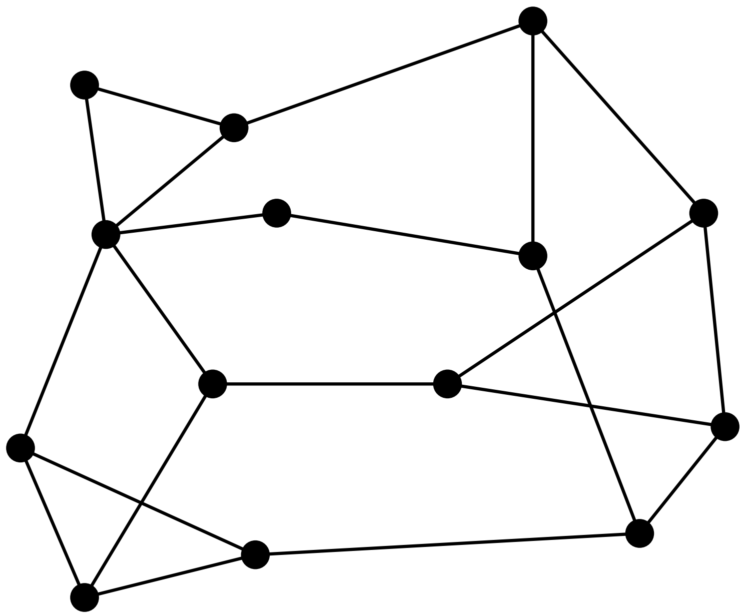
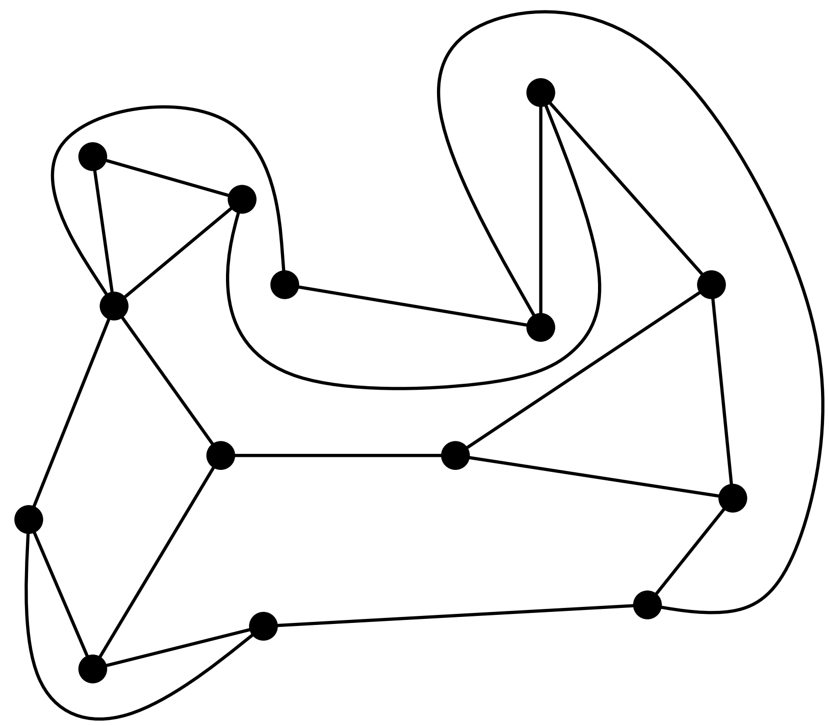
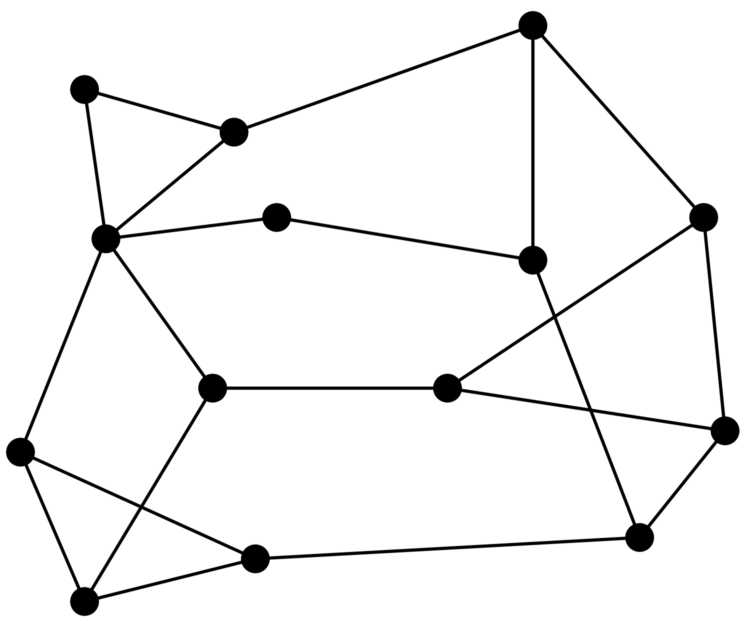
Planar graph



