Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Chair for
INFORMATICS I
Efficient Algorithms and
Knowledge-Based Systems

Institute for Informatics

Simultaneous Drawing of Planar Graphs with Right-Angle Crossings and Few Bends

Philipp Kindermann
Chair of Computer Science I
Universität Würzburg

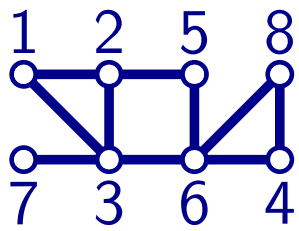
Joint work with
Michael A. Bekos, Thomas C. van Dijk & Alexander Wolff

Simultaneous Embedding

Given: Two planar graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$
on same vertex set

Simultaneous Embedding

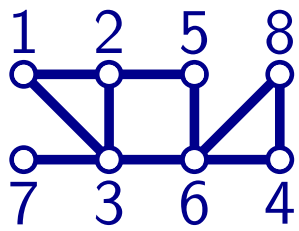
Given: Two planar graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$
on same vertex set



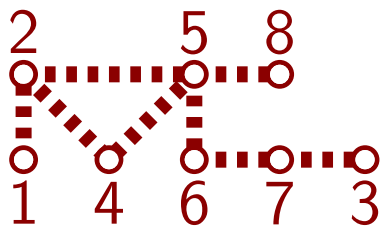
G_1

Simultaneous Embedding

Given: Two planar graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ on same vertex set



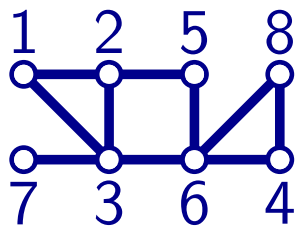
G_1



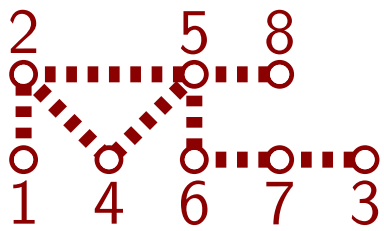
G_2

Simultaneous Embedding

Given: Two planar graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$
on same vertex set



G_1

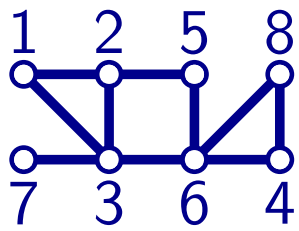


G_2

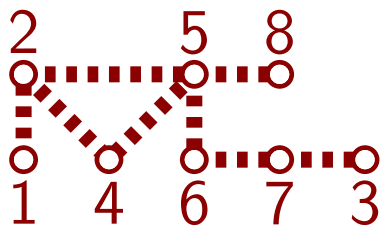
Embed both graphs in a planar way
– edges of one graph may intersect edges of the other graph

Simultaneous Embedding

Given: Two planar graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$
on same vertex set



G_1



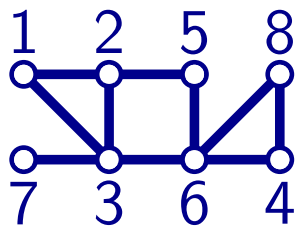
G_2

Embed both graphs in a planar way
– edges of one graph may intersect edges of the other graph

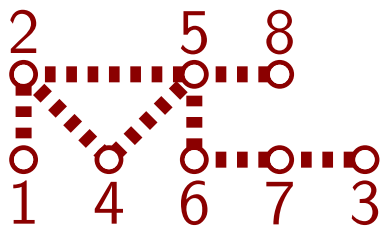
- Sim. Geometric Emb. (SGE)

Simultaneous Embedding

Given: Two planar graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$
on same vertex set



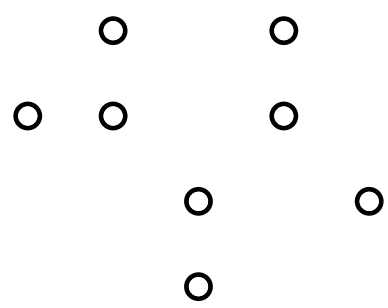
G_1



G_2

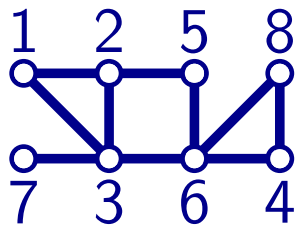
Embed both graphs in a planar way
– edges of one graph may intersect edges of the other graph

- Sim. Geometric Emb. (SGE)

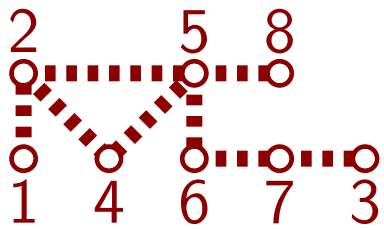


Simultaneous Embedding

Given: Two planar graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$
on same vertex set



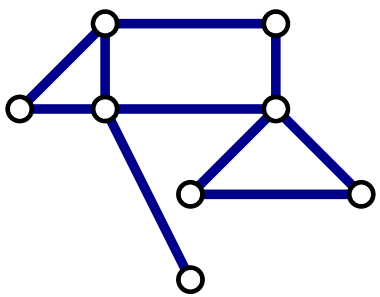
G_1



G_2

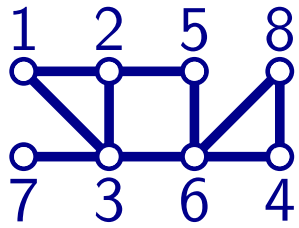
Embed both graphs in a planar way
– edges of one graph may intersect edges of the other graph

- Sim. Geometric Emb. (SGE)

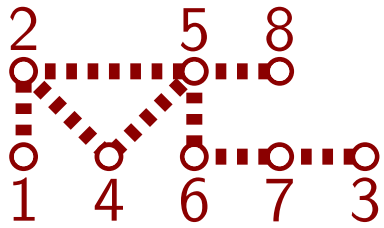


Simultaneous Embedding

Given: Two planar graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$
on same vertex set



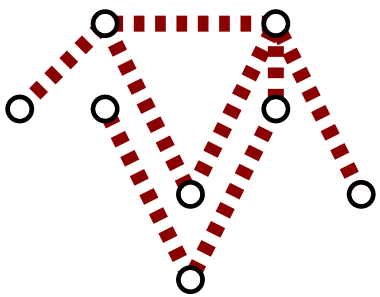
G_1



G_2

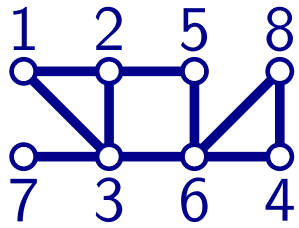
Embed both graphs in a planar way
– edges of one graph may intersect edges of the other graph

- Sim. Geometric Emb. (SGE)

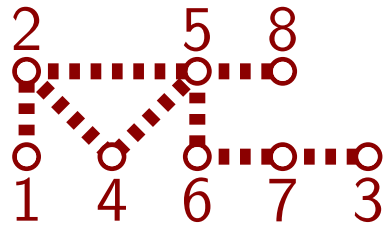


Simultaneous Embedding

Given: Two planar graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$
on same vertex set



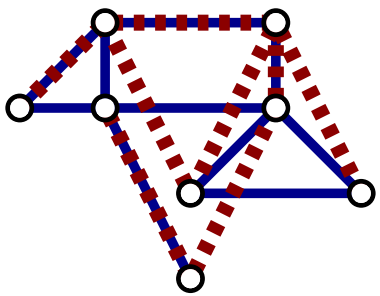
G_1



G_2

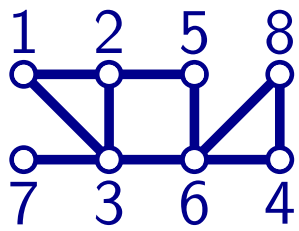
Embed both graphs in a planar way
– edges of one graph may intersect edges of the other graph

- Sim. Geometric Emb. (SGE)

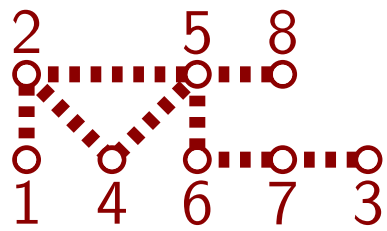


Simultaneous Embedding

Given: Two planar graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$
on same vertex set



G_1



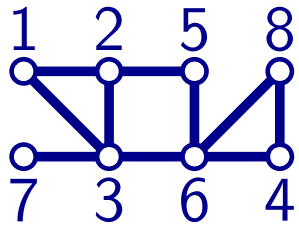
G_2

Embed both graphs in a planar way
– edges of one graph may intersect edges of the other graph

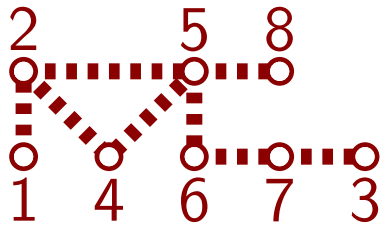
- Sim. Geometric Emb. (SGE)
- Sim. Emb. with Fixed Edges (SEFE)

Simultaneous Embedding

Given: Two planar graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$
on same vertex set



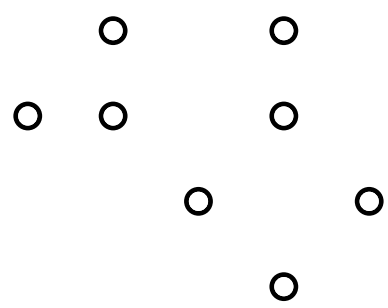
G_1



G_2

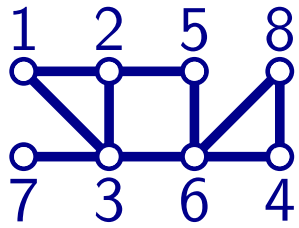
Embed both graphs in a planar way
– edges of one graph may intersect edges of the other graph

- Sim. Geometric Emb. (SGE)
- Sim. Emb. with Fixed Edges (SEFE)

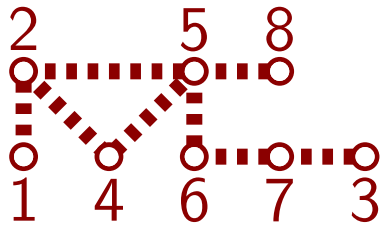


Simultaneous Embedding

Given: Two planar graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$
on same vertex set



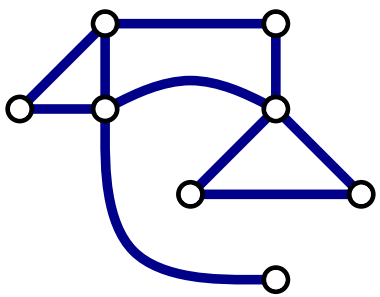
G_1



G_2

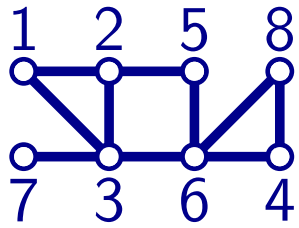
Embed both graphs in a planar way
– edges of one graph may intersect edges of the other graph

- Sim. Geometric Emb. (SGE)
- Sim. Emb. with Fixed Edges (SEFE)

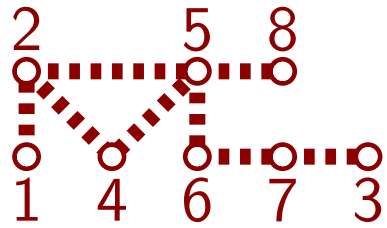


Simultaneous Embedding

Given: Two planar graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$
on same vertex set



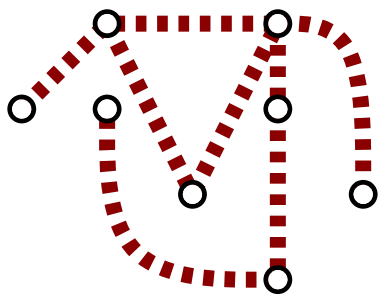
G_1



G_2

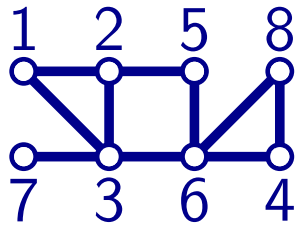
Embed both graphs in a planar way
– edges of one graph may intersect edges of the other graph

- Sim. Geometric Emb. (SGE)
- Sim. Emb. with Fixed Edges (SEFE)

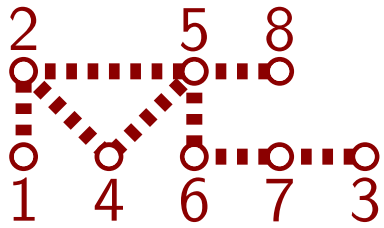


Simultaneous Embedding

Given: Two planar graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$
on same vertex set



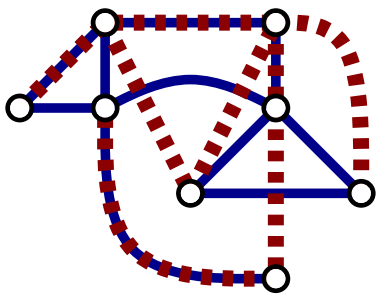
G_1



G_2

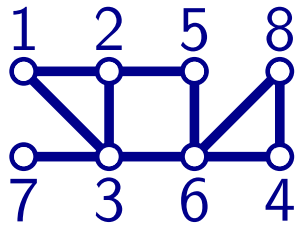
Embed both graphs in a planar way
– edges of one graph may intersect edges of the other graph

- Sim. Geometric Emb. (SGE)
- Sim. Emb. with Fixed Edges (SEFE)

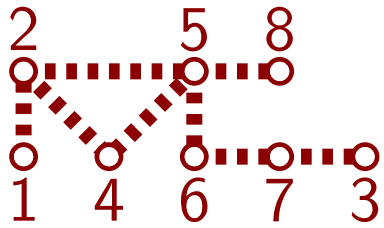


Simultaneous Embedding

Given: Two planar graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$
on same vertex set



G_1



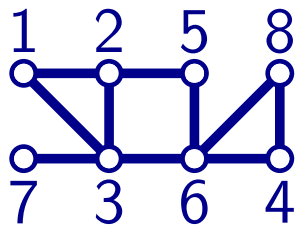
G_2

Embed both graphs in a planar way
– edges of one graph may intersect edges of the other graph

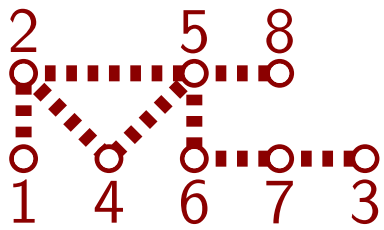
- Sim. Geometric Emb. (SGE)
- Sim. Emb. with Fixed Edges (SEFE)
- Sim. Emb. (SE)

Simultaneous Embedding

Given: Two planar graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$
on same vertex set



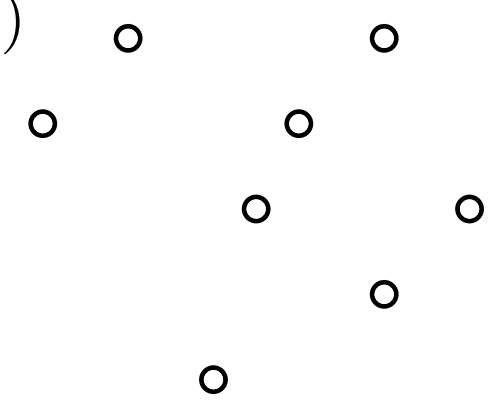
G_1



G_2

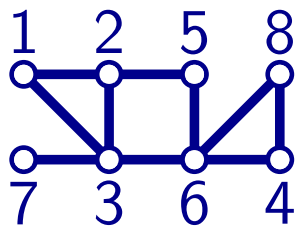
Embed both graphs in a planar way
– edges of one graph may intersect edges of the other graph

- Sim. Geometric Emb. (SGE)
- Sim. Emb. with Fixed Edges (SEFE)
- Sim. Emb. (SE)

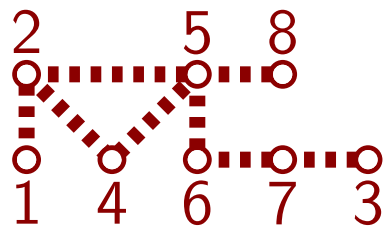


Simultaneous Embedding

Given: Two planar graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$
on same vertex set



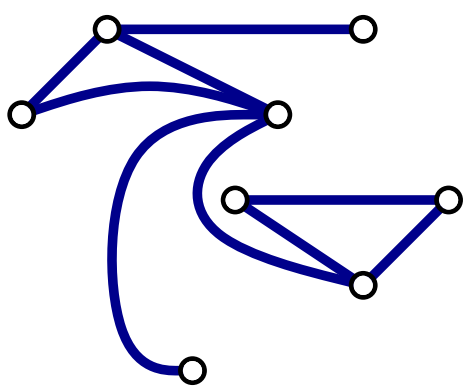
G_1



G_2

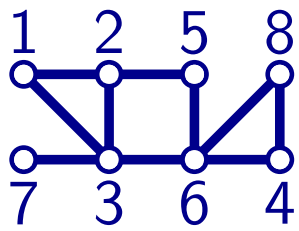
Embed both graphs in a planar way
– edges of one graph may intersect edges of the other graph

- Sim. Geometric Emb. (SGE)
- Sim. Emb. with Fixed Edges (SEFE)
- Sim. Emb. (SE)

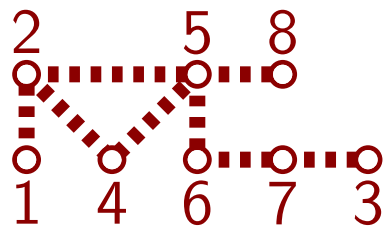


Simultaneous Embedding

Given: Two planar graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$
on same vertex set



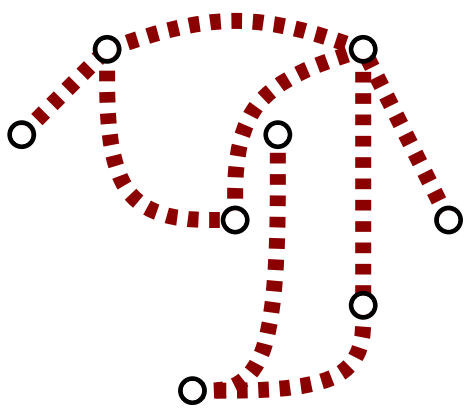
G_1



G_2

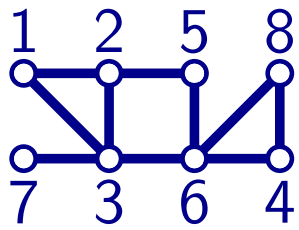
Embed both graphs in a planar way
– edges of one graph may intersect edges of the other graph

- Sim. Geometric Emb. (SGE)
- Sim. Emb. with Fixed Edges (SEFE)
- Sim. Emb. (SE)

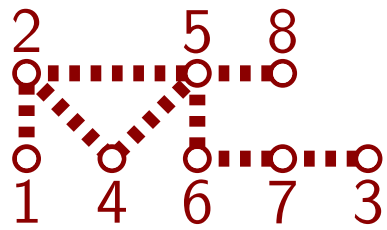


Simultaneous Embedding

Given: Two planar graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$
on same vertex set



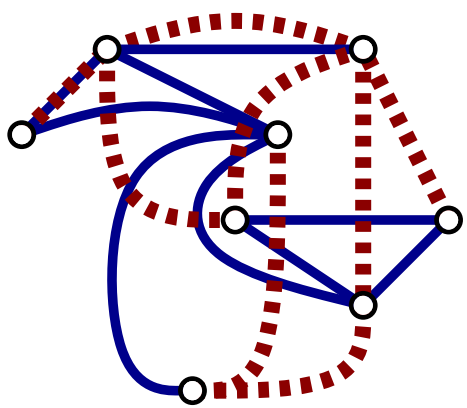
G_1



G_2

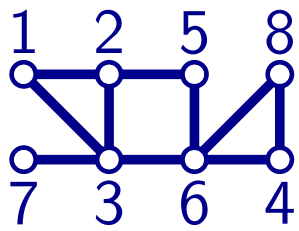
Embed both graphs in a planar way
– edges of one graph may intersect edges of the other graph

- Sim. Geometric Emb. (SGE)
- Sim. Emb. with Fixed Edges (SEFE)
- Sim. Emb. (SE)

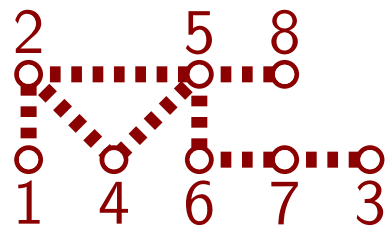


Simultaneous Embedding

Given: Two planar graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$
on same vertex set



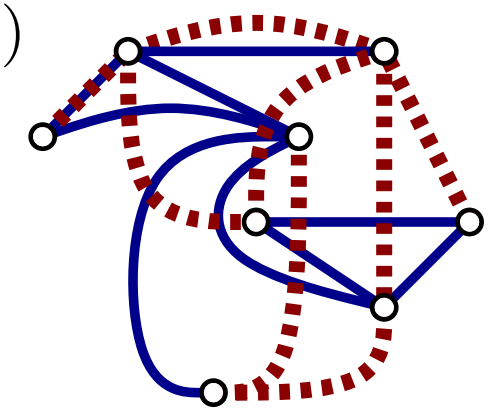
G_1



G_2

Embed both graphs in a planar way
– edges of one graph may intersect edges of the other graph

- Sim. Geometric Emb. (SGE)
- Sim. Emb. with Fixed Edges (SEFE)
- Sim. Emb. (SE)

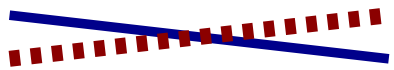


RacSim Drawings

RacSim: Simultaneous Embedding with Right-Angle Crossings

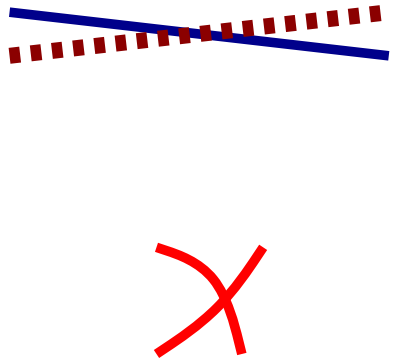
RacSim Drawings

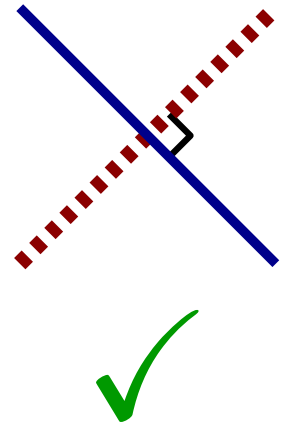
RacSim: Simultaneous Embedding with Right-Angle Crossings



RacSim Drawings

RacSim: Simultaneous Embedding with Right-Angle Crossings





RacSim Drawings

RacSim: Simultaneous Embedding with Right-Angle Crossings

Geometric RacSim: RacSim with straight-line edges

Known results

[Angelini et al. '12]
There exist a tree and a path
which don't admit an SGE drawing

Known results

[Angelini et al. '12]
There exist a tree and a path
which don't admit an SGE drawing

[Cabello et al. '11]
There exist a tree and a matching
which require exponential area for an SGE drawing

Known results

[Angelini et al. '12]
There exist a tree and a path
which don't admit an SGE drawing

[Cabello et al. '11]
There exist a tree and a matching
which require exponential area for an SGE drawing

[Erten & Kobourov '05]
Any pair of trees admits an SE drawing
with one bend and quadratic area

Known results

[Angelini et al. '12]
There exist a tree and a path
which don't admit an SGE drawing

[Cabello et al. '11]
There exist a tree and a matching
which require exponential area for an SGE drawing

[Erten & Kobourov '05]
Any pair of trees admits an SE drawing
with one bend and quadratic area

[Kammer '06]
Any pair of planar graphs admits an SE drawing
with two bends and quadratic area

Known results

[Angelini et al. '12]
There exist a tree and a path
which don't admit an SGE drawing

[Cabello et al. '11]
There exist a tree and a matching
which require exponential area for an SGE drawing

[Erten & Kobourov '05]
Any pair of trees admits an SE drawing
with one bend and quadratic area

[Kammer '06]
Any pair of planar graphs admits an SE drawing
with two bends and quadratic area

} low angle

Known results

[Argyrious et al. '13]
A cycle and a matching always admit a geometric RacSim drawing in quadratic area

[Erten & Kobourov '05]
Any pair of trees admits an SE drawing with one bend and quadratic area

[Kammer '06]
Any pair of planar graphs admits an SE drawing with two bends and quadratic area

} low angle

Known results

[Argyrious et al. '13]
A cycle and a matching always admit a geometric RacSim drawing in quadratic area

There exist a wheel and a cycle which don't admit a geometric RacSim drawing

[Erten & Kobourov '05]
Any pair of trees admits an SE drawing with one bend and quadratic area

[Kammer '06]
Any pair of planar graphs admits an SE drawing with two bends and quadratic area

} low angle

Our results

Graph classes			Number of bends
Cycle	$\times$	Cycle	1×1
Caterpillar	$\times$	Cycle	1×1
Four Matchings			$1 \times 1 \times 1 \times 1$
Tree	$\times$	Matching	1×0
Wheel	$\times$	Matching	2×0
Outerpath	$\times$	Matching	2×1
Outerplanar	$\times$	Outerplanar	3×3
2-page book emb.	$\times$	2-page book emb.	4×4
Planar	$\times$	Planar	6×6

Our results

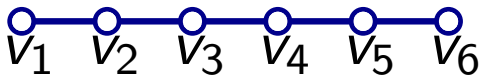
Graph classes			Number of bends
Cycle	$\times$	Cycle	1×1
Caterpillar	$\times$	Cycle	1×1
Four Matchings			$1 \times 1 \times 1 \times 1$
Tree	$\times$	Matching	1×0
Wheel	$\times$	Matching	2×0
Outerpath	$\times$	Matching	2×1
Outerplanar	$\times$	Outerplanar	3×3
2-page book emb.	$\times$	2-page book emb.	4×4
Planar	$\times$	Planar	6×6

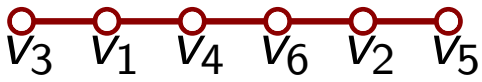
Path $\times$ Path

Path $\times$ Path

Path $\times$ Path

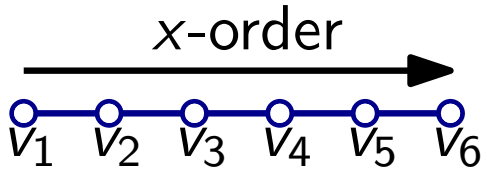
Path $\times$ Path

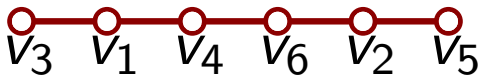


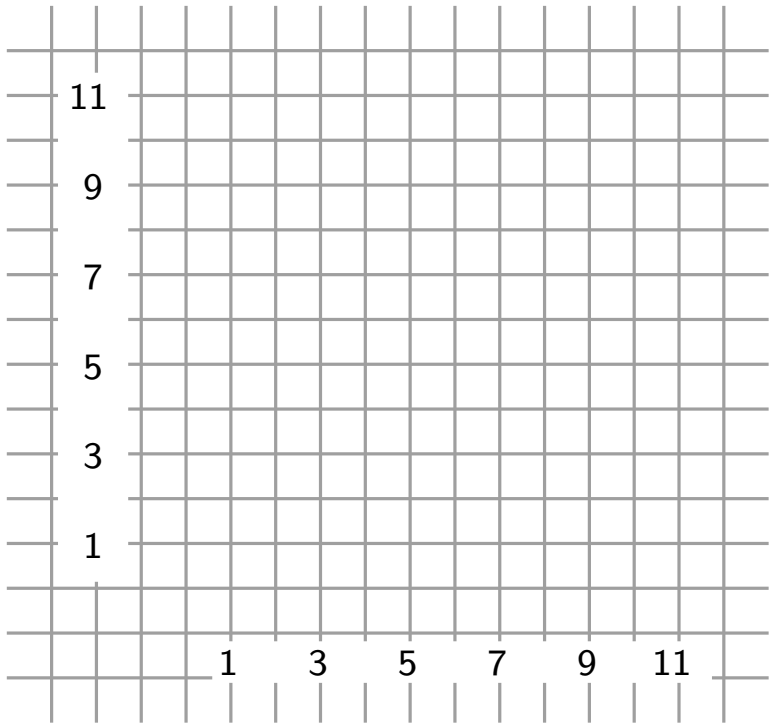


Path $\times$ Path

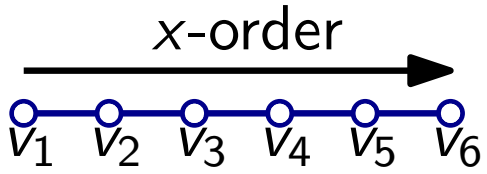
Path $\times$ Path

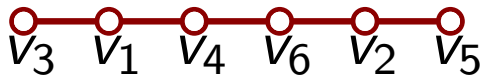


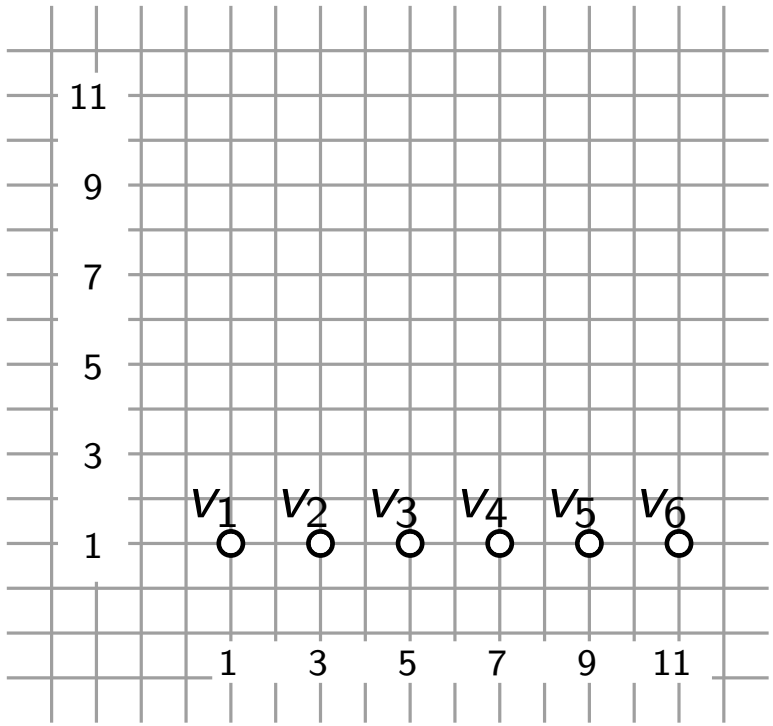




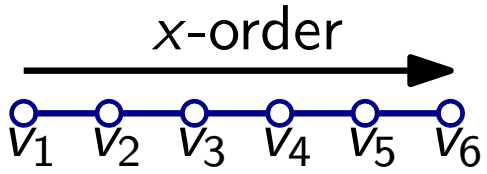
Path × Path

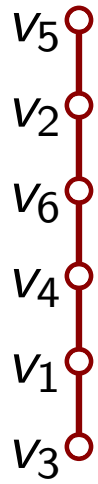

x-order
v_1 v_2 v_3 v_4 v_5 v_6

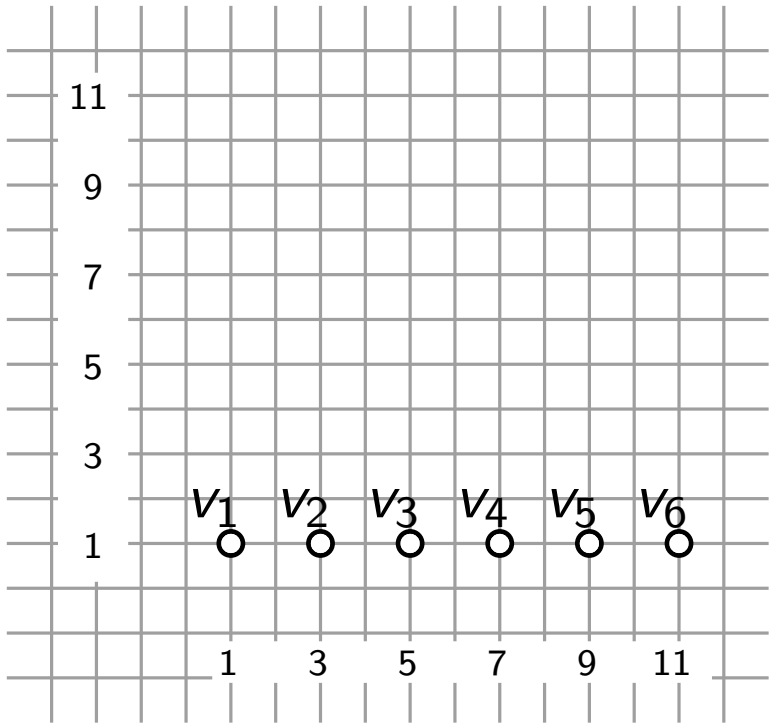

v_3 v_1 v_4 v_6 v_2 v_5


11
9
7
5
3
1
v_1 v_2 v_3 v_4 v_5 v_6
1 3 5 7 9 11

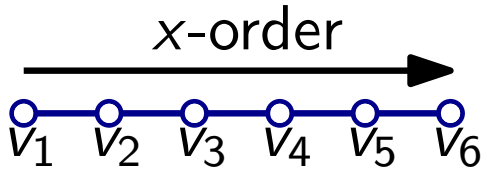
Path × Path

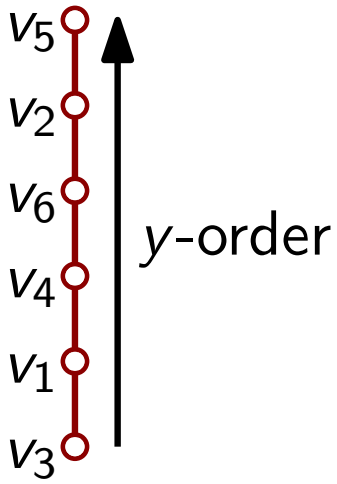

x-order
v1 v2 v3 v4 v5 v6

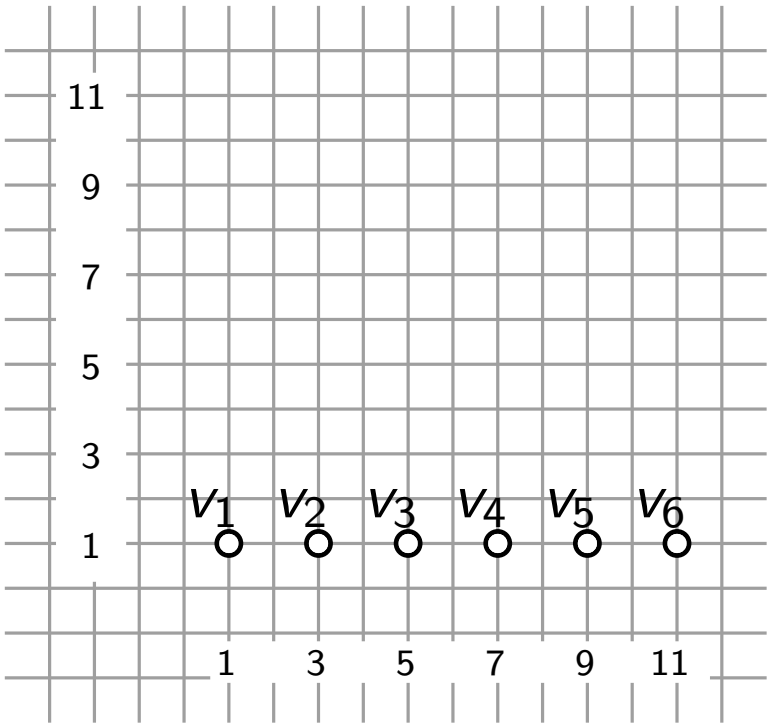

v5
v2
v6
v4
v1
v3


11
9
7
5
3
1
v1 v2 v3 v4 v5 v6
1 3 5 7 9 11

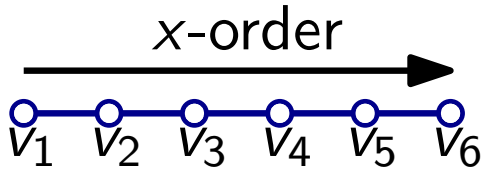
Path × Path

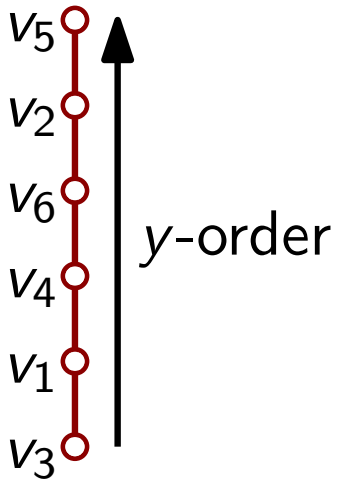

x-order
v1 v2 v3 v4 v5 v6

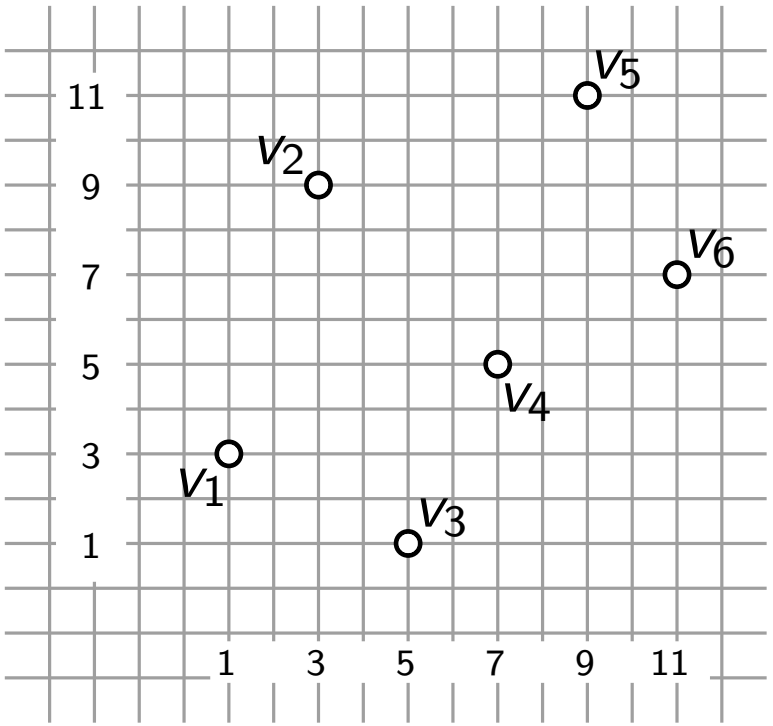

v5
v2
v6
v4
v1
v3
y-order


11
9
7
5
3
1
v1 v2 v3 v4 v5 v6
1 3 5 7 9 11

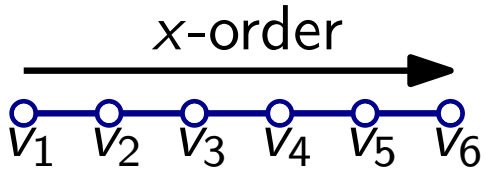
Path × Path

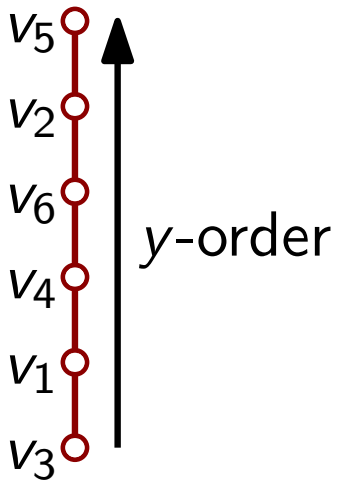

x-order
v1 v2 v3 v4 v5 v6


v5
v2
v6
v4
v1
v3
y-order

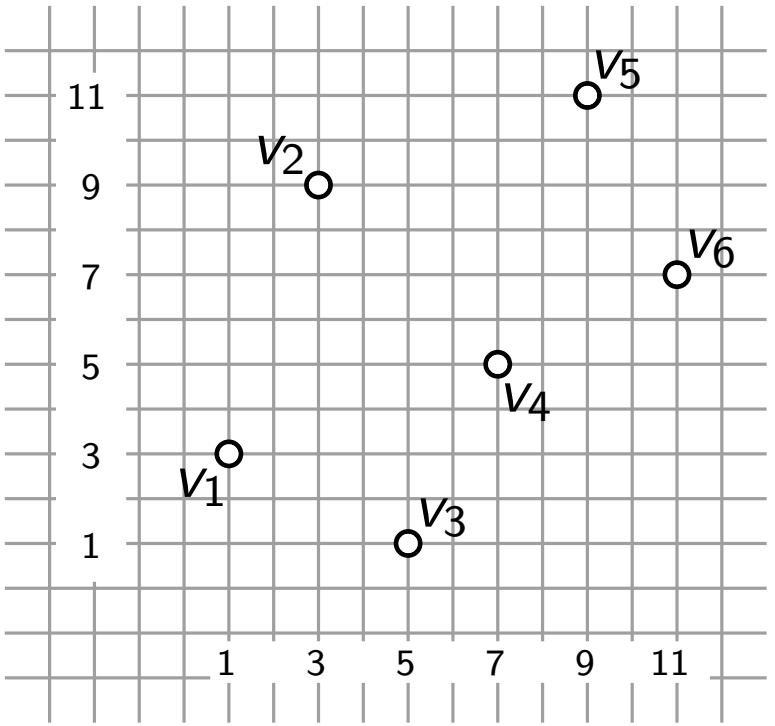

11
9
7
5
3
1
1 3 5 7 9 11
v5
v2
v6
v4
v1
v3

Path × Path

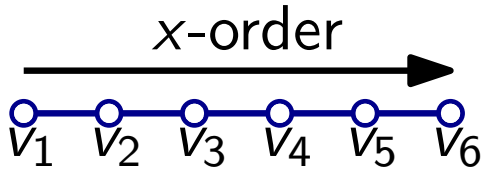


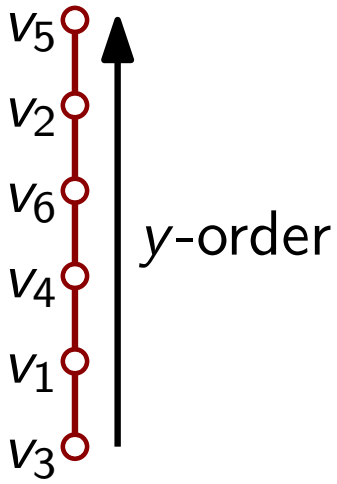


Edges:

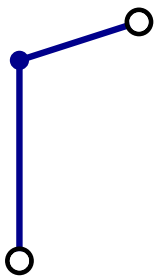


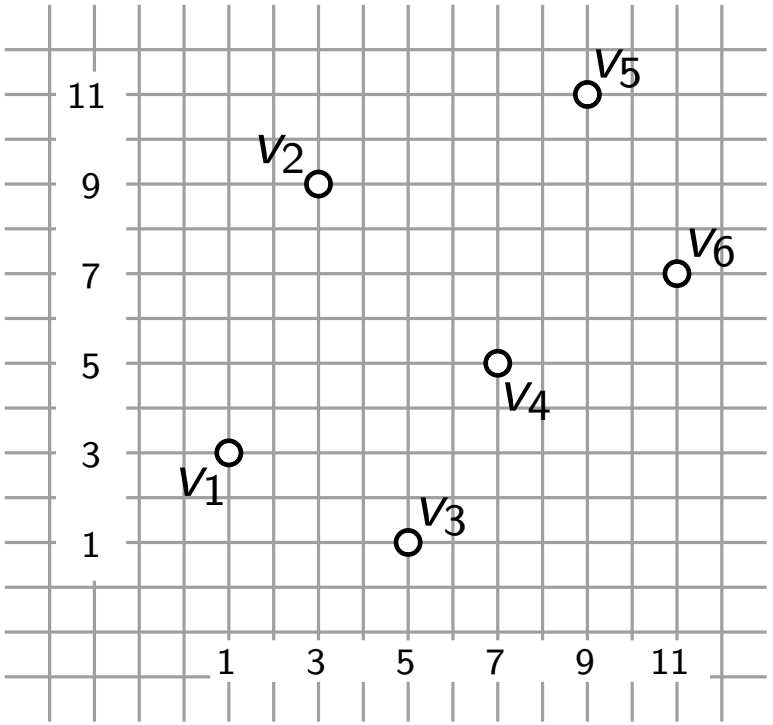
Path × Path



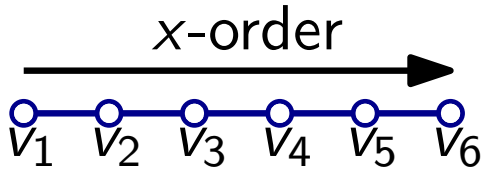


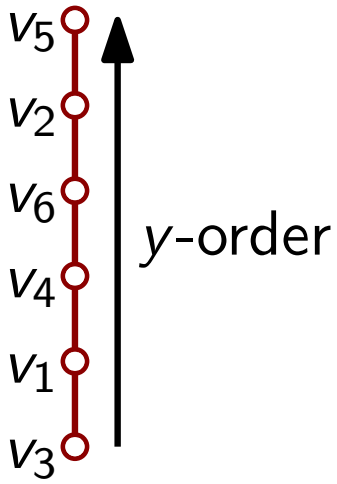
Edges:



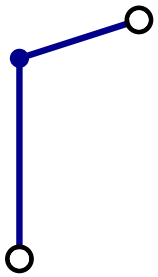


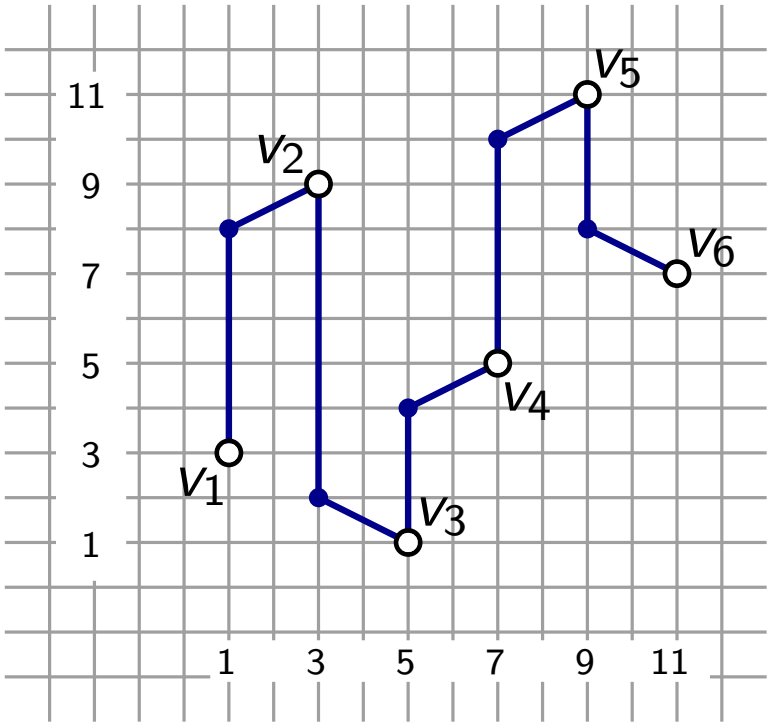
Path × Path



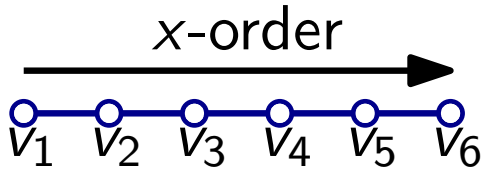


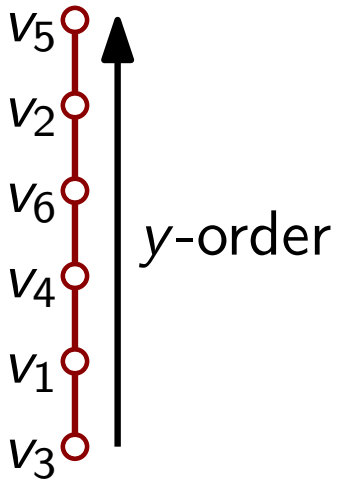
Edges:



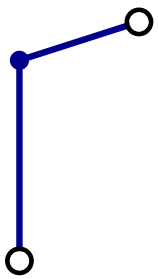


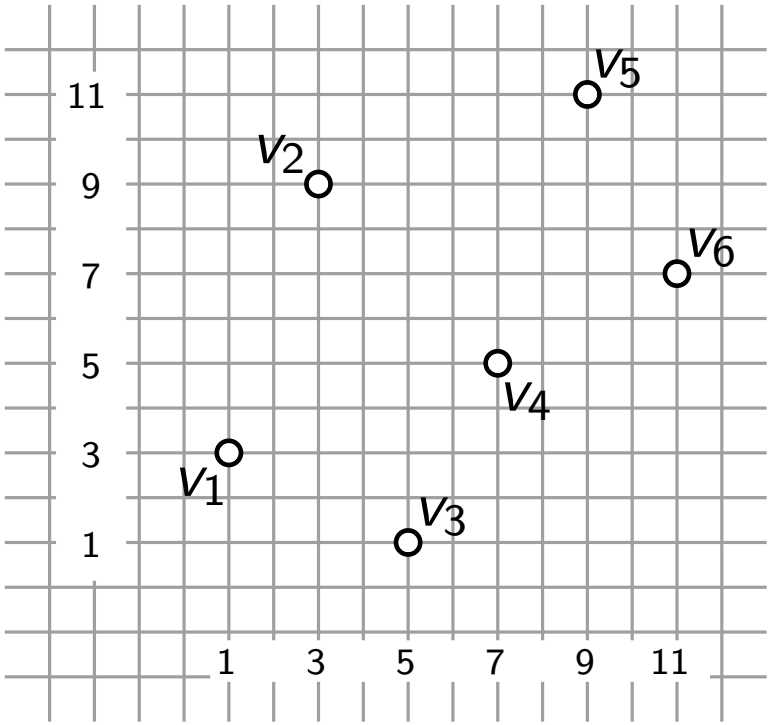
Path × Path



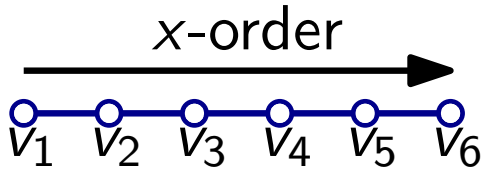


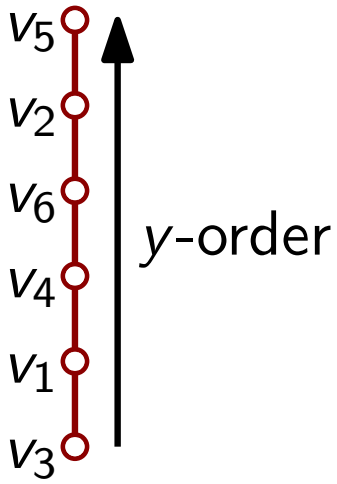
Edges:



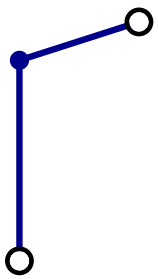


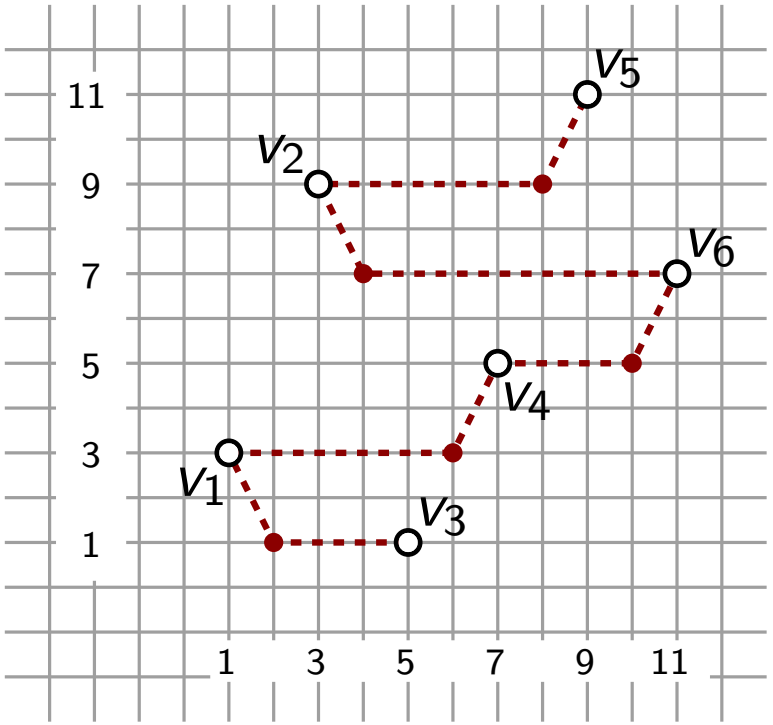
Path × Path



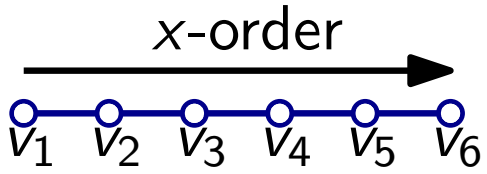


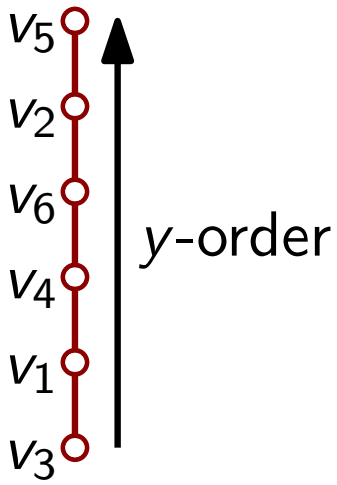
Edges:



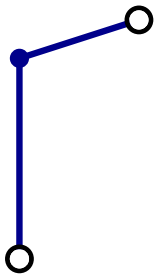


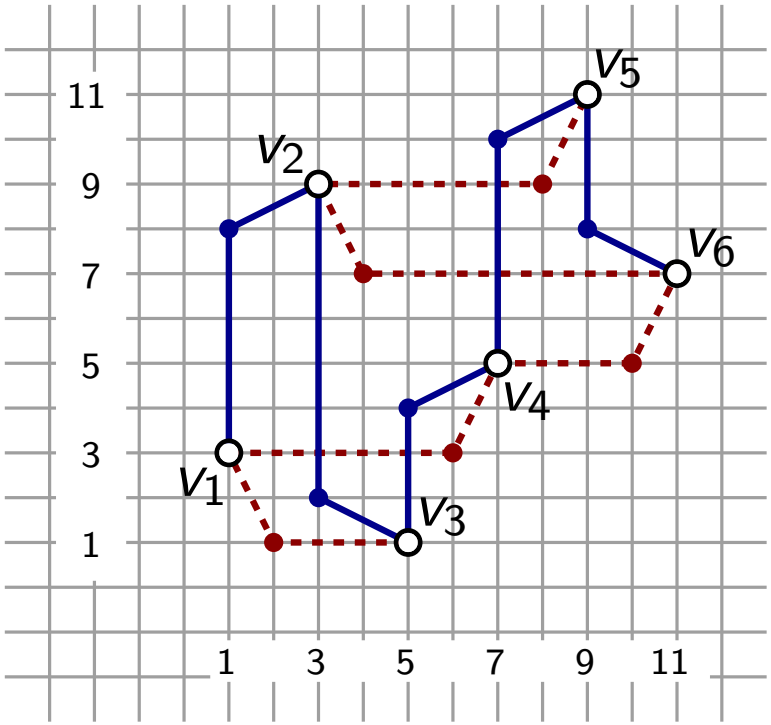
Path × Path



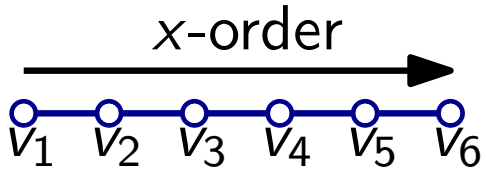


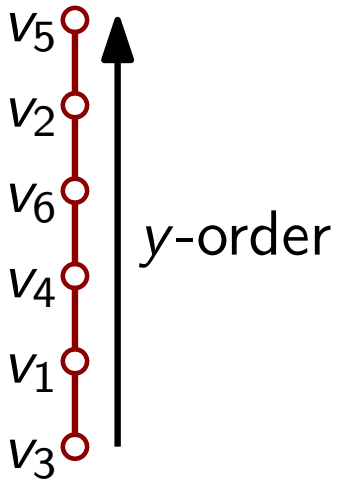
Edges:



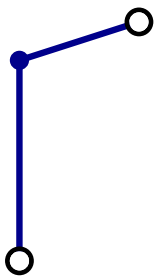


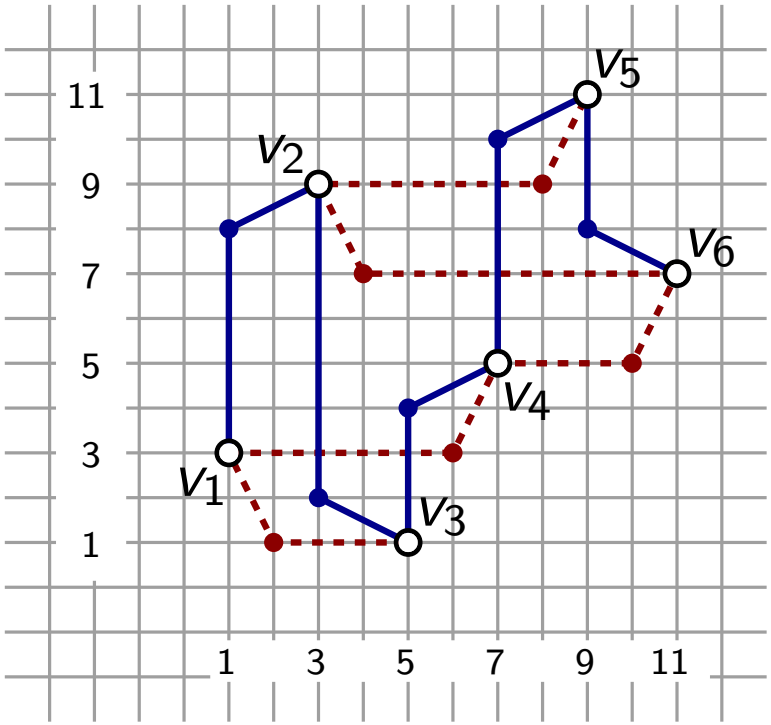
Cycle × Cycle



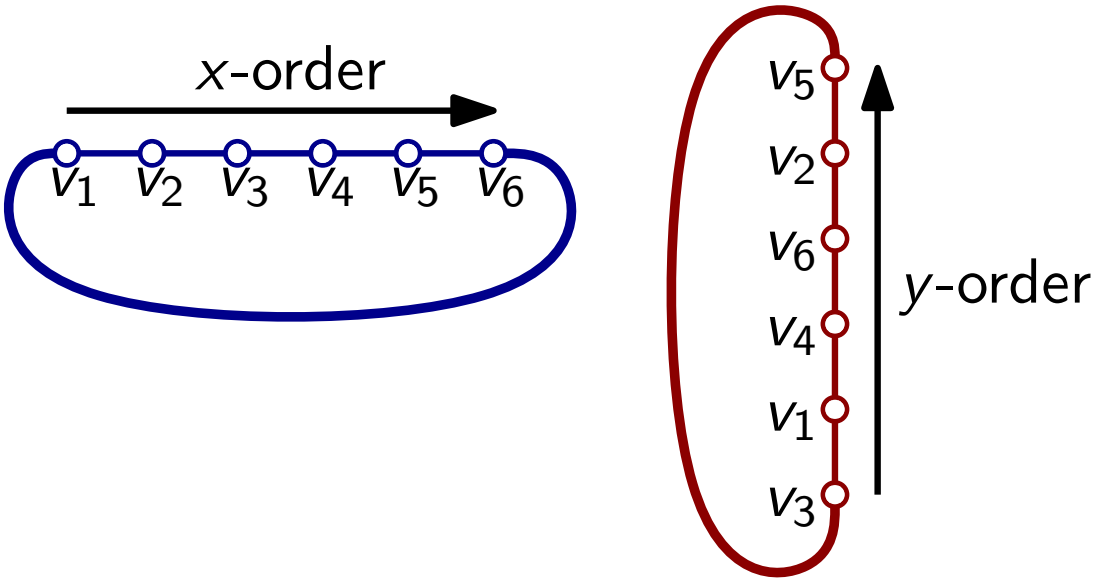


Edges:

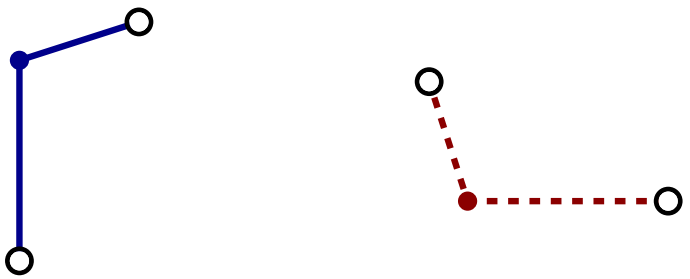


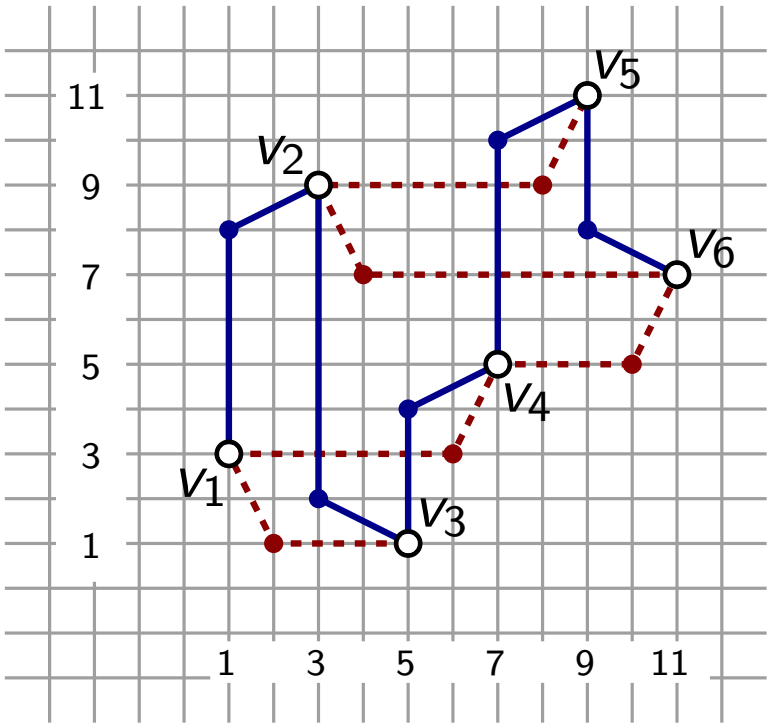


Cycle × Cycle

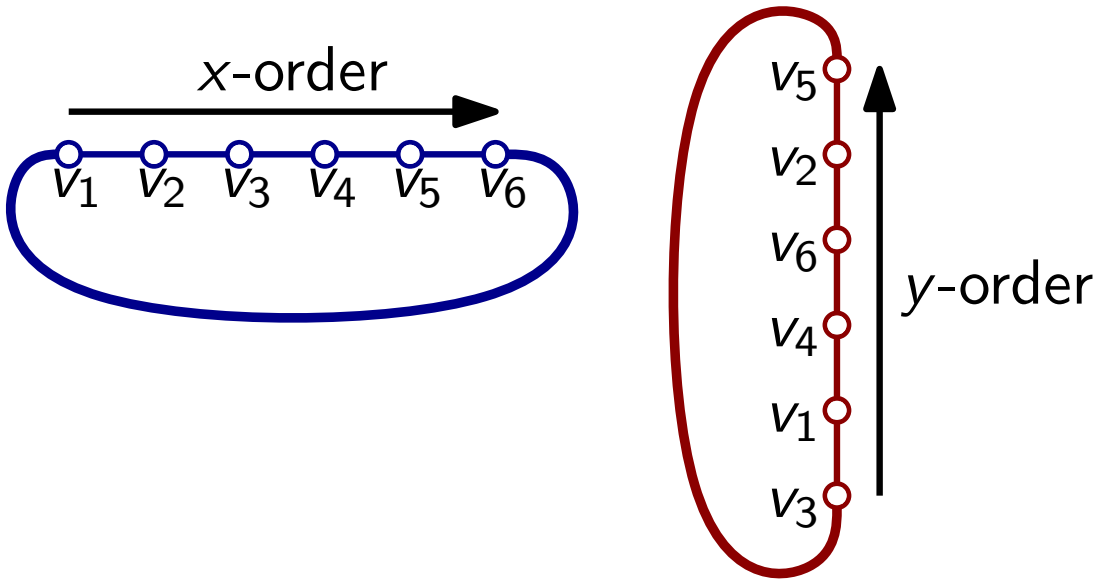


Edges:

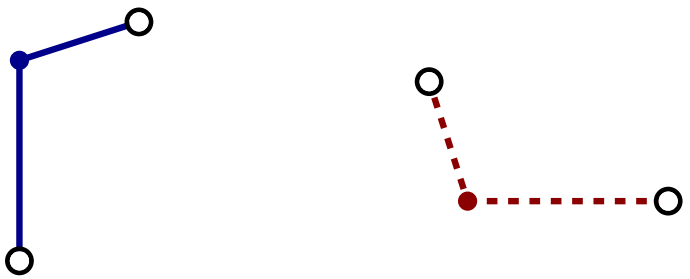


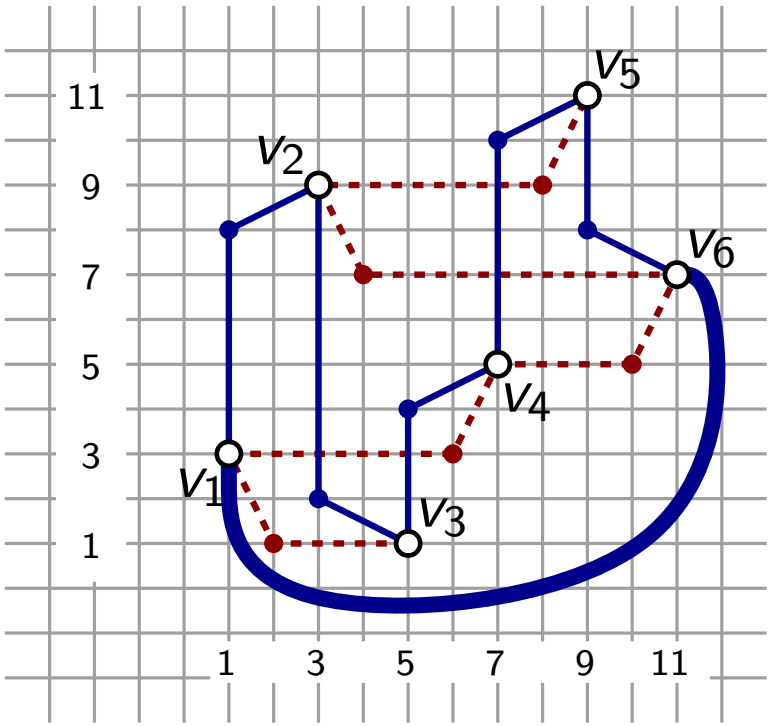


Cycle × Cycle

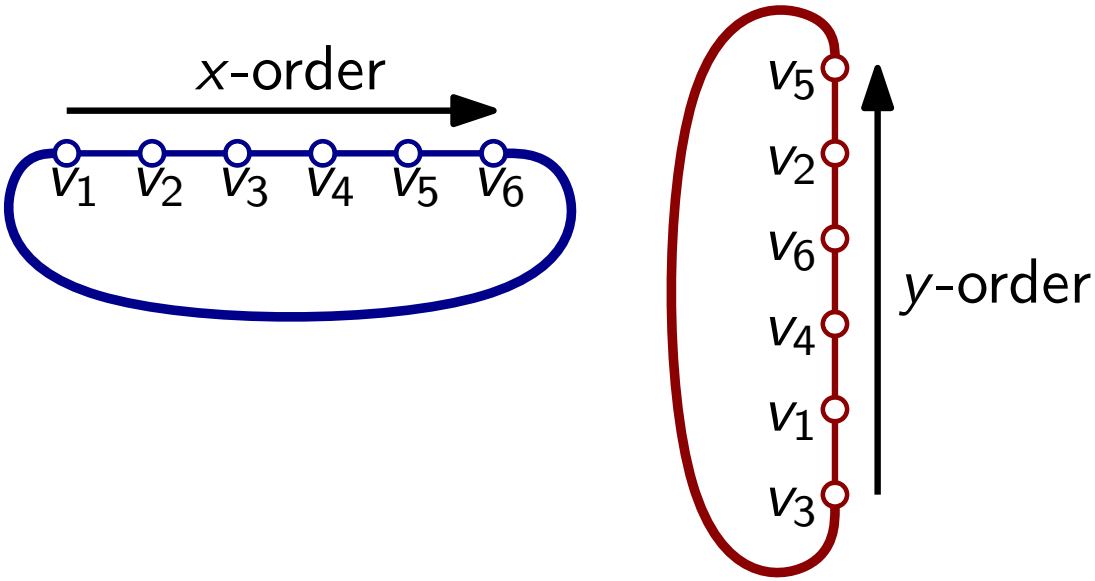


Edges:

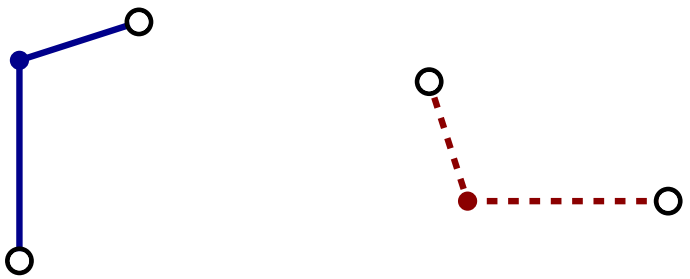


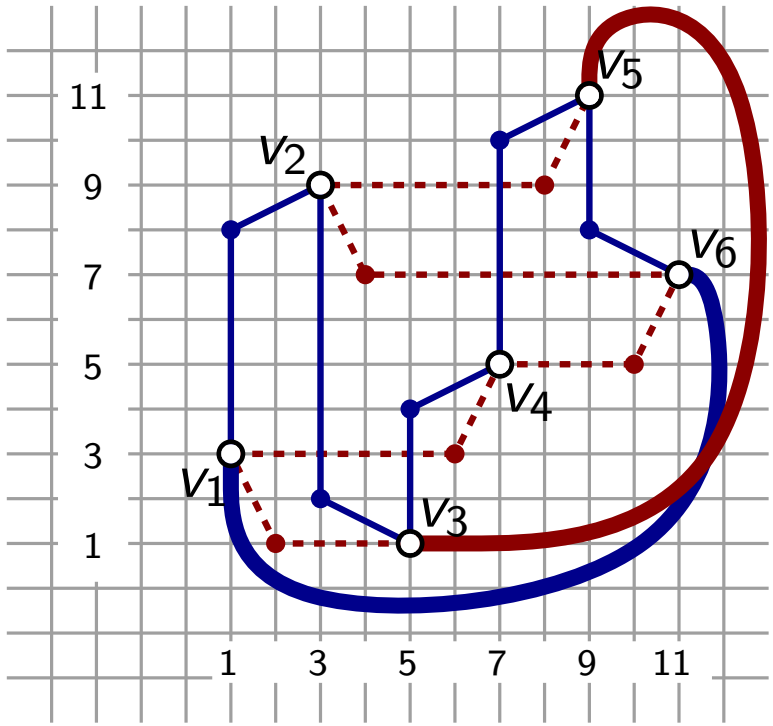


Cycle × Cycle

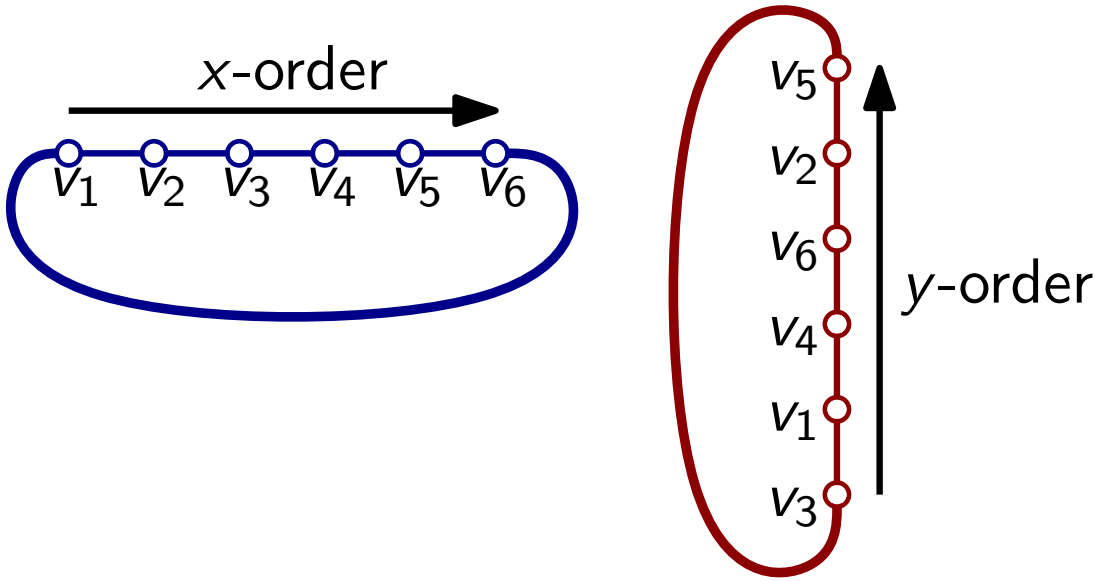


Edges:

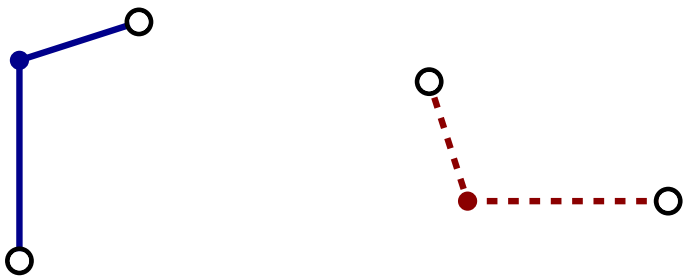


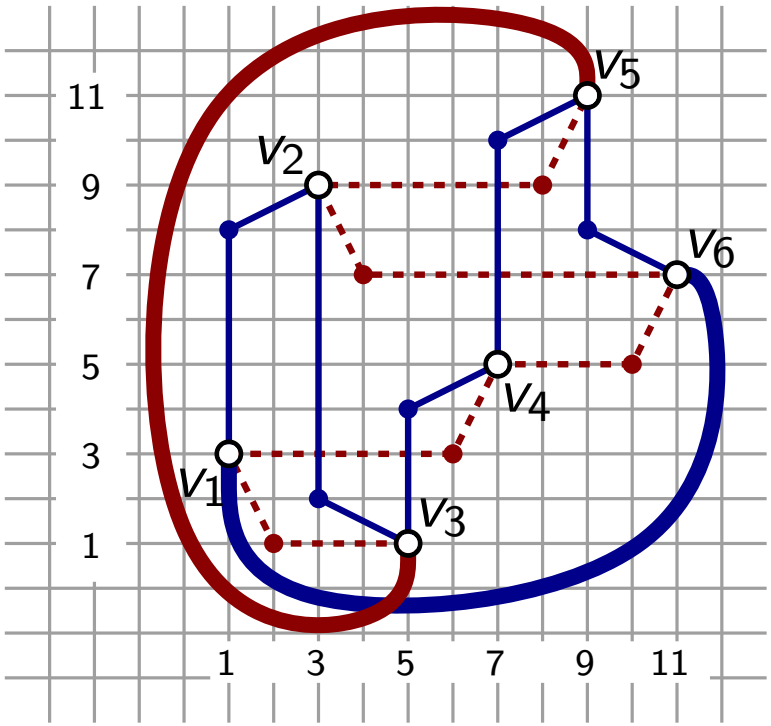


Cycle × Cycle

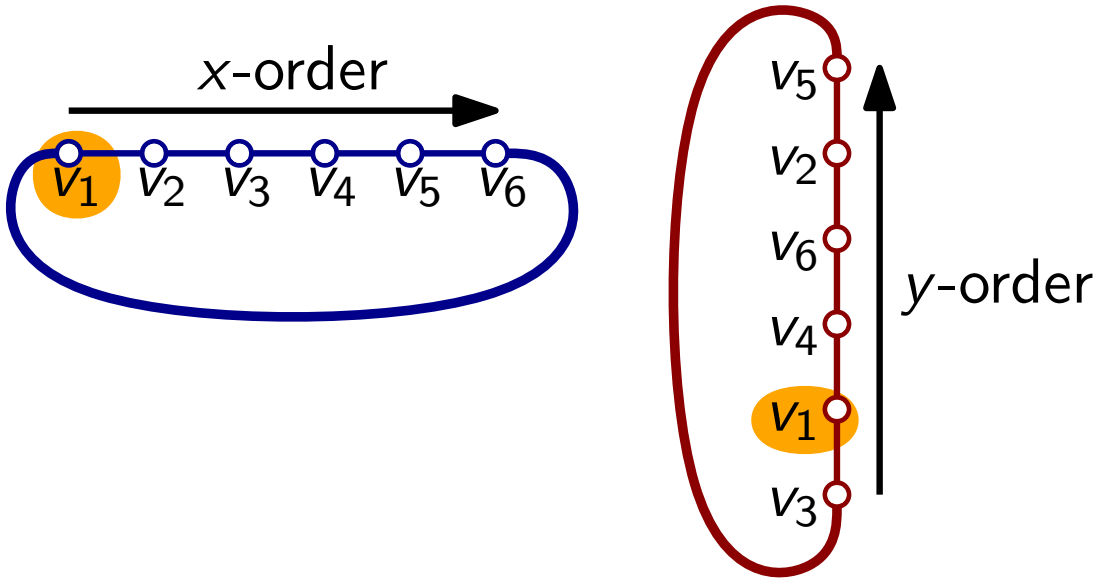


Edges:

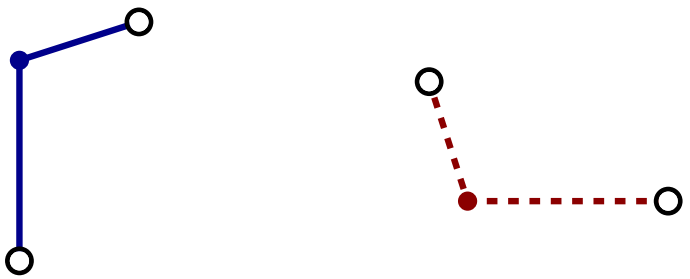


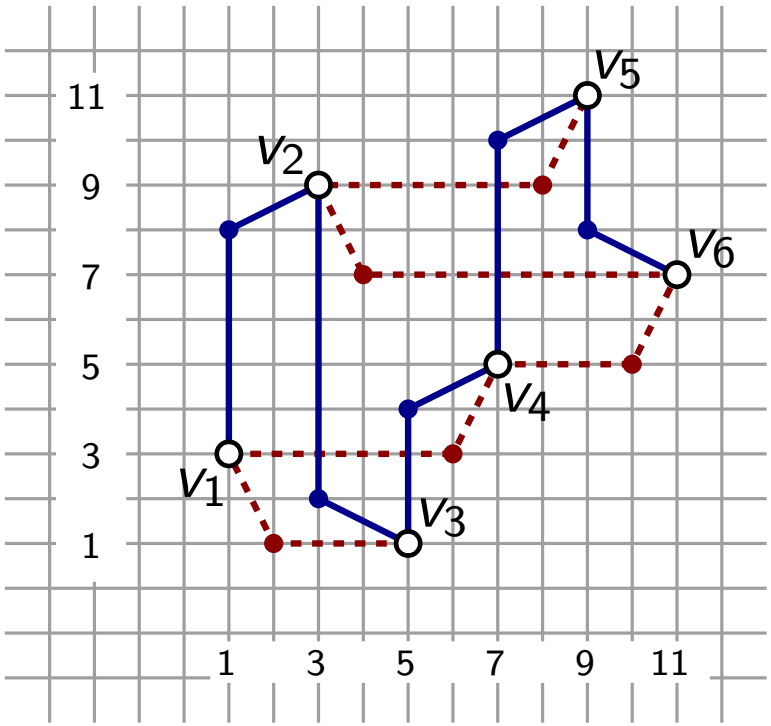


Cycle × Cycle

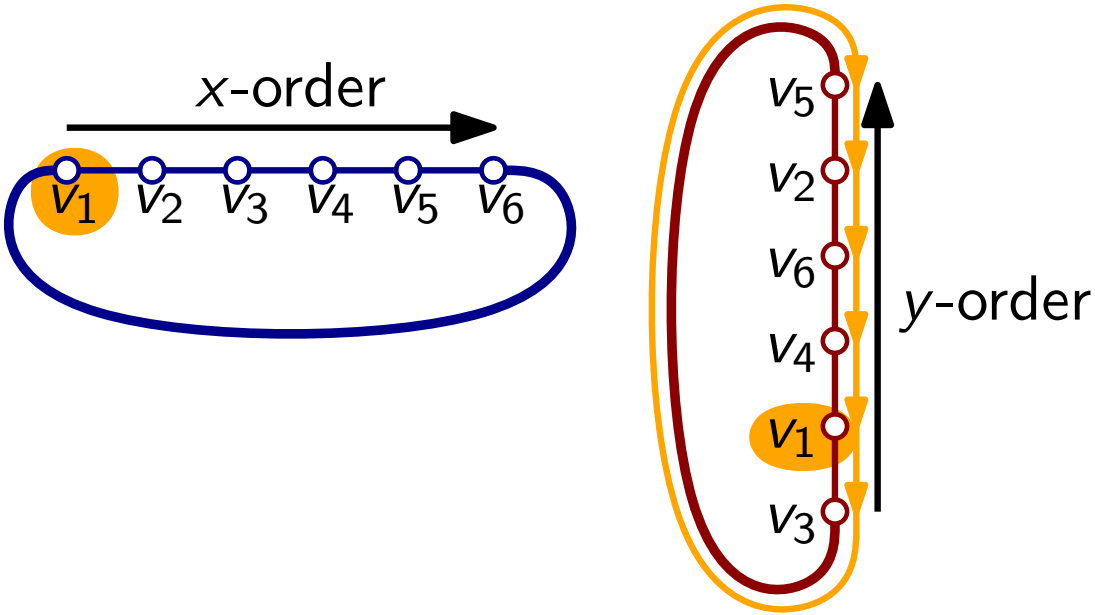


Edges:

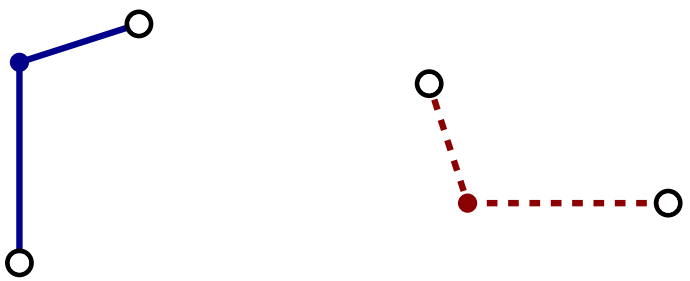


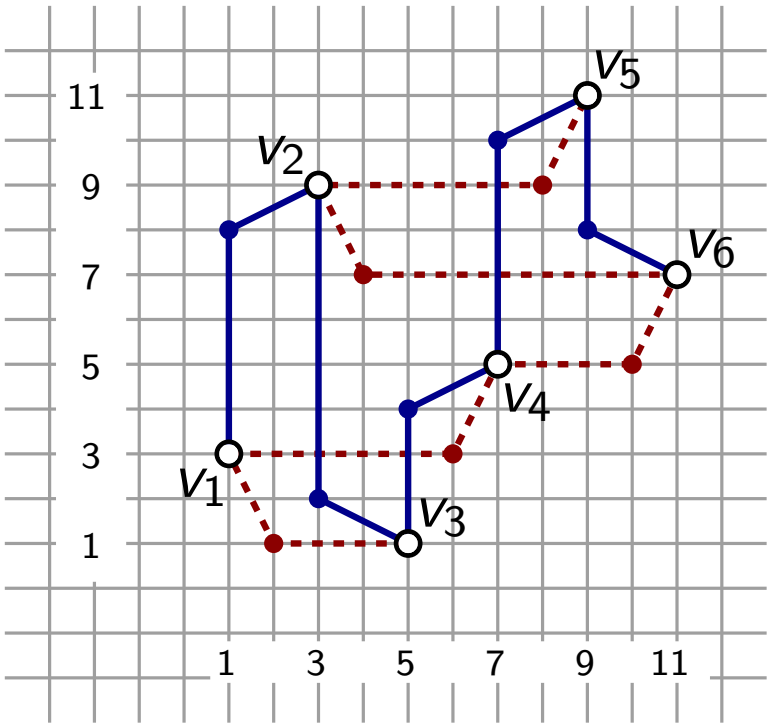


Cycle × Cycle

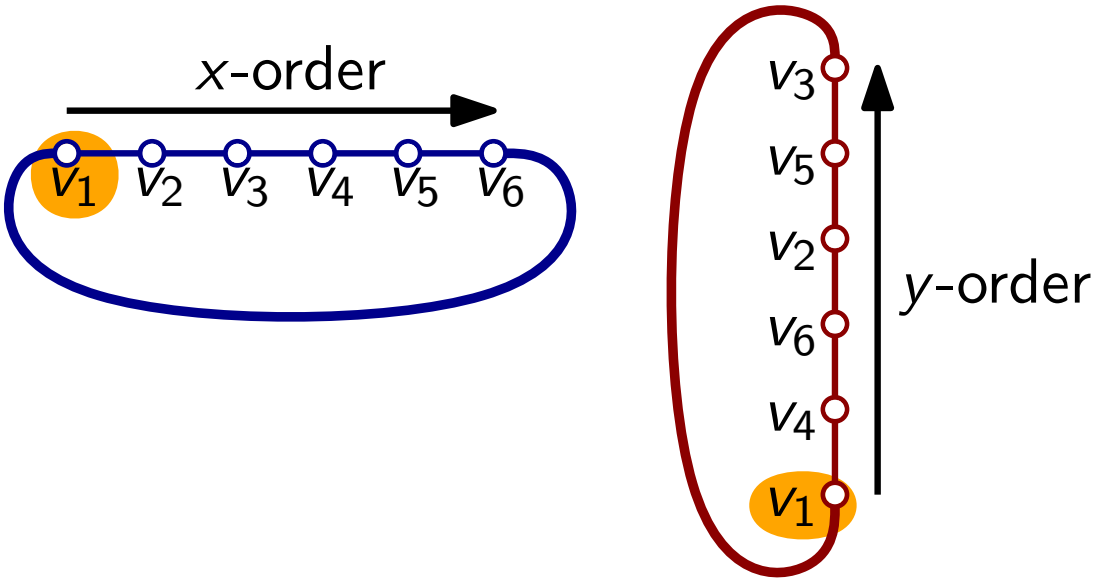


Edges:

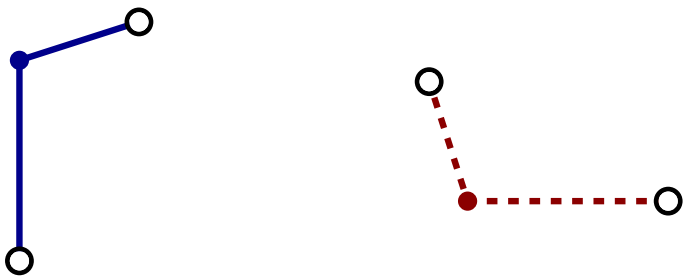


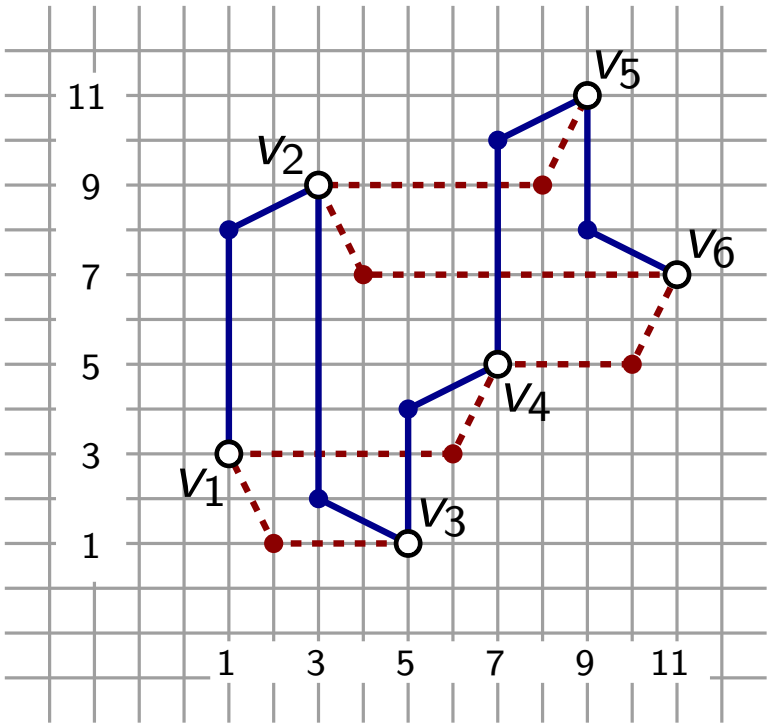


Cycle × Cycle

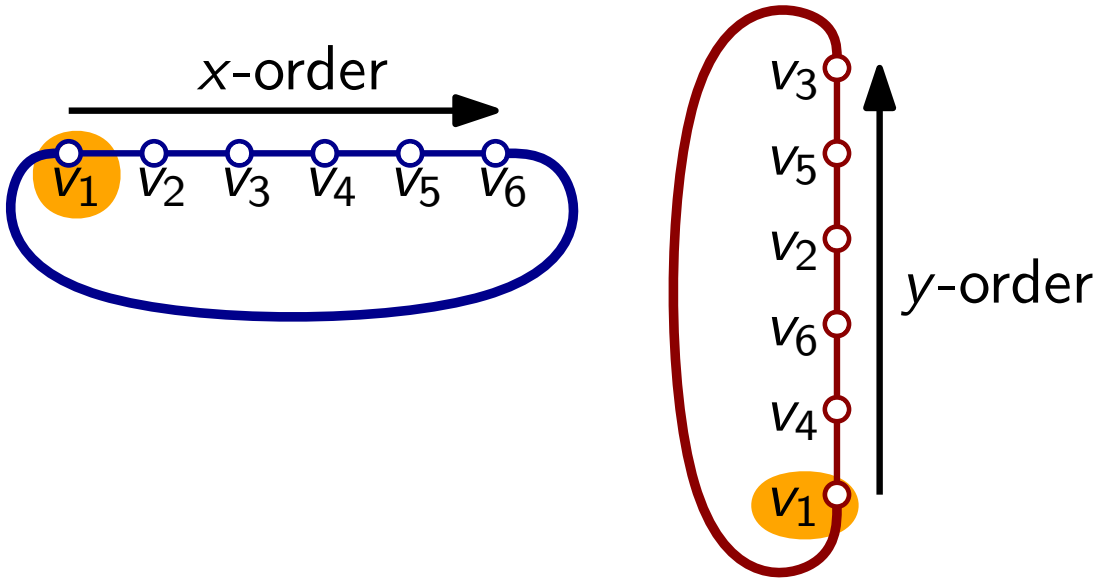


Edges:

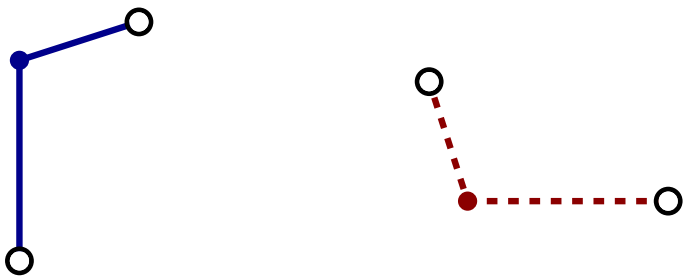


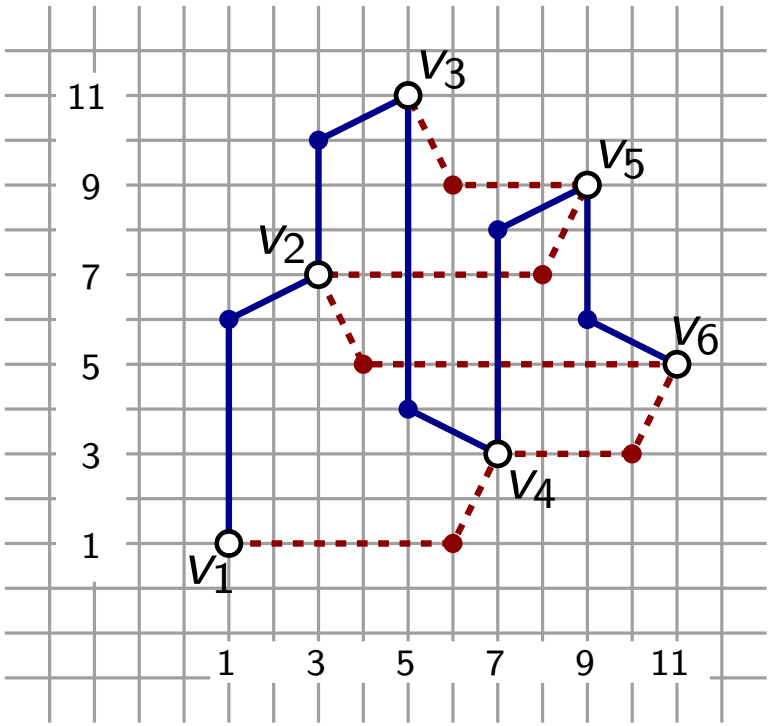


Cycle × Cycle

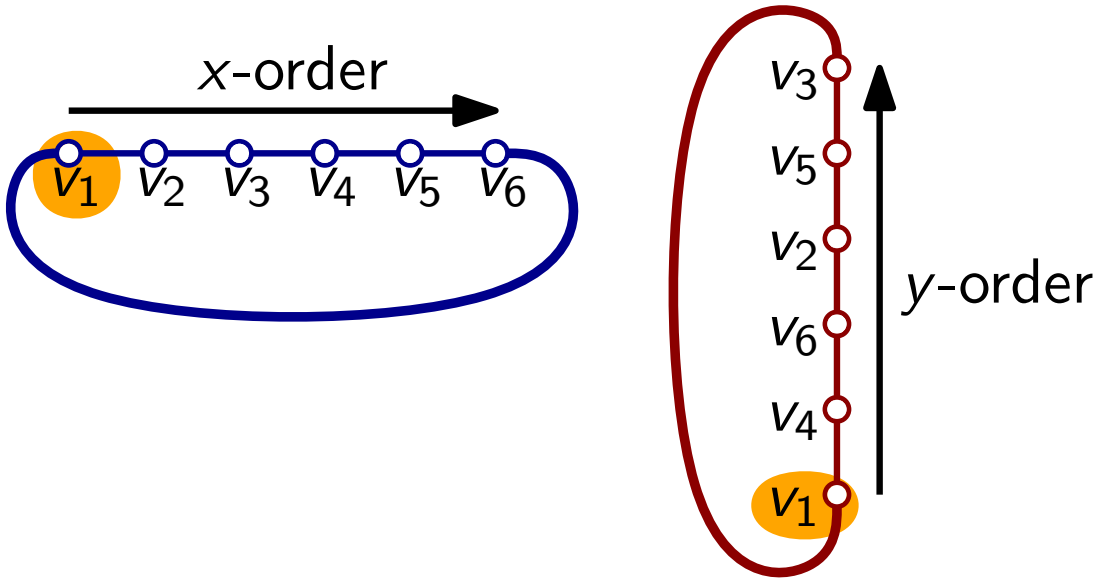


Edges:

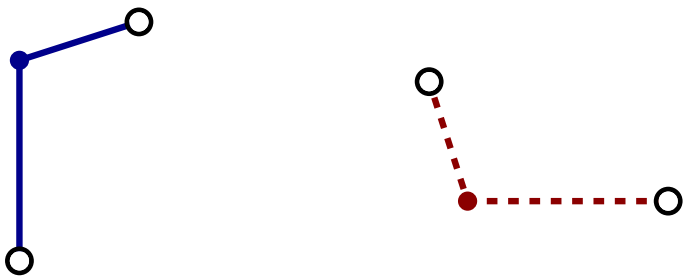


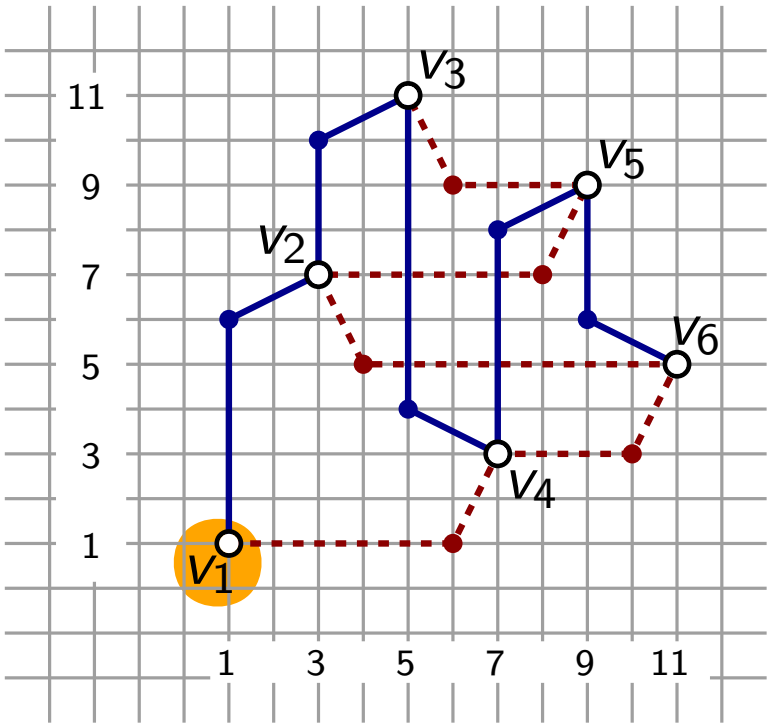


Cycle × Cycle

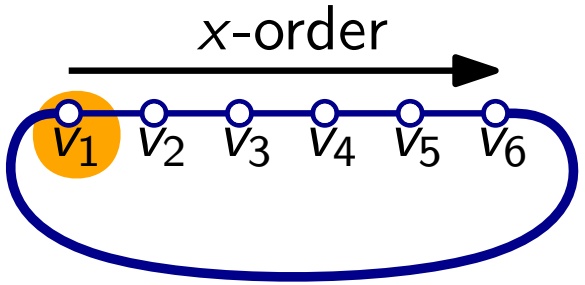


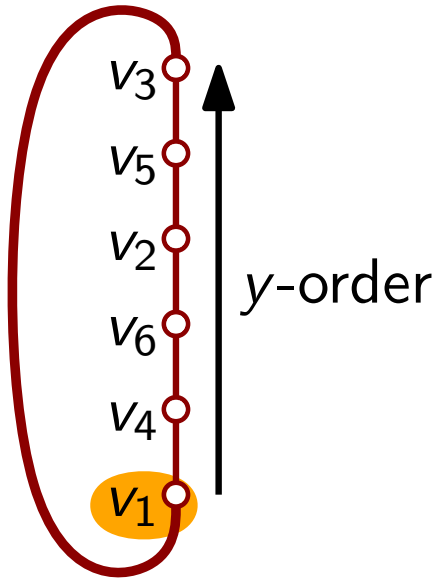
Edges:



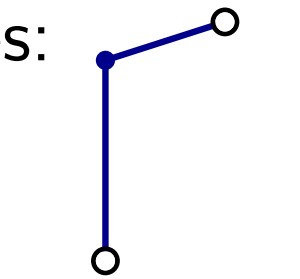


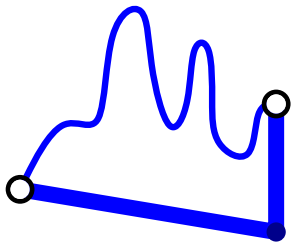
Cycle × Cycle

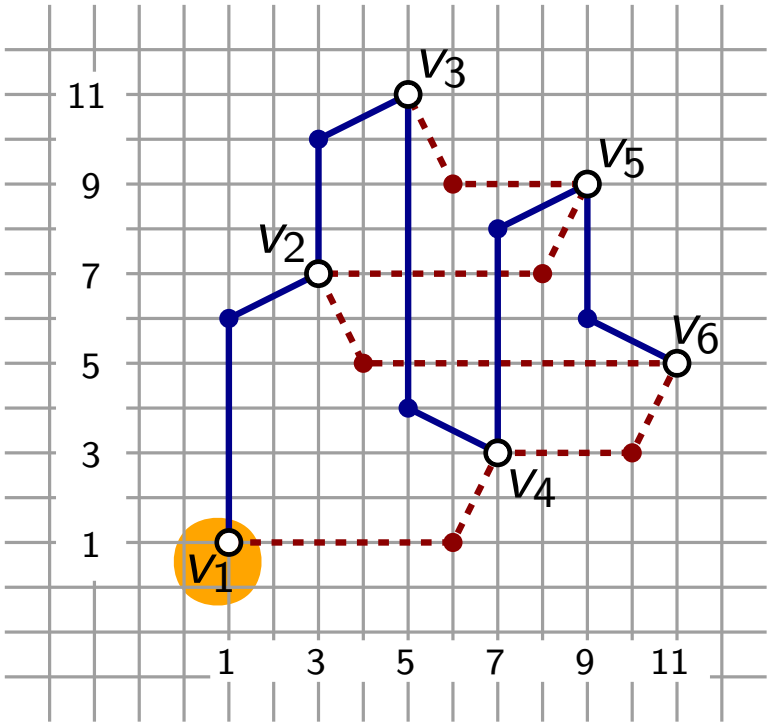




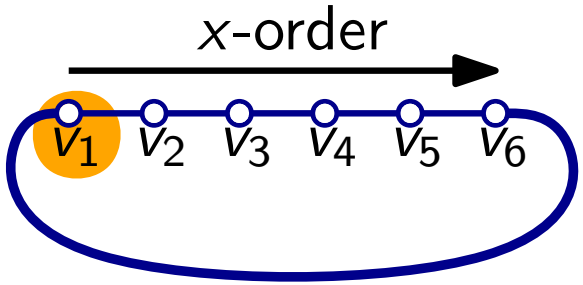
Edges:

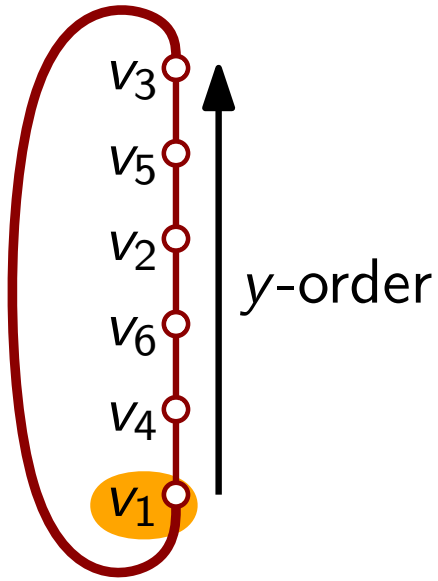




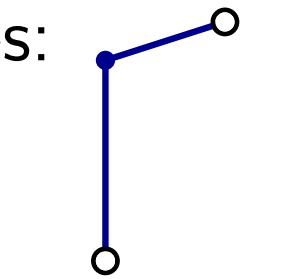


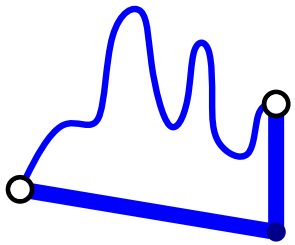
Cycle × Cycle

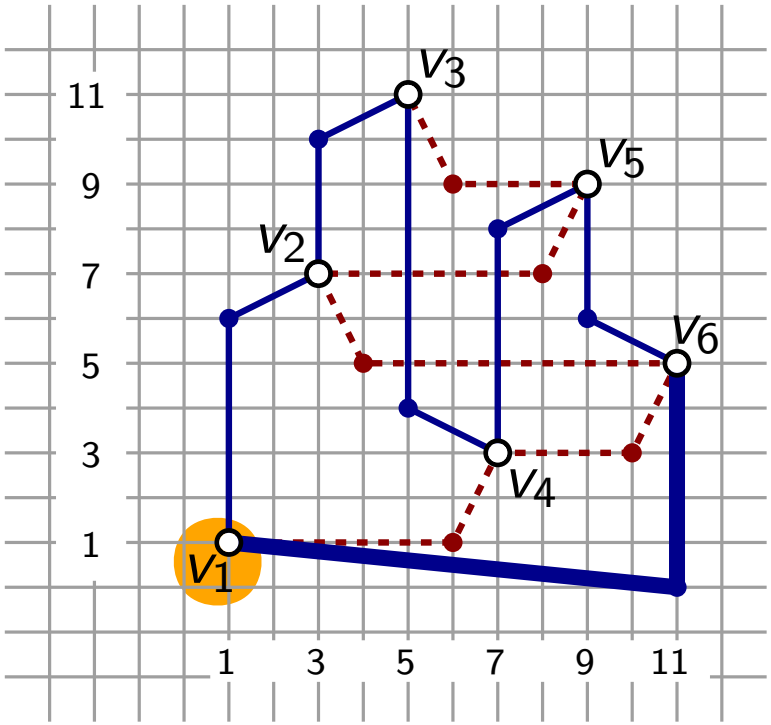




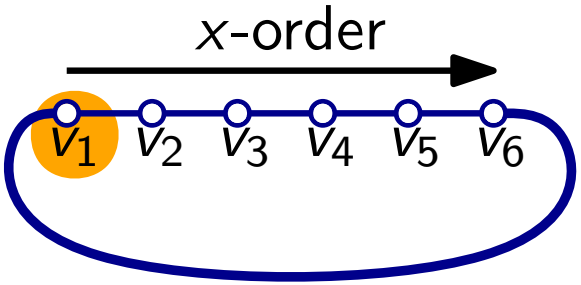
Edges:

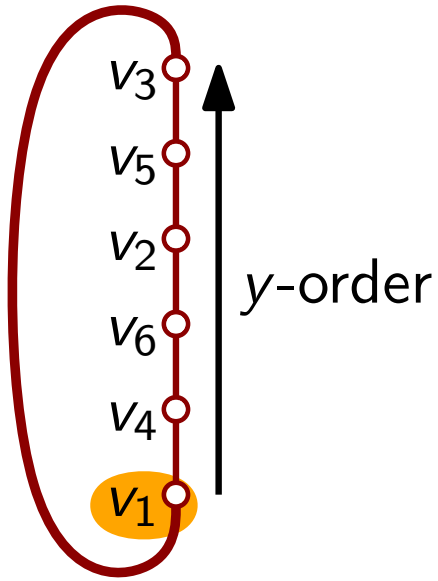




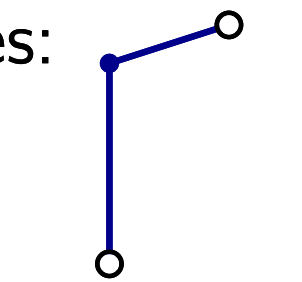


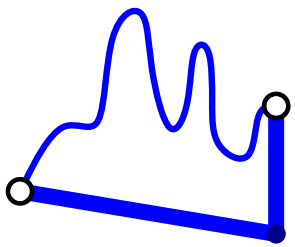
Cycle × Cycle

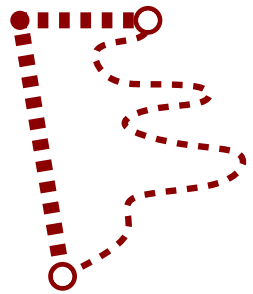


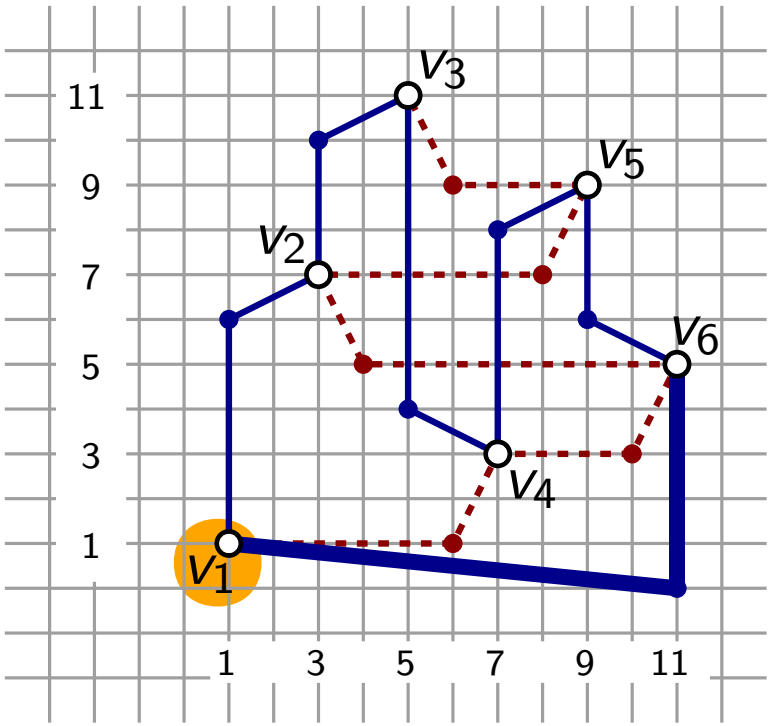


Edges:

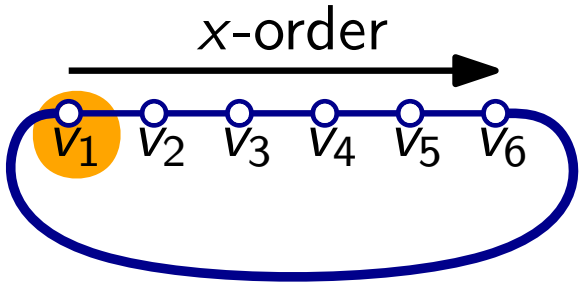


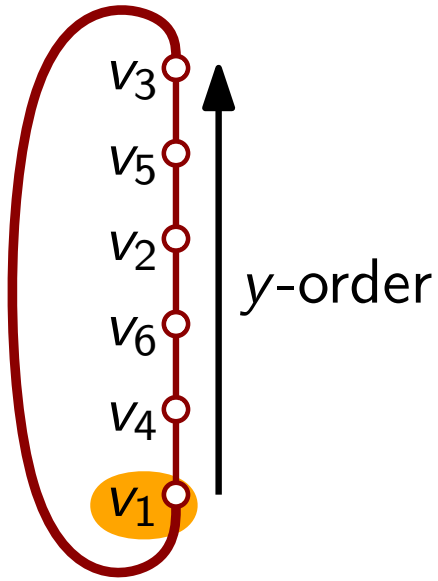




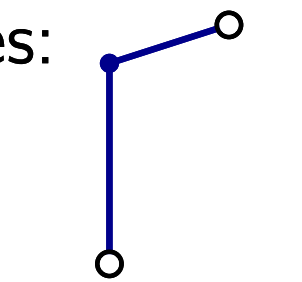


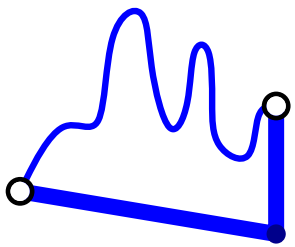
Cycle × Cycle

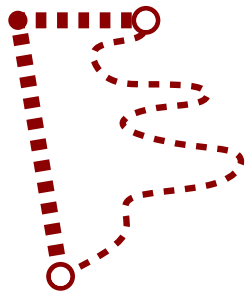


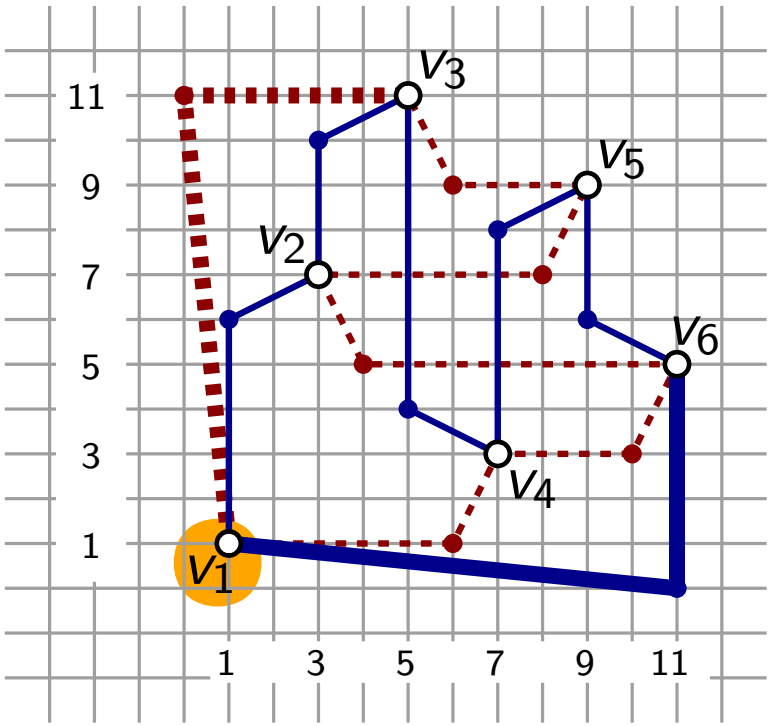


Edges:

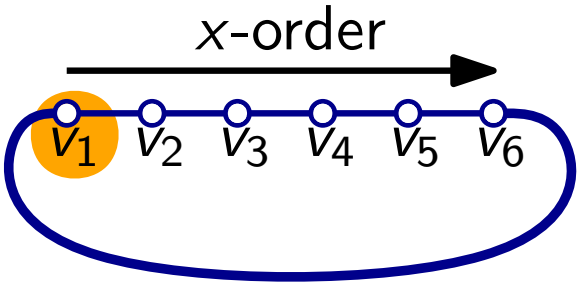


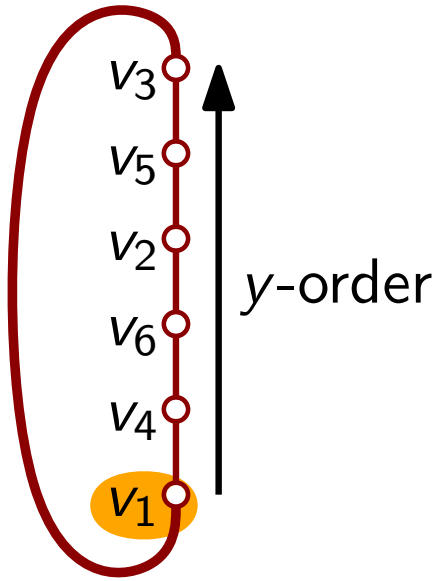






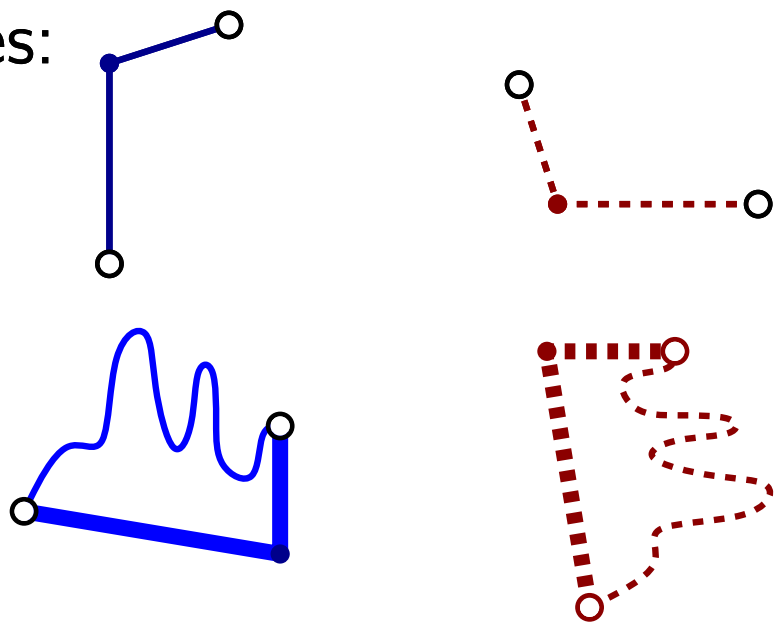
Cycle × Cycle

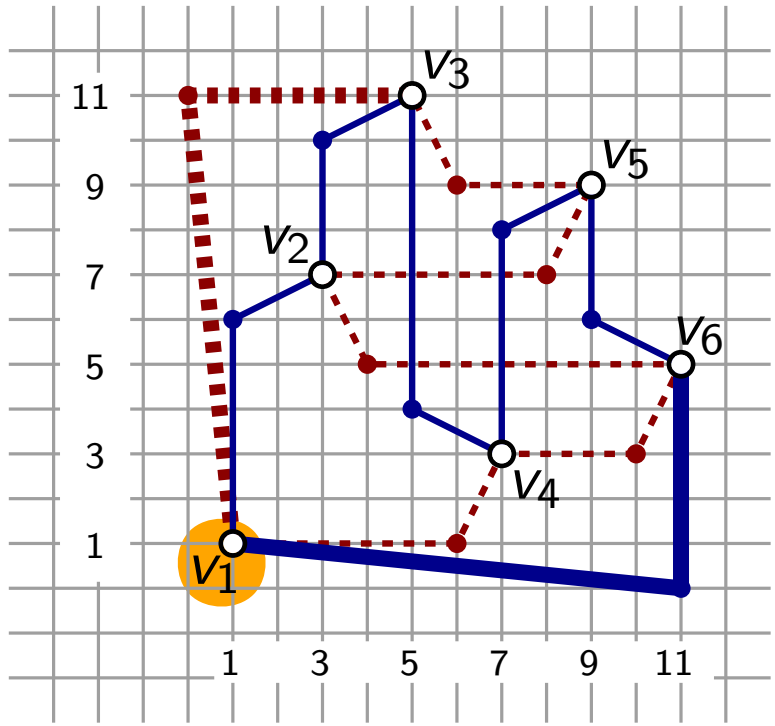




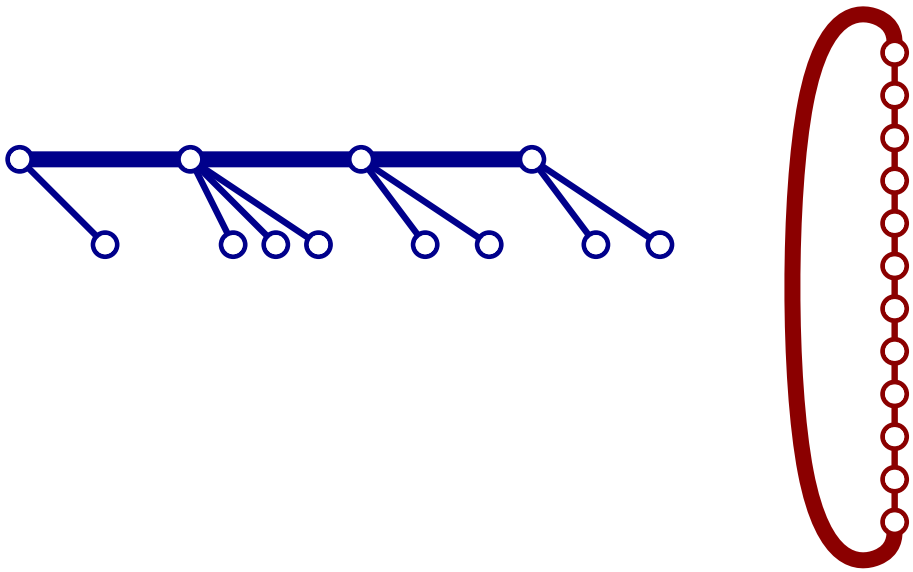
Bends: 1×1
Grid size:
$(2n-1) \times (2n-1)$

Edges:

