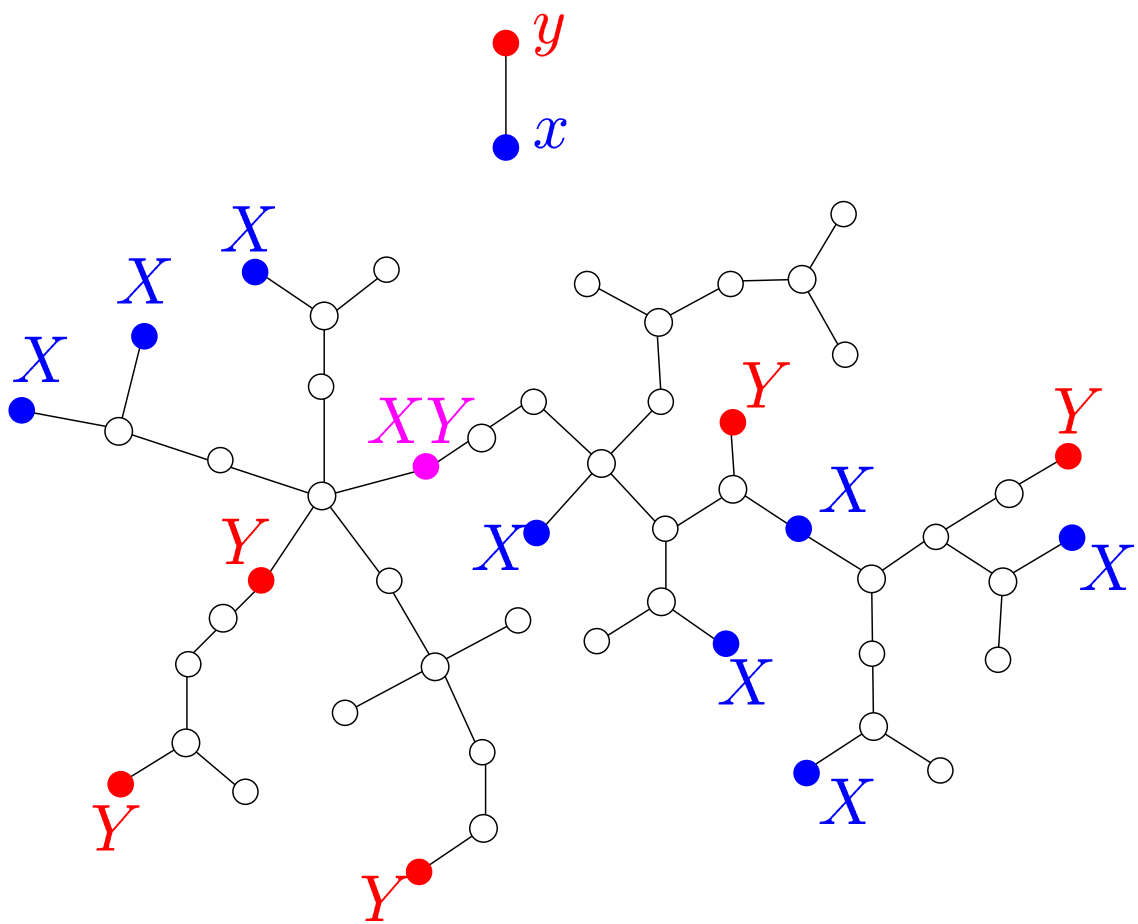
Reduction

Leaves not in $X \cup Y$ do not matter.



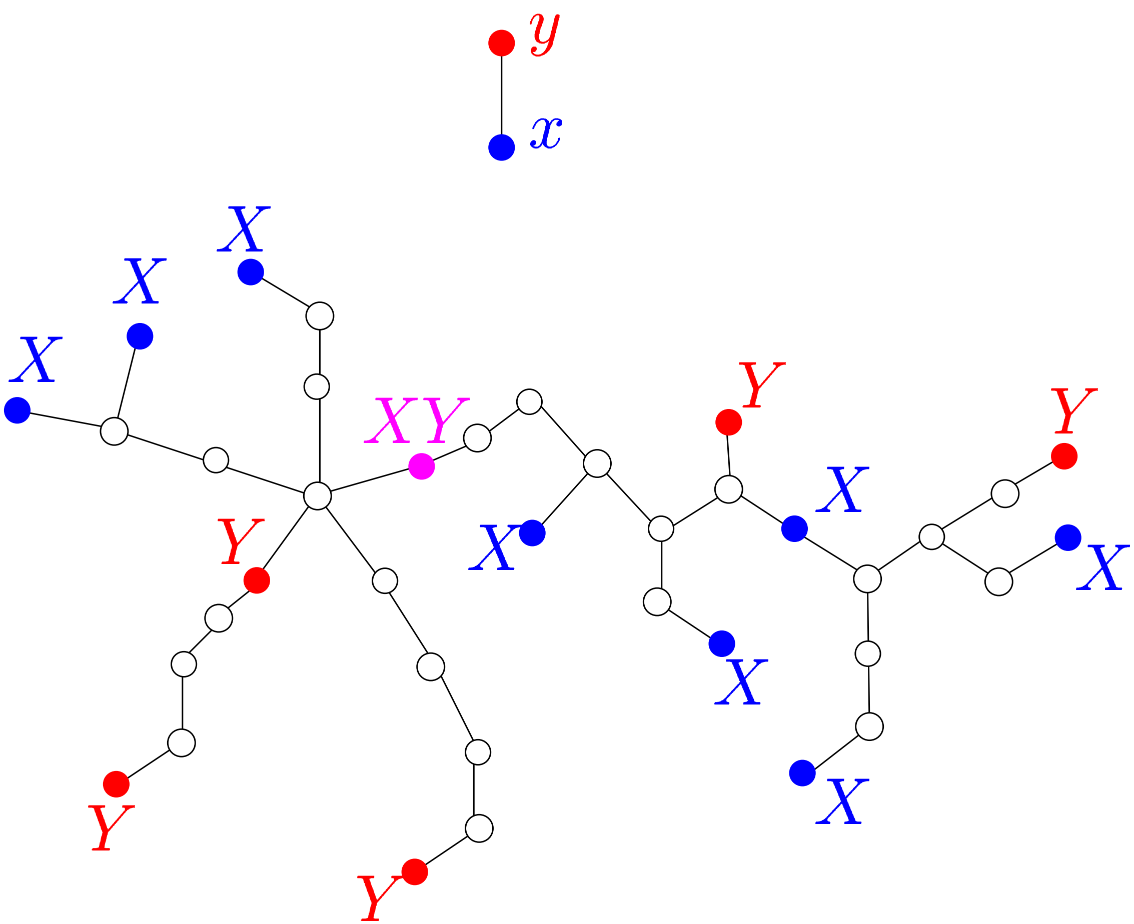
Reduction

Leaves not in $X \cup Y$ do not matter.



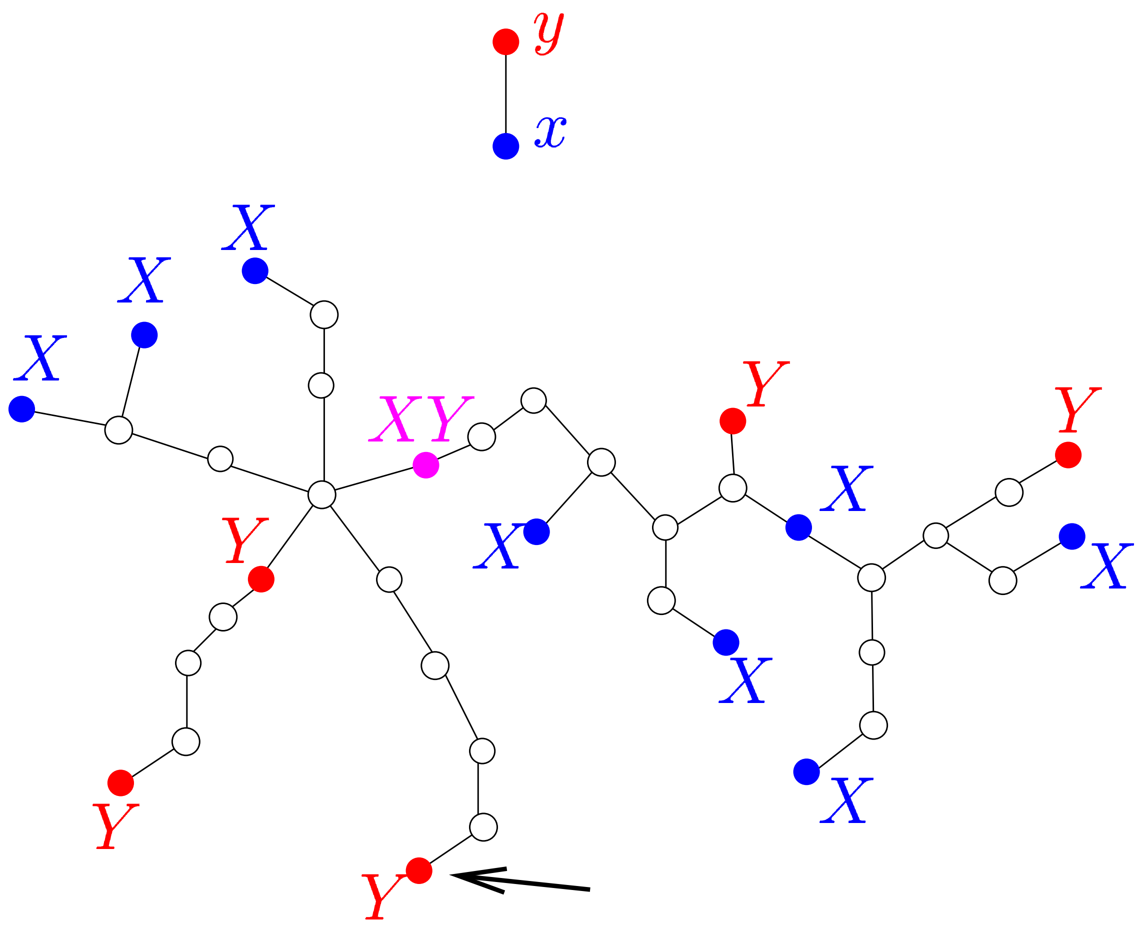
Reduction

Leaves not in $X \cup Y$ do not matter.



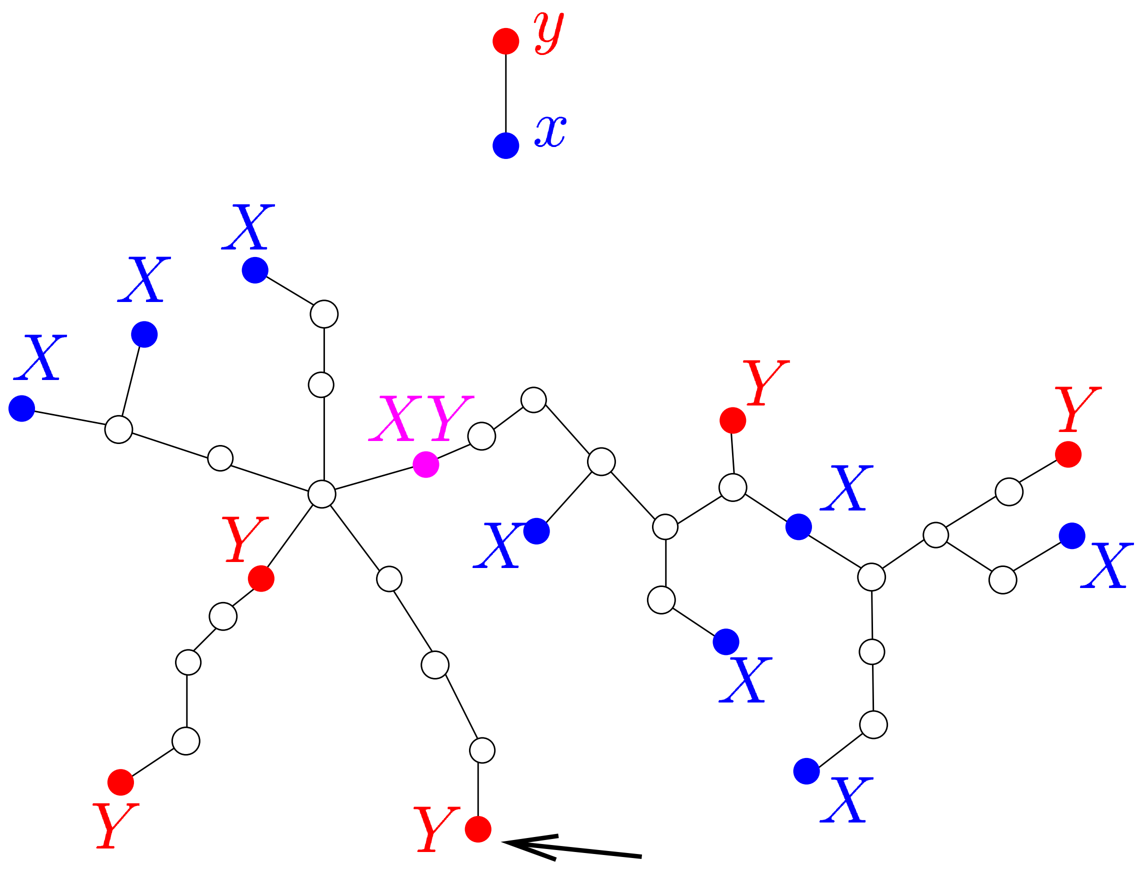
Reduction

Leaves in $Y \setminus X$ can be 'contracted'.



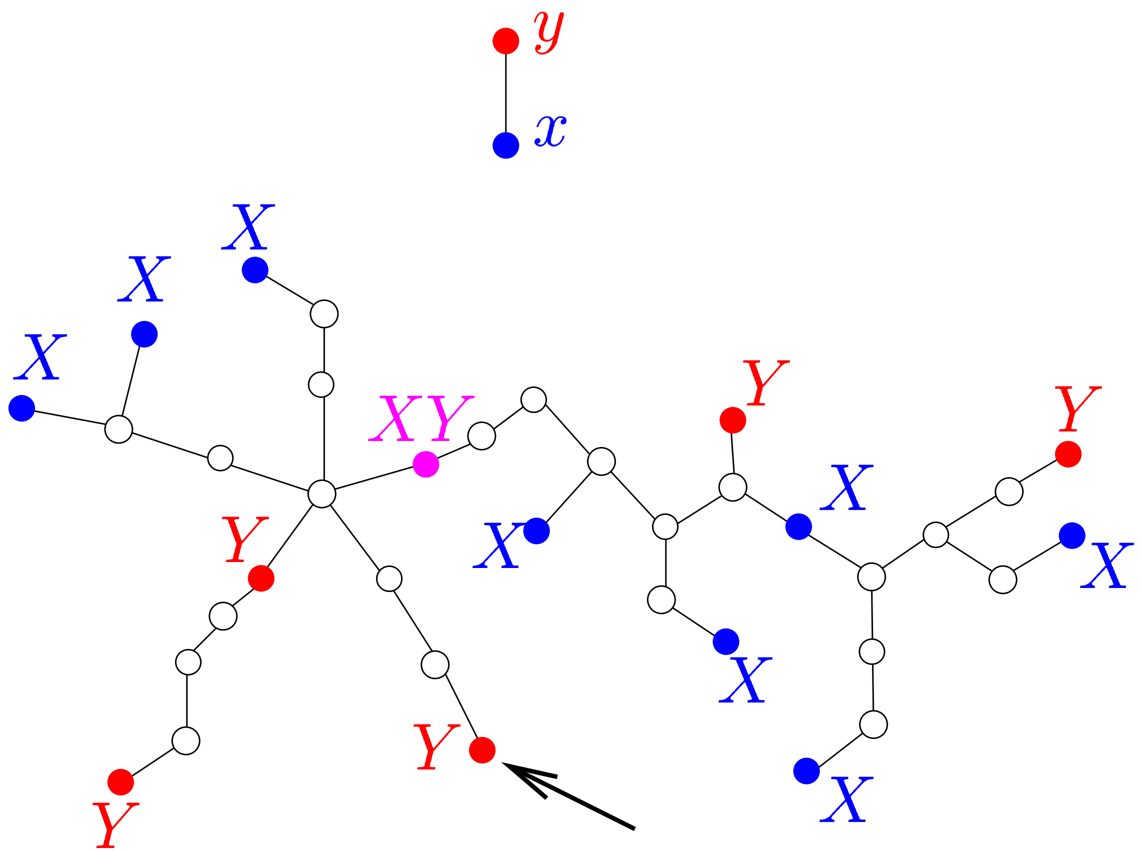
Reduction

Leaves in $Y \setminus X$ can be ‘contracted’.



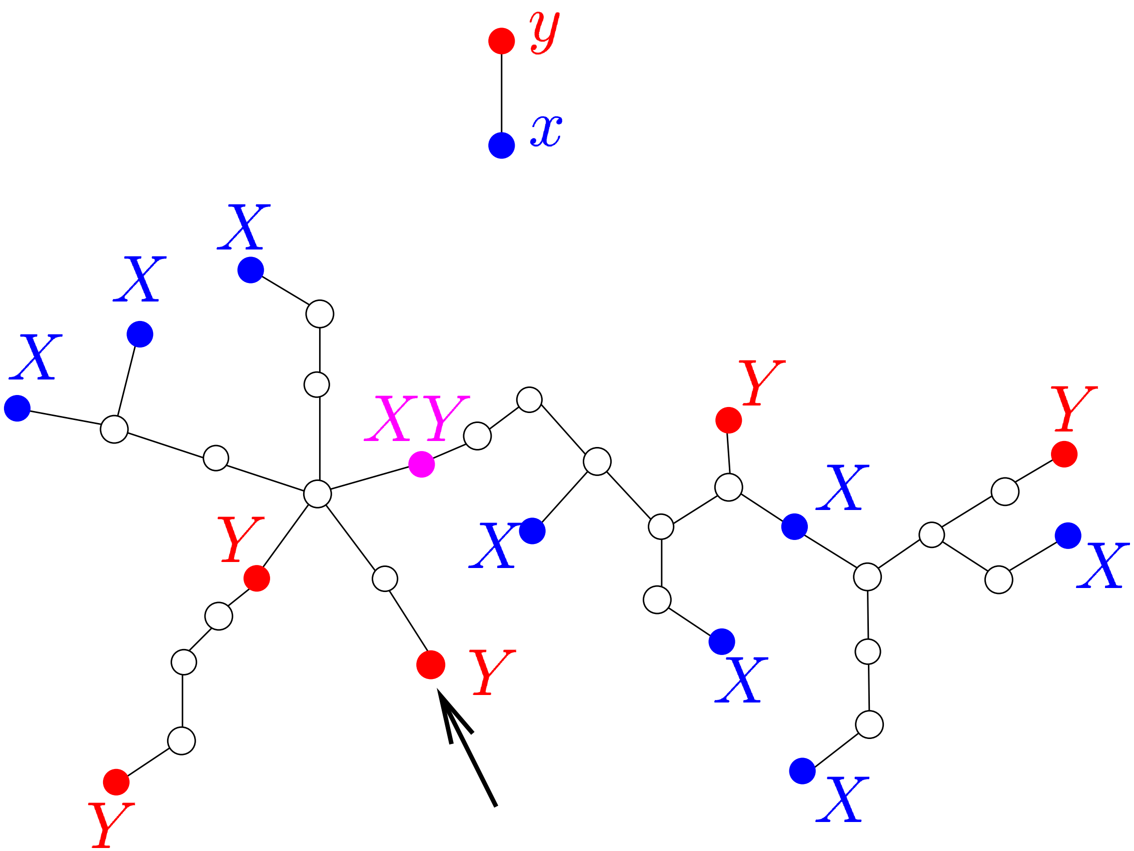
Reduction

Leaves in $Y \setminus X$ can be 'contracted'.



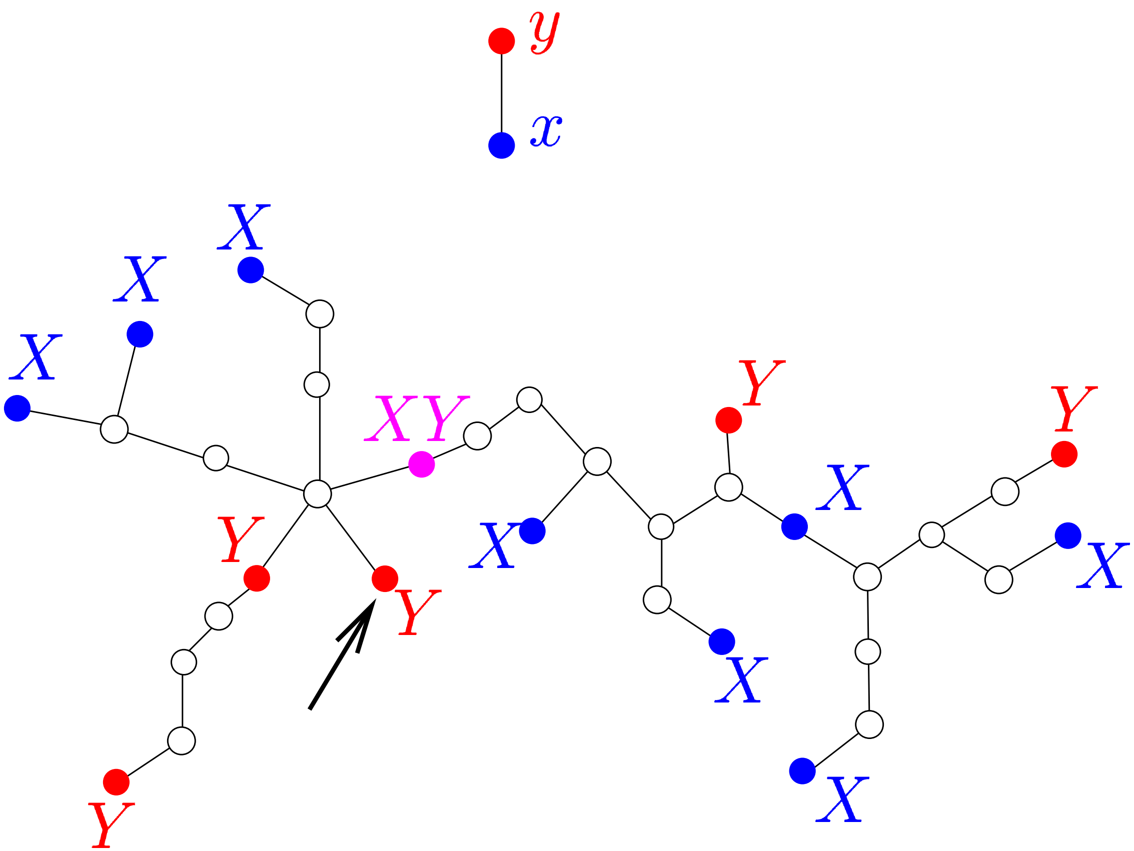
Reduction

Leaves in $Y \setminus X$ can be 'contracted'.



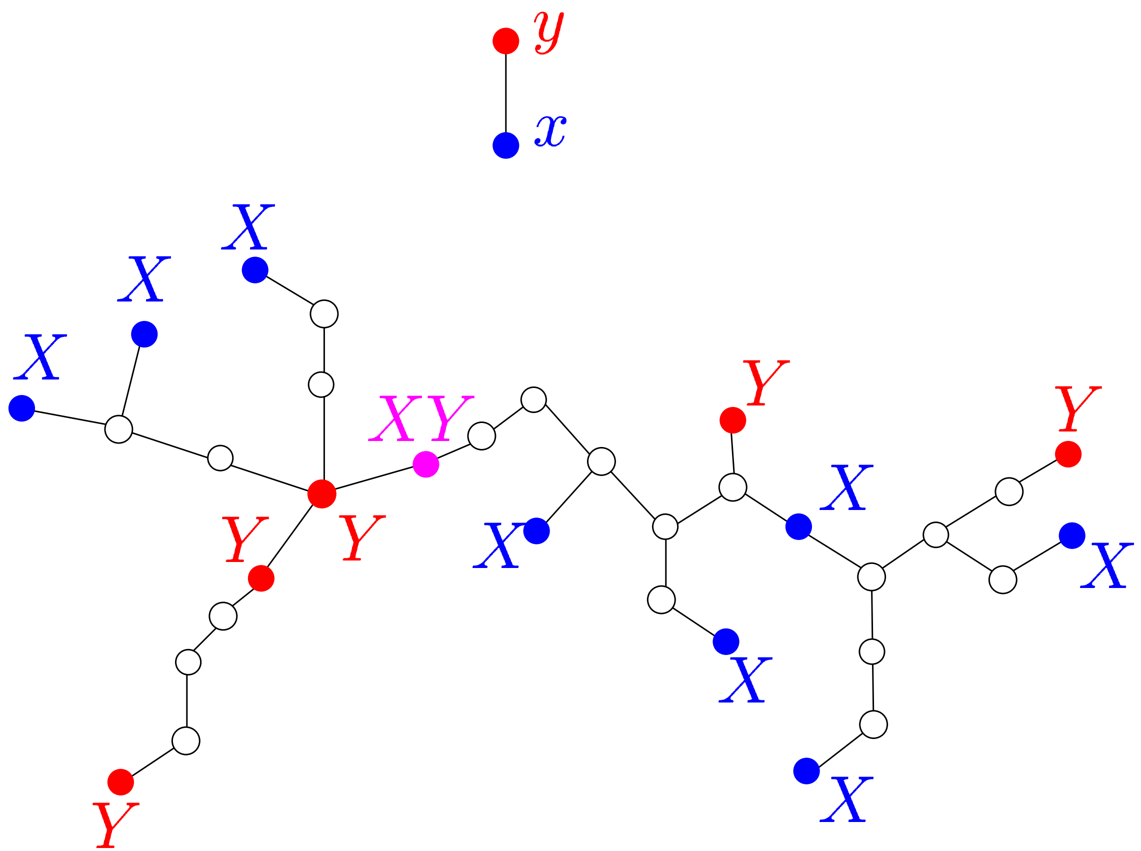
Reduction

Leaves in $Y \setminus X$ can be ‘contracted’.



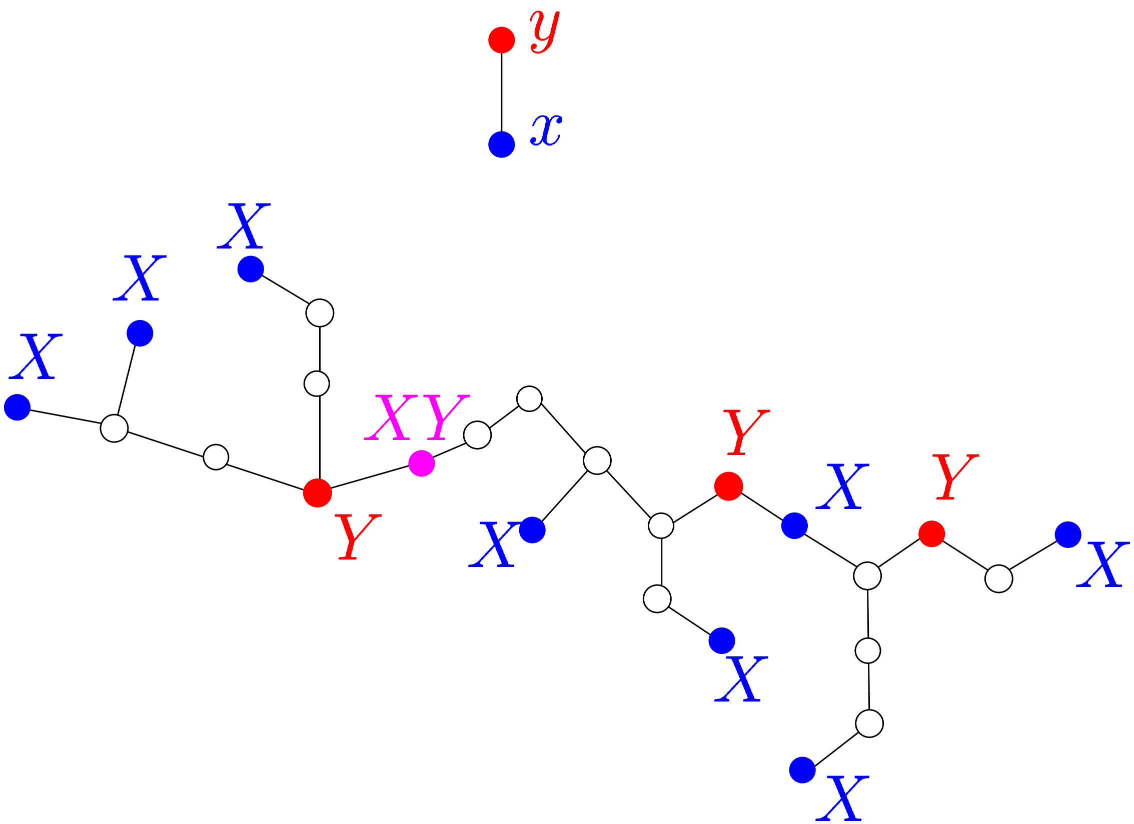
Reduction

Leaves in $Y \setminus X$ can be ‘contracted’.