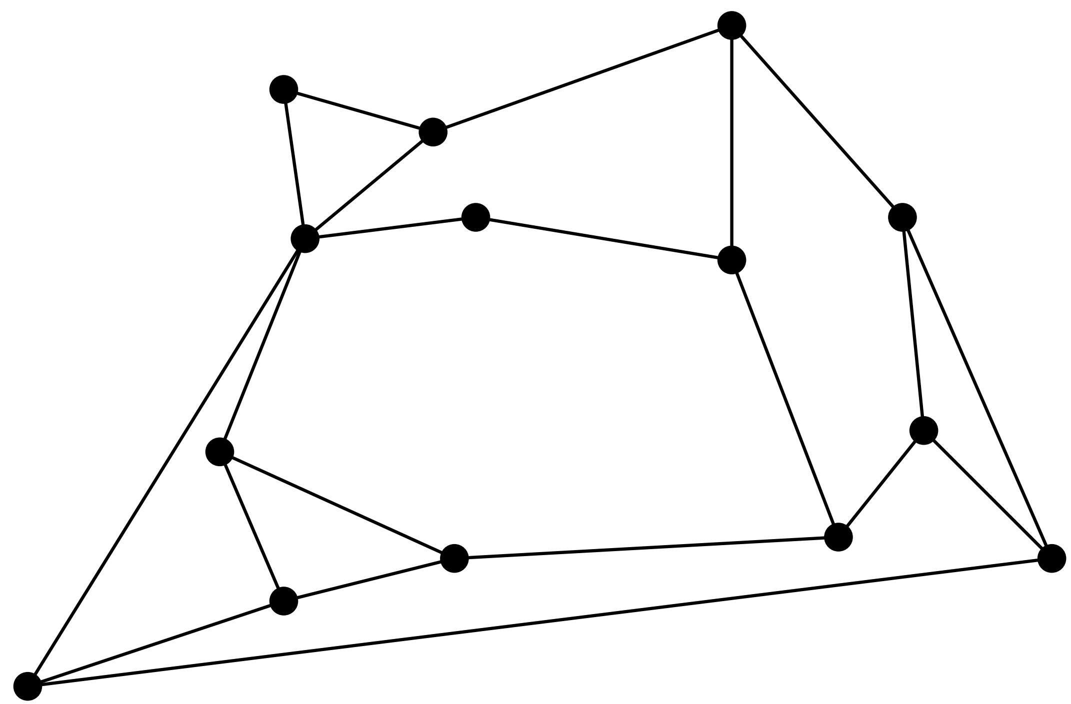
Planar drawing



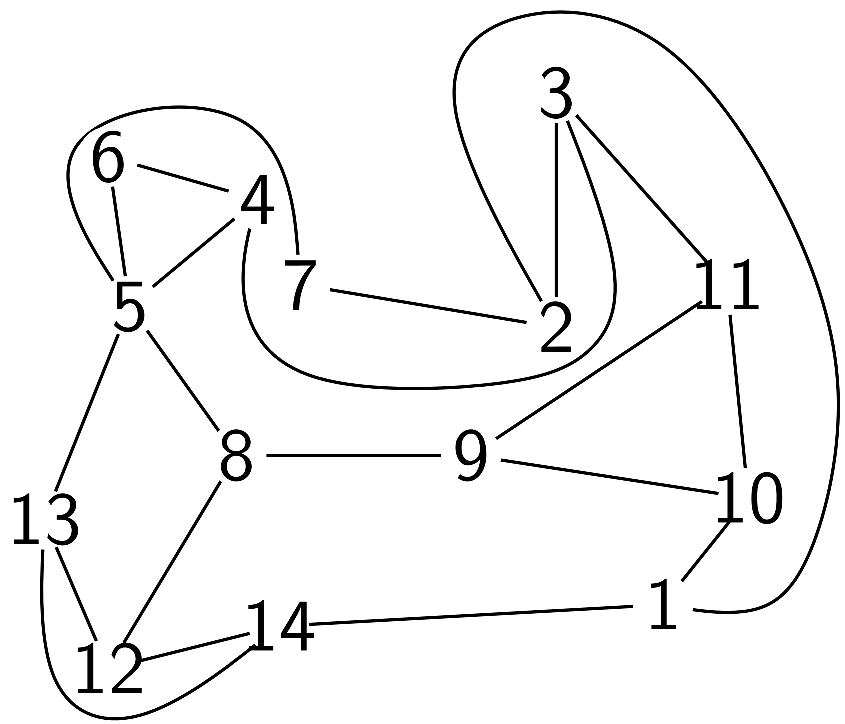
Planar graph



Planar (straight-line) drawing



Planar drawing

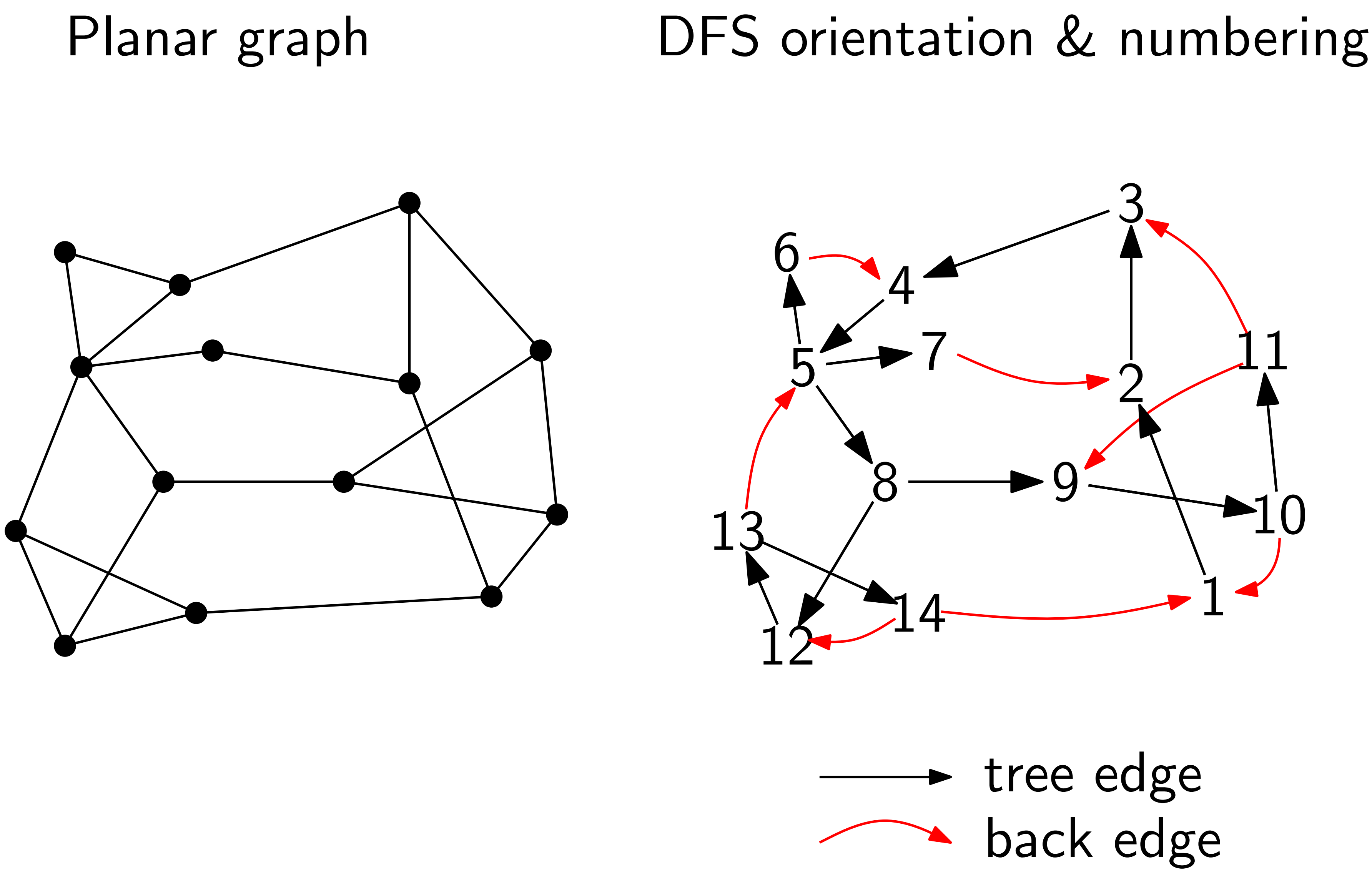


Planar Embedding

1: 2,14,10
2: 1,3,7
3: 2,11,4
4: 3,5,6
5: 4,8,13,7,6
6: 4,5
7: 2,5
8: 5,9,12
