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

Alexander Wolff
Chair of Computer Science I
Universität Würzburg

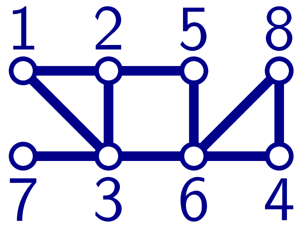
Joint work with
Michael A. Bekos · Thomas C. van Dijk · *Philipp Kindermann*

Simultaneous Embedding

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

Simultaneous Embedding

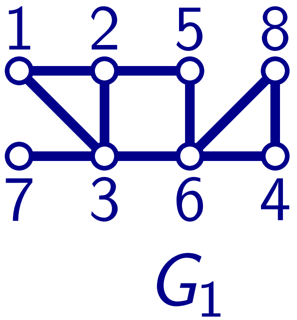
Given: Two planar graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ on same vertex set

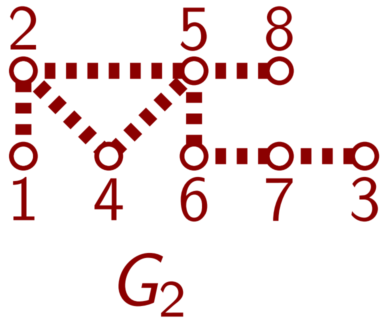


G_1

Simultaneous Embedding

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





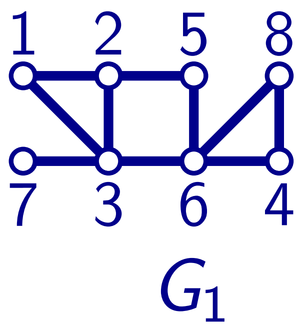
Simultaneous Embedding

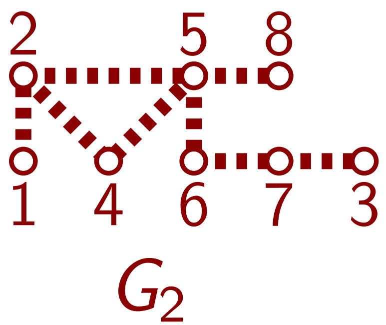
Given: Two planar graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$
on same vertex set

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



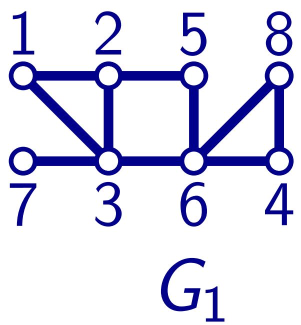


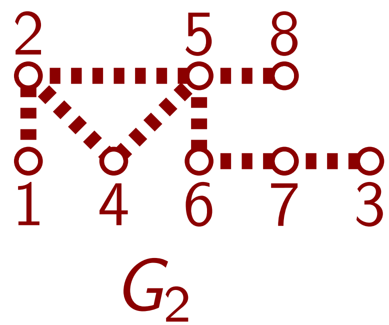
Embed both graphs in a planar way
– edges of one graph may intersect edges of the other graph

- Sim. Geometric Emb. (SGE)

Simultaneous Embedding

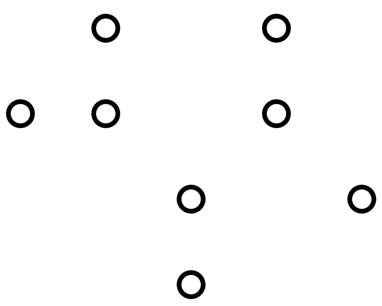
Given: Two planar graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$
on same vertex set





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

- Sim. Geometric Emb. (SGE)

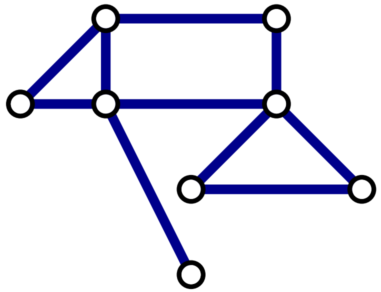


Simultaneous Embedding

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

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

- Sim. Geometric Emb. (SGE)

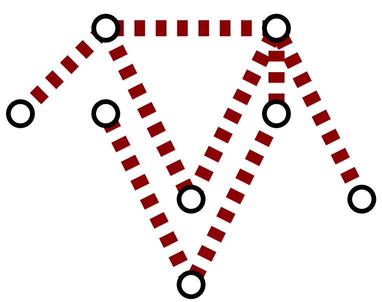


Simultaneous Embedding

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

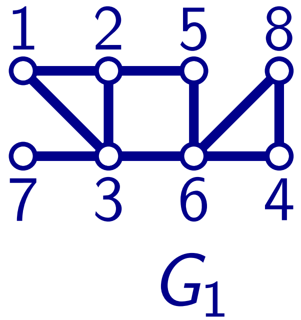
Embed both graphs in a planar way
– edges of one graph may intersect edges of the other graph

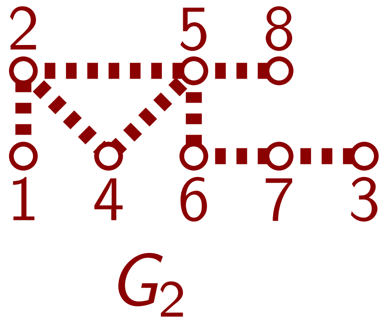
- SIM. GEOMETRIC EMB. (SGE)



Simultaneous Embedding

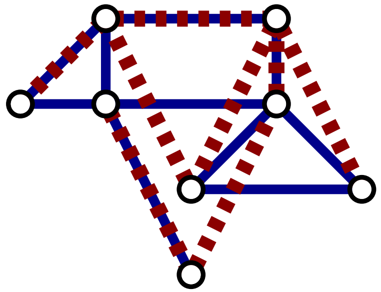
Given: Two planar graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$
on same vertex set





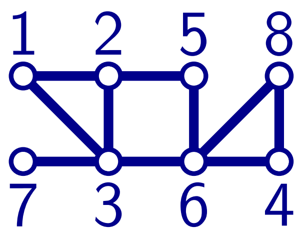
Embed both graphs in a planar way
– edges of one graph may intersect edges of the other graph

- Sim. Geometric Emb. (SGE)

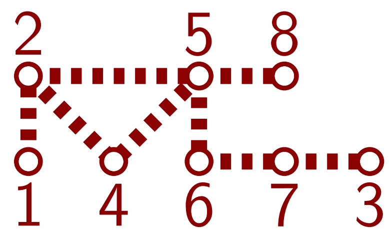


Simultaneous Embedding

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



G_1



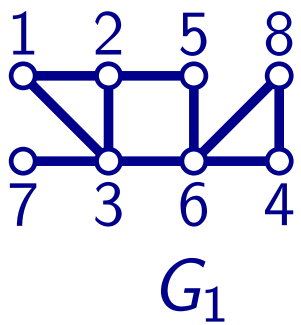
G_2

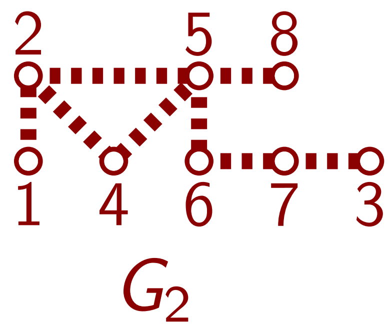
Embed both graphs in a planar way
– edges of one graph may intersect edges of the other graph

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

Simultaneous Embedding

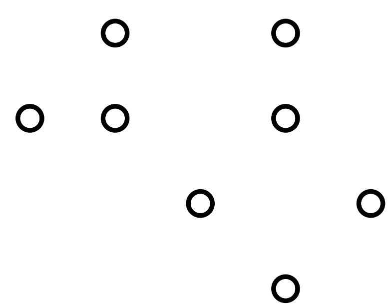
Given: Two planar graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$
on same vertex set





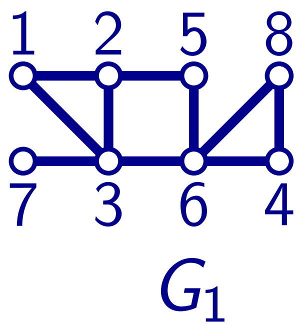
Embed both graphs in a planar way
– edges of one graph may intersect edges of the other graph

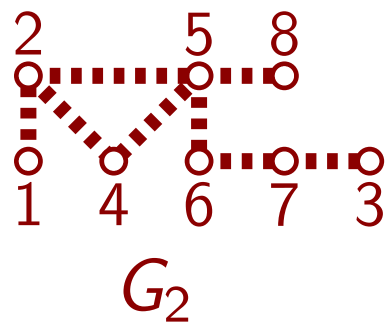
- Sim. Geometric Emb. (SGE)
- Sim. Emb. with Fixed Edges (SEFE)



Simultaneous Embedding

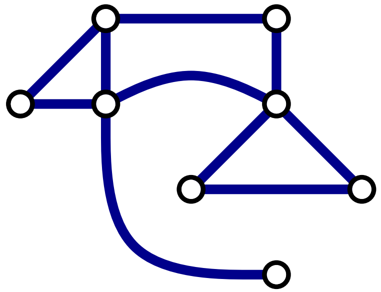
Given: Two planar graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ on same vertex set





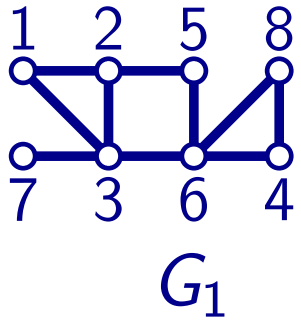
Embed both graphs in a planar way
– edges of one graph may intersect edges of the other graph

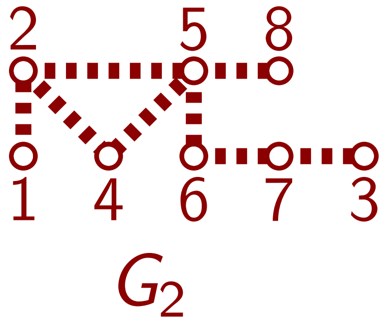
- Sim. Geometric Emb. (SGE)
- Sim. Emb. with Fixed Edges (SEFE)



Simultaneous Embedding

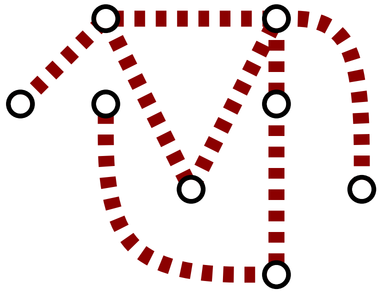
Given: Two planar graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$
on same vertex set





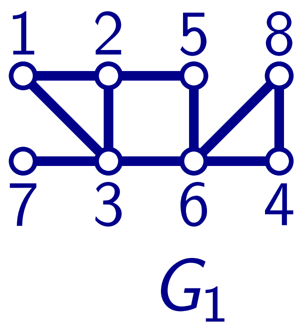
Embed both graphs in a planar way
– edges of one graph may intersect edges of the other graph

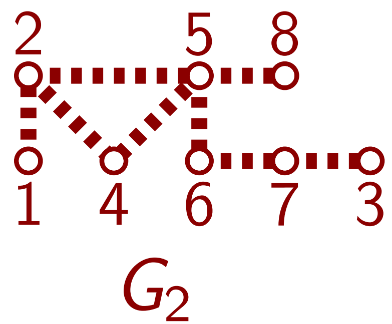
- Sim. Geometric Emb. (SGE)
- Sim. Emb. with Fixed Edges (SEFE)



Simultaneous Embedding

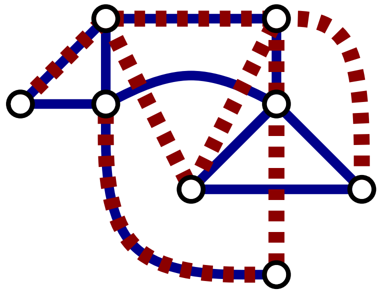
Given: Two planar graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$
on same vertex set





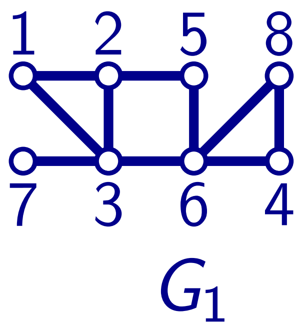
Embed both graphs in a planar way
– edges of one graph may intersect edges of the other graph

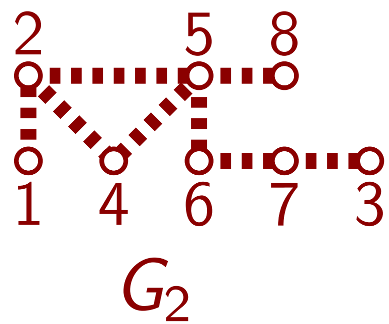
- Sim. Geometric Emb. (SGE)
- Sim. Emb. with Fixed Edges (SEFE)



Simultaneous Embedding

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



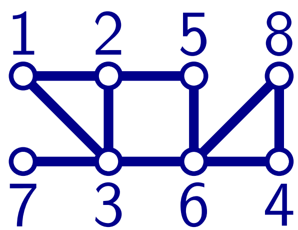


Embed both graphs in a planar way
– edges of one graph may intersect edges of the other graph

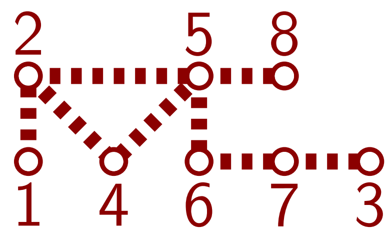
- Sim. Geometric Emb. (SGE)
- Sim. Emb. with Fixed Edges (SEFE)
- Sim. Emb. (SE)

Simultaneous Embedding

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



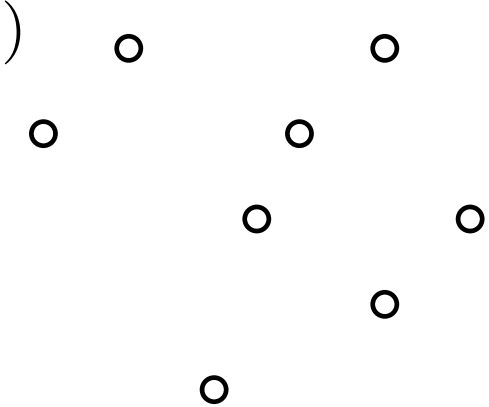
G_1



G_2

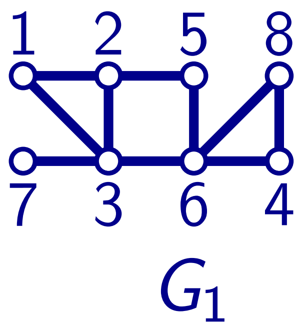
Embed both graphs in a planar way
– edges of one graph may intersect edges of the other graph

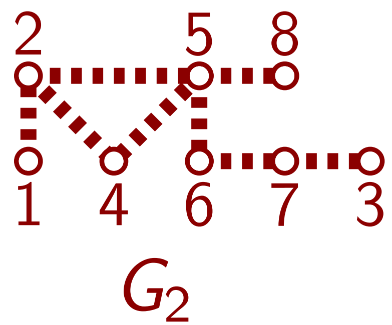
- Sim. Geometric Emb. (SGE)
- Sim. Emb. with Fixed Edges (SEFE)
- Sim. Emb. (SE)



Simultaneous Embedding

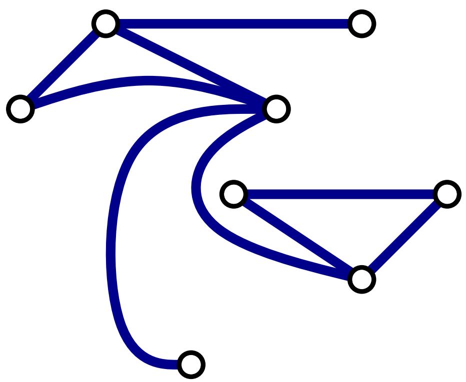
Given: Two planar graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$
on same vertex set





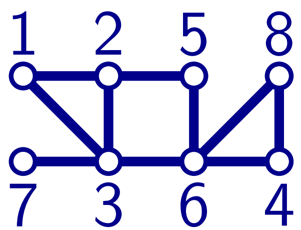
Embed both graphs in a planar way
– edges of one graph may intersect edges of the other graph