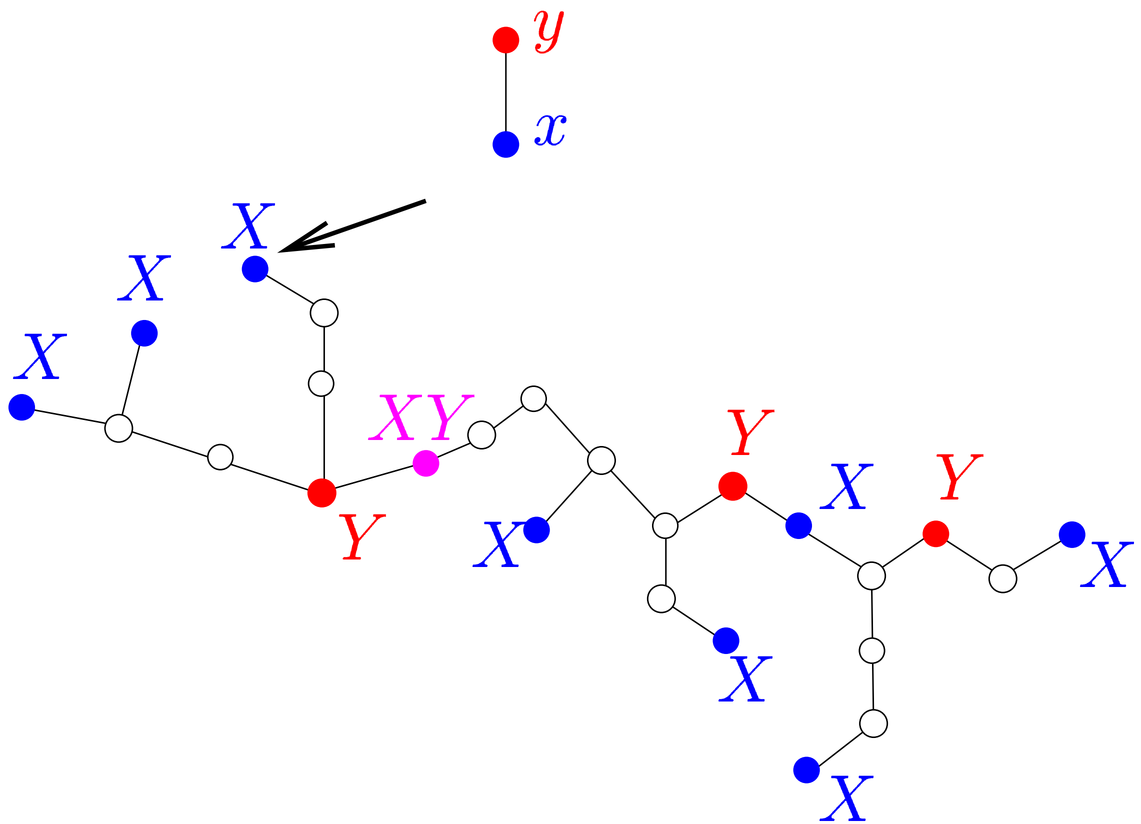
Reduction

Leaves in $Y \setminus X$ can be 'contracted'.



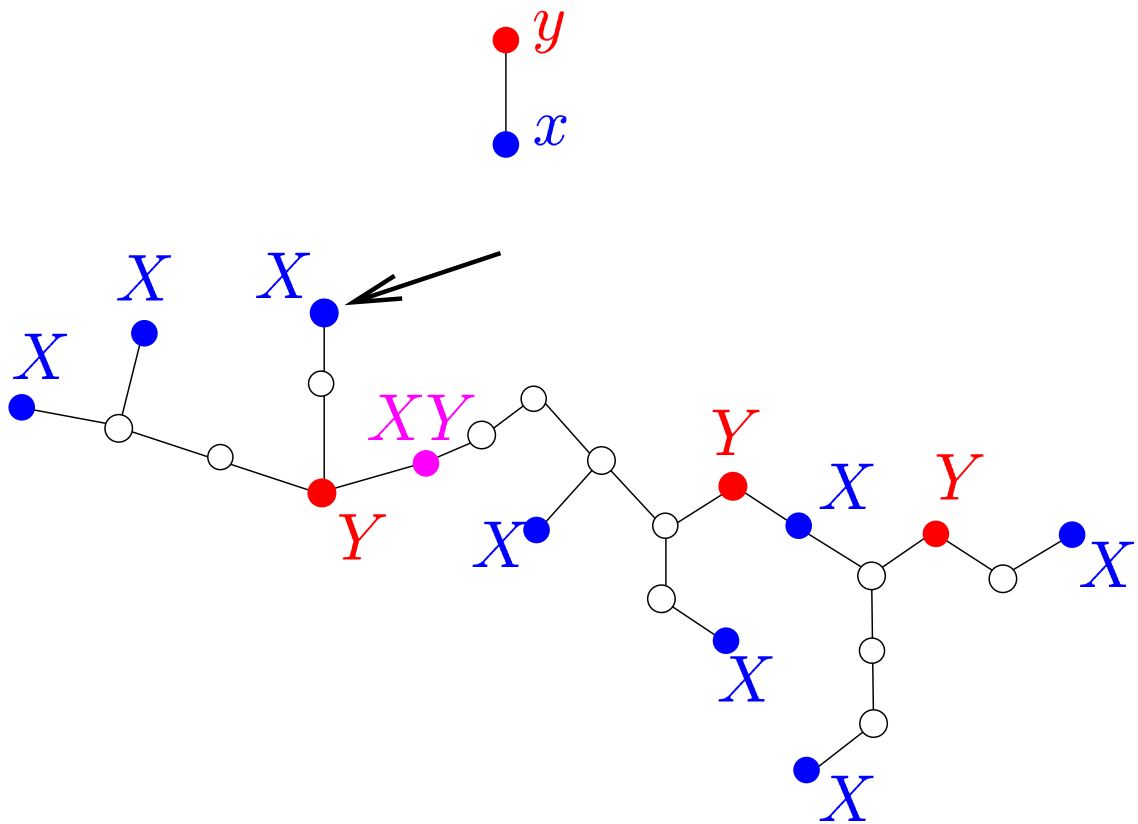
Reduction

Leaves in $X \setminus Y$ can be 'contracted'.



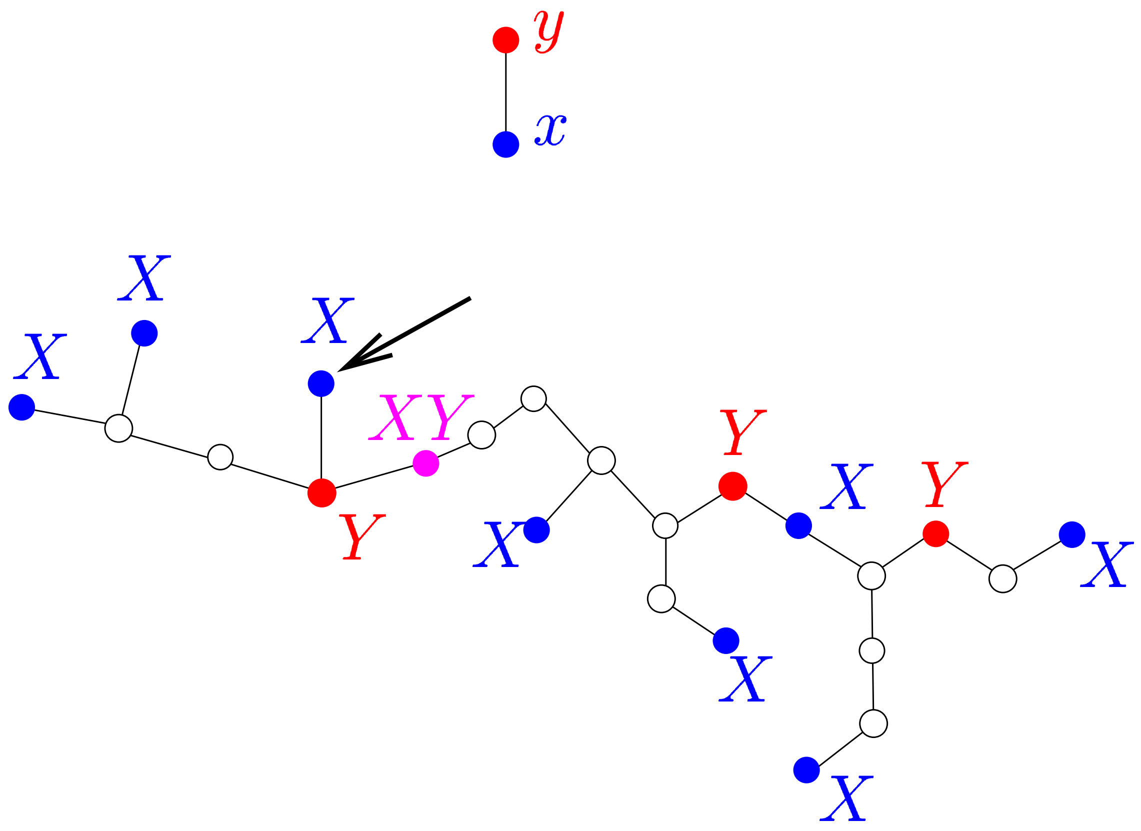
Reduction

Leaves in $X \setminus Y$ can be 'contracted'.



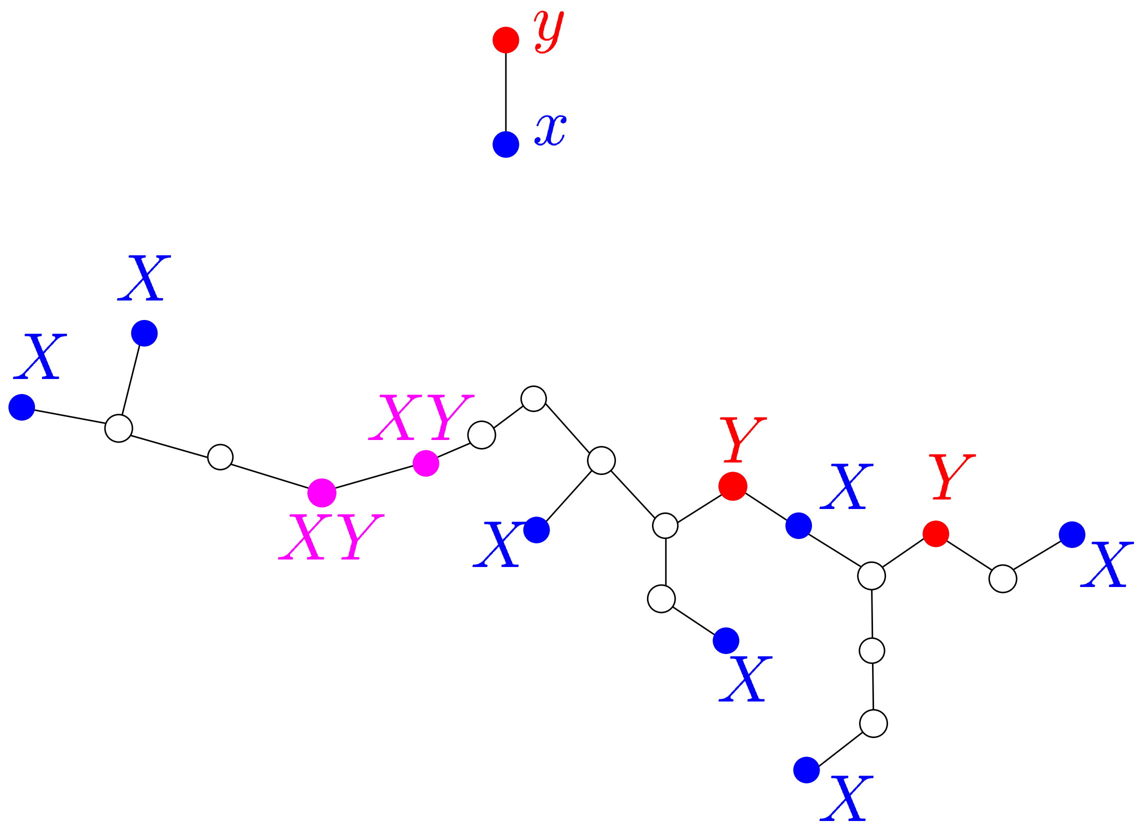
Reduction

Leaves in $X \setminus Y$ can be ‘contracted’.



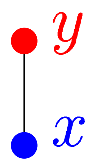
Reduction

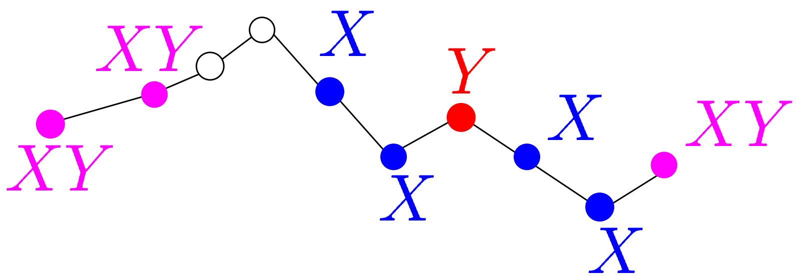
Leaves in $X \setminus Y$ can be 'contracted'.



Reduction

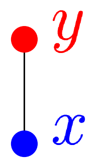
Leaves in $X \setminus Y$ can be 'contracted'.
All leaves are in $XY = X \cap Y$.

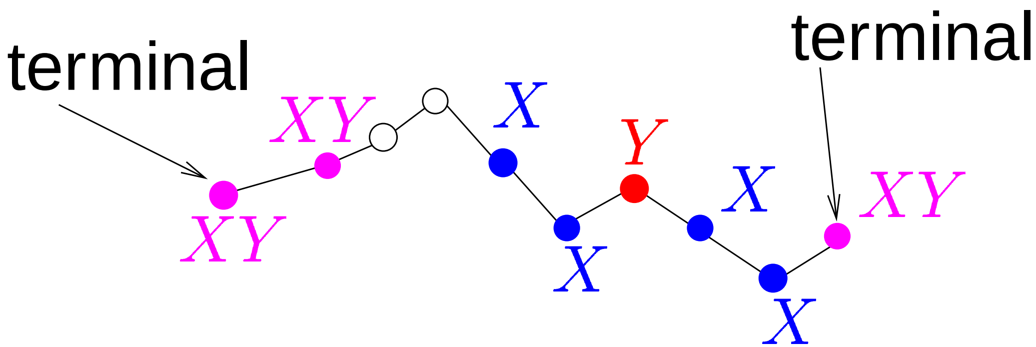




Terminals

terminals := leaves of the reduced tree.

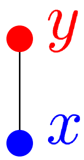


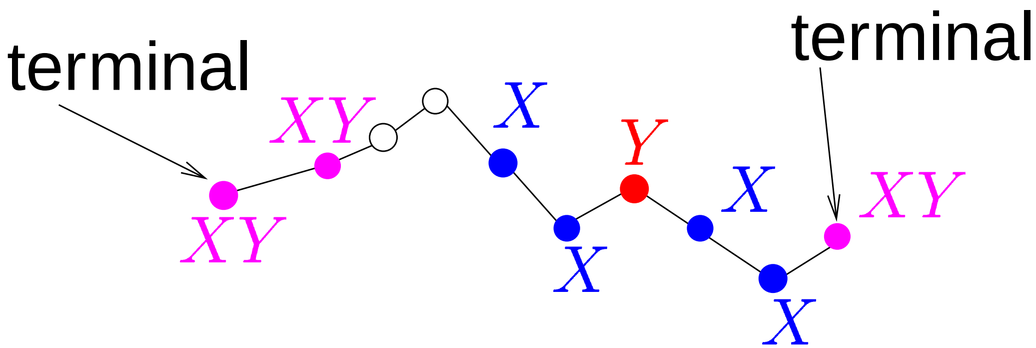


At Most 2 Terminals

Answer YES

Why?

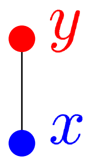


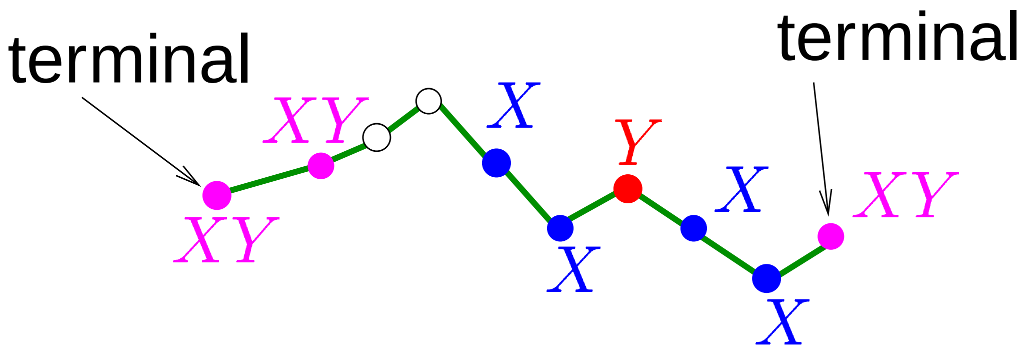


At Most 2 Terminals

Answer YES

≤ 2 terminals $\Rightarrow$ reduced tree is a path

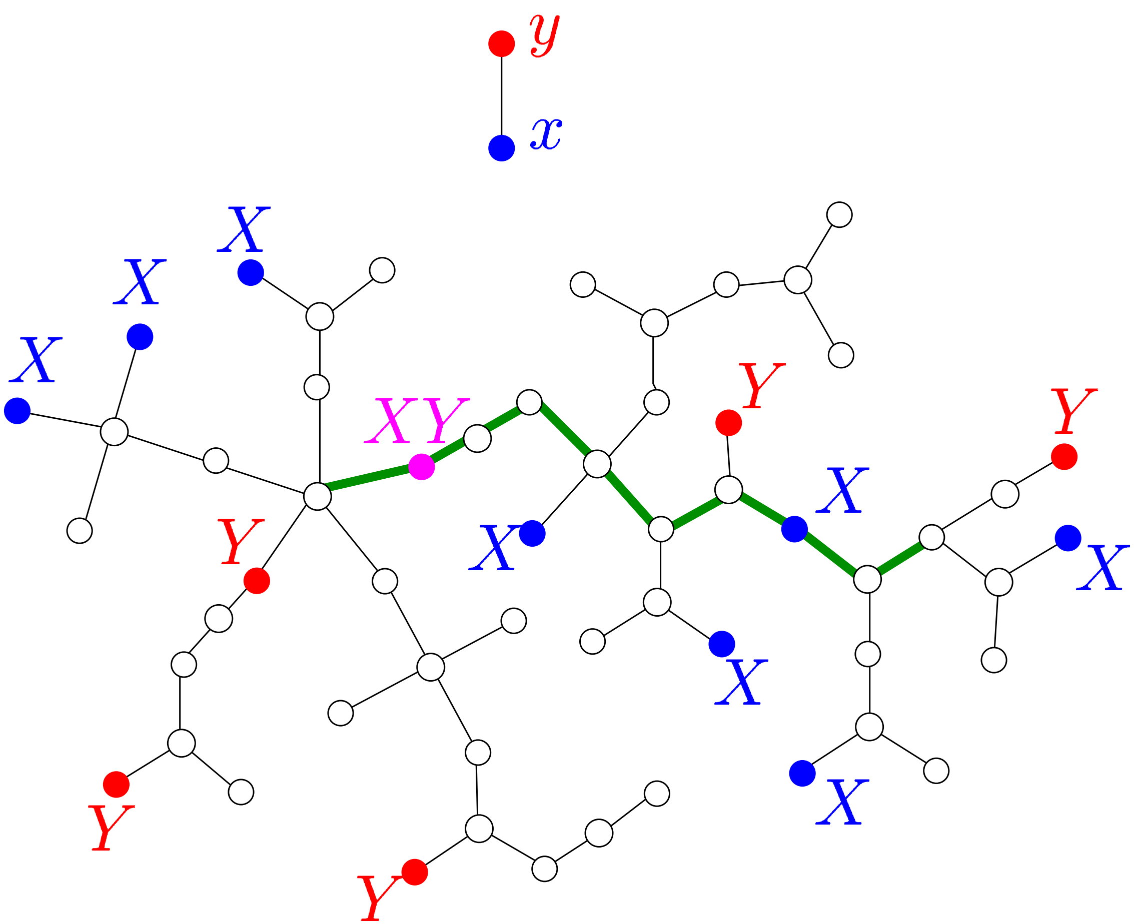




At Most 2 Terminals

Answer YES

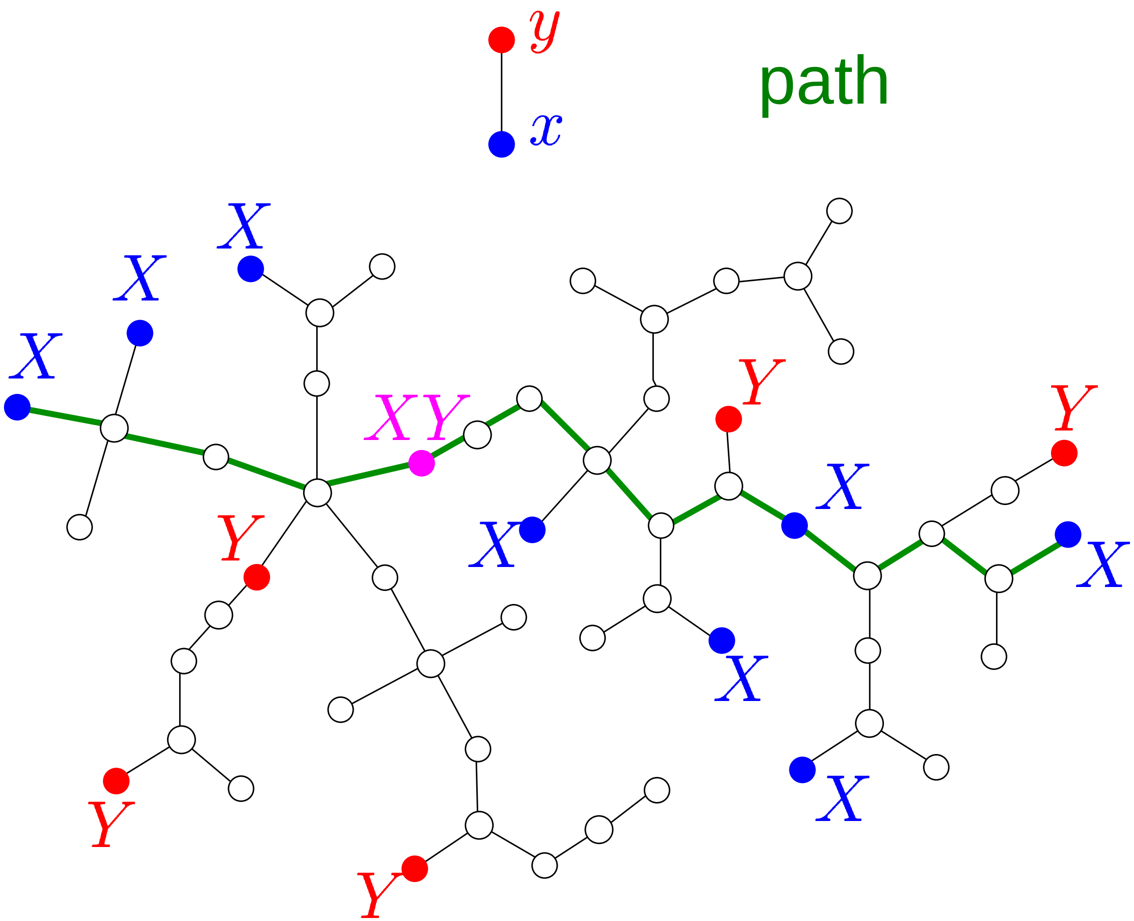
≤ 2 terminals $\Rightarrow$ path in G



At Most 2 Terminals

Answer YES

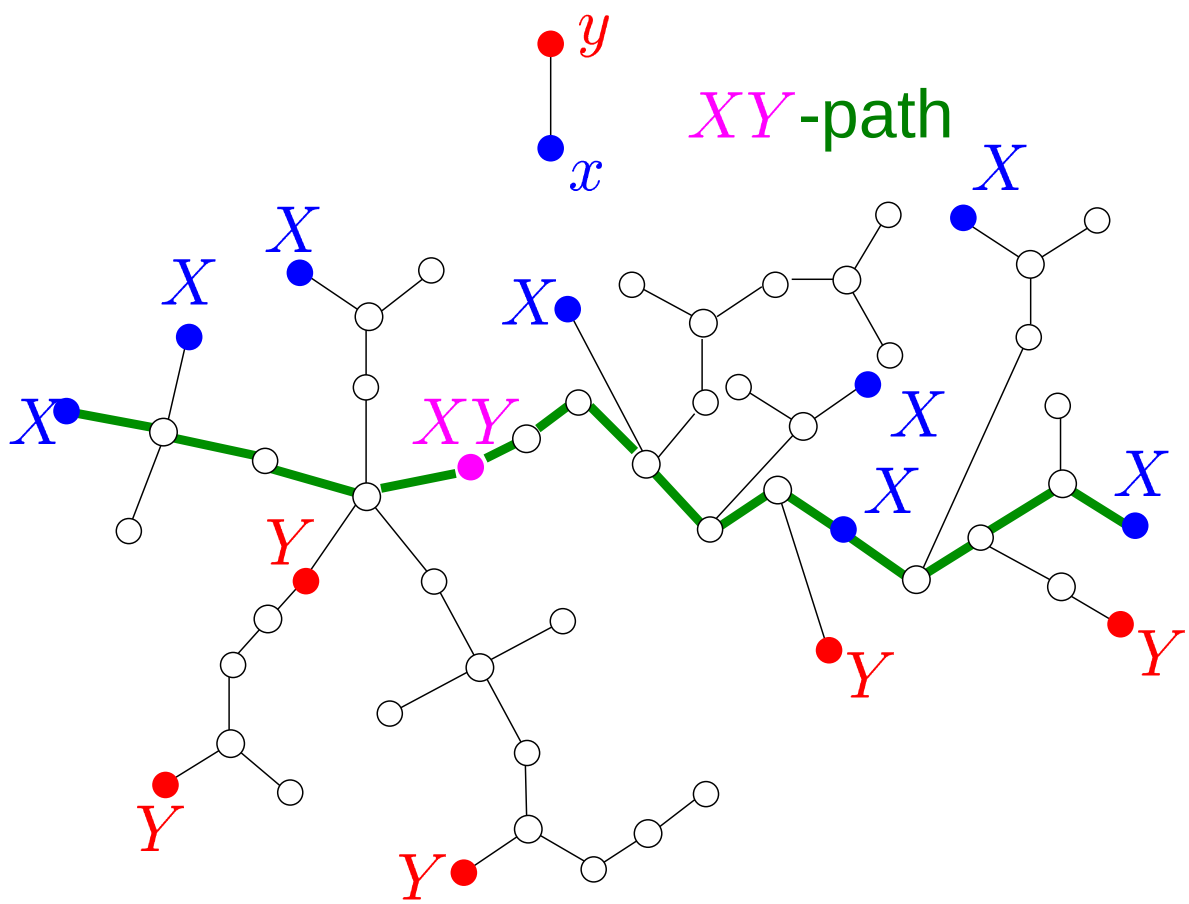
≤ 2 terminals $\Rightarrow$ path in G connecting vertices in X