9: 8,11,10
10: 1,9,11
11: 3,10,9
12: 8,14,13
13: 5,12,14
14: 1,13,12

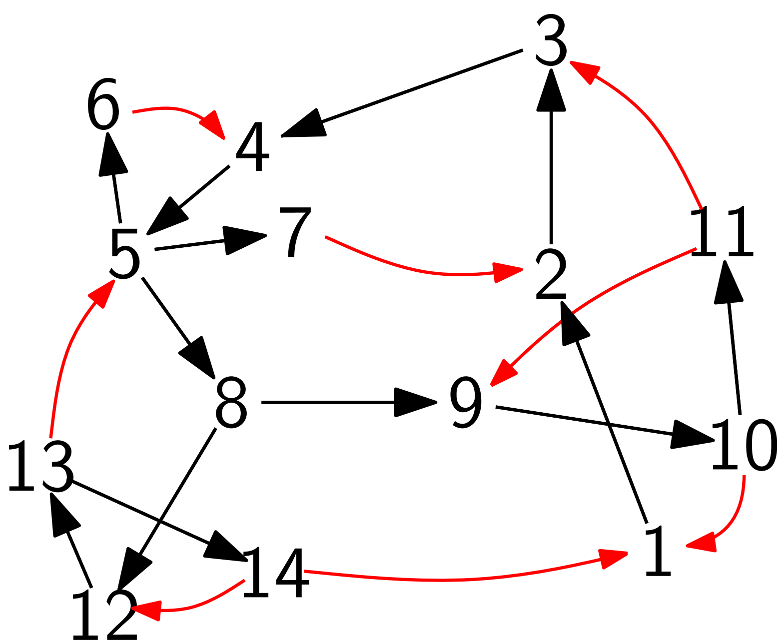
Outer face: 1,14,13,5,7,2

(Clockwise order of neighbors.)

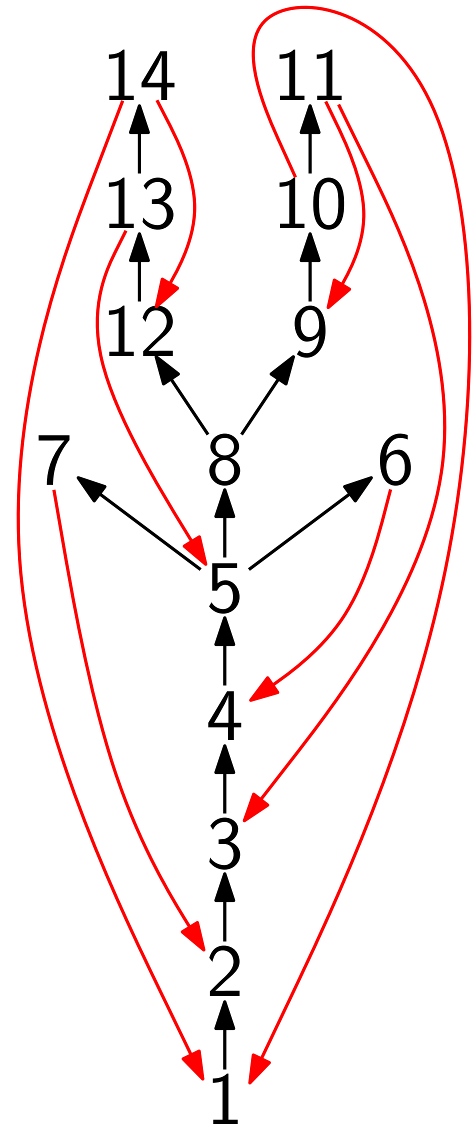


Planar graph
DFS orientation & numbering
3
6
4
7
5
2
11
8
9
13
10
1
14
12
tree edge
back edge