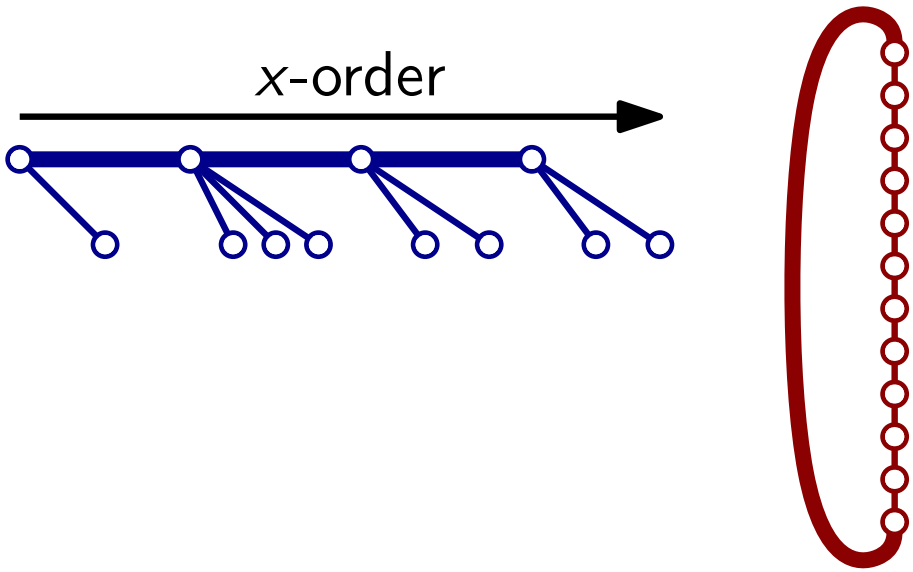




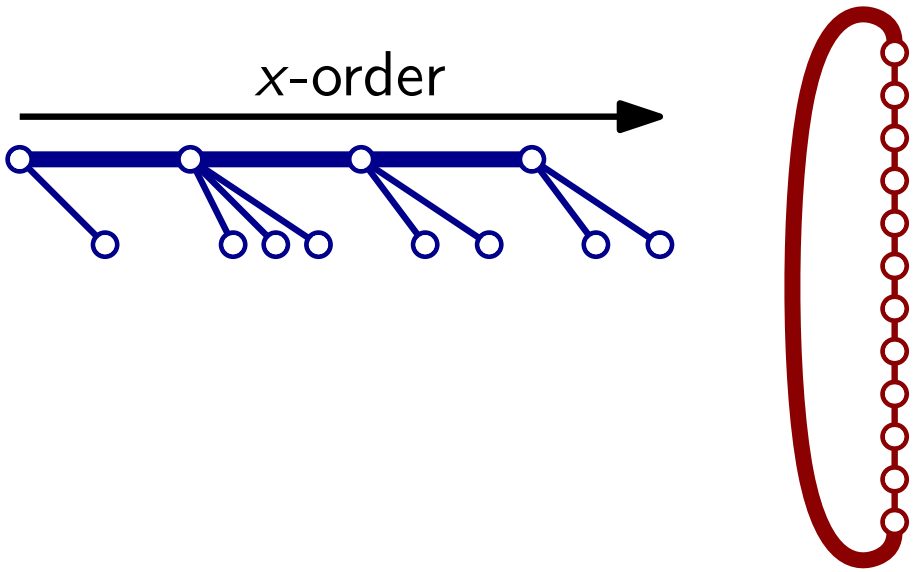
Caterpillar $\times$ Cycle

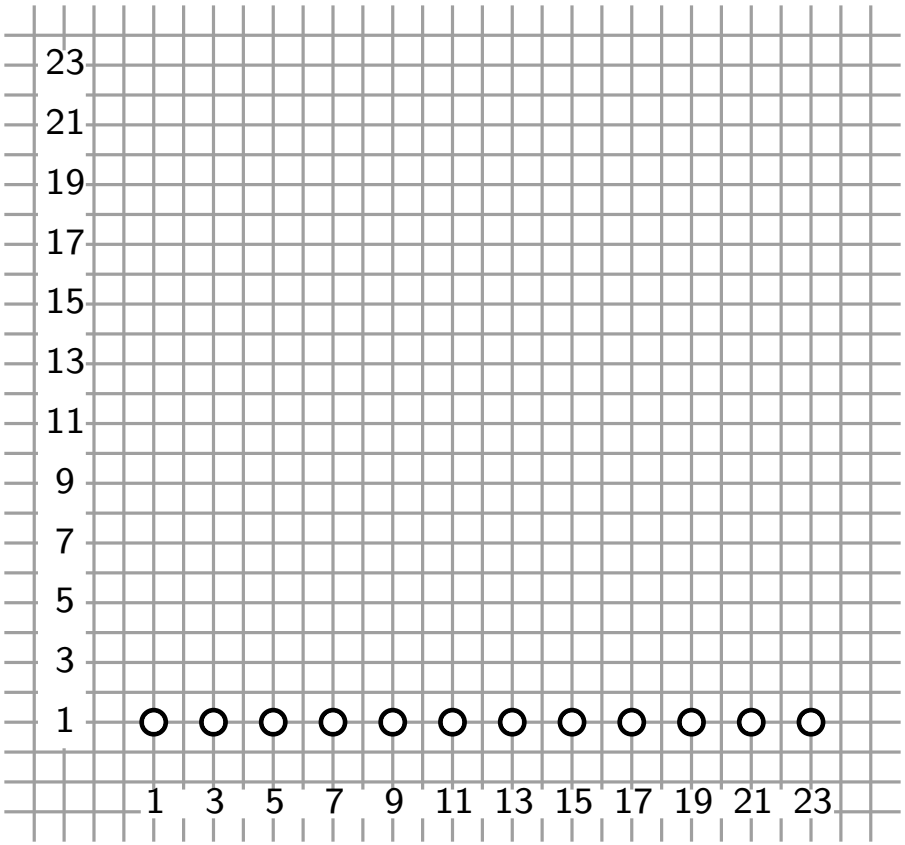


Caterpillar $\times$ Cycle

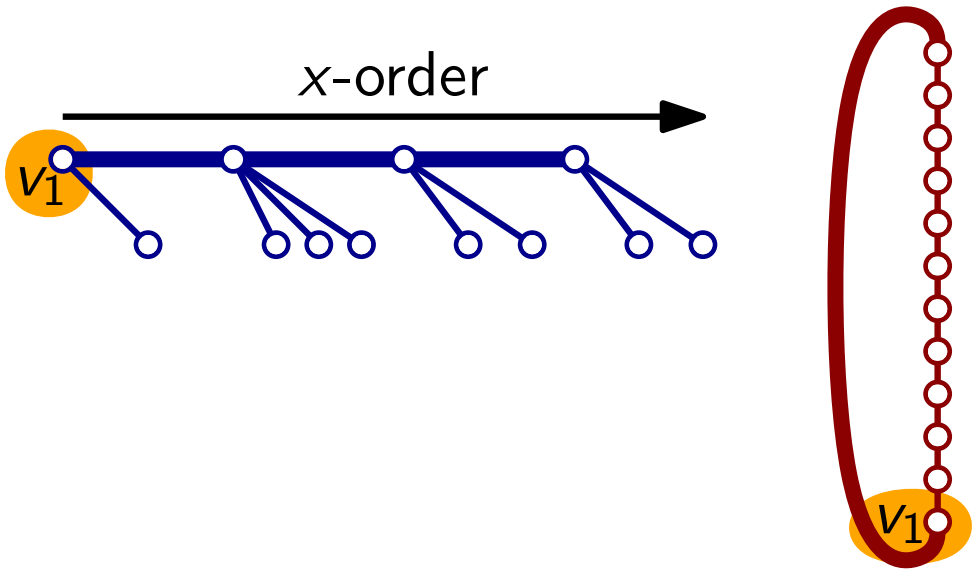


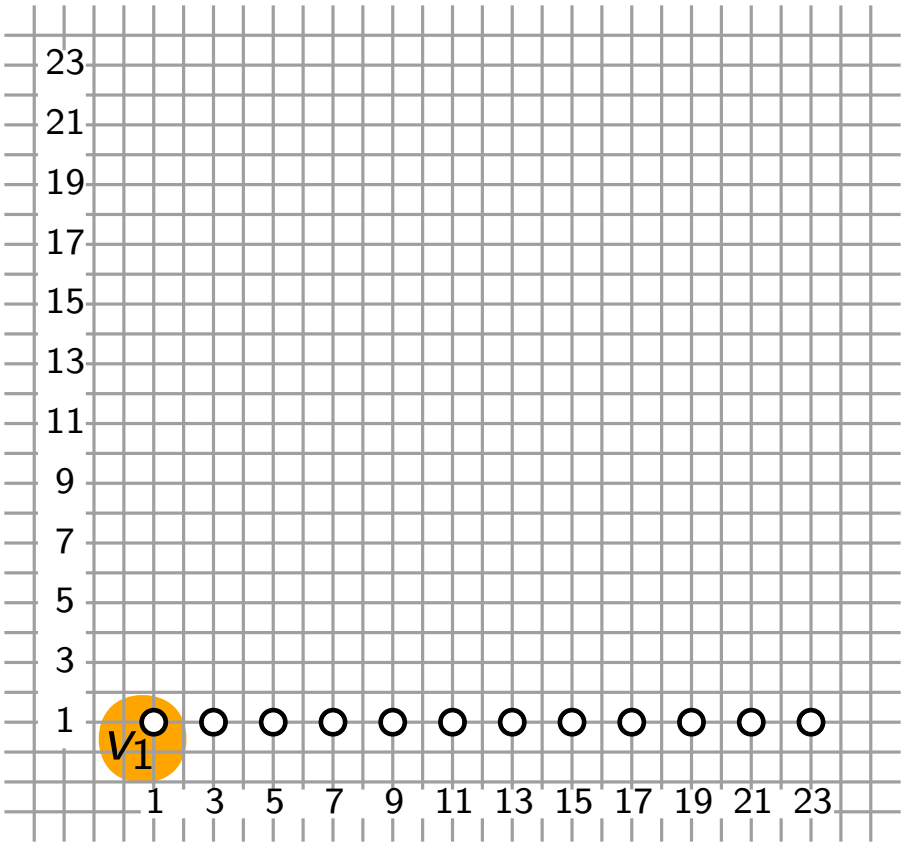
Caterpillar × Cycle



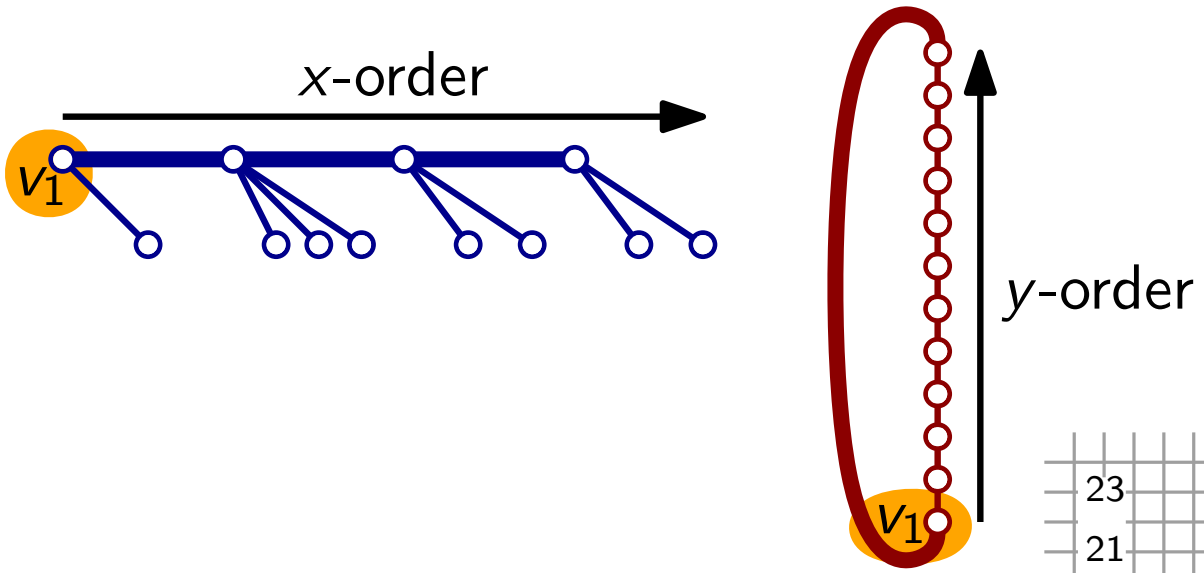


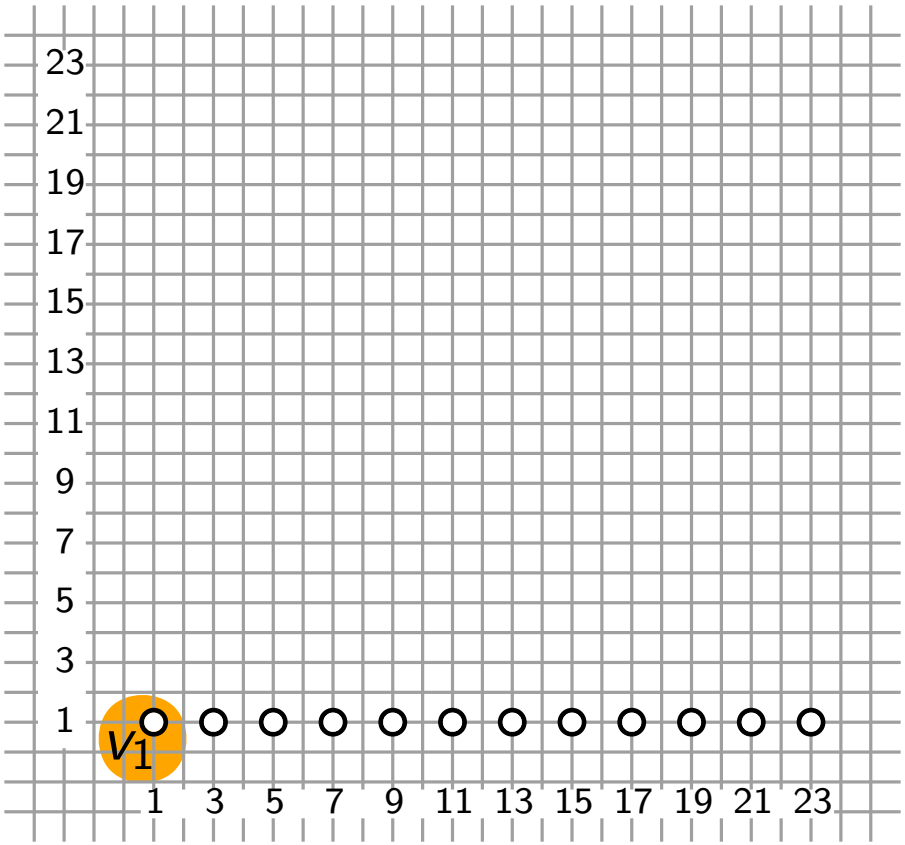
Caterpillar × Cycle



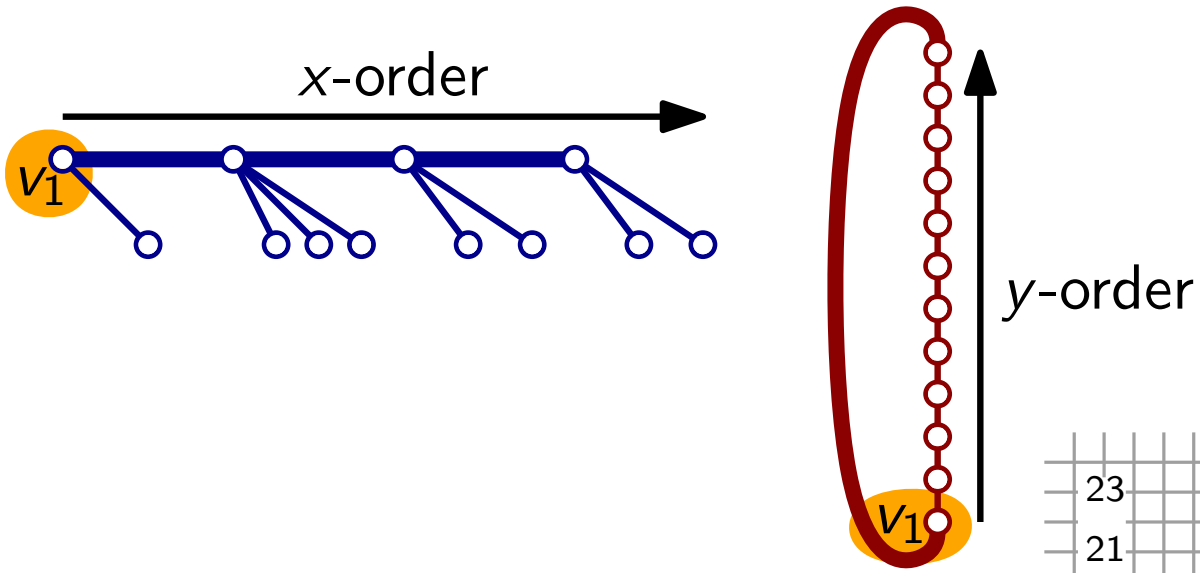


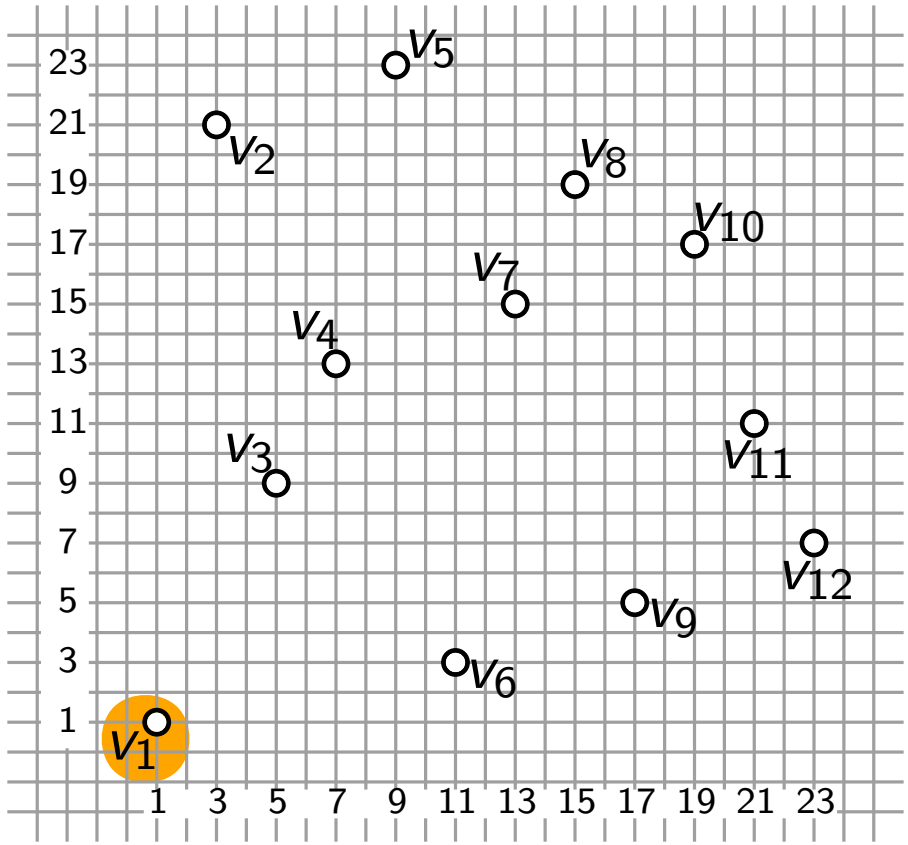
Caterpillar × Cycle



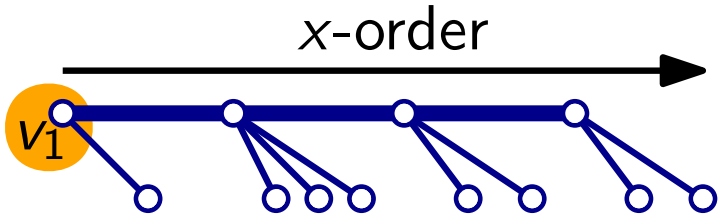


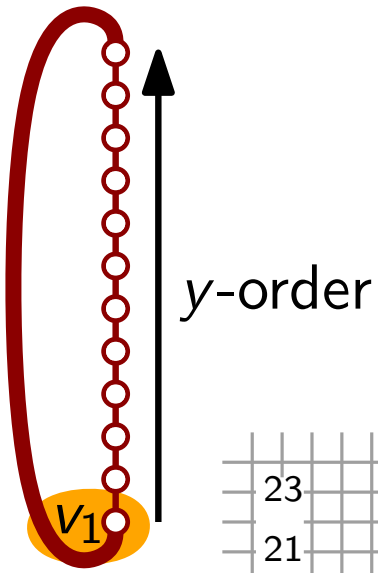
Caterpillar × Cycle



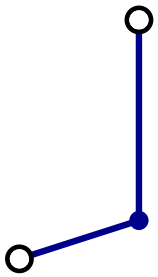


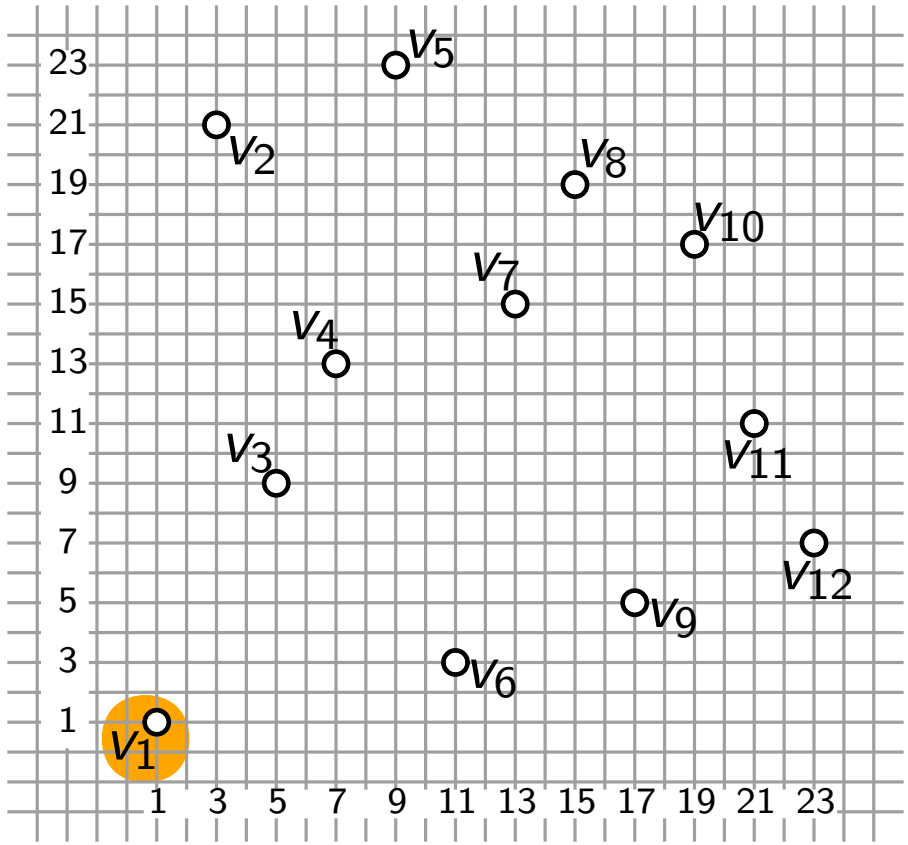
Caterpillar × Cycle



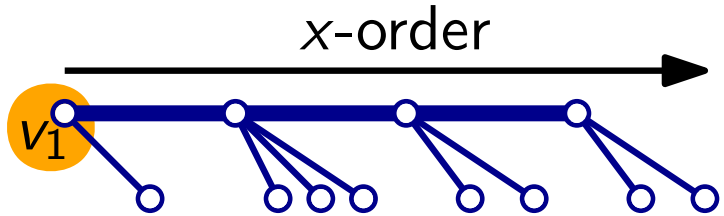


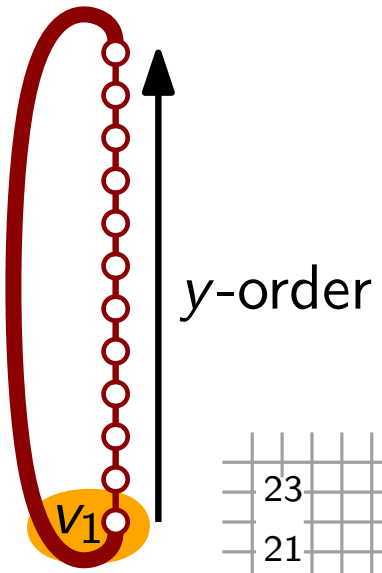
Edges:



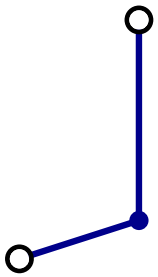


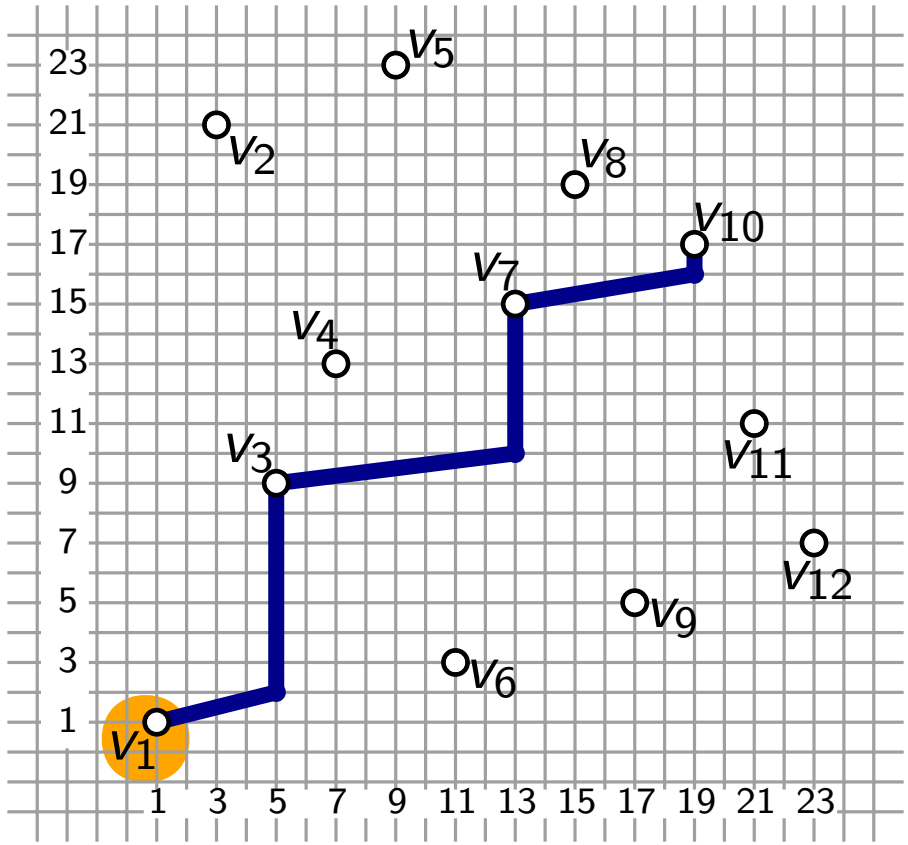
Caterpillar × Cycle



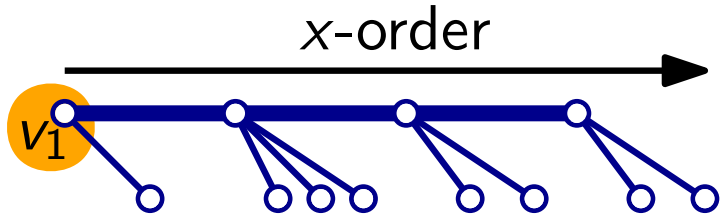


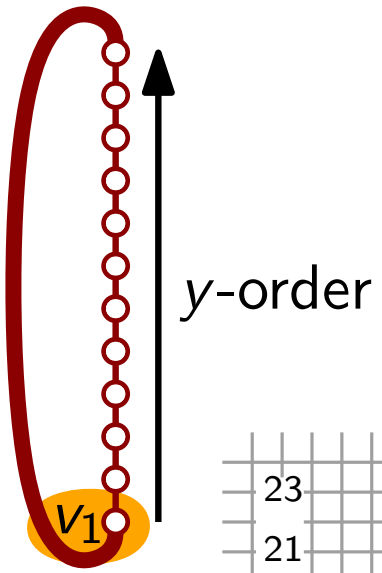
Edges:



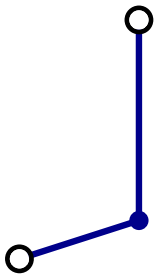


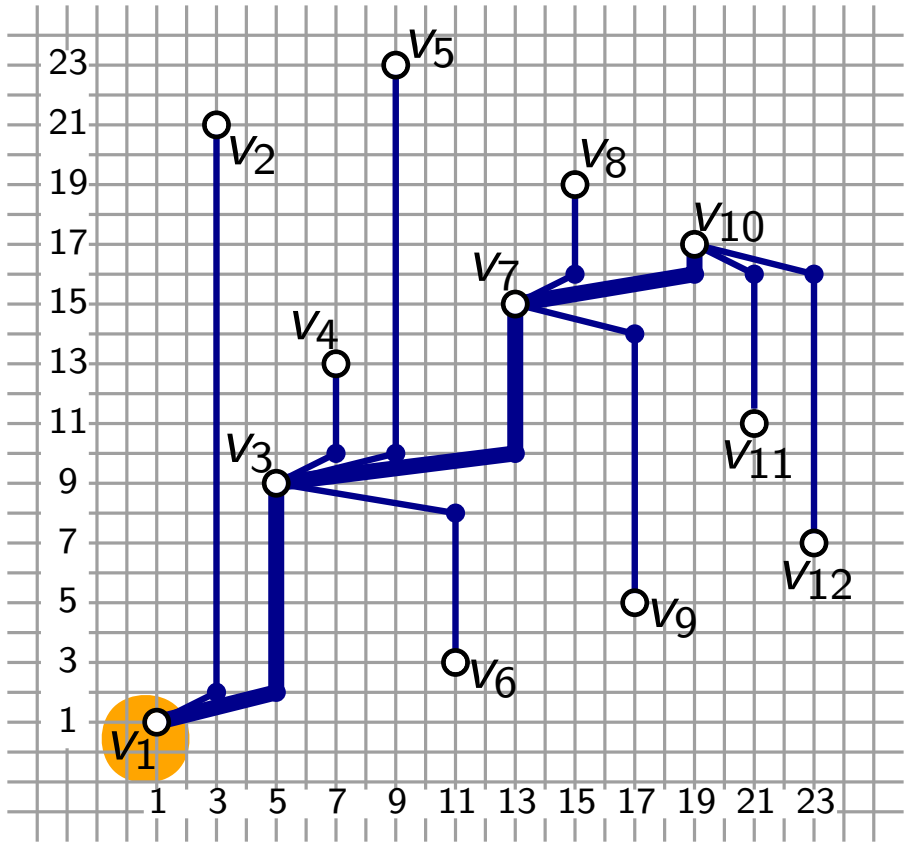
Caterpillar × Cycle



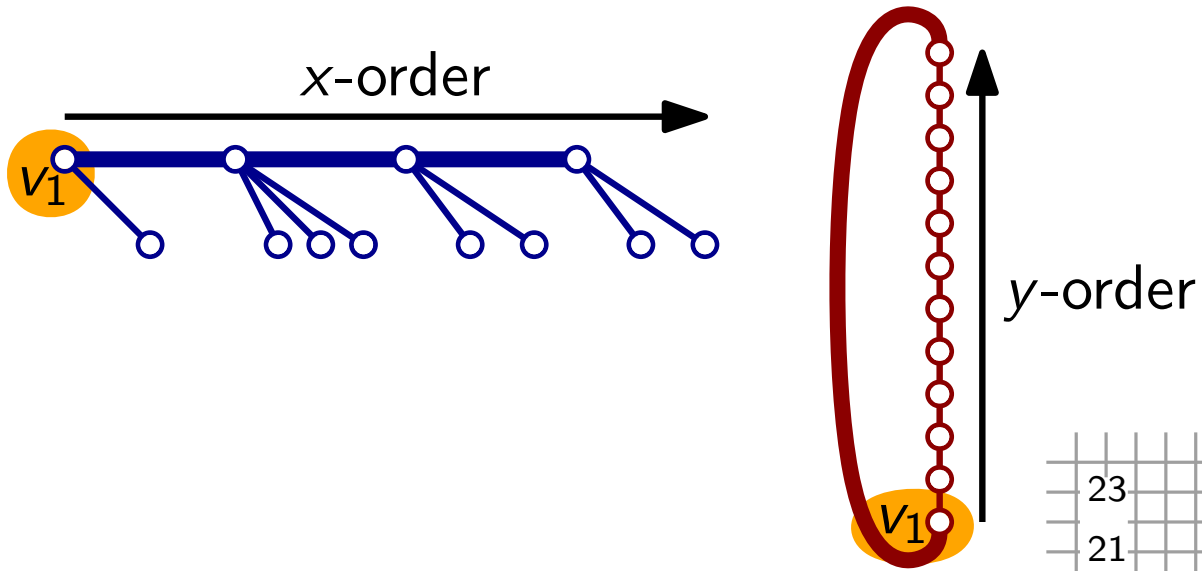


Edges:

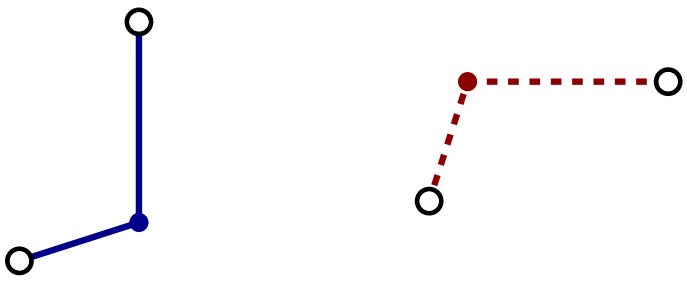


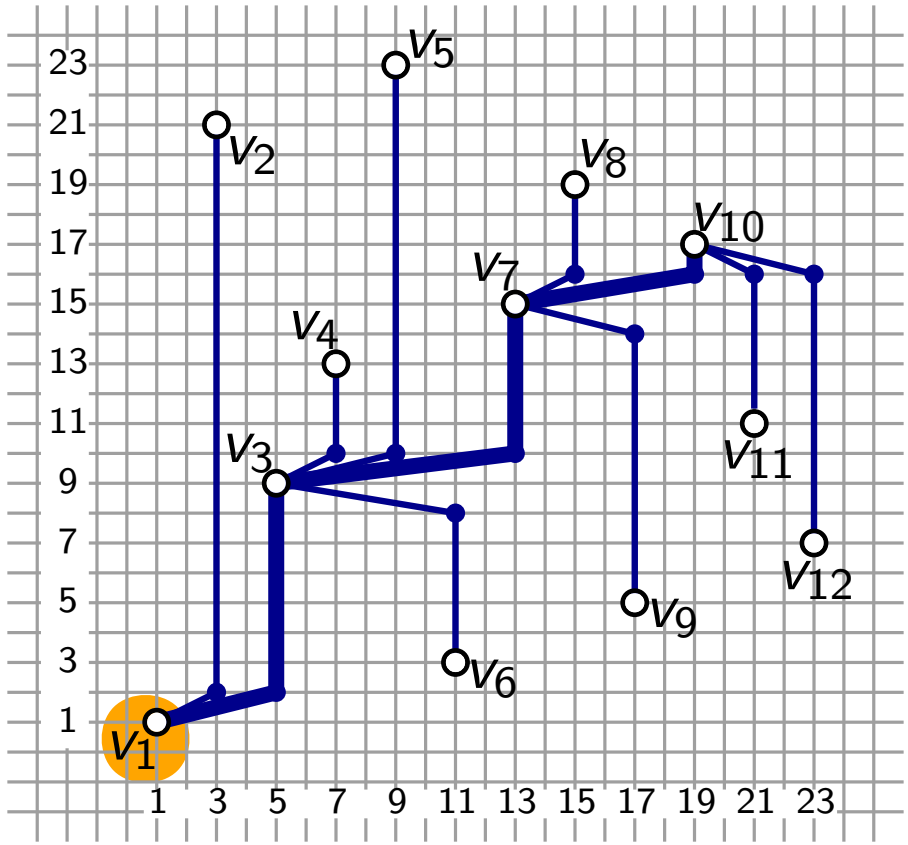


Caterpillar × Cycle

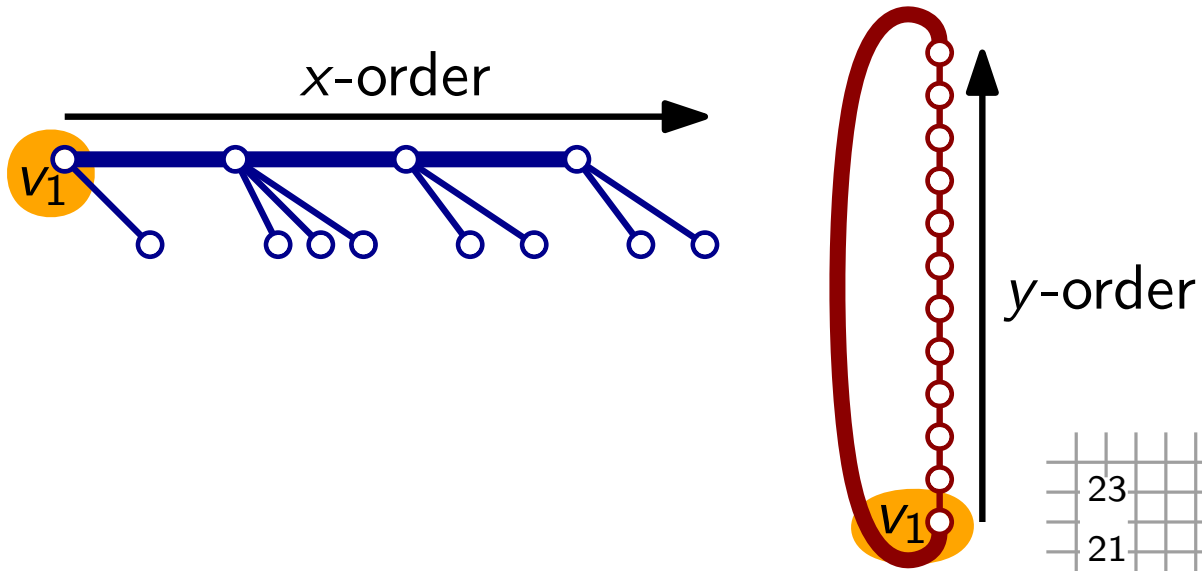


Edges:

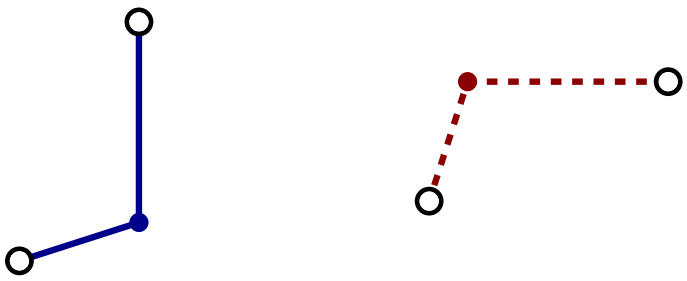


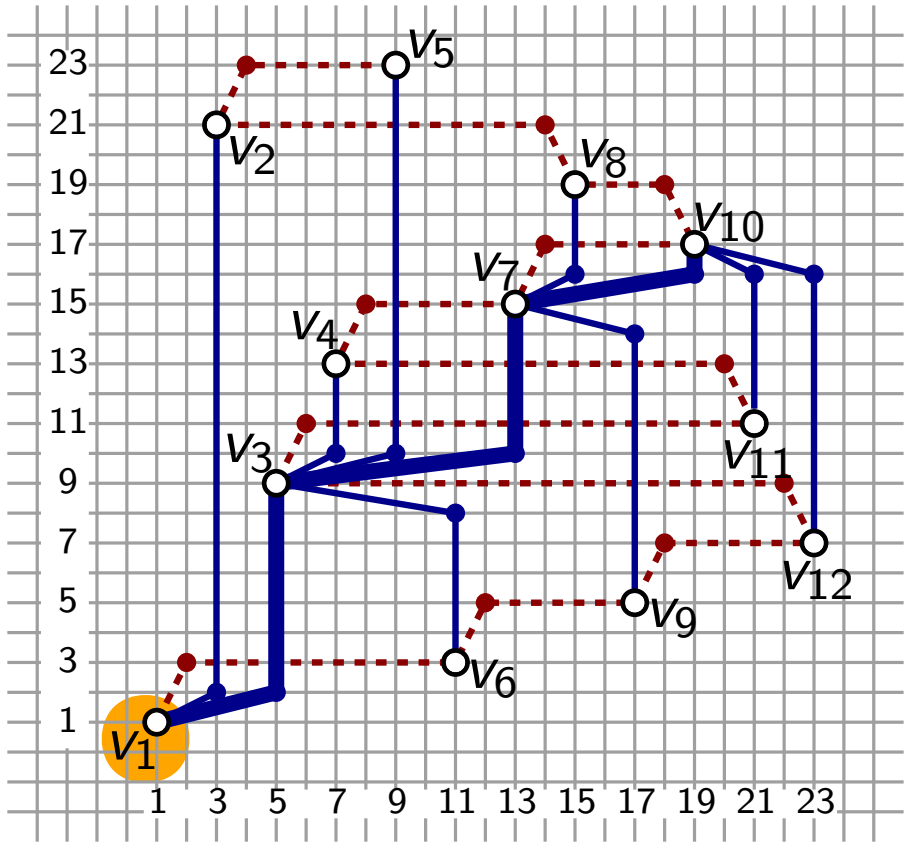


Caterpillar × Cycle

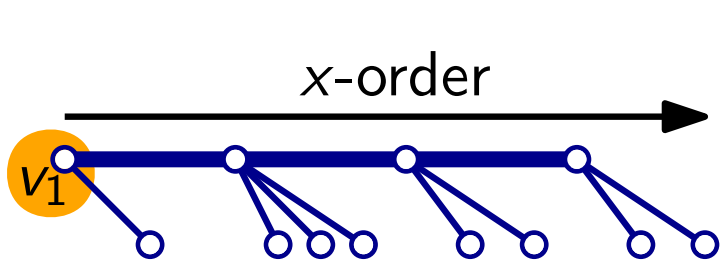


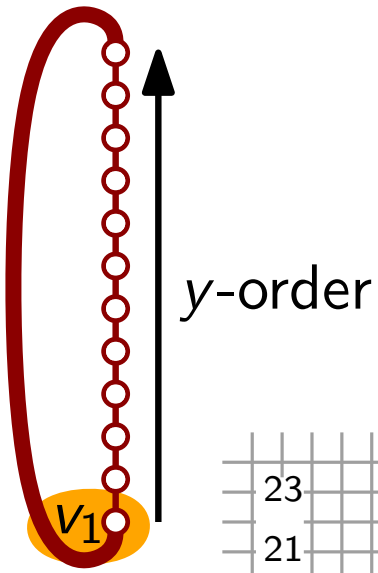
Edges:



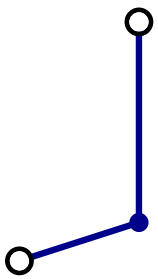


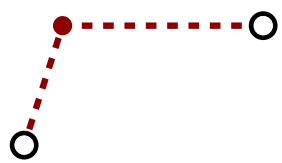
Caterpillar × Cycle

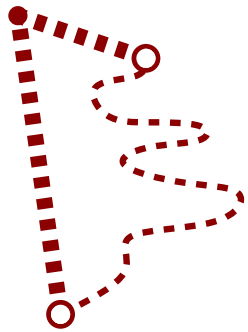


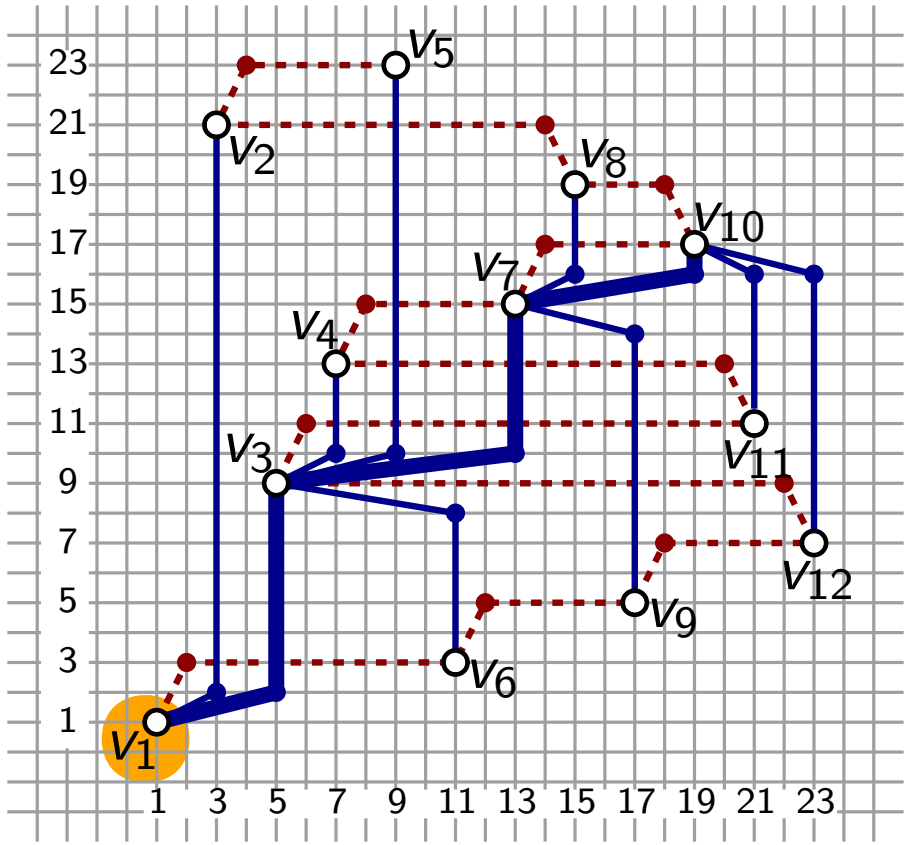


Edges:

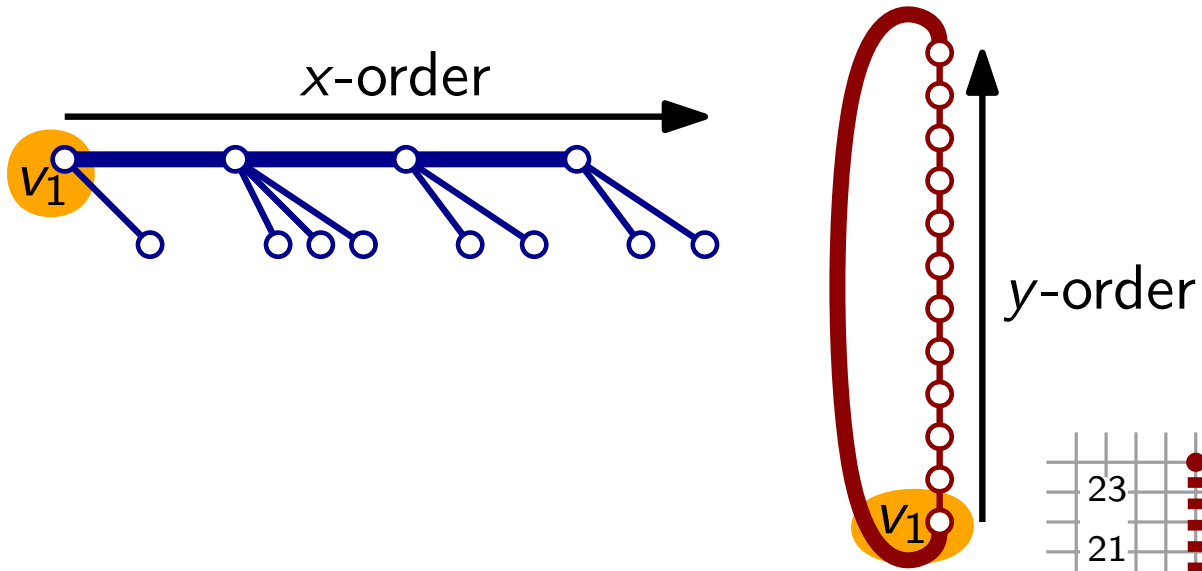




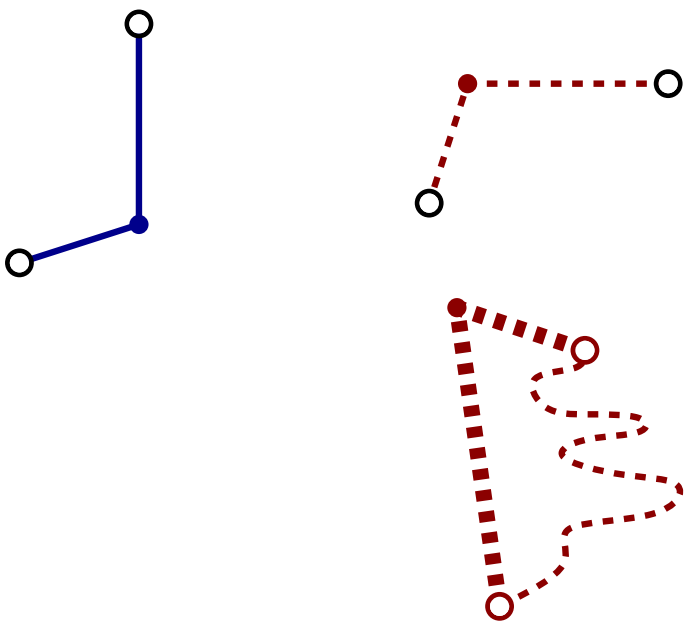


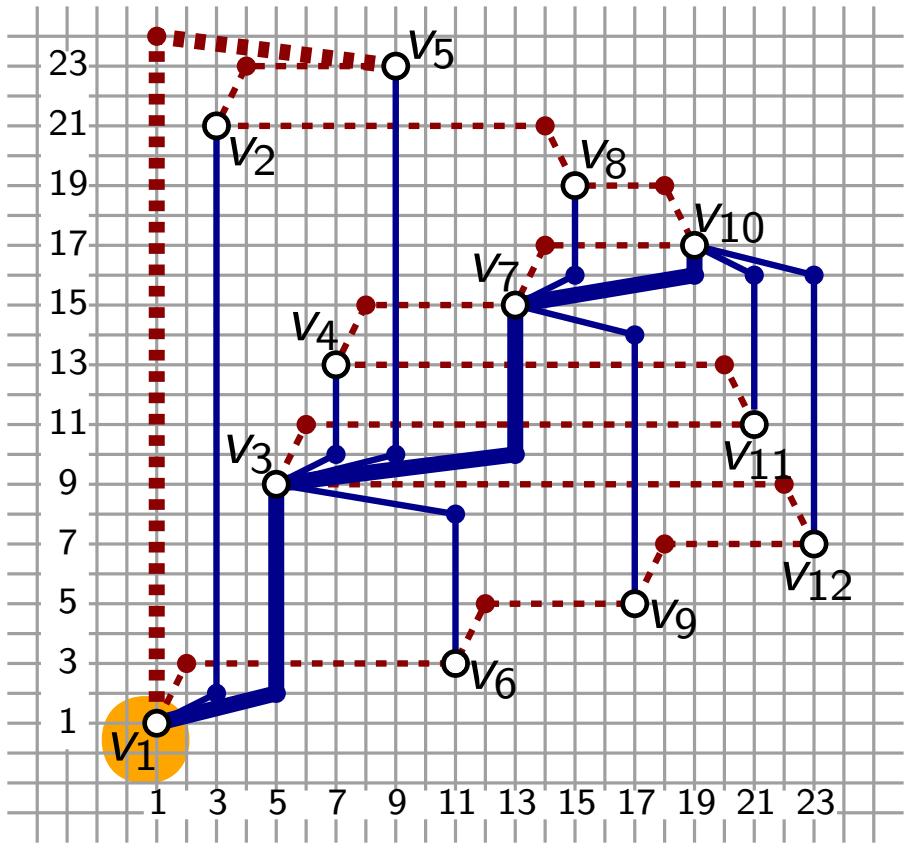


Caterpillar × Cycle

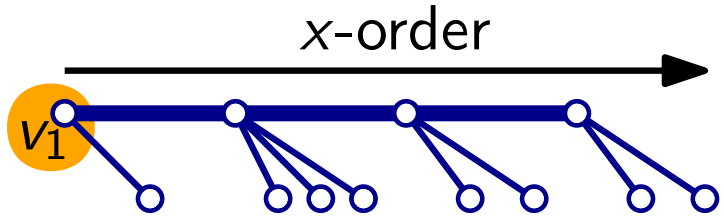


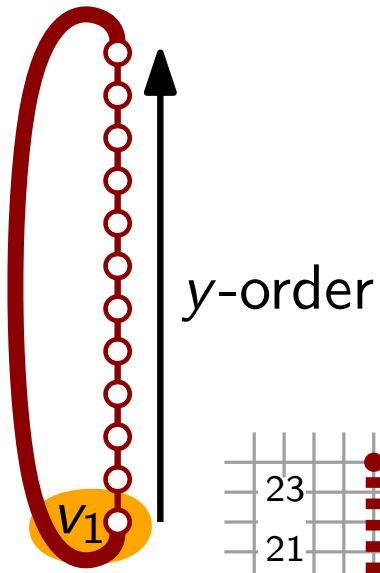
Edges:





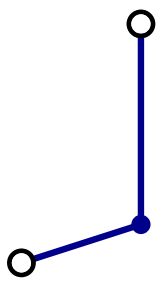
Caterpillar × Cycle

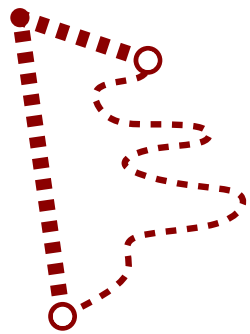


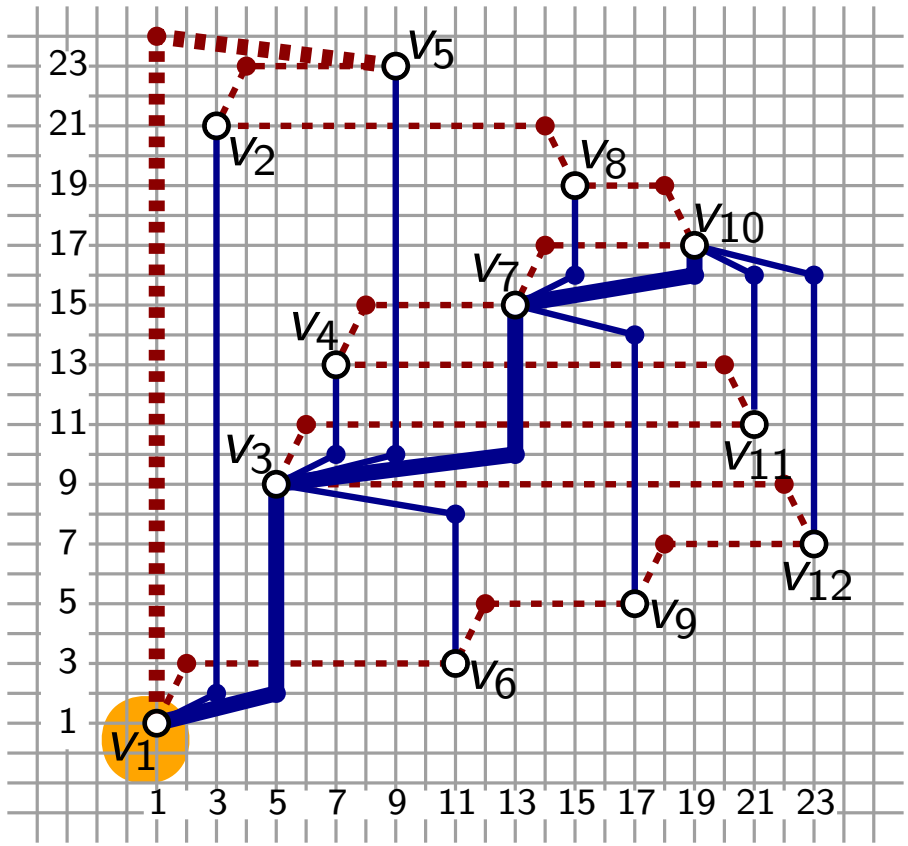


Bends: 1×1
Grid size:
$(2n-1) \times 2n$

Edges:



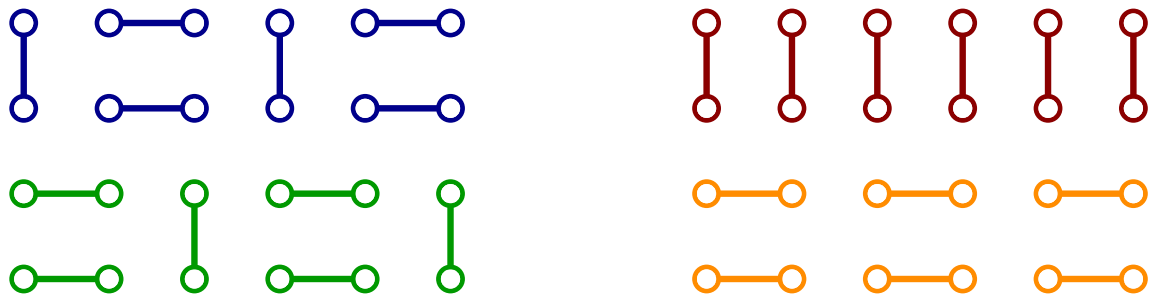




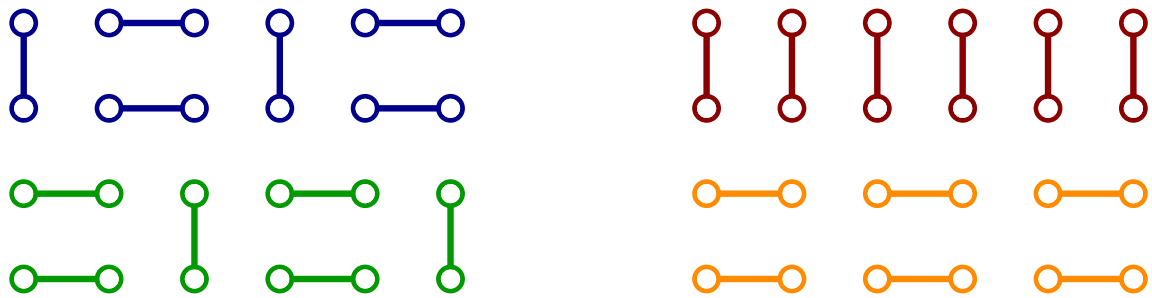
Overview

Graph classes			Number of bends
Cycle	$\times$	Cycle	1×1
Caterpillar	$\times$	Cycle	1×1
Four Matchings			$1 \times 1 \times 1 \times 1$
Tree	$\times$	Matching	1×0
Wheel	$\times$	Matching	2×0
Outerpath	$\times$	Matching	2×1
Outerplanar	$\times$	Outerplanar	3×3
2-page book emb.	$\times$	2-page book emb.	4×4
Planar	$\times$	Planar	6×6

Four Matchings

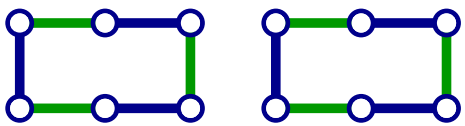


Four Matchings

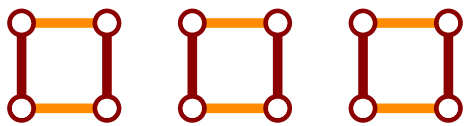


Combine $\Rightarrow$ two sets of cycles $\mathcal{C}_1$, $\mathcal{C}_2$

Four Matchings



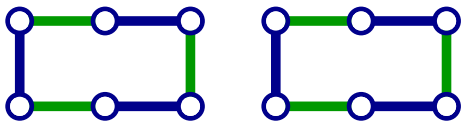
$\mathcal{C}_1$



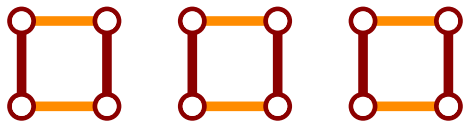
$\mathcal{C}_2$

Combine $\Rightarrow$ two sets of cycles $\mathcal{C}_1$, $\mathcal{C}_2$

Four Matchings

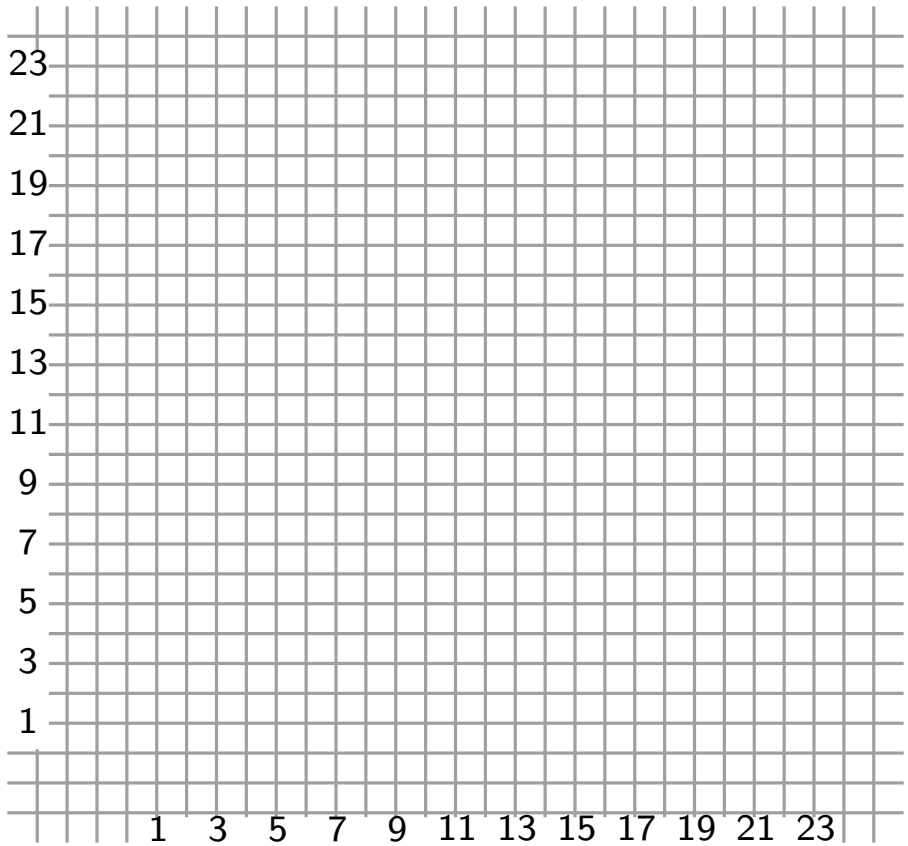


$\mathcal{C}_1$

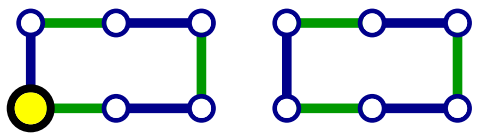


$\mathcal{C}_2$

Placement algorithm:



Four Matchings

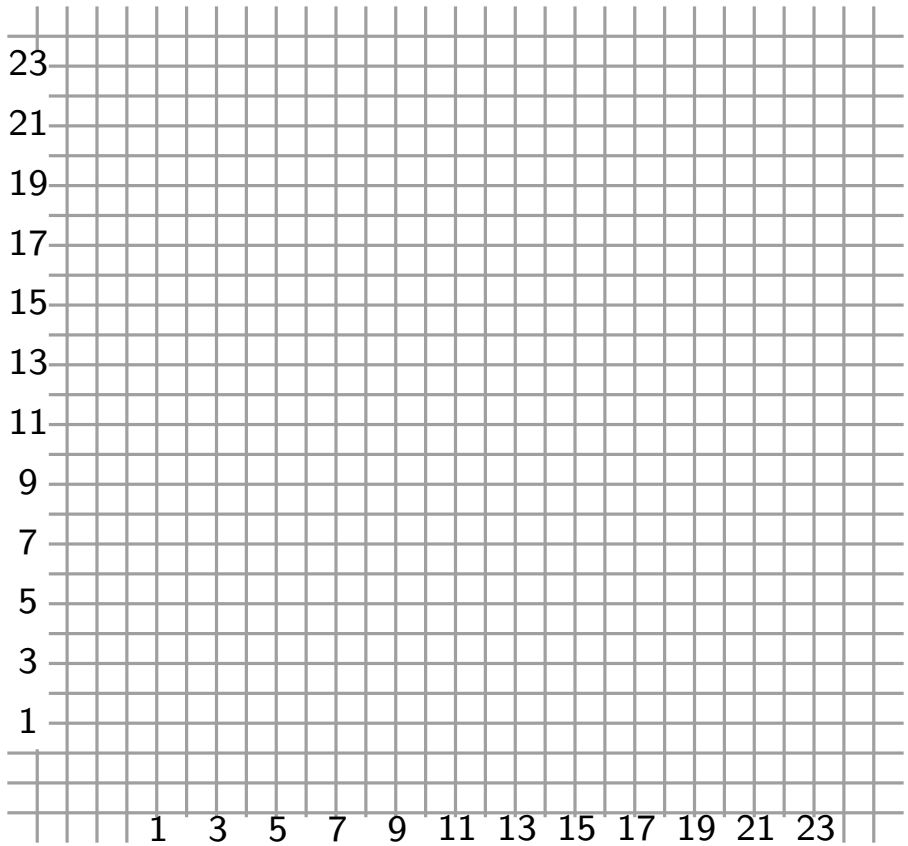


$\mathcal{C}_1$

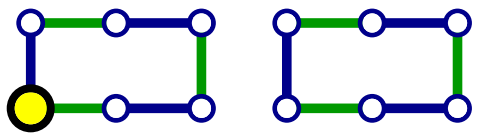
$\mathcal{C}_2$

Placement algorithm:

- Pick v_1



Four Matchings

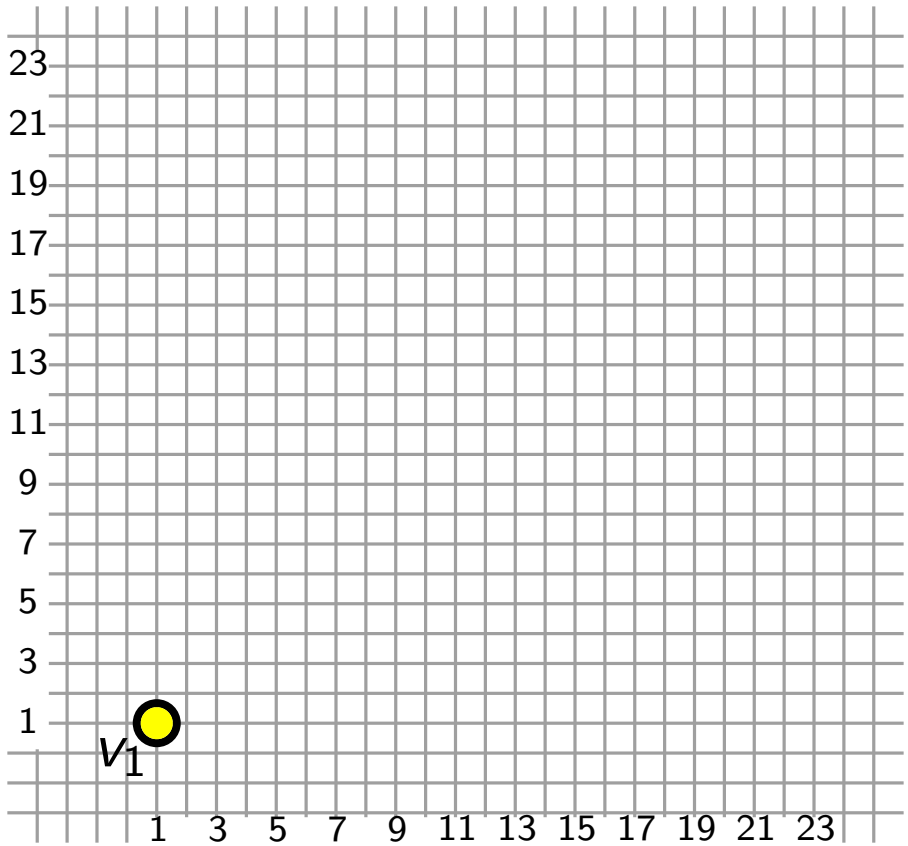


$\mathcal{C}_1$

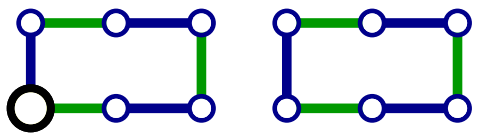
$\mathcal{C}_2$

Placement algorithm:

- Pick v_1 , place it



Four Matchings

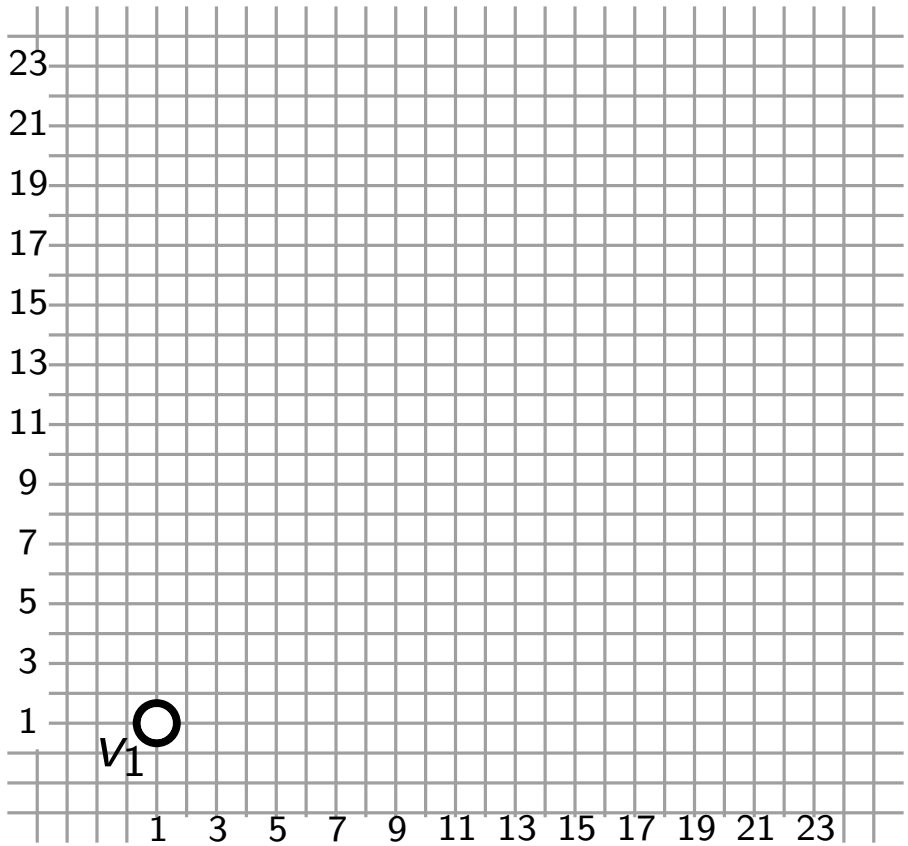


$\mathcal{C}_1$

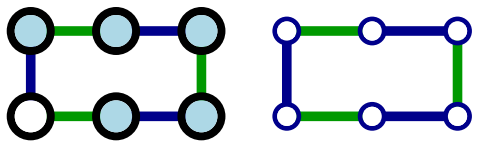
$\mathcal{C}_2$

Placement algorithm:

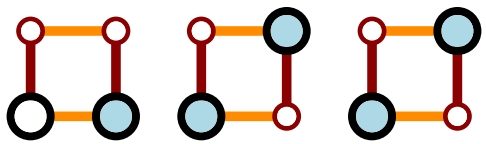
- Pick v_1 , place it



Four Matchings



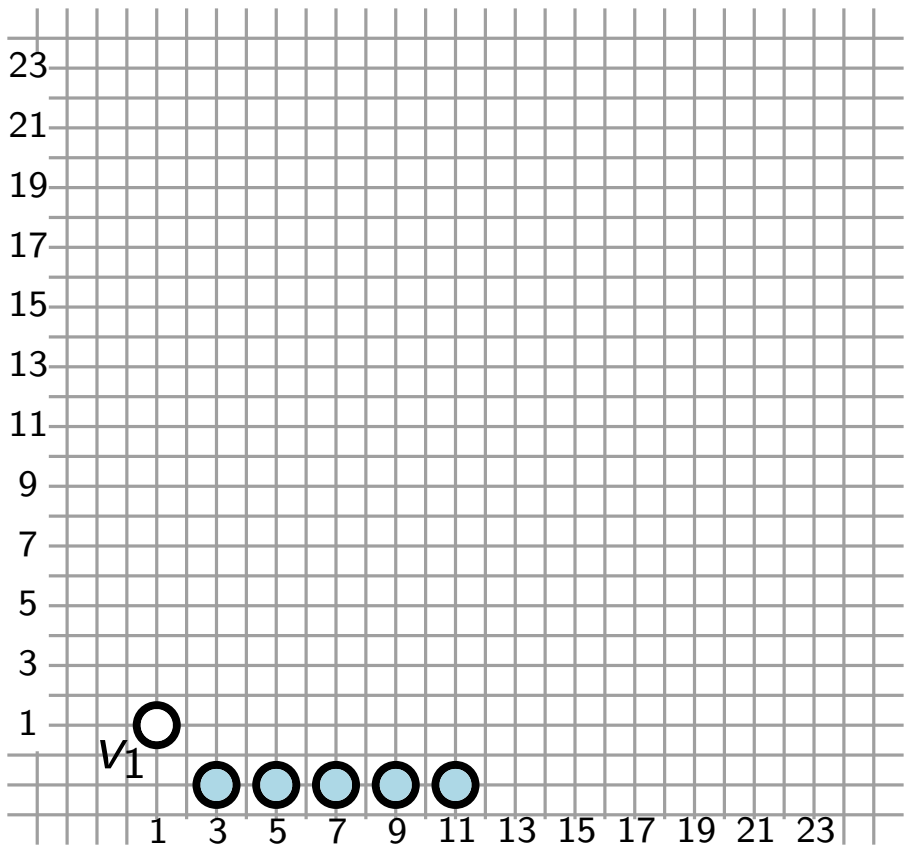
$\mathcal{C}_1$



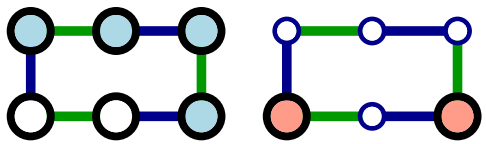
$\mathcal{C}_2$

Placement algorithm:

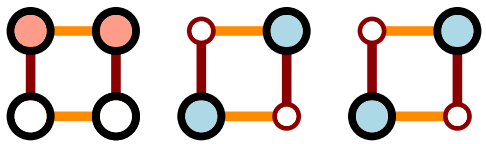
- Pick v_1 , place it
- Assign x-coords. to cycle in $\mathcal{C}_1$



Four Matchings



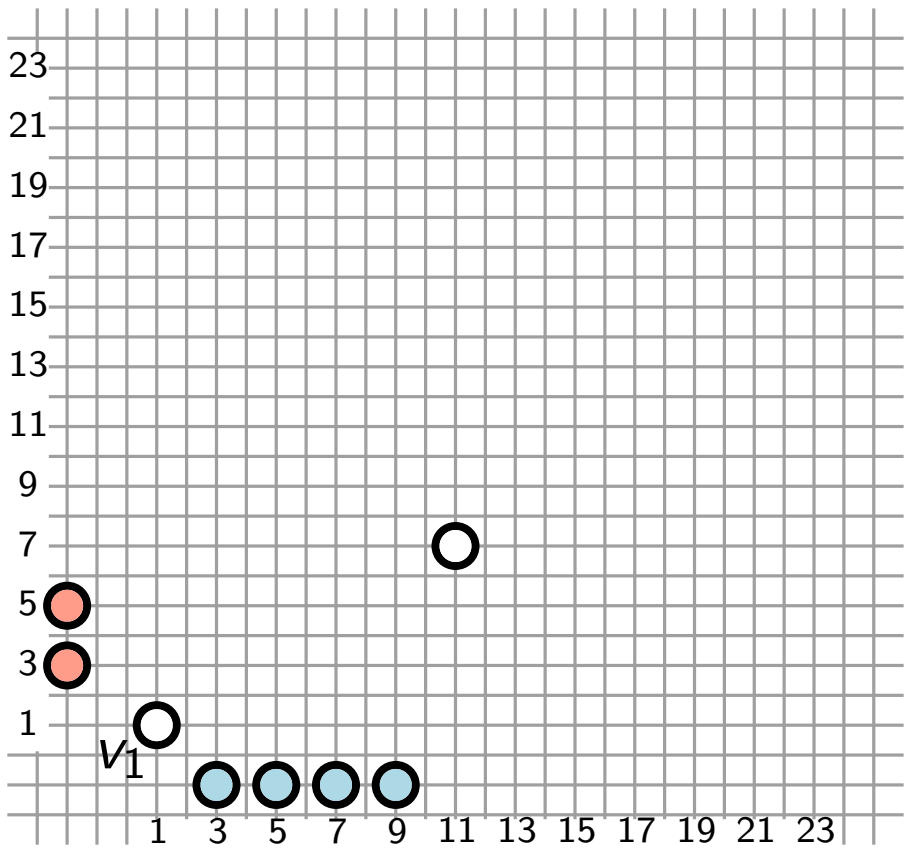
$\mathcal{C}_1$



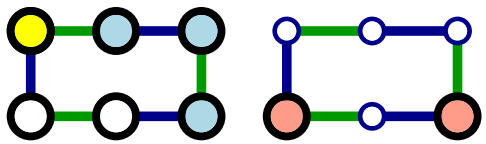
$\mathcal{C}_2$

Placement algorithm:

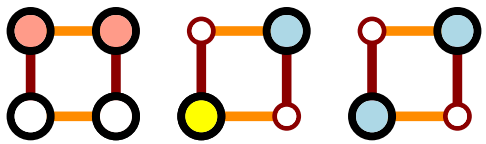
- Pick v_1 , place it
- Assign x-coords. to cycle in $\mathcal{C}_1$
- Assign y-coords. to cycle in $\mathcal{C}_2$



Four Matchings



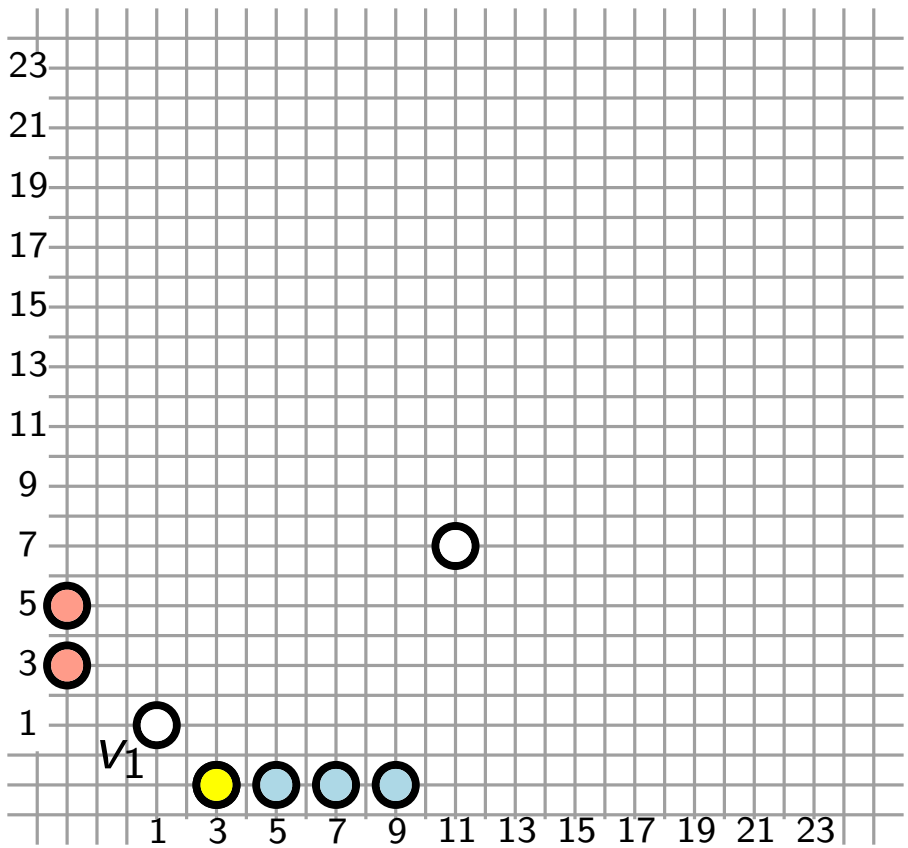
$\mathcal{C}_1$



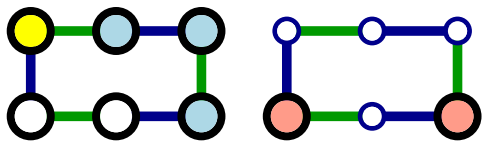
$\mathcal{C}_2$

Placement algorithm:

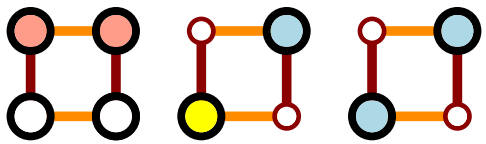
- Pick v_1 , place it
- Assign x-coords. to cycle in $\mathcal{C}_1$
- Assign y-coords. to cycle in $\mathcal{C}_2$
- Pick v with 1 assigned coord.



Four Matchings



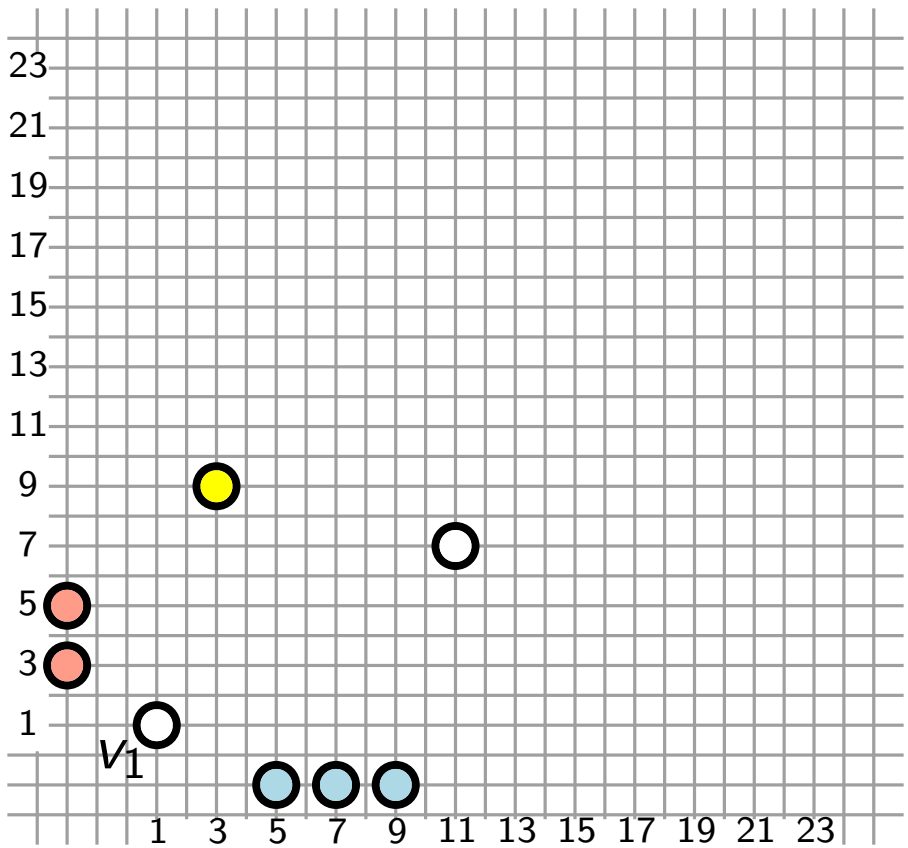
$\mathcal{C}_1$



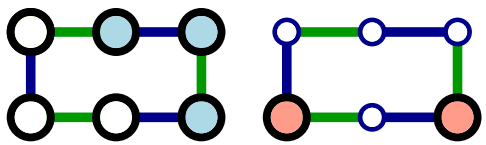
$\mathcal{C}_2$

Placement algorithm:

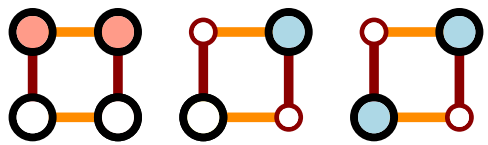
- Pick v_1 , place it
- Assign x-coords. to cycle in $\mathcal{C}_1$
- Assign y-coords. to cycle in $\mathcal{C}_2$
- Pick v with 1 assigned coord.
- Place v



Four Matchings



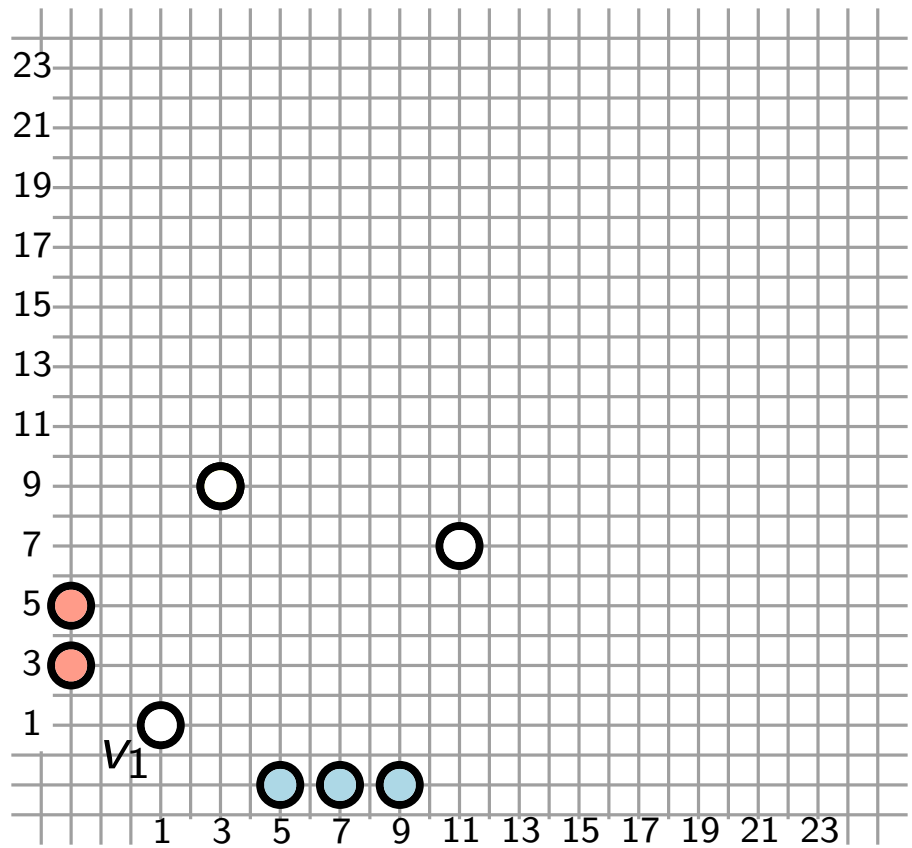
$\mathcal{C}_1$



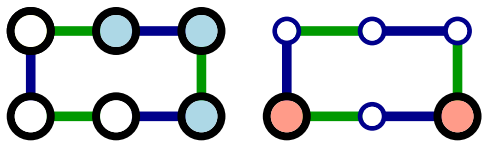
$\mathcal{C}_2$

Placement algorithm:

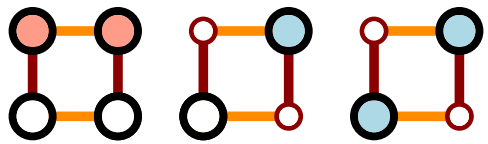
- Pick v_1 , place it
- Assign x-coords. to cycle in $\mathcal{C}_1$
- Assign y-coords. to cycle in $\mathcal{C}_2$
- Pick v with 1 assigned coord.
- Place v



Four Matchings



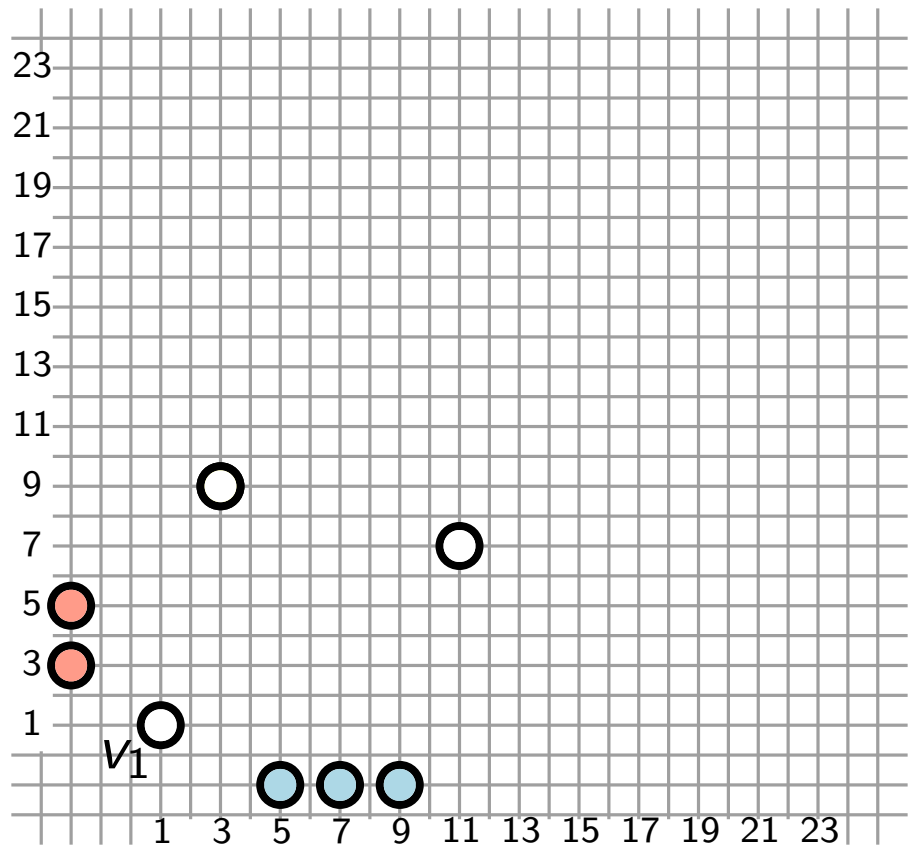
$\mathcal{C}_1$



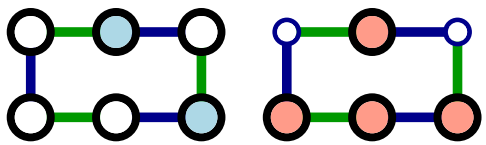
$\mathcal{C}_2$

Placement algorithm:

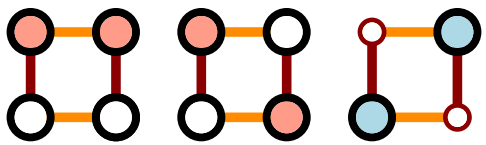
- Pick v_1 , place it
- Assign x-coords. to cycle in $\mathcal{C}_1$
- Assign y-coords. to cycle in $\mathcal{C}_2$
- Pick v with 1 assigned coord.
- Place v



Four Matchings



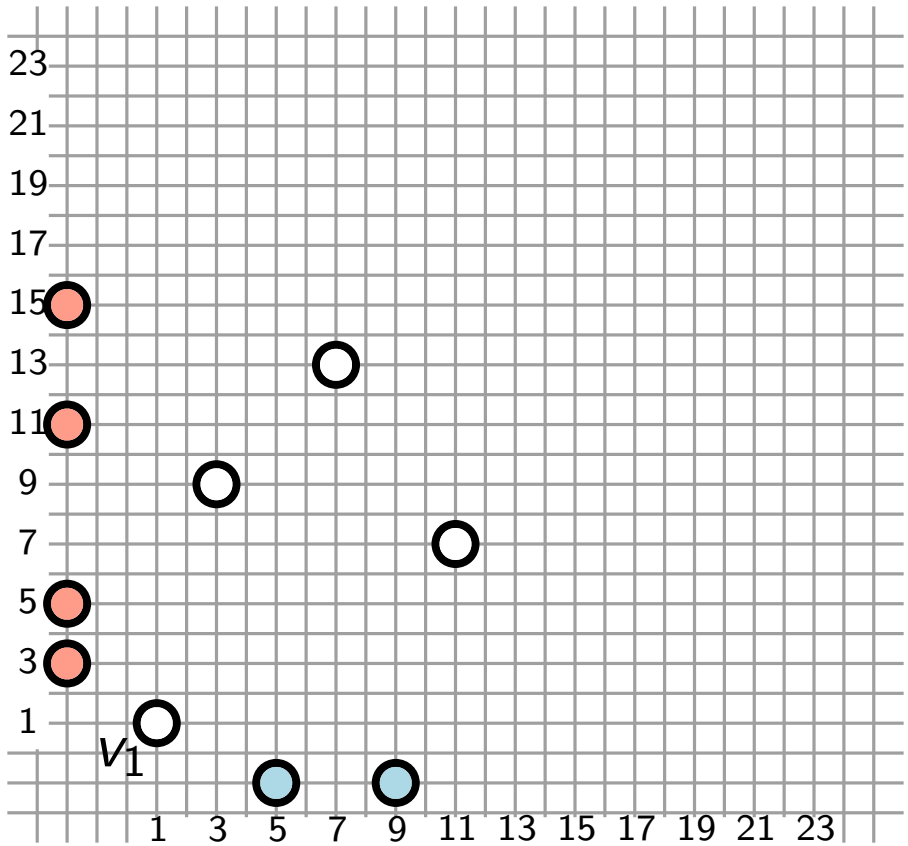
$\mathcal{C}_1$



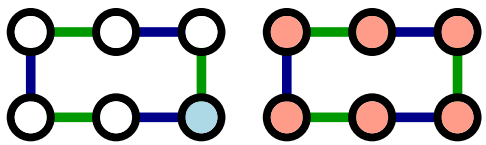
$\mathcal{C}_2$

Placement algorithm:

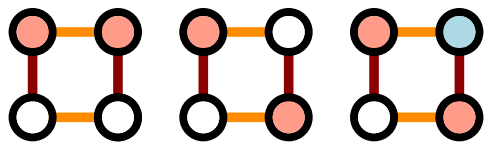
- Pick v_1 , place it
- Assign x-coords. to cycle in $\mathcal{C}_1$
- Assign y-coords. to cycle in $\mathcal{C}_2$
- Pick v with 1 assigned coord.
- Place v



Four Matchings



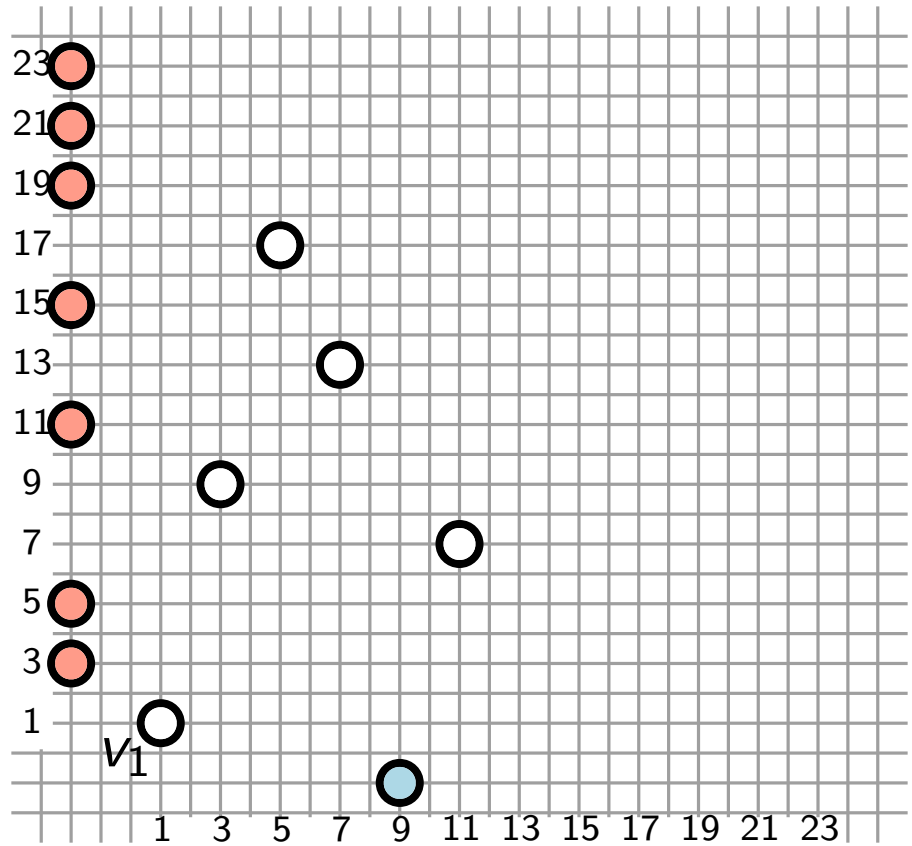
$\mathcal{C}_1$



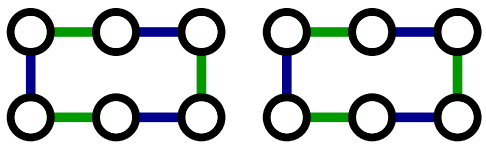
$\mathcal{C}_2$

Placement algorithm:

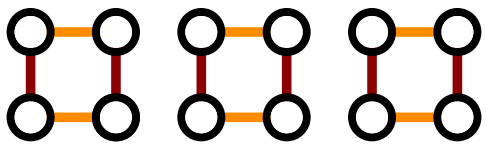
- Pick v_1 , place it
- Assign x-coords. to cycle in $\mathcal{C}_1$
- Assign y-coords. to cycle in $\mathcal{C}_2$
- Pick v with 1 assigned coord.
- Place v



Four Matchings



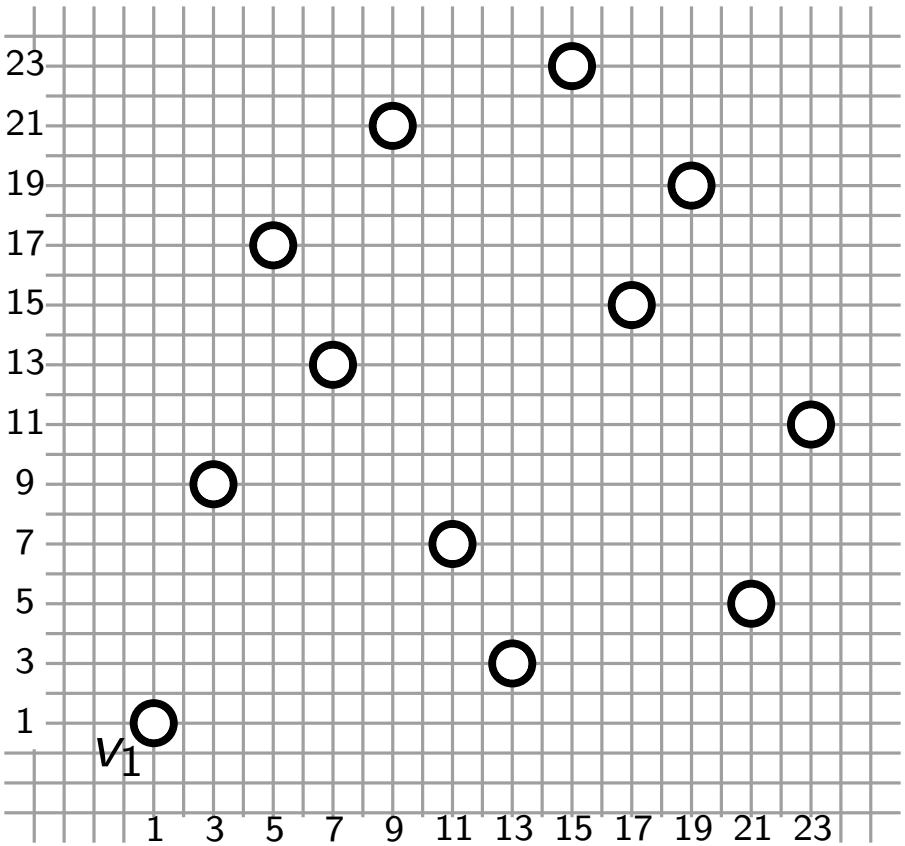
$\mathcal{C}_1$



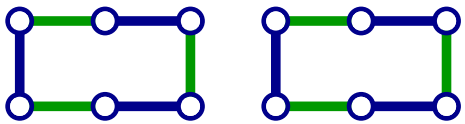
$\mathcal{C}_2$

Placement algorithm:

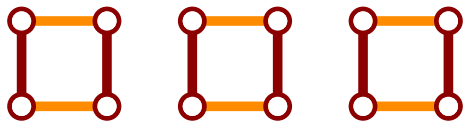
- Pick v_1 , place it
- Assign x-coords. to cycle in $\mathcal{C}_1$
- Assign y-coords. to cycle in $\mathcal{C}_2$
- Pick v with 1 assigned coord.
- Place v



Four Matchings



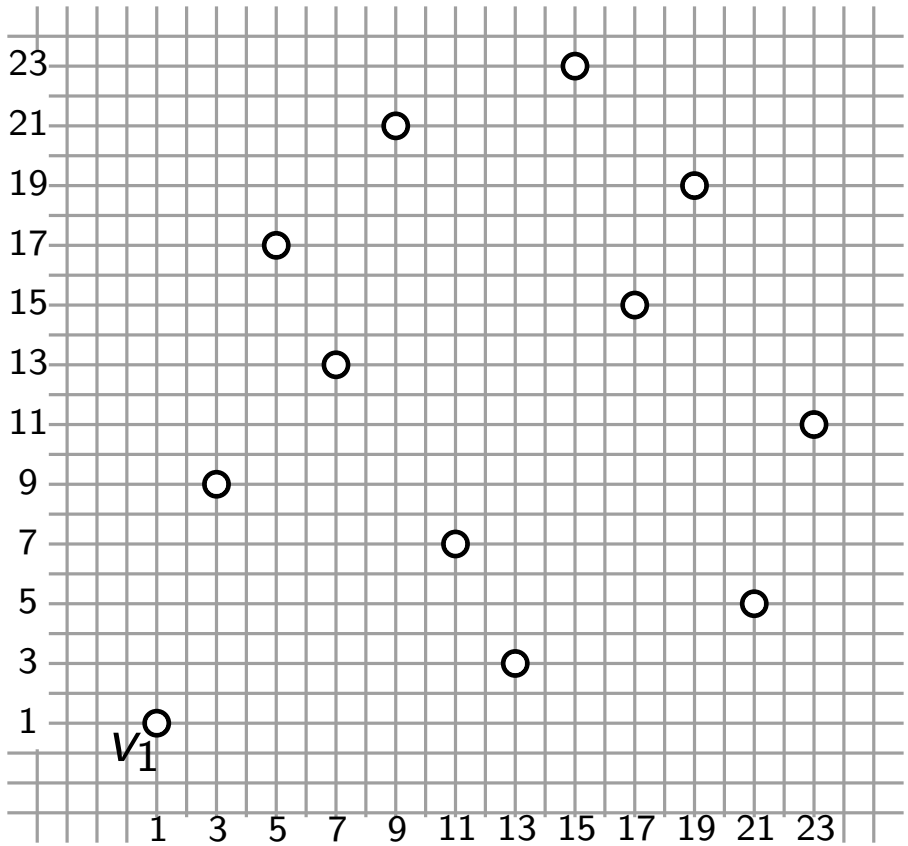
$\mathcal{C}_1$



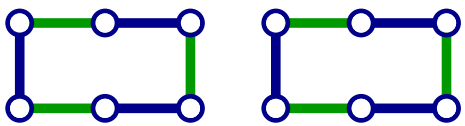
$\mathcal{C}_2$

Placement algorithm:

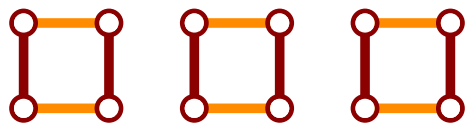
- Pick v_1 , place it
- Assign x-coords. to cycle in $\mathcal{C}_1$
- Assign y-coords. to cycle in $\mathcal{C}_2$
- Pick v with 1 assigned coord.
- Place v



Four Matchings



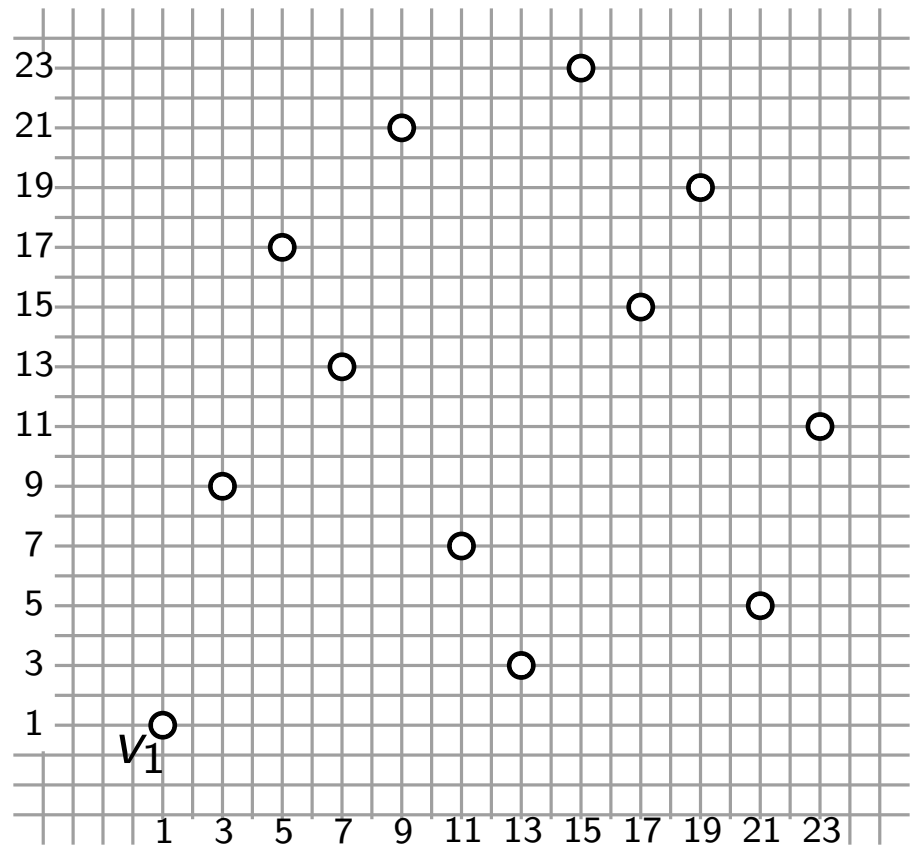
$\mathcal{C}_1$



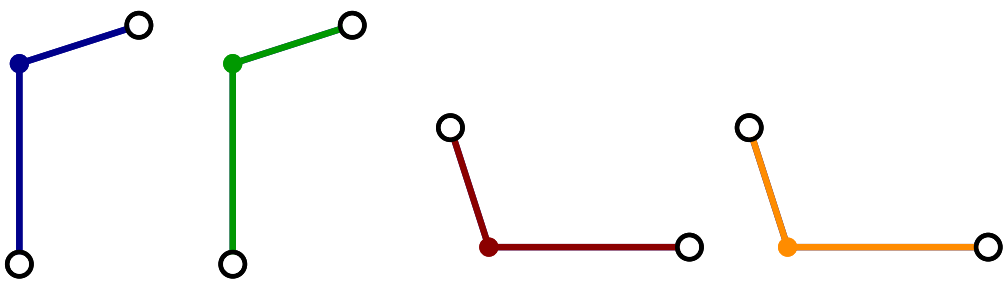
$\mathcal{C}_2$

Placement algorithm:

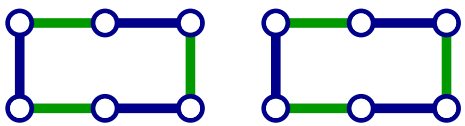
- Pick v_1 , place it
- Assign x-coords. to cycle in $\mathcal{C}_1$
- Assign y-coords. to cycle in $\mathcal{C}_2$
- Pick v with 1 assigned coord.
- Place v



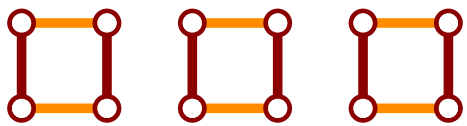
Edges:



Four Matchings



$\mathcal{C}_1$

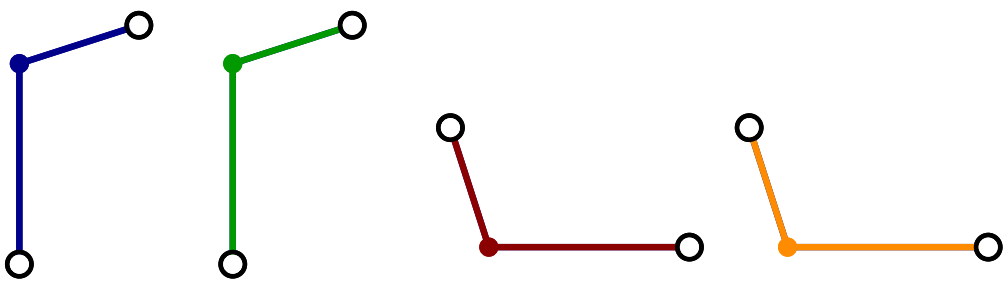


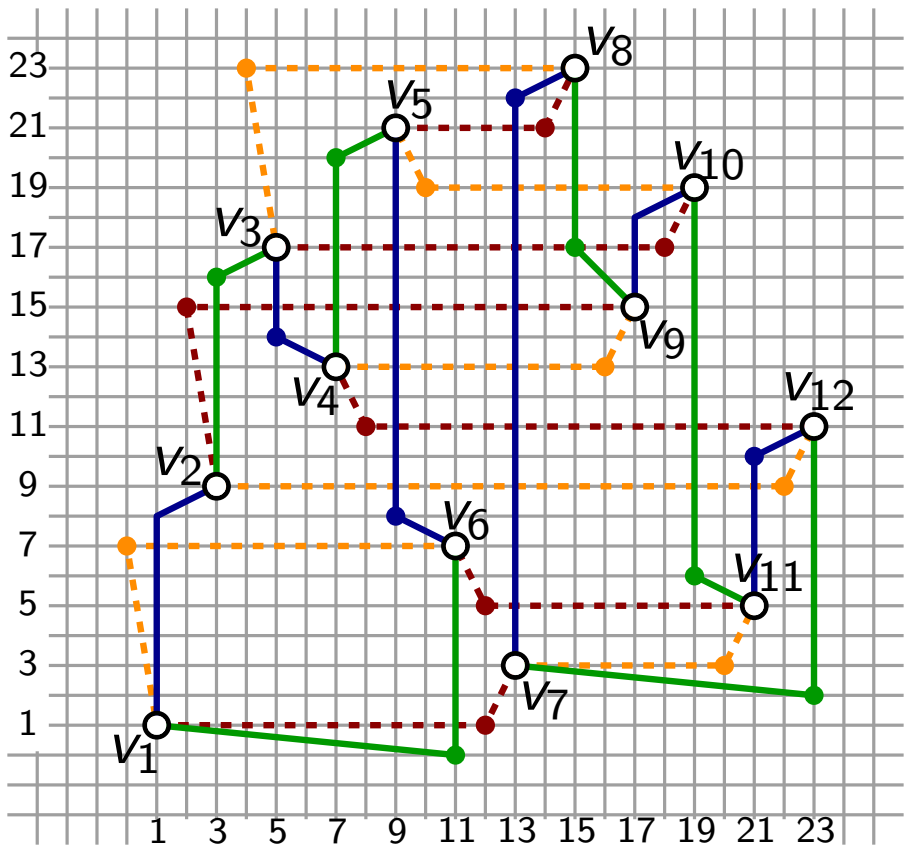
$\mathcal{C}_2$

Placement algorithm:

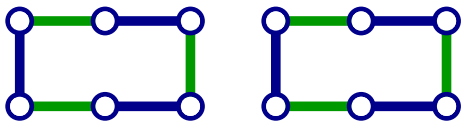
- Pick v_1 , place it
- Assign x-coords. to cycle in $\mathcal{C}_1$
- Assign y-coords. to cycle in $\mathcal{C}_2$
- Pick v with 1 assigned coord.
- Place v

Edges:

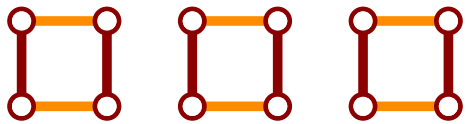




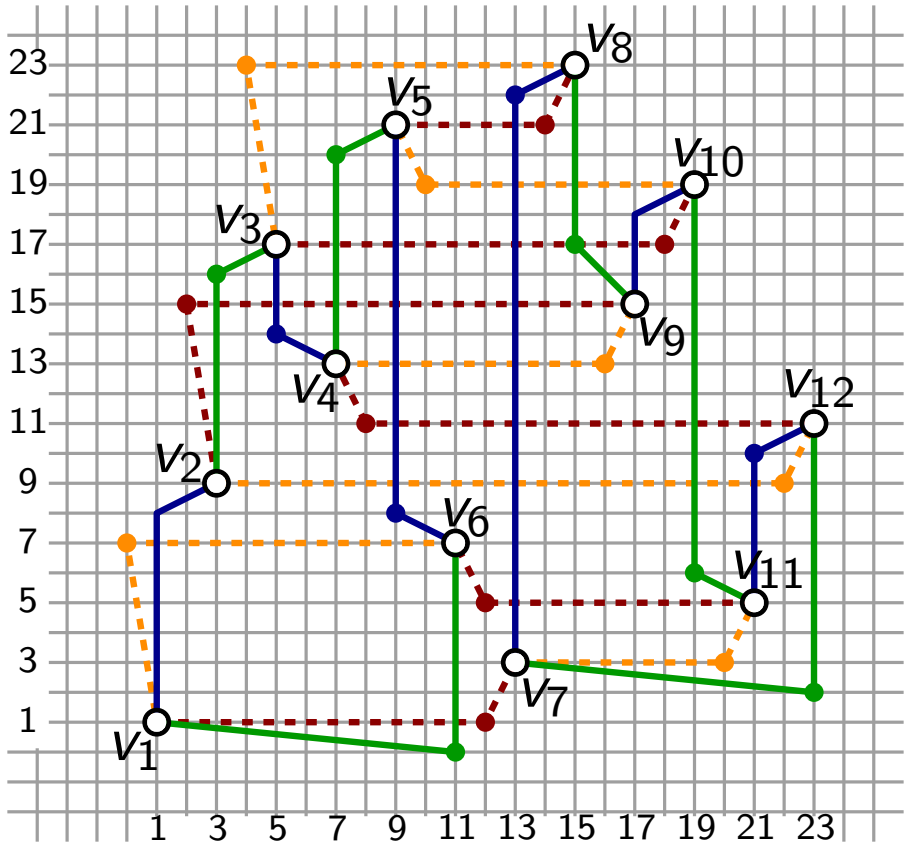
Four Matchings



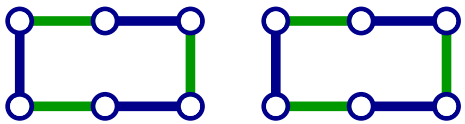
$\mathcal{C}_1$



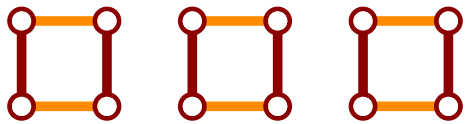
$\mathcal{C}_2$



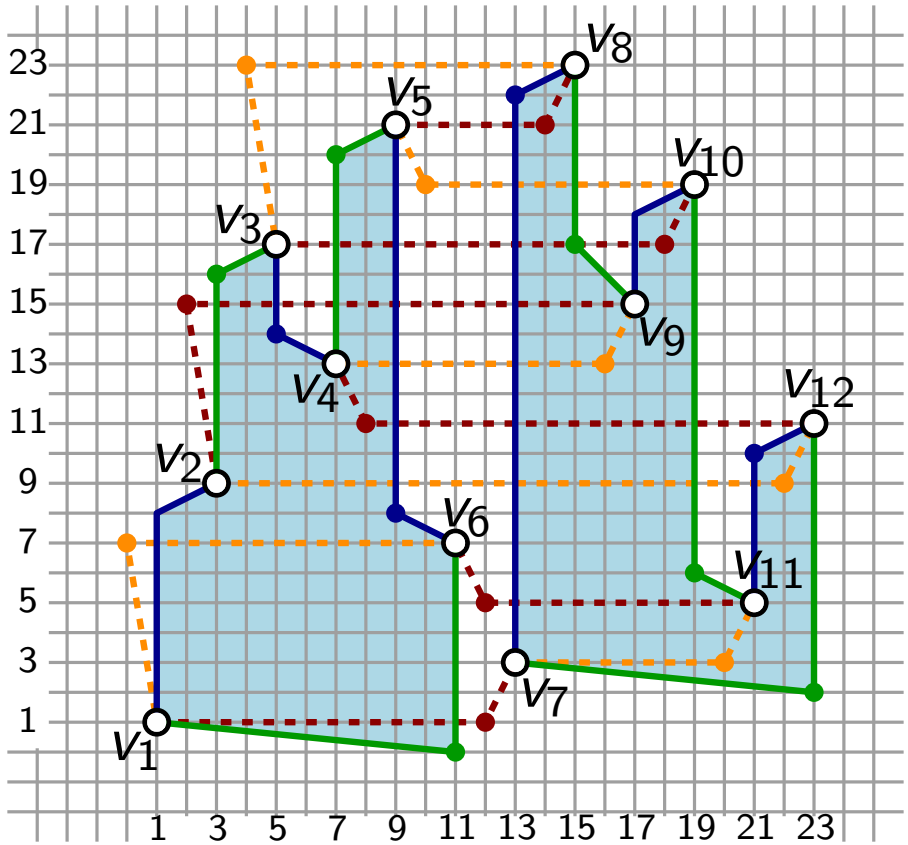
Four Matchings



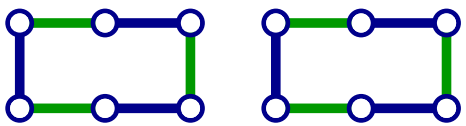
$\mathcal{C}_1$



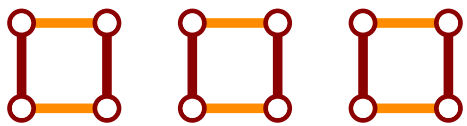
$\mathcal{C}_2$



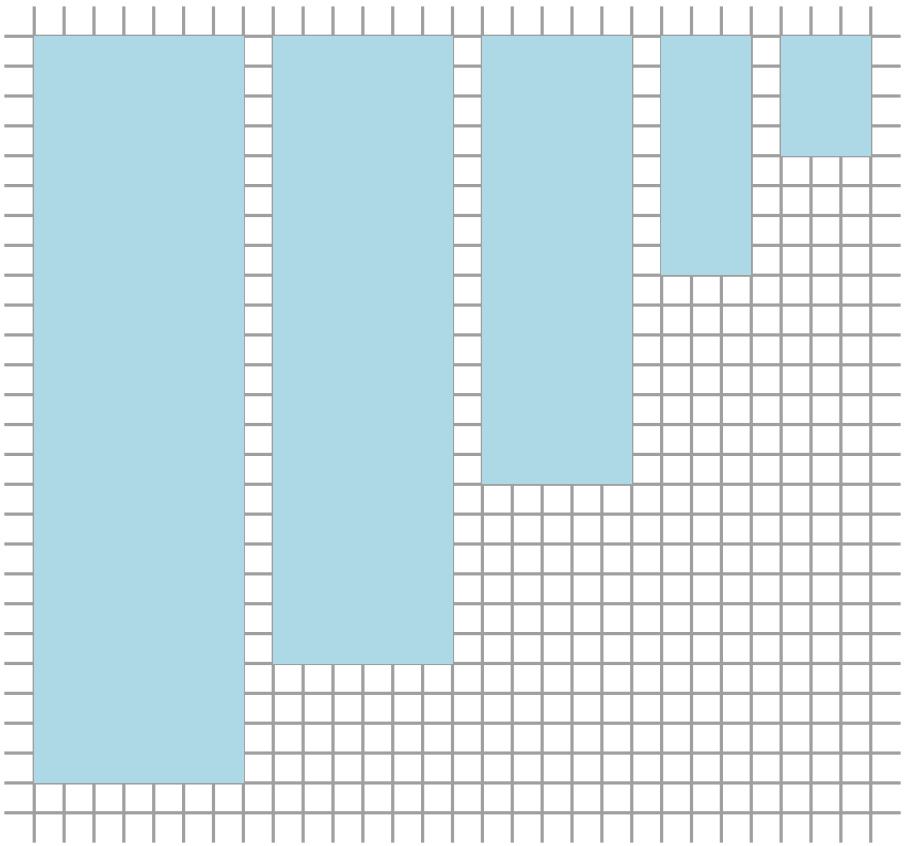
Four Matchings

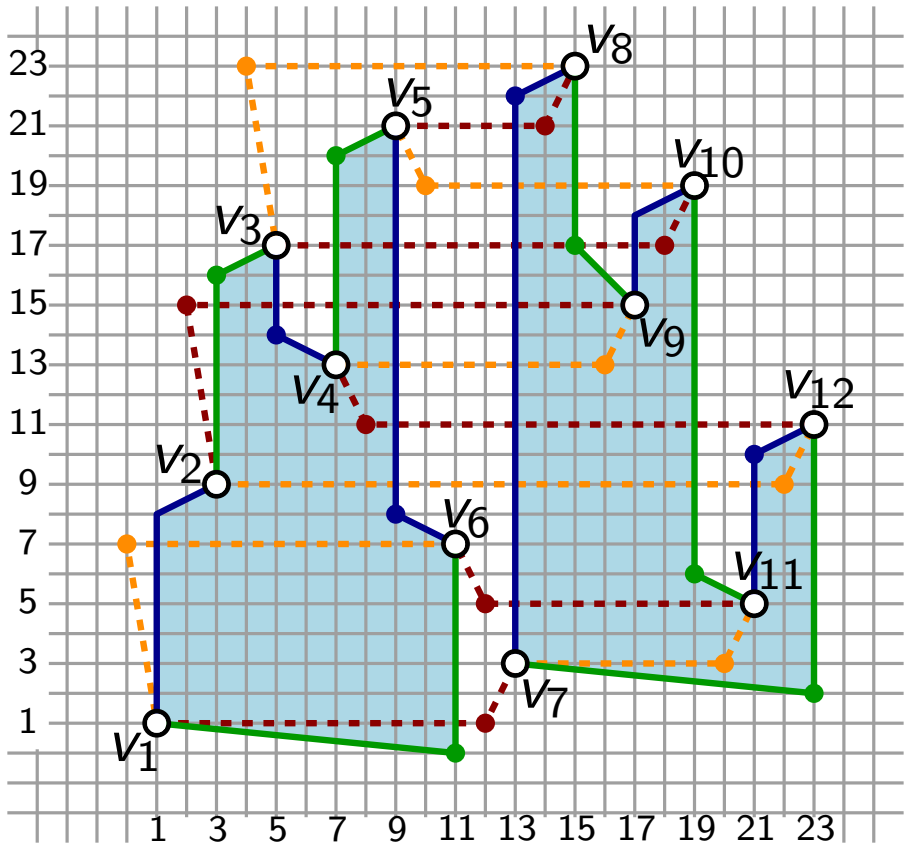


$\mathcal{C}_1$

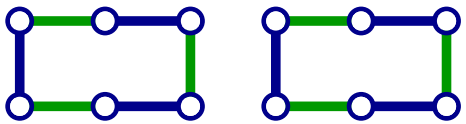


$\mathcal{C}_2$

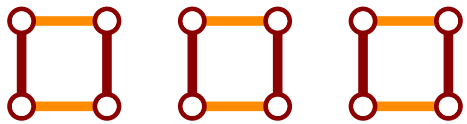




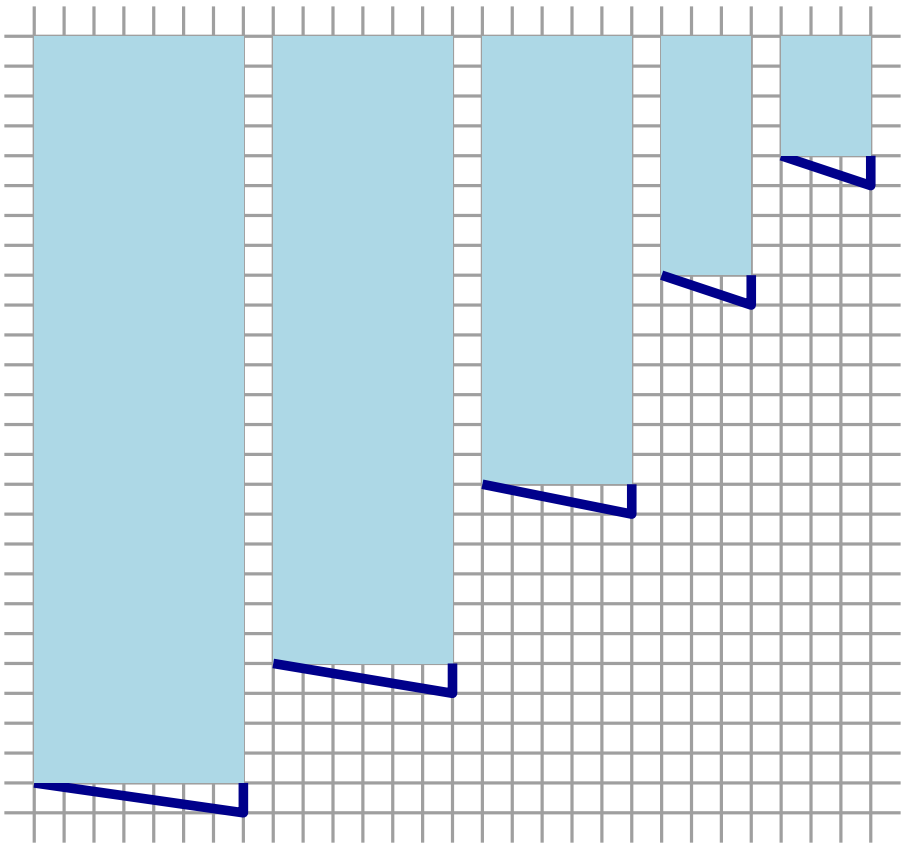
Four Matchings

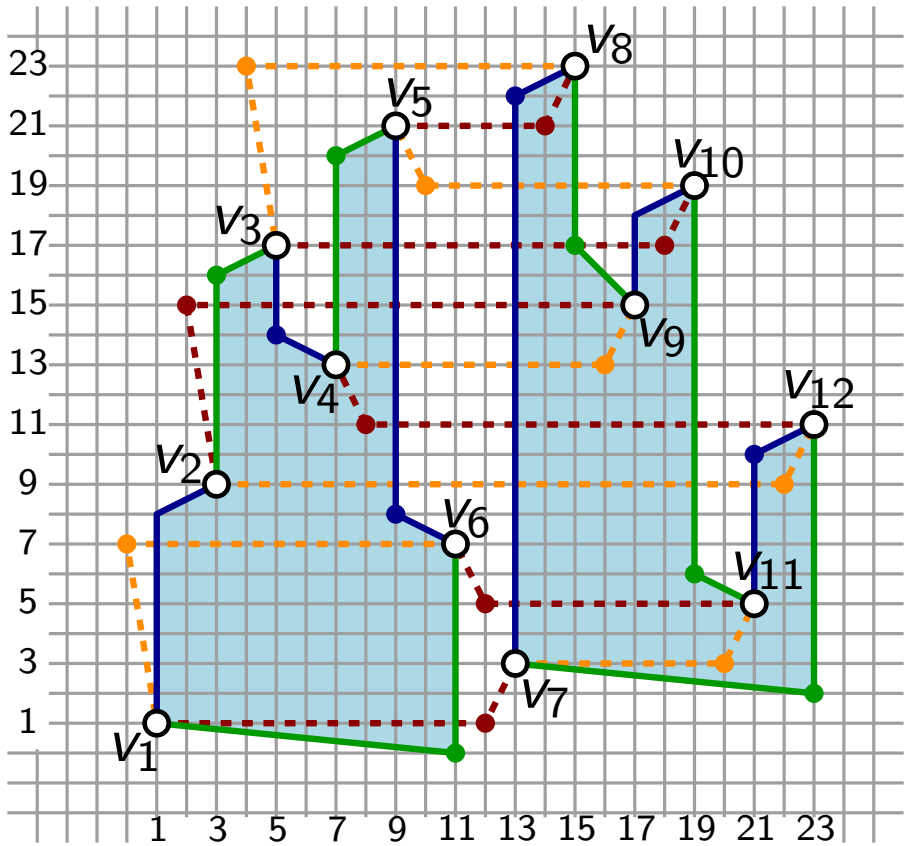


$\mathcal{C}_1$

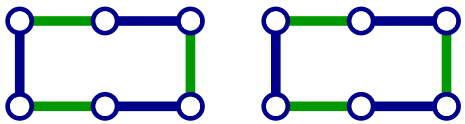


$\mathcal{C}_2$

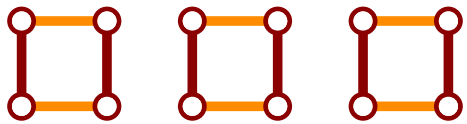




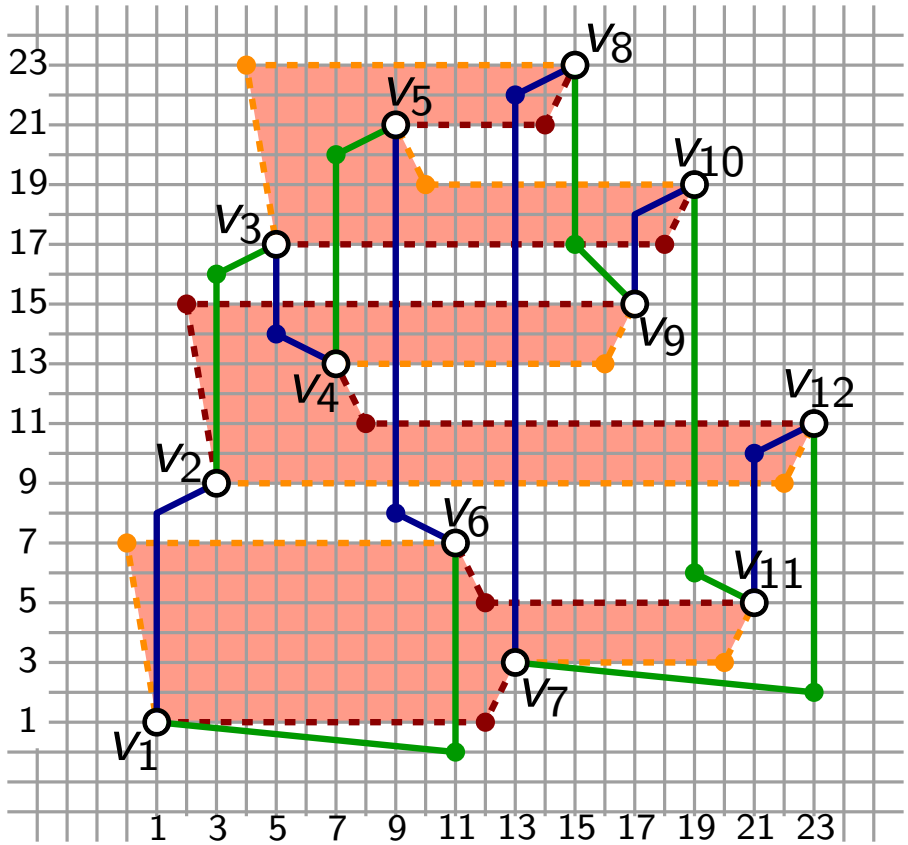
Four Matchings



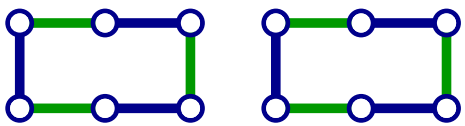
$\mathcal{C}_1$



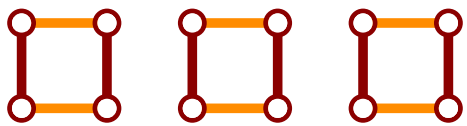
$\mathcal{C}_2$



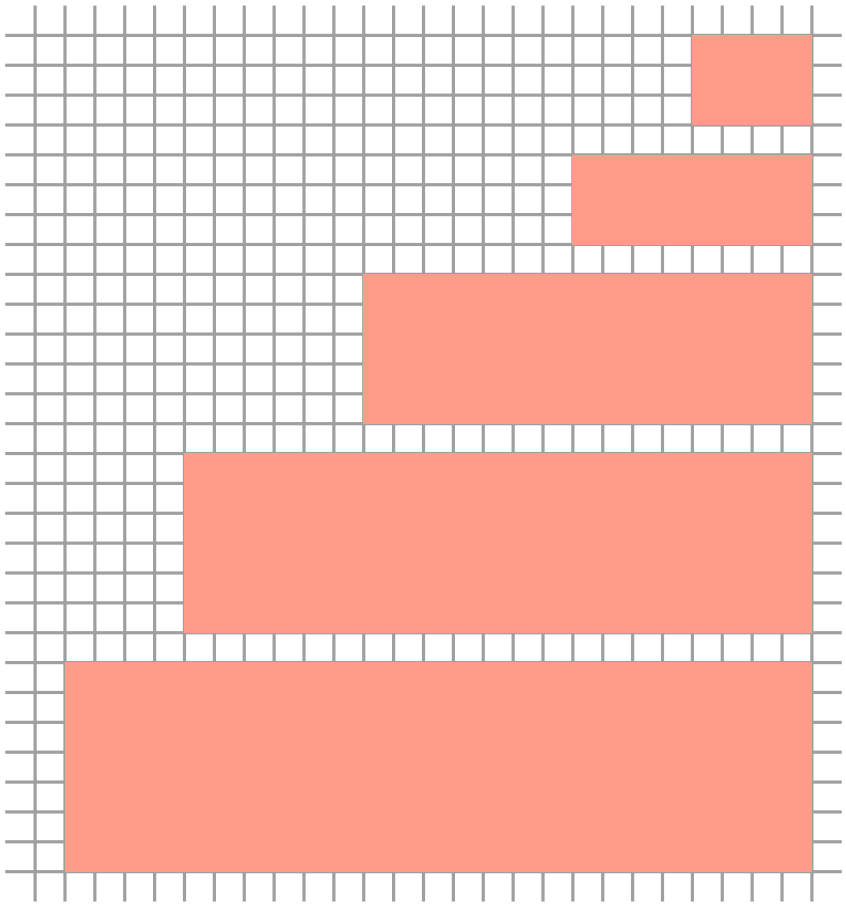
Four Matchings

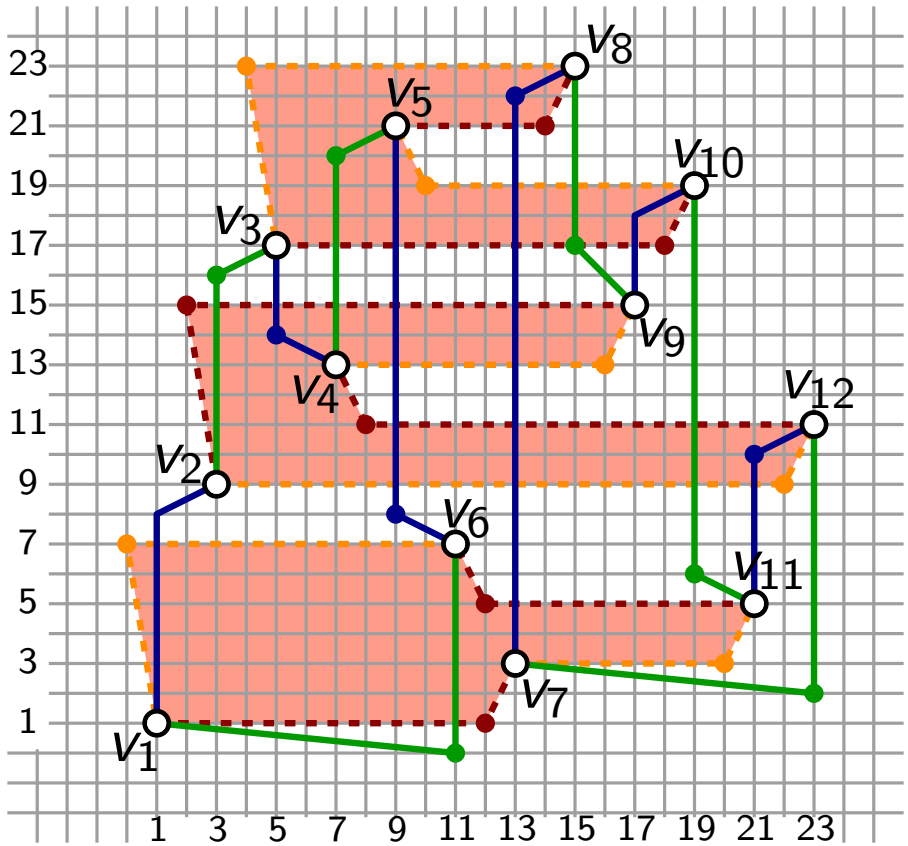


$\mathcal{C}_1$

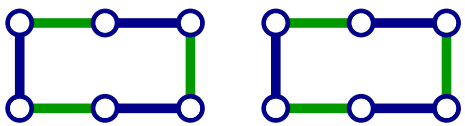


$\mathcal{C}_2$

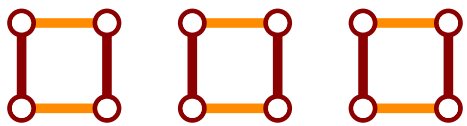




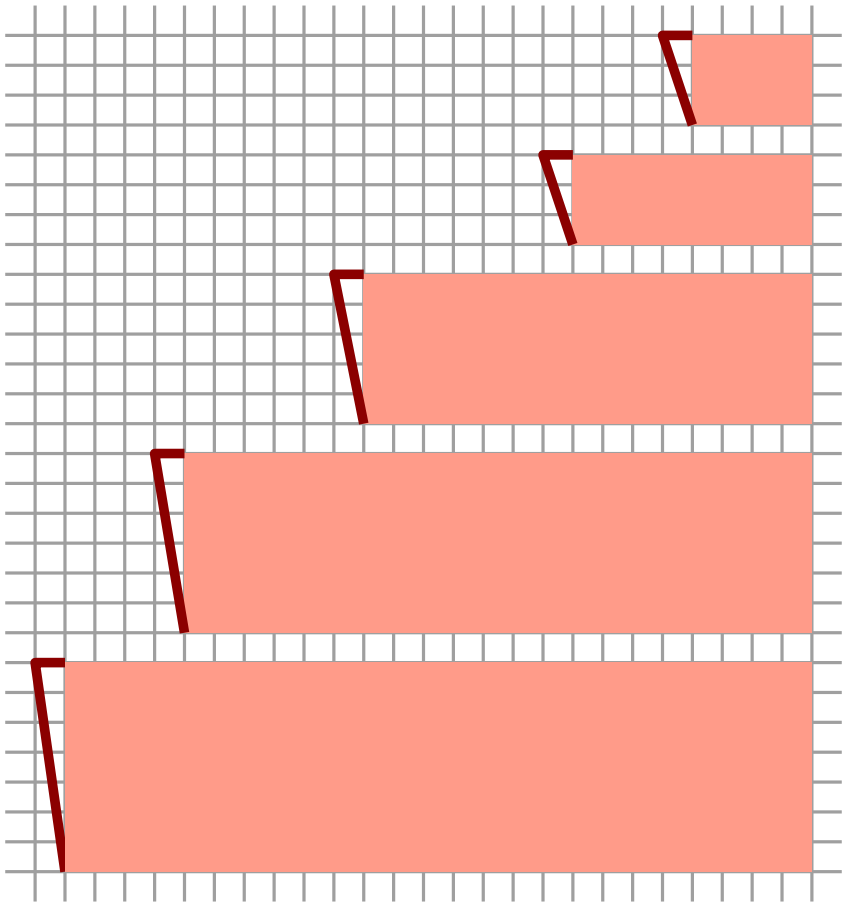
Four Matchings

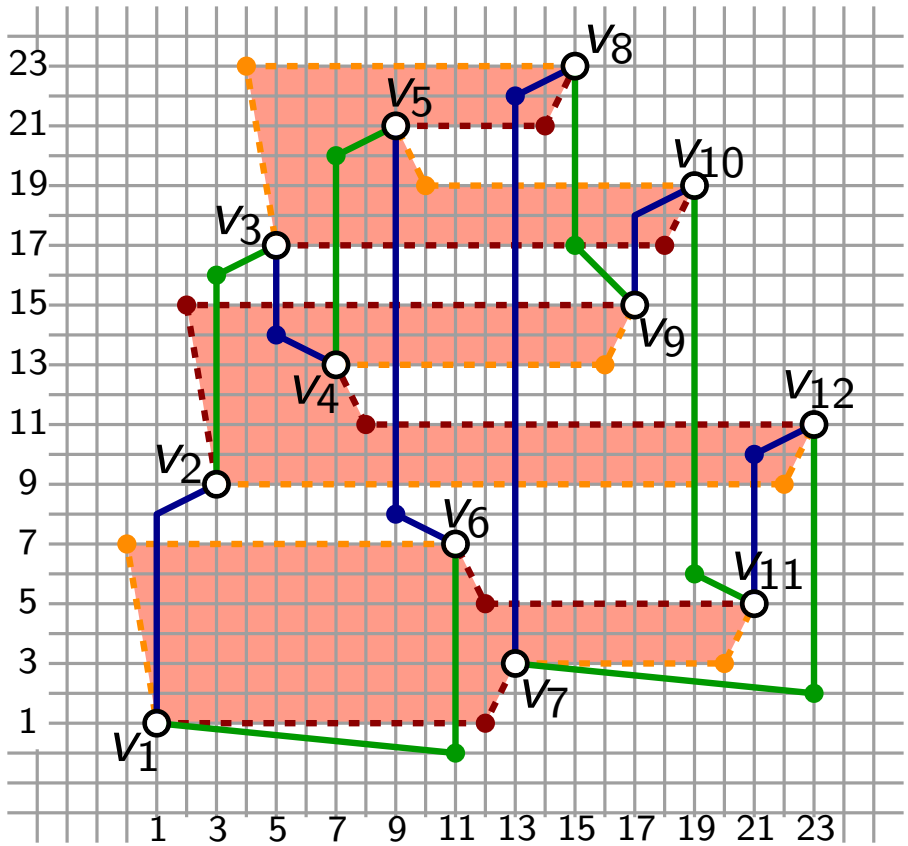


$\mathcal{C}_1$

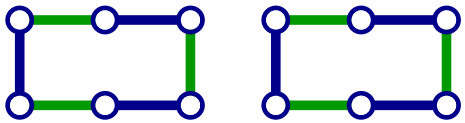


$\mathcal{C}_2$

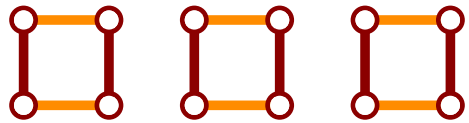




Four Matchings



$\mathcal{C}_1$



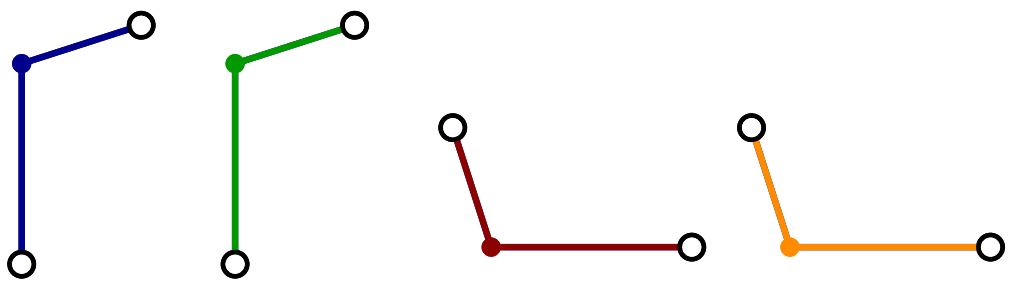
$\mathcal{C}_2$

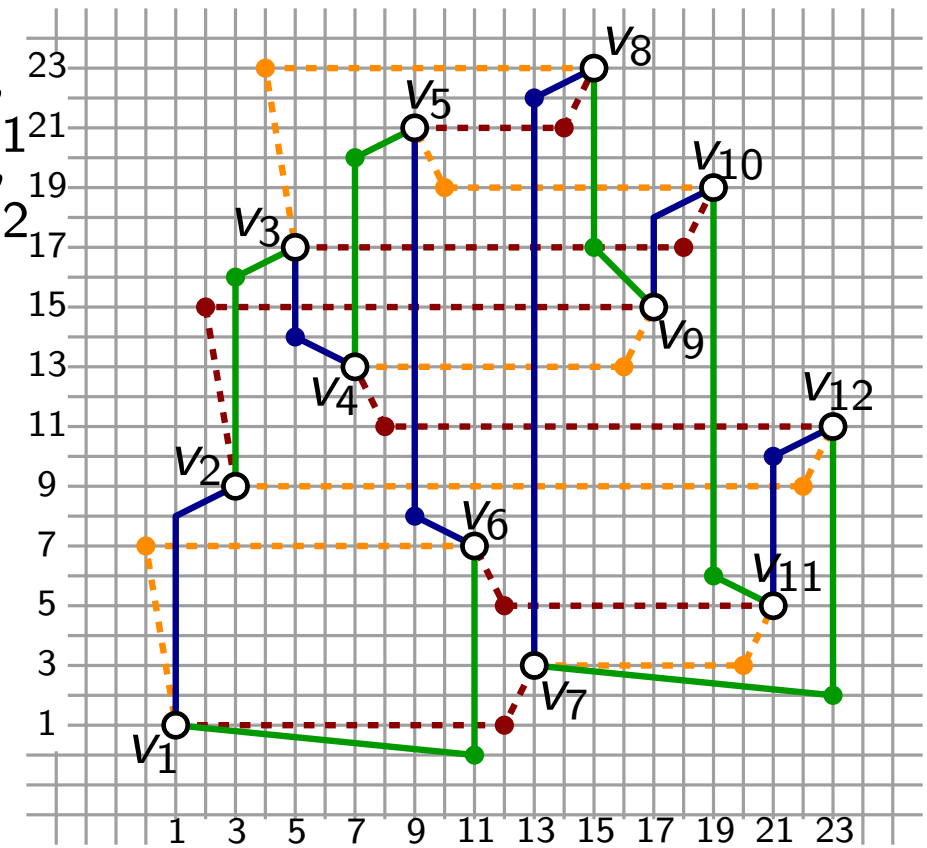
Bends: 1×1
Grid size:
$2n \times 2n$

Placement algorithm:

- Pick v_1 , place it
- Assign x-coords. to cycle in $\mathcal{C}_1$
- Assign y-coords. to cycle in $\mathcal{C}_2$
- Pick v with 1 assigned coord.
- Place v

Edges:





Overview

Graph classes			Number of bends
Cycle	$\times$	Cycle	1×1
Caterpillar	$\times$	Cycle	1×1
Four Matchings			$1 \times 1 \times 1 \times 1$
Tree	$\times$	Matching	1×0
Wheel	$\times$	Matching	2×0
Outerpath	$\times$	Matching	2×1
Outerplanar	$\times$	Outerplanar	3×3
2-page book emb.	$\times$	2-page book emb.	4×4
Planar	$\times$	Planar	6×6

Tree $\times$ Matching

Idea:
- Matching as horizontal line, Tree with 1 bend

Tree $\times$ Matching

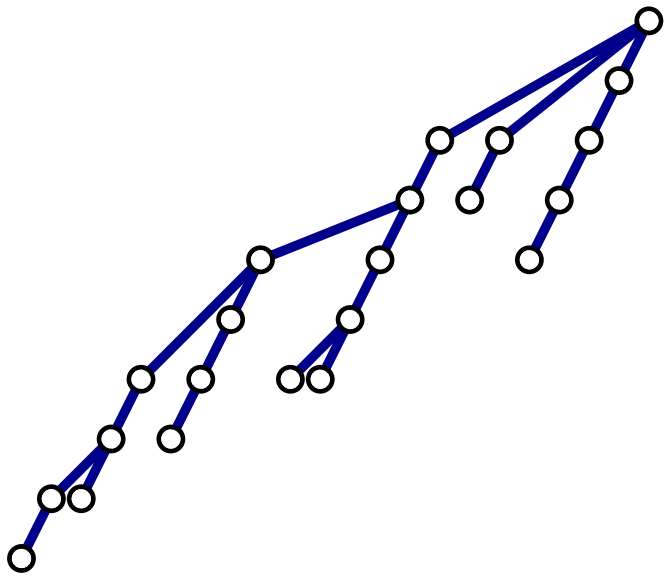
Idea:
- Matching as horizontal line, Tree with 1 bend
- Place Matching edges inductively $\Rightarrow$ y-coord.

Tree $\times$ Matching

Idea:
- Matching as horizontal line, Tree with 1 bend
- Place Matching edges inductively $\Rightarrow$ y-coord.
- Use post-order on Tree $\Rightarrow$ x-coord.

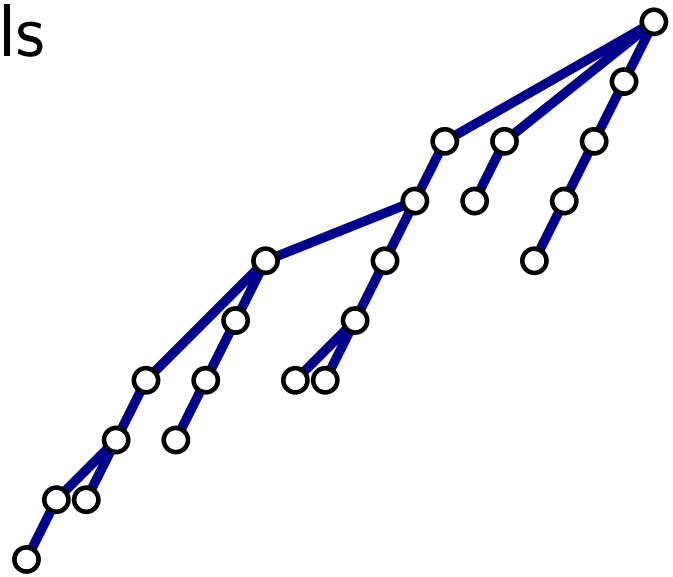
Tree $\times$ Matching

Idea:
- Matching as horizontal line, Tree with 1 bend
- Place Matching edges inductively $\Rightarrow$ y-coord.
- Use post-order on Tree $\Rightarrow$ x-coord.



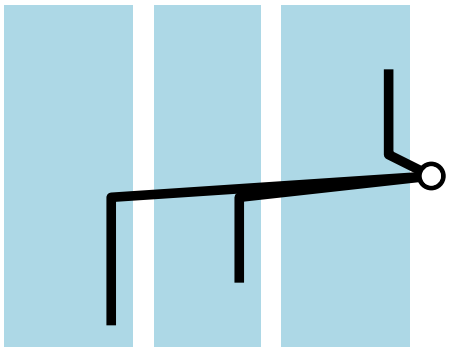
Tree $\times$ Matching

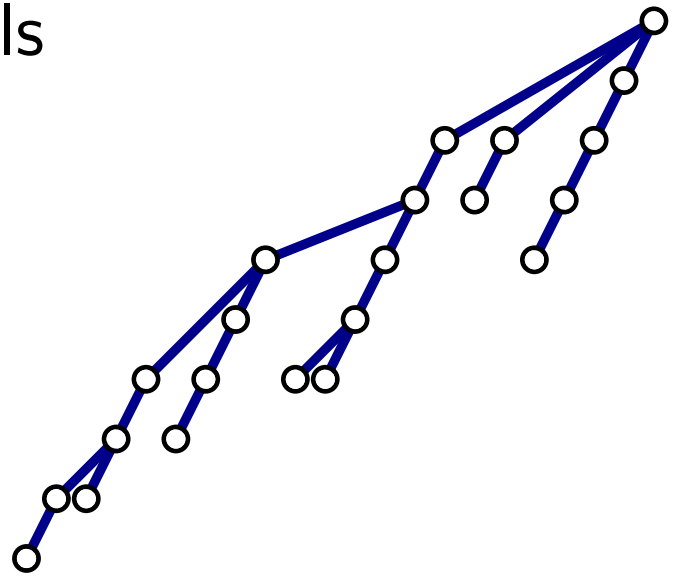
Idea:
- Matching as horizontal line, Tree with 1 bend
- Place Matching edges inductively $\Rightarrow$ y-coord.
- Use post-order on Tree $\Rightarrow$ x-coord.
- $\Rightarrow$ Subtrees in disjoint x-intervals



Tree $\times$ Matching

Idea:
- Matching as horizontal line, Tree with 1 bend
- Place Matching edges inductively $\Rightarrow$ y-coord.
- Use post-order on Tree $\Rightarrow$ x-coord.
- $\Rightarrow$ Subtrees in disjoint x-intervals

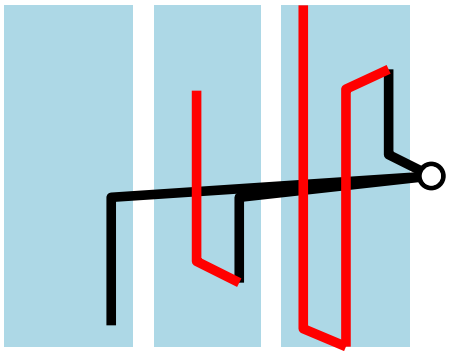


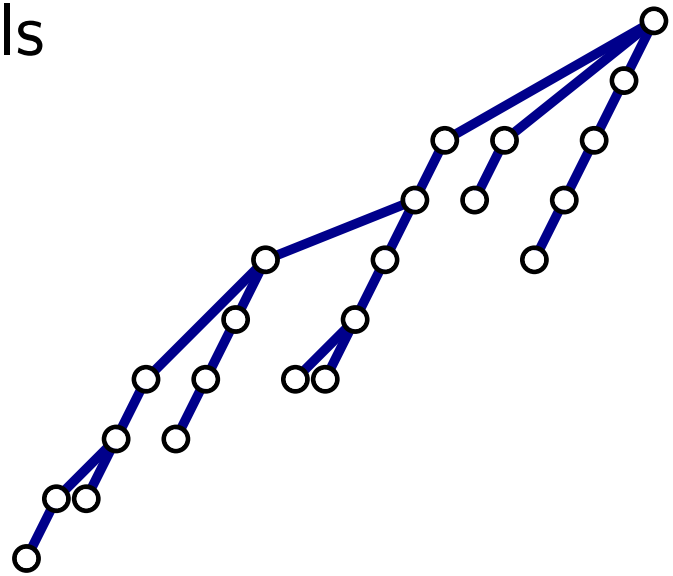


Tree $\times$ Matching

Idea:
- Matching as horizontal line, Tree with 1 bend
- Place Matching edges inductively $\Rightarrow$ y-coord.
- Use post-order on Tree $\Rightarrow$ x-coord.
- $\Rightarrow$ Subtrees in disjoint x-intervals

Problem:

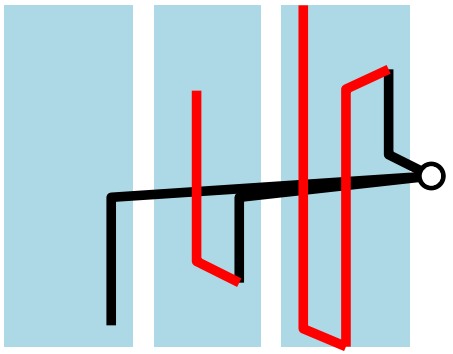


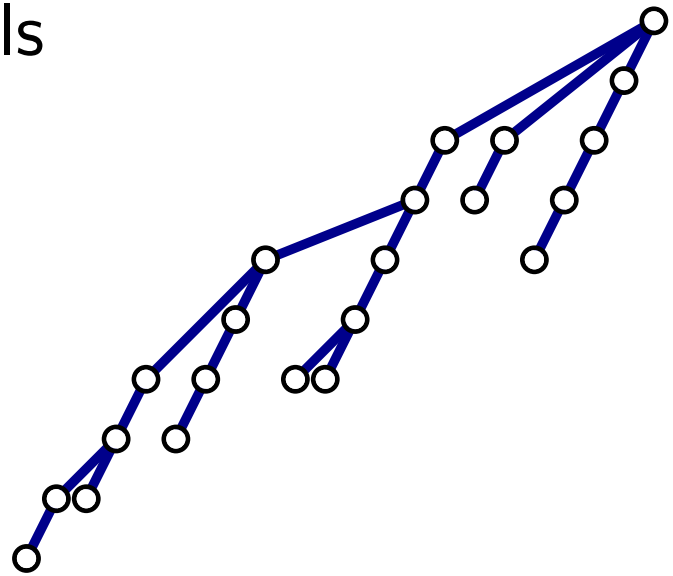


Tree $\times$ Matching

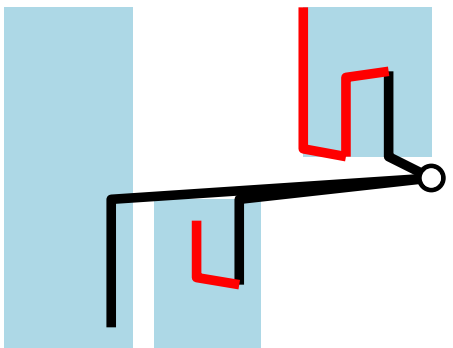
Idea:
- Matching as horizontal line, Tree with 1 bend
- Place Matching edges inductively $\Rightarrow$ y-coord.
- Use post-order on Tree $\Rightarrow$ x-coord.
- $\Rightarrow$ Subtrees in disjoint x-intervals

Problem:





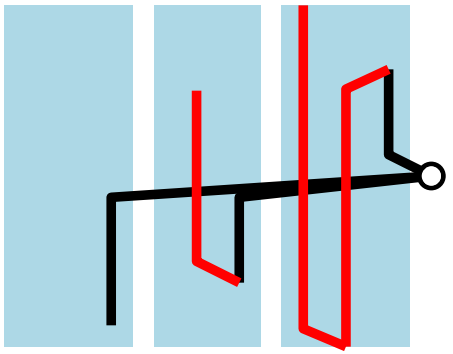
Solution:

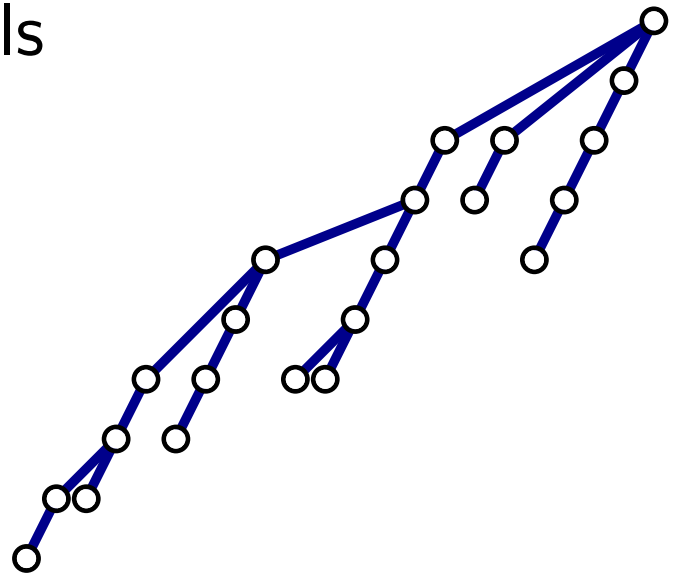


Tree $\times$ Matching

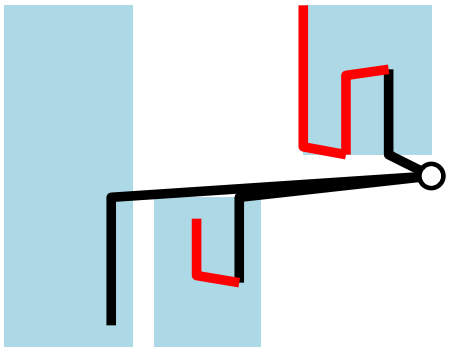
Idea:
- Matching as horizontal line, Tree with 1 bend
- Place Matching edges inductively $\Rightarrow$ y-coord.
- Use post-order on Tree $\Rightarrow$ x-coord.
- $\Rightarrow$ Subtrees in disjoint x-intervals

Problem:





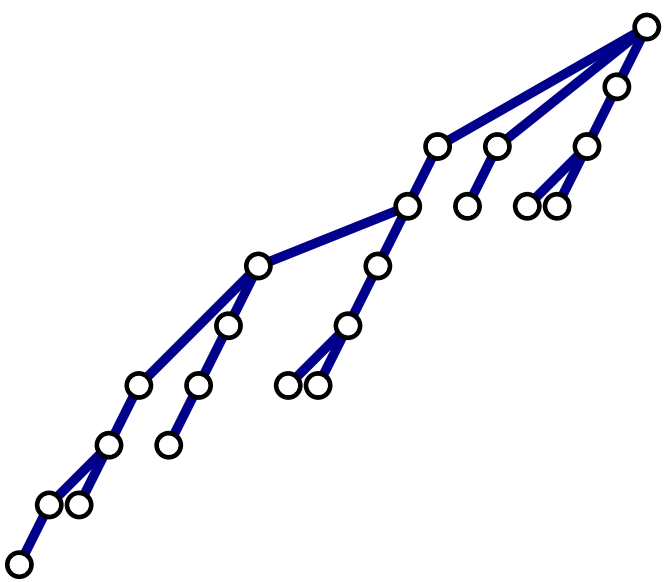
Solution:

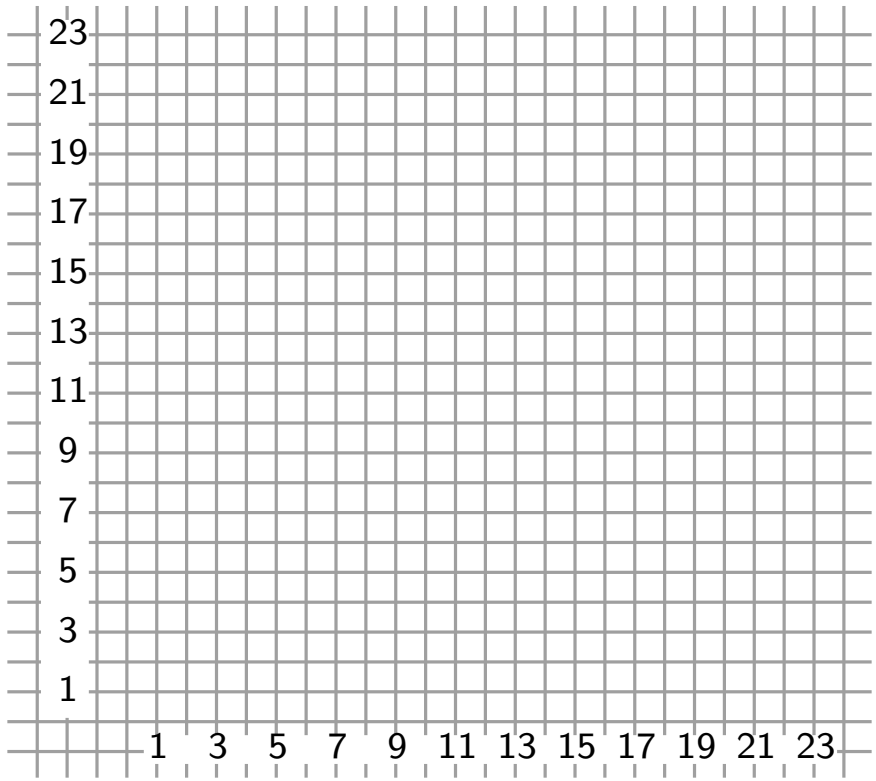


All but one subtree completely above or completely below

Tree $\times$ Matching

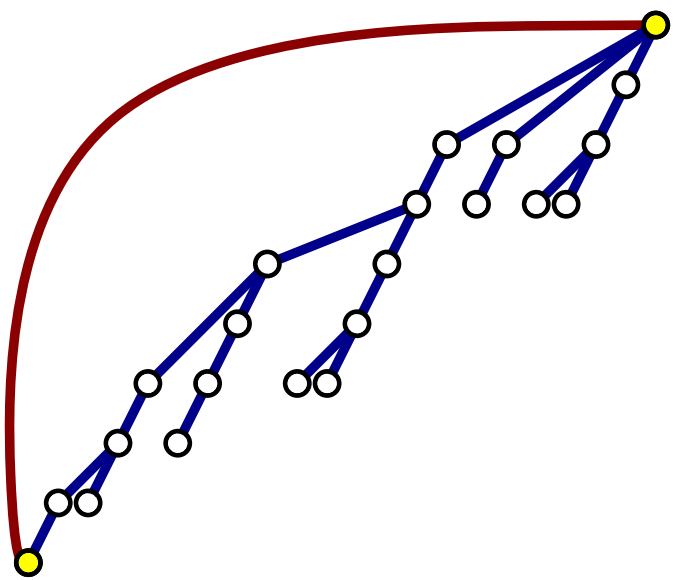
- Place root + matching at the top

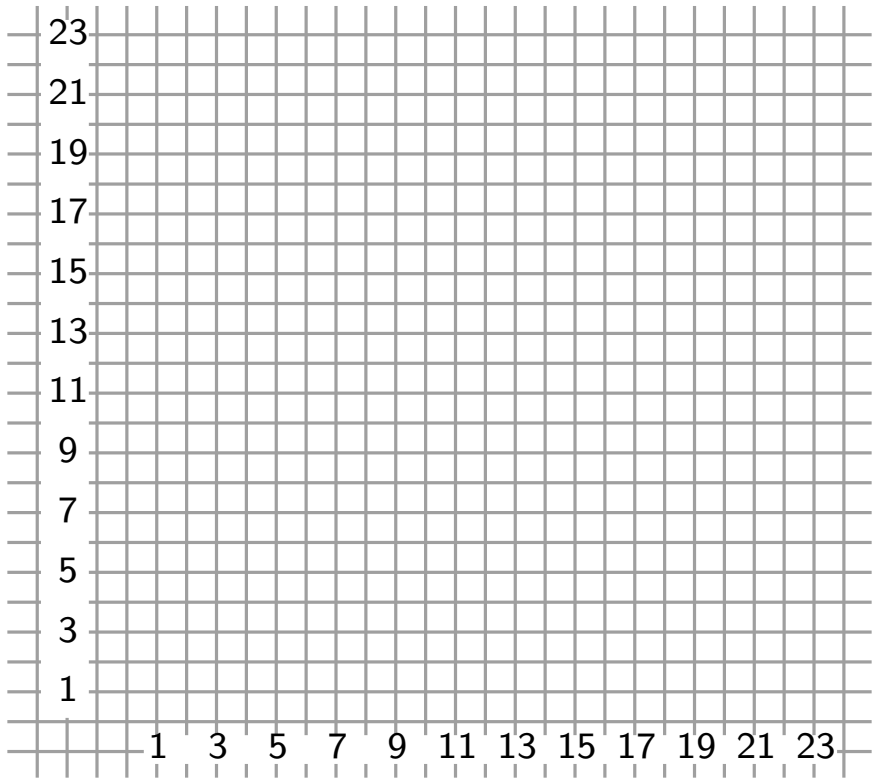




Tree $\times$ Matching

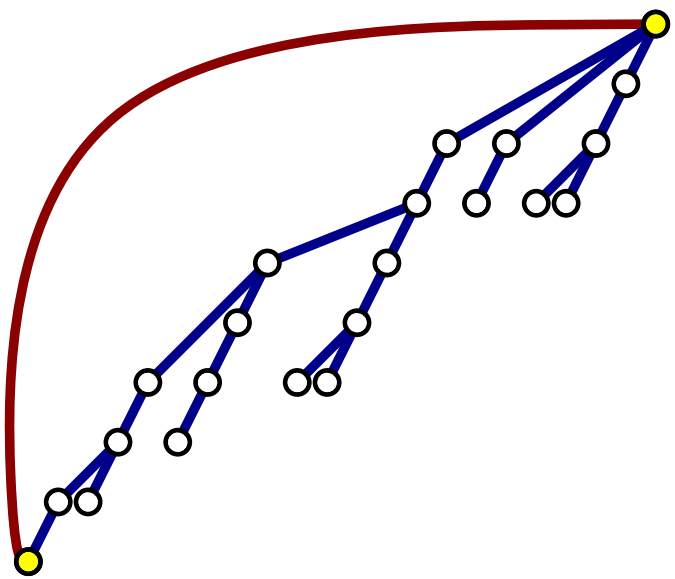
- Place root + matching at the top

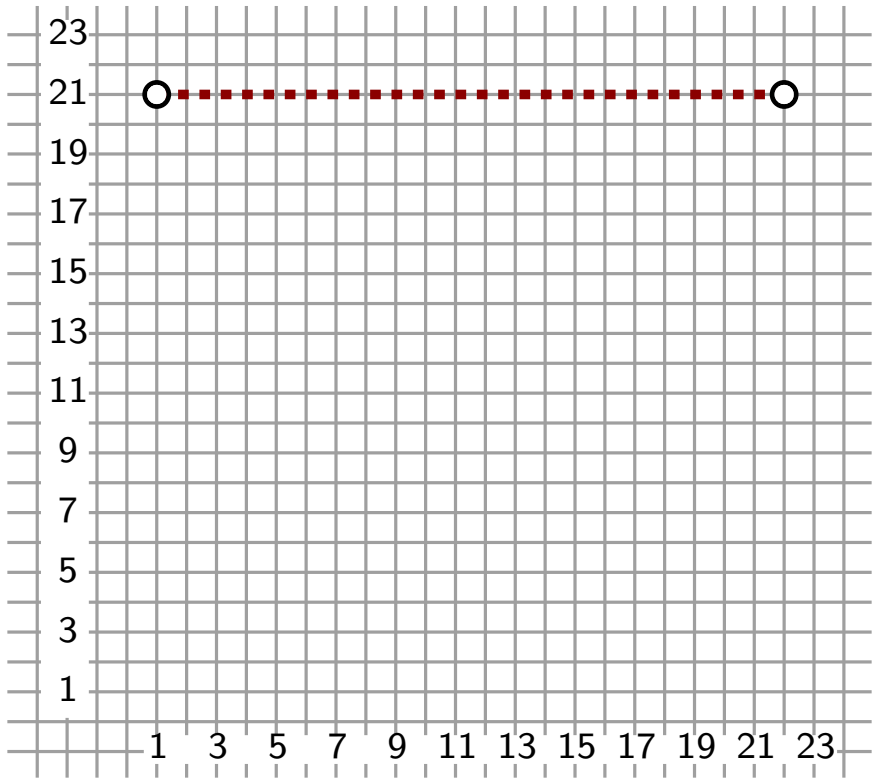




Tree $\times$ Matching

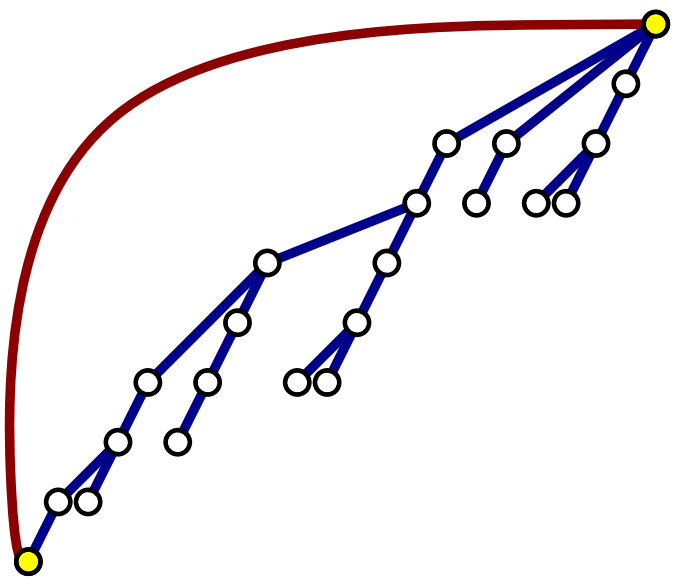
- Place root + matching at the top

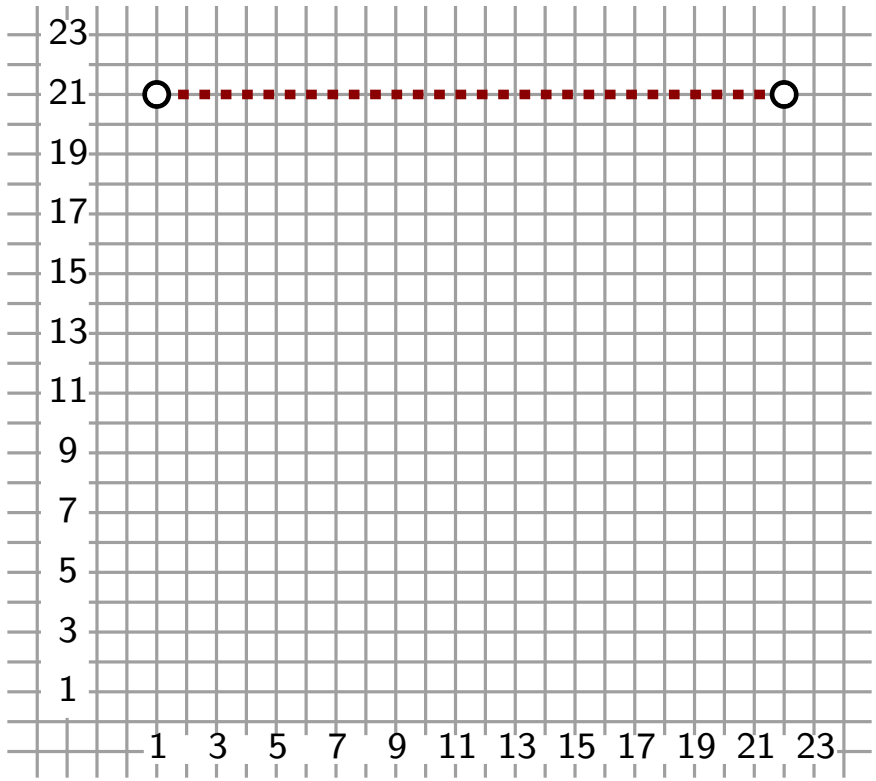




Tree $\times$ Matching

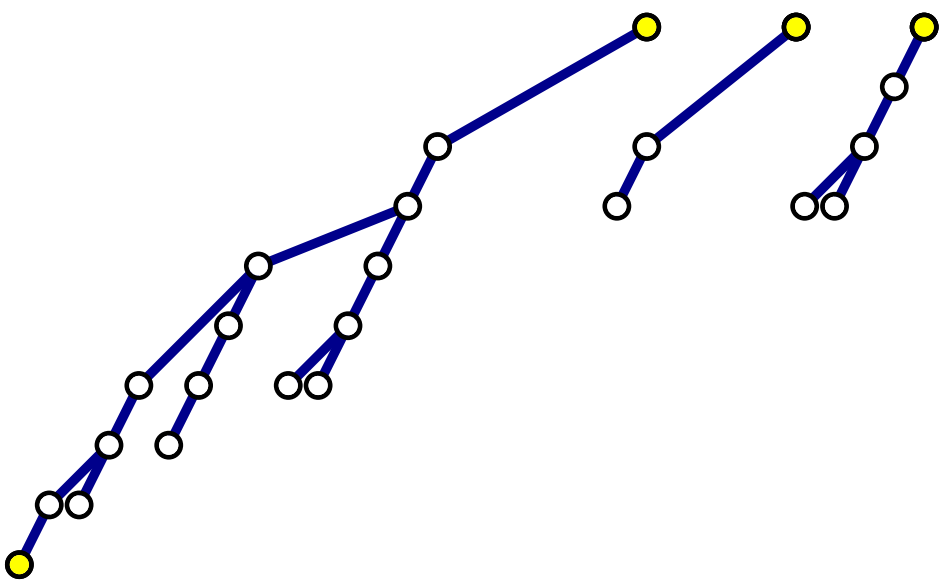
- Place root + matching at the top
- Split the tree

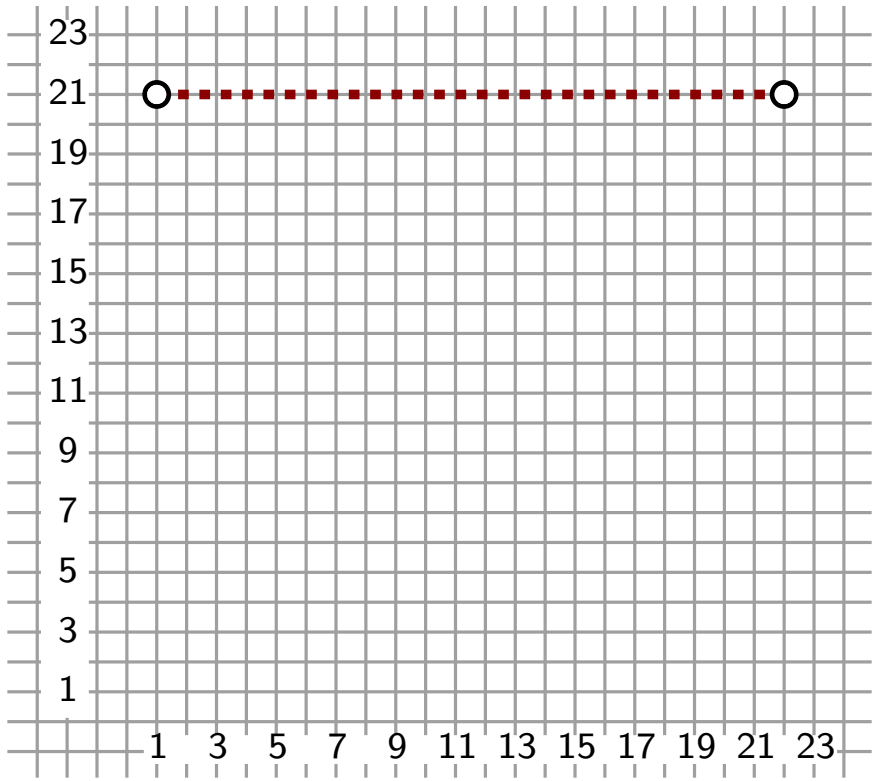




Tree $\times$ Matching

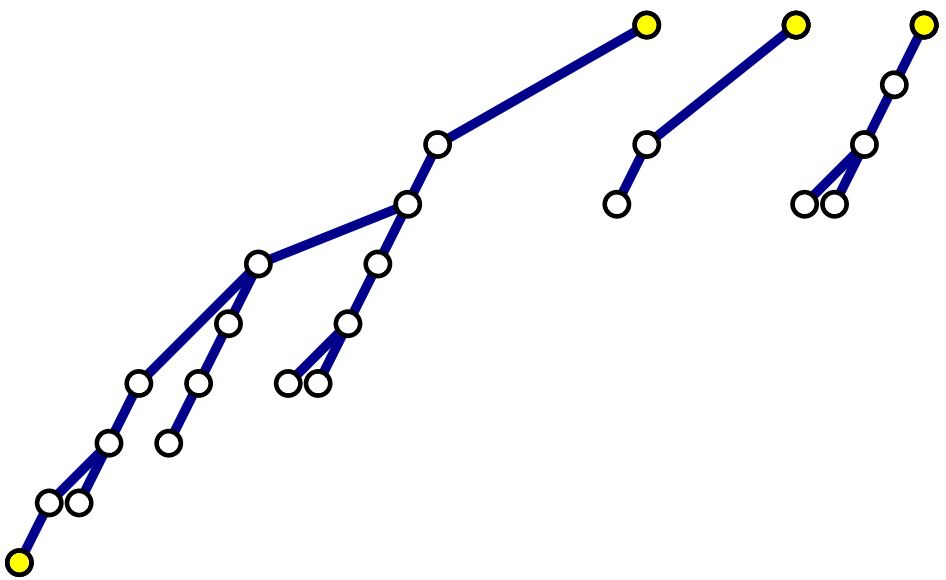
- Place root + matching at the top
- Split the tree

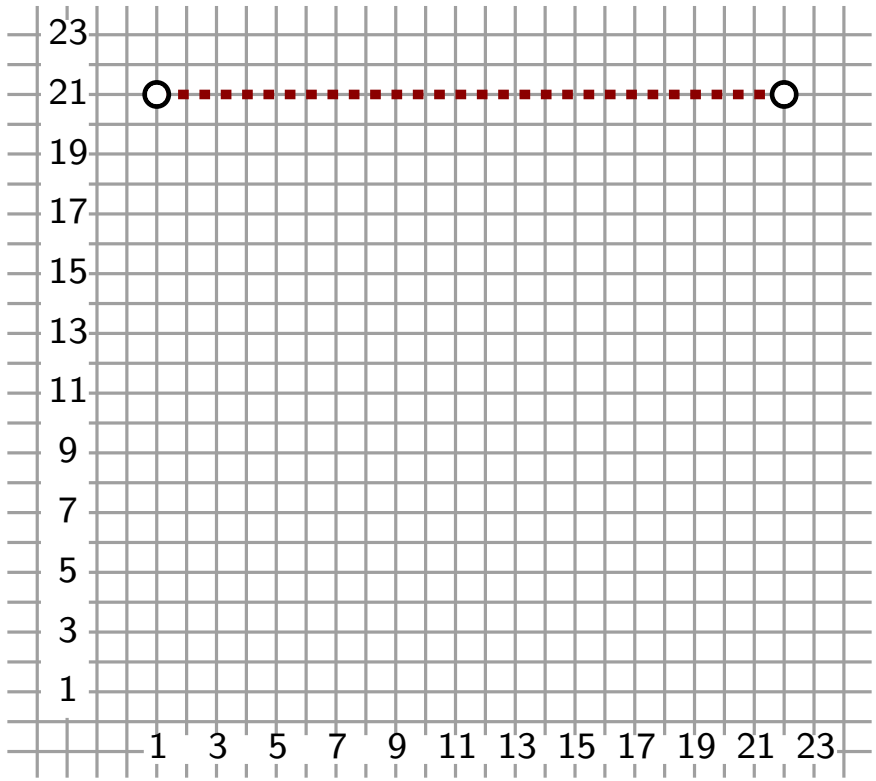




Tree $\times$ Matching

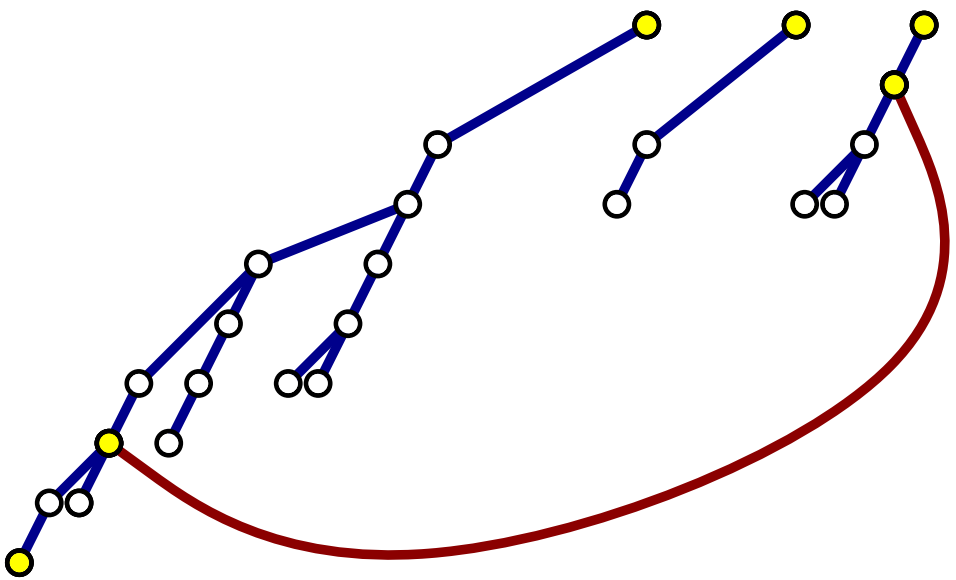
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top

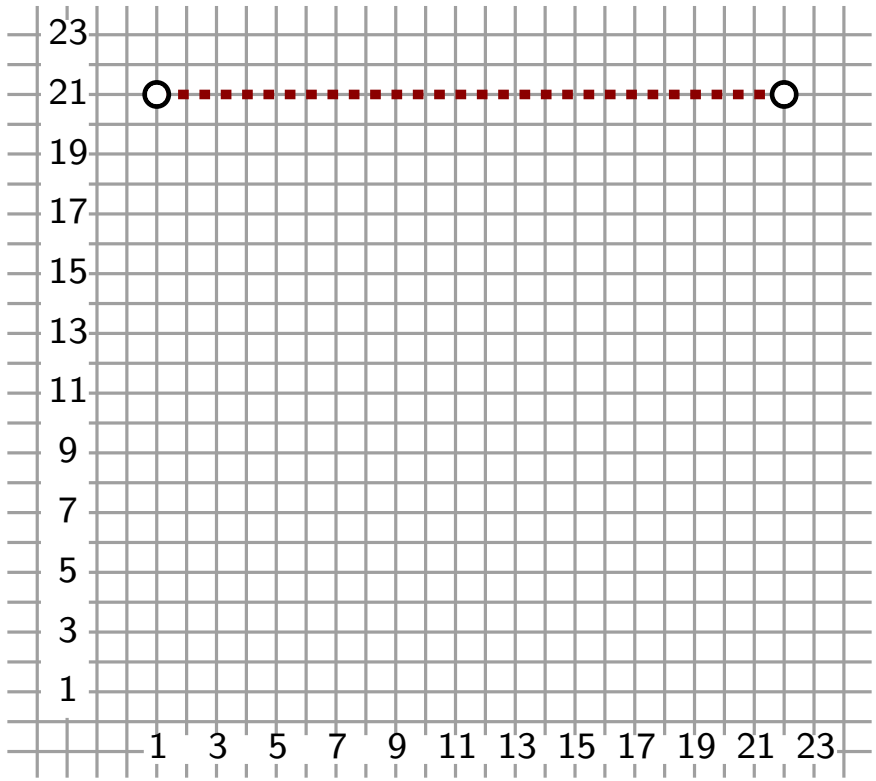




Tree × Matching

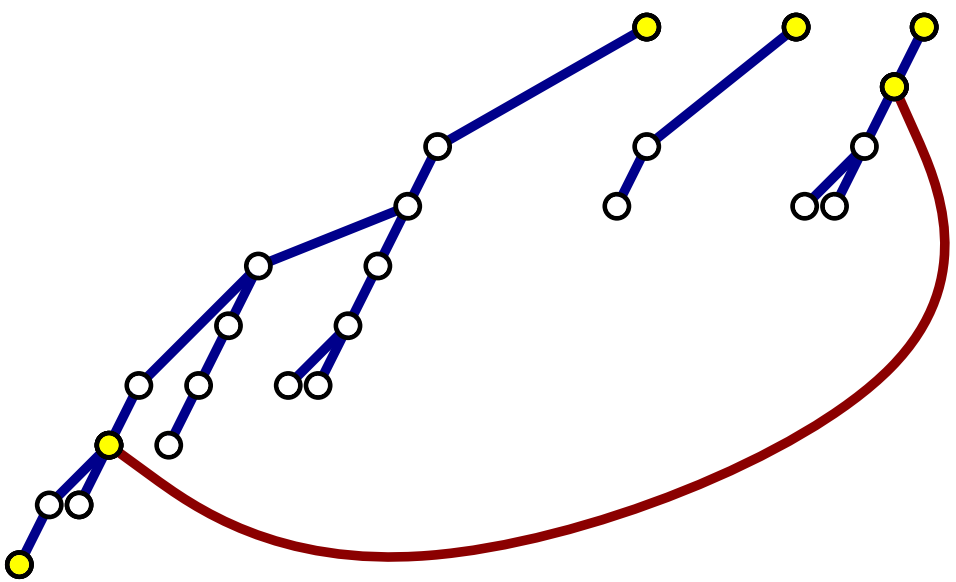
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top