At Most 2 Terminals

Answer YES

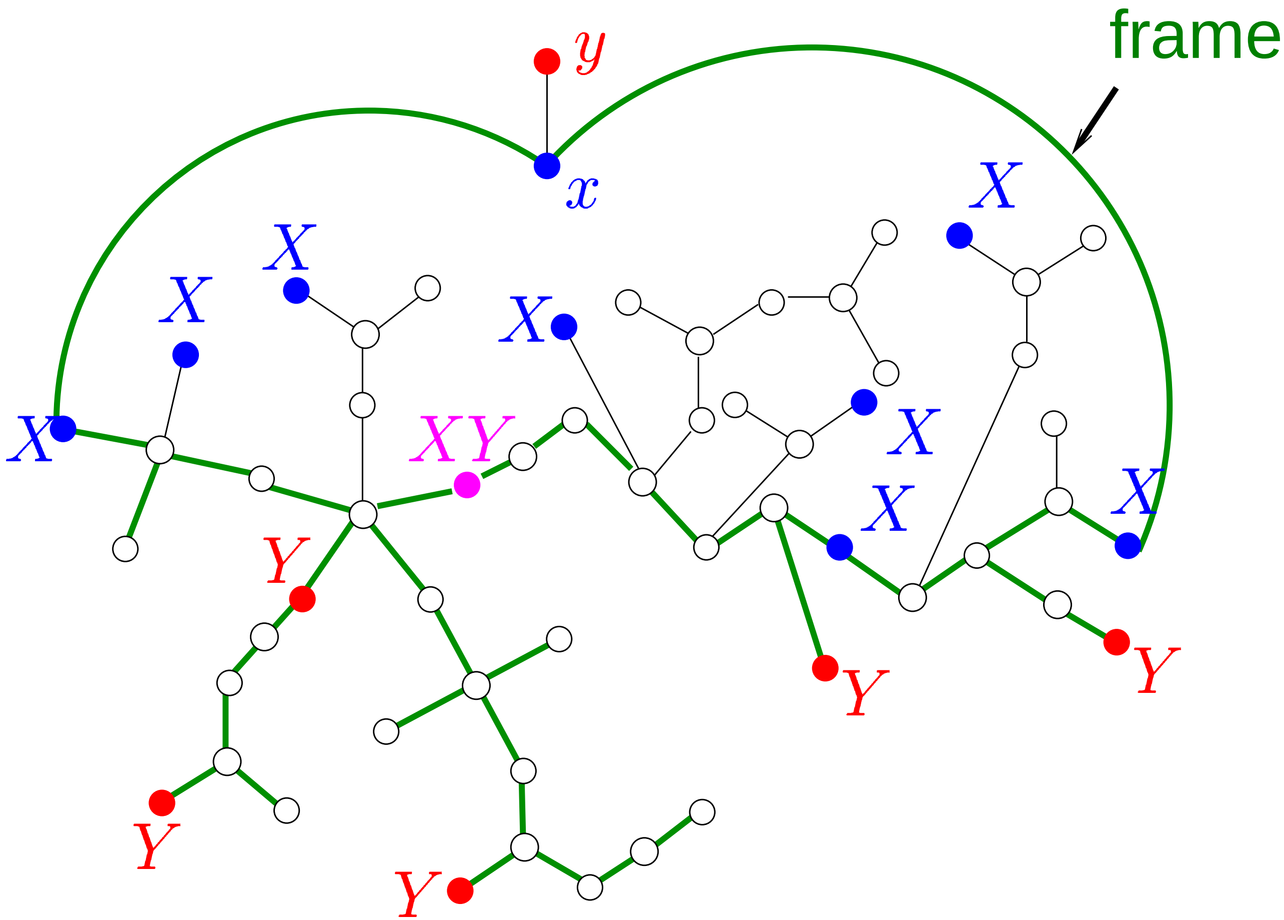
≤ 2 terminals $\Rightarrow$ XY-path



At Most 2 Terminals

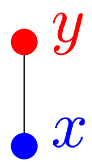
Answer YES

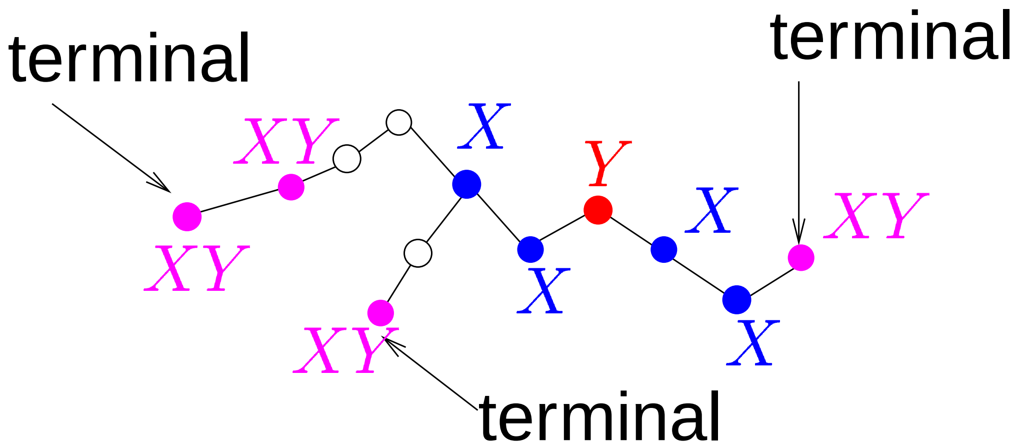
≤ 2 terminals $\Rightarrow$ frame



3 Terminals

Answer is NO
Why?

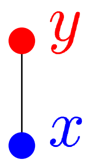


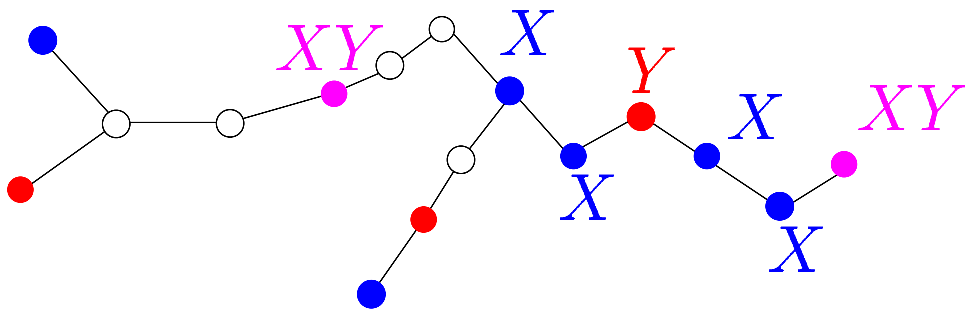


3 Terminals

Answer is NO

In T we have:

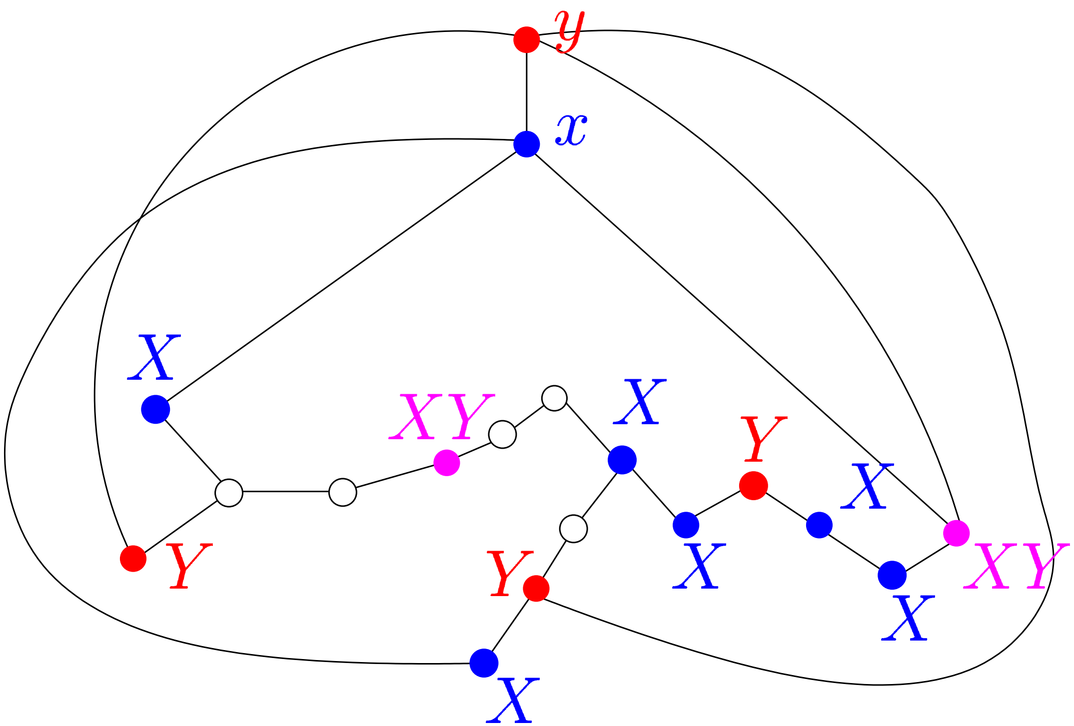




3 Terminals

Answer is NO

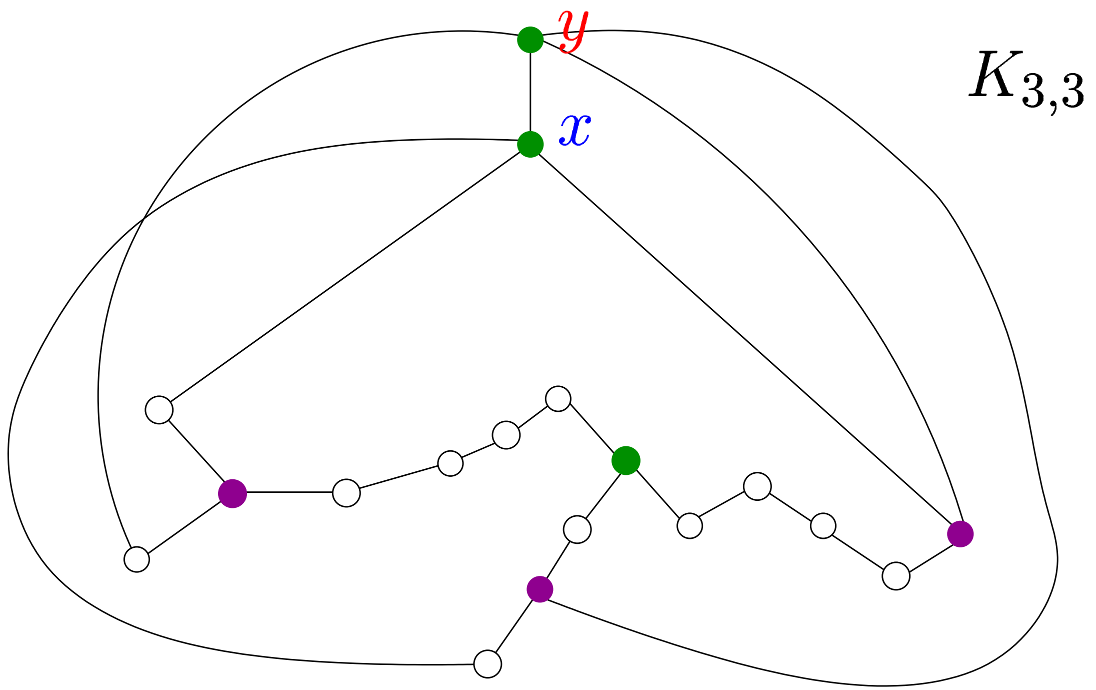
G nonplanar $\Rightarrow$ NO



3 Terminals

Answer is NO

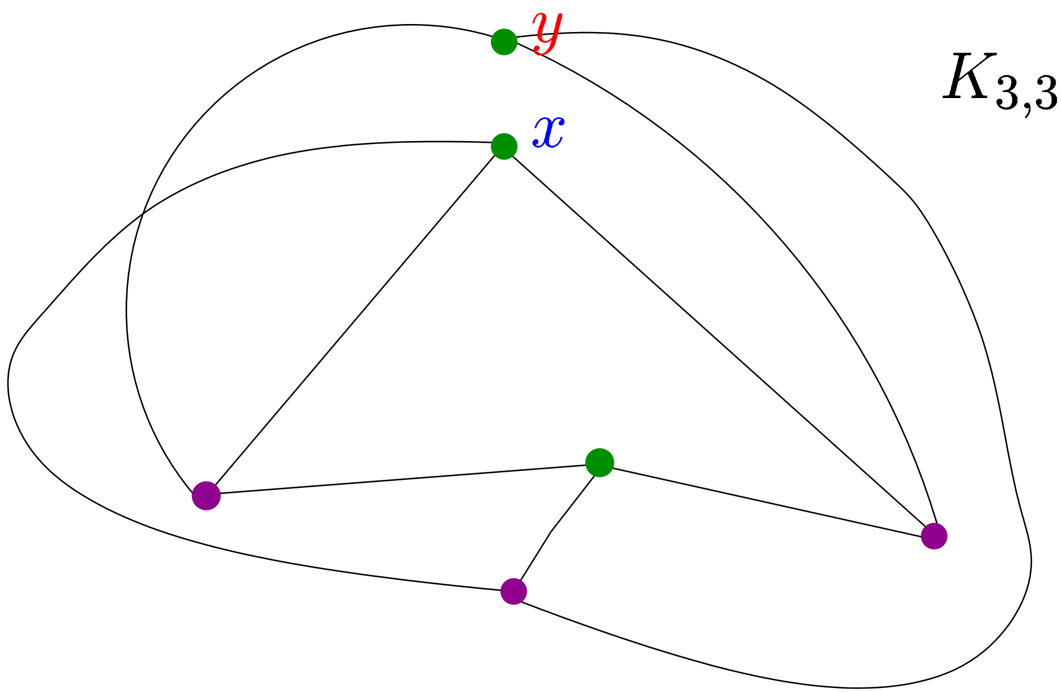
G nonplanar $\Rightarrow$ NO



3 Terminals

Answer is NO

G nonplanar $\Rightarrow$ NO



Algorithm

Each iteration begins with: T, X, Y
Each iteration consists of:

Case 1: All leaves of T are in $X \cap Y$

terminals := leaves of T

Case 1A: T has ≤ 2 terminals

Return YES and stop

Case 1B: T has 3 terminals

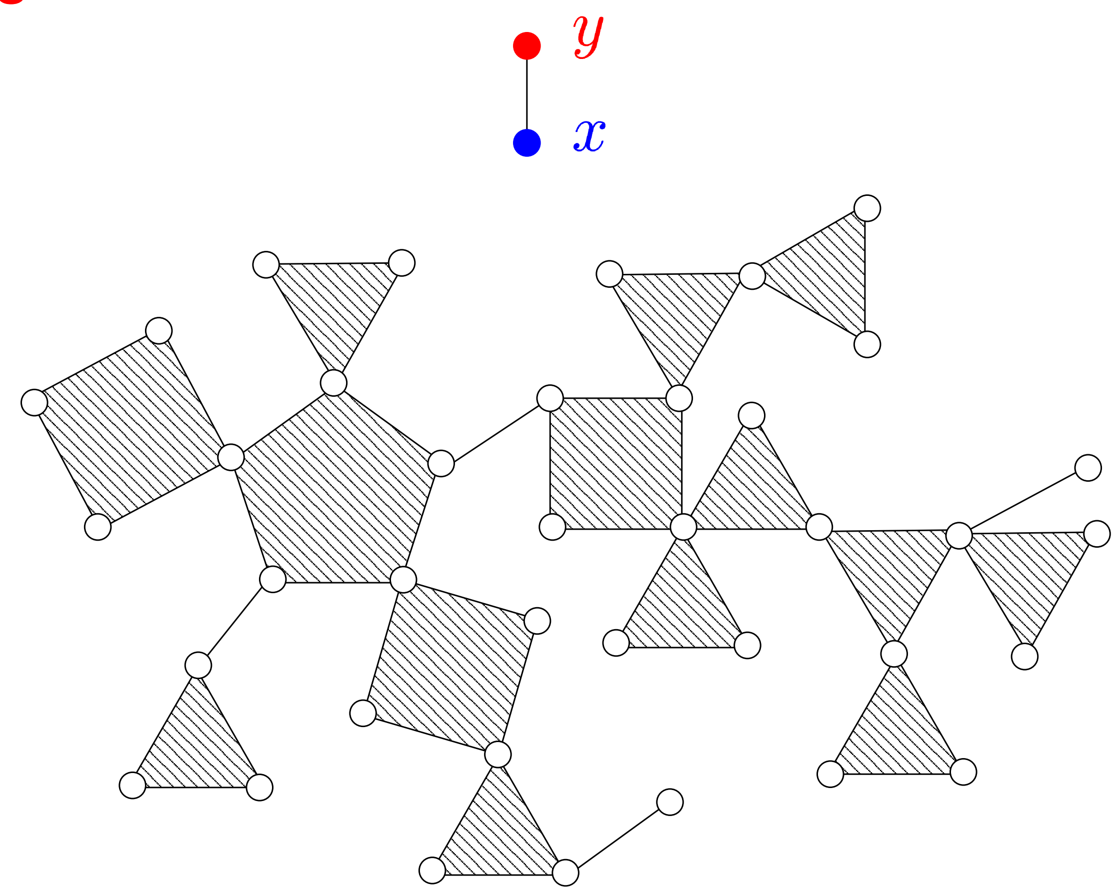
Return NO and stop $\triangleright$ G has $K_{3,3} \Rightarrow$ nonplanar

Case 2: T has a leaf not in $X \cap Y$

$T', X', Y' \leftarrow \textsc{Reduce}(T, X, Y)$
Start anew with T', X', Y' in the role of T, X, Y

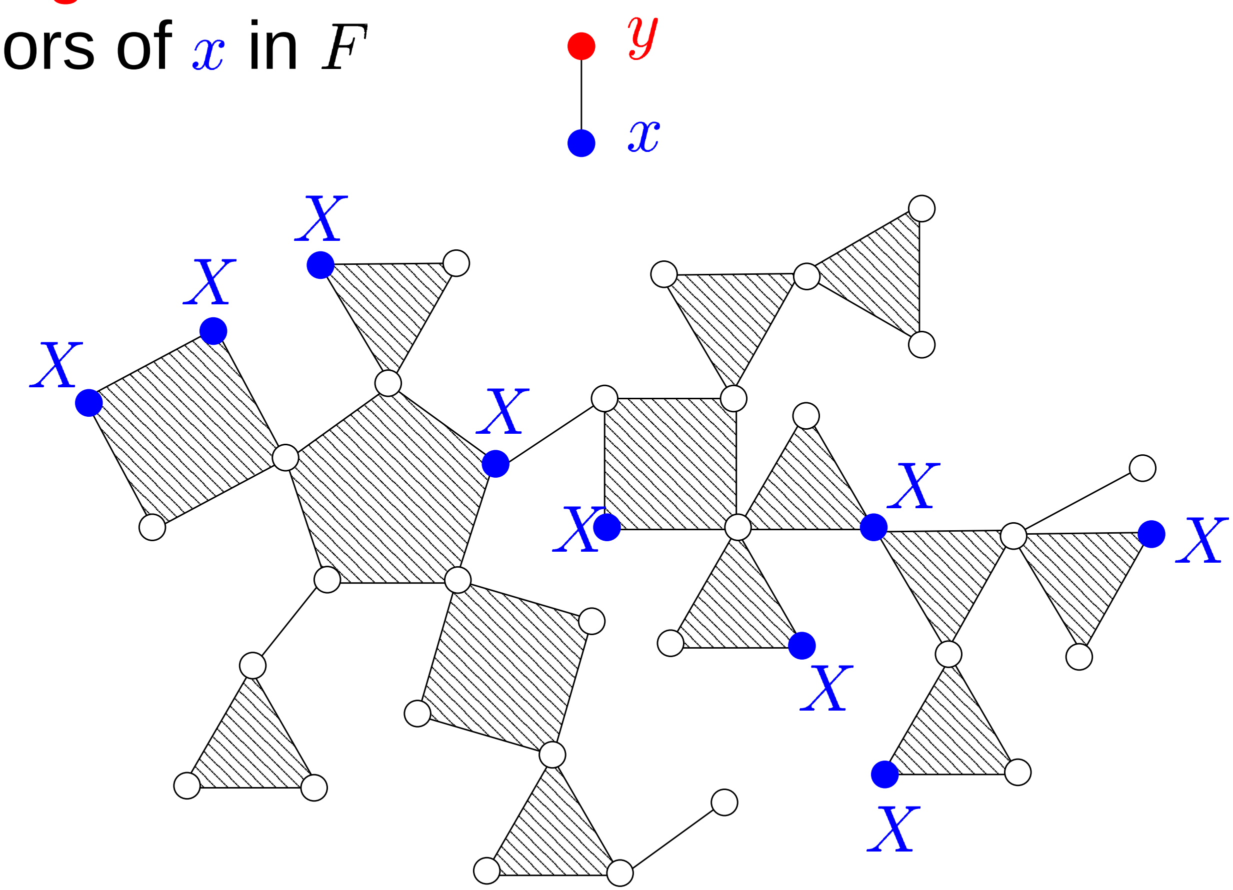
Slightly More General Problem

G graph on vertex set $V_H \cup \{x, y\}$
H planar induced subgraph of G
F frame of H
xy is an edge



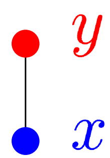
Slightly More General Problem

G graph on vertex set $V_H \cup \{x, y\}$
H planar induced subgraph of G
F frame of H
xy is an edge
X neighbors of x in F

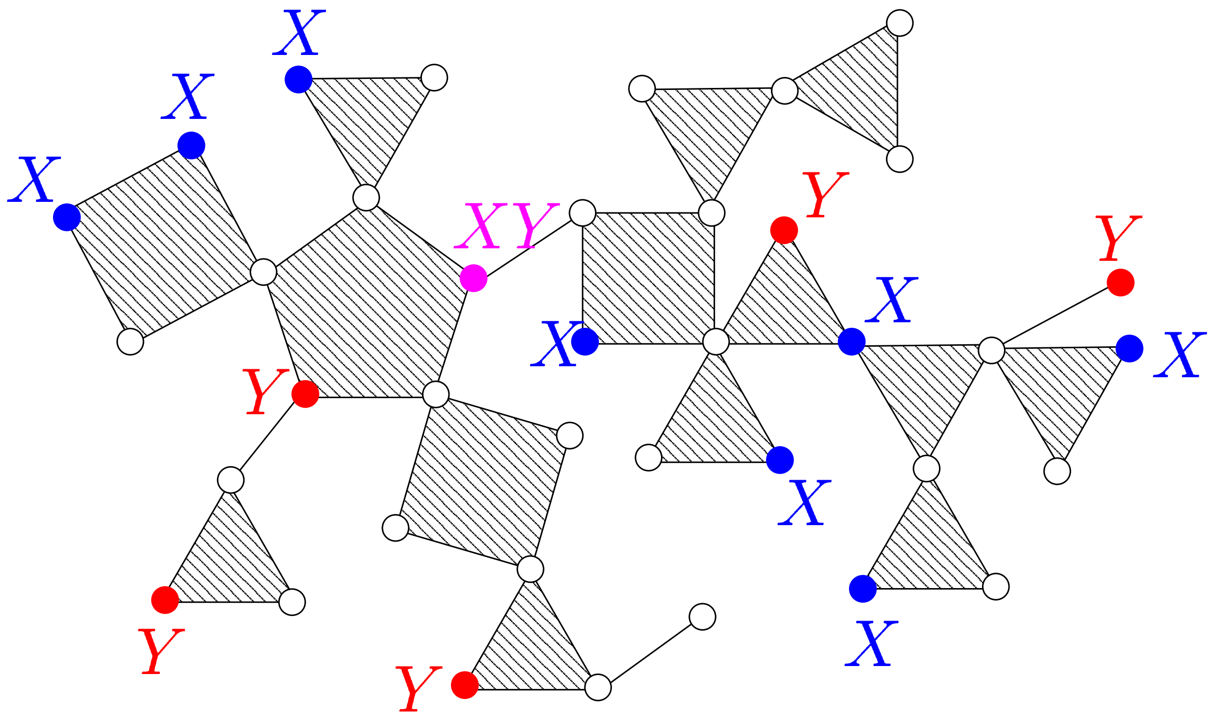


Slightly More General Problem

G graph on vertex set $V_H \cup \{x, y\}$
H planar induced subgraph of G
F frame of H
xy is an edge
X neighbors of x in F
Y neighbors of y in F

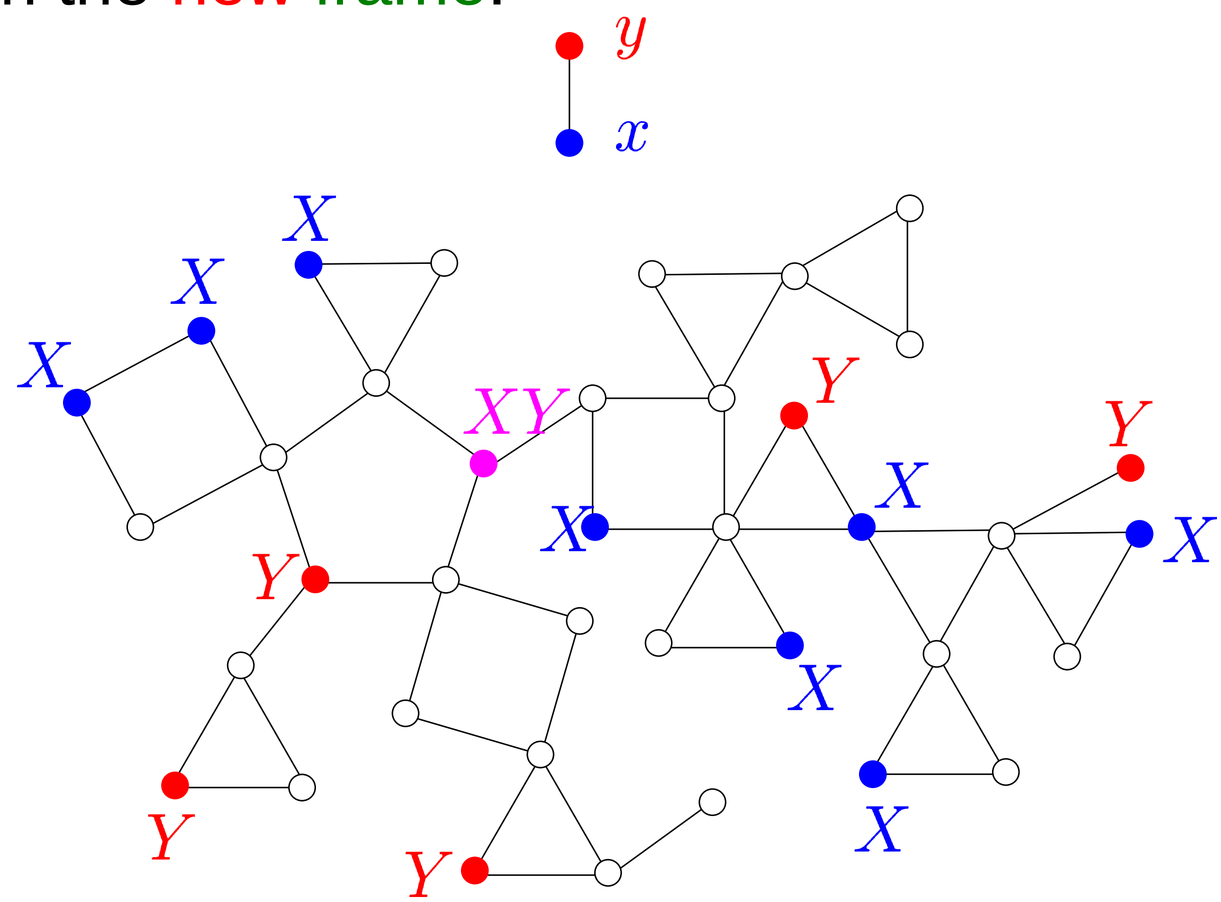


$XY = X \cap Y$



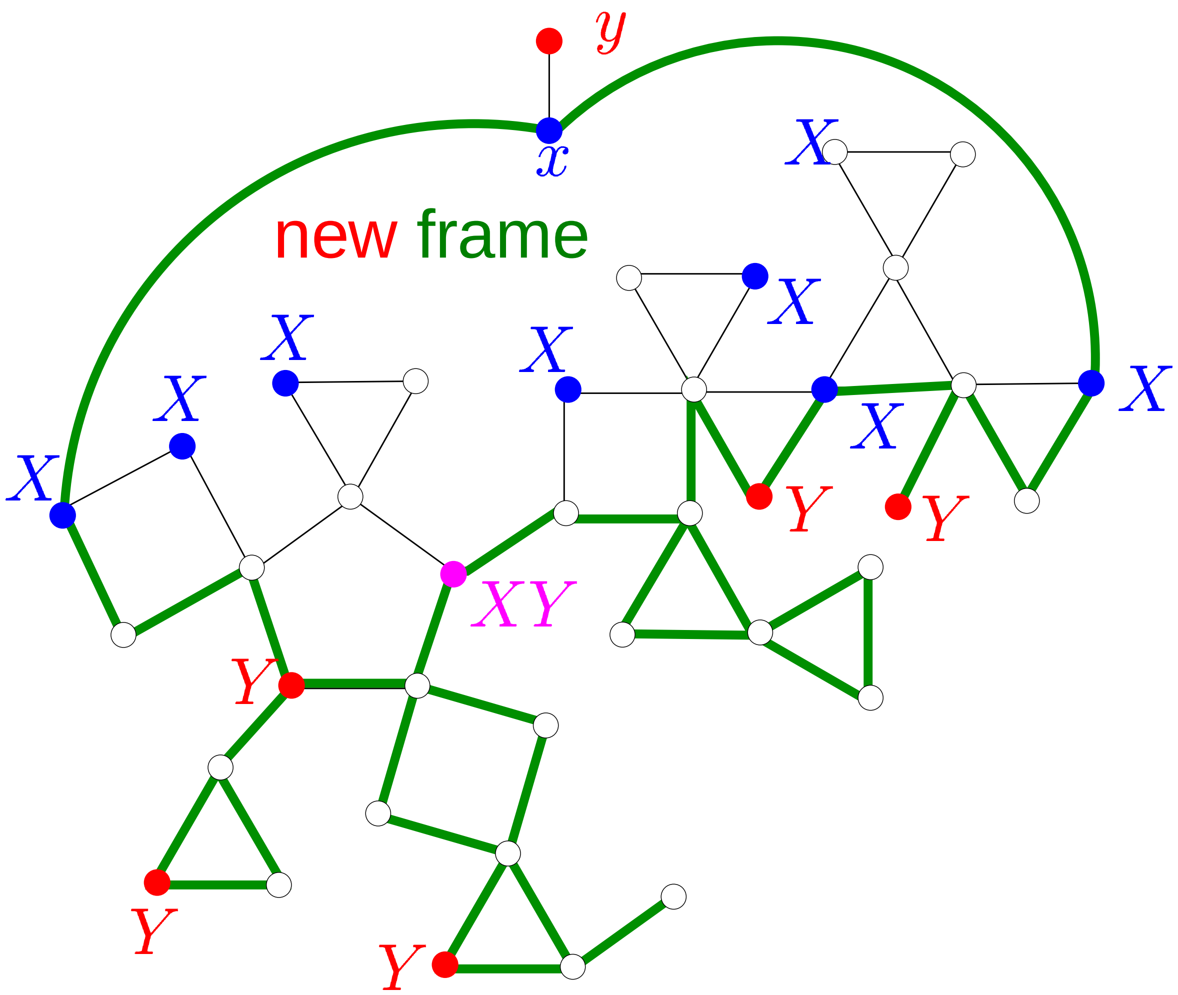
Slightly More General Problem

Give an algorithm that receives F, xy, X, Y and decides whether there is an embedding of $F + x$ leaving all Y vertices on the new frame.