DFS orientation & numbering



LR partition [Brandes 11]

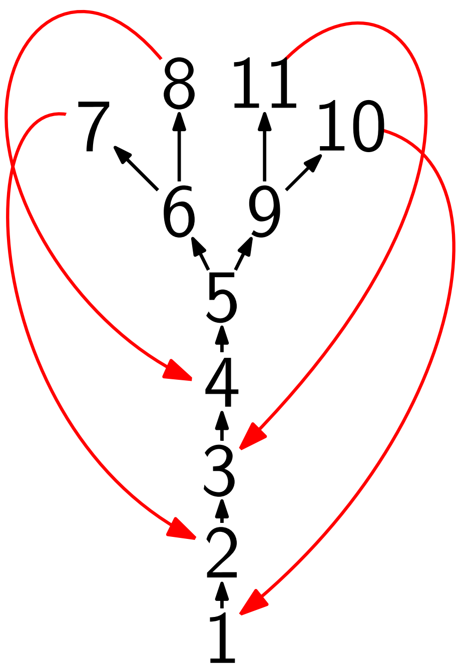


How to order and orient cycles?

Two (related) mechanisms:

Nesting (tree edges)

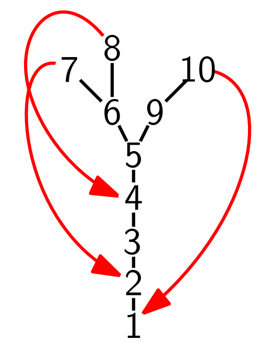
(6,8) cycle must be nested inside
(6,7) cycle (bend same way) Why?



Left / Right Partitioning (back edges)

both (7,2) and (11,3) can't
"bend" the same way Why?

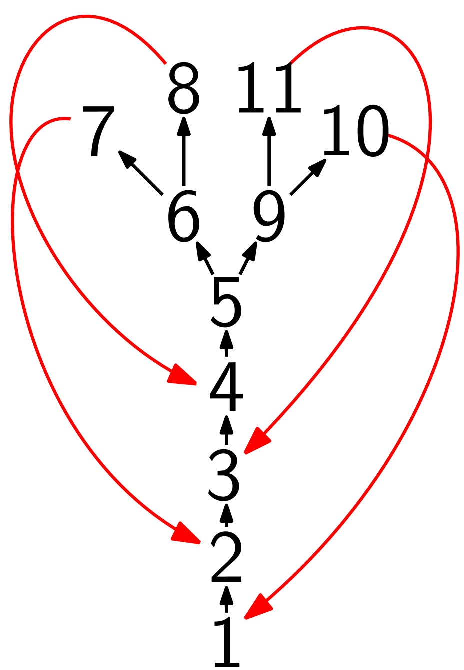
How to order and orient cycles?



Two (related) mechanisms:

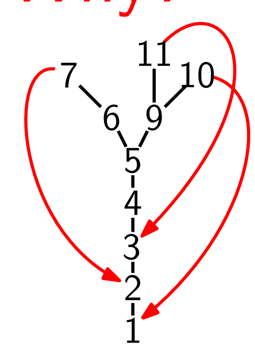
Nesting (tree edges)

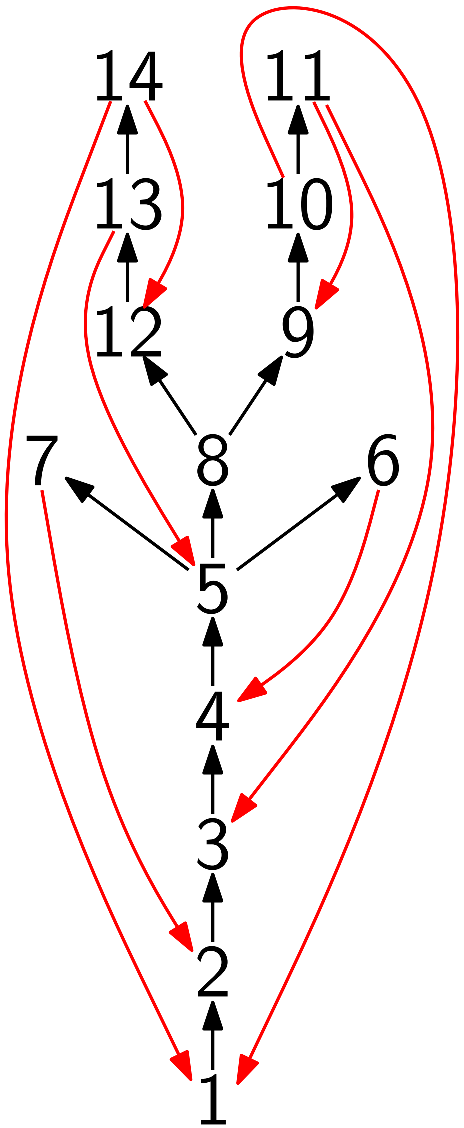
(6,8) cycle must be nested inside
(6,7) cycle (bend same way) Why?