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

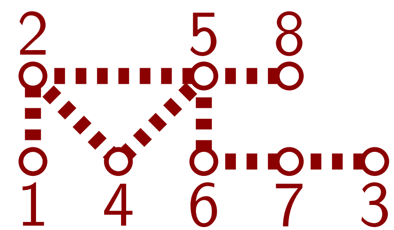


Simultaneous Embedding

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



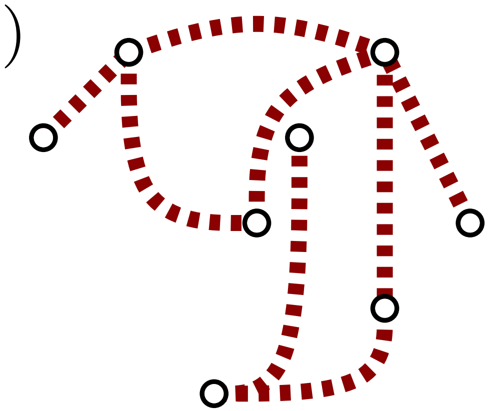
G_1



G_2

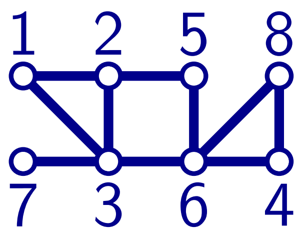
Embed both graphs in a planar way
– edges of one graph may intersect edges of the other graph

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

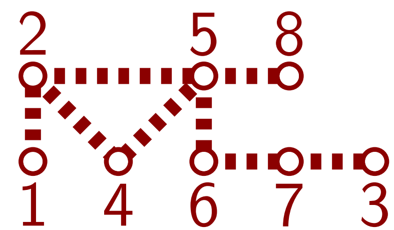


Simultaneous Embedding

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



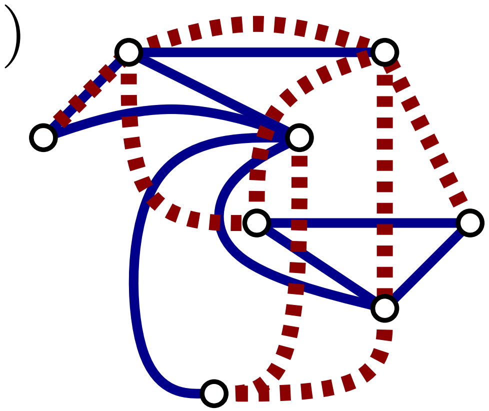
G_1



G_2

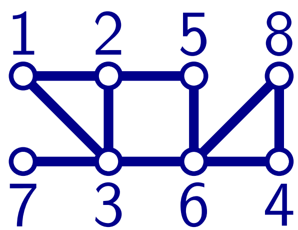
Embed both graphs in a planar way
– edges of one graph may intersect edges of the other graph

- SIM. GEOMETRIC EMB. (SGE)
- SIM. EMB. WITH FIXED EDGES (SEFE)
- SIM. EMB. (SE)

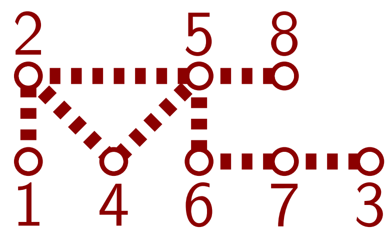


Simultaneous Embedding

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



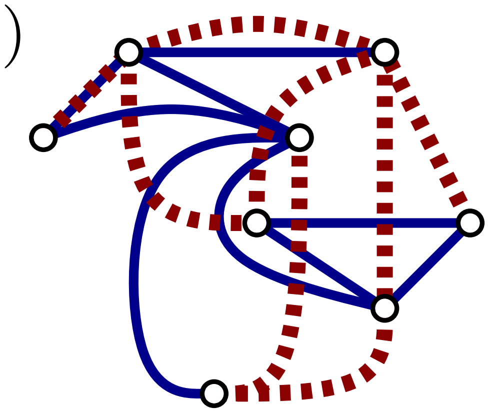
G_1



G_2

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

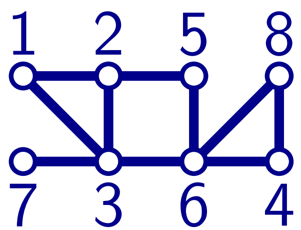
- Sim. Geometric Emb. (SGE)
- Sim. Emb. with Fixed Edges (SEFE)
- Sim. Emb. (SE)



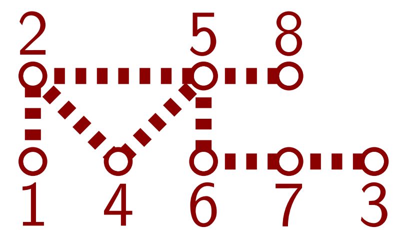
Large body of literature; see the survey
by Bläsius, Rutter & Kobourov [HGD'13]!

Simultaneous Embedding

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



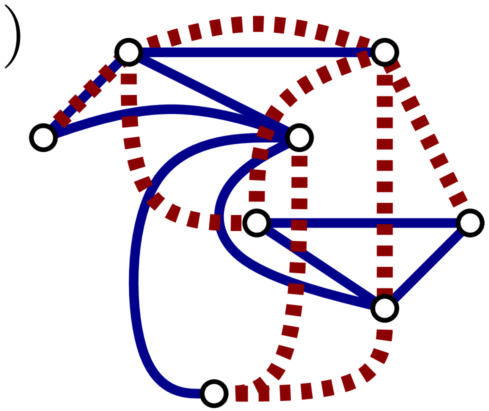
G_1



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)**



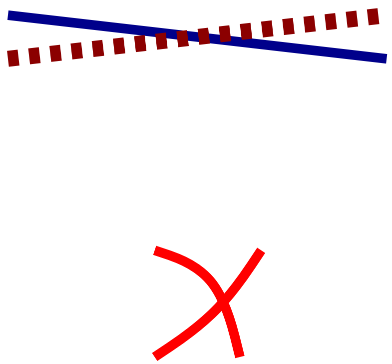
Large body of literature; see the survey by Bläsius, Rutter & Kobourov [HGD'13]!

RacSim Drawings

RacSim: Simultaneous Embedding with Right-Angle Crossings

RacSim Drawings

RacSim: Simultaneous Embedding with Right-Angle Crossings

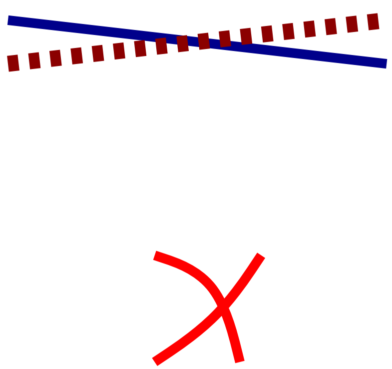


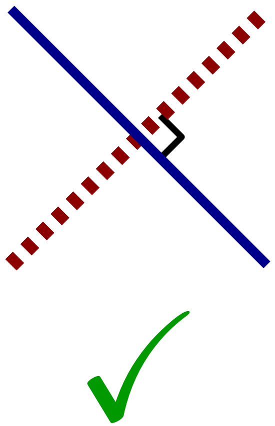
RacSim Drawings

RacSim: Simultaneous Embedding with Right-Angle Crossings

RacSim Drawings

RacSim: Simultaneous Embedding with Right-Angle Crossings





Geometric RacSim: RacSim with straight-line edges

A Few Known Results

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

A Few Known Results

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

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

A Few Known Results

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

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

[Erten & Kobourov JGAA'05]
Any pair of trees admits an SE drawing
with 1 bend and quadratic area.

A Few Known Results

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

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

[Erten & Kobourov JGAA'05]
Any pair of trees admits an SE drawing
with 1 bend and quadratic area.

[Kammer SWAT'06]
Any pair of planar graphs admits an SE drawing
with 2 bends and quadratic area.

A Few Known Results

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

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

[Erten & Kobourov JGAA'05]
Any pair of trees admits an SE drawing
with 1 bend and quadratic area.

[Kammer SWAT'06]
Any pair of planar graphs admits an SE drawing
with 2 bends and quadratic area.

} small angles

A Few Known Results

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

[Erten & Kobourov JGAA'05]
Any pair of trees admits an SE drawing with 1 bend and quadratic area.

[Kammer SWAT'06]
Any pair of planar graphs admits an SE drawing with 2 bends and quadratic area.

} small angles

A Few Known Results

[Argyrious et al. JGAA'13]

A cycle and a matching always admit a geometric RacSim drawing in quadratic area.

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

[Erten & Kobourov JGAA'05]

Any pair of trees admits an SE drawing with 1 bend and quadratic area.

[Kammer SWAT'06]

Any pair of planar graphs admits an SE drawing with 2 bends and quadratic area.

} small angles

A Few Known Results

[Argyrious et al. JGAA'13]

A cycle and a matching always admit a geometric RacSim drawing in quadratic area.

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

[Frati, Hoffmann, Kusters GD'15]

New!

A Few Known Results

[Argyrious et al. JGAA'13]

A cycle and a matching always admit a geometric RacSim drawing in quadratic area.

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

[Frati, Hoffmann, Kusters GD'15]

New!

- Two trees admit a $(1, 4)$-Sefe (i.e., 1 bend per edge and 4 crossings per edge pair).

A Few Known Results

[Argyrious et al. JGAA'13]

A cycle and a matching always admit a geometric RacSim drawing in quadratic area.

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

New!

[Frati, Hoffmann, Kusters GD'15]

- Two trees admit a $(1, 4)$-Sefe (i.e., 1 bend per edge and 4 crossings per edge pair).
- A tree and a planar graph admit a $(6, 8)$-Sefe.

A Few Known Results

[Argyrious et al. JGAA'13]

A cycle and a matching always admit a geometric RacSim drawing in quadratic area.

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

[Frati, Hoffmann, Kusters GD'15]

New!

- Two trees admit a $(1, 4)$-Sefe (i.e., 1 bend per edge and 4 crossings per edge pair).
- A tree and a planar graph admit a $(6, 8)$-Sefe.
- Two planar graphs admit a $(6, 16)$-Sefe.

Our Results for SE

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 for SE

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

Bend complexity can be seen as a measure that shows how difficult it is to simultaneously embed two graphs.

Our Results for SE

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

Bend complexity can be seen as a measure that shows how difficult it is to simultaneously embed two graphs.