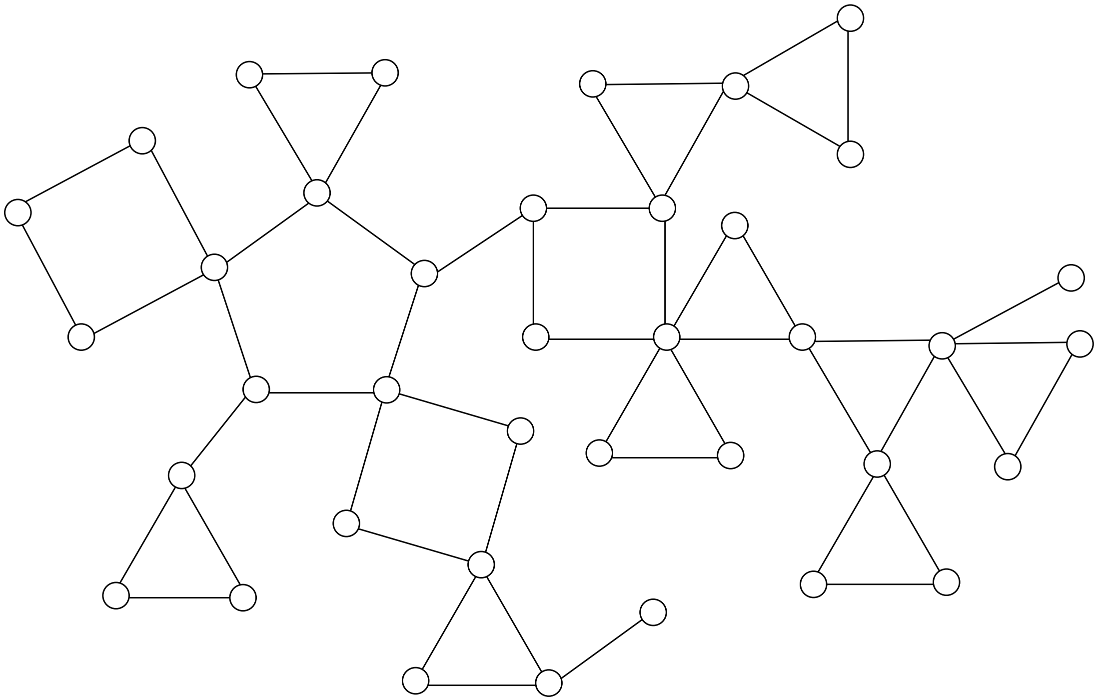
YES $\Leftrightarrow$ G is planar

Slightly More General Problem



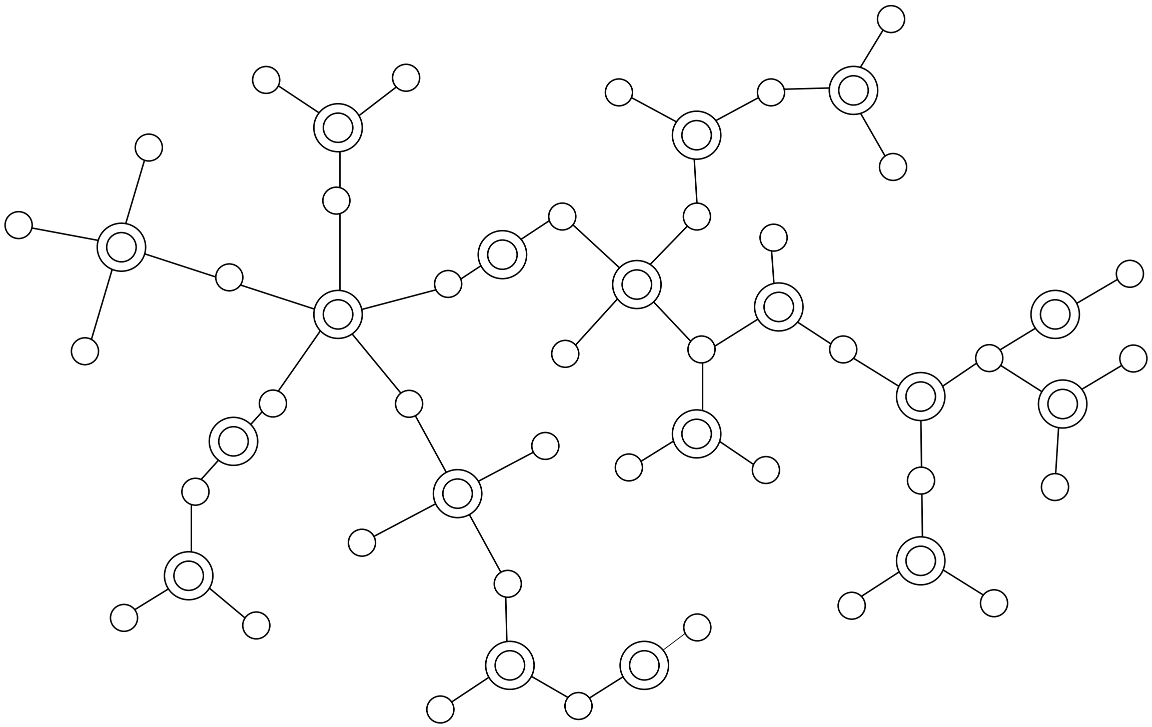
YES $\Leftrightarrow$ G is planar

Block Tree



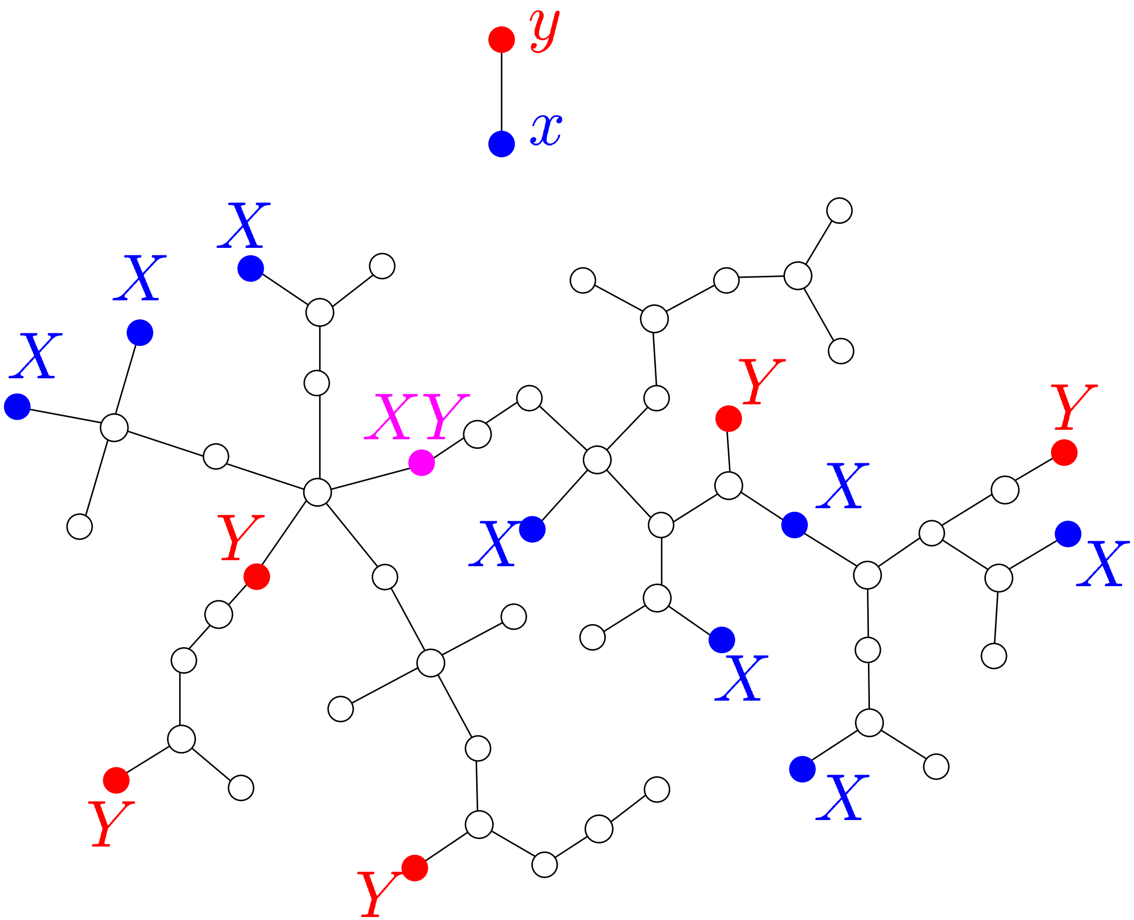
Block Tree

P-node := vertex of F
C-node := block of F



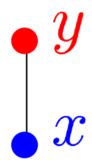
Apply Previous Algorithm

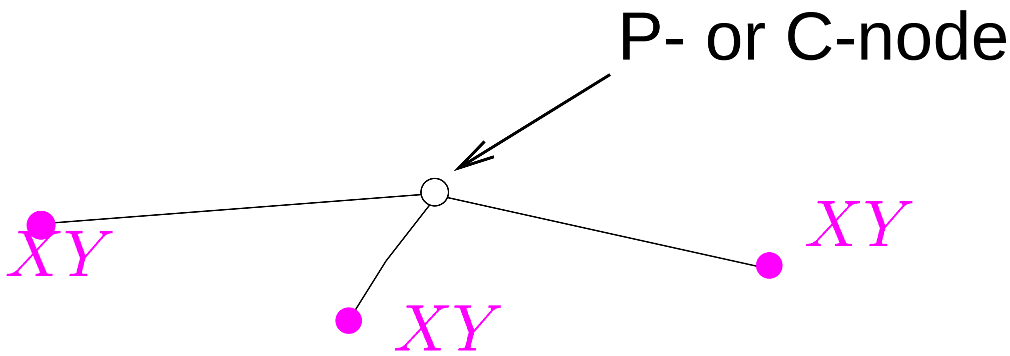
Apply the previous algorithm to the block tree T of F



Apply Previous Algorithm

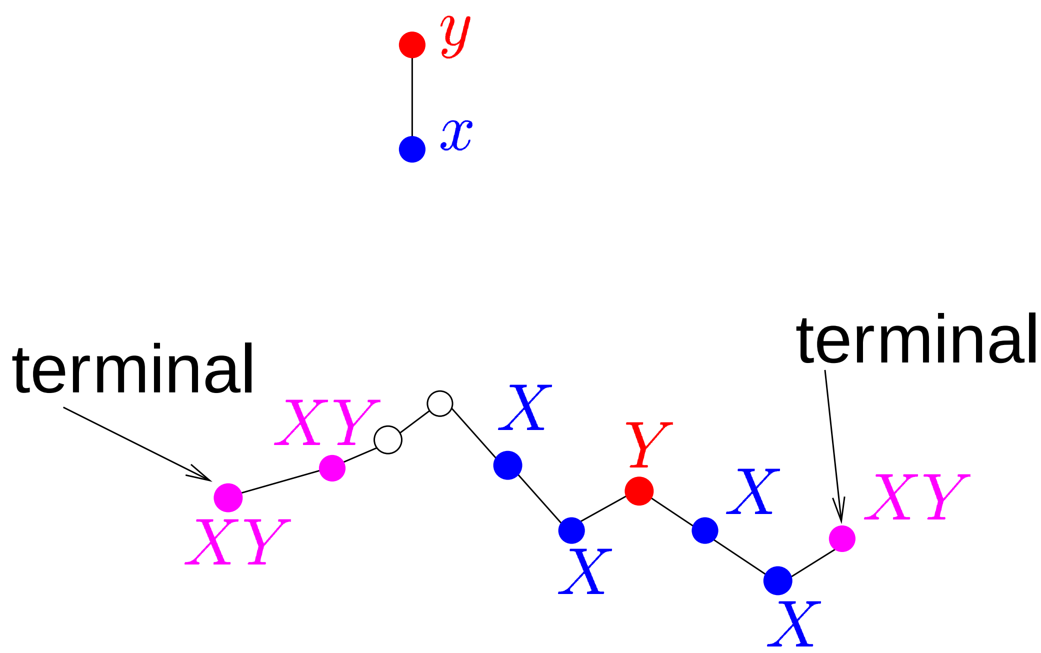
NO $\Rightarrow$ NO





Apply Previous Algorithm

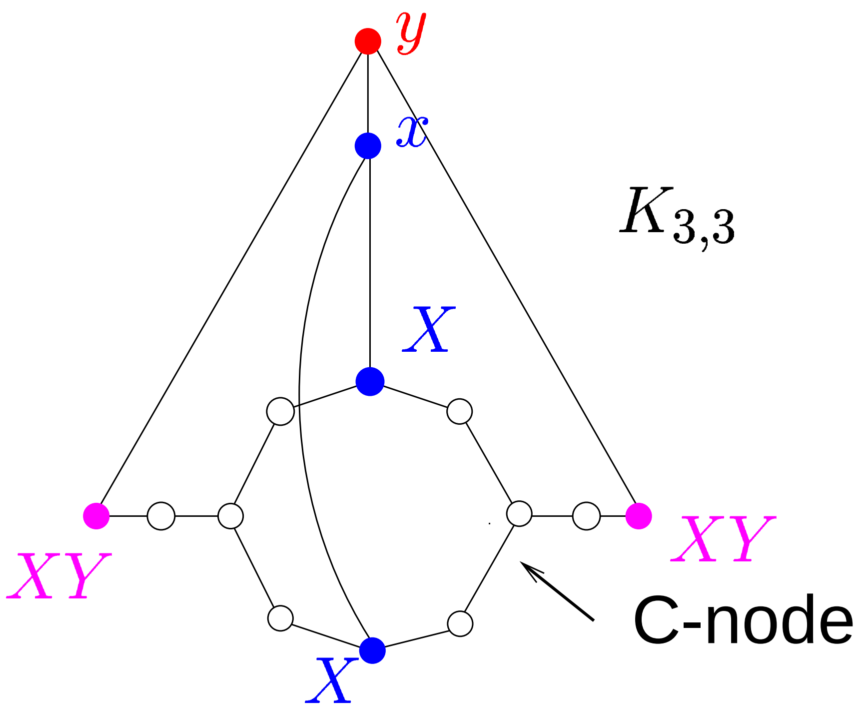
YES $\Rightarrow$ PERHAPS



Apply Previous Algorithm

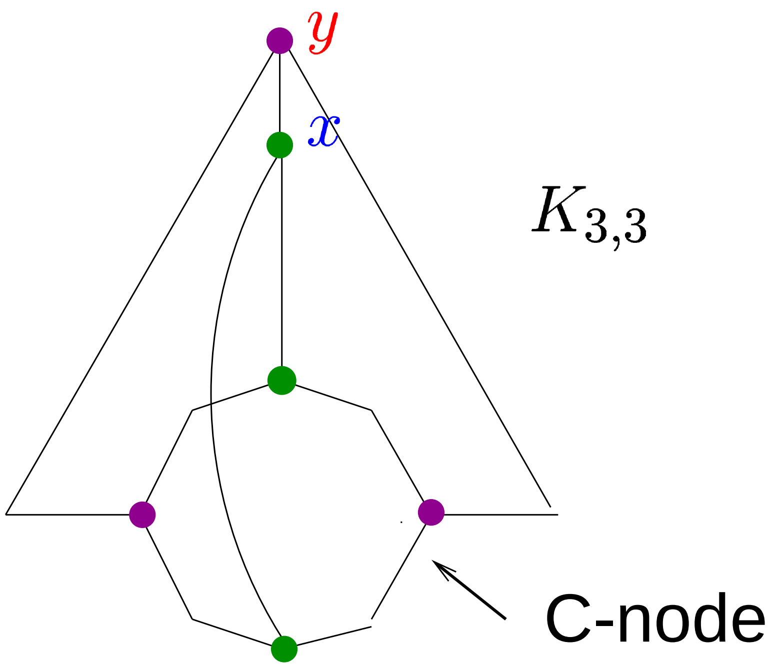
YES $\Rightarrow$ PERHAPS

G nonplanar $\Rightarrow$ NO



Apply Previous Algorithm

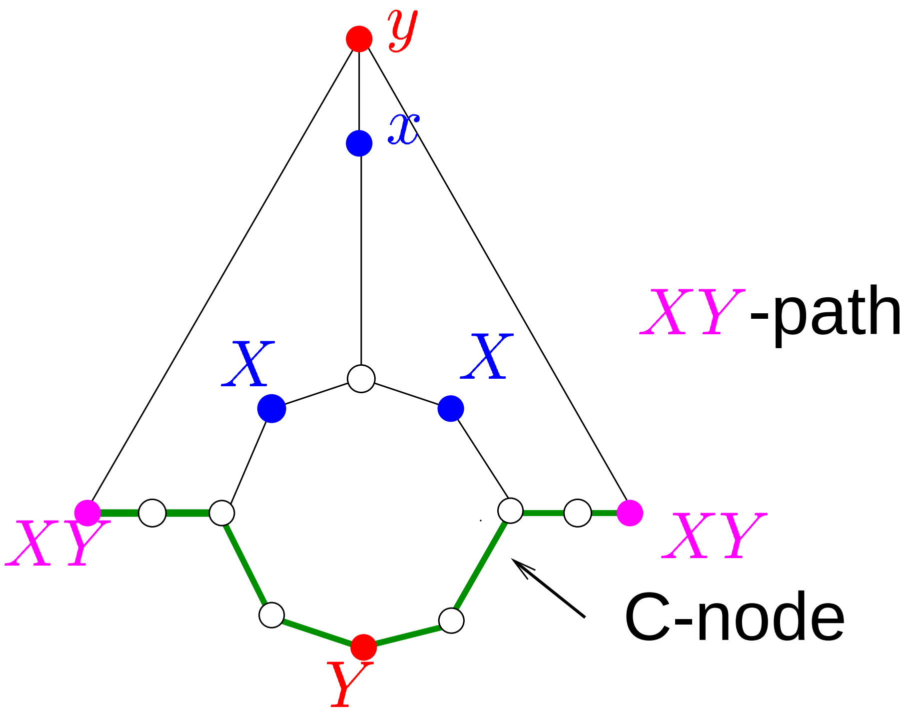
YES $\Rightarrow$ PERHAPS
G nonplanar $\Rightarrow$ NO



Apply Previous Algorithm

YES $\Rightarrow$ PERHAPS

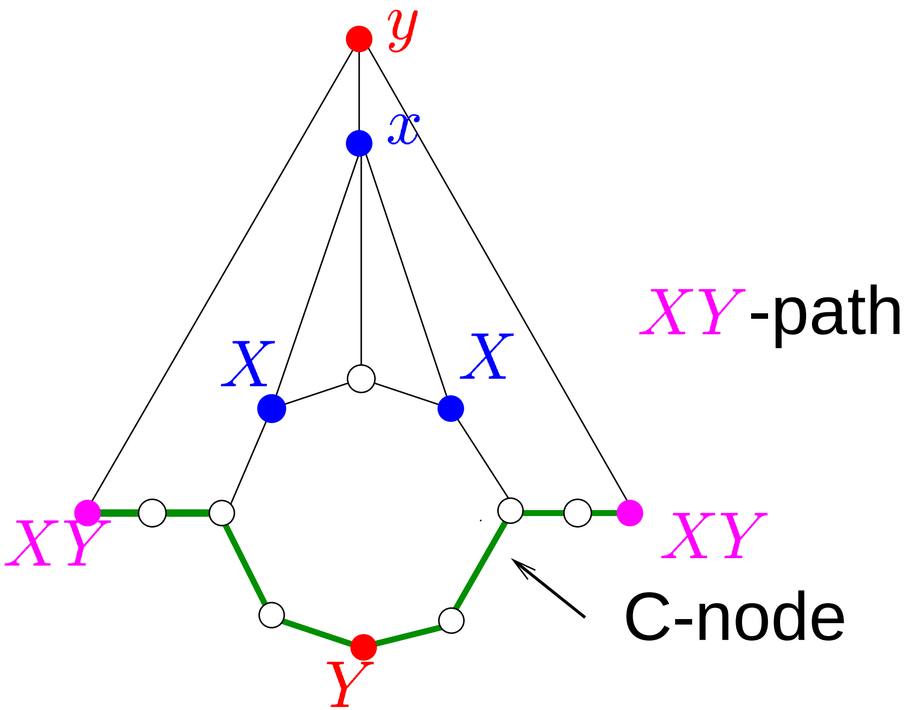
G planar $\Rightarrow$ YES



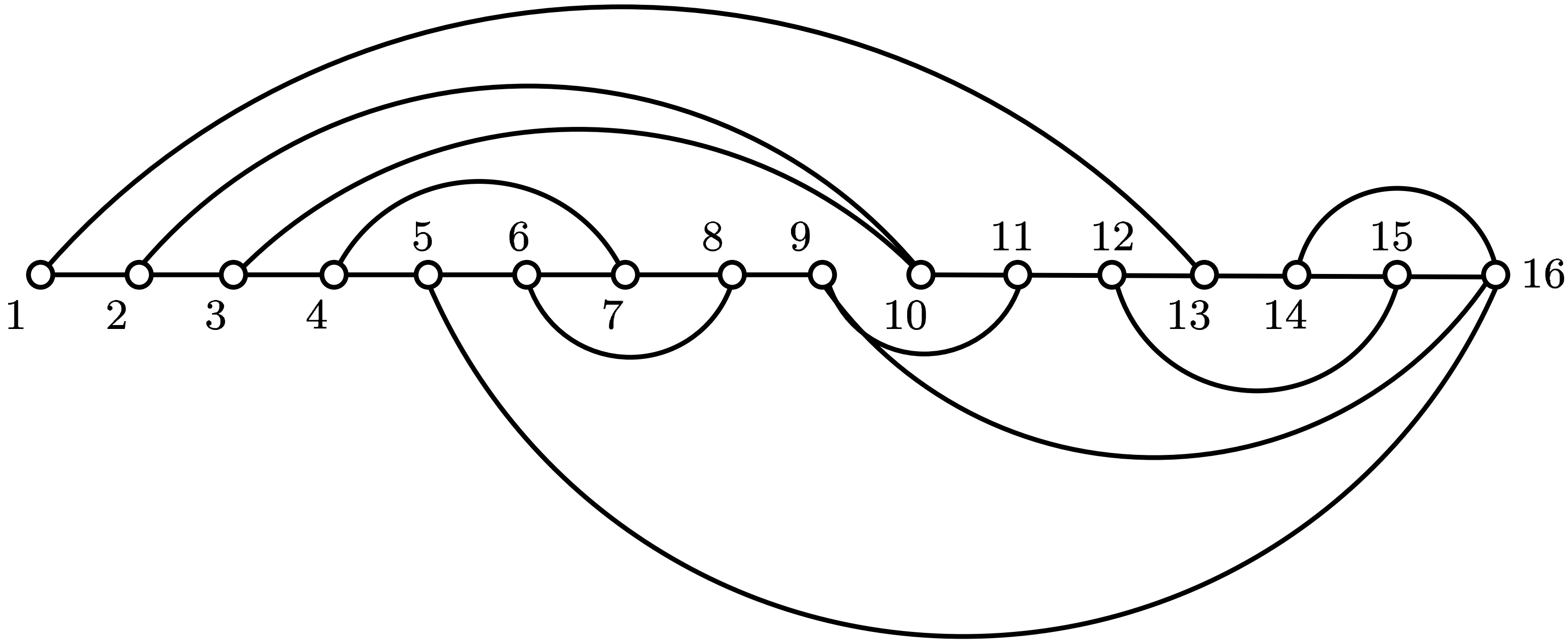
Apply Previous Algorithm

YES $\Rightarrow$ PERHAPS

G planar $\Rightarrow$ YES



LEC-numbering



Property: For each i, $G[1\,..\,i-1]$ and $G[i\,..\,n]$ are connected.

LEC Method

Receives a biconnected graph G and returns YES if G is planar, and NO otherwise.