Left / Right Partitioning (back edges)

both (7,2) and (11,3) can't
"bend" the same way Why?



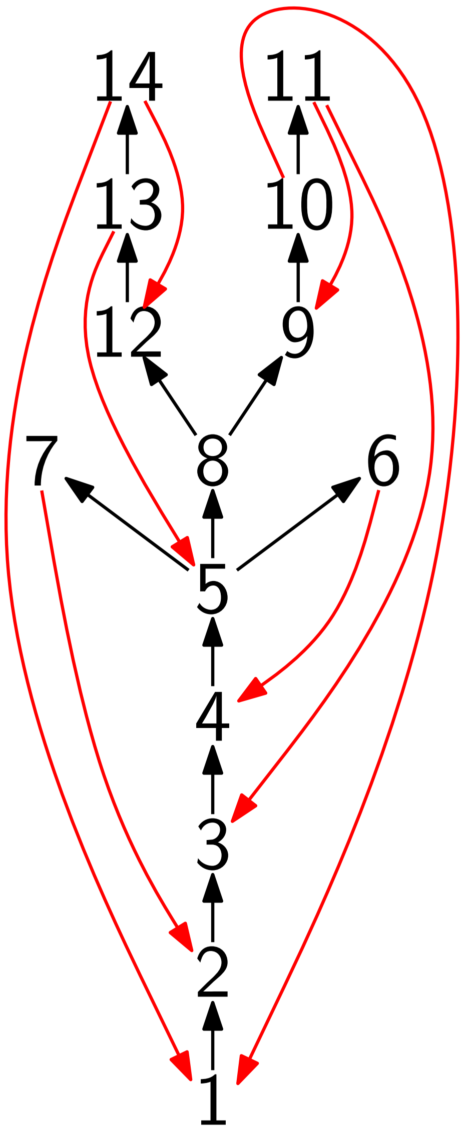


The return point of a back edge (v, w) is w.
The return points of a tree edge (v, w) are
u such that $u \xrightarrow{+} v \rightarrow w \xrightarrow{*} x \hookrightarrow u$.

return point of $(6, 4)$ is 4.

return point of $(5, 7)$ is 2.

return points of $(4, 5)$ are 3,2,1.



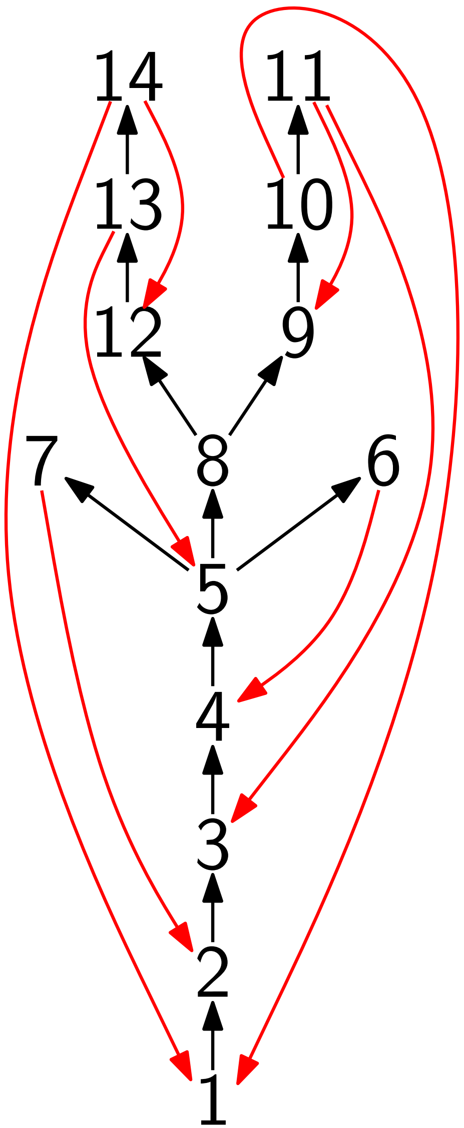
The return point of a back edge (v, w) is w.
The return points of a tree edge (v, w) are u such that $u \xrightarrow{+} v \rightarrow w \xrightarrow{*} x \hookrightarrow u$.

The lowpt of an edge (v, w) is its lowest return point (or w if none exists).

lowpt of $(6, 4)$ is 4.

lowpt of $(5, 7)$ is 2.

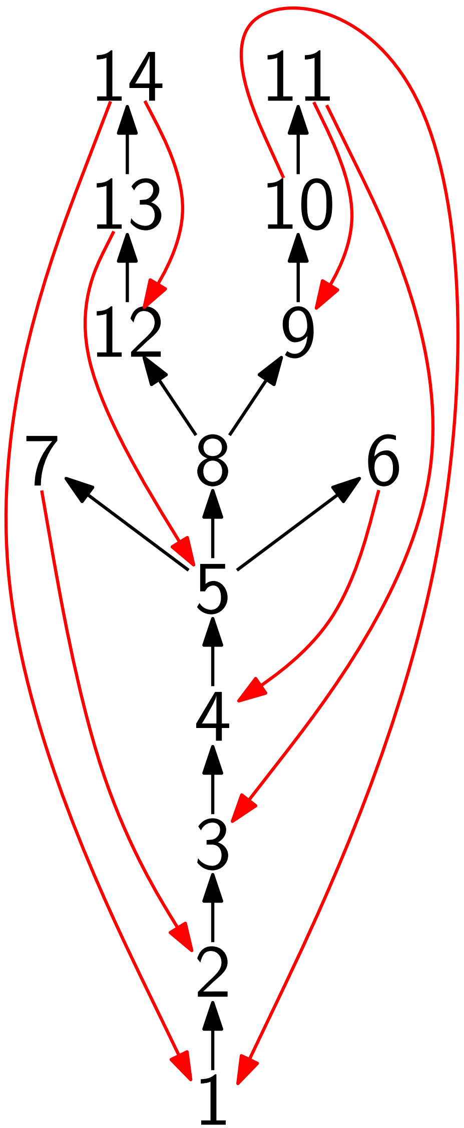
lowpt of $(4, 5)$ is 1.