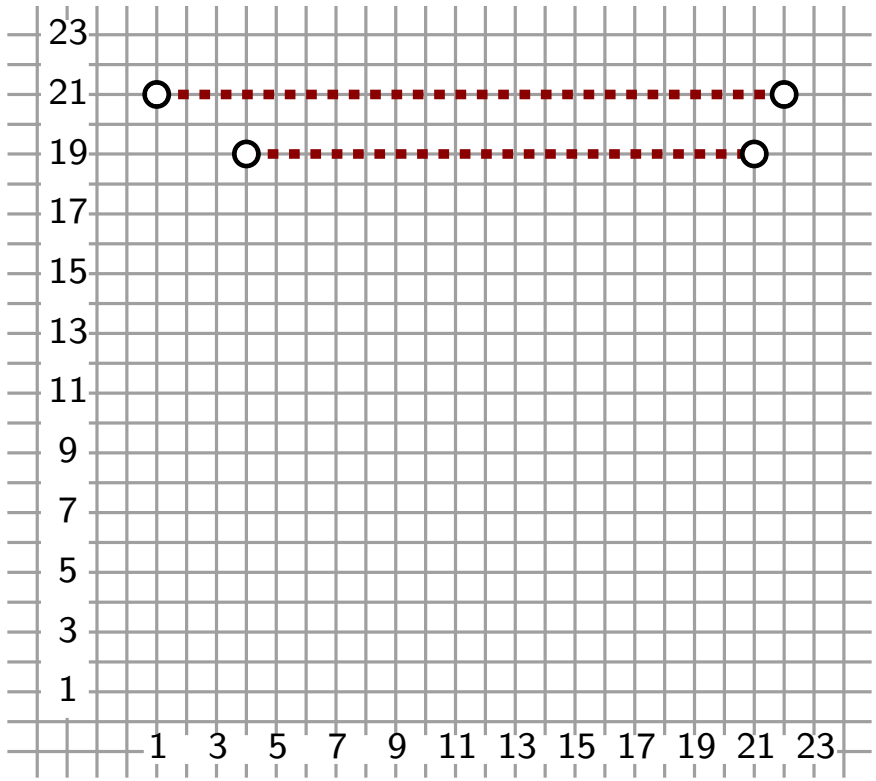




Tree × Matching

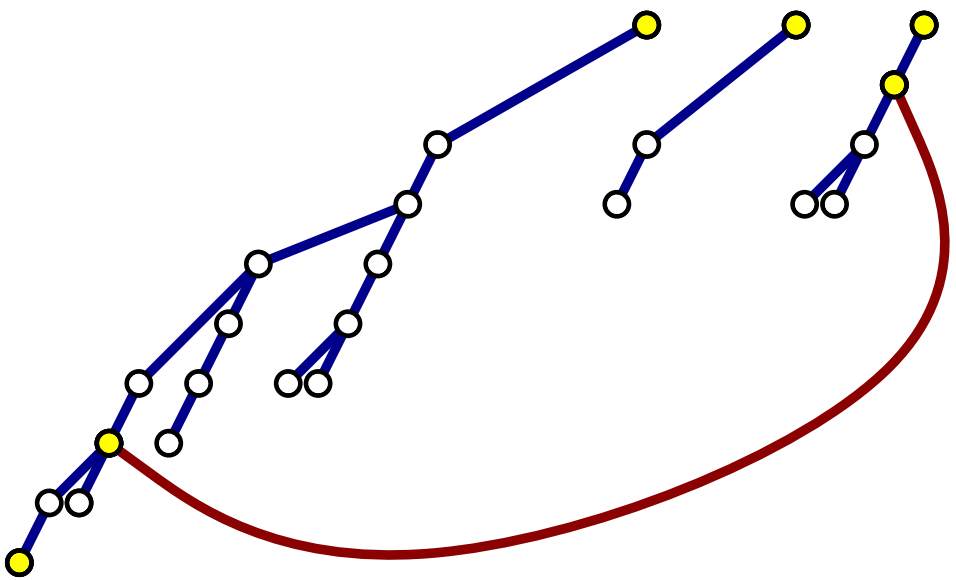
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top

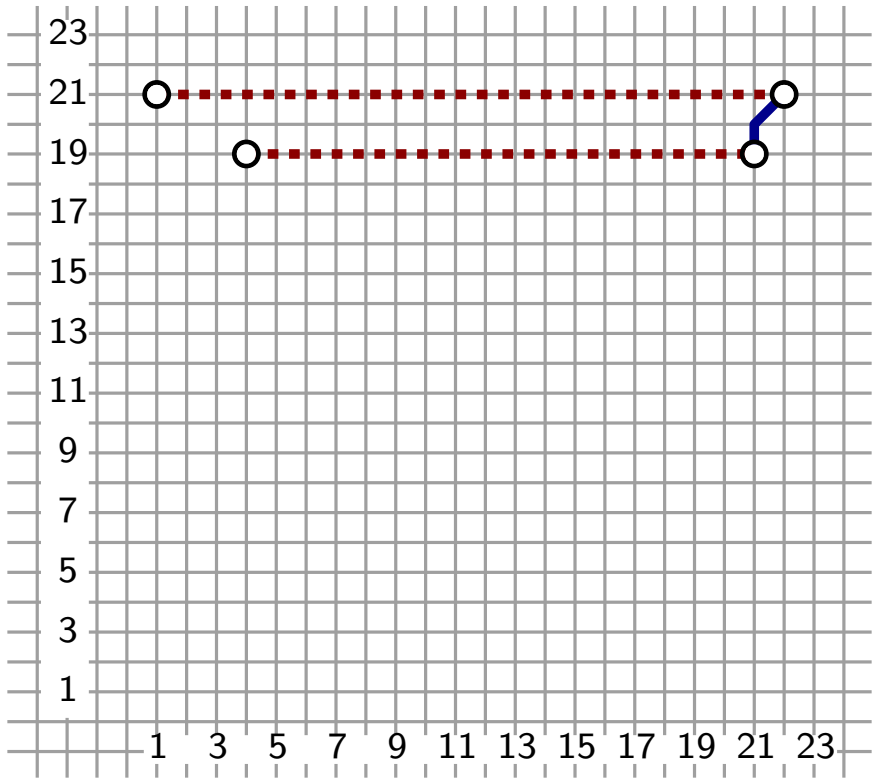




Tree × Matching

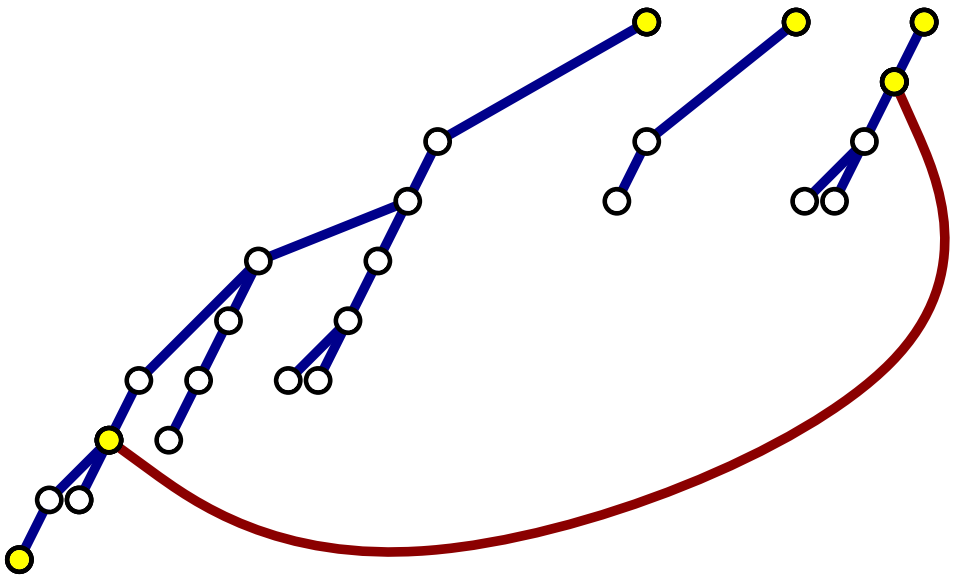
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top

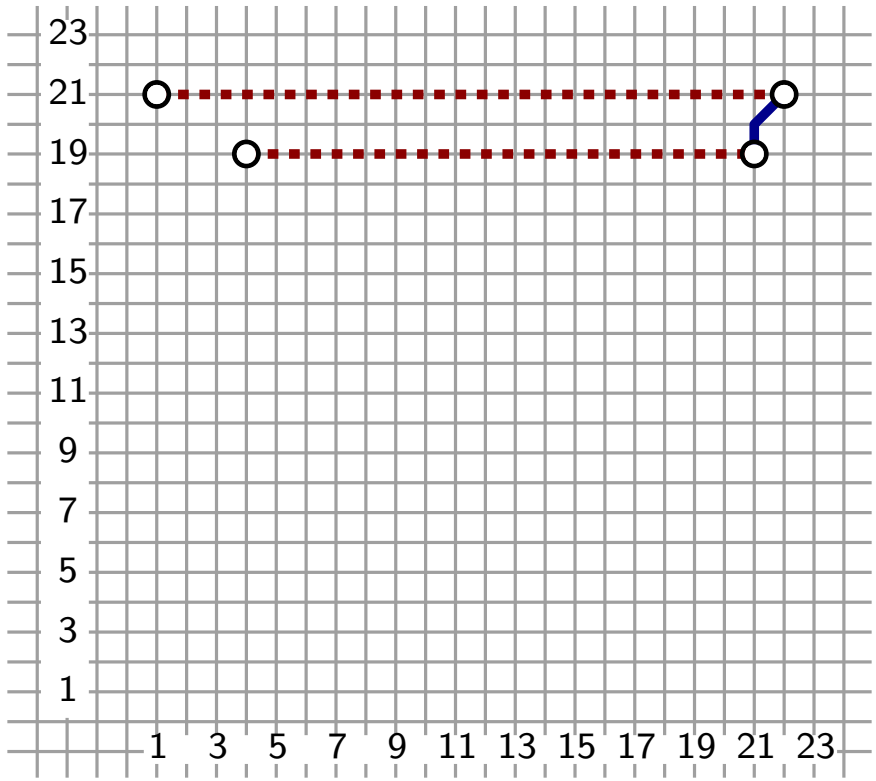




Tree × Matching

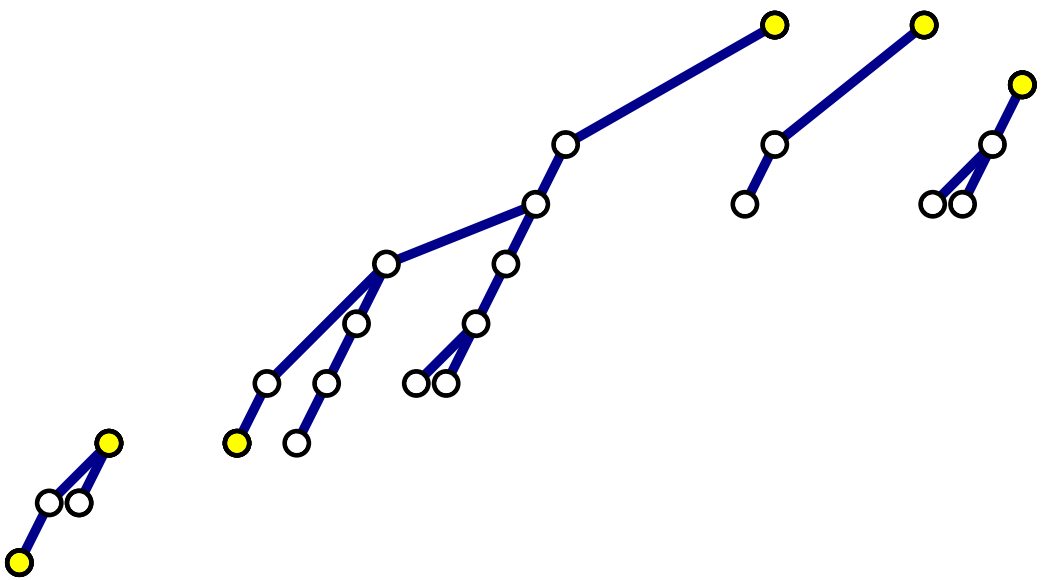
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top

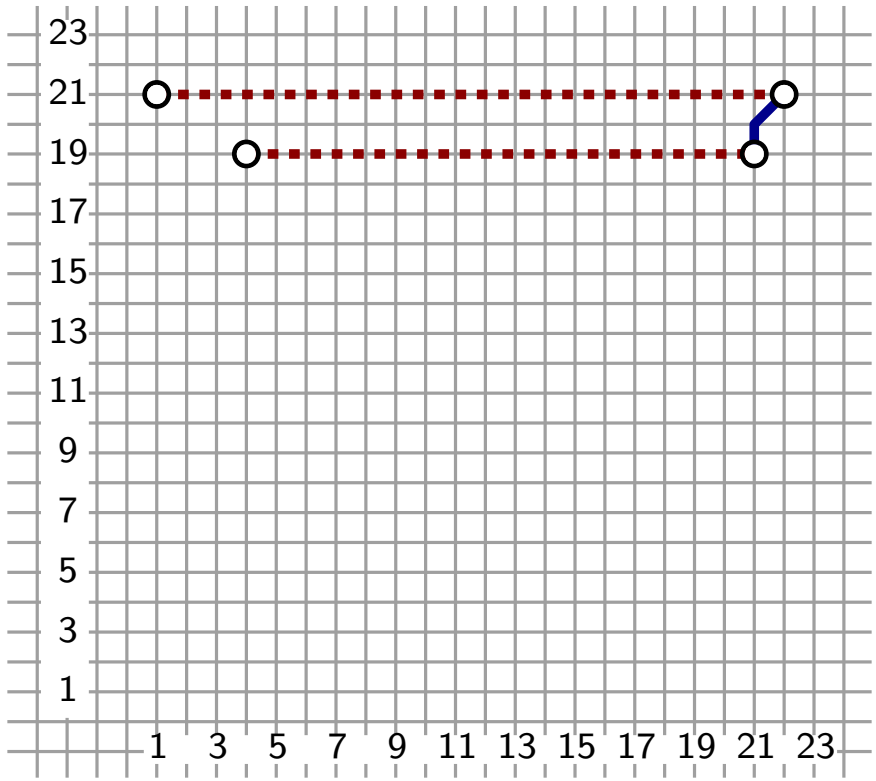




Tree × Matching

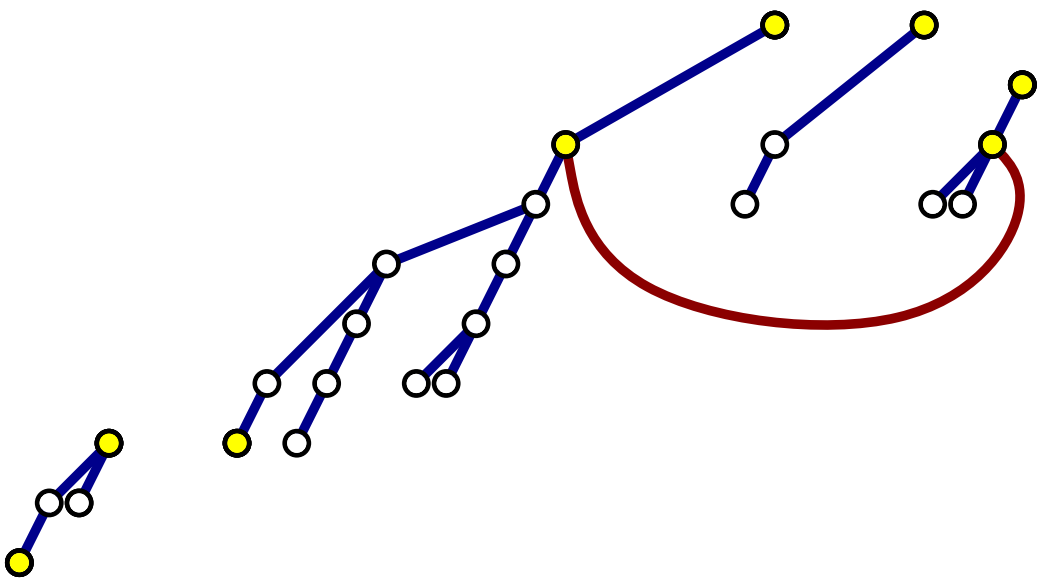
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top

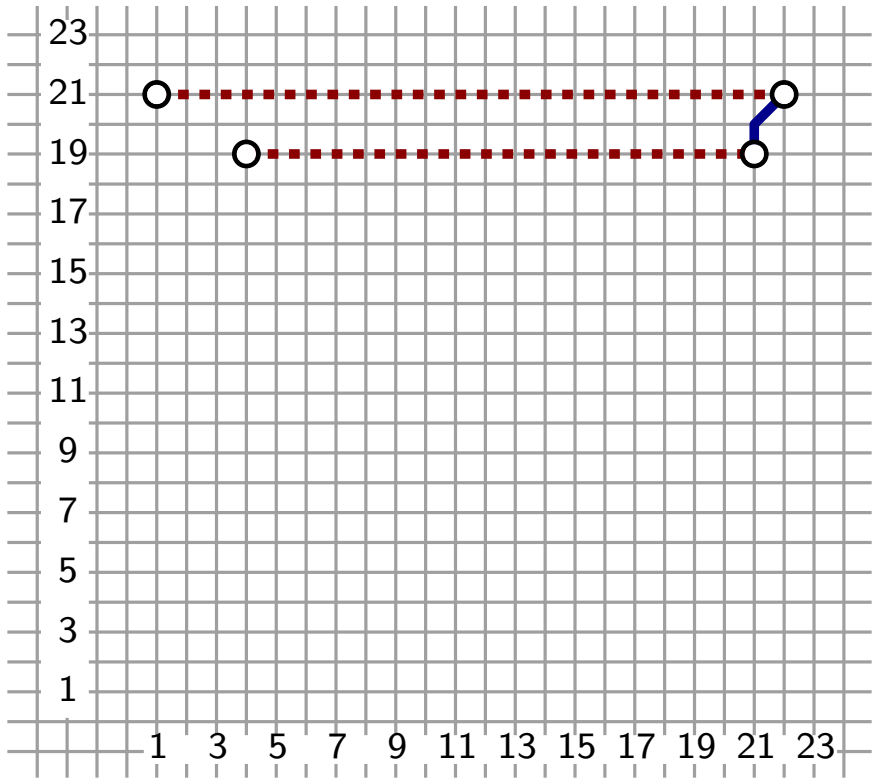




Tree × Matching

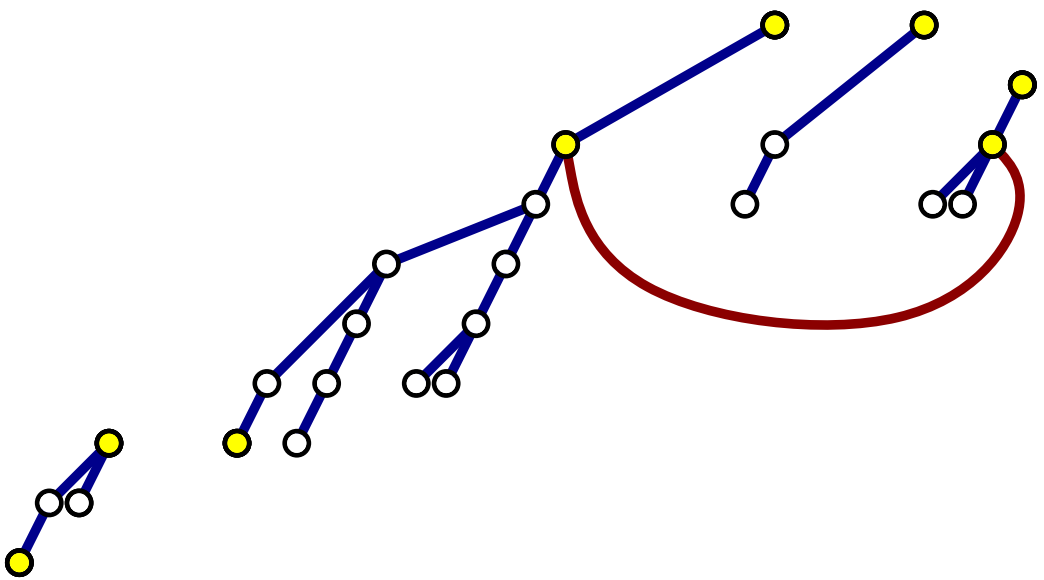
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top

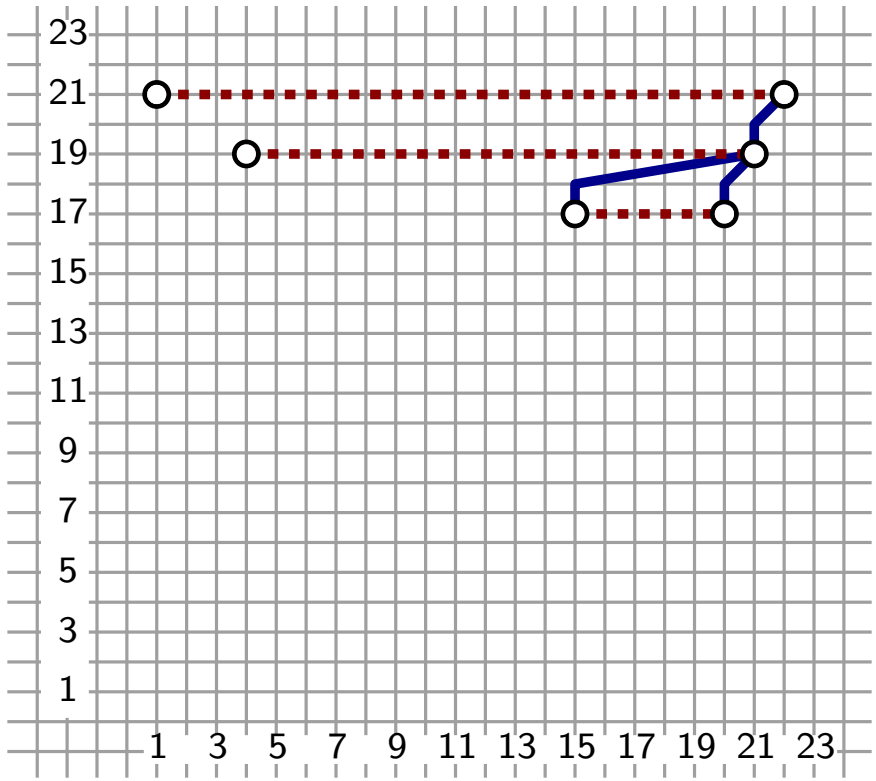




Tree × Matching

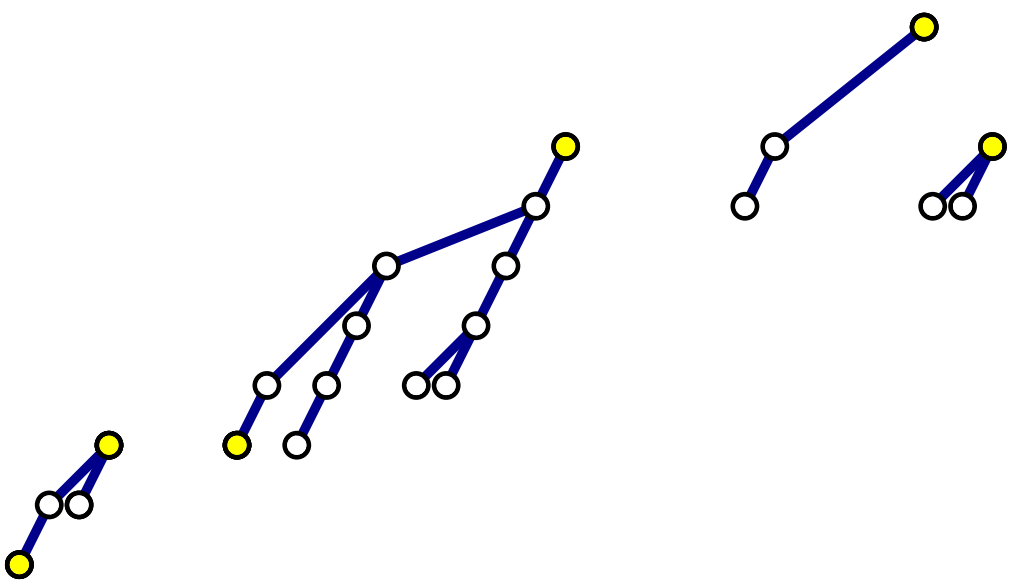
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top

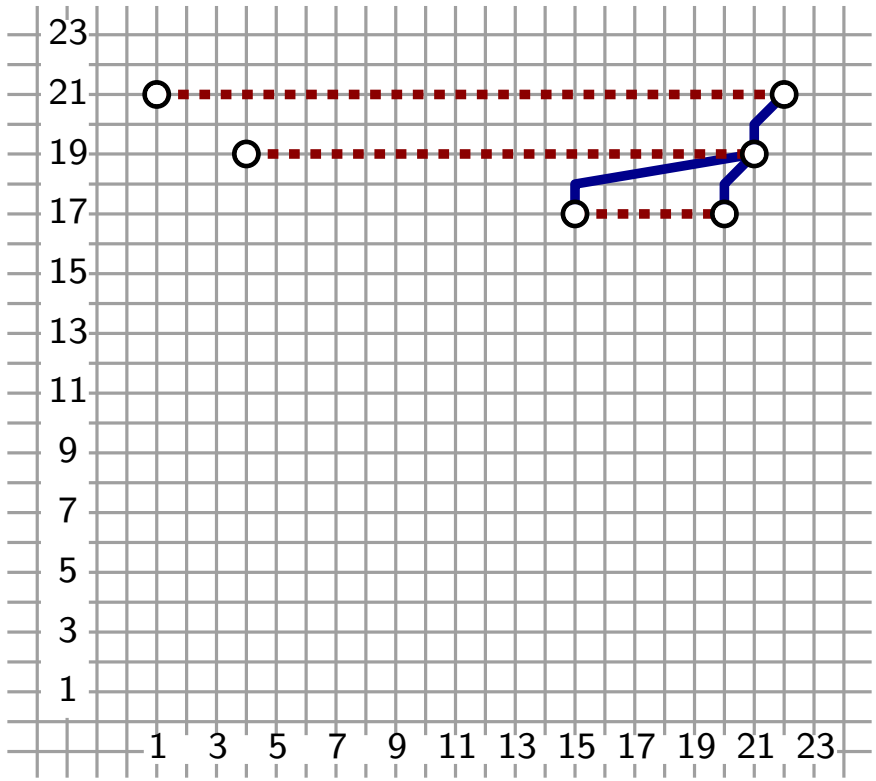




Tree × Matching

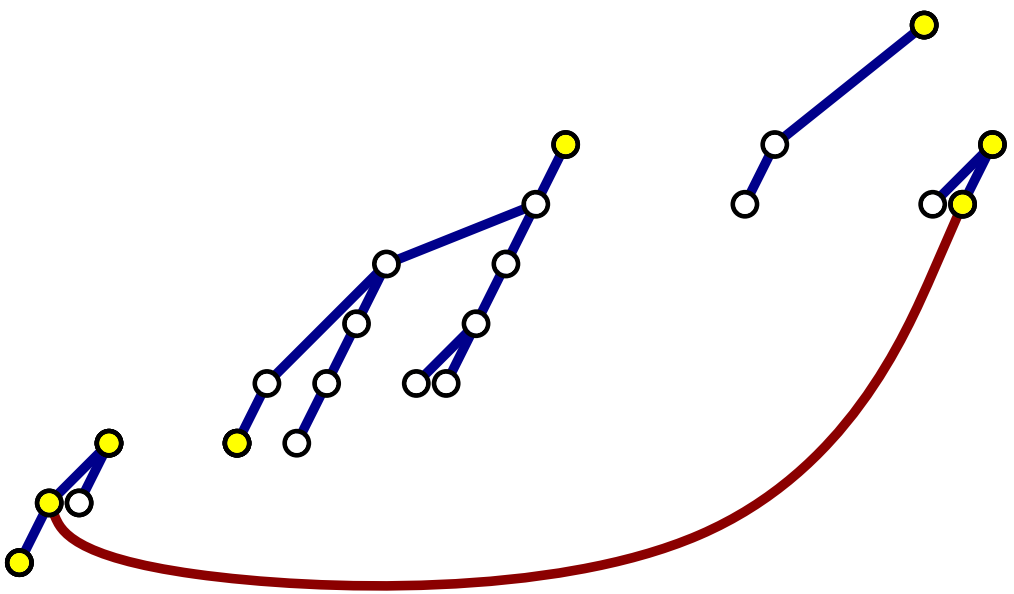
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top

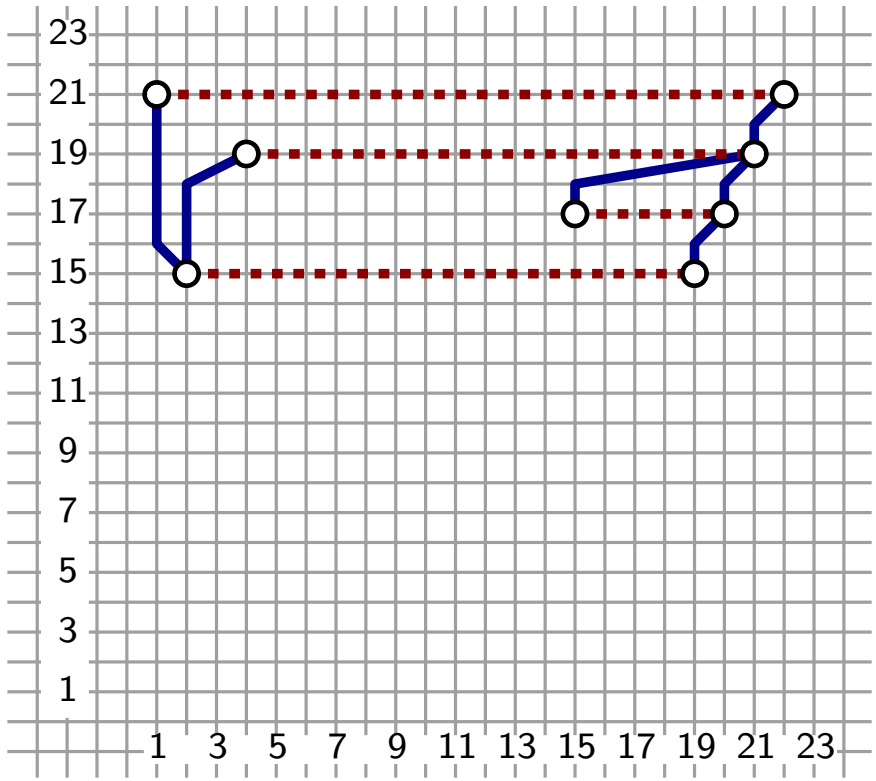




Tree × Matching

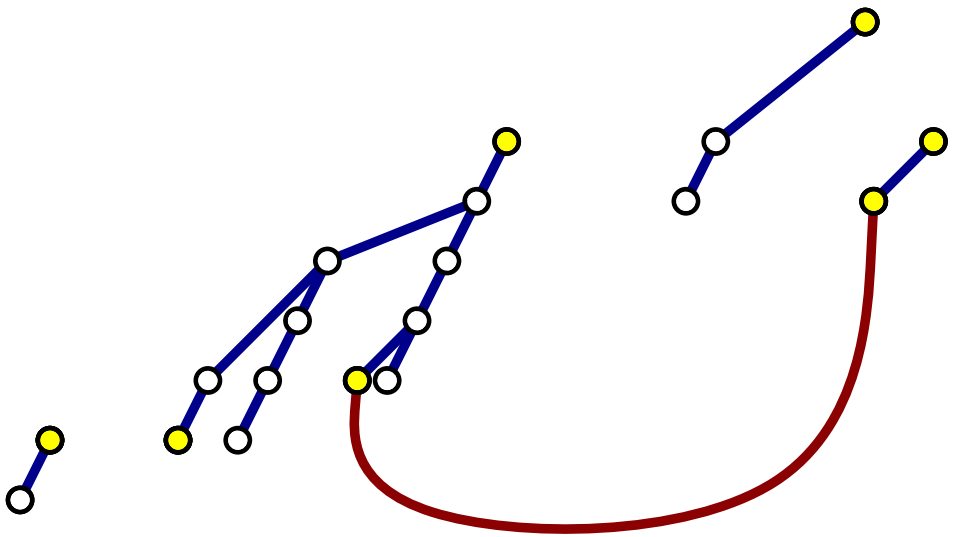
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top

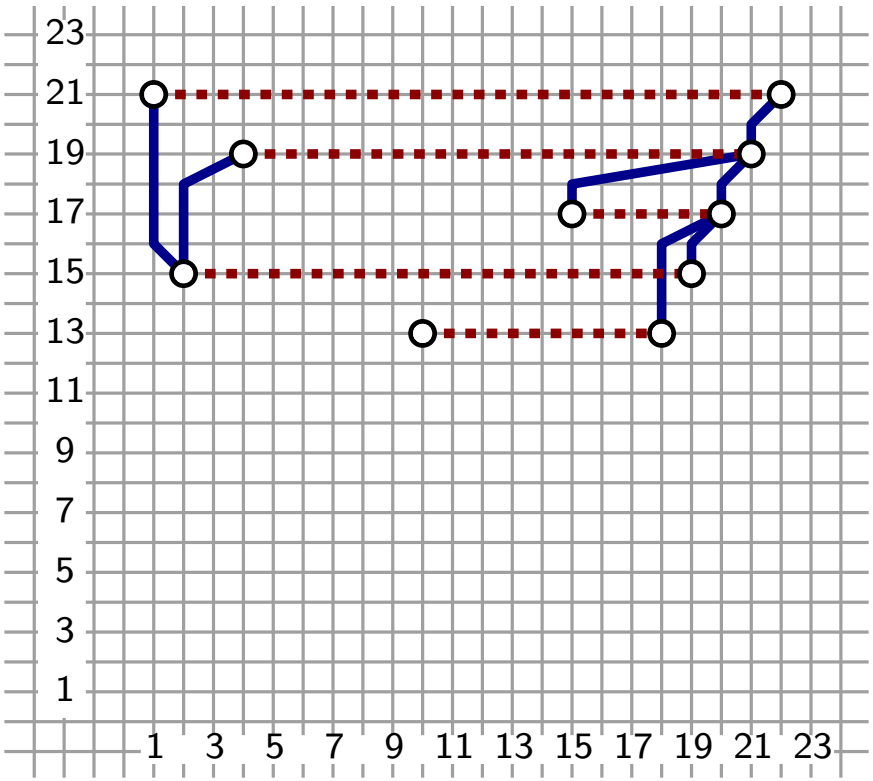




Tree × Matching

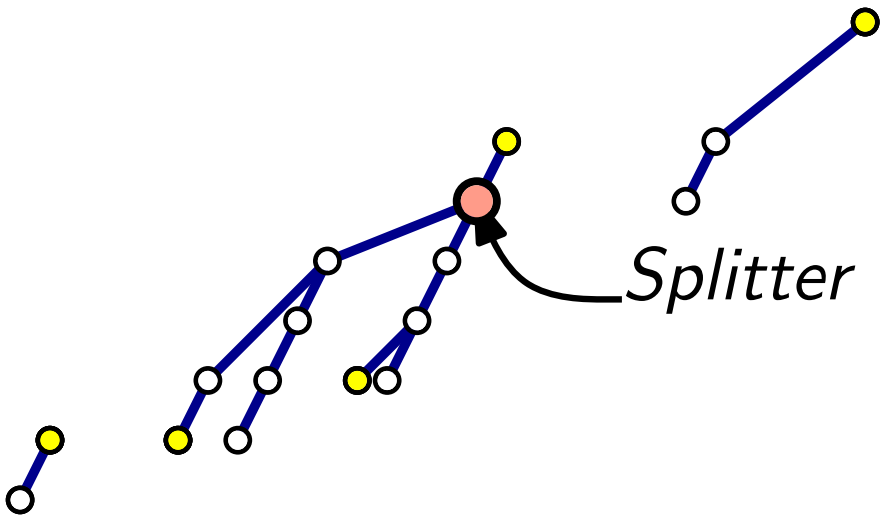
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top

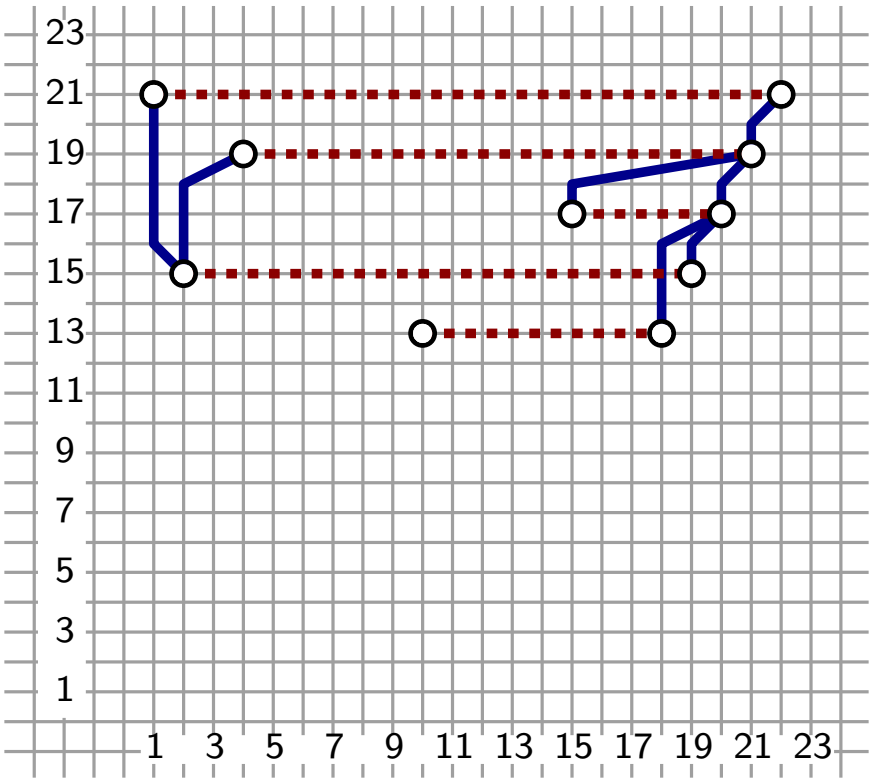




Tree × Matching

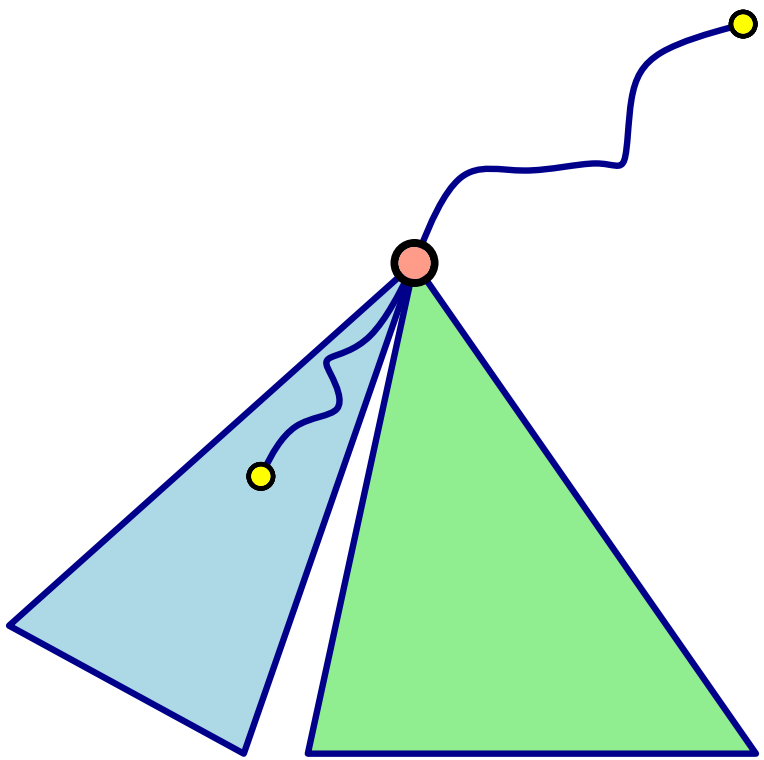
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top





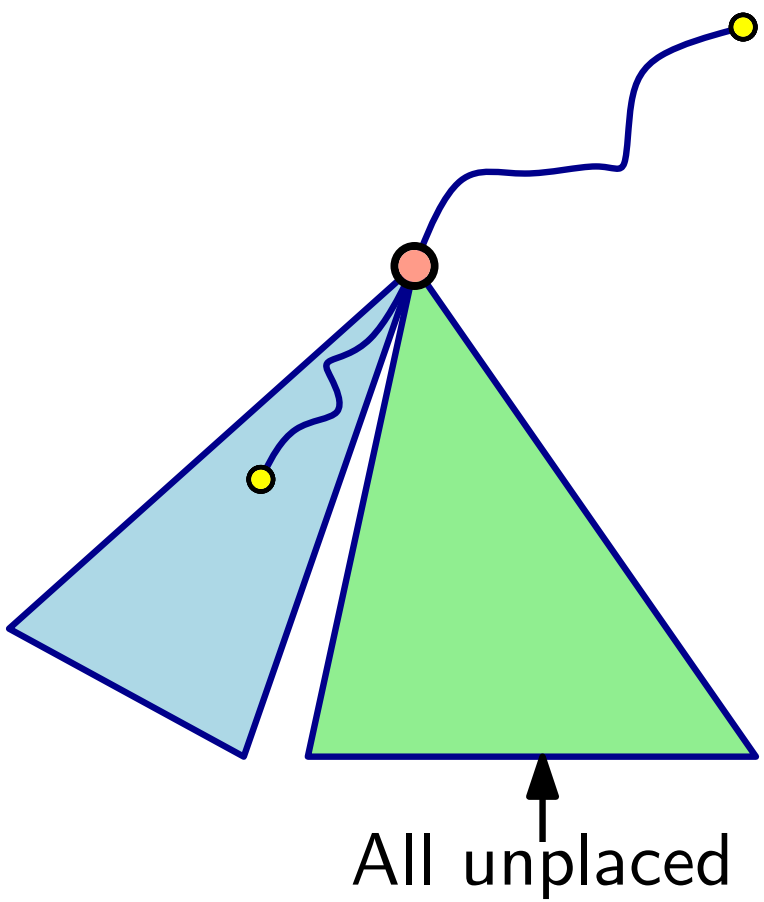
Tree $\times$ Matching

- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top



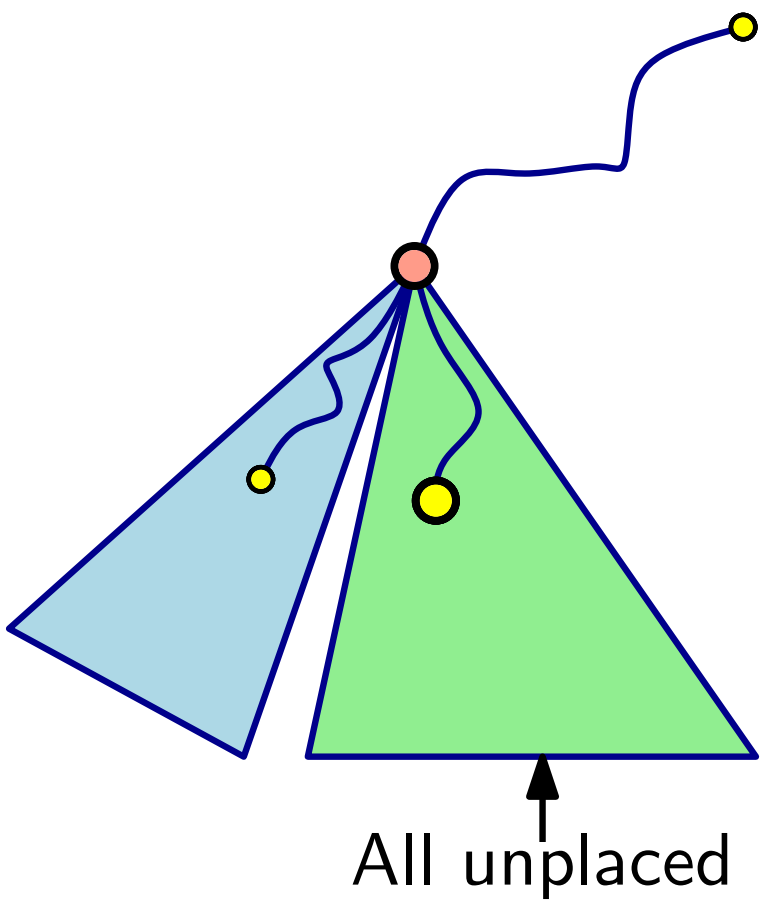
Tree $\times$ Matching

- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top



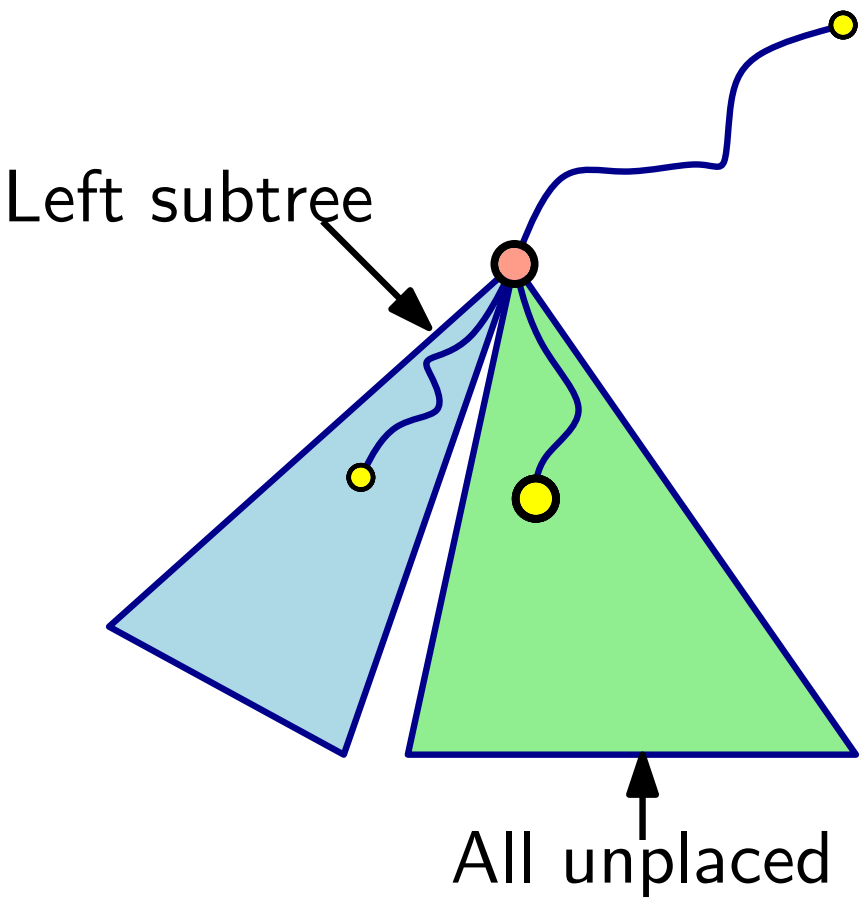
Tree $\times$ Matching

- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top



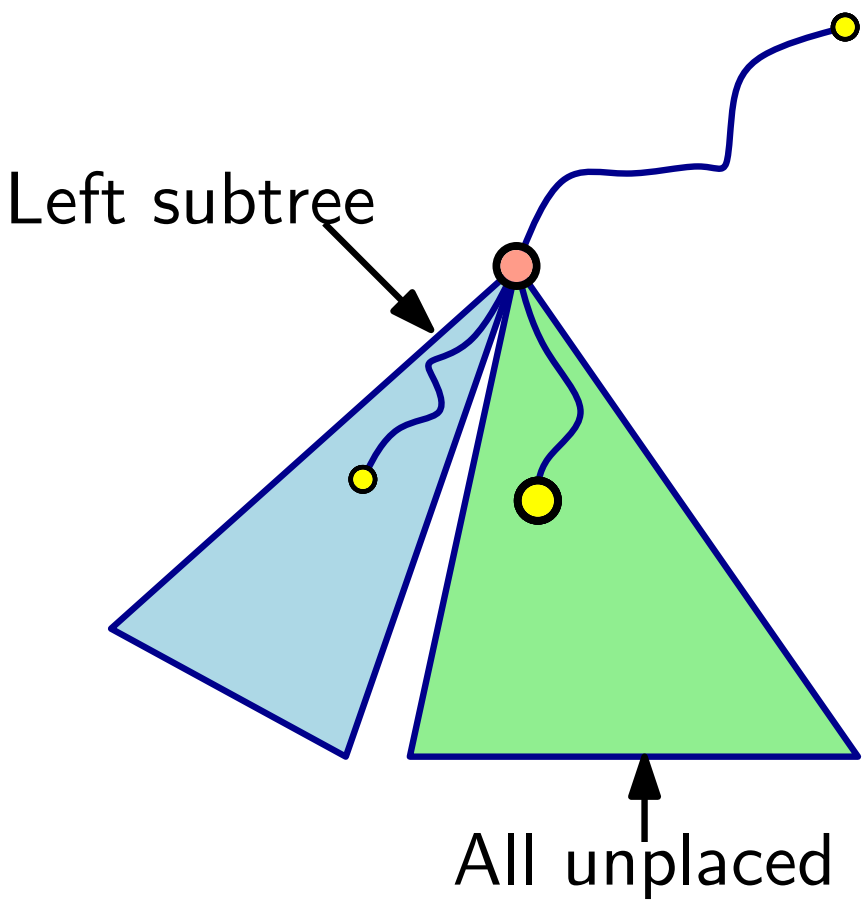
Tree × Matching

- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top



Tree × Matching

- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top

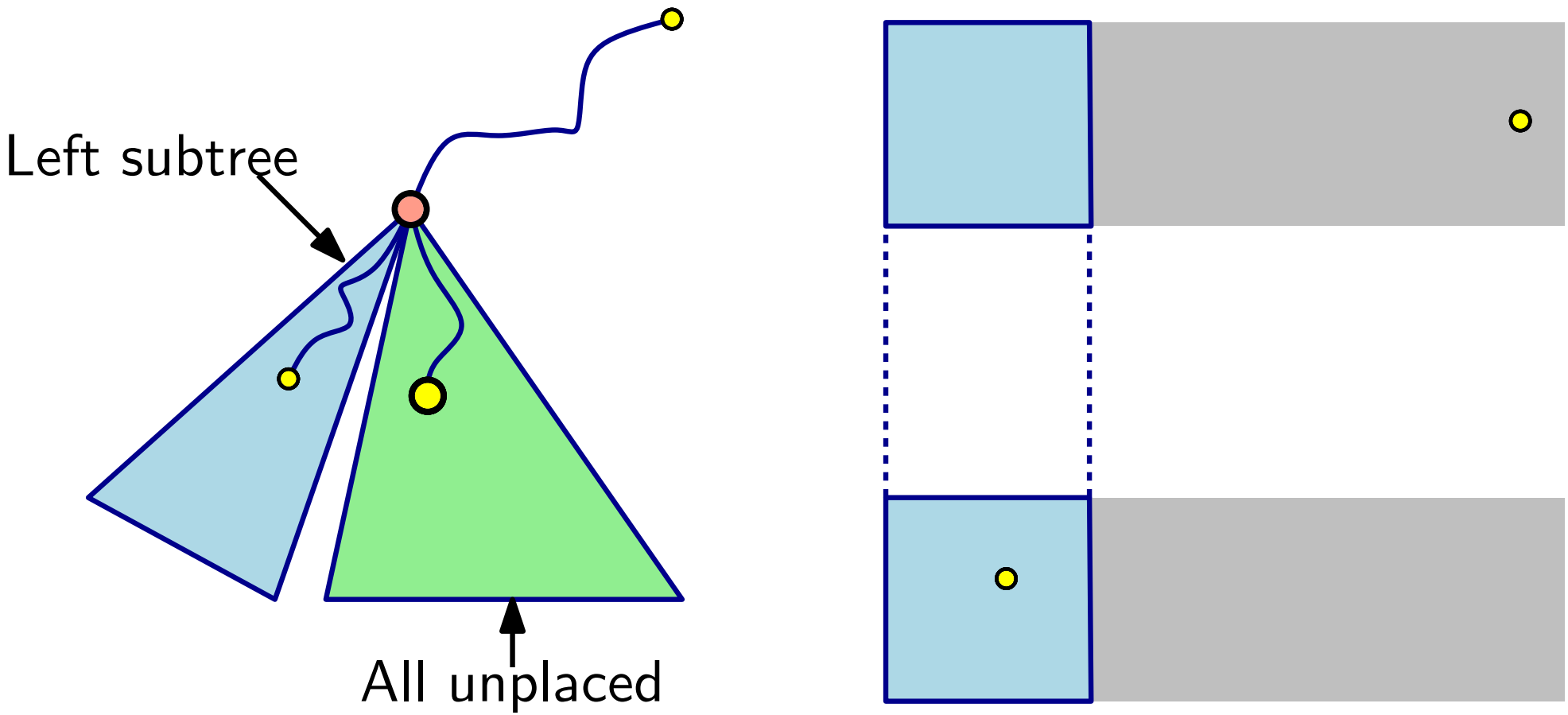


Placed vertices

Placed vertices

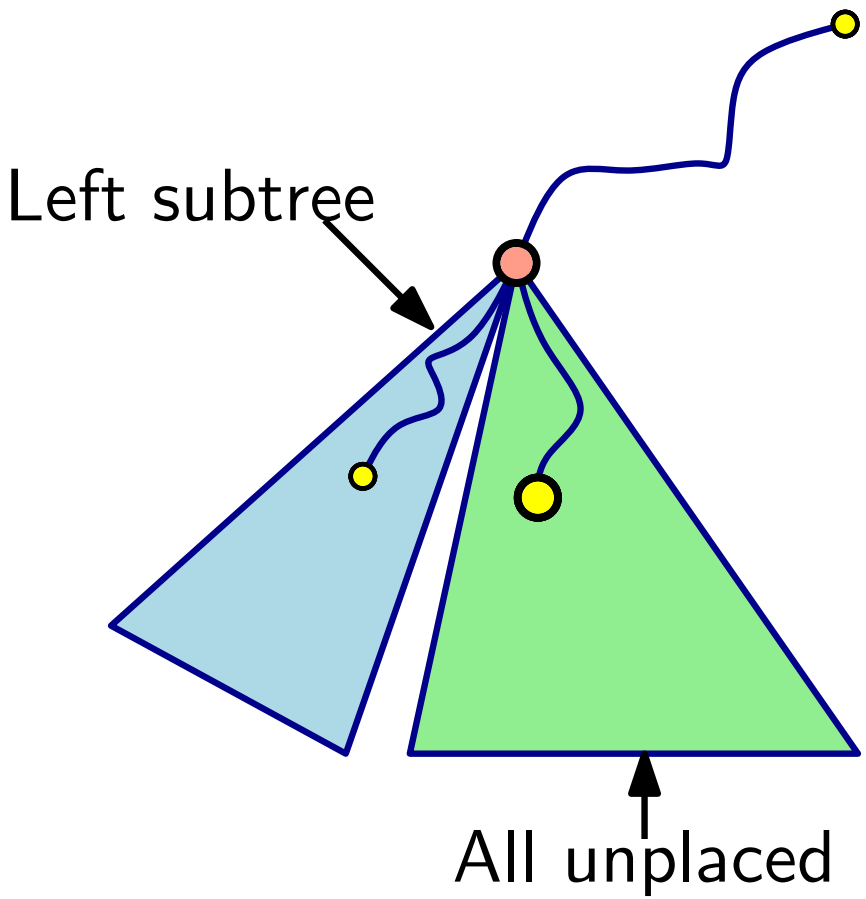
Tree × Matching

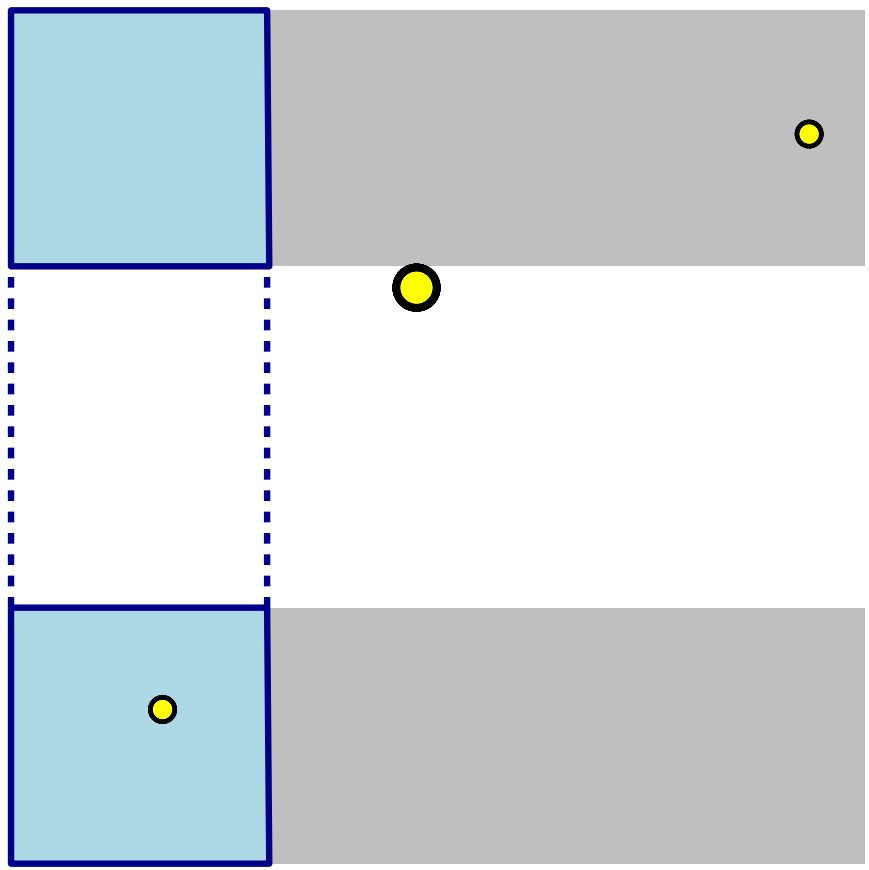
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top



Tree × Matching

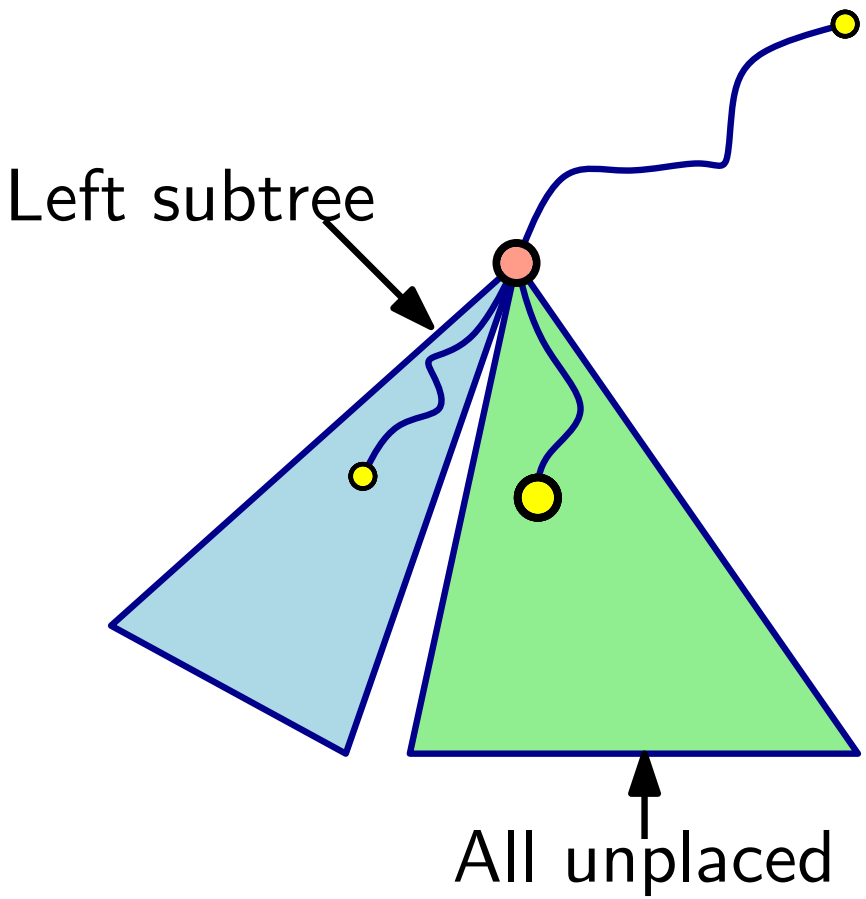
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top

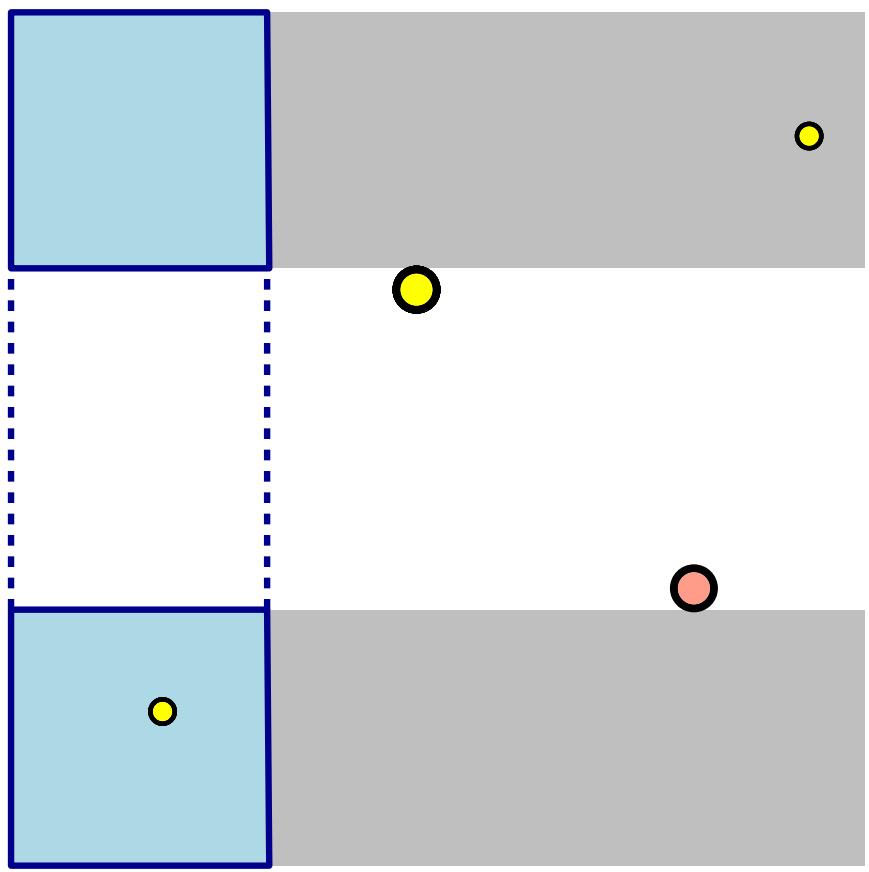




Tree × Matching

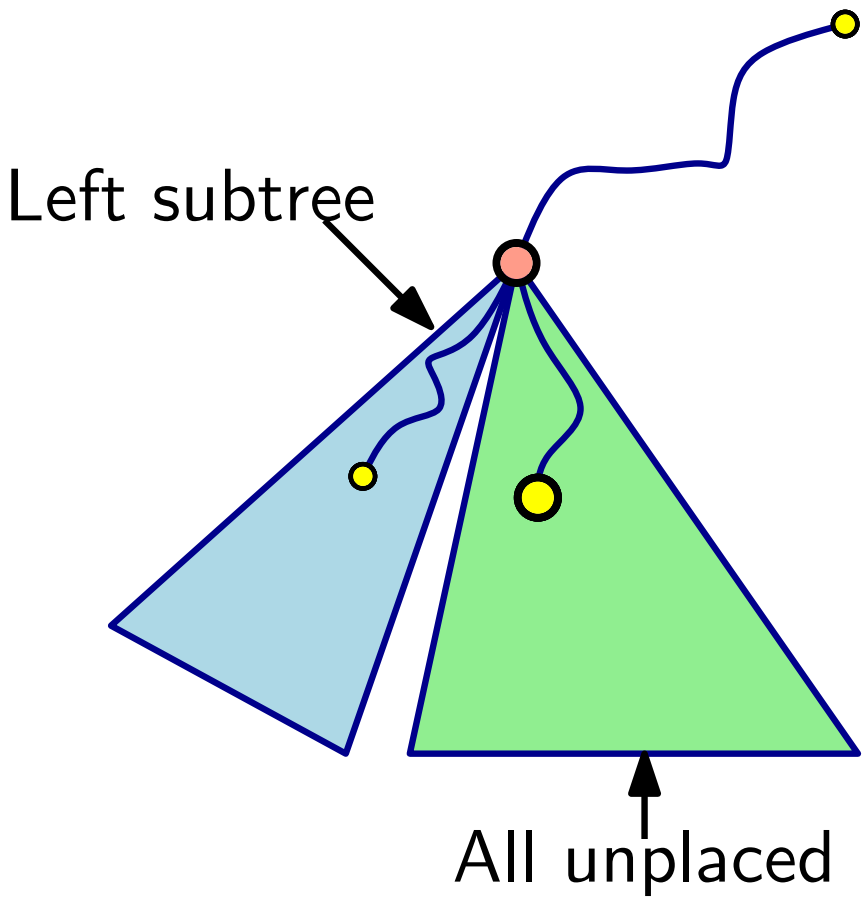
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top