Number the vertices of G according to an LEC-numbering

Each iteration begins with:

- H planar induced subgraph of G
- F frame of H

At beginning of the 1st iteration:

$$H = \emptyset \qquad F = \emptyset$$

LEC Method

Each iteration consists of:

Case 1: $H = G$

Return YES and stop

Case 2: $H \neq G$

$x \leftarrow$ smallest numbered vertex in $G - V_H$
$X \leftarrow$ neighbors of x in F
$Y \leftarrow$ neighbors of $V_G - (V_H + x)$ in F $\quad \triangleright\ y$

Case 2A: There is no XY-path in F

Return NO and stop

Case 2B: There is an XY-path in F

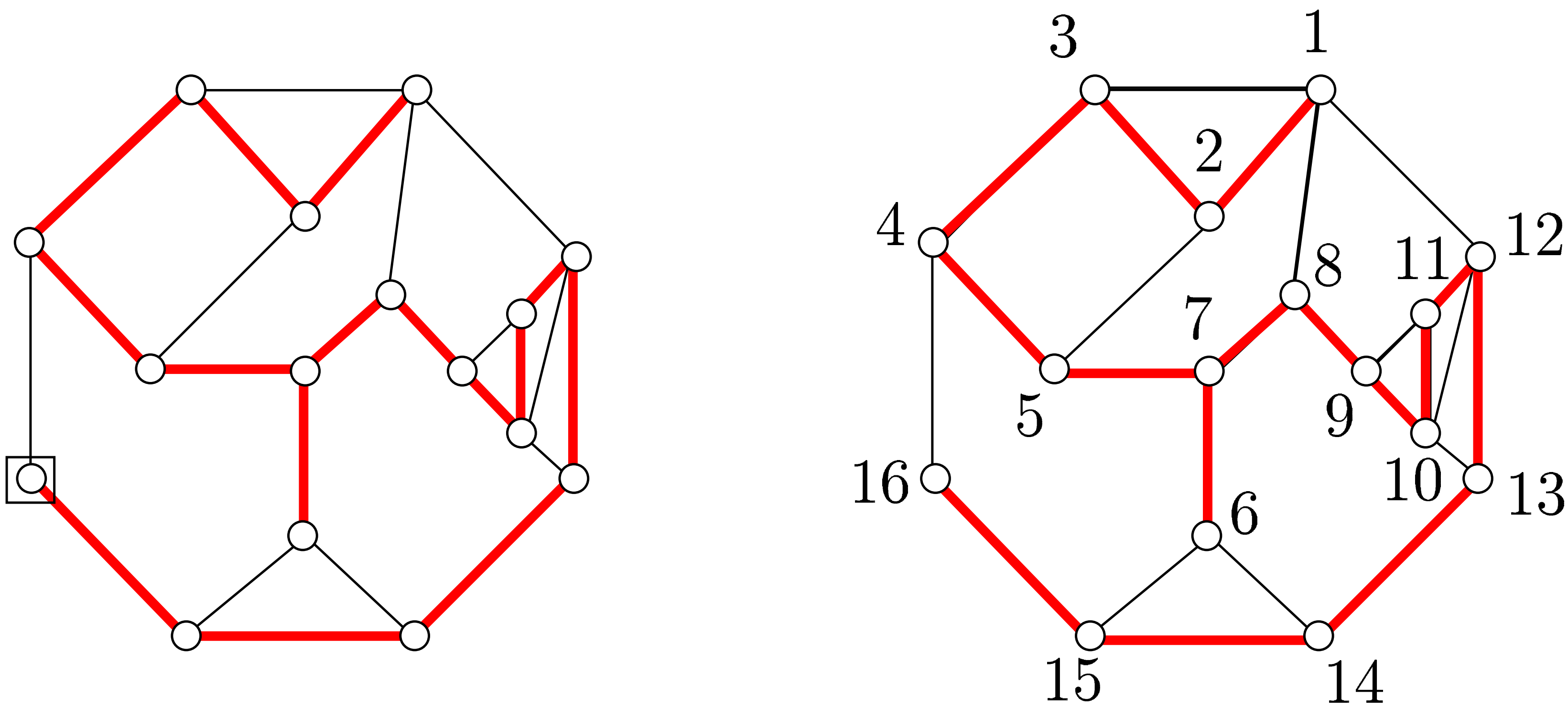
$H', F' \leftarrow \textsc{Central}(F, X, Y)$
Start anew with H' and F' in the role of H and F

Shih and Hsu implementation

LEC-numbering	$\Longleftrightarrow$	DFS-numbering
block tree of frame	$\Longleftrightarrow$	PC-tree
search for XY-path	$\Longleftrightarrow$	finding terminals fast
updating the frame	$\Longleftrightarrow$	updating the PC-tree

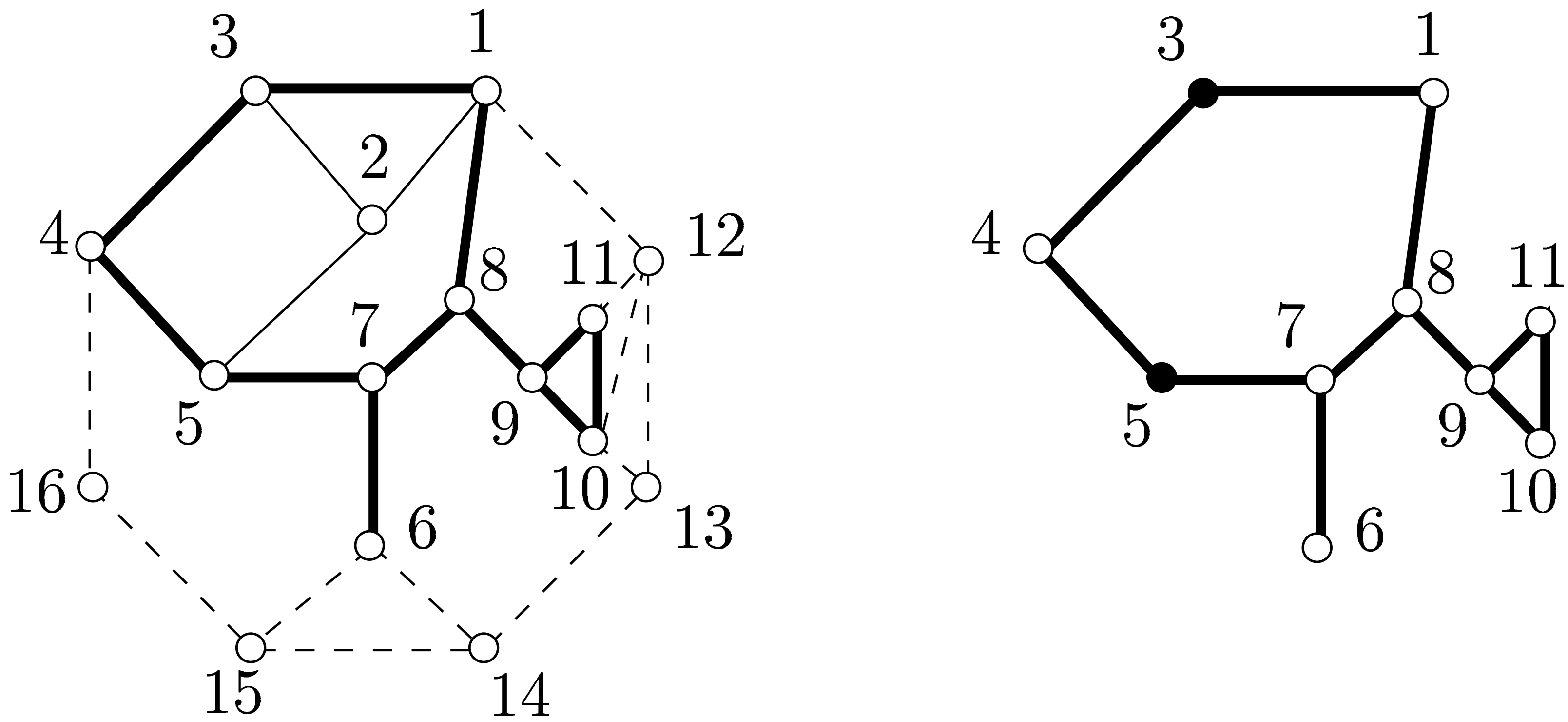
DFS-numbering

Number the vertices according to
a postorder traversal of a DFS-tree of G.

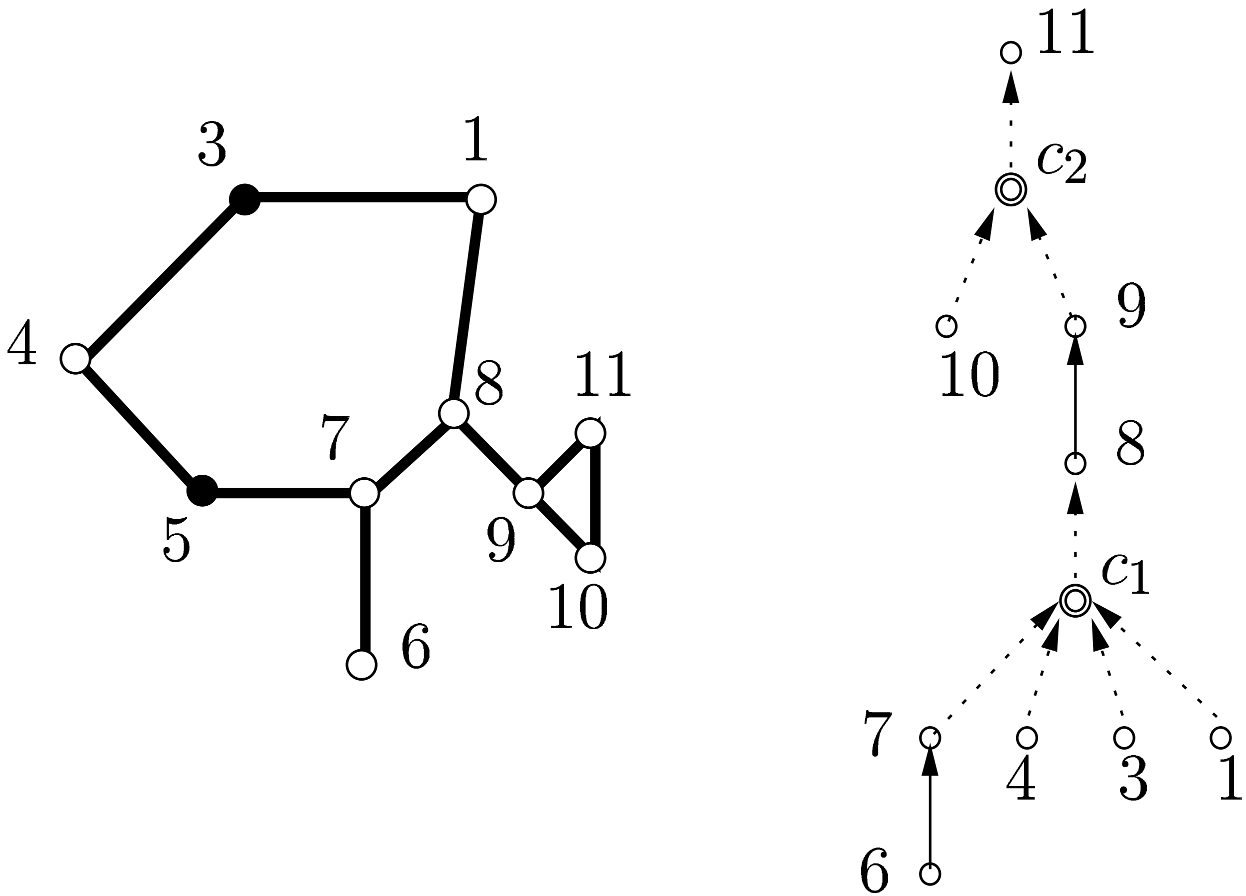


Property: For each i, $G[i \mathinner{..} n]$ is connected.