Path × Path

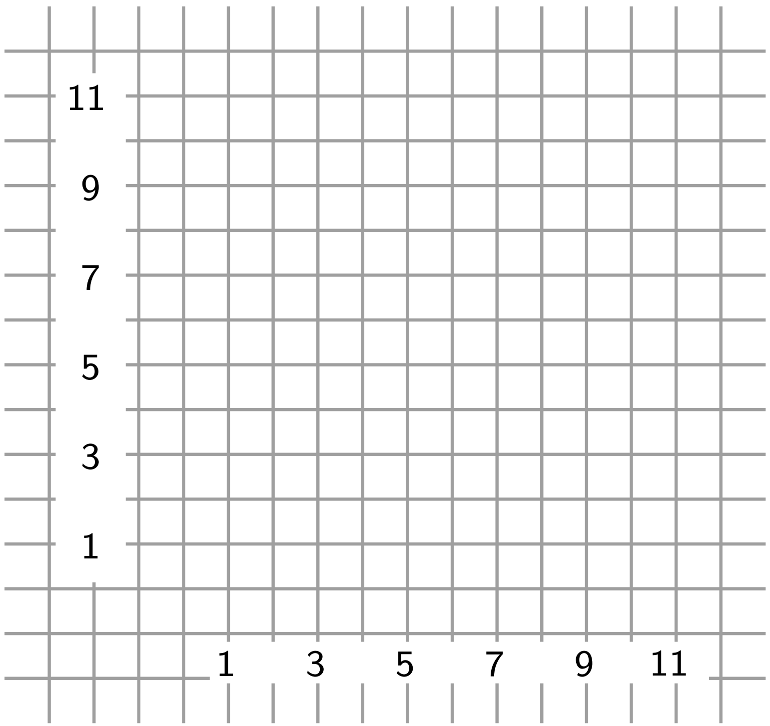
Path $\times$ Path

Path × Path

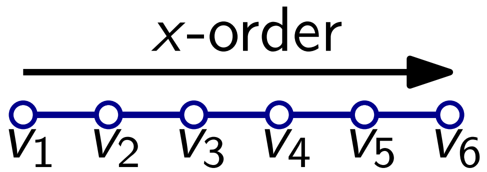
Path × Path

Path × Path

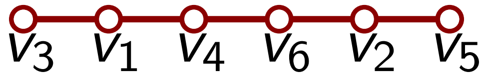
Path × Path



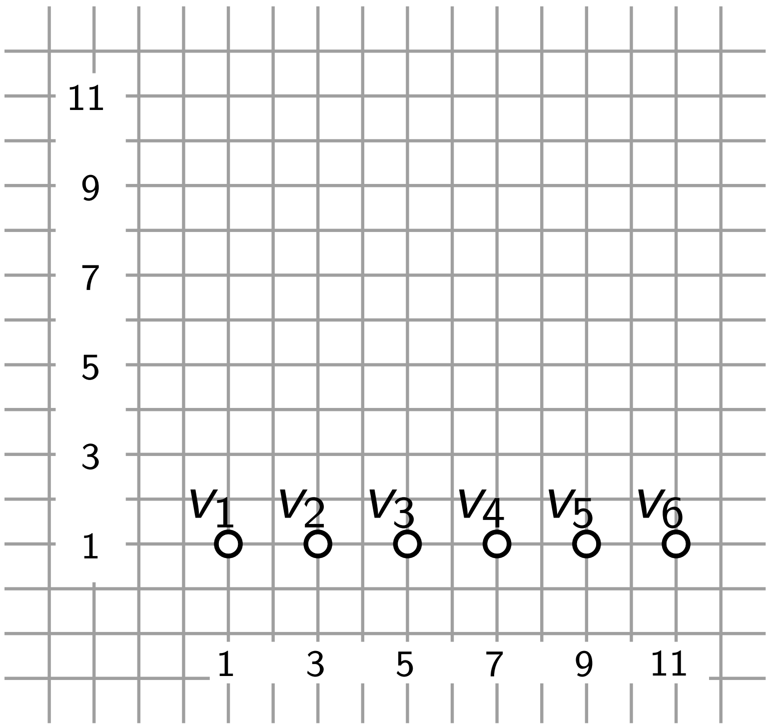
Path × Path



x-order
v_1 v_2 v_3 v_4 v_5 v_6

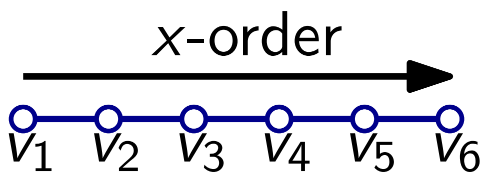


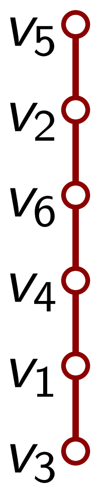
v_3 v_1 v_4 v_6 v_2 v_5

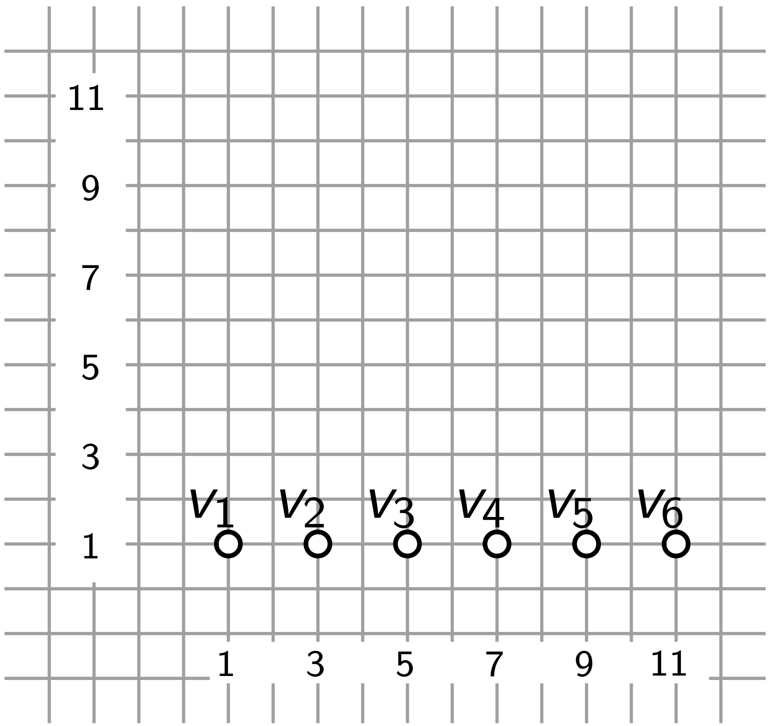


11
9
7
5
3
1
v_1 v_2 v_3 v_4 v_5 v_6
1 3 5 7 9 11

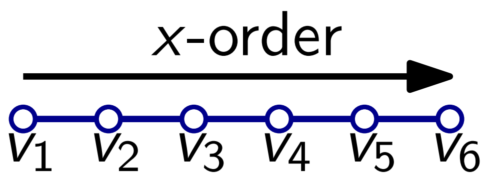
Path × Path


x-order
v1
v2
v3
v4
v5
v6

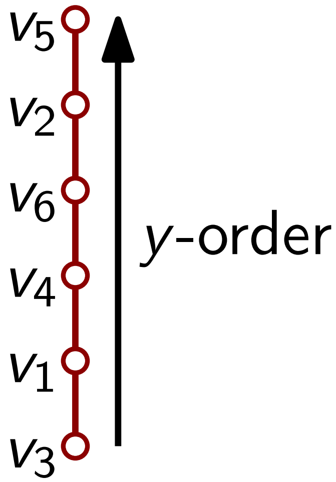

v5
v2
v6
v4
v1
v3


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

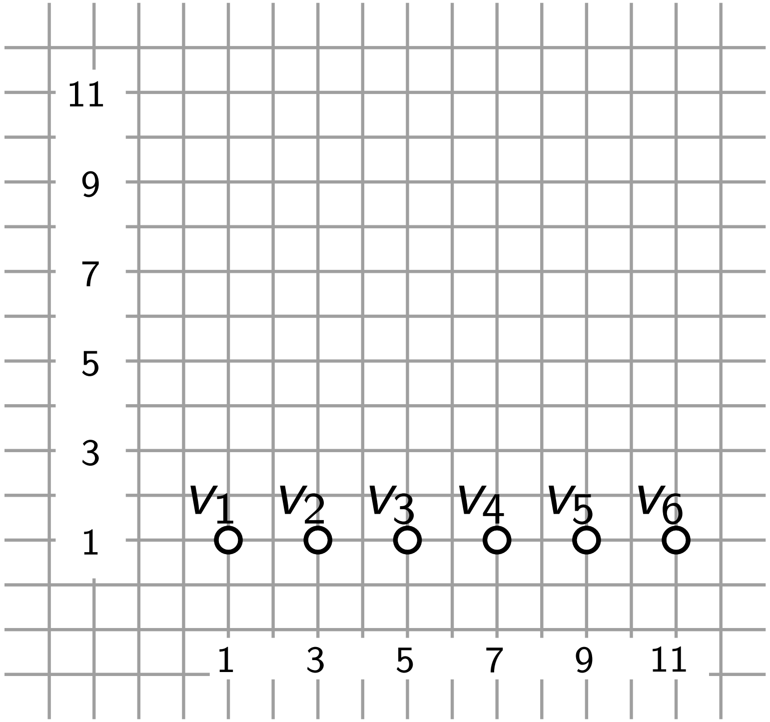
Path × Path



x-order
v1
v2
v3
v4
v5
v6

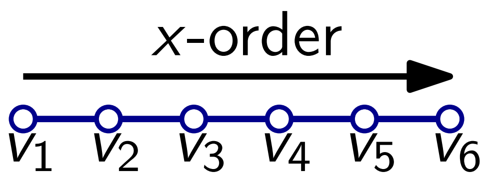


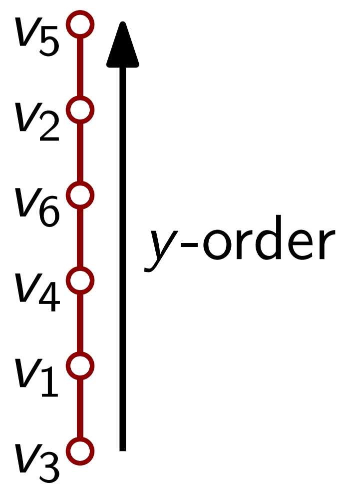
v5
v2
v6
v4
v1
v3
y-order

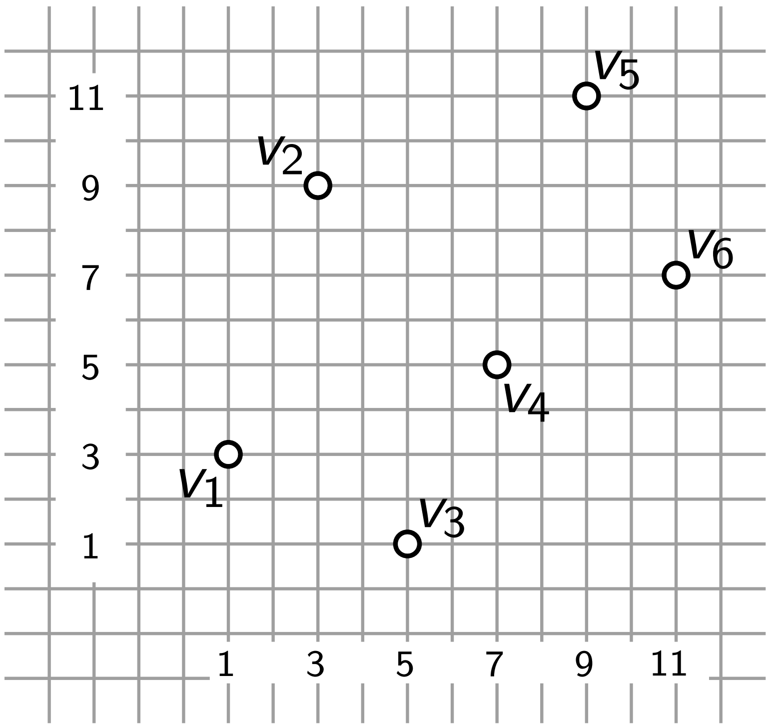


11
9
7
5
3
1
v1
v2
v3
v4
v5
v6
1
3
5
7
9
11

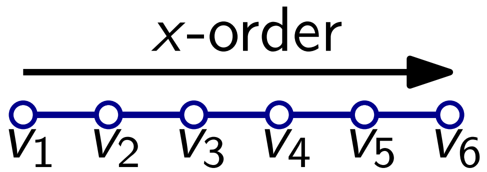
Path × Path

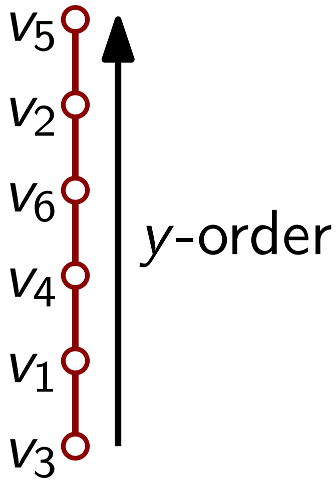

x-order
v_1 v_2 v_3 v_4 v_5 v_6


v_5
v_2
v_6
v_4
v_1
v_3
y-order

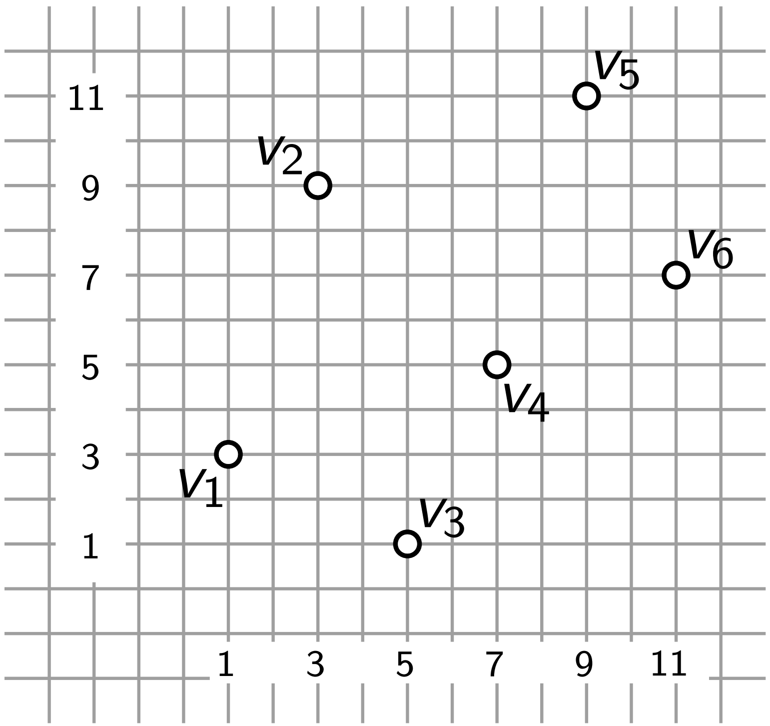

11
9
7
5
3
1
1 3 5 7 9 11
v_1
v_2
v_3
v_4
v_5
v_6

Path × Path

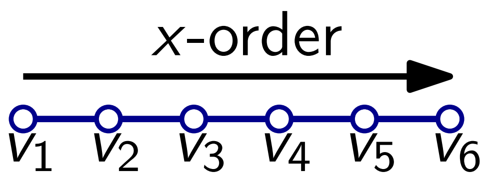


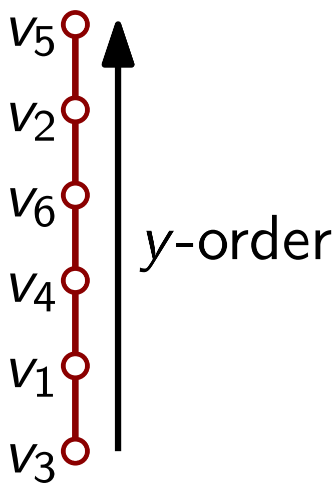


Edges:

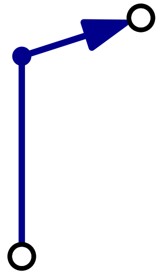


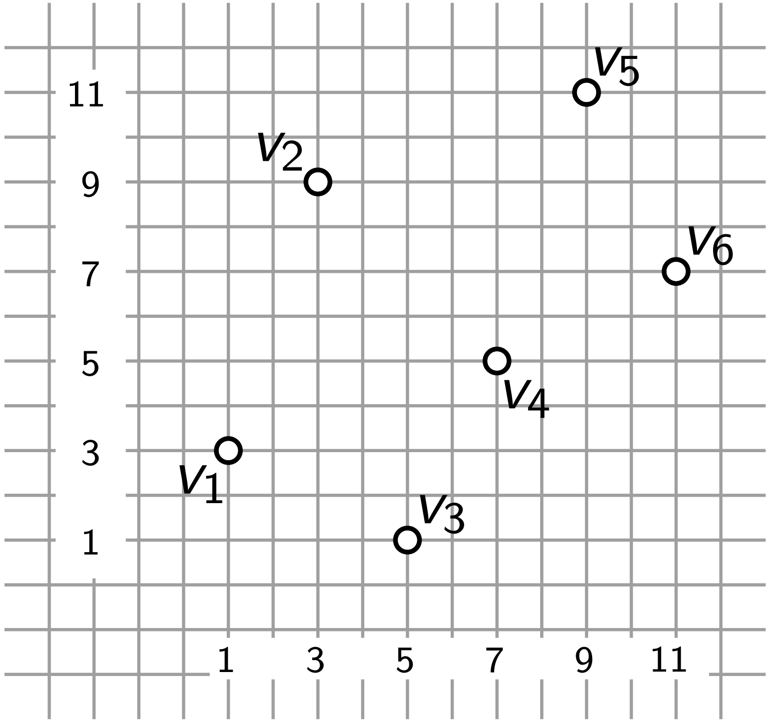
Path × Path



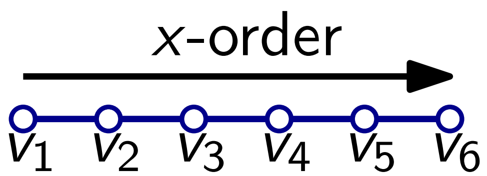


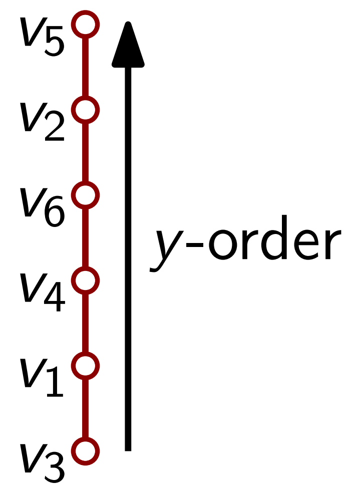
Edges:



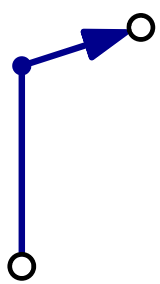


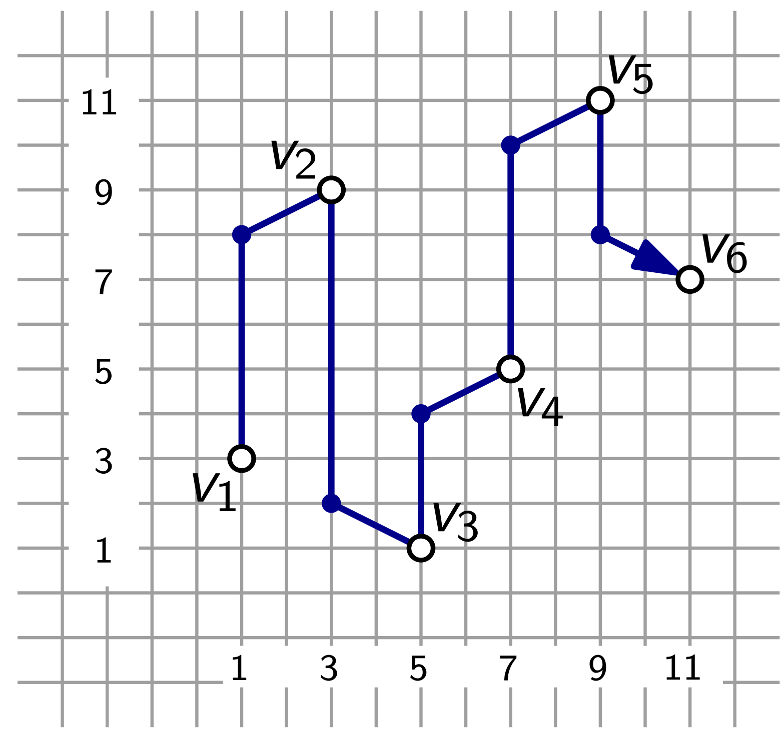
Path × Path



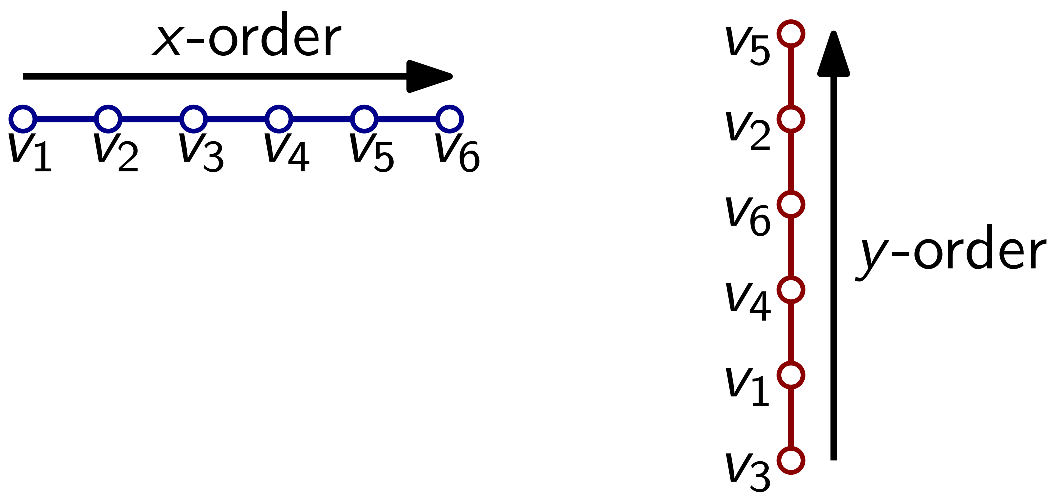


Edges:

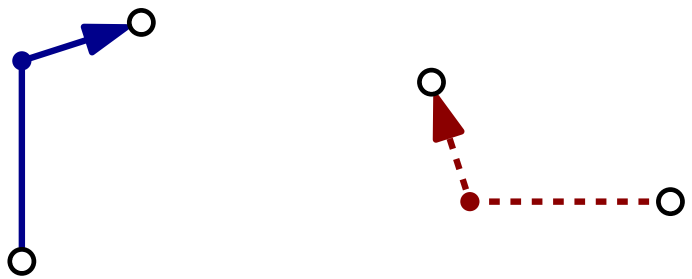


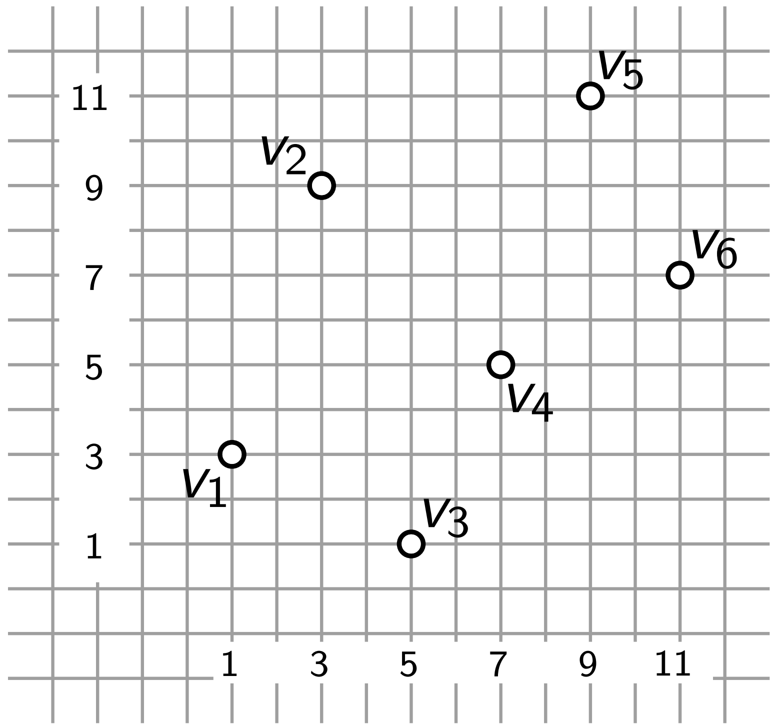


Path × Path

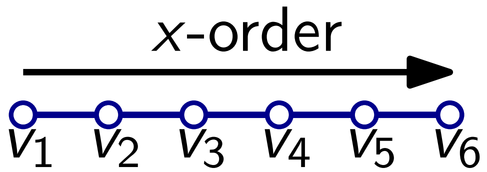


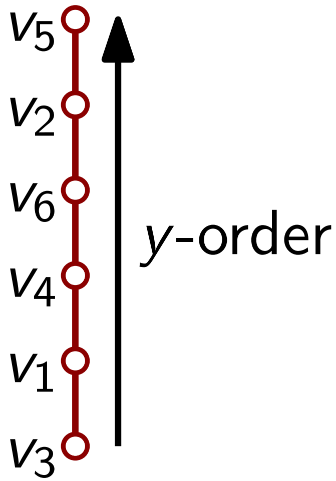
Edges:



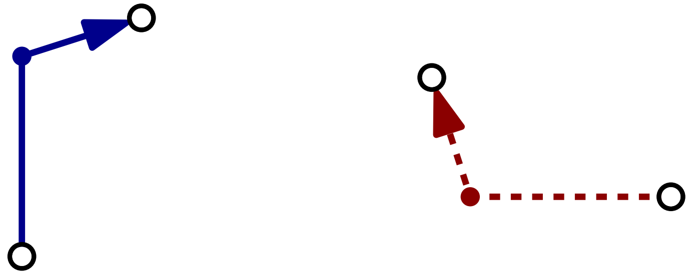


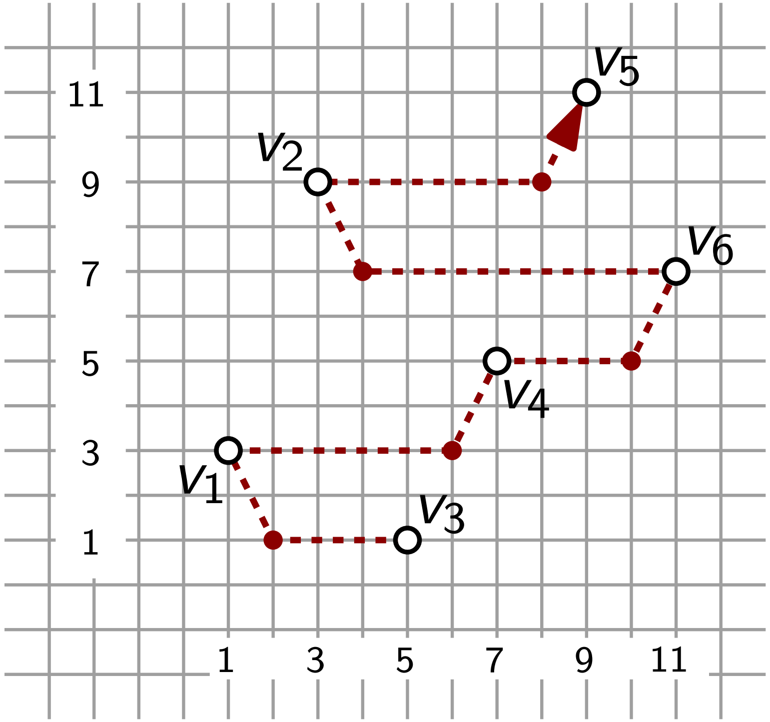
Path × Path



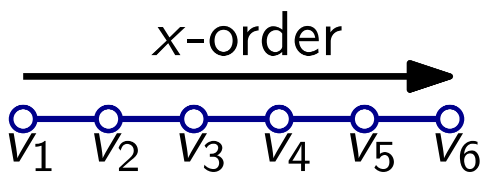


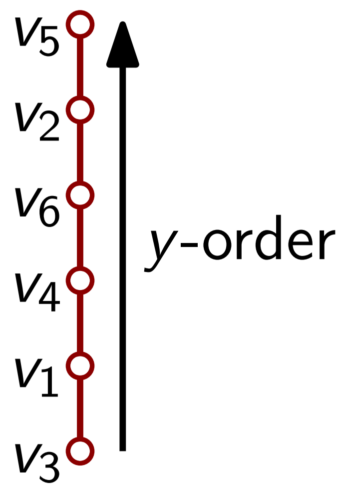
Edges:



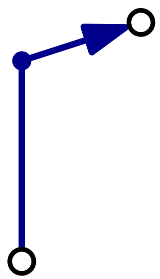


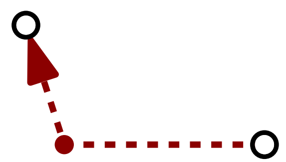
Path × Path

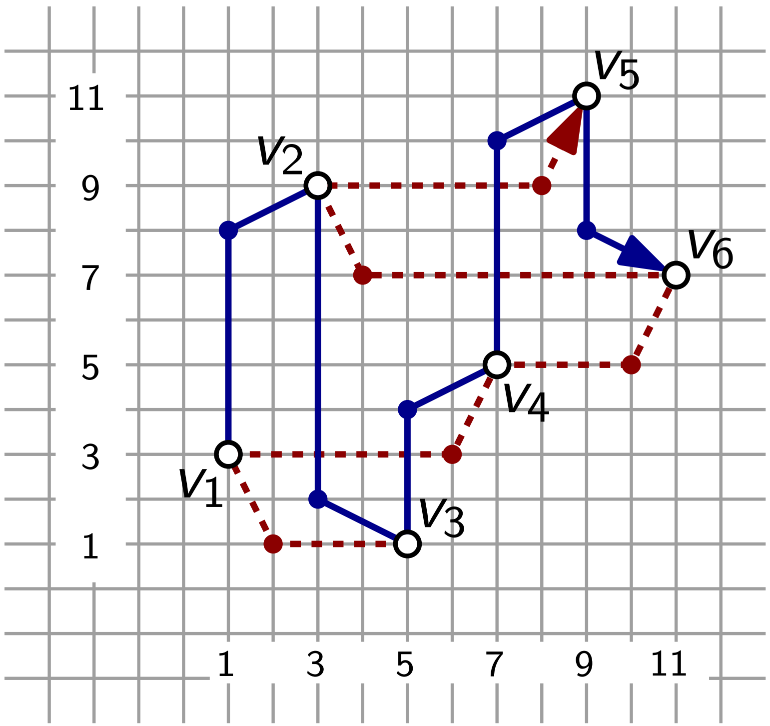




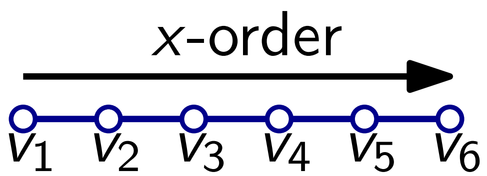
Edges:

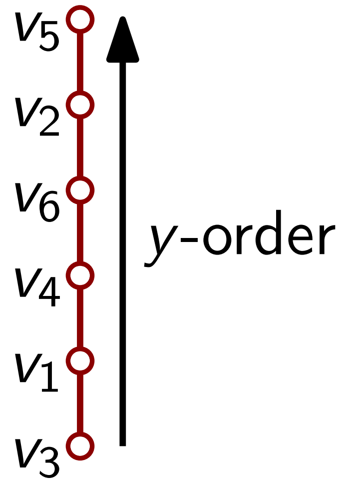




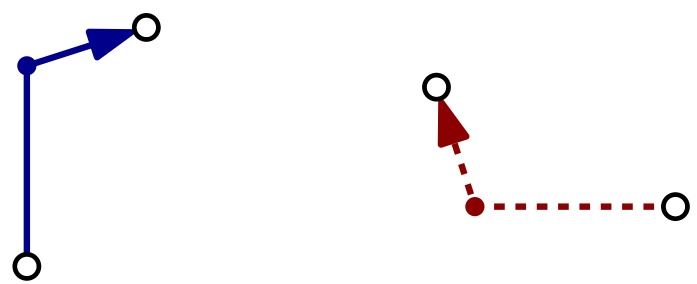


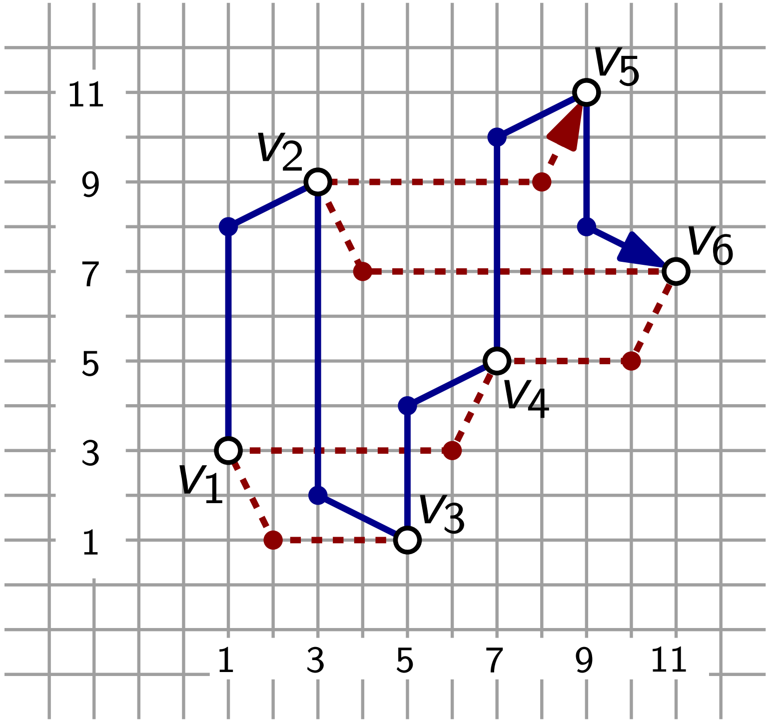
Path × Path





Edges:

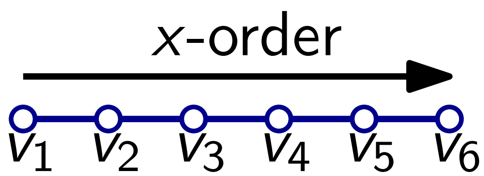


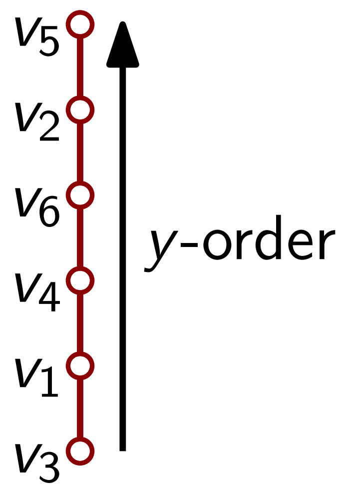


Main ideas:

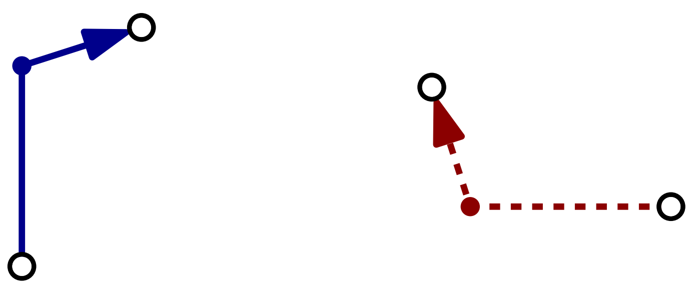
- Combine x-order and y-order.

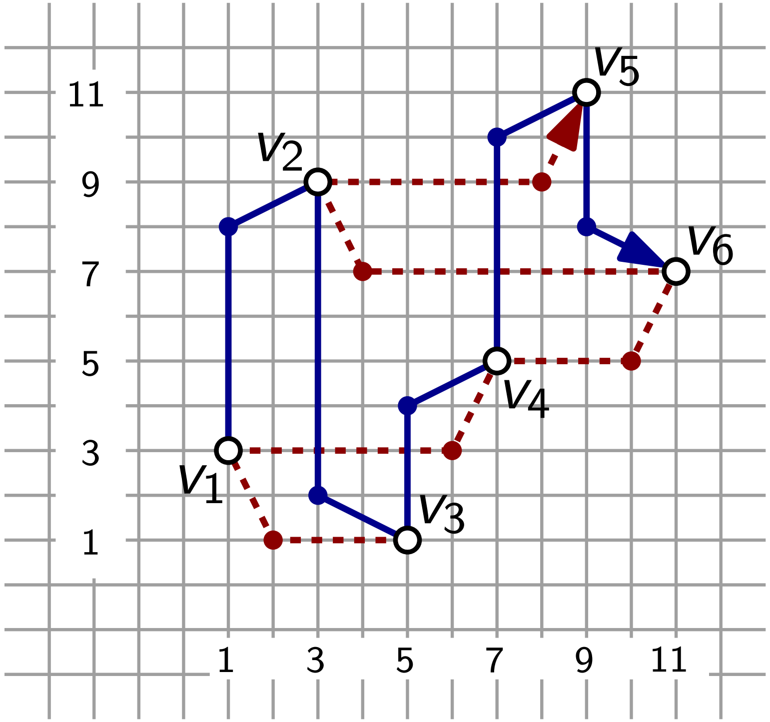
Path $\times$ Path





Edges:

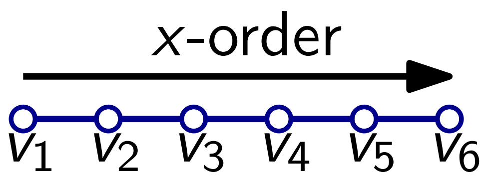


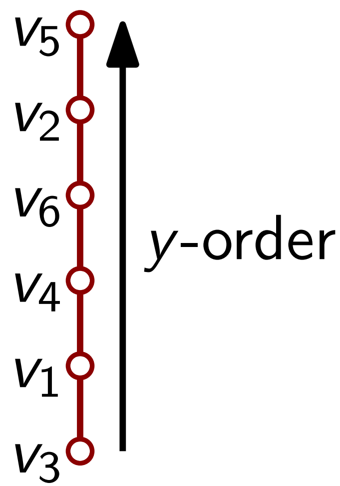


Main ideas:

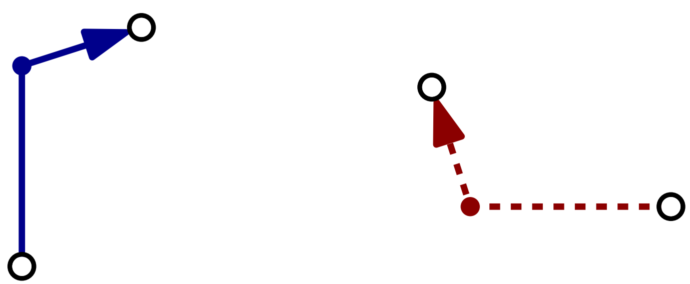
- Combine x-order and y-order.
- Keep slanted segm. short in 1 dim.