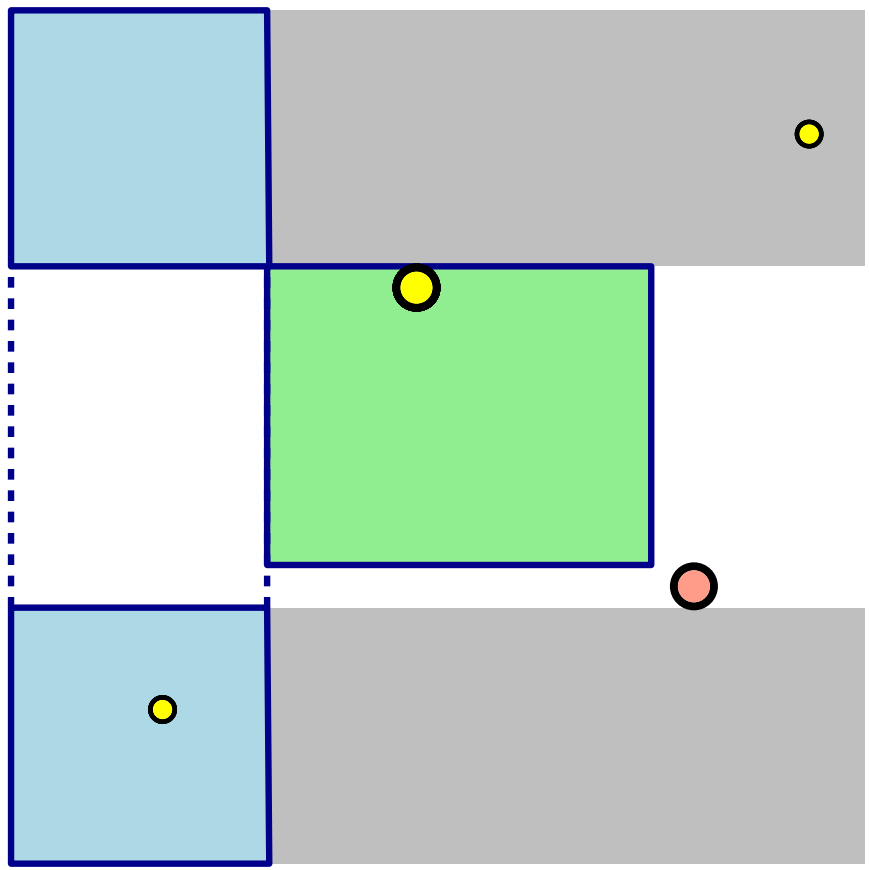




Tree × Matching

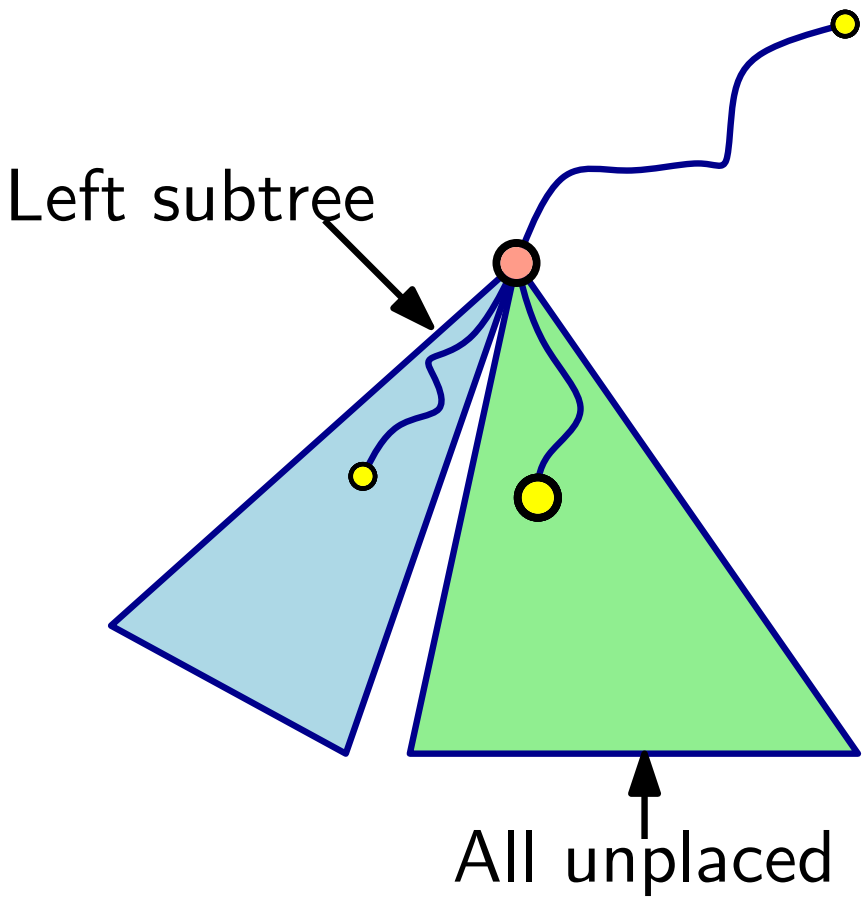
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top

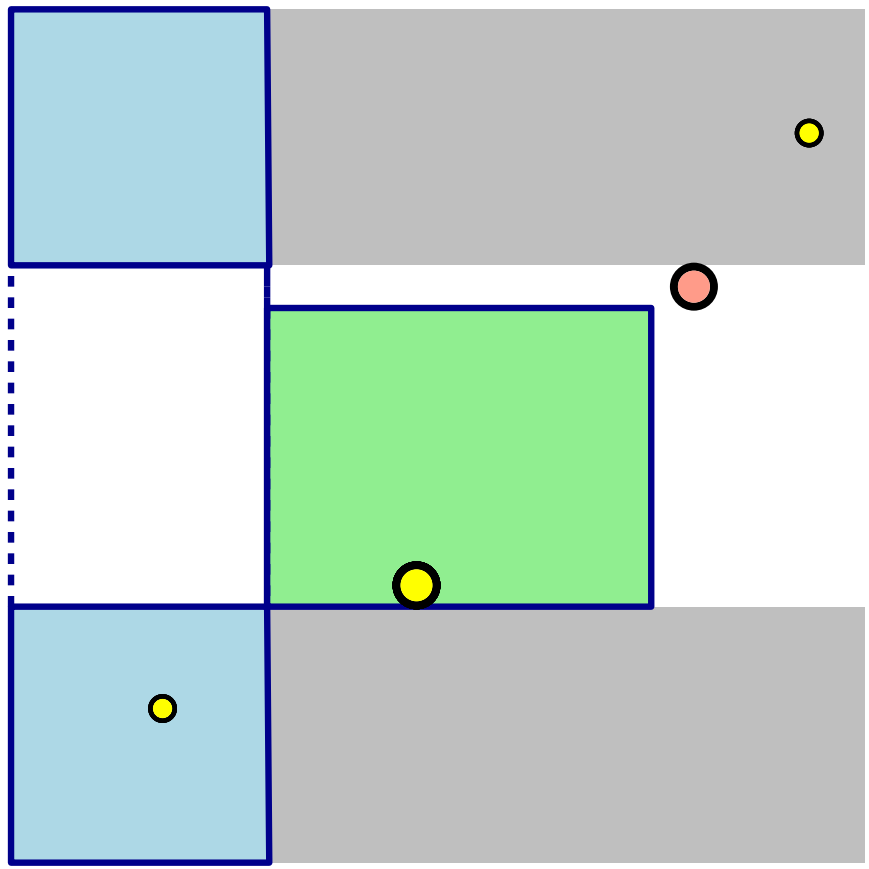




Tree × Matching

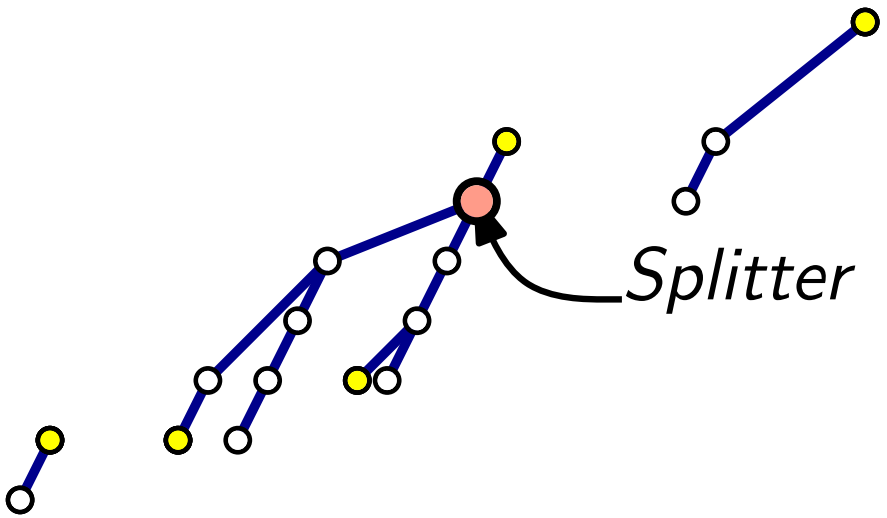
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top

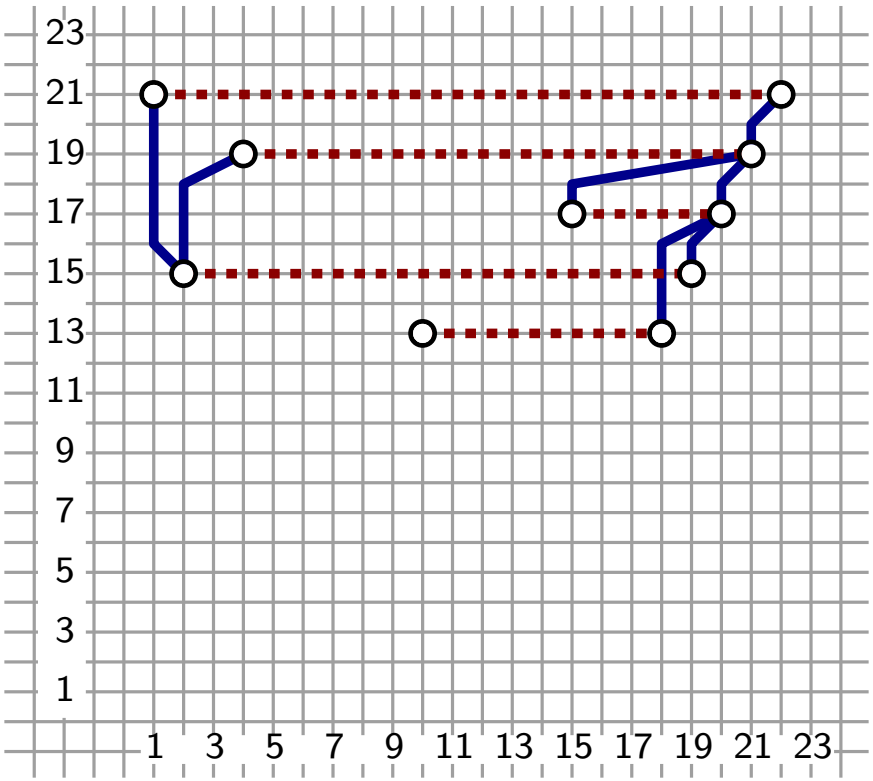




Tree × Matching

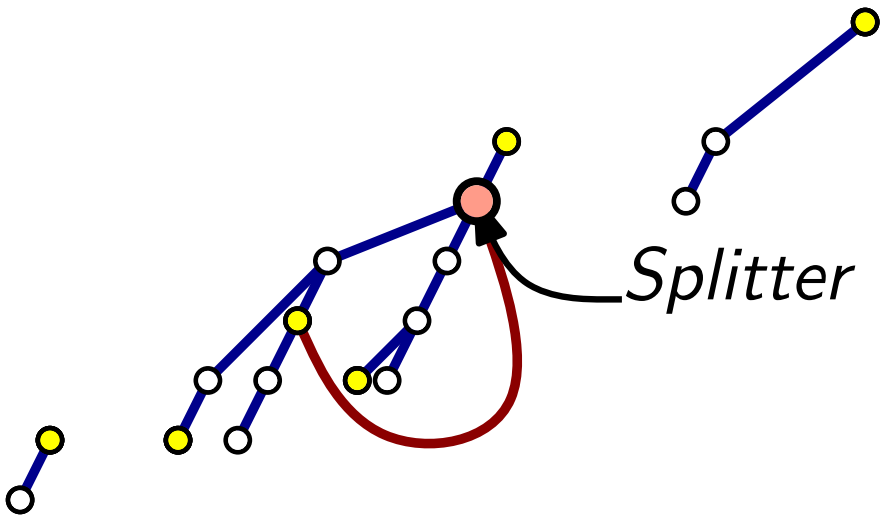
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top
- If splitter: place on opposite side

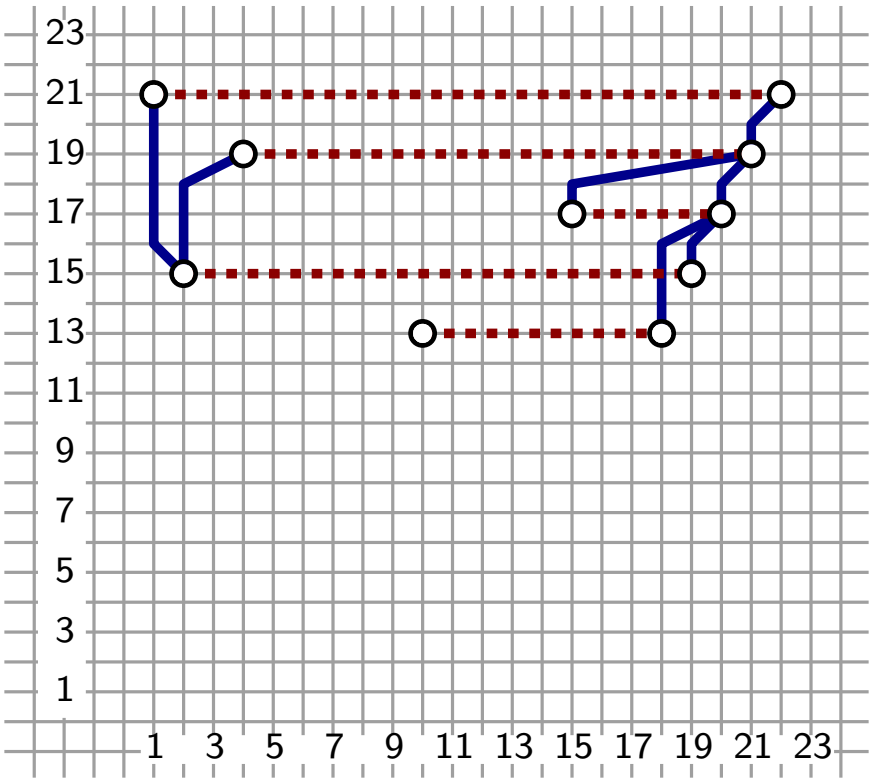




Tree × Matching

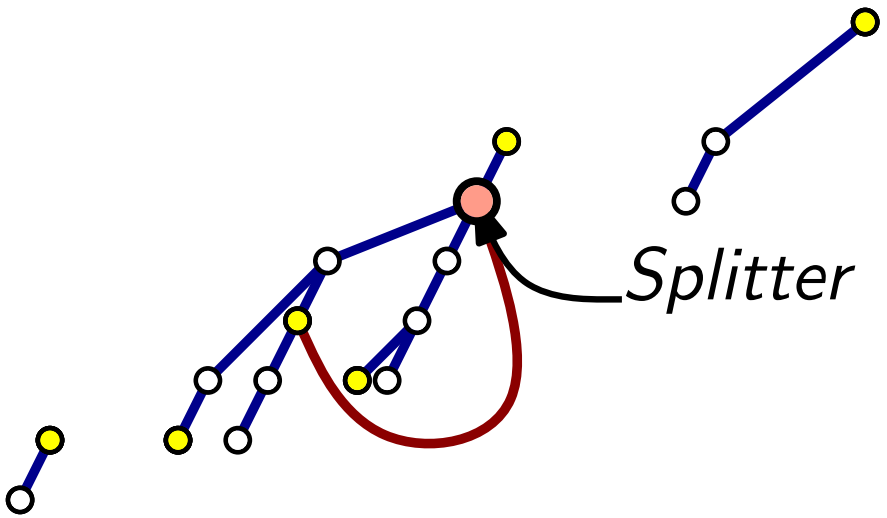
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top
- If splitter: place on opposite side

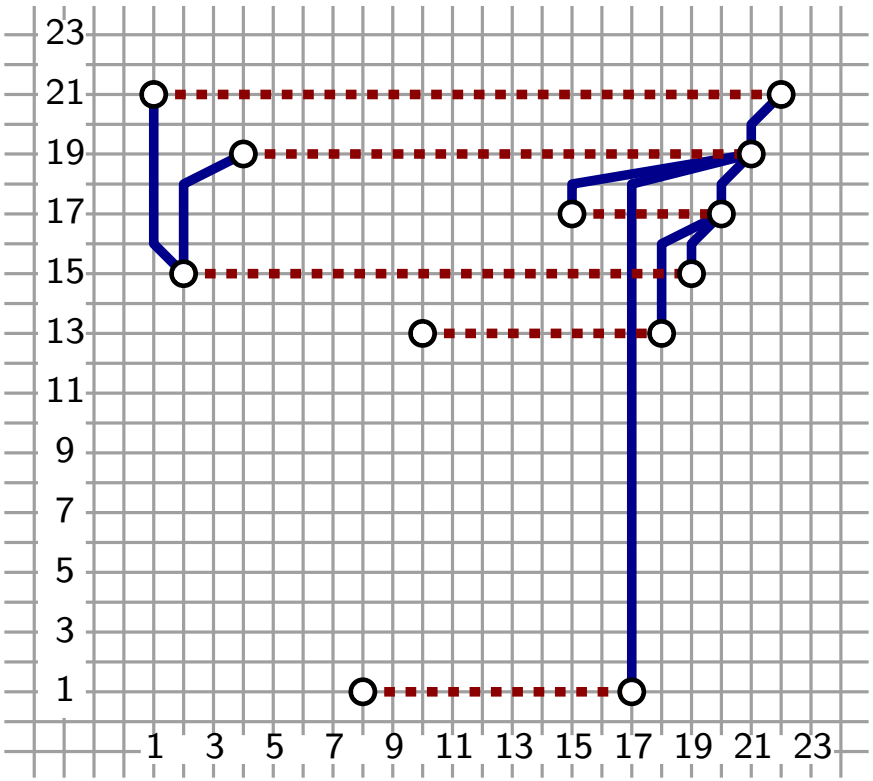




Tree × Matching

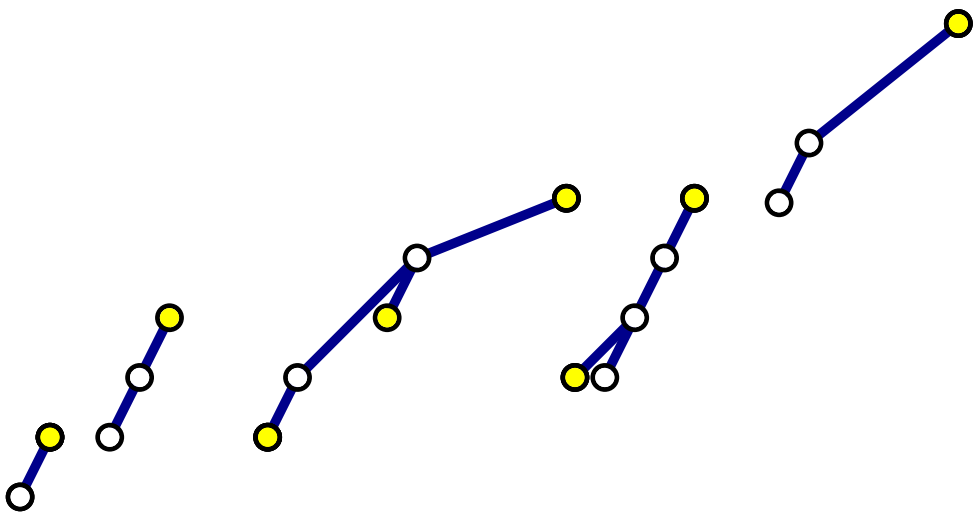
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top
- If splitter: place on opposite side

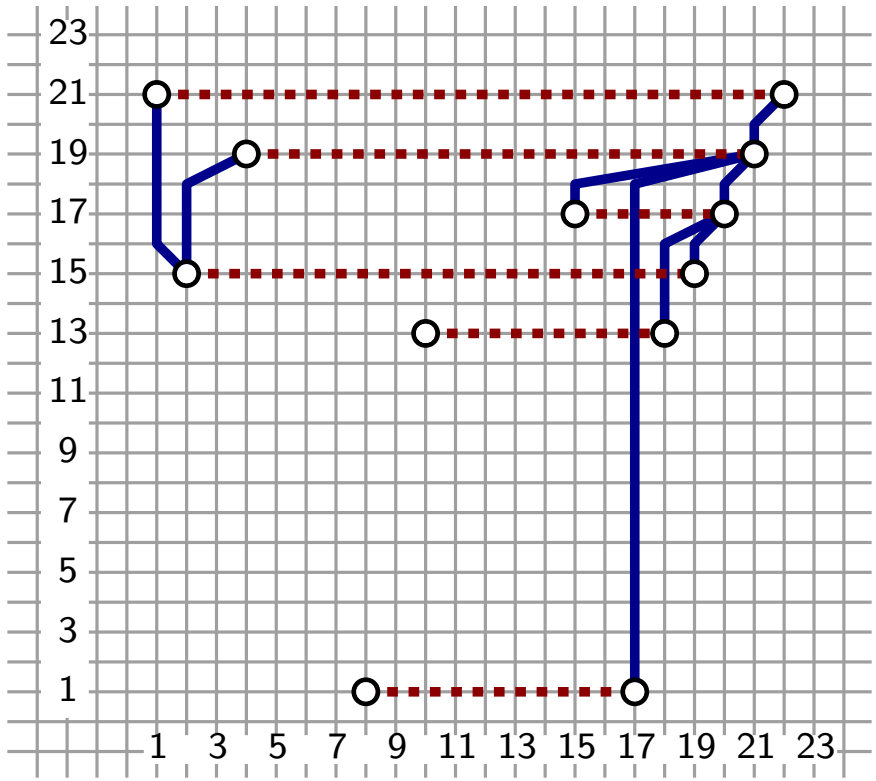




Tree × Matching

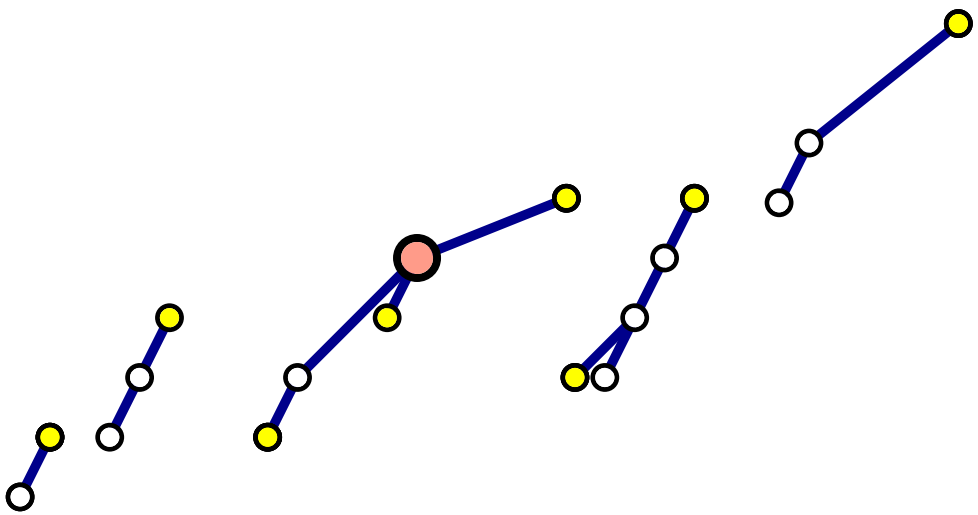
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top
- If splitter: place on opposite side

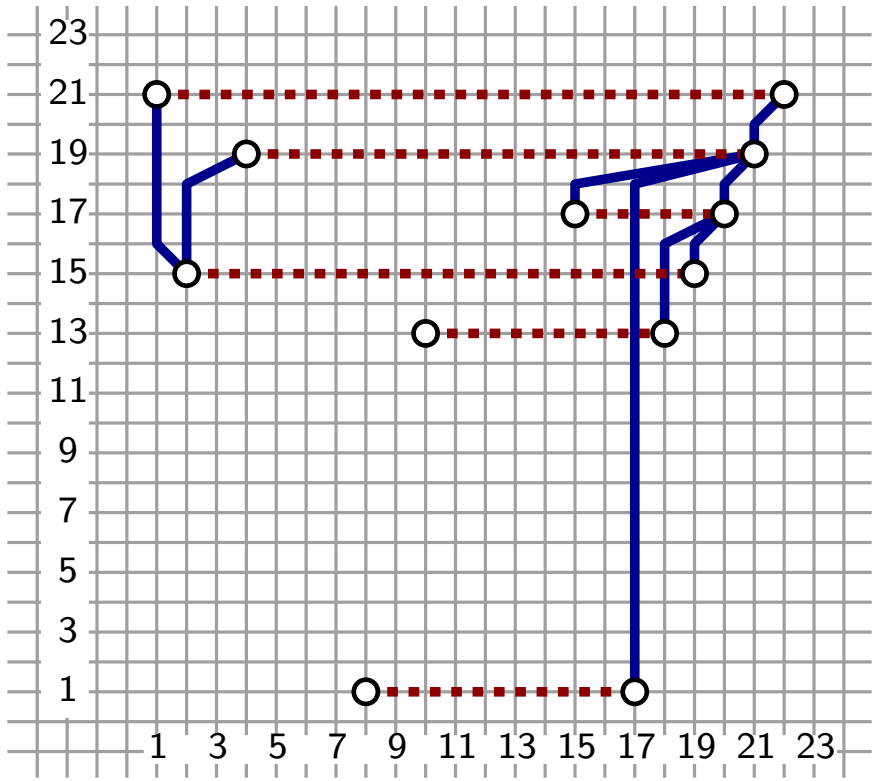




Tree × Matching

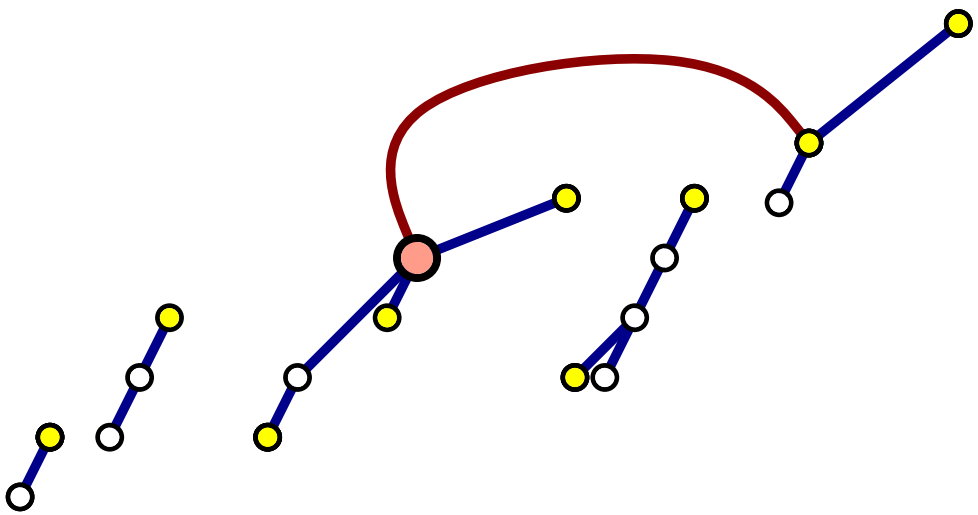
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top
- If splitter: place on opposite side

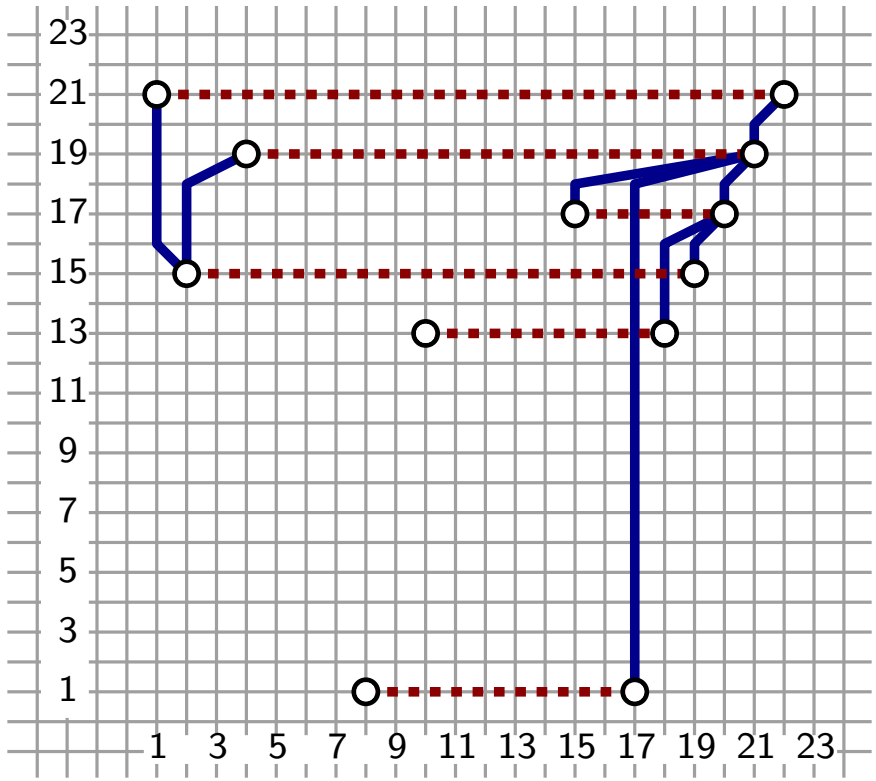




Tree × Matching

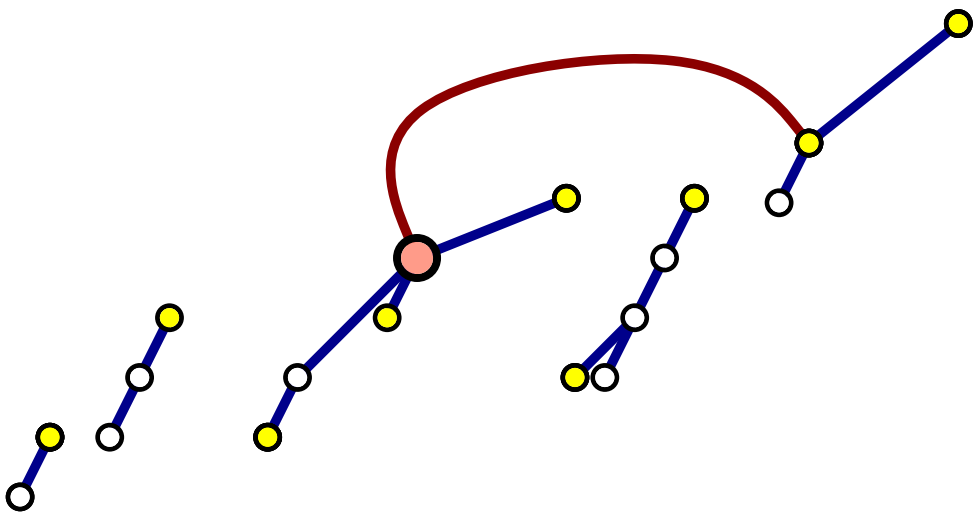
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top
- If splitter: place on opposite side

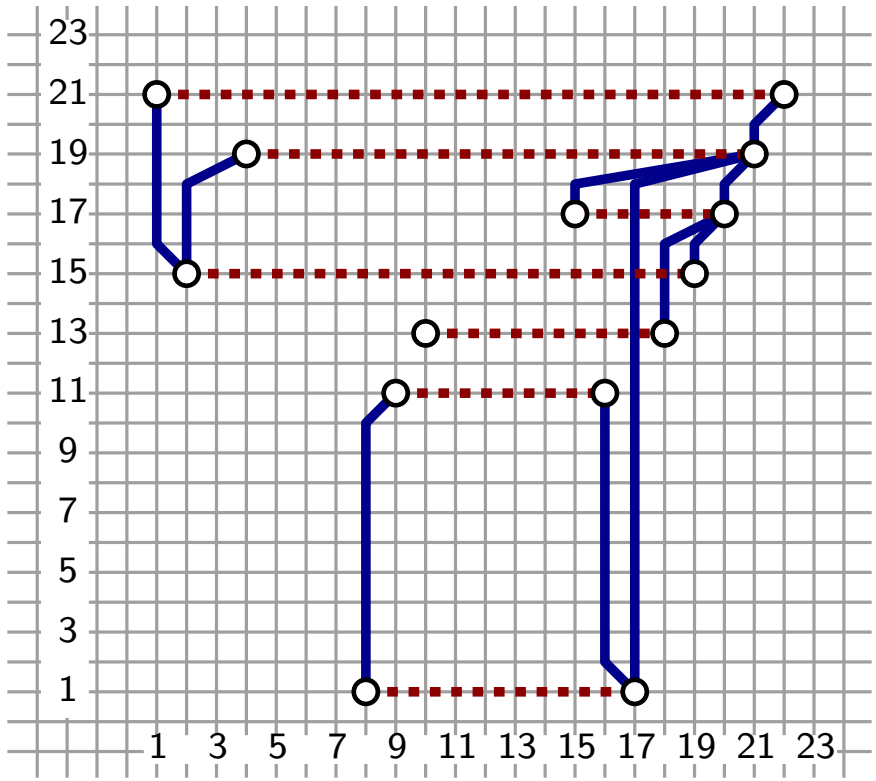




Tree × Matching

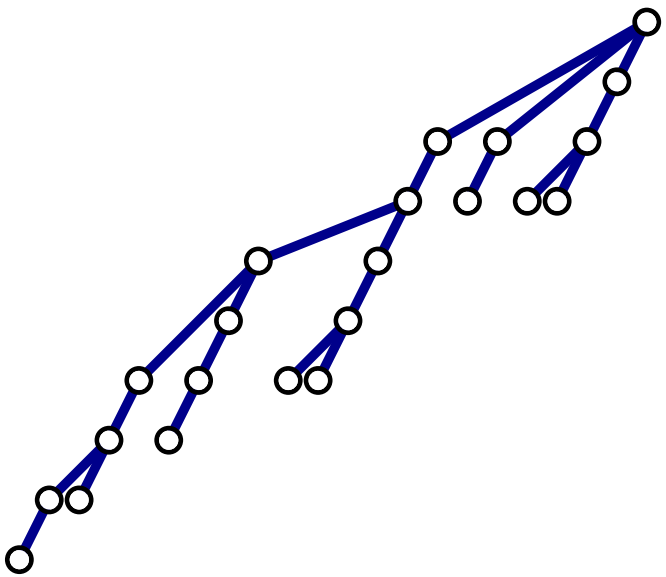
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top
- If splitter: place on opposite side

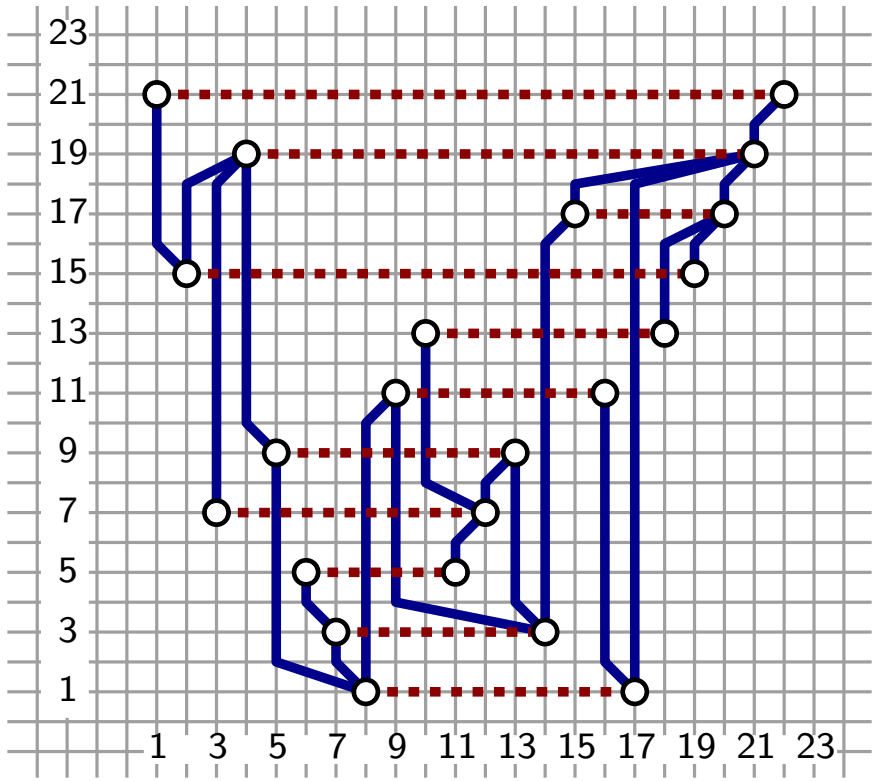




Tree × Matching

- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top
- If splitter: place on opposite side

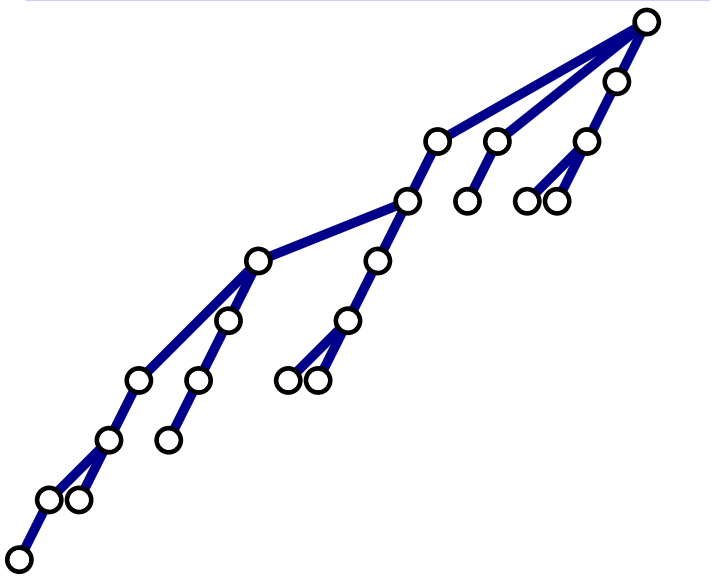


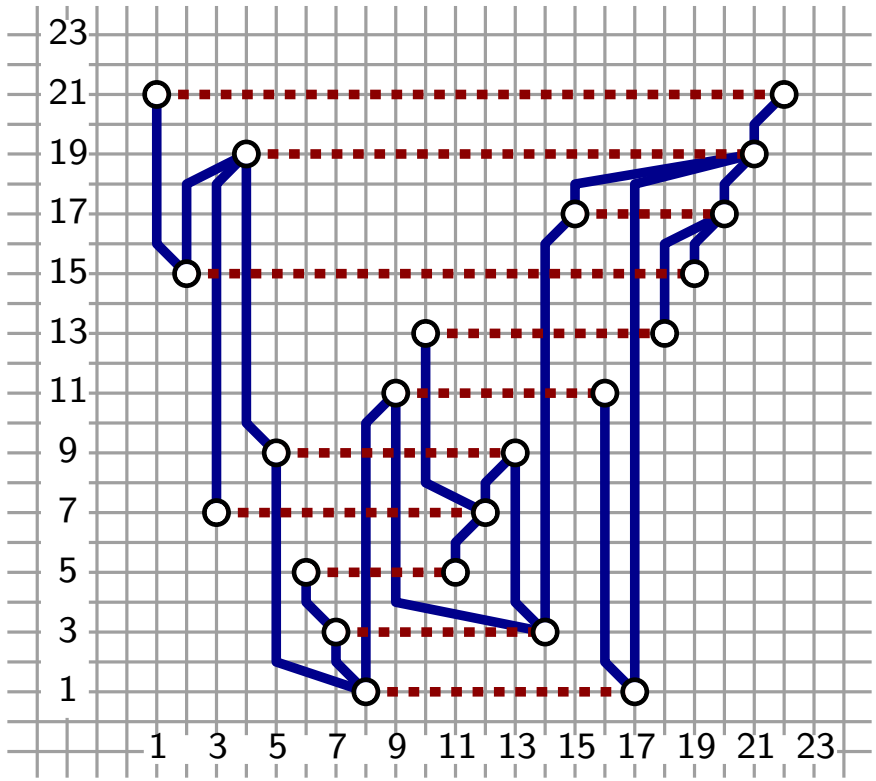


Tree × Matching

- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top
- If splitter: place on opposite side

Bends: 1×0
Grid size:
$n \times (n-1)$

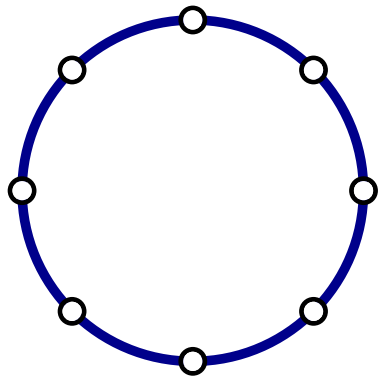




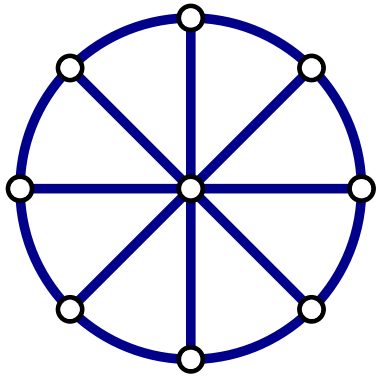
Overview

Graph classes			Number of bends
Cycle	$\times$	Cycle	1×1
Caterpillar	$\times$	Cycle	1×1
Four Matchings			$1 \times 1 \times 1 \times 1$
Tree	$\times$	Matching	1×0
Wheel	$\times$	Matching	2×0
Outerpath	$\times$	Matching	2×1
Outerplanar	$\times$	Outerplanar	3×3
2-page book emb.	$\times$	2-page book emb.	4×4
Planar	$\times$	Planar	6×6

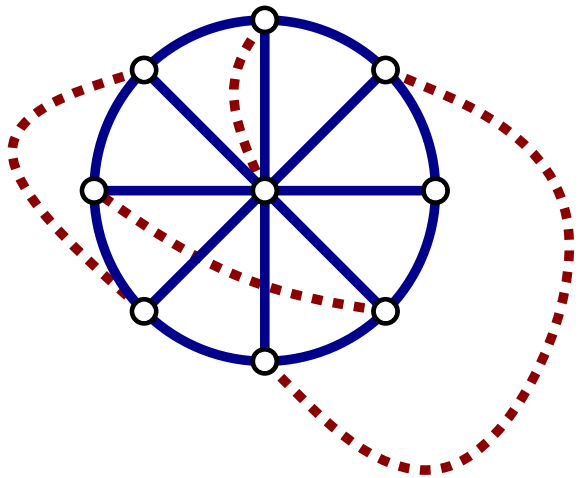
Wheel $\times$ Matching



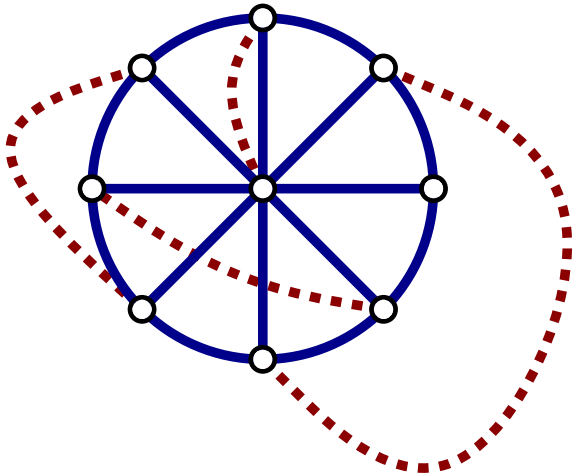
Wheel $\times$ Matching

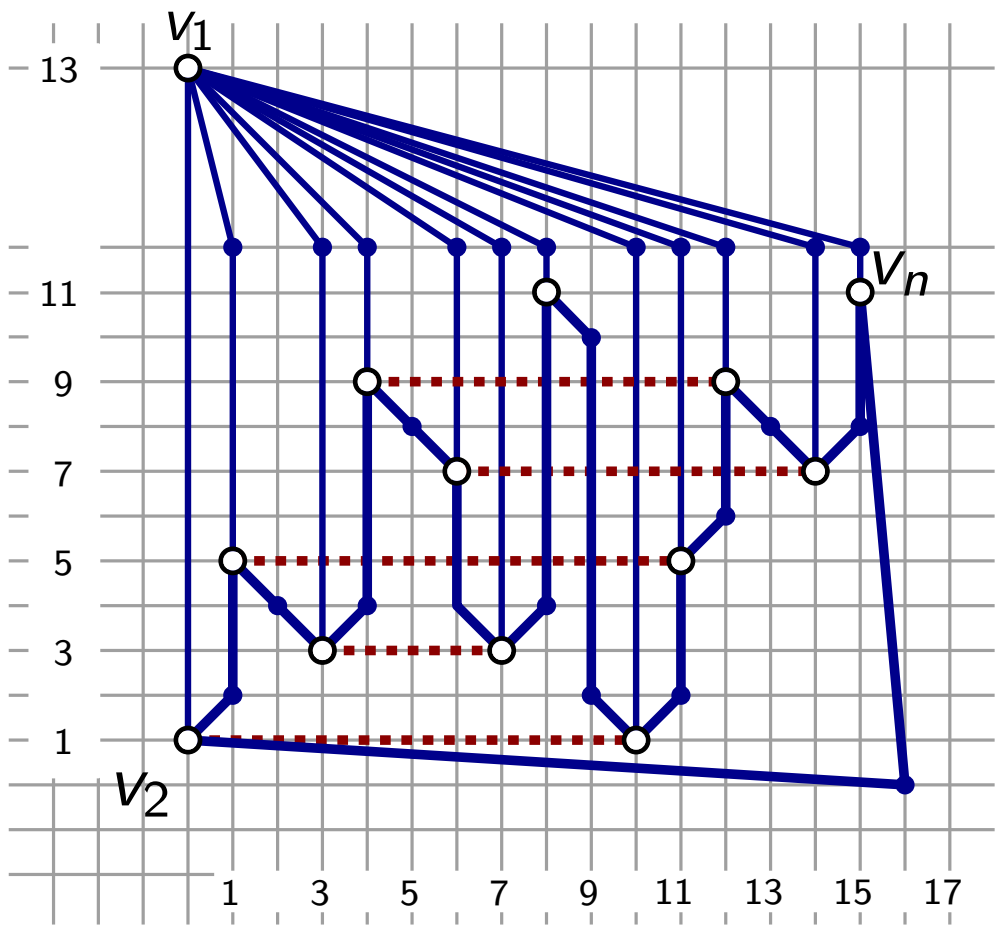


Wheel $\times$ Matching

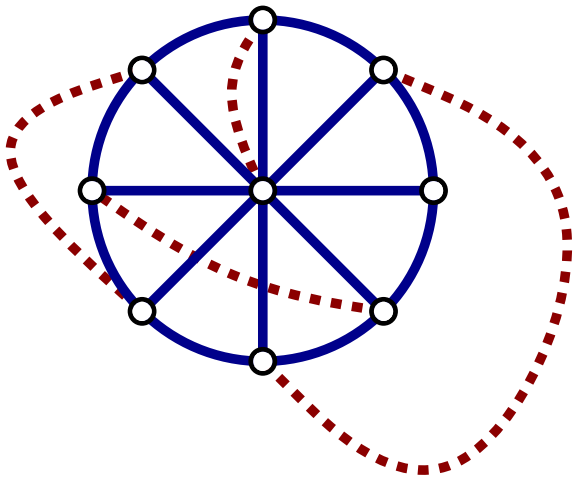


Wheel $\times$ Matching

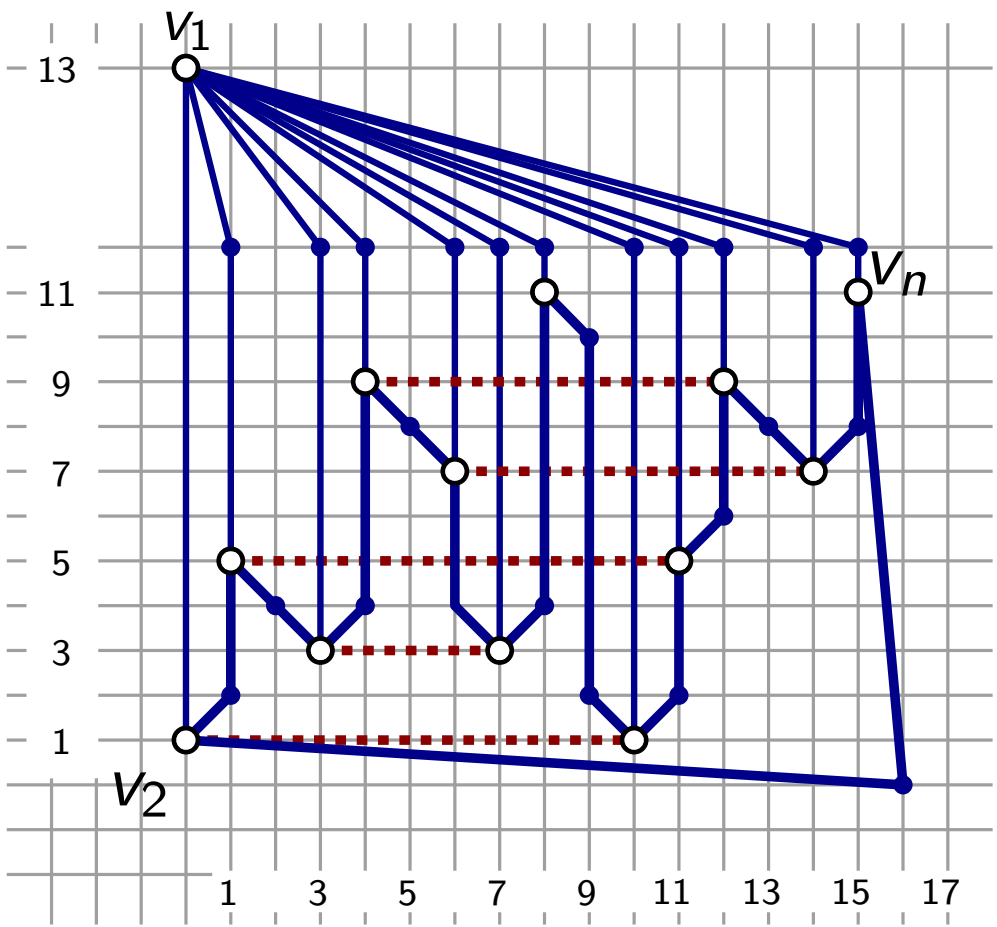




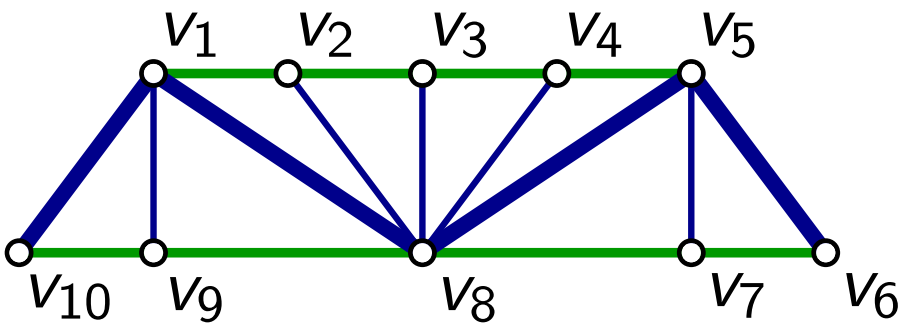
Wheel $\times$ Matching



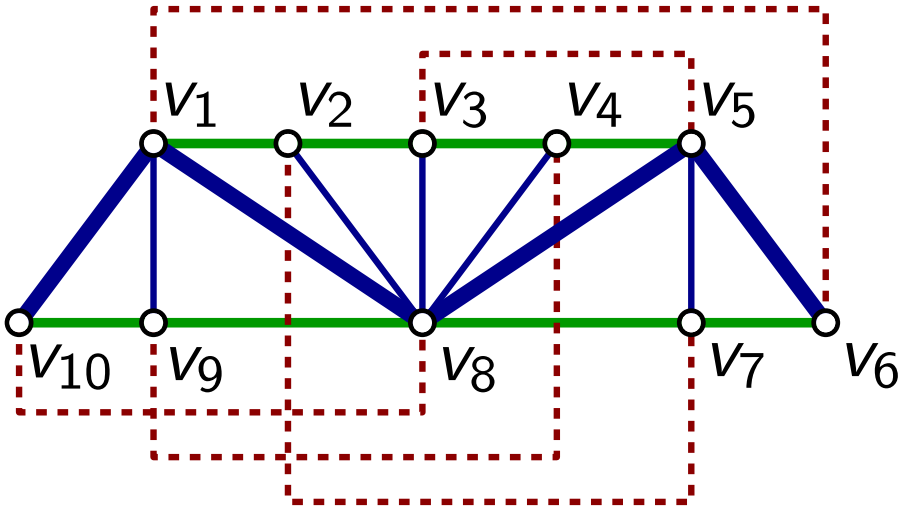
Bends: 2×0
Grid size:
$(1.5n - 1) \times (n + 2)$



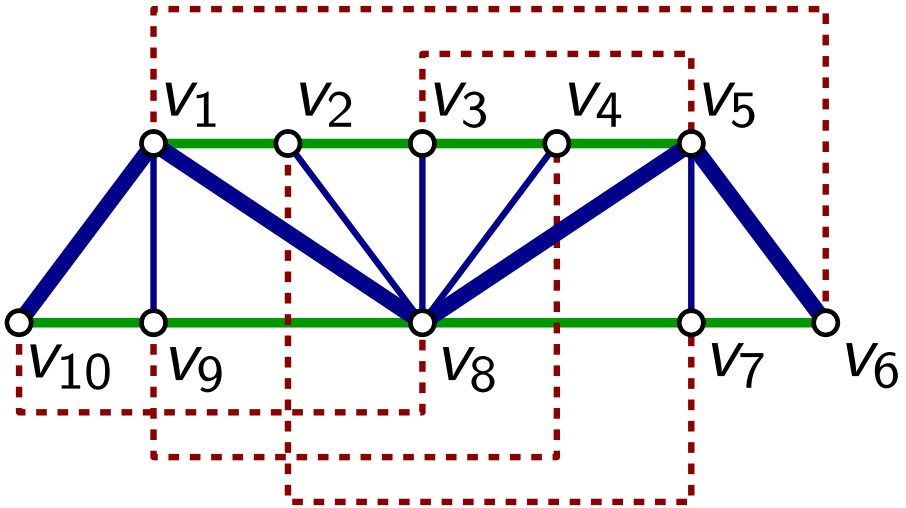
Outerpath $\times$ matching

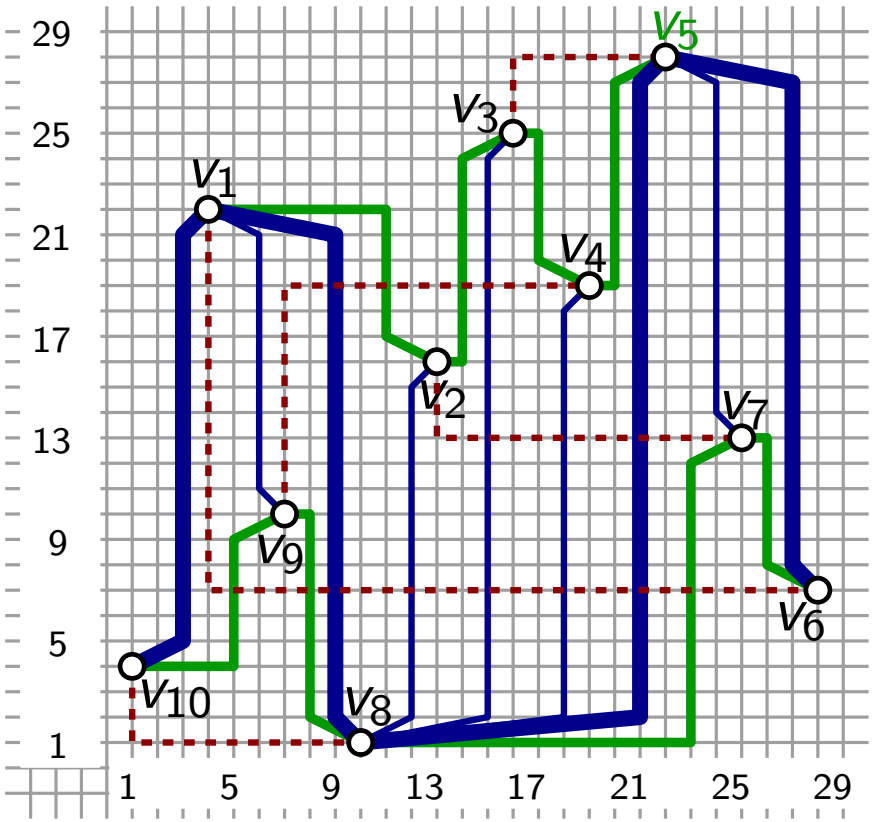


Outerpath $\times$ matching

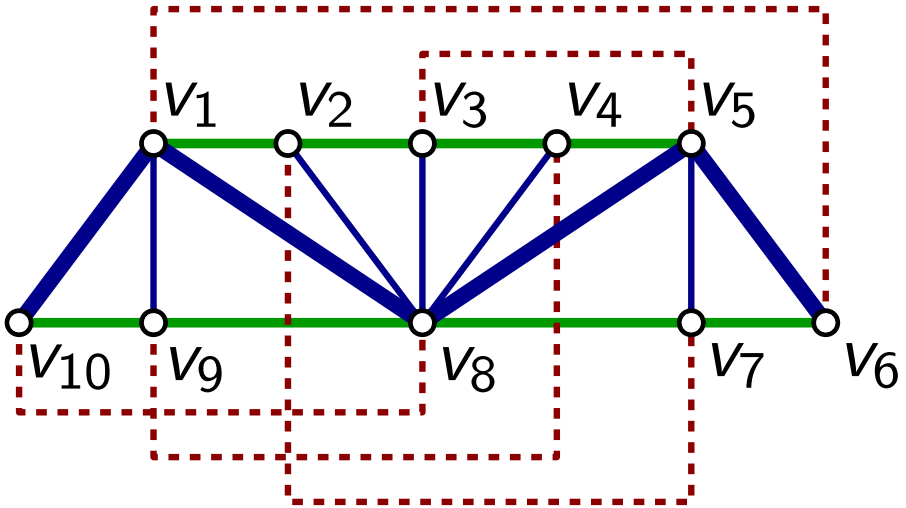


Outerpath × matching

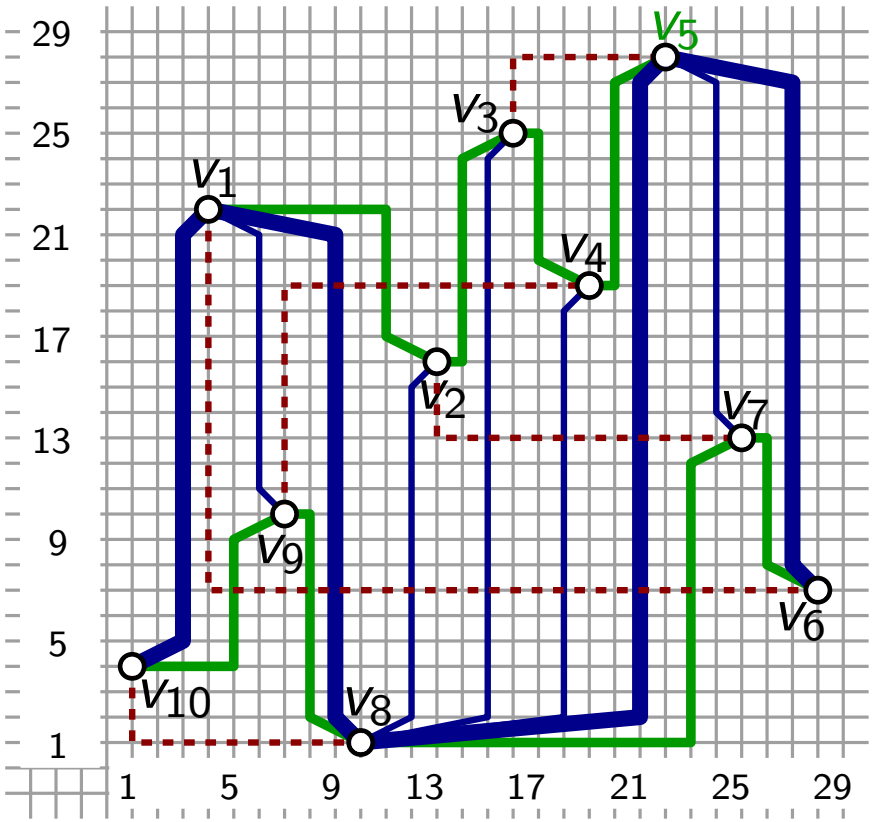




Outerpath × matching



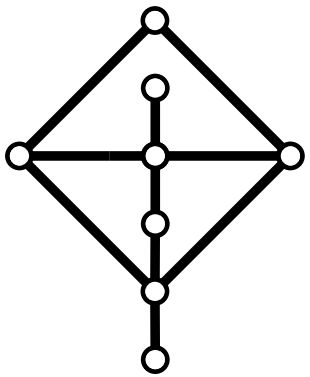
Bends: 2×1
Grid size:
$(3n-2) \times (3n-2)$