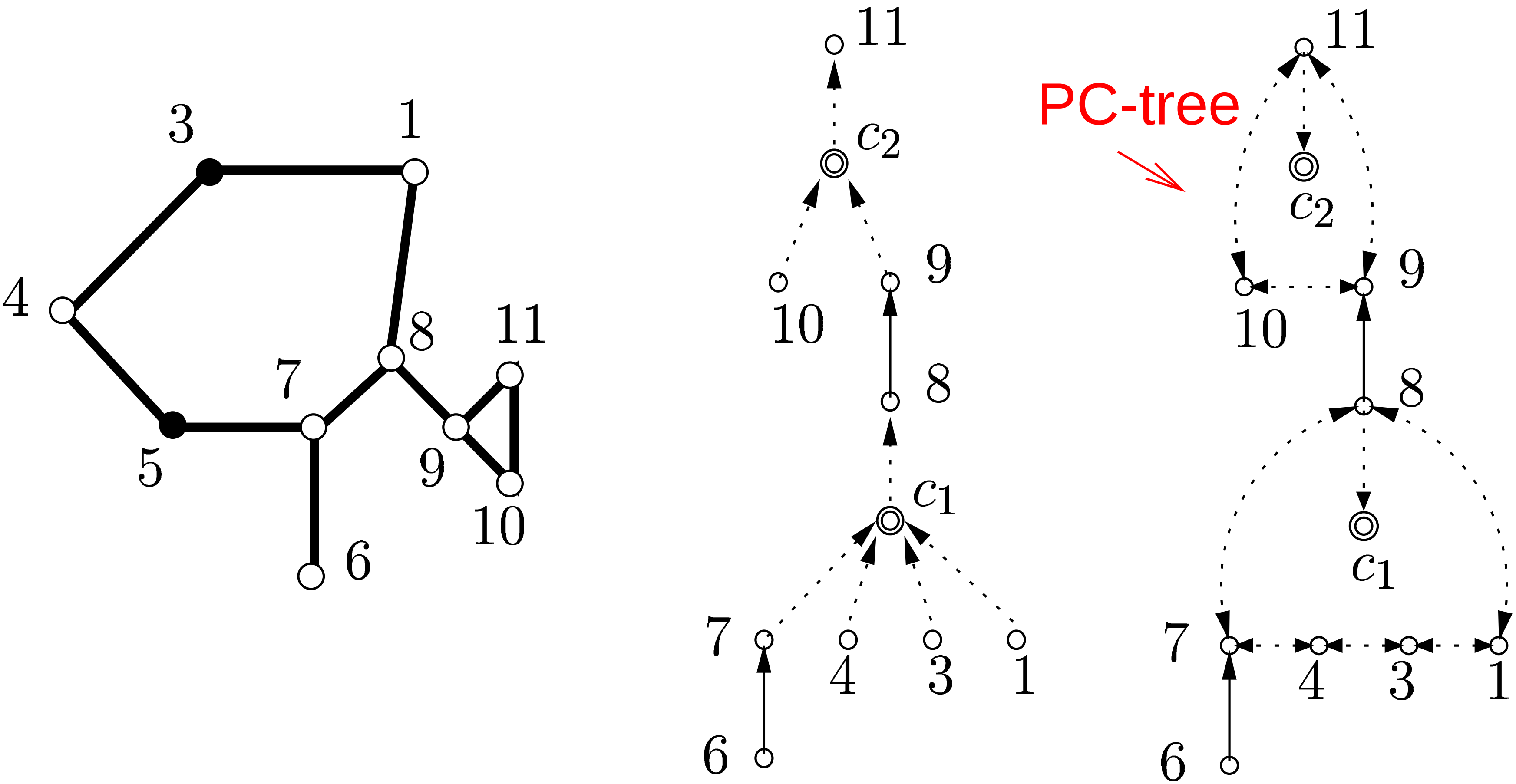
PC-tree



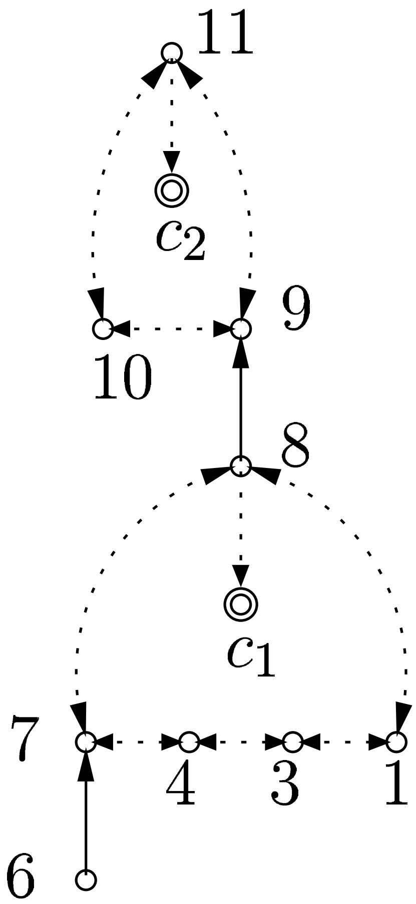
PC-tree

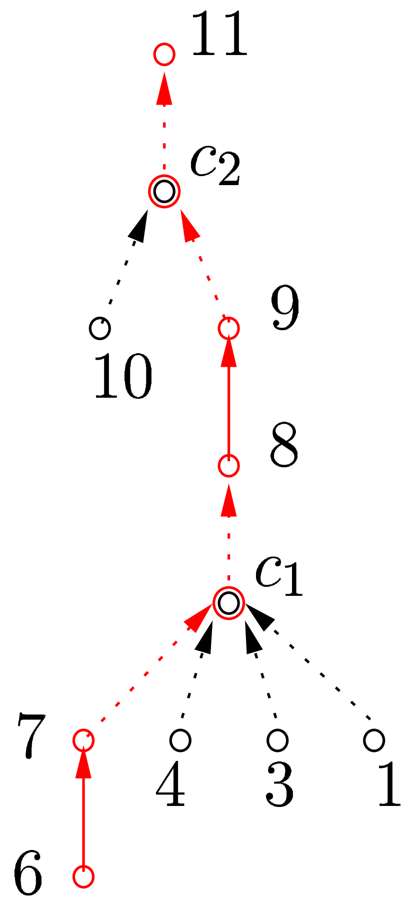


PC-tree

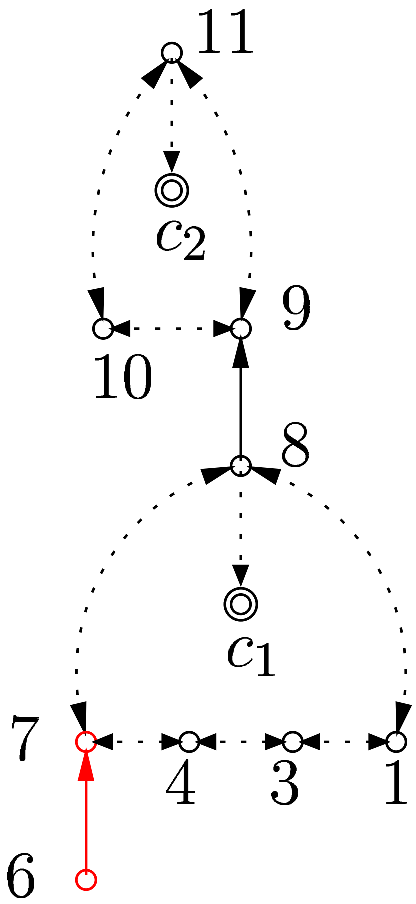


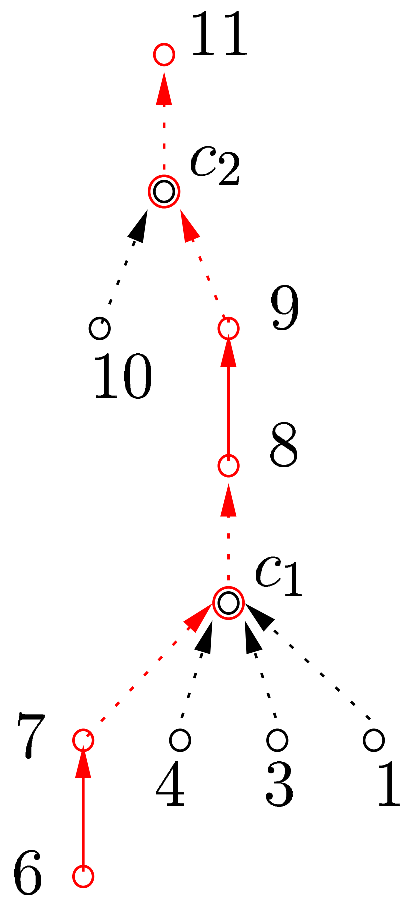
Traversal of the PC-tree



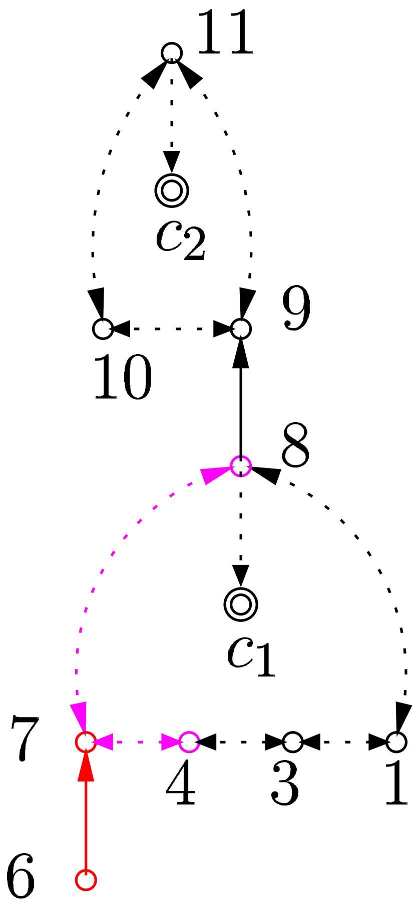


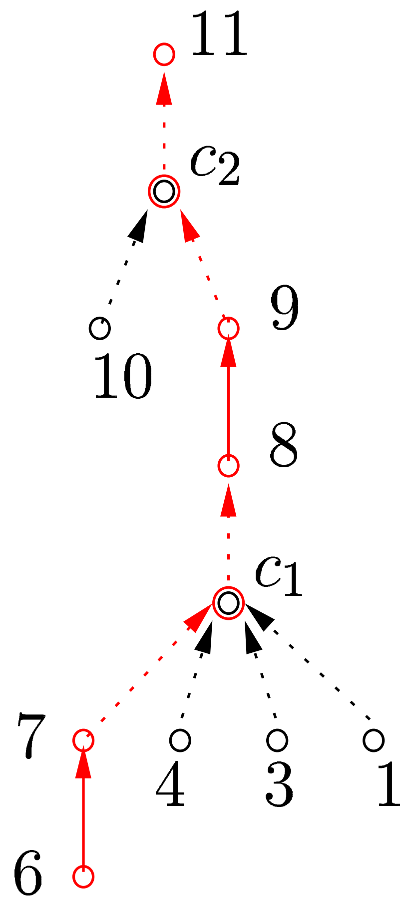
Traversal of the PC-tree



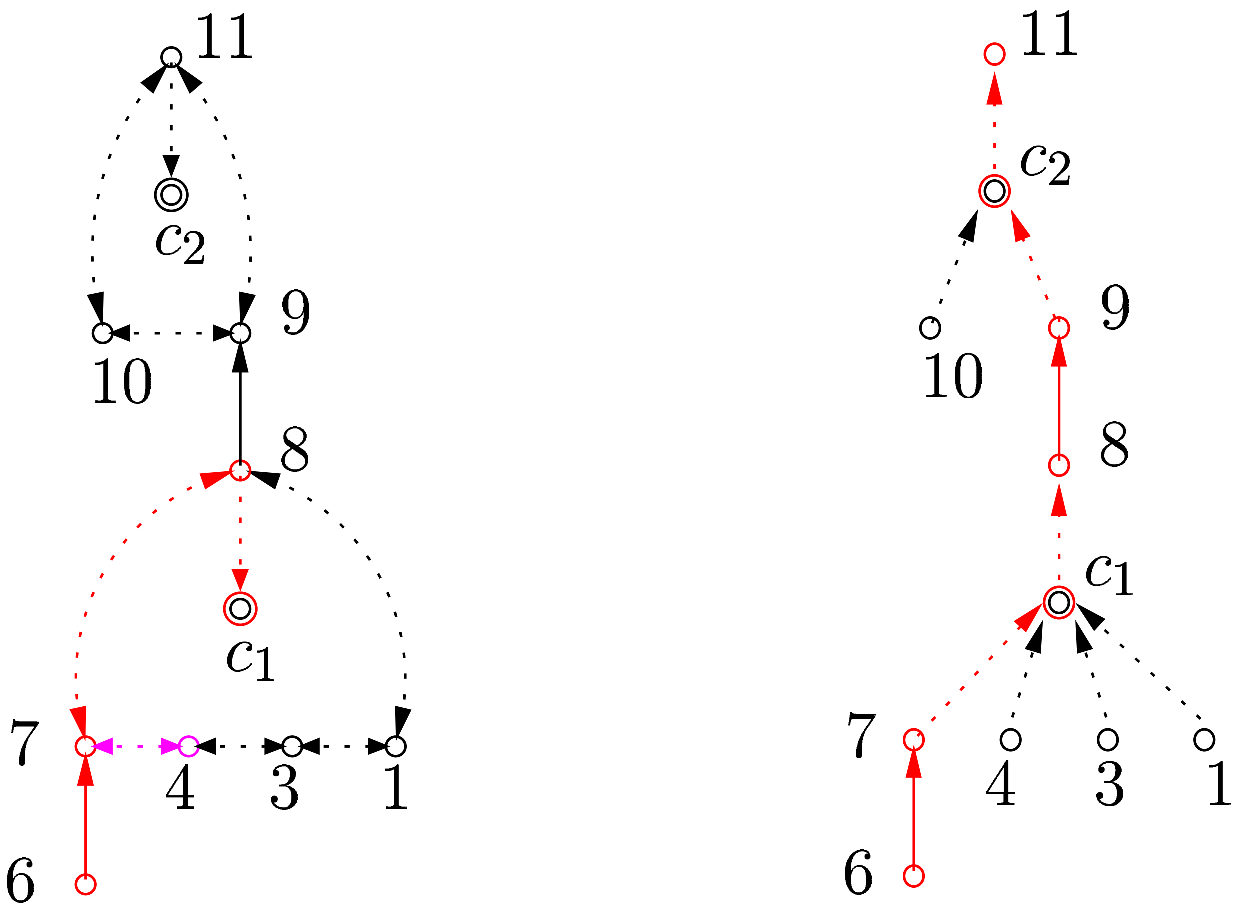


Traversal of the PC-tree

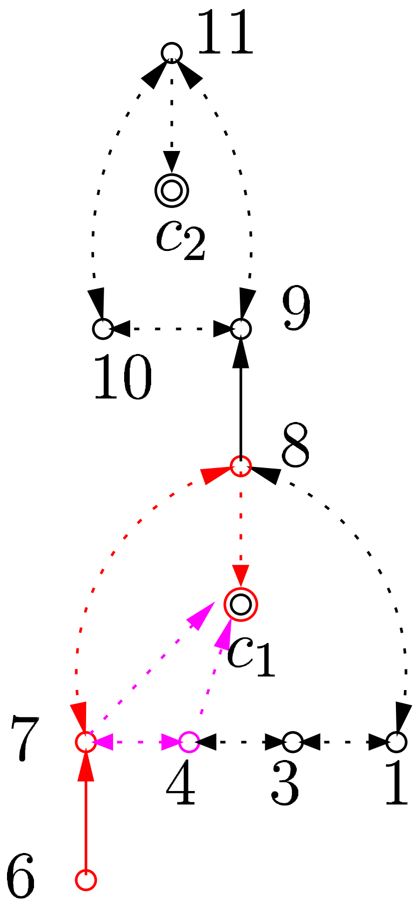


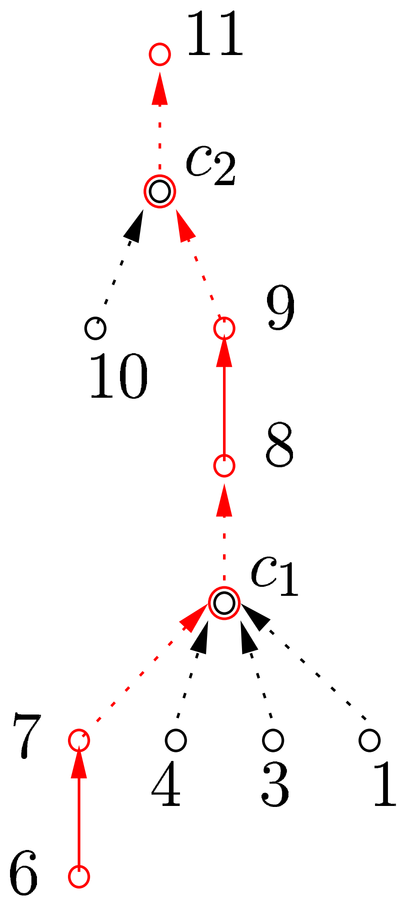


Traversal of the PC-tree

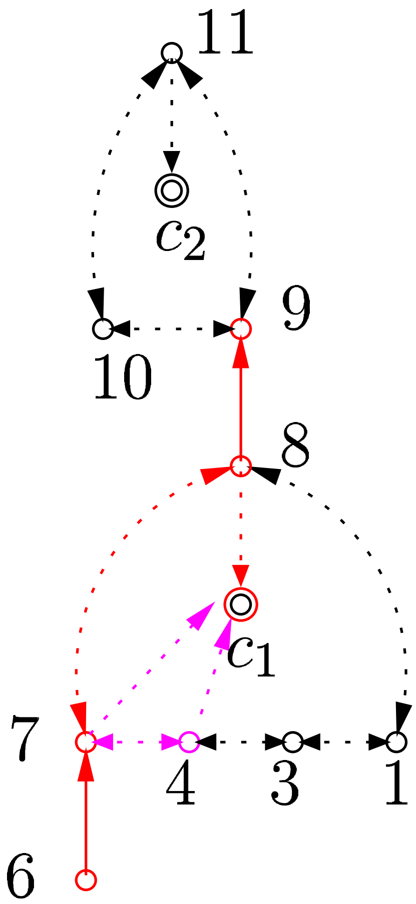


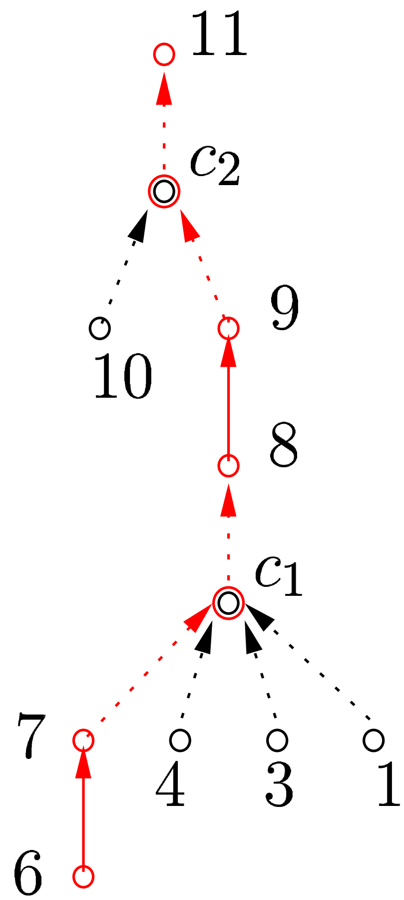
Traversal of the PC-tree



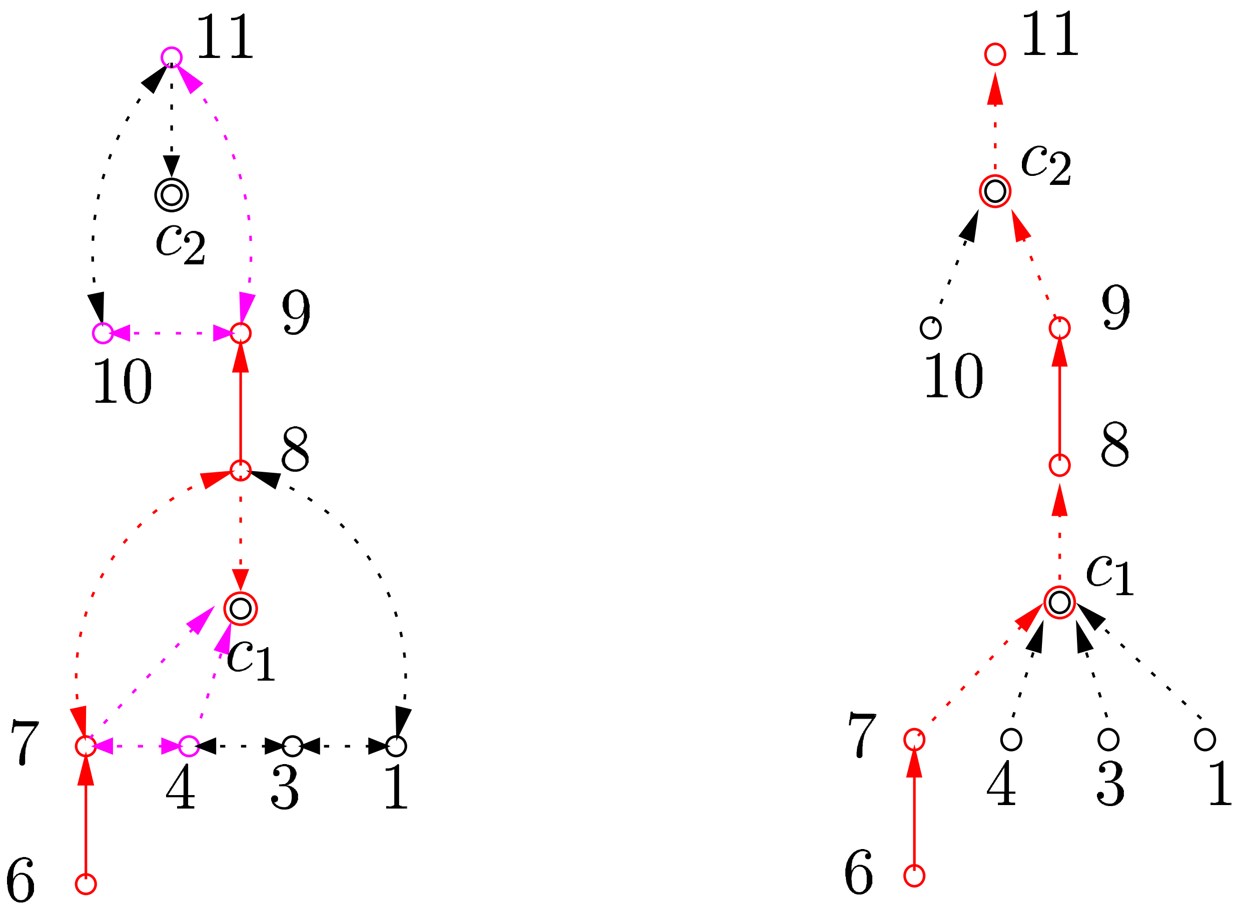


Traversal of the PC-tree

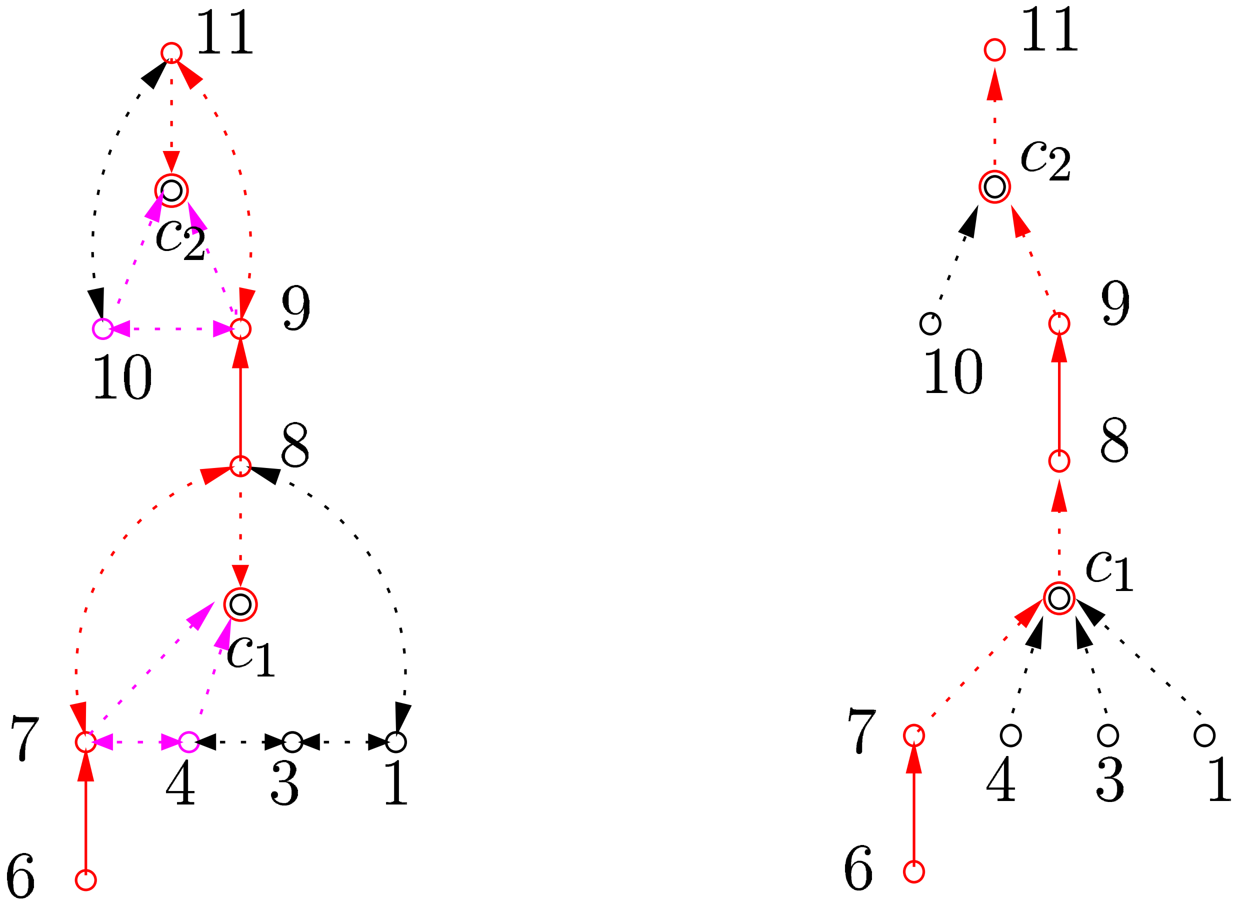




Traversal of the PC-tree

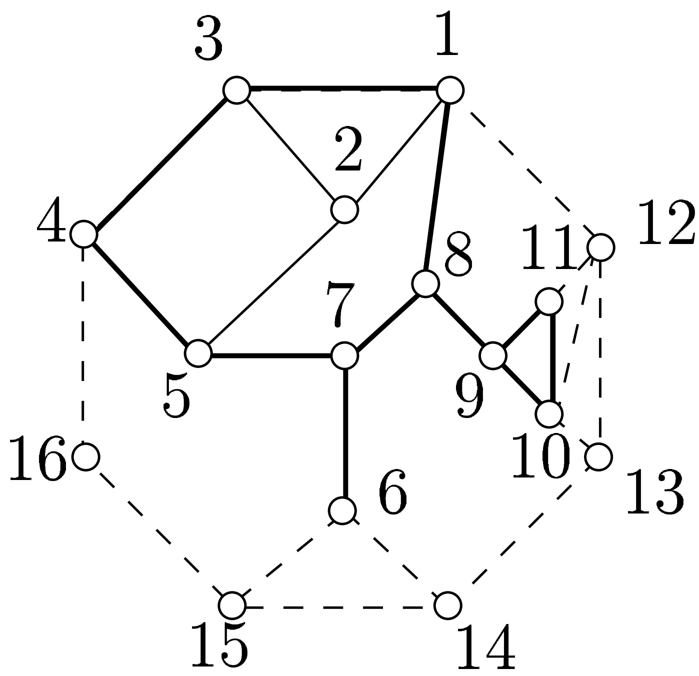


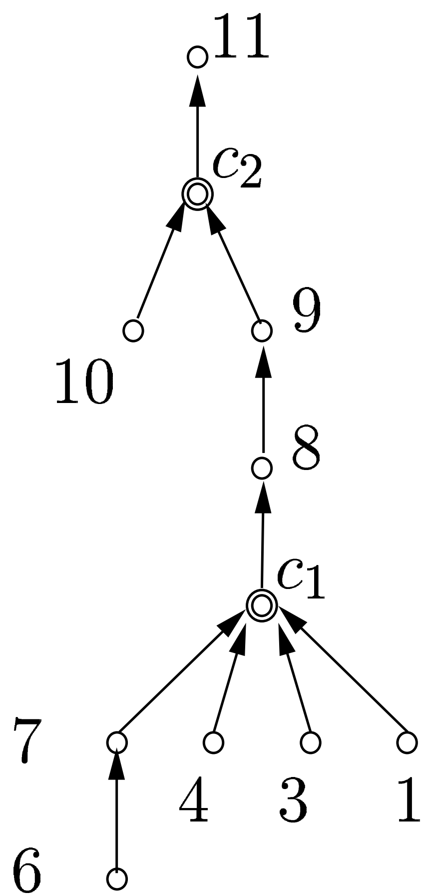
Traversal of the PC-tree



Terminals

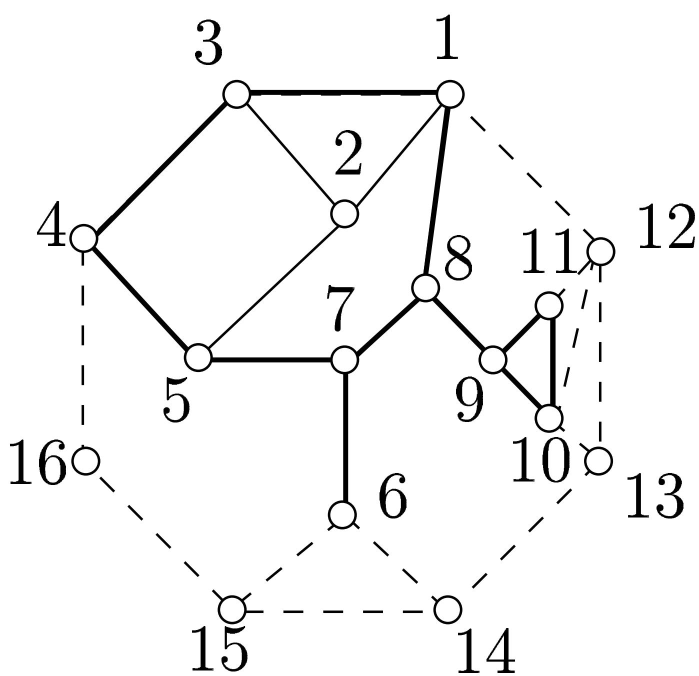
$b(v)$ - largest i such that a descendant of v in T is a neighbor of i in G.

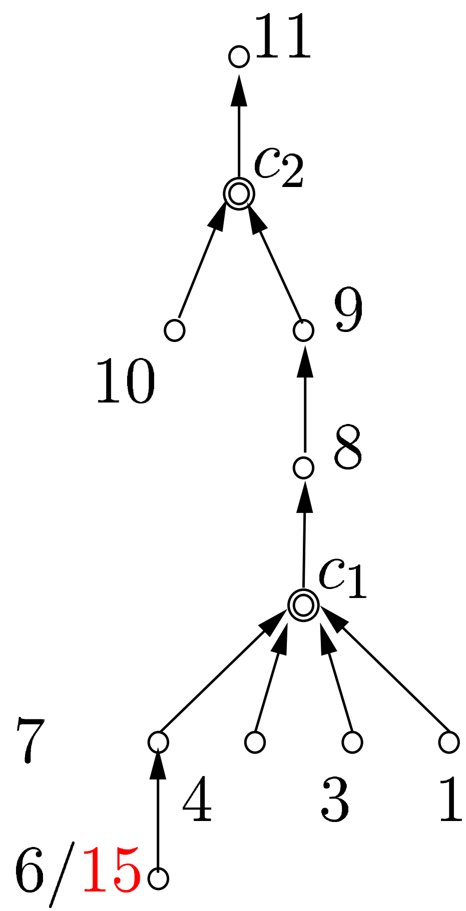




Terminals

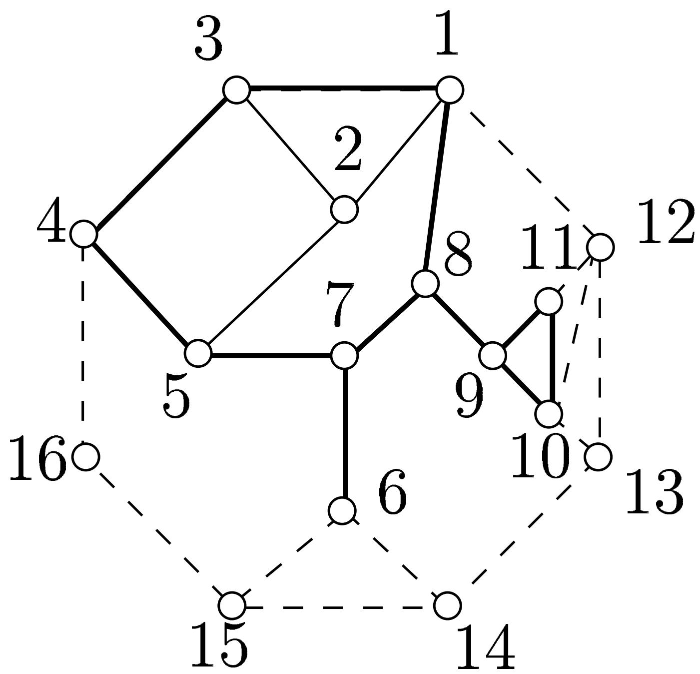
$b(v)$ - largest i such that a descendant of v in T is a neighbor of i in G.

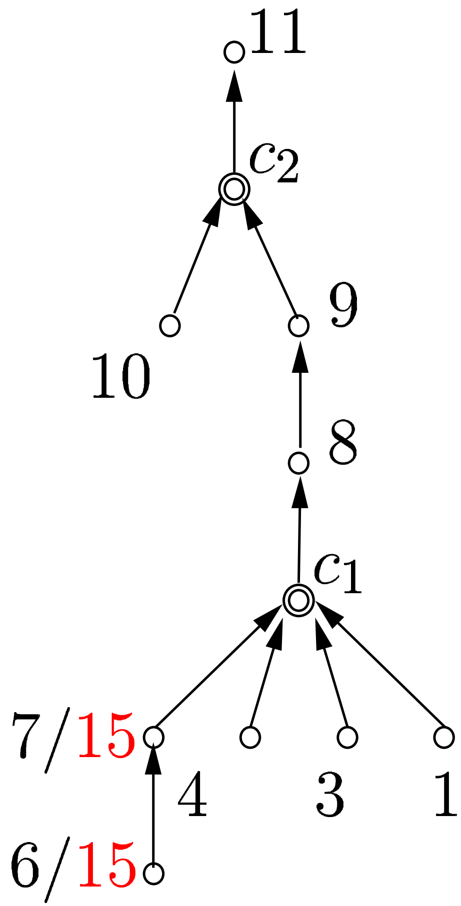




Terminals

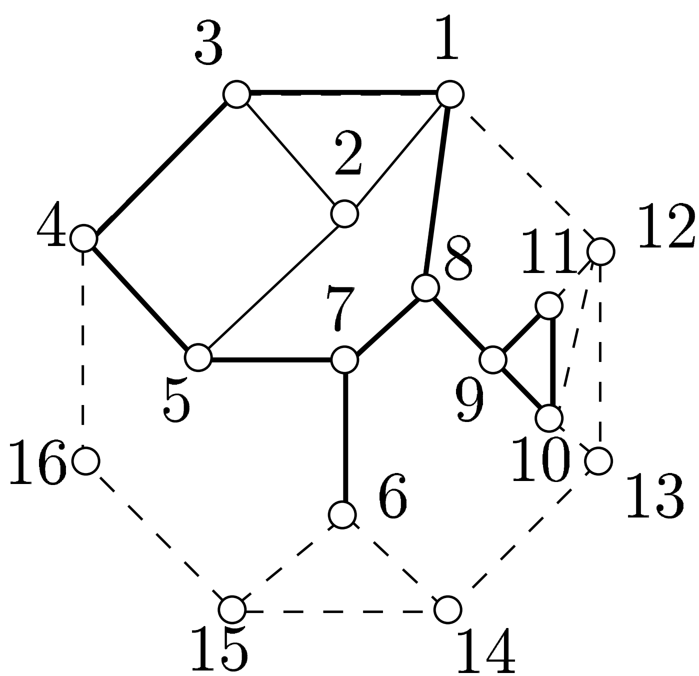
$b(v)$ - largest i such that a descendant of v in T is a neighbor of i in G.

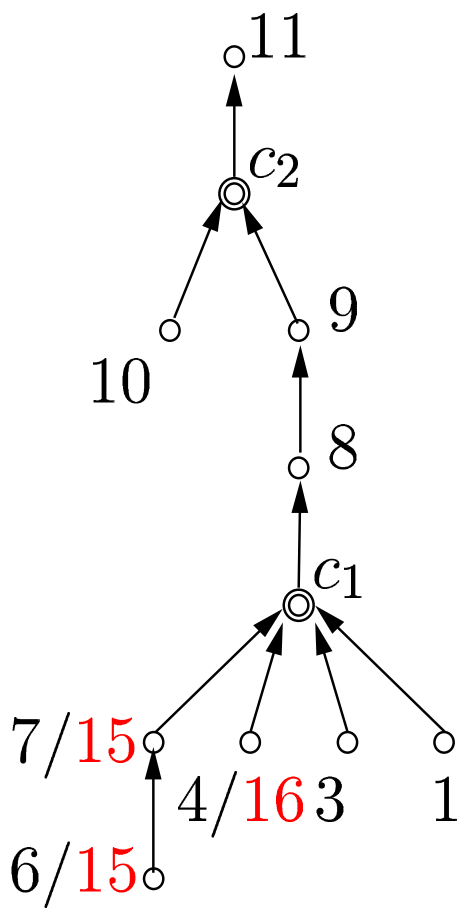




Terminals

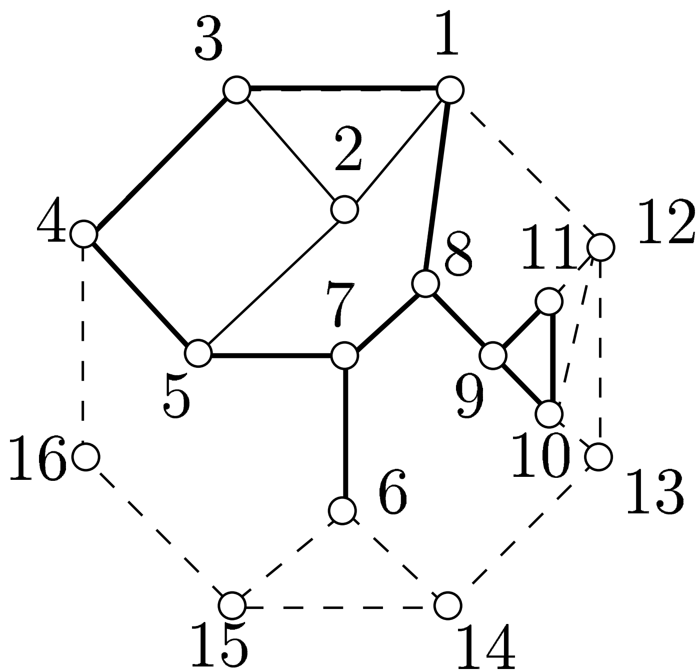
$b(v)$ - largest i such that a descendant of v in T is a neighbor of i in G.

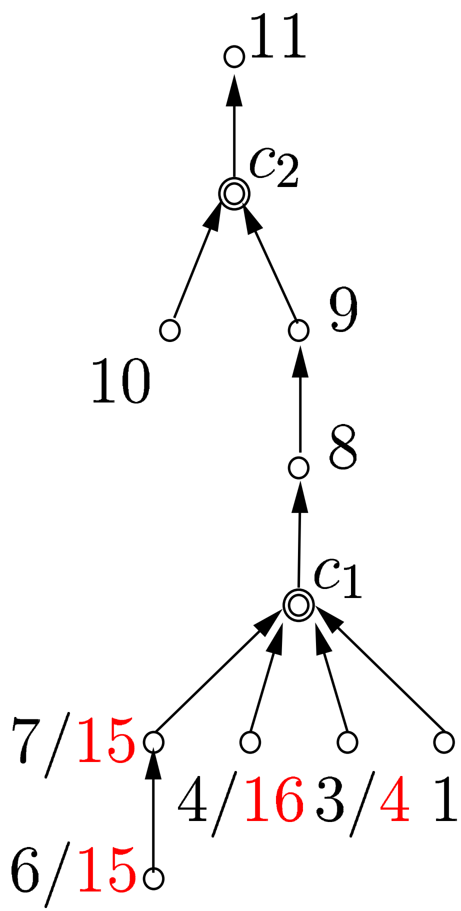




Terminals

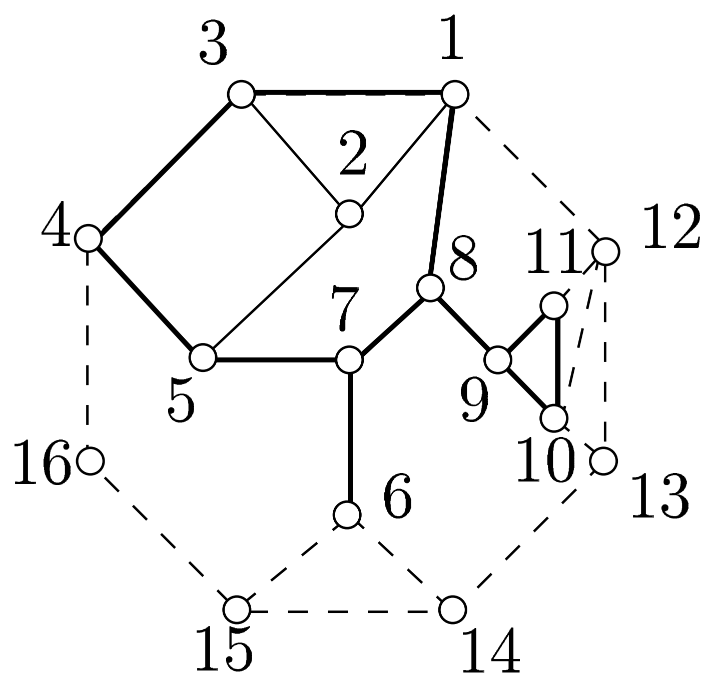
$b(v)$ - largest i such that a descendant of v in T is a neighbor of i in G.

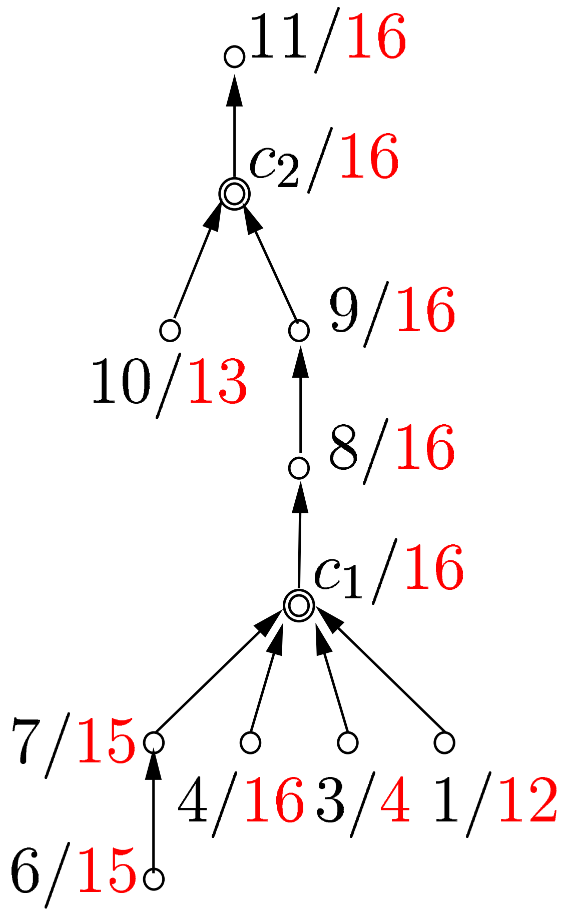




Terminals

$b(v)$ - largest i such that a descendant of v in T is a neighbor of i in G.