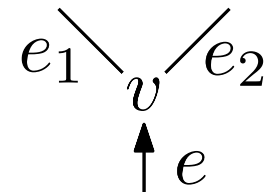
The return point of a back edge (v, w) is w.
The return points of a tree edge (v, w) are u such that $u \xrightarrow{+} v \rightarrow w \xrightarrow{*} x \hookrightarrow u$.

The lowpt of an edge (v, w) is its lowest return point (or w if none exists).

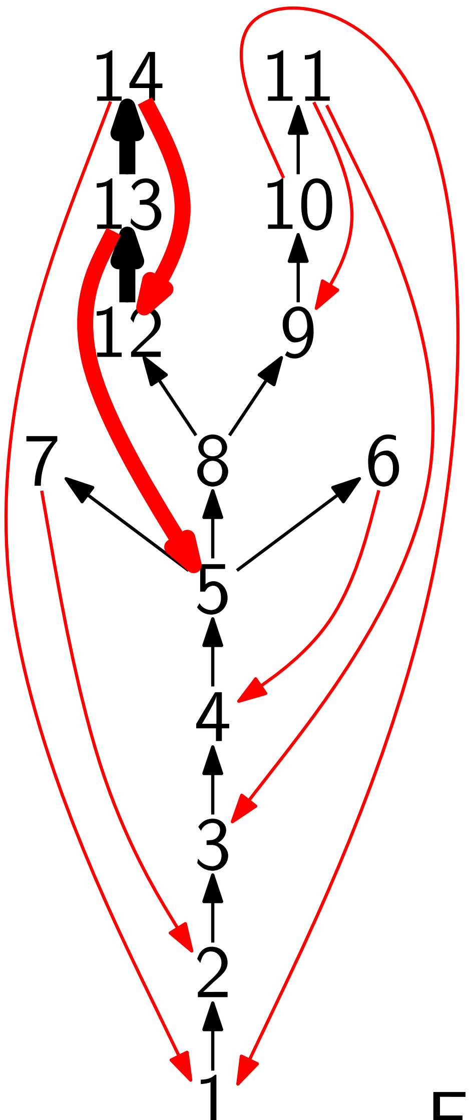
Back edge (x, u) is the return edge for itself and every tree edge (v, w) with $u \xrightarrow{+} v \rightarrow w \xrightarrow{*} x \hookrightarrow u$.



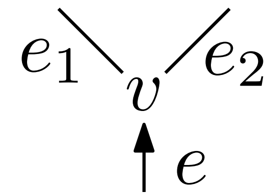
An LR partition is a partition of the back edges into Left and Right so that for every fork



- all return edges of e_1 ending strictly higher than lowpt(e_2) belong to one partition, and
- all return edges of e_2 ending strictly higher than lowpt(e_1) belong to the other.

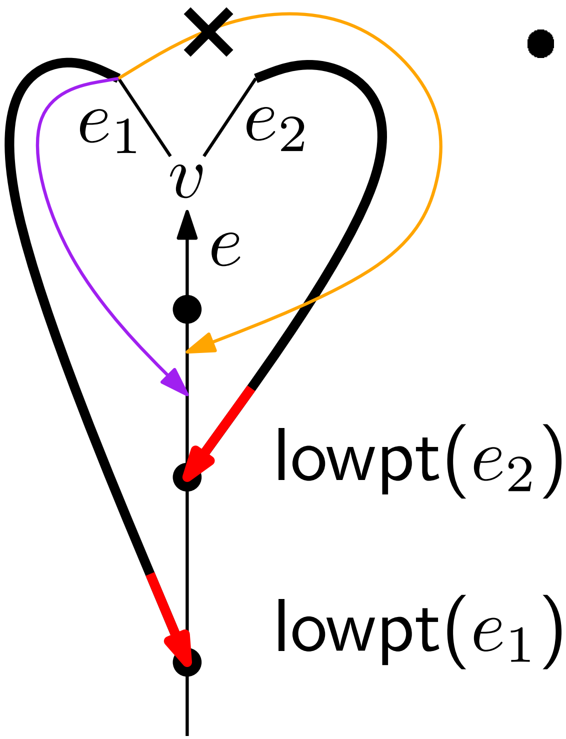


An LR partition is a partition of the back edges into Left and Right so that for every fork