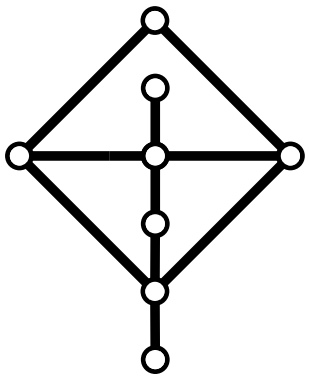
Overview

Graph classes			Number of bends
Cycle	$\times$	Cycle	1×1
Caterpillar	$\times$	Cycle	1×1
Four Matchings			$1 \times 1 \times 1 \times 1$
Tree	$\times$	Matching	1×0
Wheel	$\times$	Matching	2×0 ✓
Outerpath	$\times$	Matching	2×1 ✓
Outerplanar	$\times$	Outerplanar	3×3
2-page book emb.	$\times$	2-page book emb.	4×4
Planar	$\times$	Planar	6×6

Planar $\times$ Planar

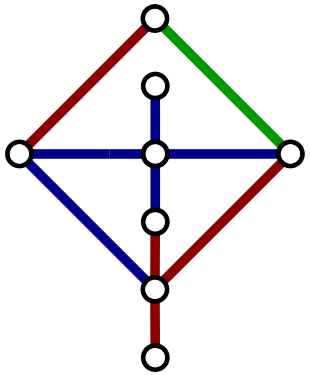


Planar $\times$ Planar

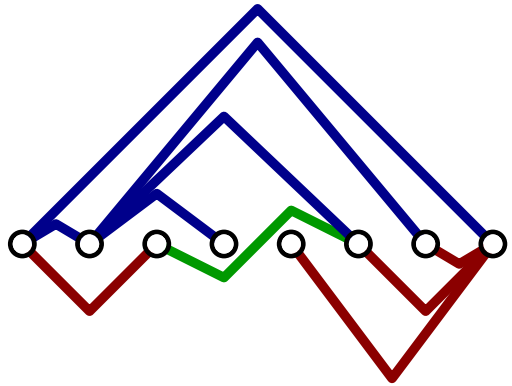


[Kaufmann & Wiese '02]:

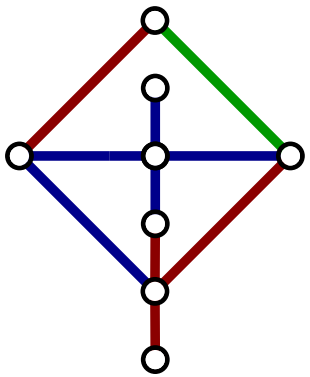
Planar $\times$ Planar



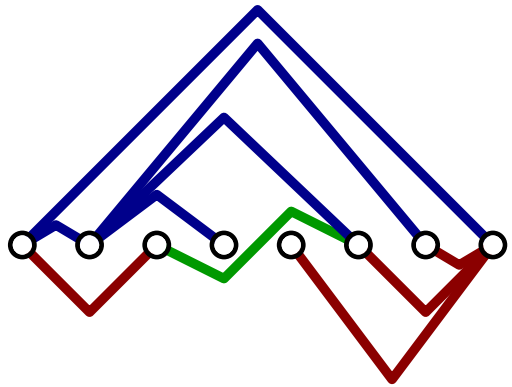
[Kaufmann & Wiese '02]:



Planar $\times$ Planar

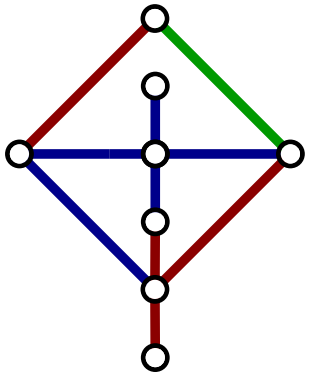


[Kaufmann & Wiese '02]:

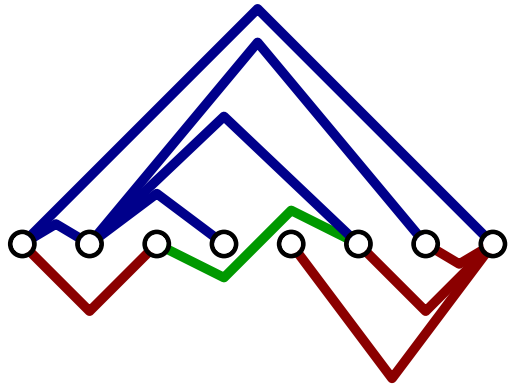


Graph 1: x-coordinates
Graph 2: y-coordinates

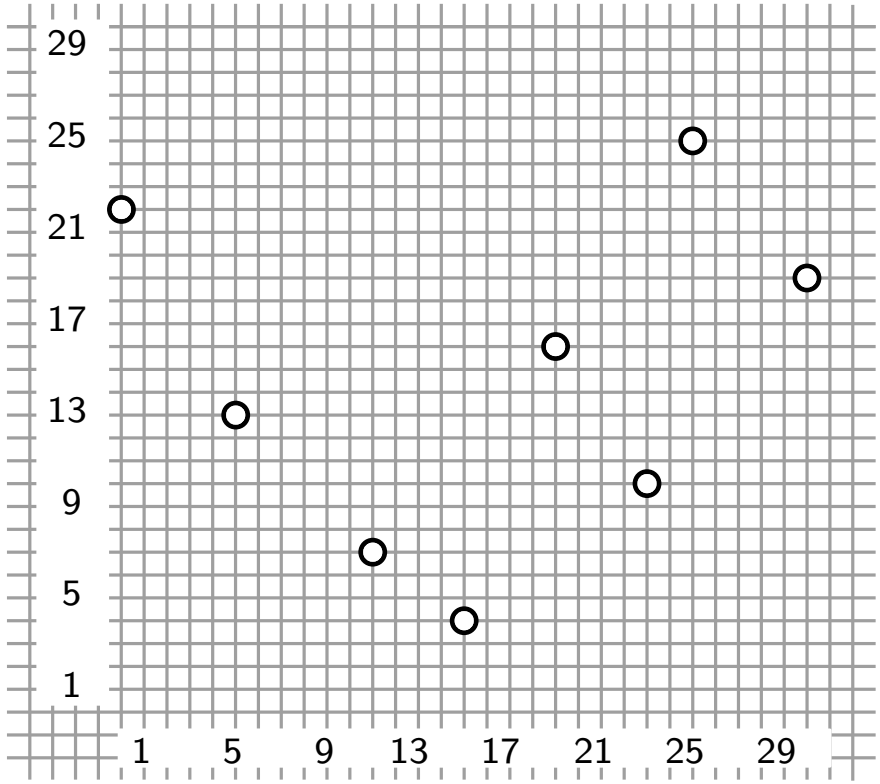
Planar × Planar



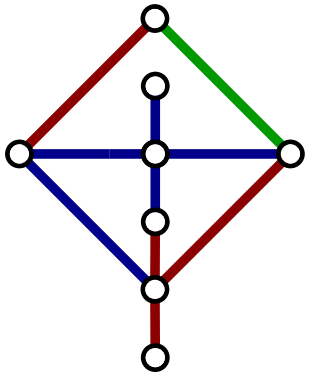
[Kaufmann & Wiese '02]:



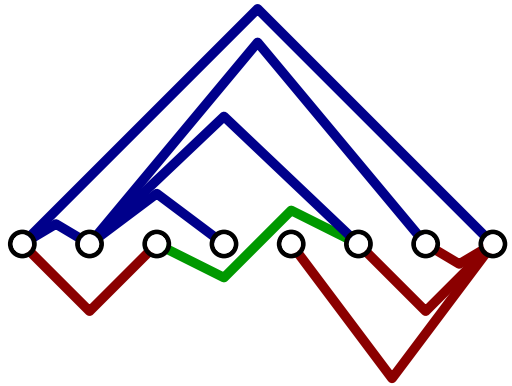
Graph 1: x-coordinates
Graph 2: y-coordinates



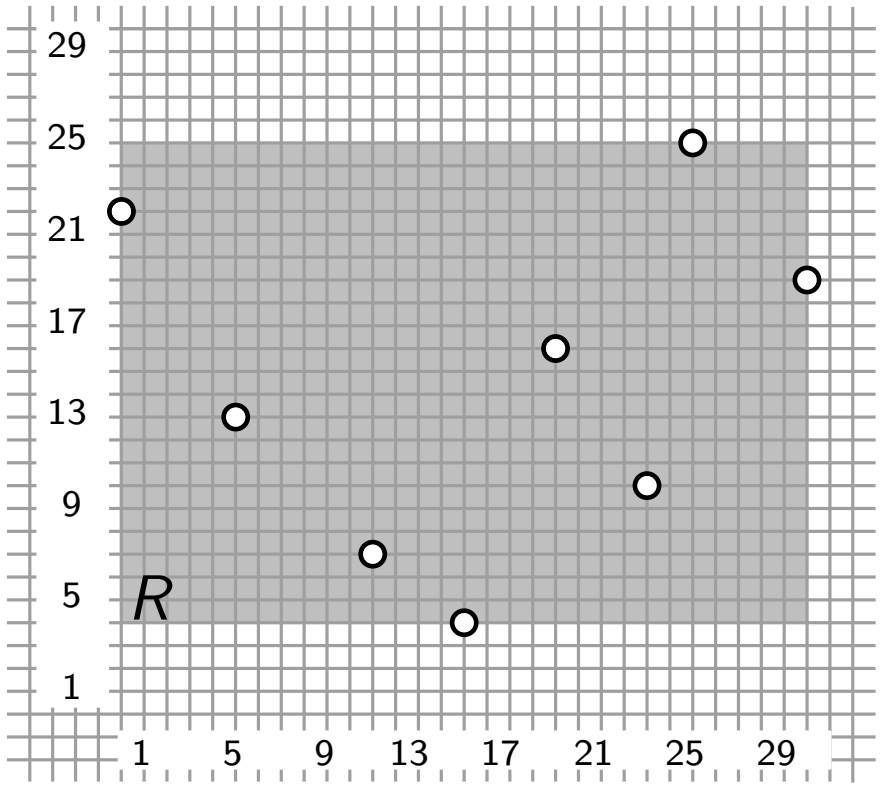
Planar × Planar



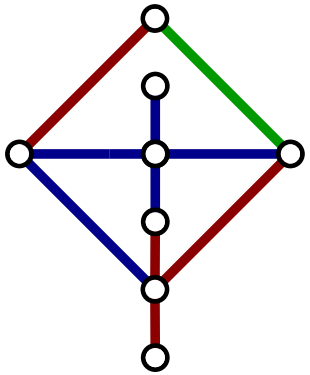
[Kaufmann & Wiese '02]:



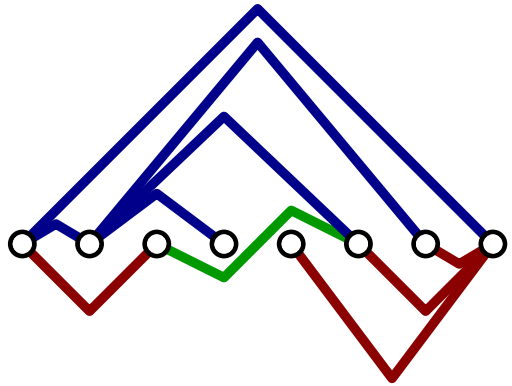
Graph 1: x-coordinates
Graph 2: y-coordinates



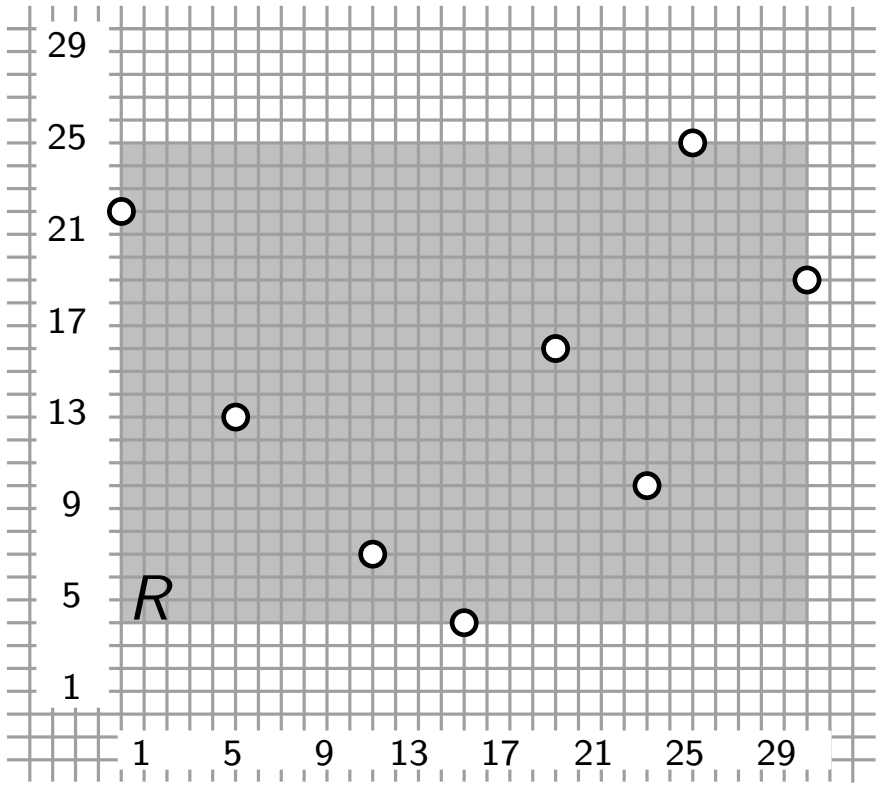
Planar × Planar



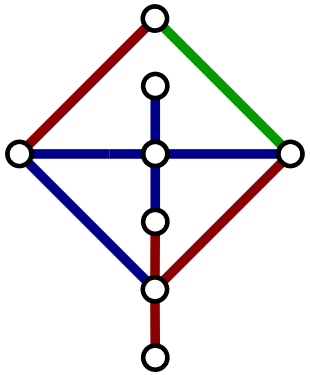
[Kaufmann & Wiese '02]:



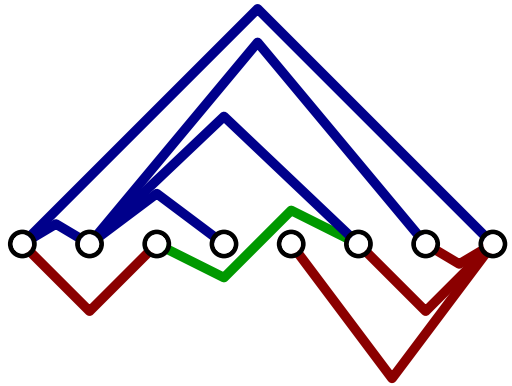
Graph 1: x-coordinates
Graph 2: y-coordinates
In R: all segments vertical
or of y-length 1



Planar × Planar



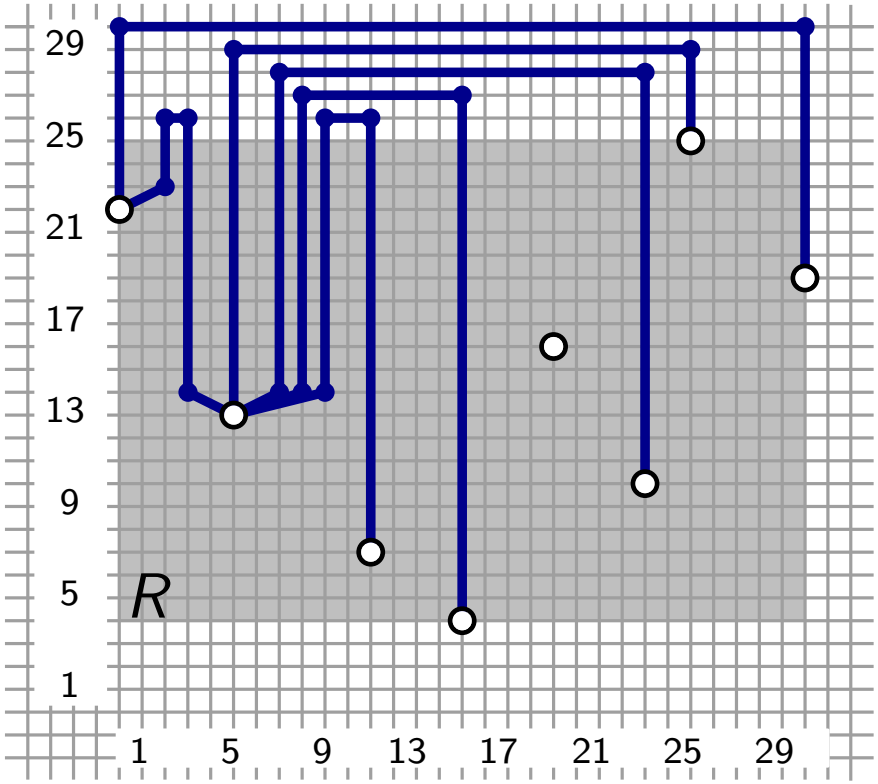
[Kaufmann & Wiese '02]:



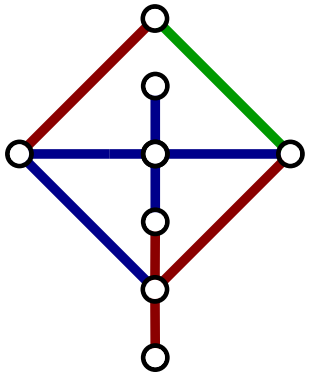
Graph 1: x-coordinates
Graph 2: y-coordinates
In R: all segments vertical
or of y-length 1

Edges:

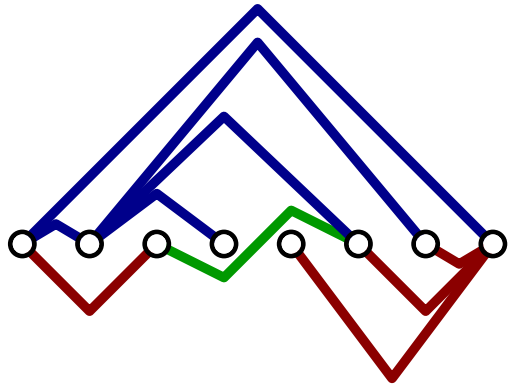
4



Planar × Planar

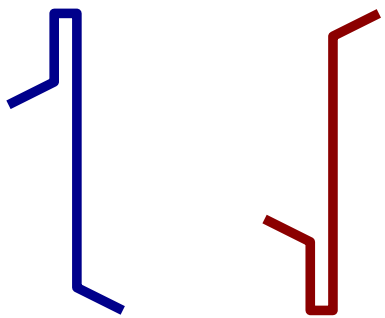


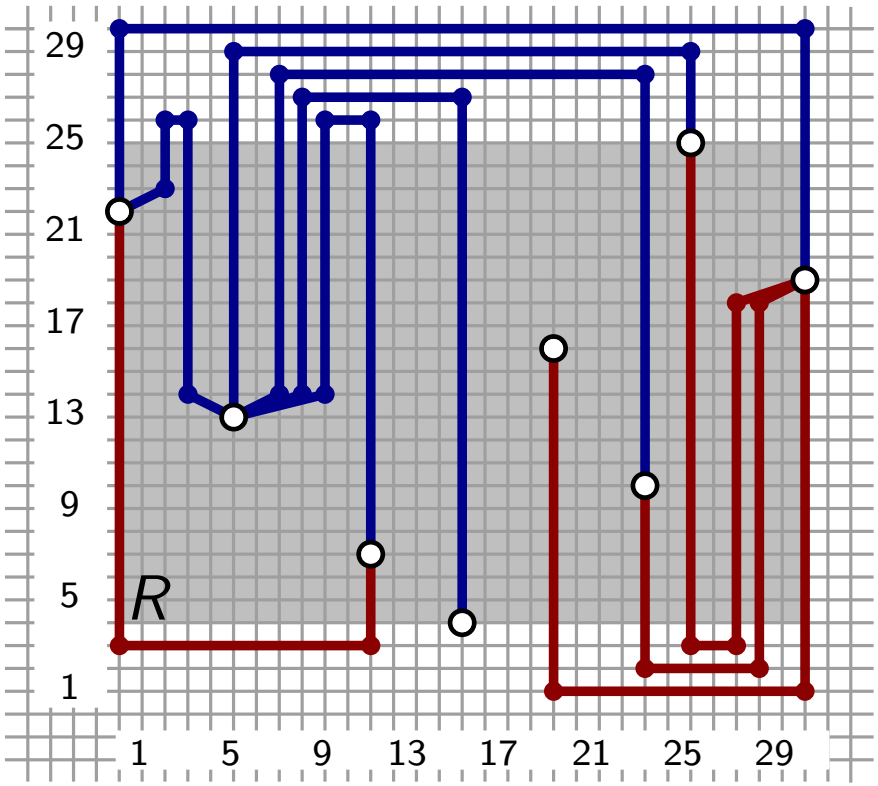
[Kaufmann & Wiese '02]:



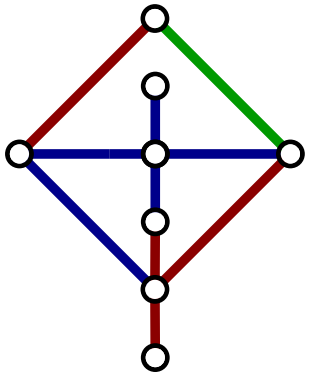
Graph 1: x-coordinates
Graph 2: y-coordinates
In R: all segments vertical
or of y-length 1

Edges:

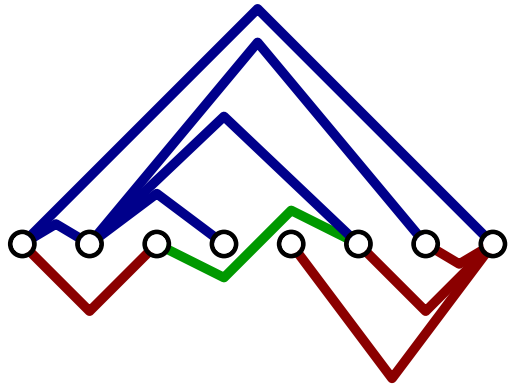




Planar × Planar

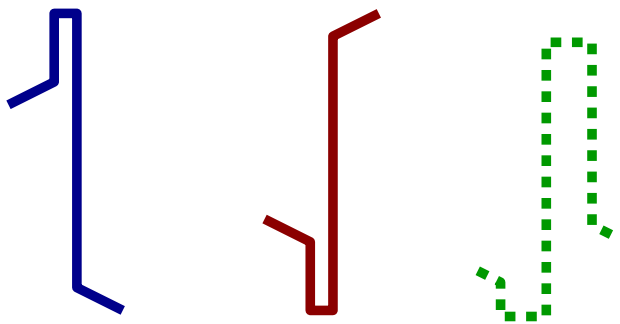


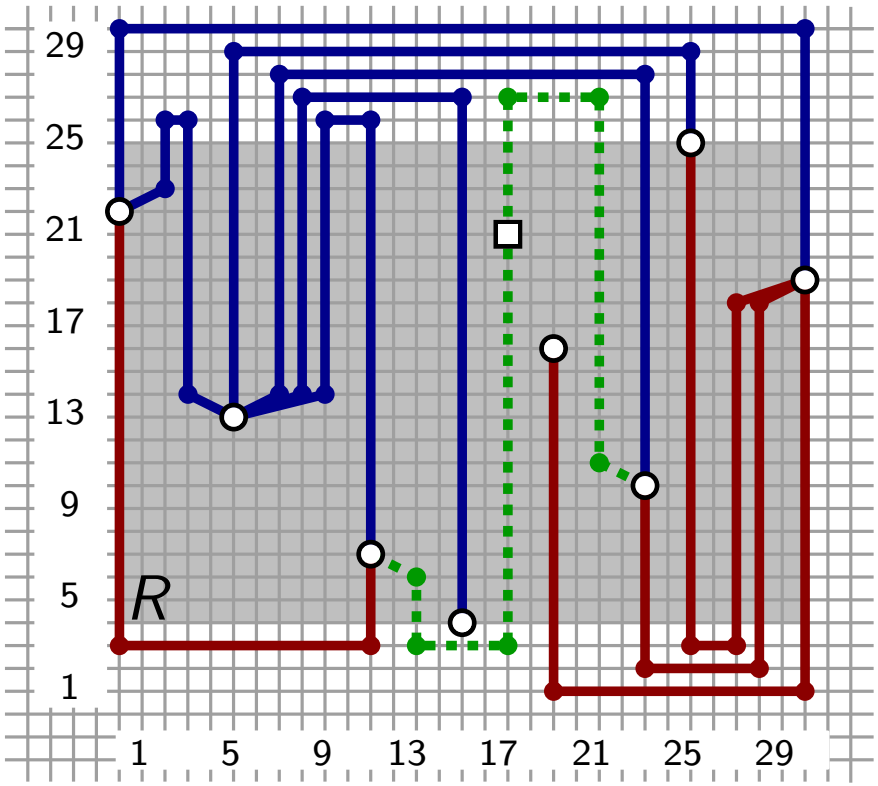
[Kaufmann & Wiese '02]:



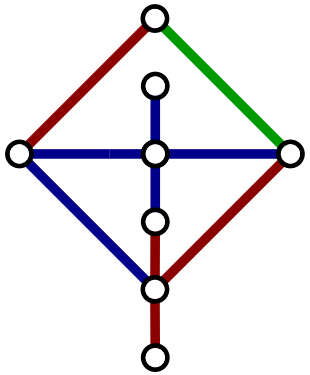
Graph 1: x-coordinates
Graph 2: y-coordinates
In R: all segments vertical
or of y-length 1

Edges:

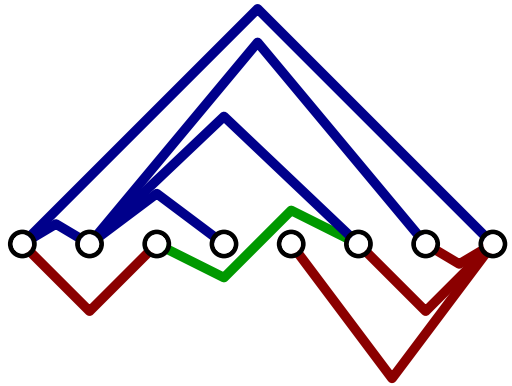




Planar × Planar

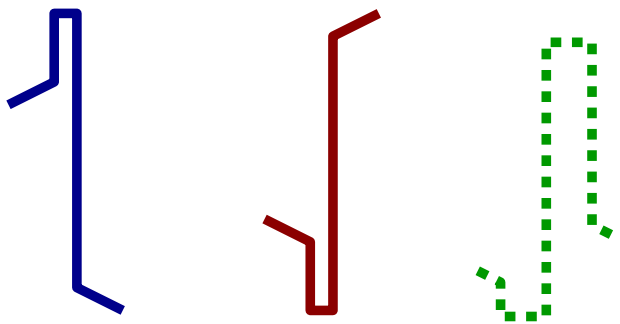


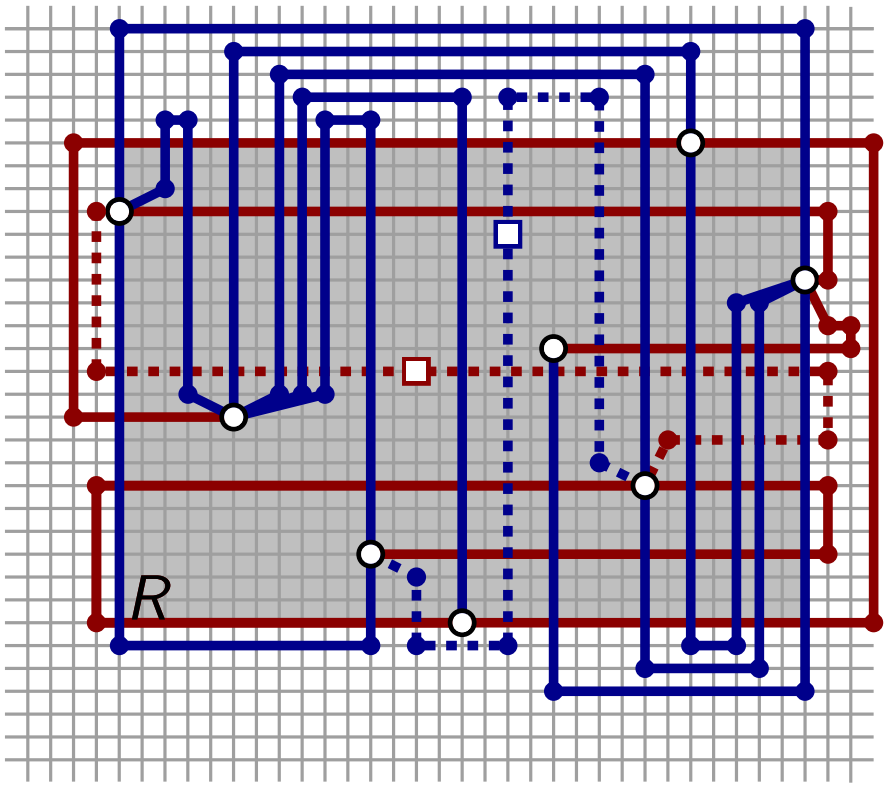
[Kaufmann & Wiese '02]:



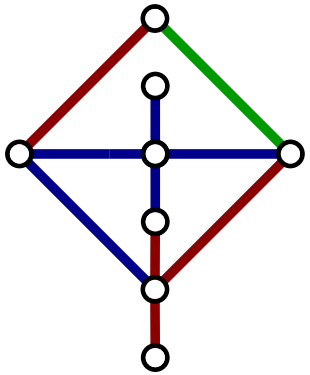
Graph 1: x-coordinates
Graph 2: y-coordinates
In R: all segments vertical
or of y-length 1

Edges:

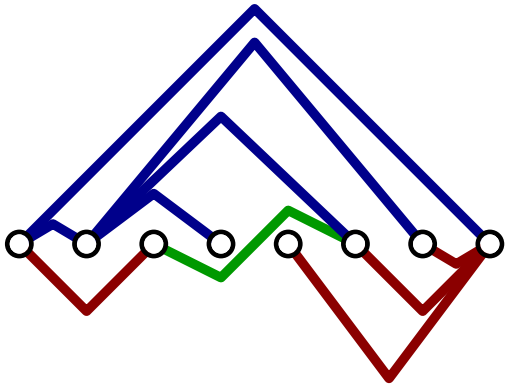




Planar × Planar



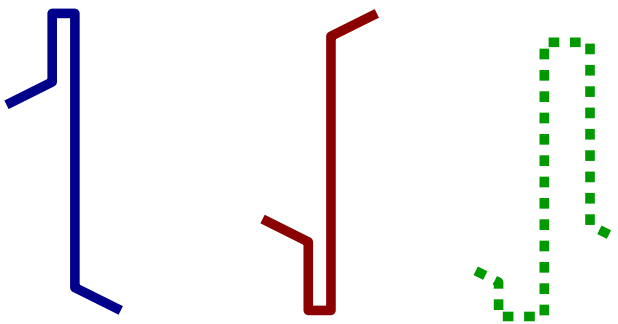
[Kaufmann & Wiese '0

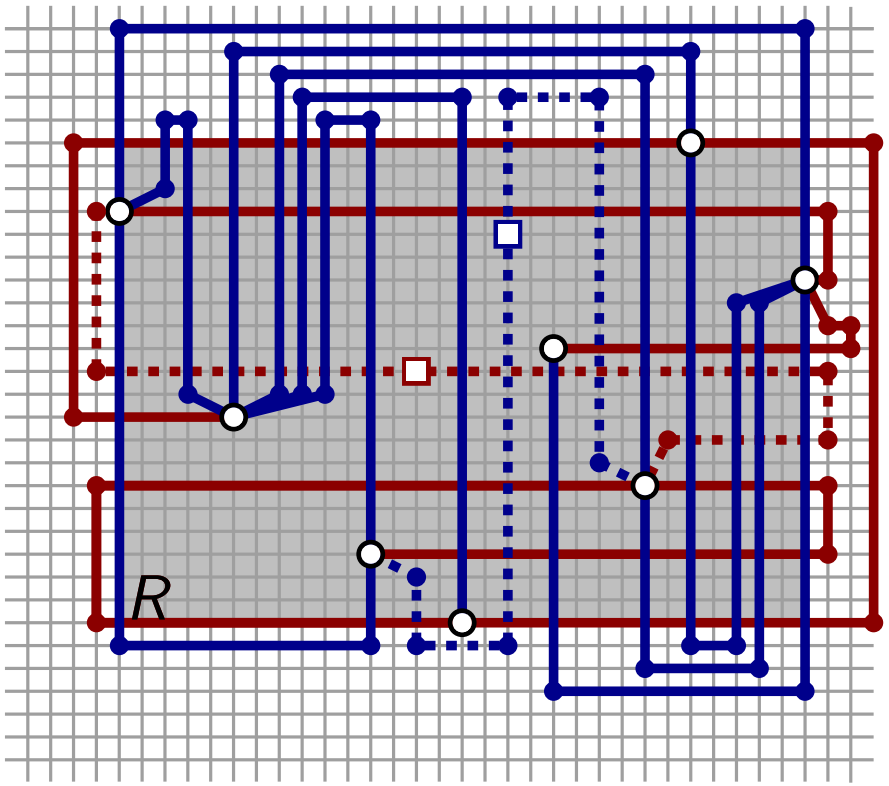


Bends: 6×6
Grid size:
$(14n - 26) \times (14n - 26)$

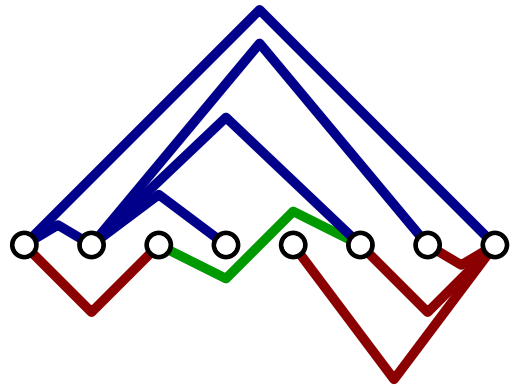
Graph 1: x-coordinates
Graph 2: y-coordinates
In R: all segments vertical
or of y-length 1

Edges:



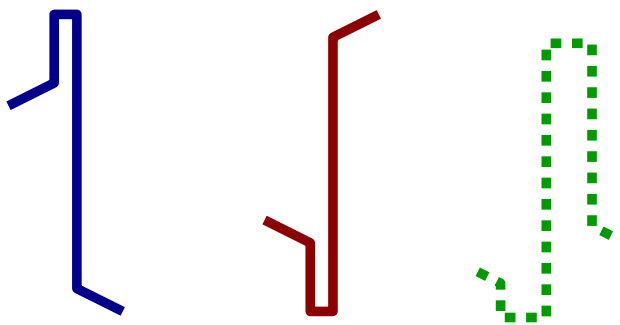


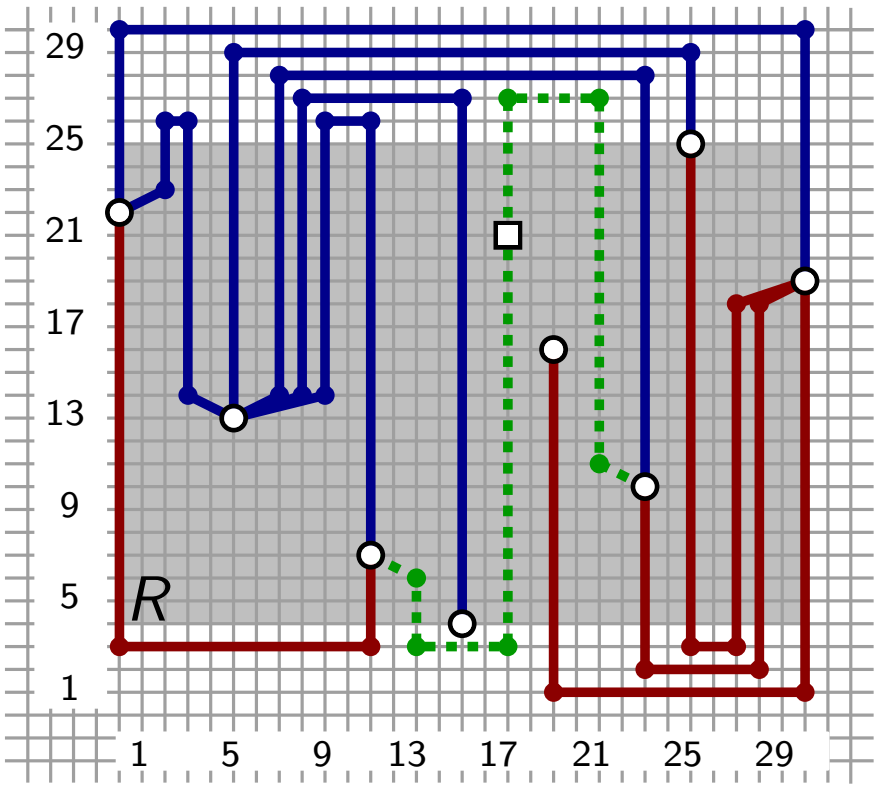
2-page book embeddable



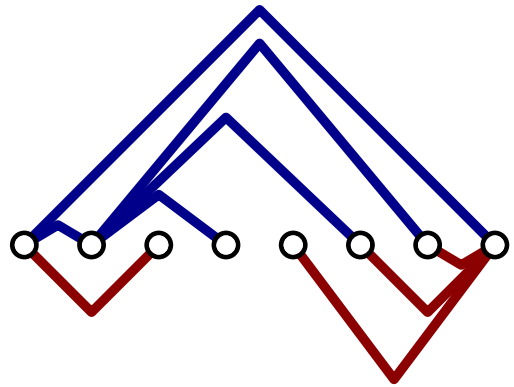
Graph 1: x-coordinates
Graph 2: y-coordinates
In R: all segments vertical
or of y-length 1

Edges:



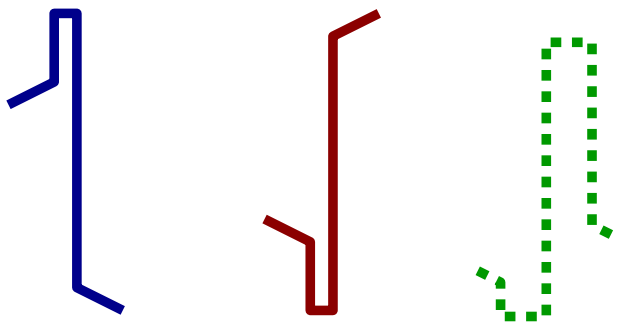


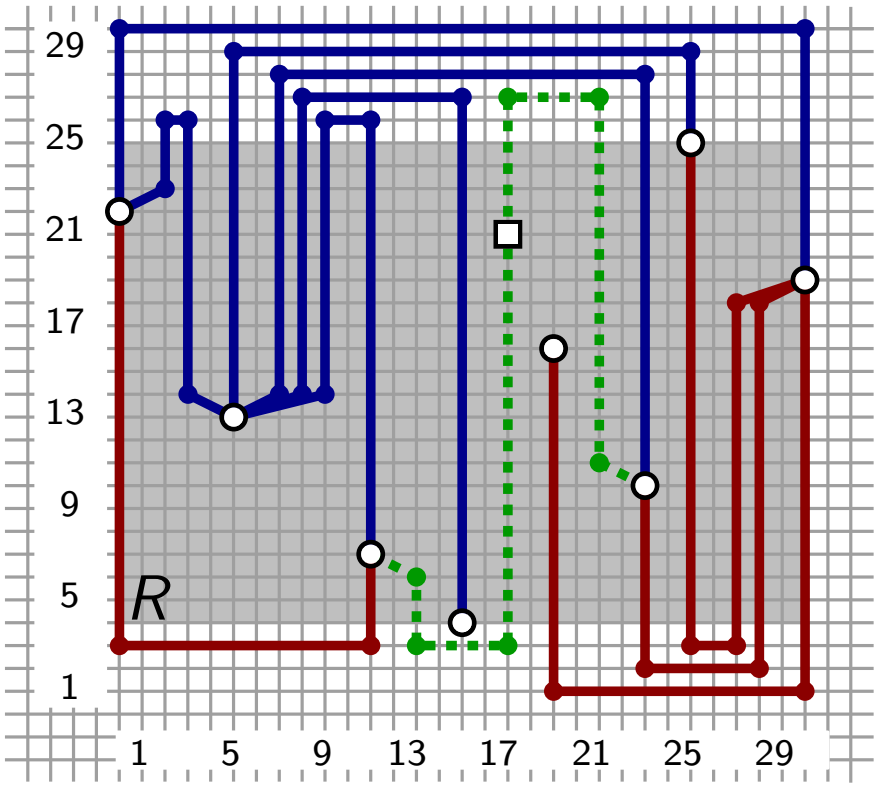
2-page book embeddable



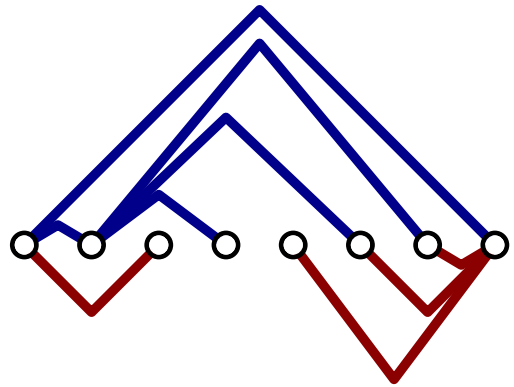
Graph 1: x-coordinates
Graph 2: y-coordinates
In R: all segments vertical
or of y-length 1

Edges:



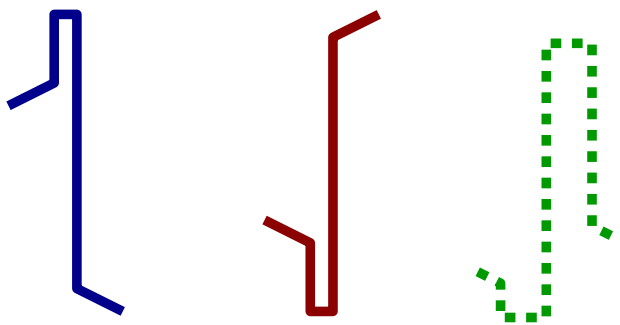


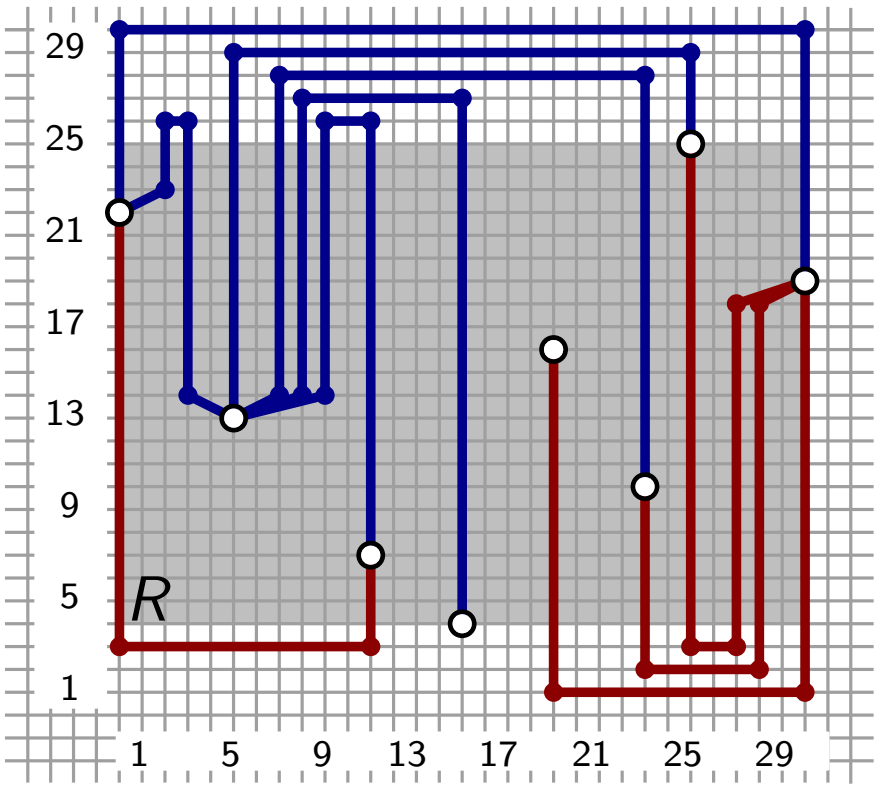
2-page book embeddable



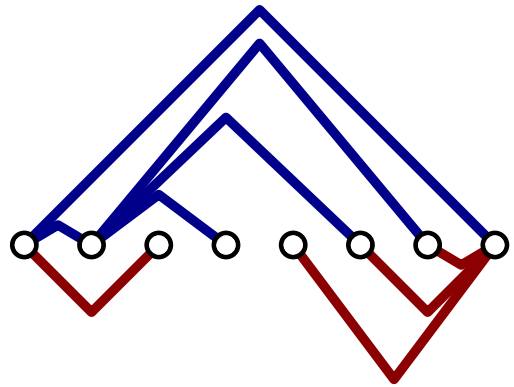
Graph 1: x-coordinates
Graph 2: y-coordinates
In R: all segments vertical
or of y-length 1

Edges:



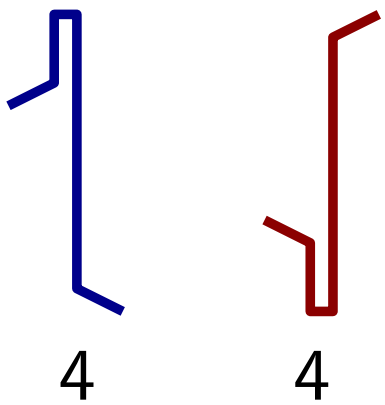


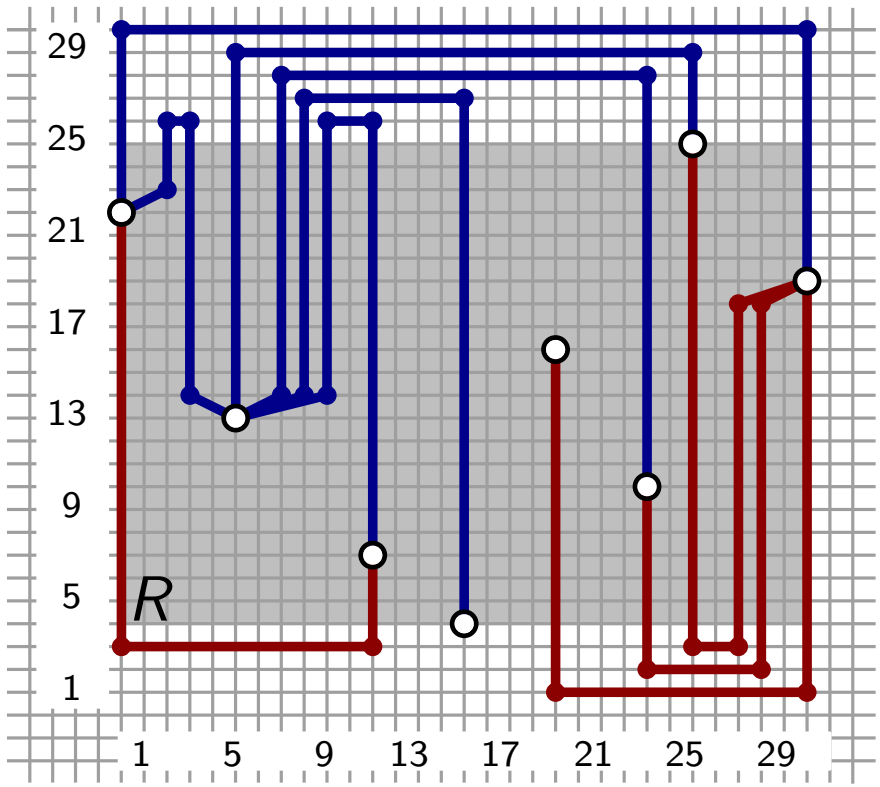
2-page book embeddable



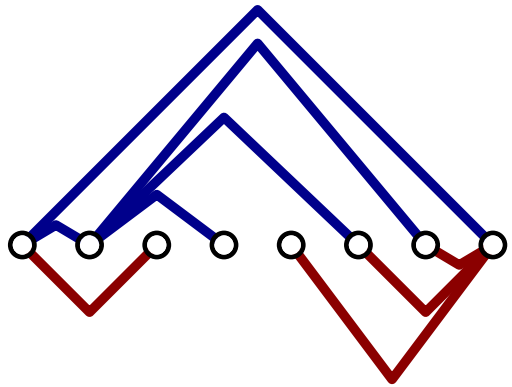
Graph 1: x-coordinates
Graph 2: y-coordinates
In R: all segments vertical
or of y-length 1

Edges:





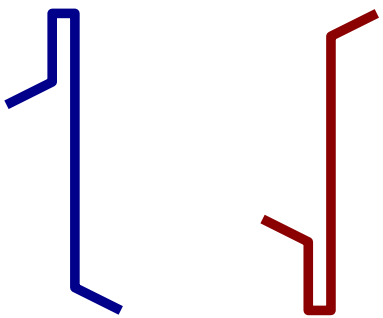
2-page book embeddable

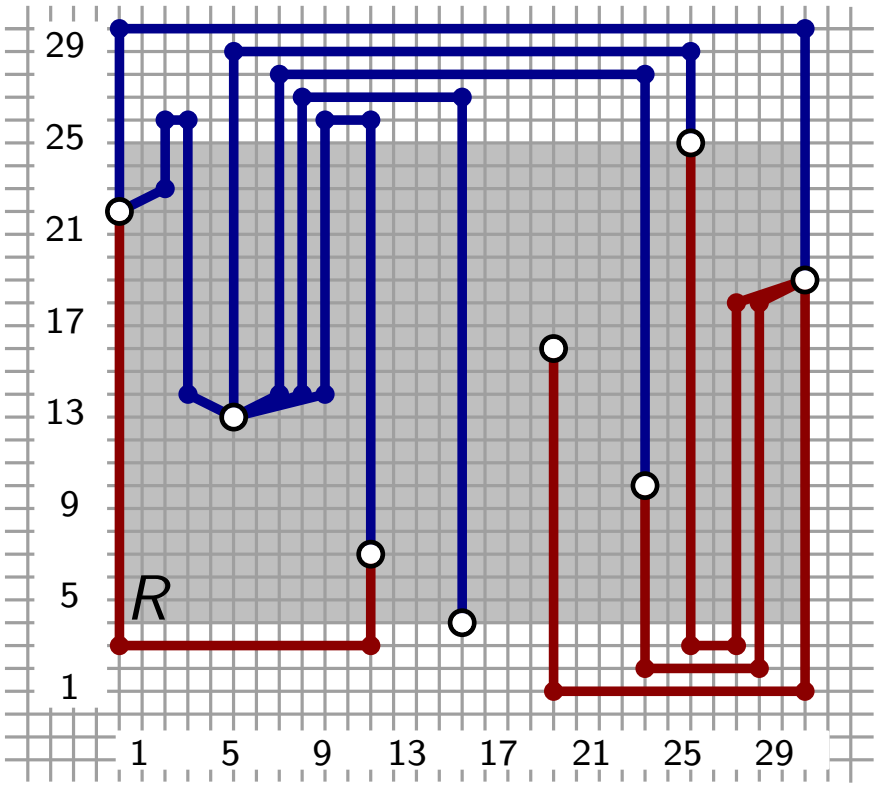


Bends: 4×4
Grid size:
$(11n - 32) \times (11n - 32)$

Graph 1: x-coordinates
Graph 2: y-coordinates
In R: all segments vertical
or of y-length 1

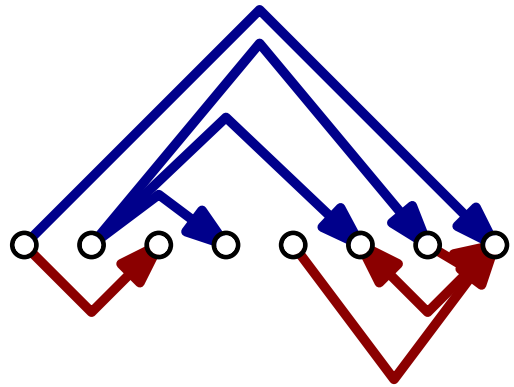
Edges:





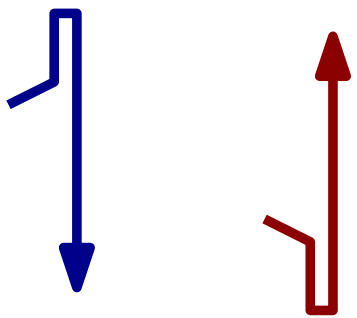
Outerplanar × Outerplanar

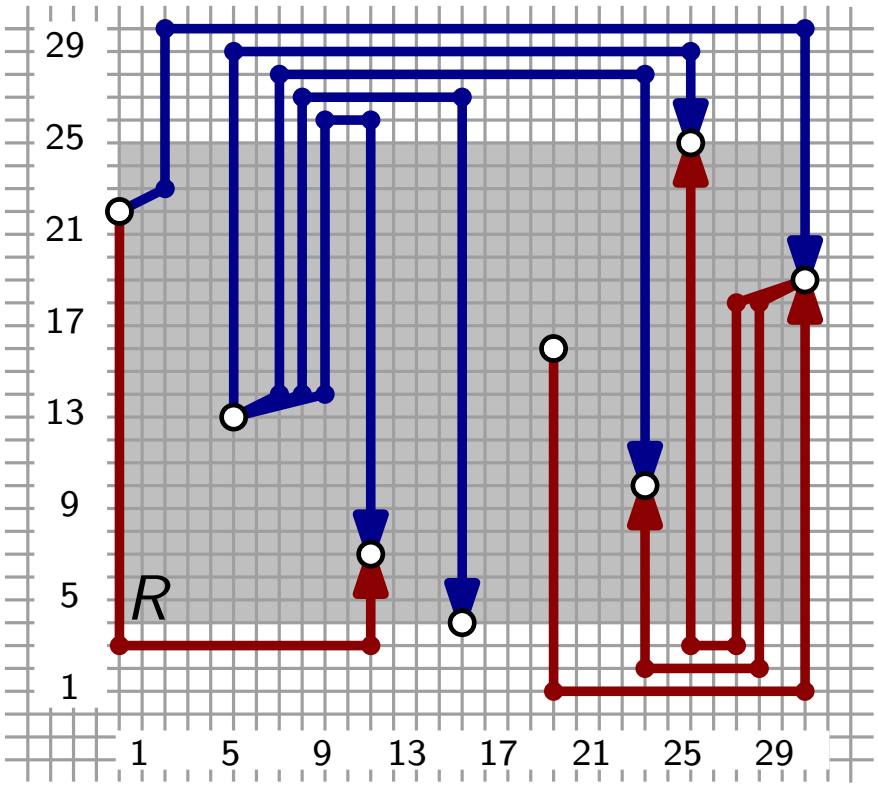
Decompose
into two
directed trees



Graph 1: x-coordinates
Graph 2: y-coordinates
In R: all segments vertical
or of y-length 1

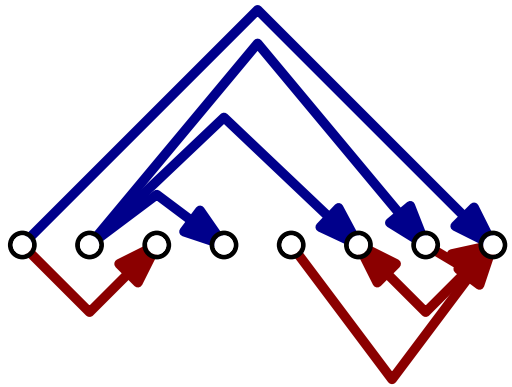
Edges:





Outerplanar × Outerplanar

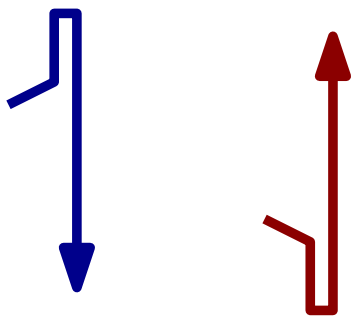
Decompose
into two
directed trees

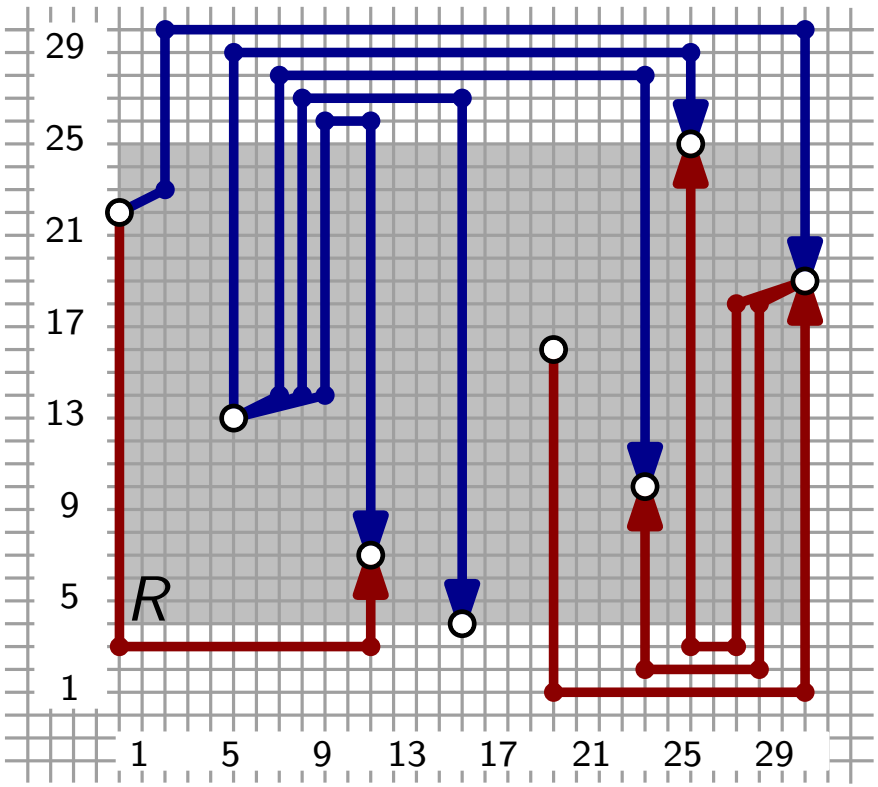


Bends: 3×3
Grid size:
$(7n - 10) \times (7n - 10)$

Graph 1: x-coordinates
Graph 2: y-coordinates
In R: all segments vertical
or of y-length 1

Edges:





Overview

Graph classes			Number of bends	
Cycle	$\times$	Cycle	1×1	
Caterpillar	$\times$	Cycle	1×1	
Four Matchings			$1 \times 1 \times 1 \times 1$	
Tree	$\times$	Matching	1×0	
Wheel	$\times$	Matching	2×0	
Outerpath	$\times$	Matching	2×1	
Outerplanar	$\times$	Outerplanar	3×3	✓
2-page book emb.	$\times$	2-page book emb.	4×4	✓
Planar	$\times$	Planar	6×6	✓

Overview

Graph classes			Number of bends
Cycle	$\times$	Cycle	1×1
Caterpillar	$\times$	Cycle	1×1
Four Matchings			$1 \times 1 \times 1 \times 1$
Tree	$\times$	Matching	1×0
Wheel	$\times$	Matching	2×0
Outerpath	$\times$	Matching	2×1
Outerplanar	$\times$	Outerplanar	3×3
2-page book emb.	$\times$	2-page book emb.	4×4
Planar	$\times$	Planar	6×6

- All graphs are drawn on the $O(n) \times O(n)$-grid

Overview

Graph classes			Number of bends
Cycle	$\times$	Cycle	1×1
Caterpillar	$\times$	Cycle	1×1
Four Matchings			$1 \times 1 \times 1 \times 1$
Tree	$\times$	Matching	1×0
Wheel	$\times$	Matching	2×0
Outerpath	$\times$	Matching	2×1
Outerplanar	$\times$	Outerplanar	3×3
2-page book emb.	$\times$	2-page book emb.	4×4
Planar	$\times$	Planar	6×6

- All graphs are drawn on the $O(n) \times O(n)$-grid
- All algorithms run in $O(n)$ time

Open Problems

- Relax constraints on crossing resolution
  $\Rightarrow$ LAC (*Large Angle Crossings*)

Open Problems

- Relax constraints on crossing resolution
  $\Rightarrow$ LAC (*Large Angle Crossings*)

- Comp. Complexity of general problem:
  Given two graphs, is there a RACSIM drawing with at most k bends per edge

Open Problems

- Relax constraints on crossing resolution
  $\Rightarrow$ LAC (*Large Angle Crossings*)

- Comp. Complexity of general problem:
  Given two graphs, is there a RACSIM drawing with at most k bends per edge

- SIMRAC: Each graph is drawn RAC