Terminals

x - current vertex

T - current PC-tree

Terminals

x - current vertex

T - current PC-tree

Node t in T is a *terminal* if

1. $b(t) > x$

Terminals

x - current vertex

T - current PC-tree

Node t in T is a *terminal* if

1. $b(t) > x$
2. t has a descendant in T that is a neighbor of x in G

Terminals

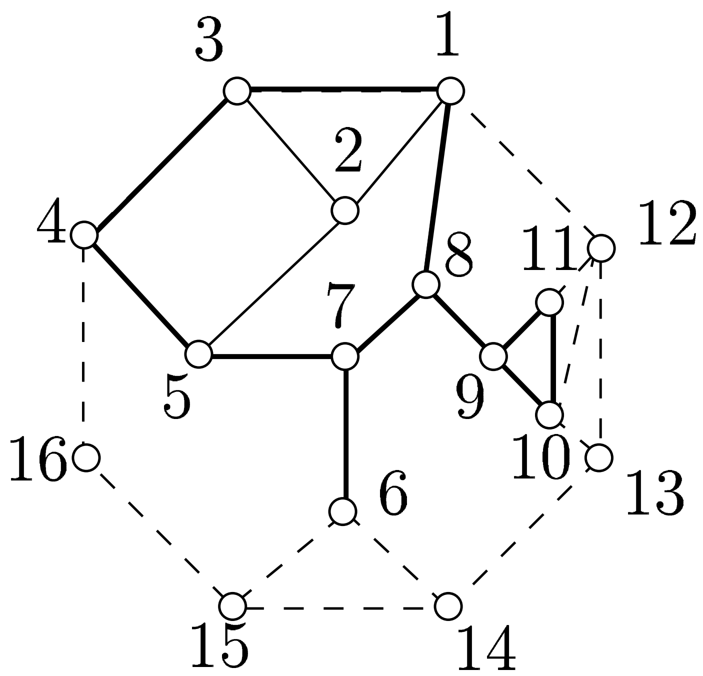
x - current vertex

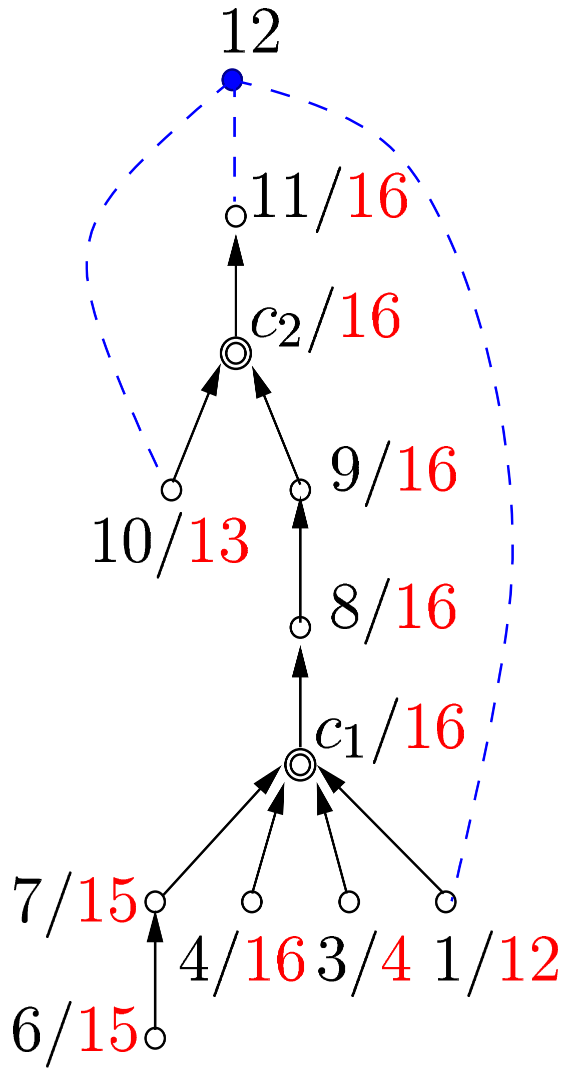
T - current PC-tree

Node t in T is a *terminal* if

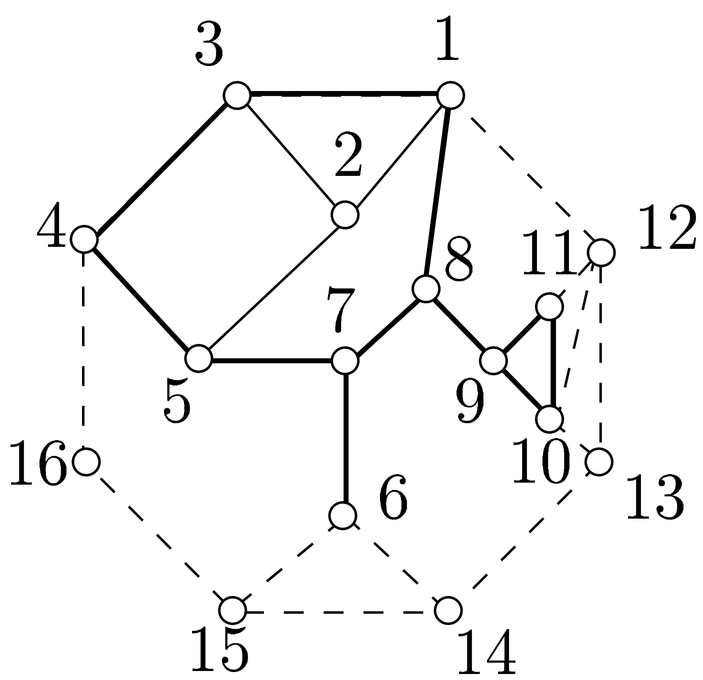
1. $b(t) > x$
2. t has a descendant in T that is a neighbor of x in G
3. no proper descendant of t satisfies (1) and (2)

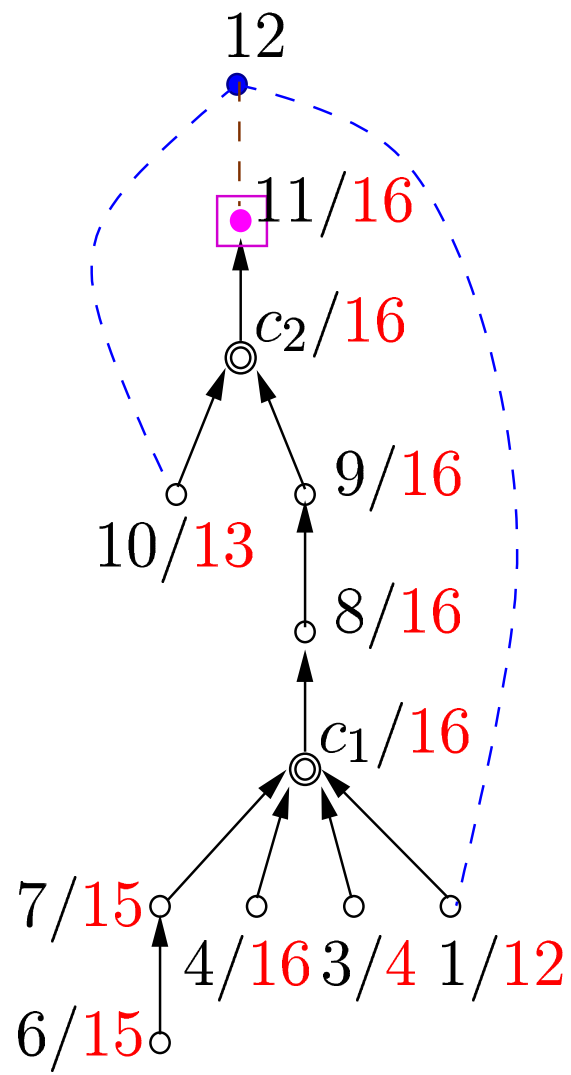
Finding terminals



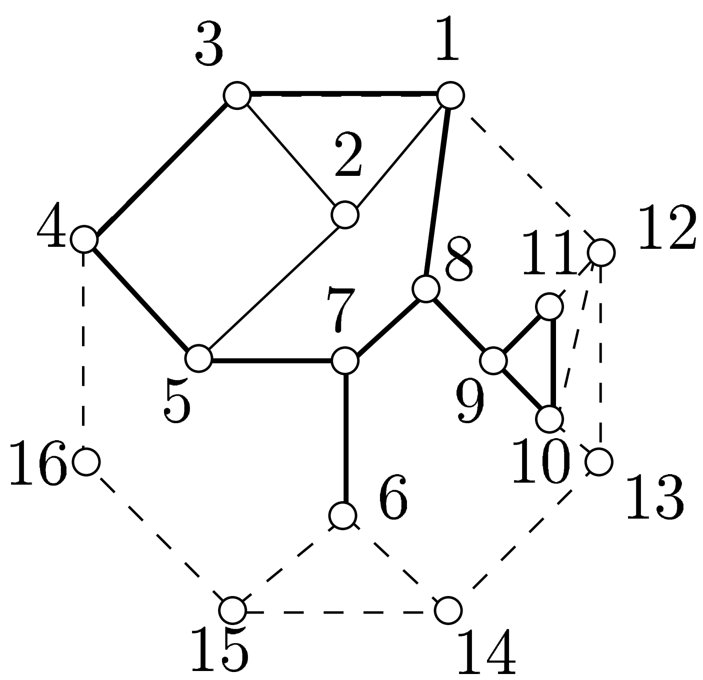


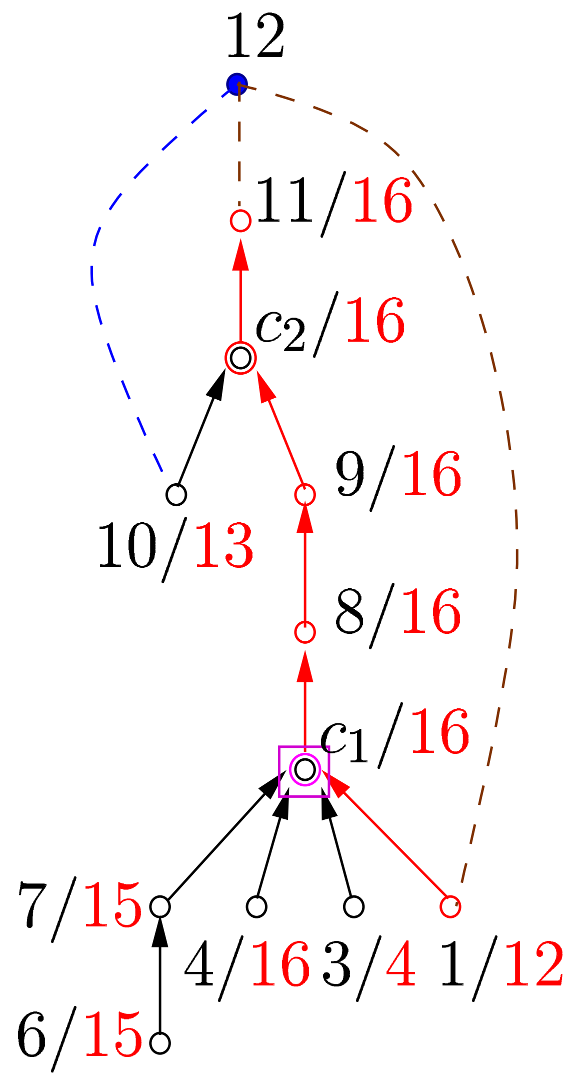
Finding terminals



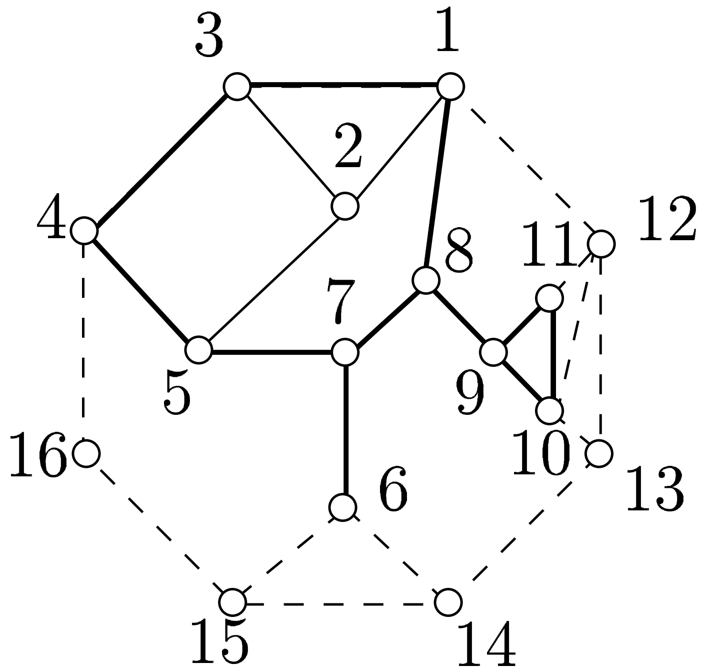


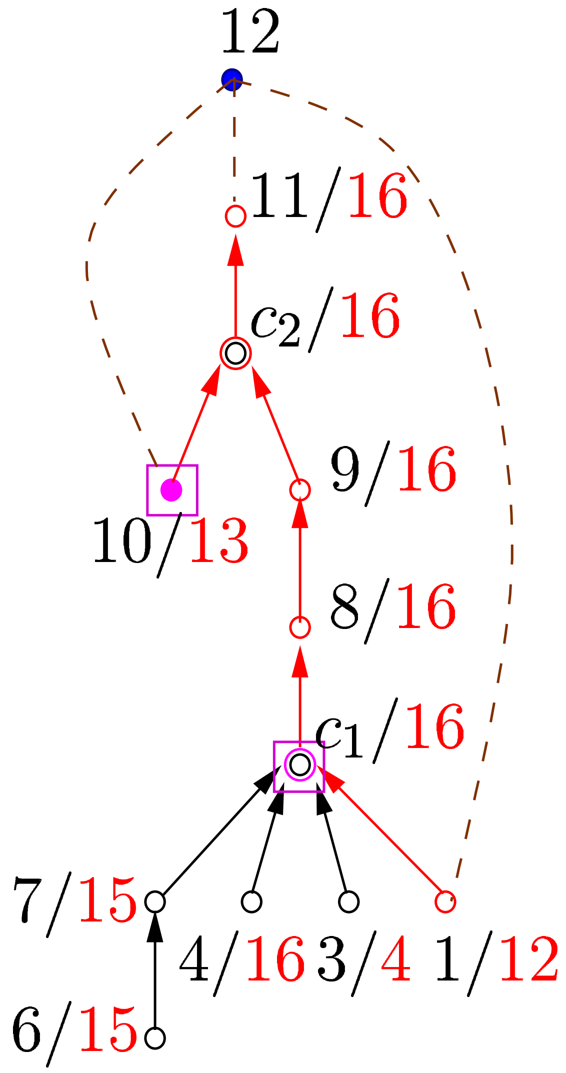
Finding terminals



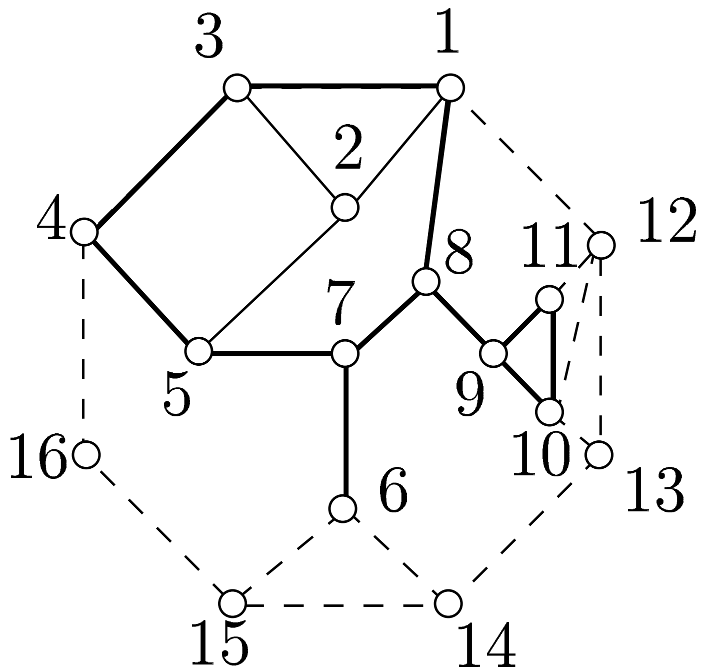


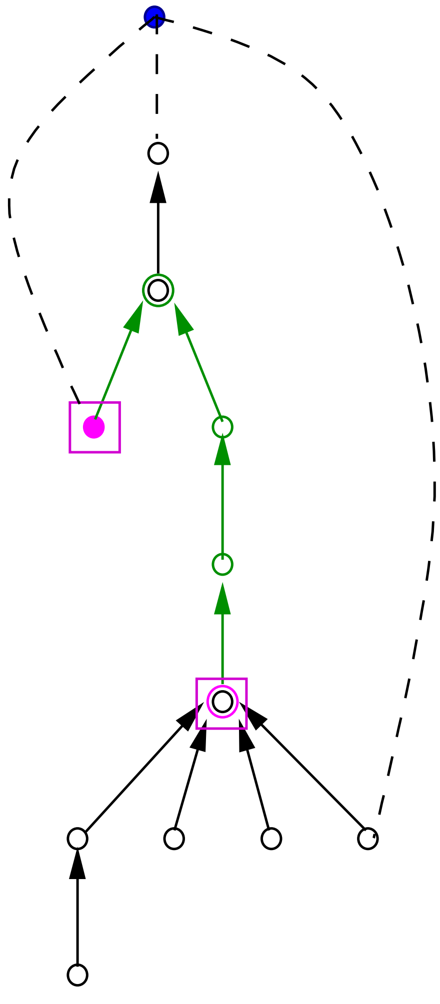
Finding terminals



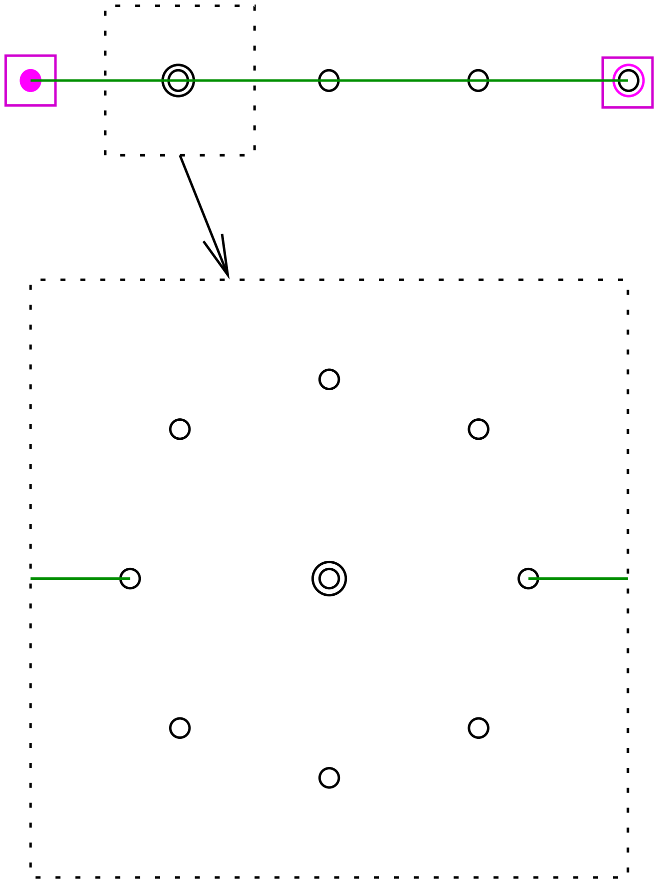


Finding terminals

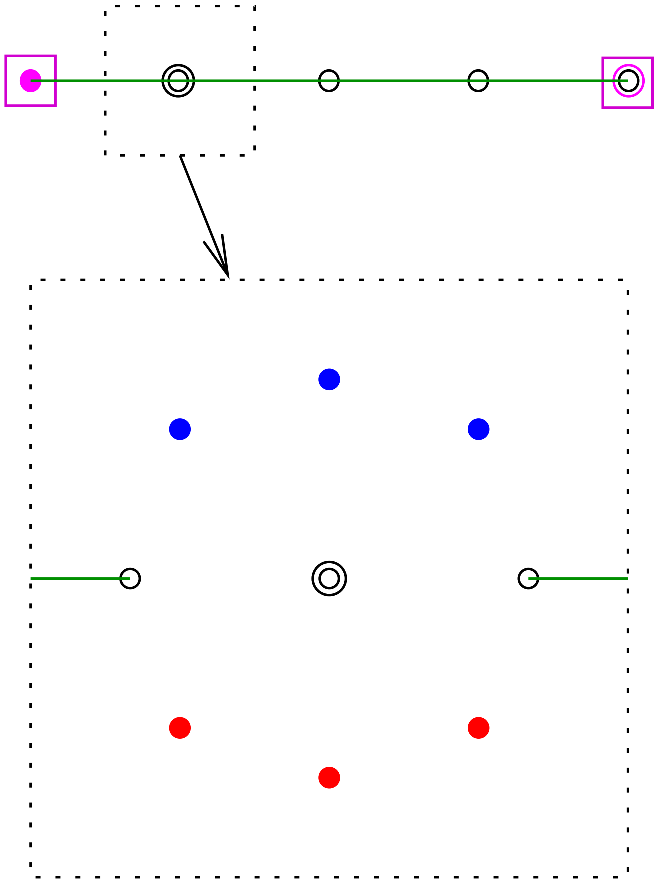




C-node test



C-node test



- $b(v) = x$
- $b(v) > x$ and...

Updating the PC-tree

- creation of new C-node;
- updating some b labels;
- unmarking marked vertices.

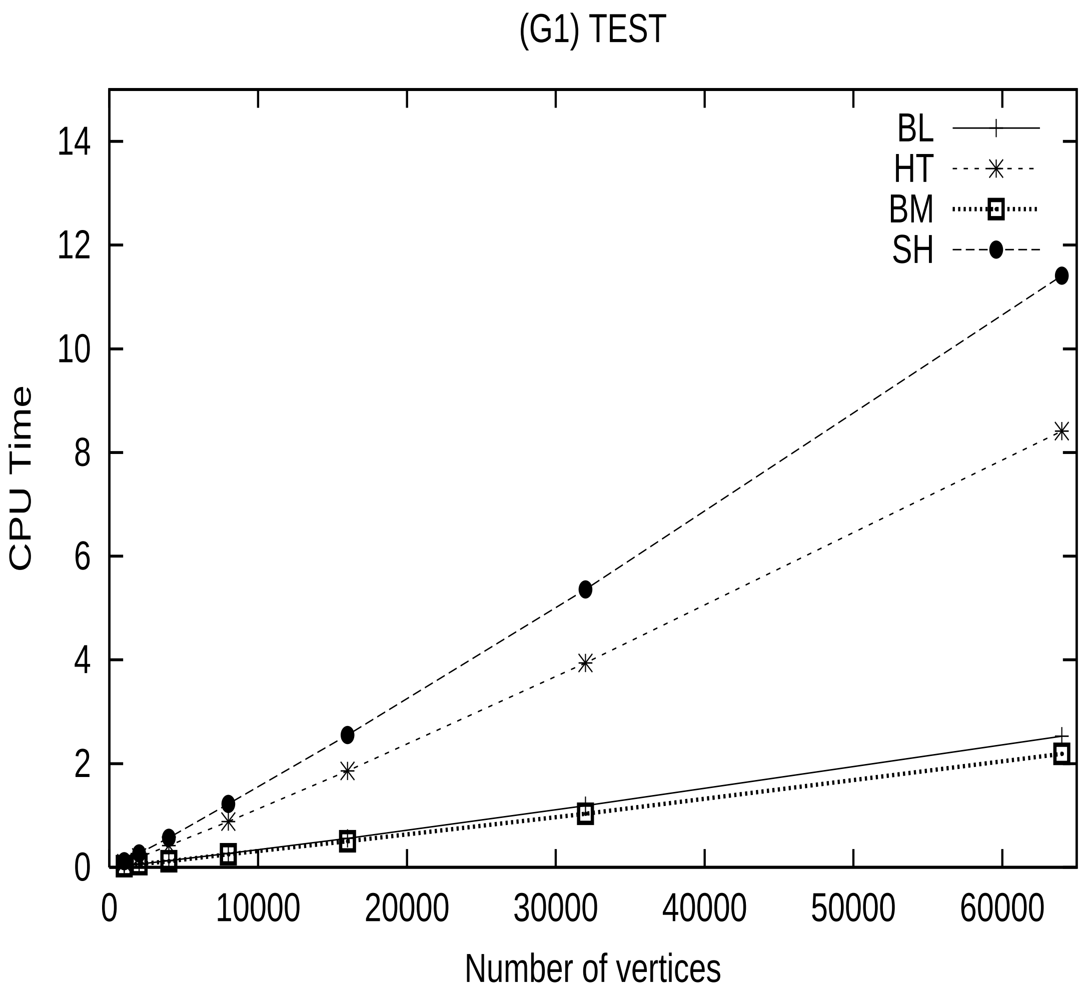
Linear running-time

Each iteration takes time proporcional
to the number of vertices traversed.

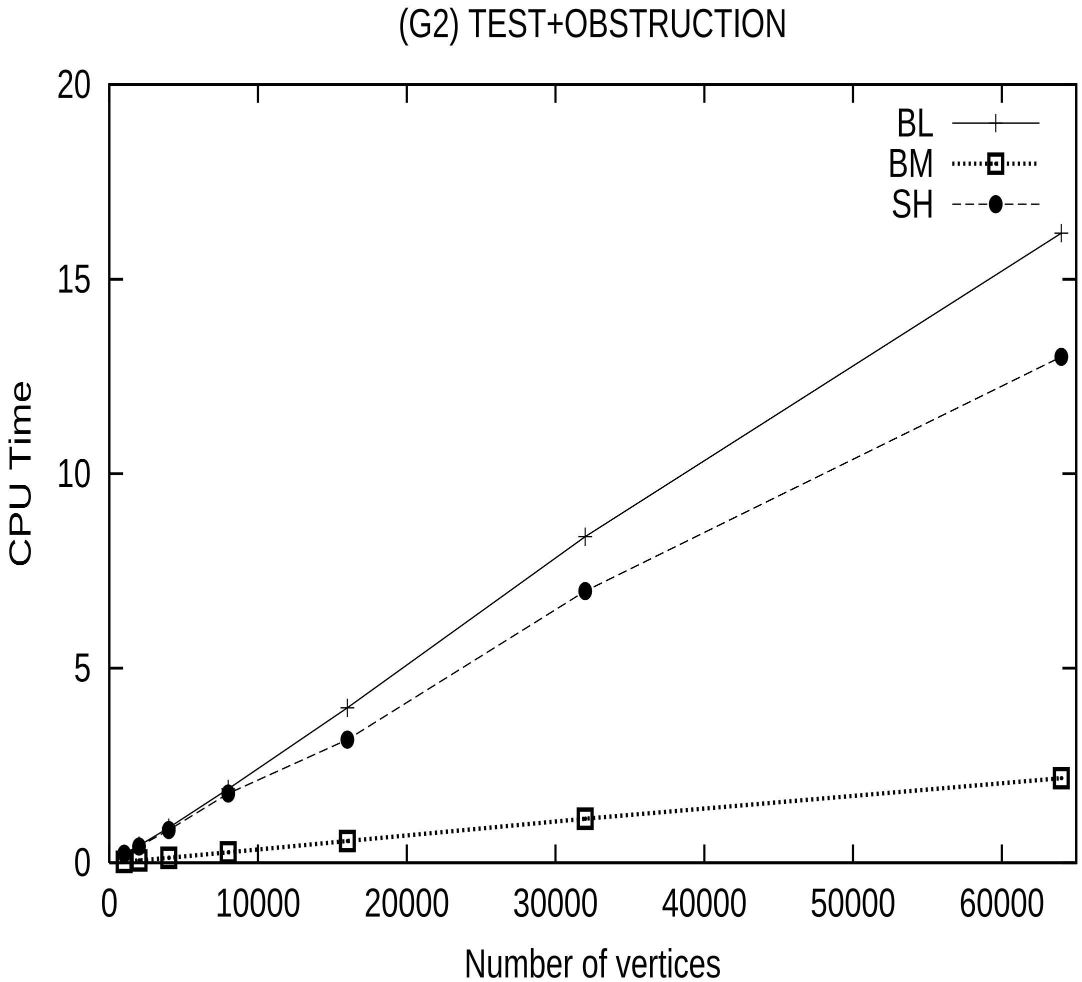
A traversed vertex
either (corresponds to one that) "desappears",
or enters for the first time in a C-node,
or changes to a new C-node.

At most two traversed vertices per C-node
move to another C-node, and
the number of C-nodes is bounded by the number of edges.