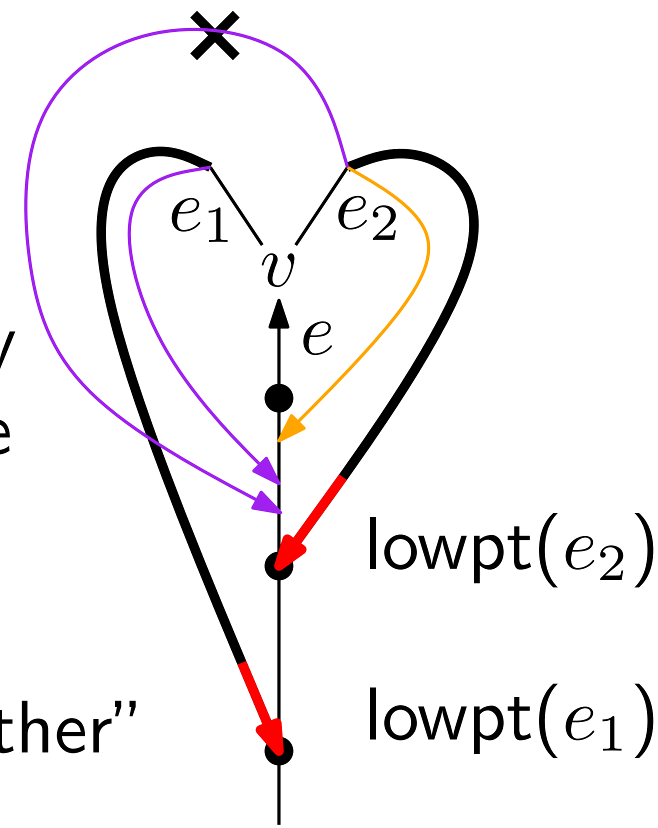


- all return edges of e_1 ending strictly higher than lowpt(e_2) belong to one partition, and
- all return edges of e_2 ending strictly higher than lowpt(e_1) belong to the other.

Fork at $v = 13$ implies
$(14, 12)$ and $(13, 5)$ must be in different LR partitions



- all return edges of e_1 ending strictly higher than lowpt(e_2) belong to one partition



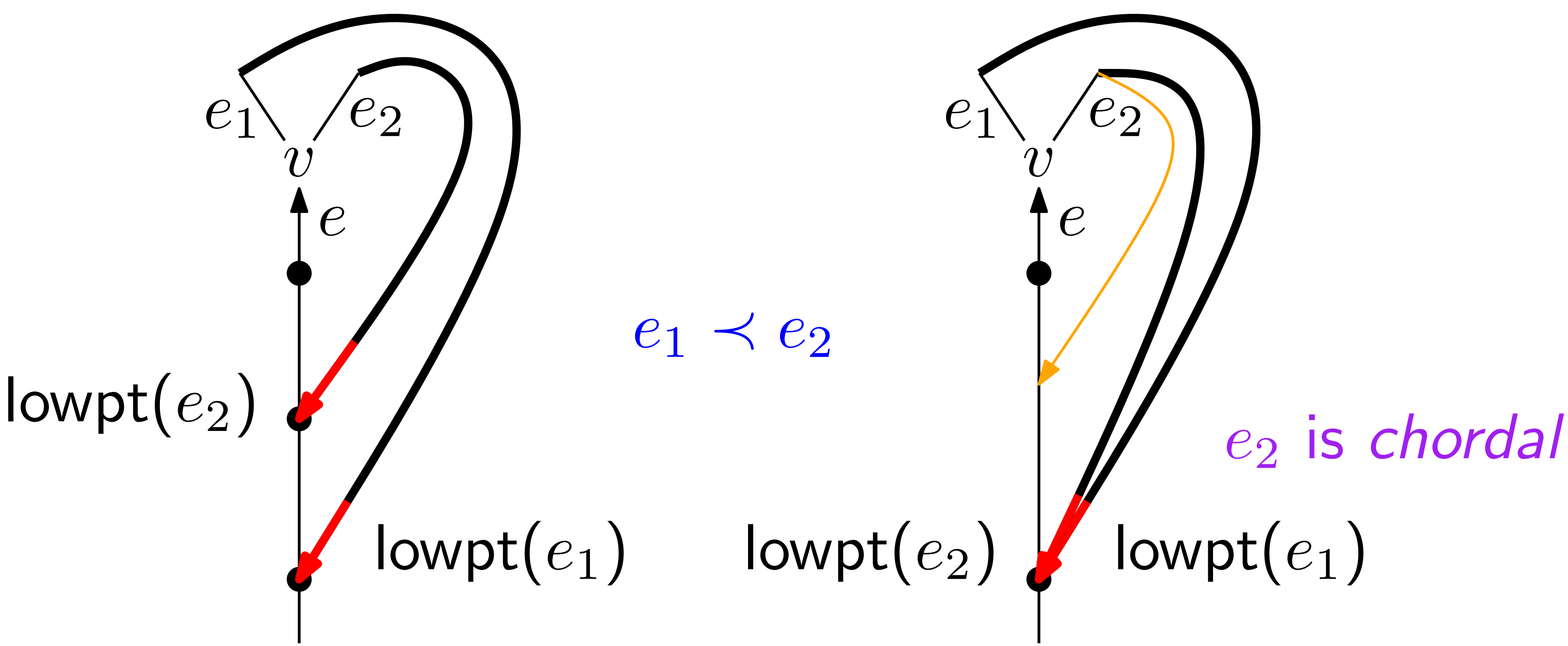
- all return edges of e_2 ending strictly higher than lowpt(e_1) belong to the other.

same constraint $=$ “all...belong”
different constraint $=$ “belong to the other”

Theorem. A graph is planar if and only if it has an LR partition (based on an arbitrary DFS orientation).