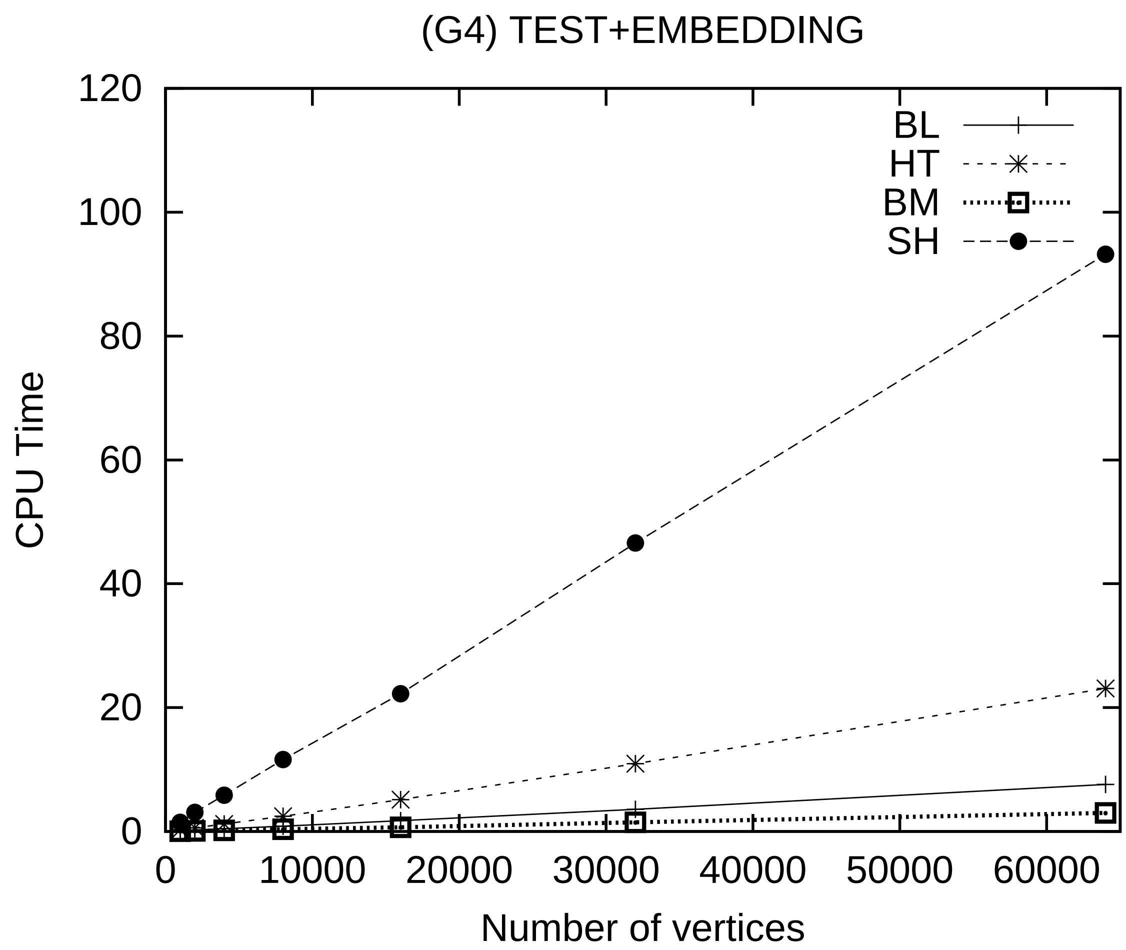
Experimental results



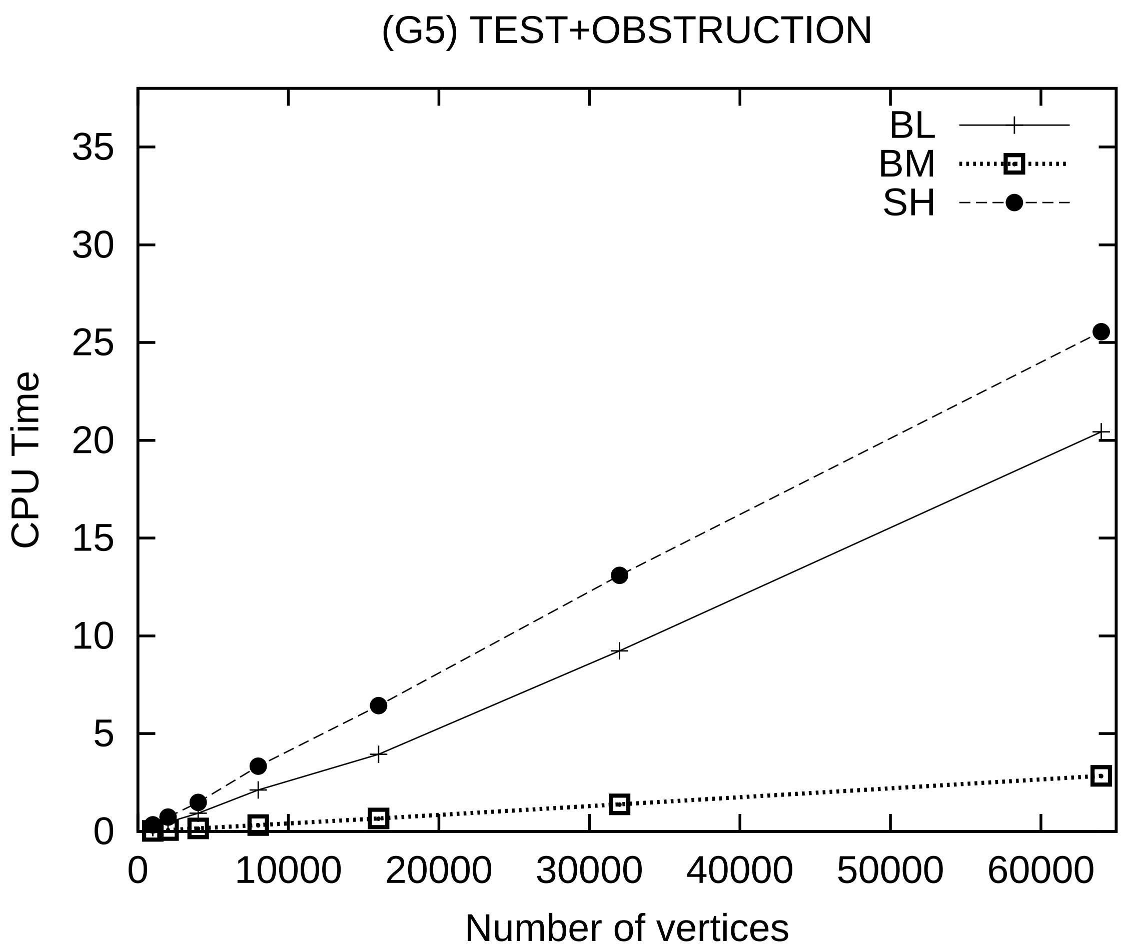
Experimental results



Experimental results



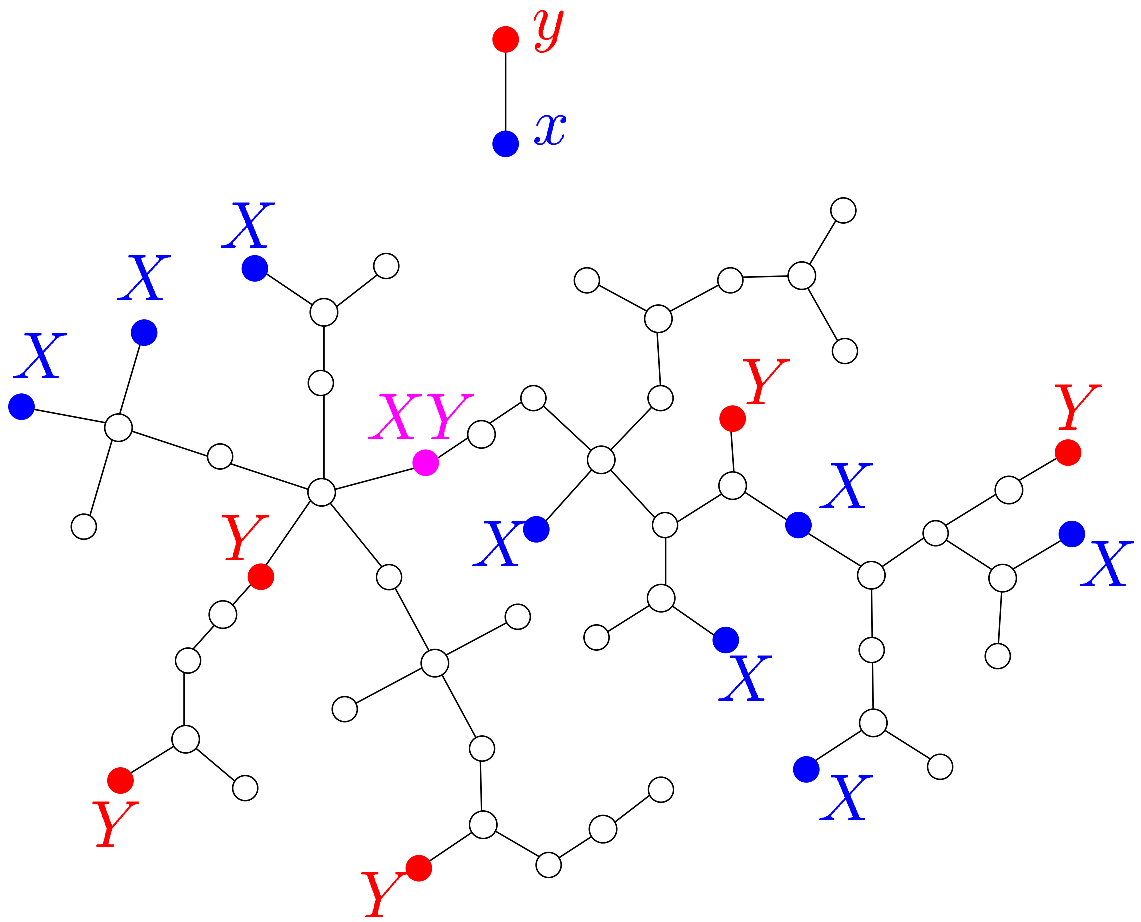
Experimental results



END

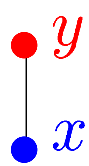
Apply Previous Algorithm

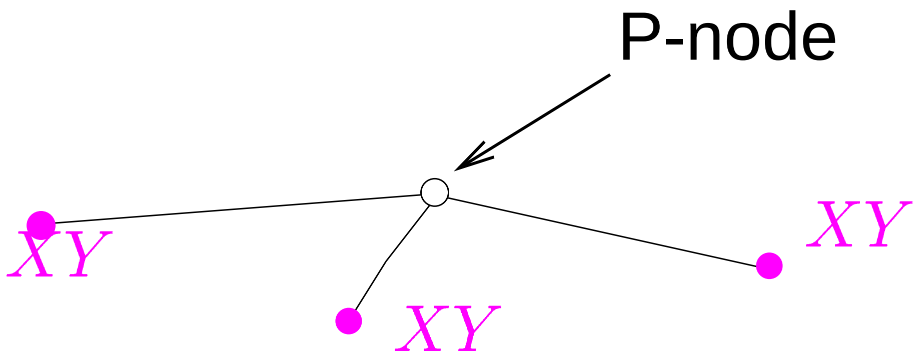
Apply the previous algorithm to the block tree T of F



Apply Previous Algorithm

NO $\Rightarrow$ NO

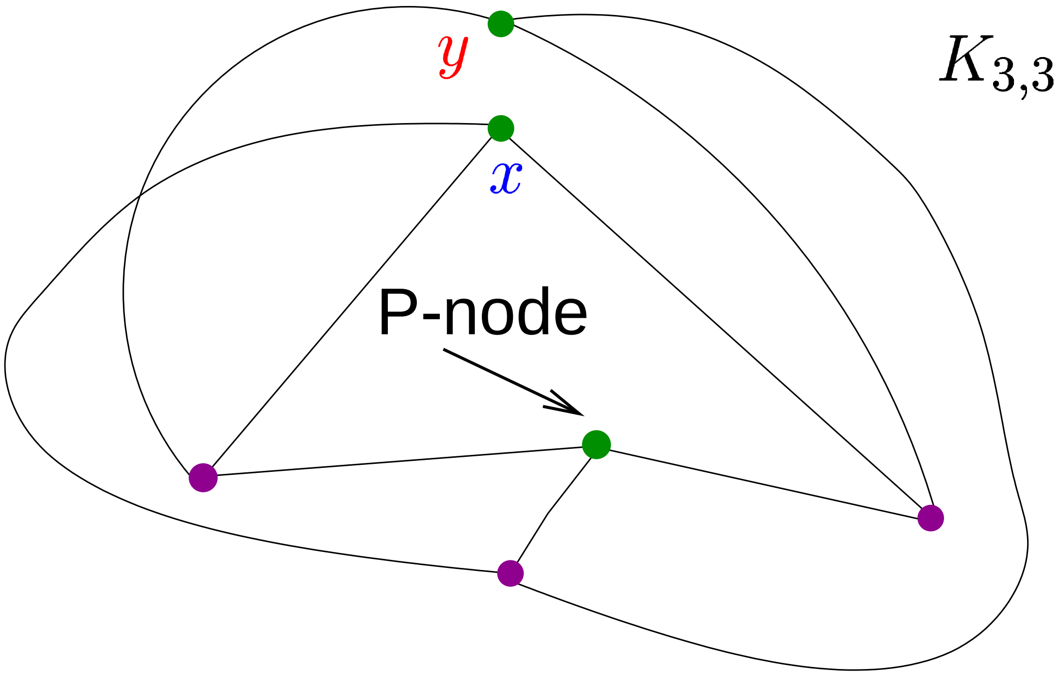




Apply Previous Algorithm

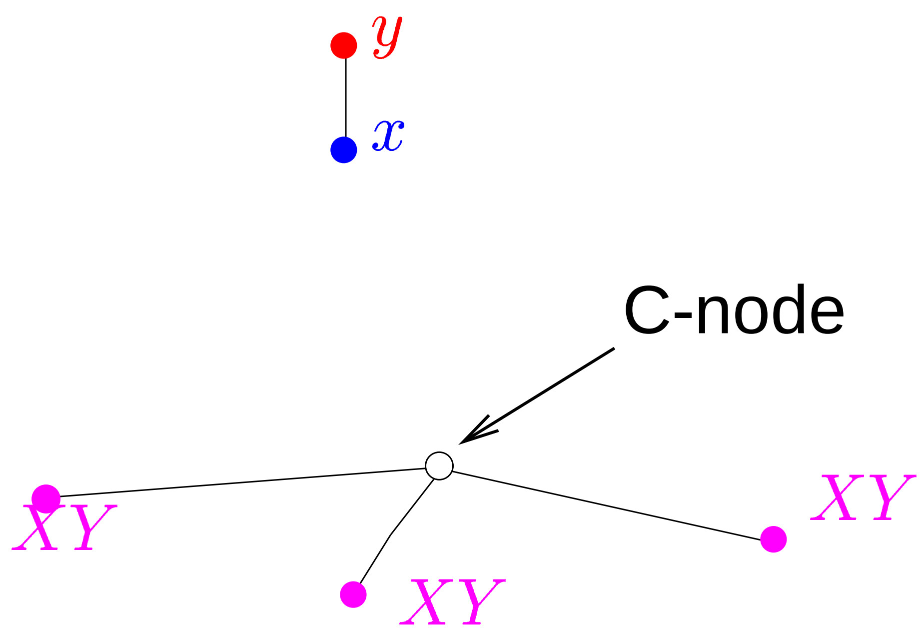
NO $\Rightarrow$ NO

G nonplanar $\Rightarrow$ NO



Apply Previous Algorithm

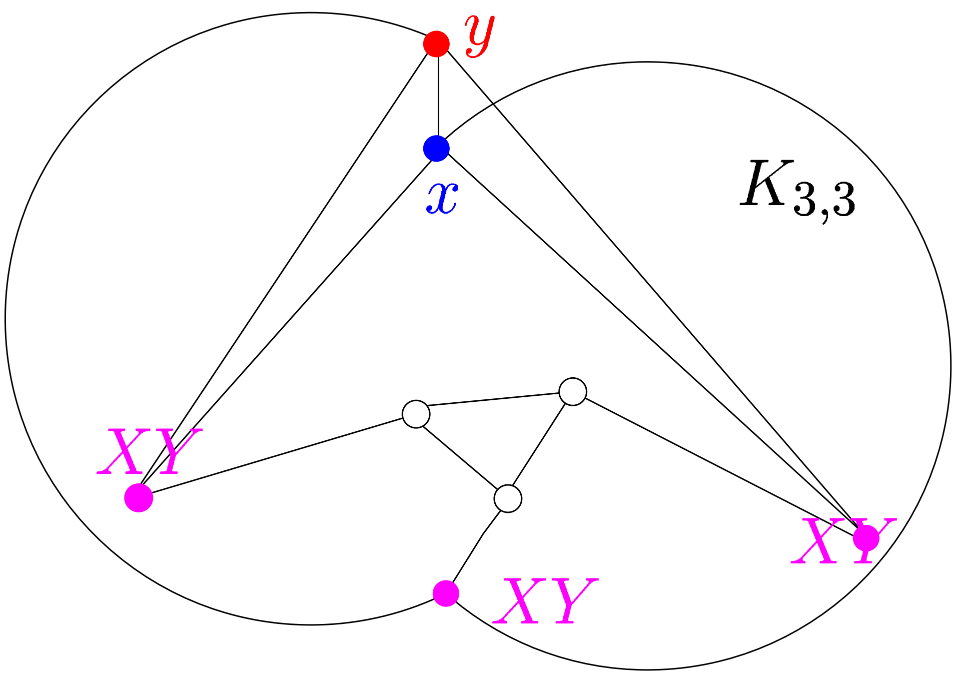
NO $\Rightarrow$ NO



Apply Previous Algorithm

NO $\Rightarrow$ NO

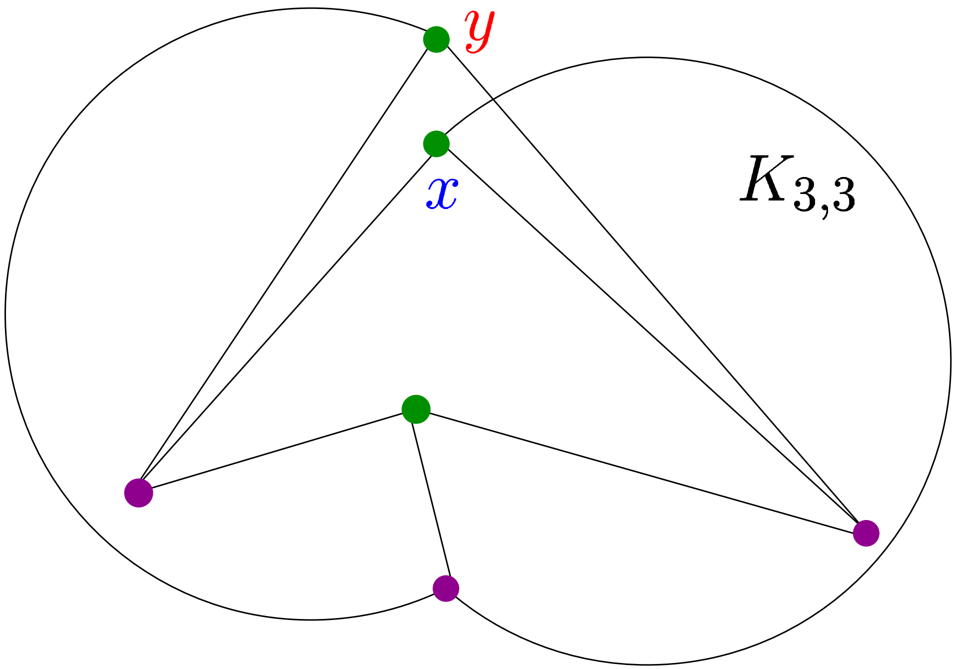
G nonplanar $\Rightarrow$ NO



Apply Previous Algorithm

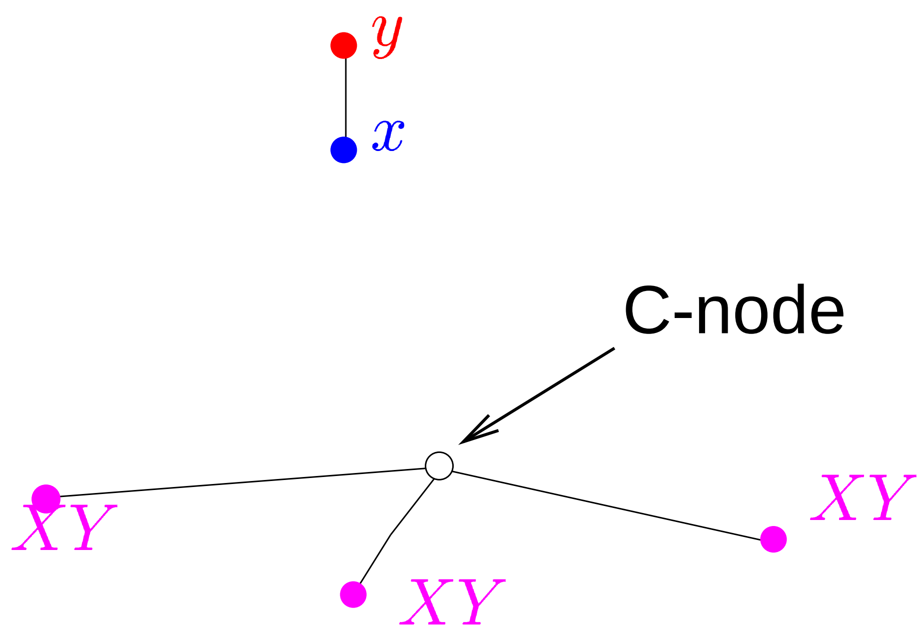
NO $\Rightarrow$ NO

G nonplanar $\Rightarrow$ NO



Apply Previous Algorithm

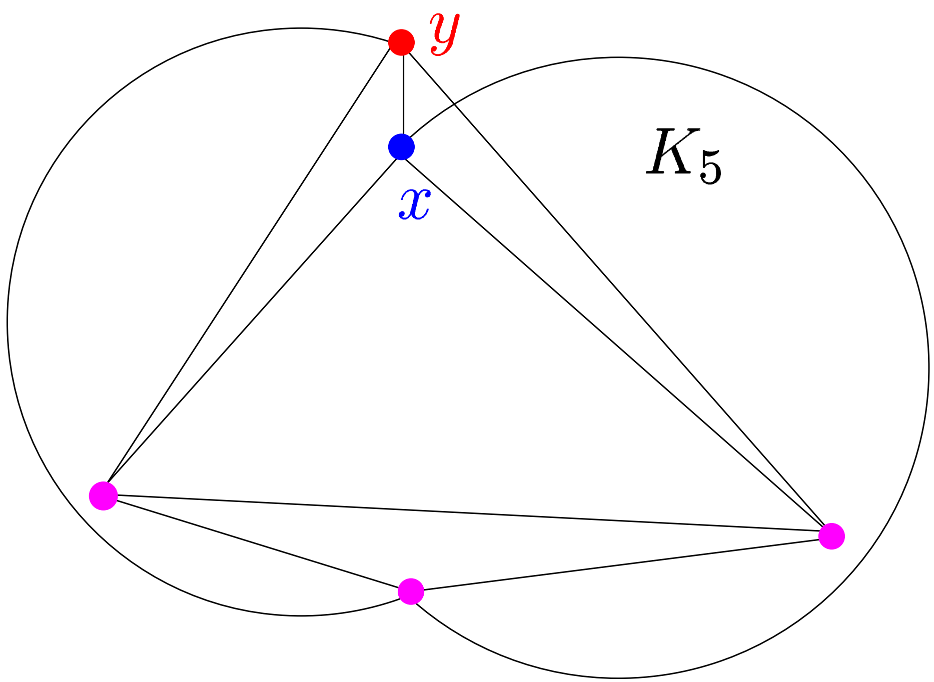
NO $\Rightarrow$ NO



Apply Previous Algorithm

NO $\Rightarrow$ NO

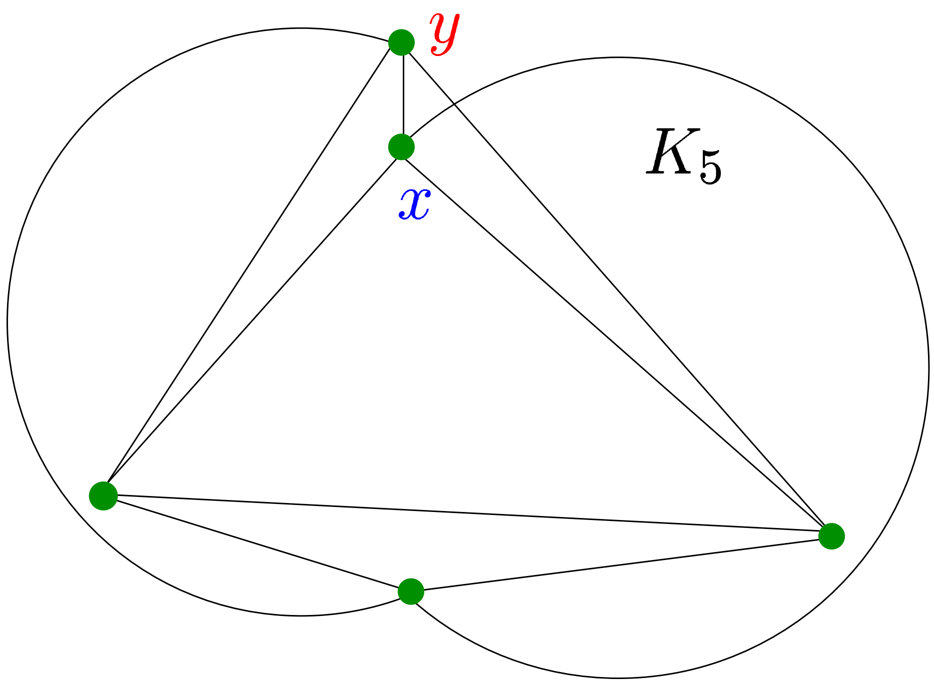
G nonplanar $\Rightarrow$ NO



Apply Previous Algorithm

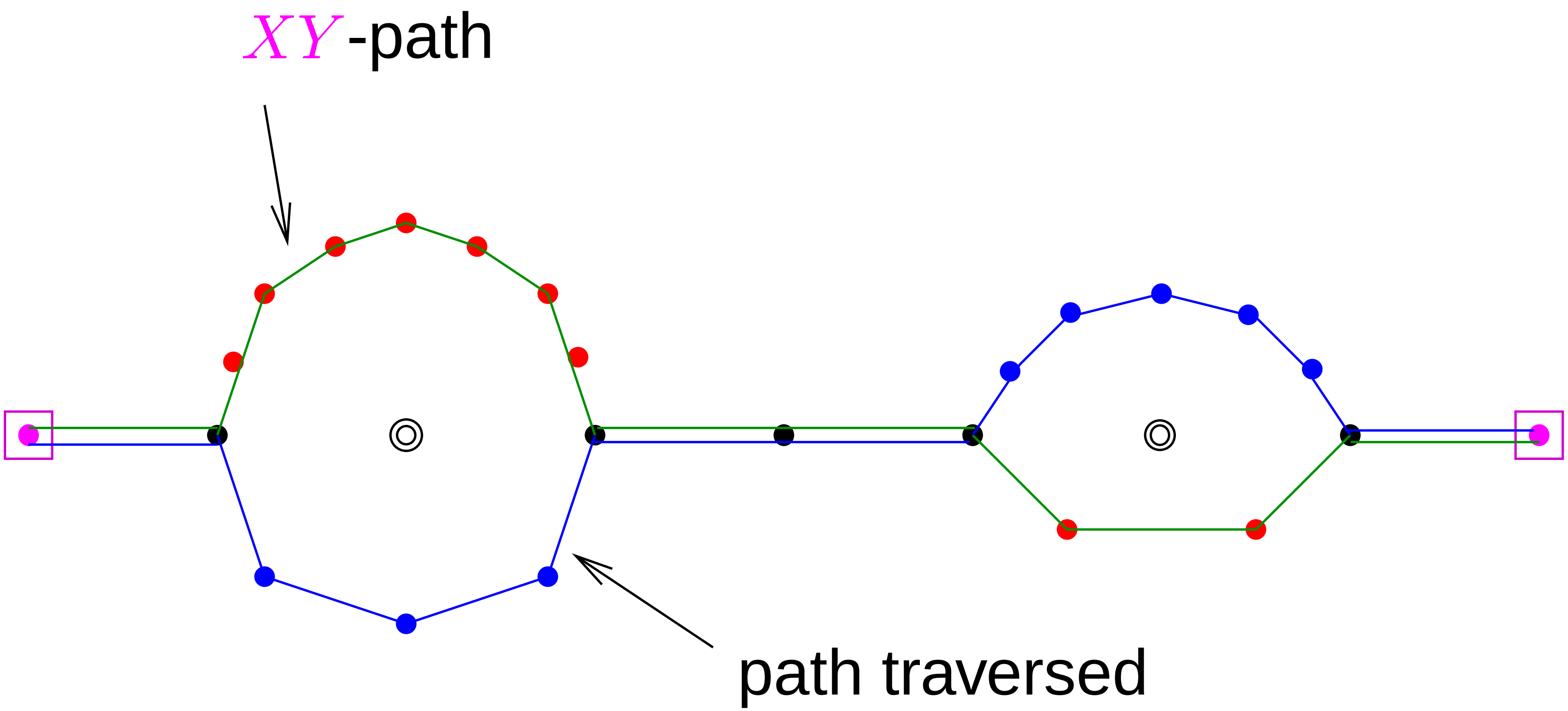
NO $\Rightarrow$ NO

G nonplanar $\Rightarrow$ NO



XY-path

- $b(v) = x$
- $b(v) > x$ and...



$\hookleftarrow$

END