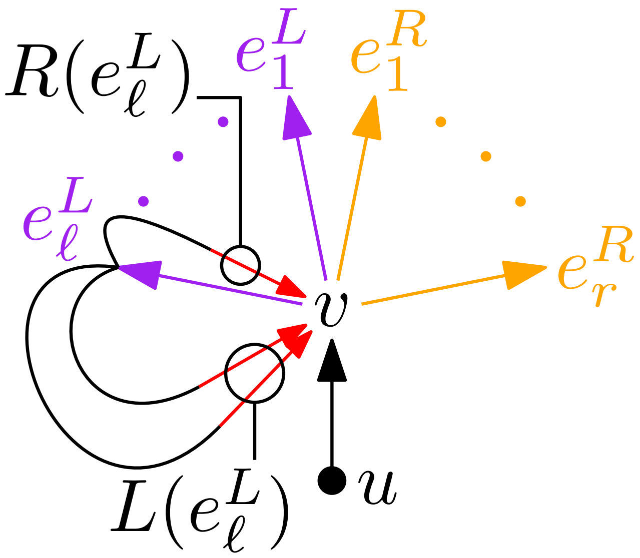
Proof. Use LR-partition to create a nesting order $\prec$ on outgoing edges at each vertex.

(Assign tree edge to same LR-partition as its return edge with the highest return point.)



Let $e_1^L \prec \cdots \prec e_\ell^L$ be the left outgoing edges and $e_1^R \prec \cdots \prec e_r^R$ be the right outgoing edges at v then the (edge) embedding at v is:

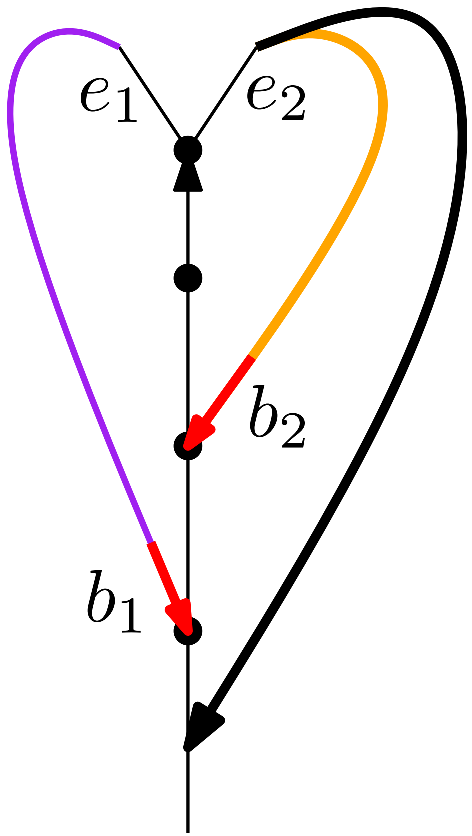
$$\begin{aligned}&(u, v),\\ &L(e_\ell^L), e_\ell^L, R(e_\ell^L), \ldots, L(e_1^L), e_1^L, R(e_1^L),\\ &L(e_1^R), e_1^R, R(e_1^R), \ldots, L(e_r^R), e_r^R, R(e_r^R)\end{aligned}$$



where $L(e)$ and $R(e)$ denote the left and right back edges to v whose cycles share e. Within $R(e)$ (and $L(e)$), back edges are ordered using $\prec$ (and $\succ$) applied to the fork of their cycles.

Algorithm

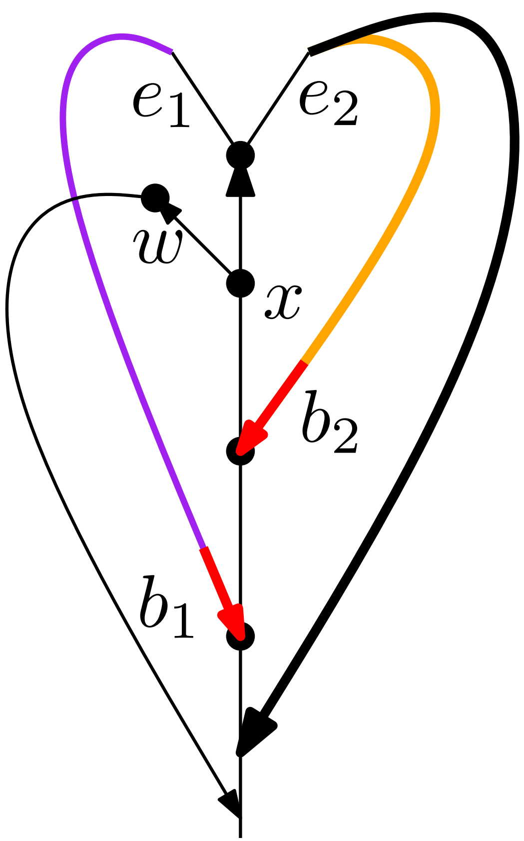
1) For every pair of back edges b_1 and b_2, determine if they should be in the same or different LR-partitions.



different iff $\text{lowpt}(e_2) < \text{lowpt}(b_1)$ and $\text{lowpt}(e_1) < \text{lowpt}(b_2)$

Algorithm

1) For every pair of back edges b_1 and b_2, determine if they should be in the same or different LR-partitions.



different iff lowpt(e_2) $<$ lowpt(b_1) and lowpt(e_1) $<$ lowpt(b_2)

same if lowpt$((x,w))$ $< \min\{$lowpt(b_1), lowpt$(b_2)\}$ for some (x, w) where x is shared by $C(b_1)$ and $C(b_2)$ but w is not.

Algorithm

2) Create a constraint graph on back edges: constraint edge (b_1, b_2) is red if b_1 and b_2 are different, or blue if the same.

3) Find a balanced bipartition of the constraint graph: red edges connect vertices (i.e., back edges) in different partitions, blue edges connect vertices in the same partition.