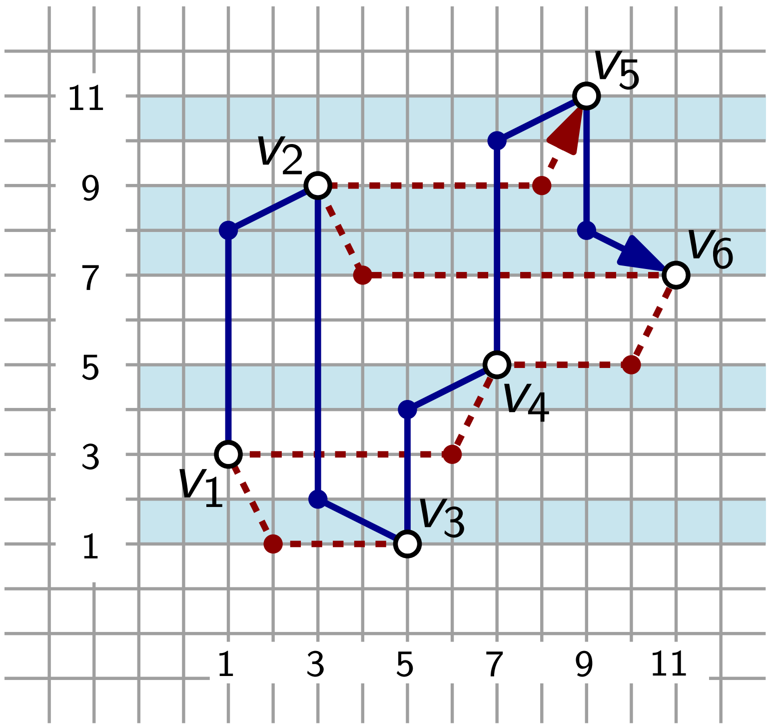
Path × Path





Edges:

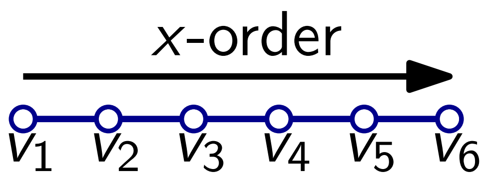


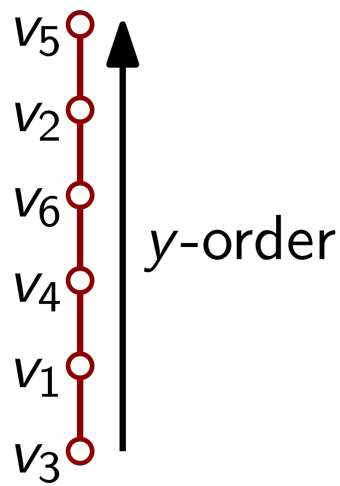


Main ideas:

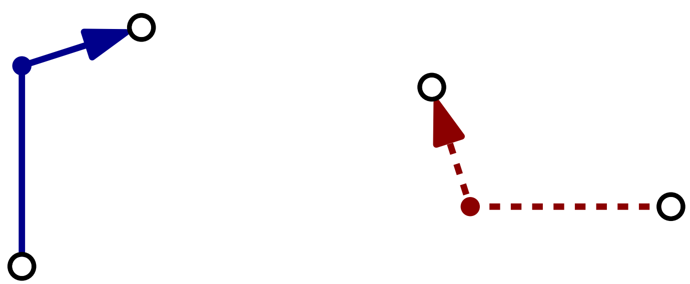
- Combine x-order and y-order.
- Keep slanted segm. short in 1 dim.

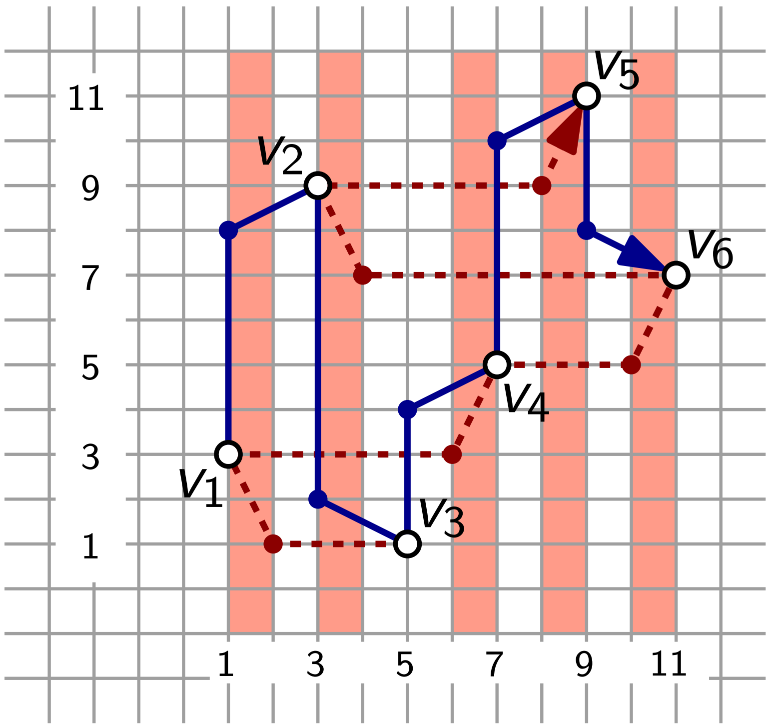
Path × Path





Edges:

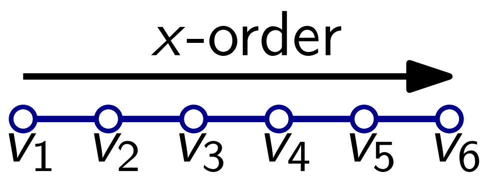


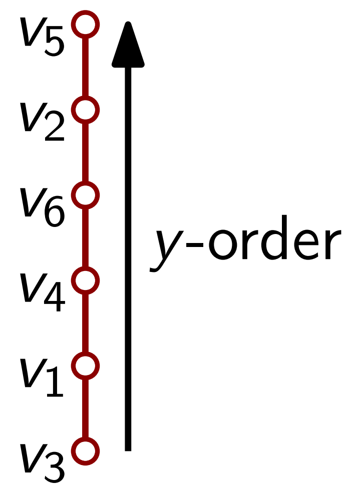


Main ideas:

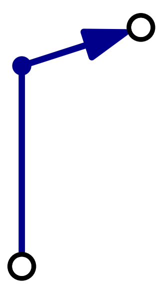
- Combine x-order and y-order.
- Keep slanted segm. short in 1 dim.

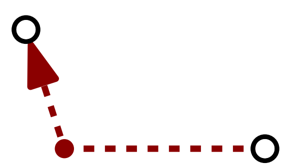
Path × Path

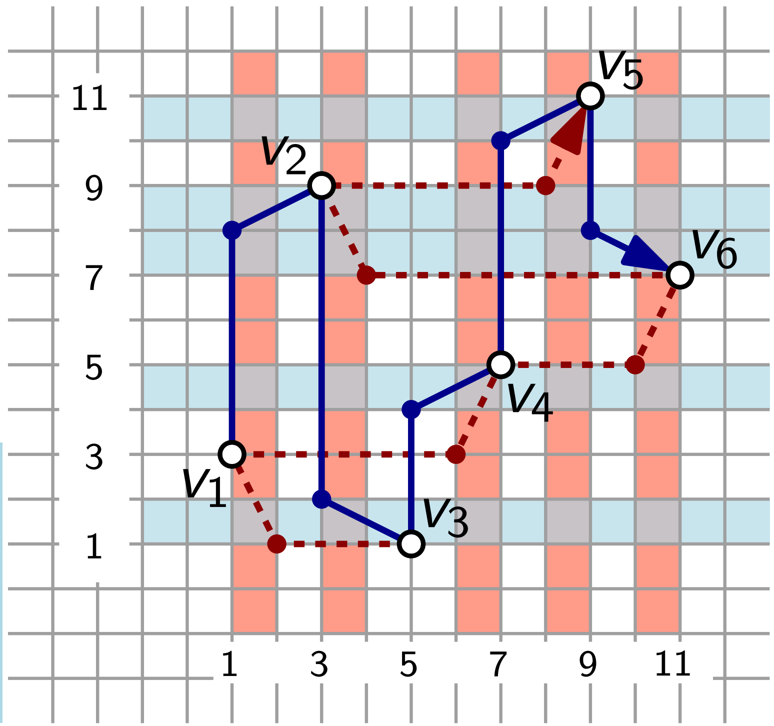




Edges:



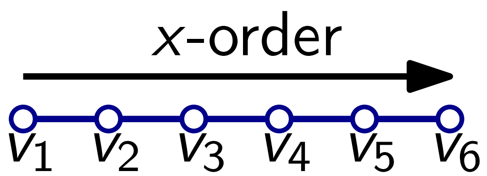


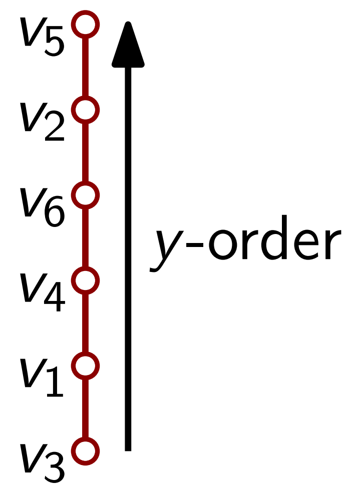


Main ideas:

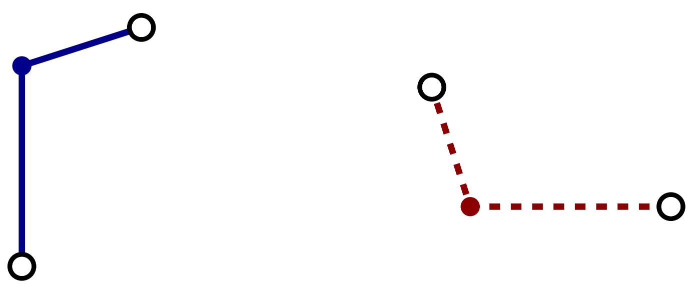
- Combine x-order and y-order.
- Keep slanted segm. short in 1 dim.

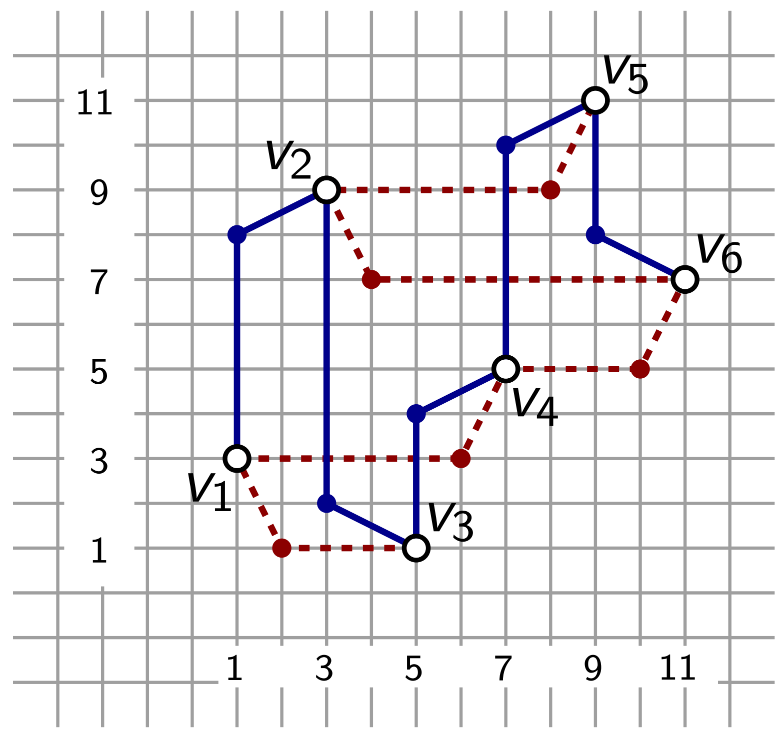
Cycle × Cycle



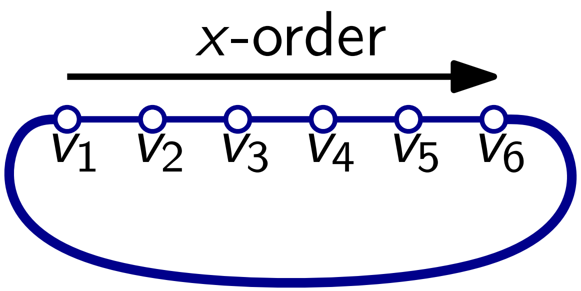


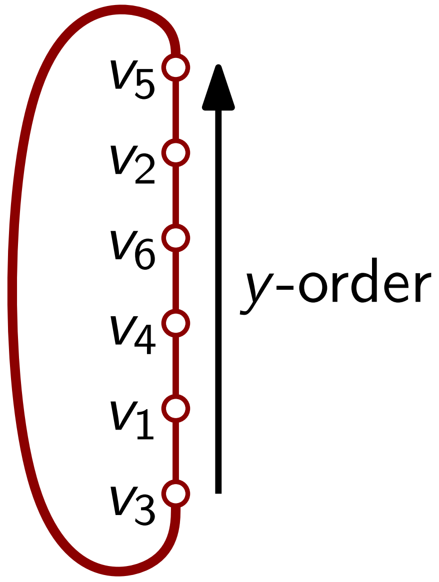
Edges:



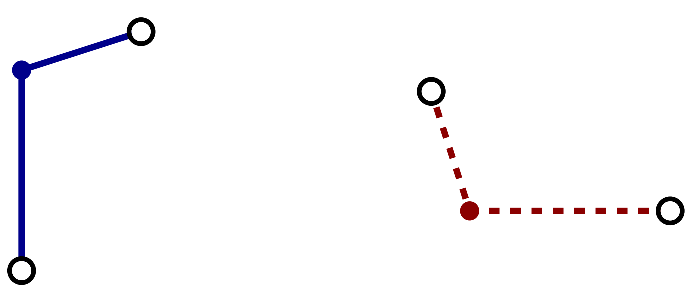


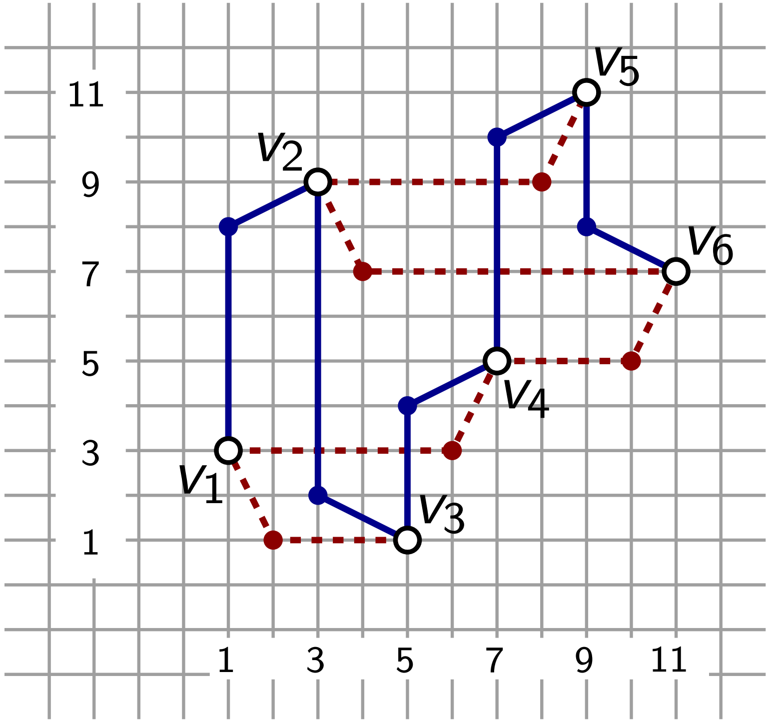
Cycle × Cycle



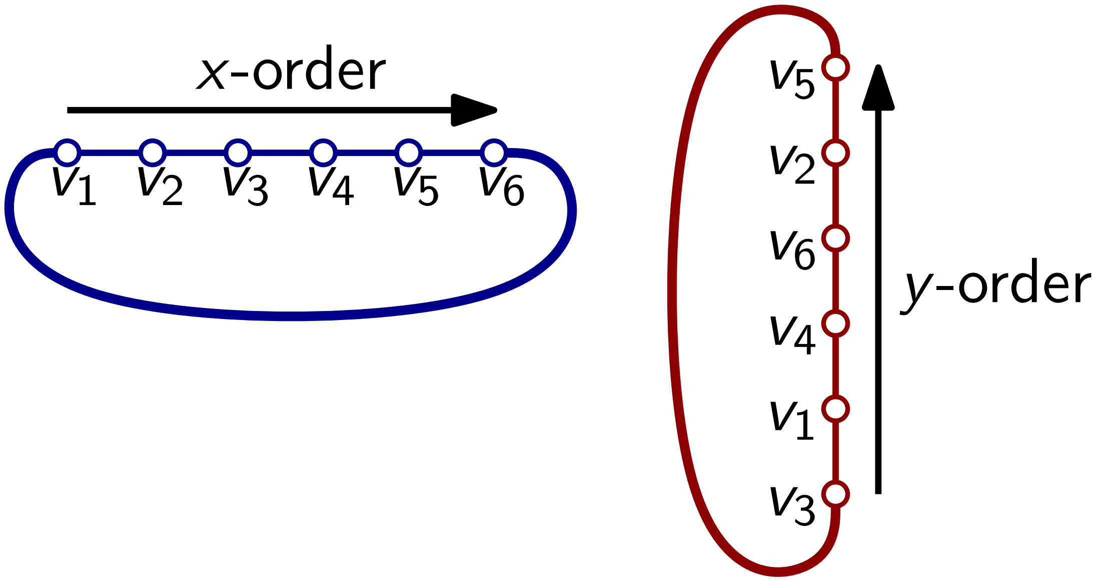


Edges:

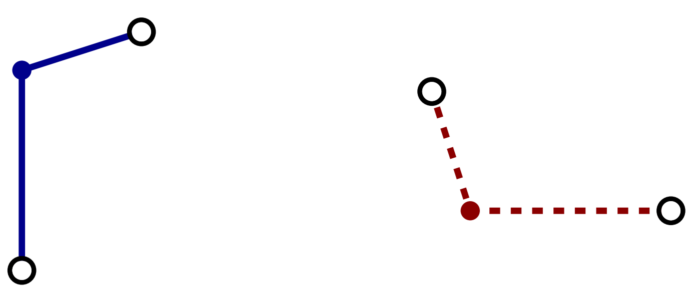


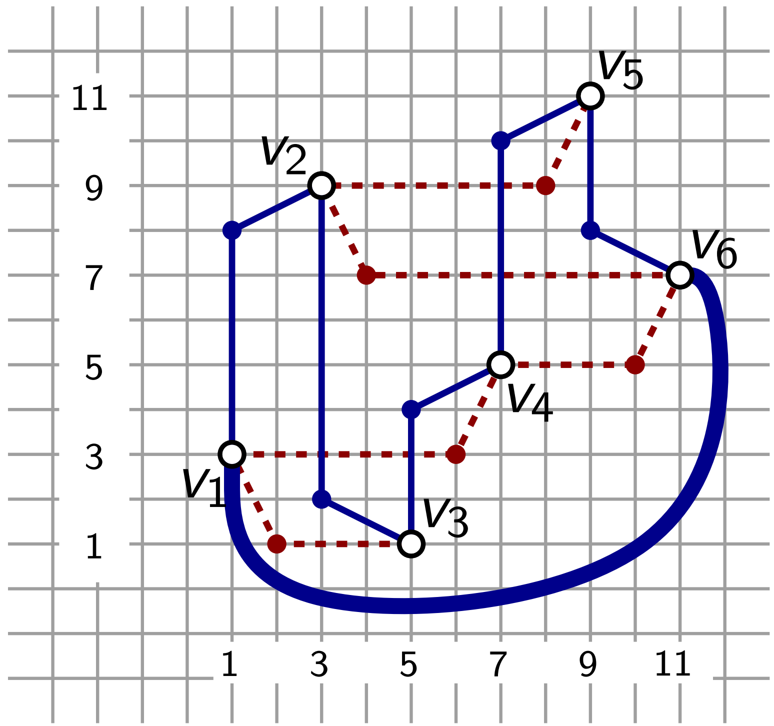


Cycle × Cycle

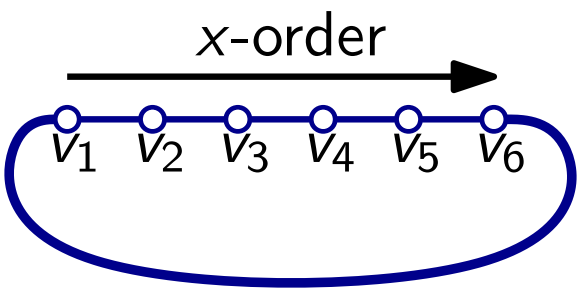


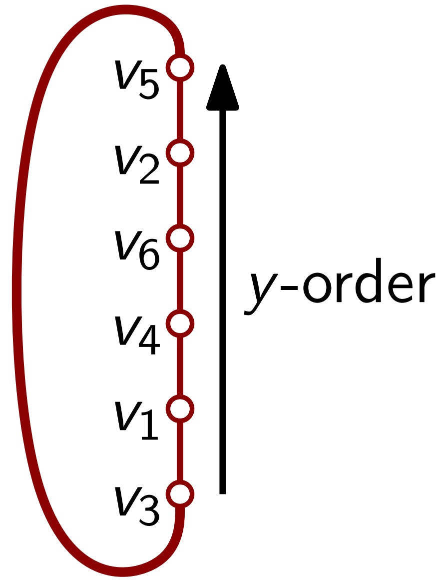
Edges:



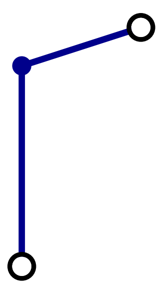


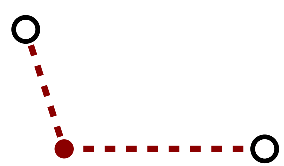
Cycle × Cycle

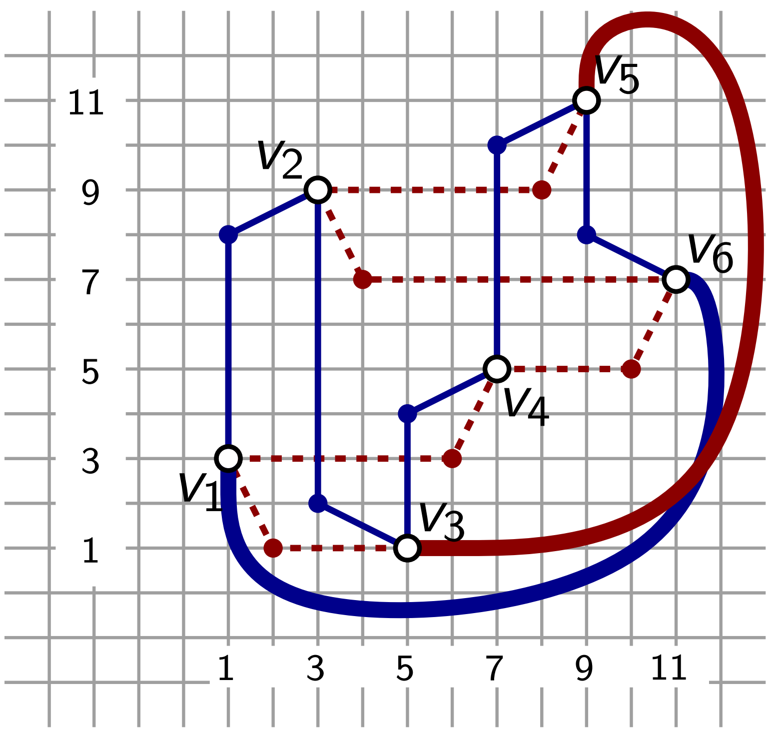




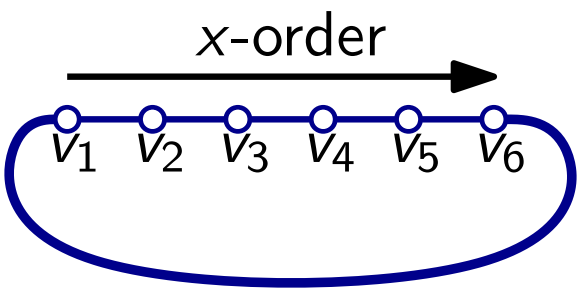
Edges:

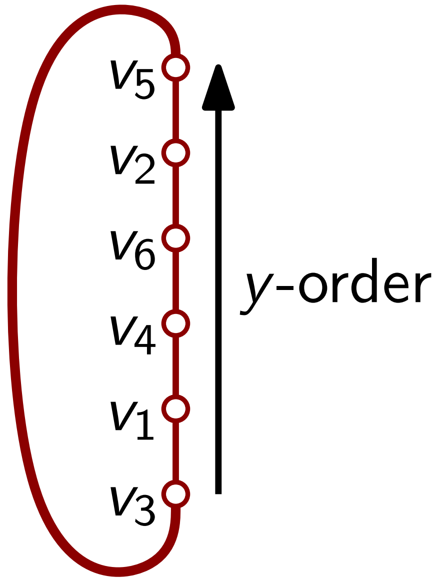




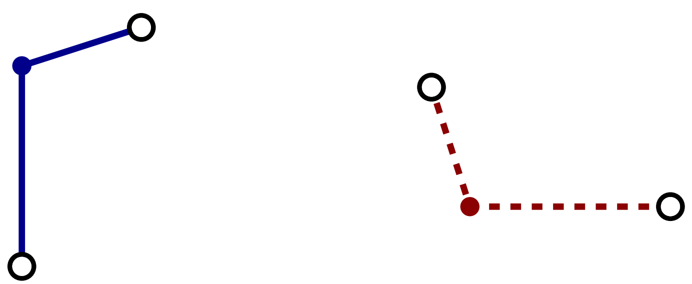


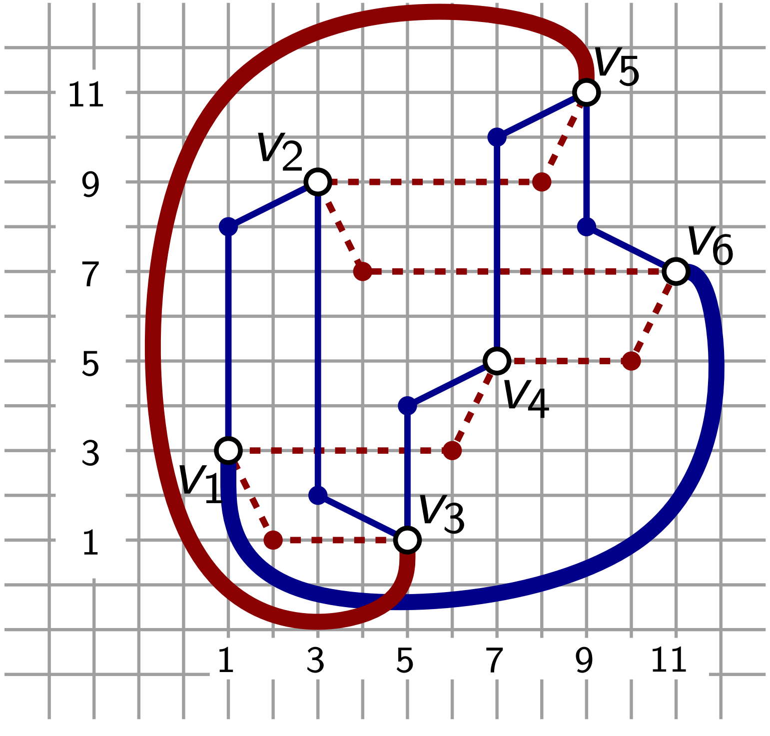
Cycle × Cycle



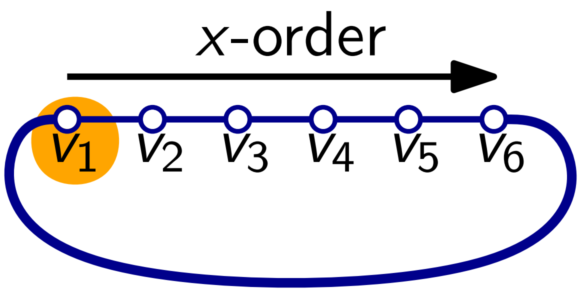


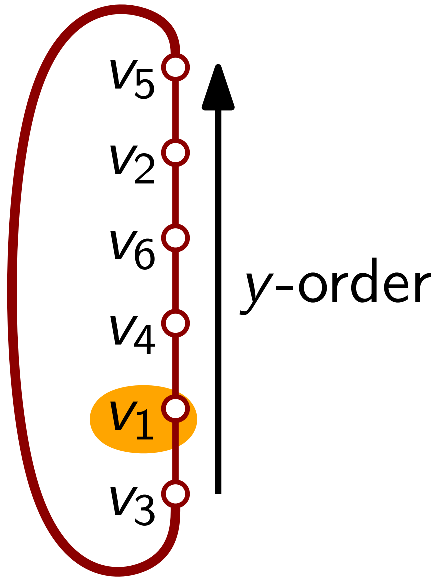
Edges:



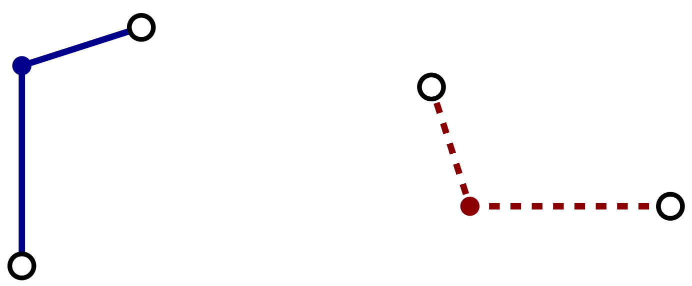


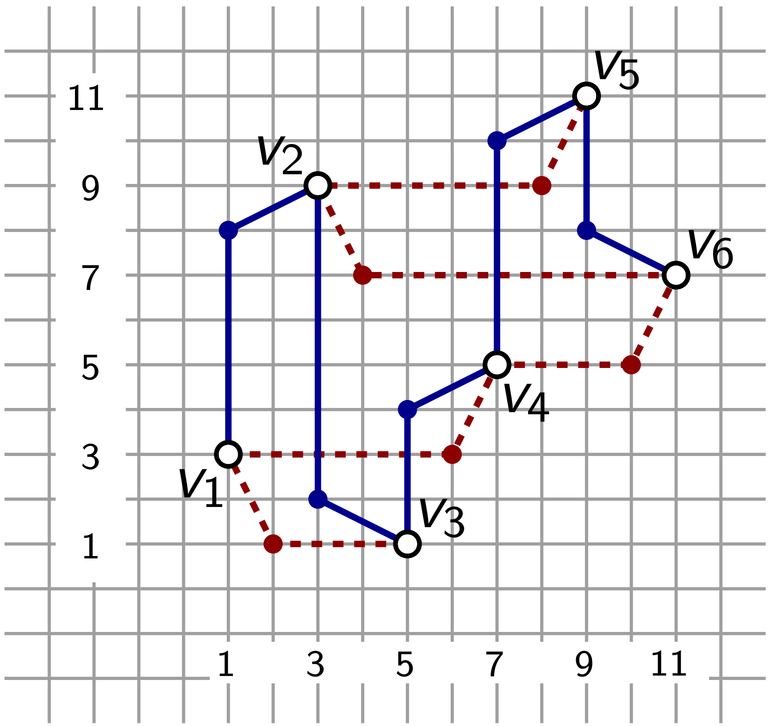
Cycle × Cycle



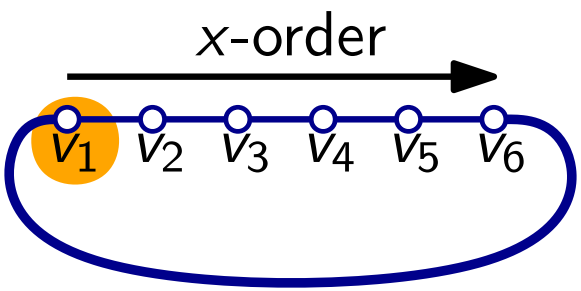


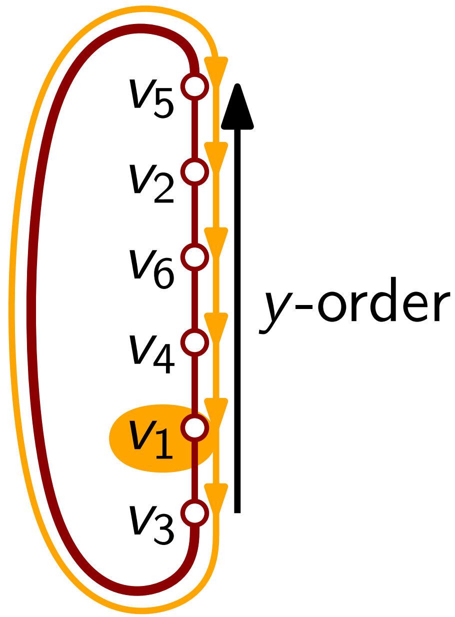
Edges:



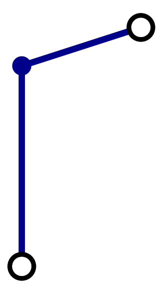


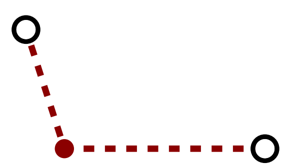
Cycle × Cycle

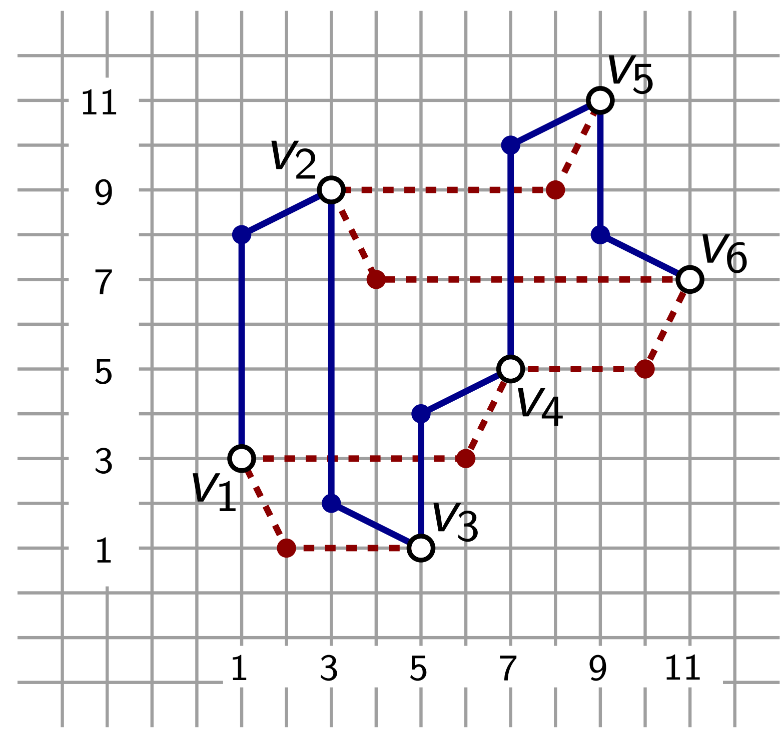




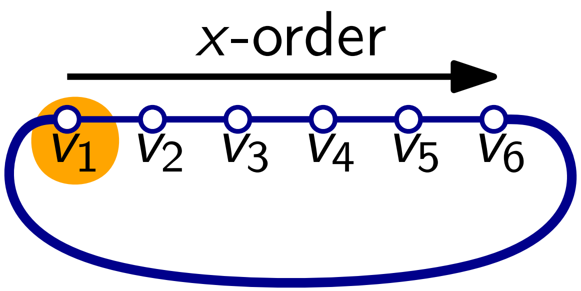
Edges:

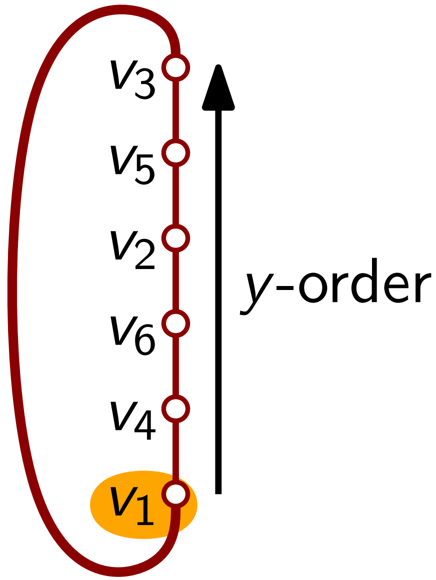




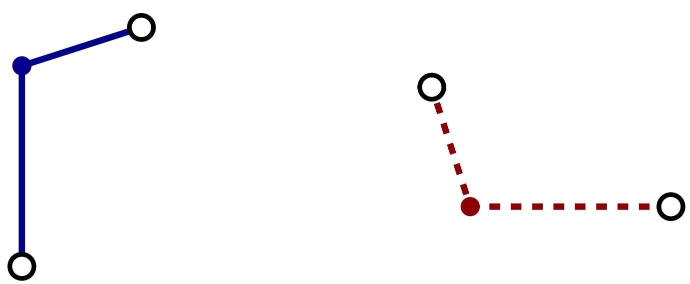


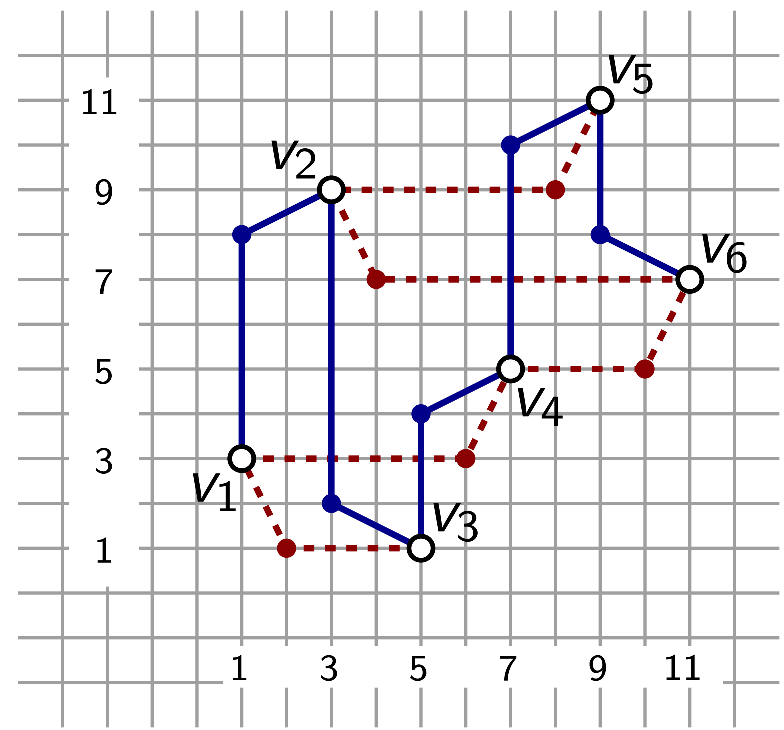
Cycle × Cycle



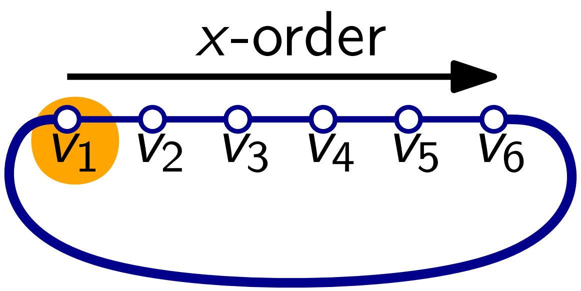


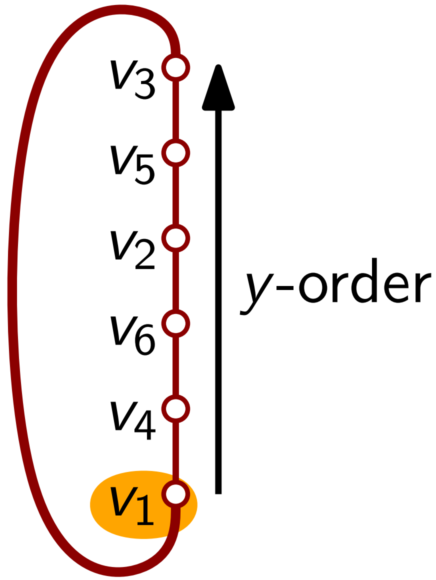
Edges:



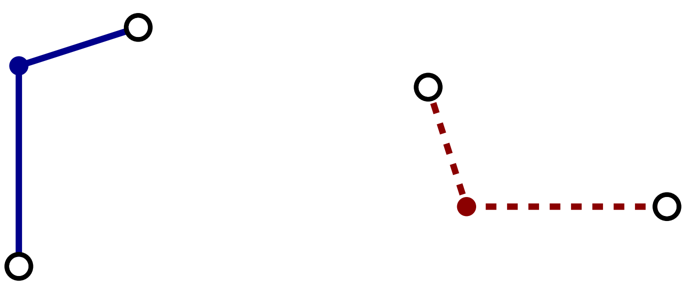


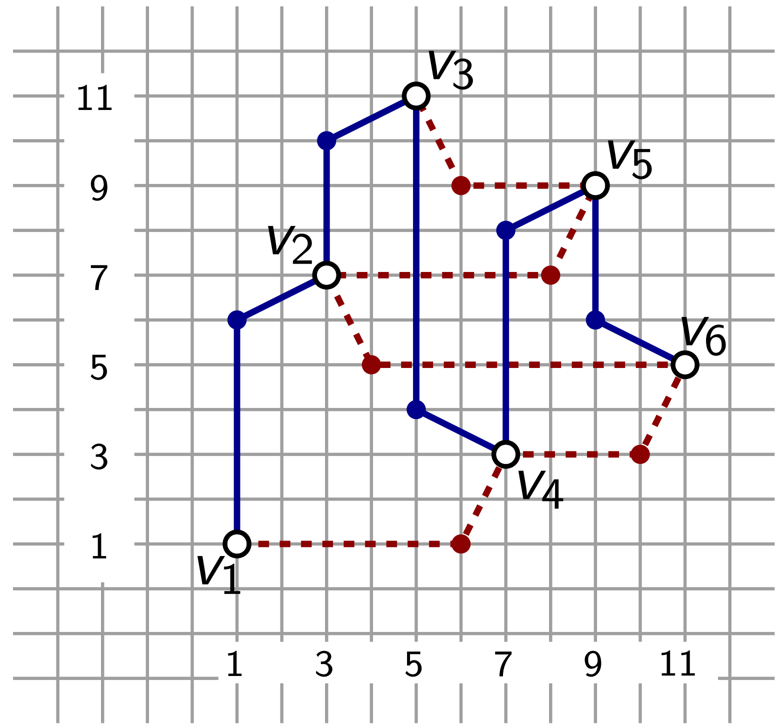
Cycle × Cycle



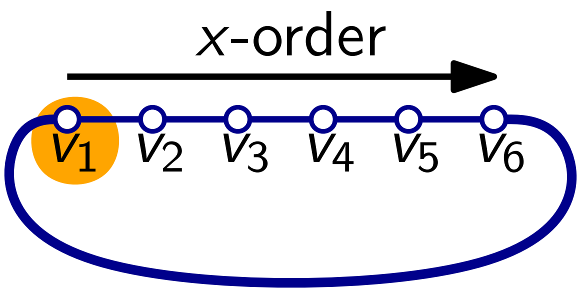


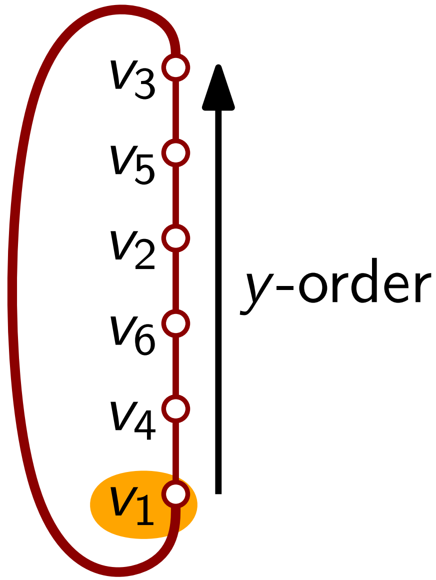
Edges:



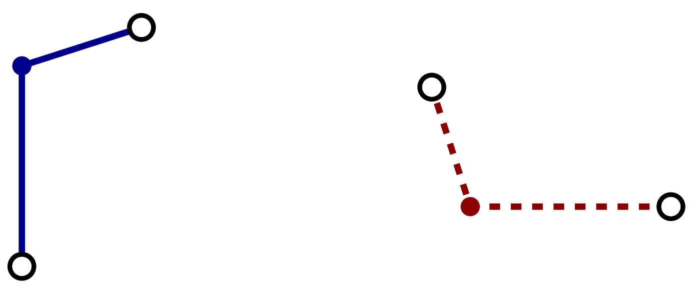


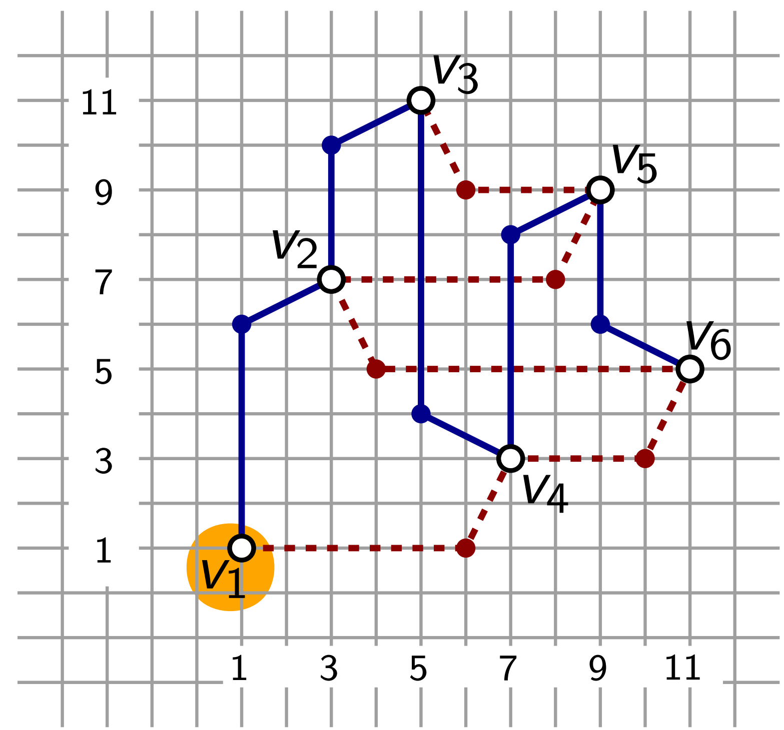
Cycle × Cycle



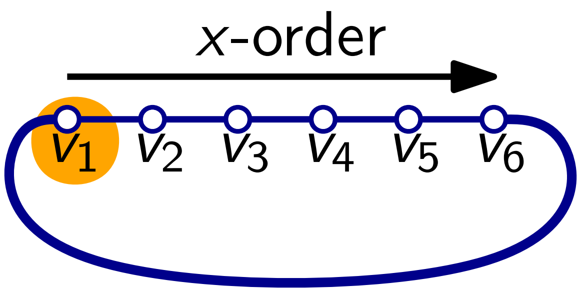


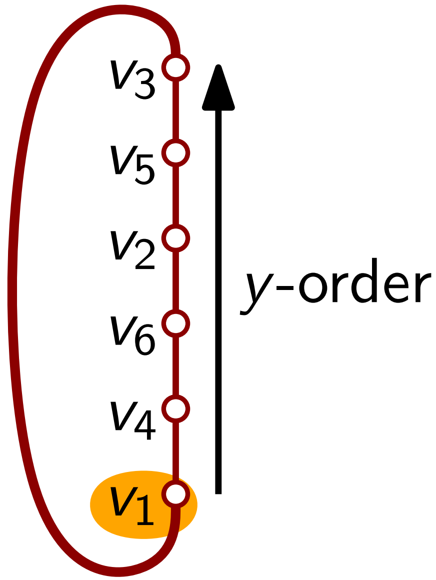
Edges:



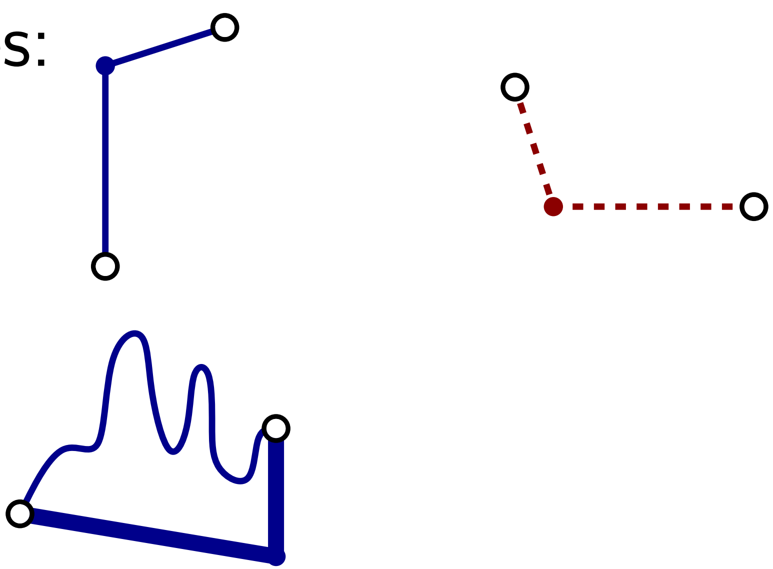


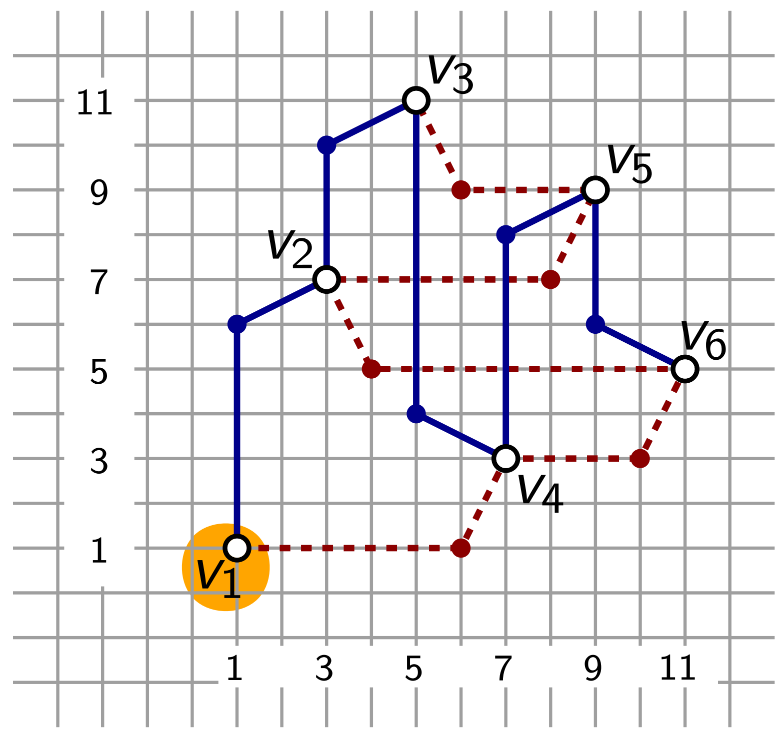
Cycle × Cycle



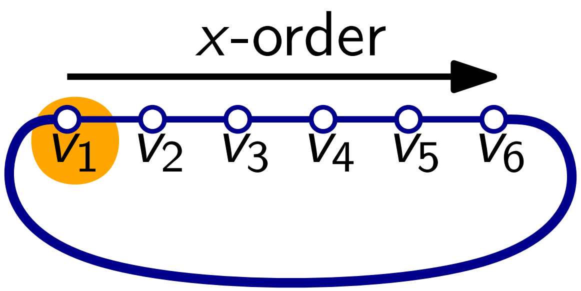


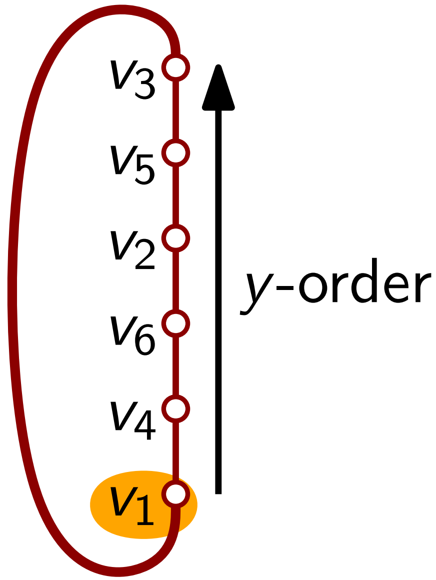
Edges:



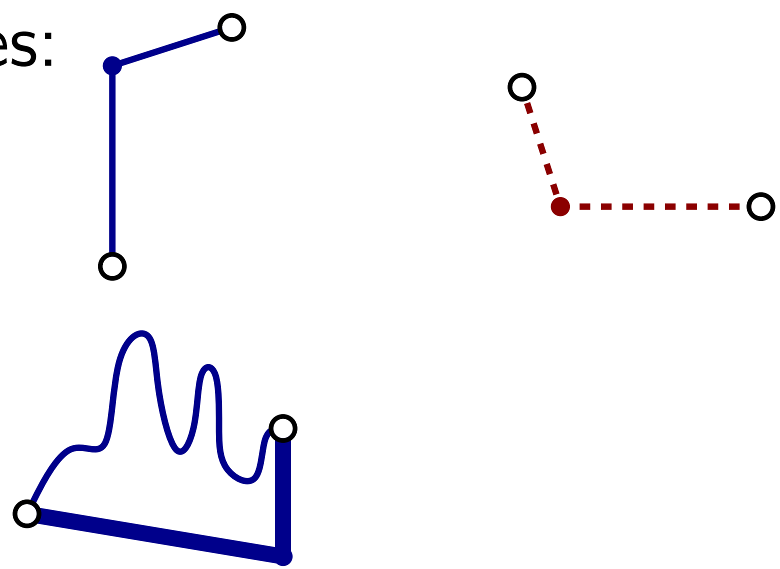


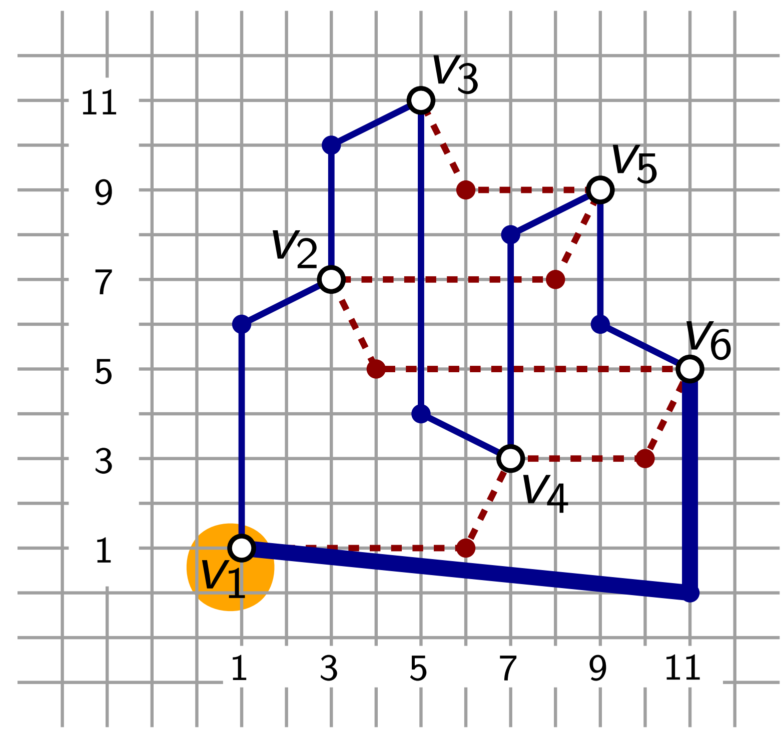
Cycle × Cycle



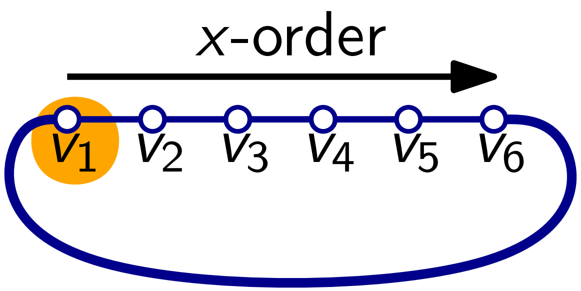


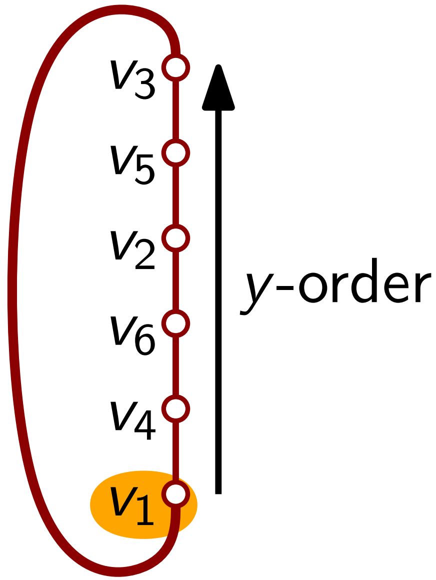
Edges:



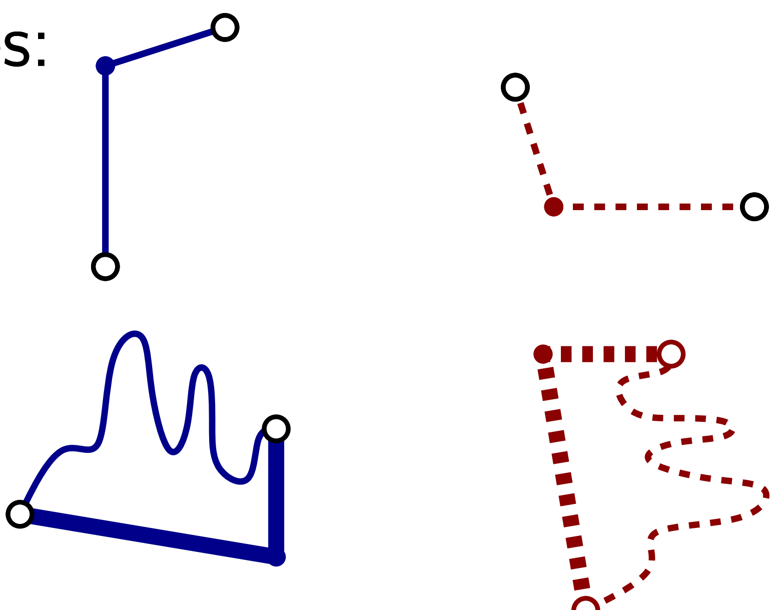


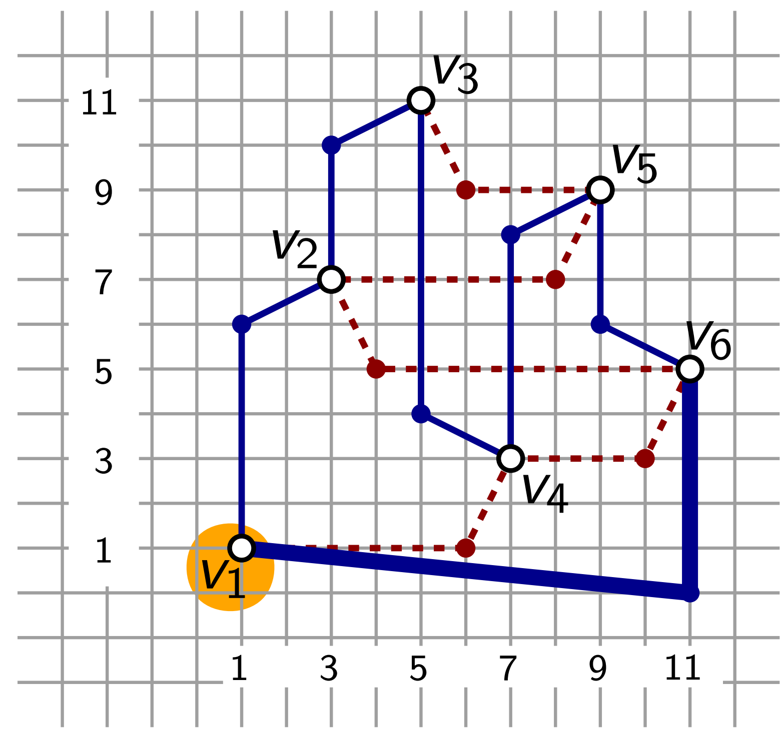
Cycle × Cycle



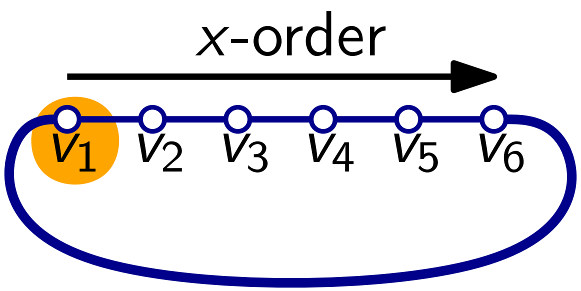


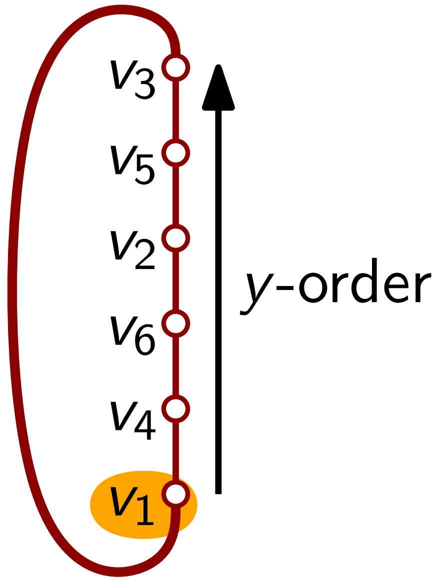
Edges:



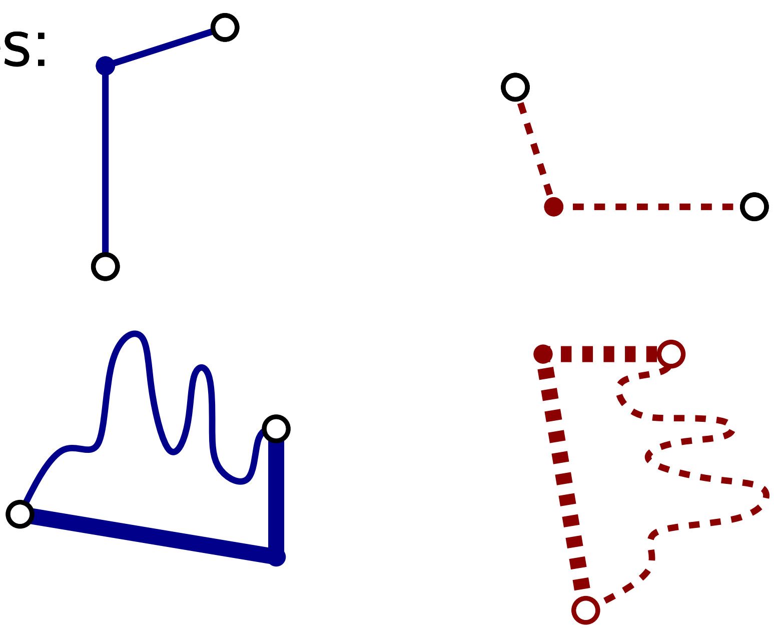


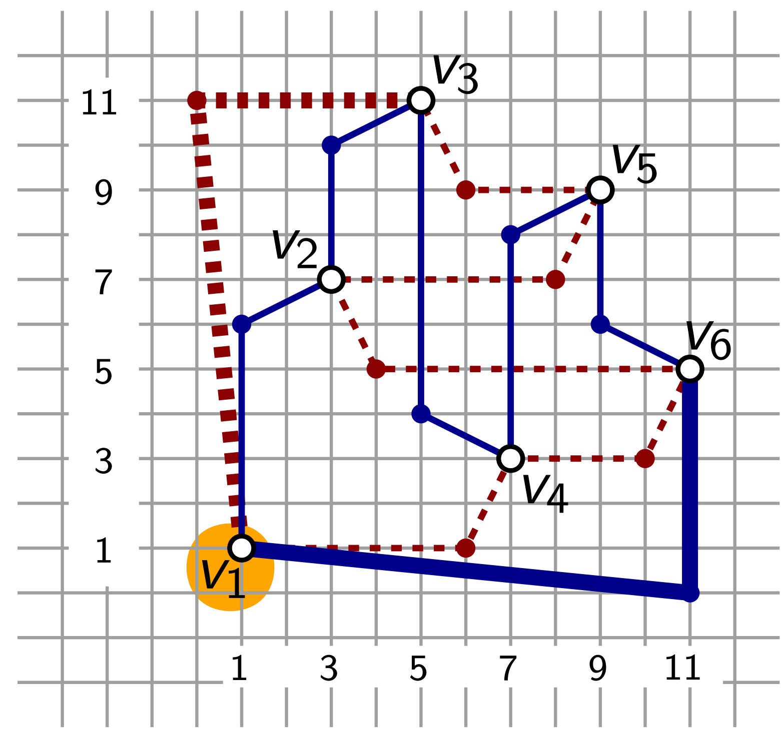
Cycle × Cycle



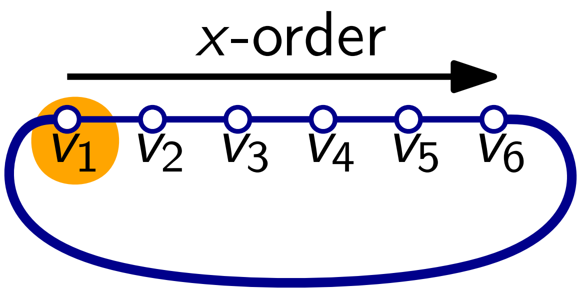


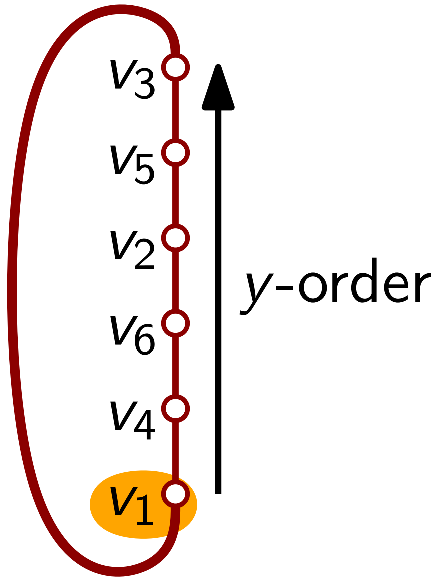
Edges:





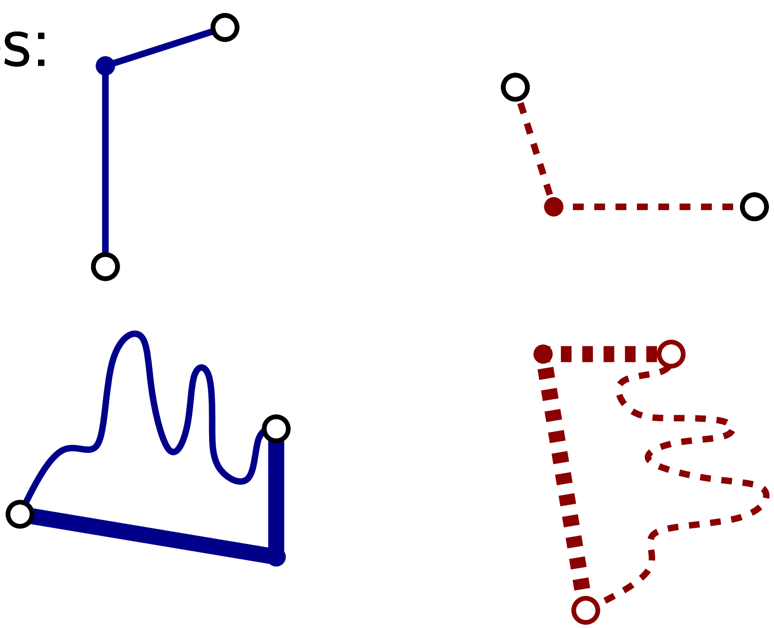
Cycle × Cycle

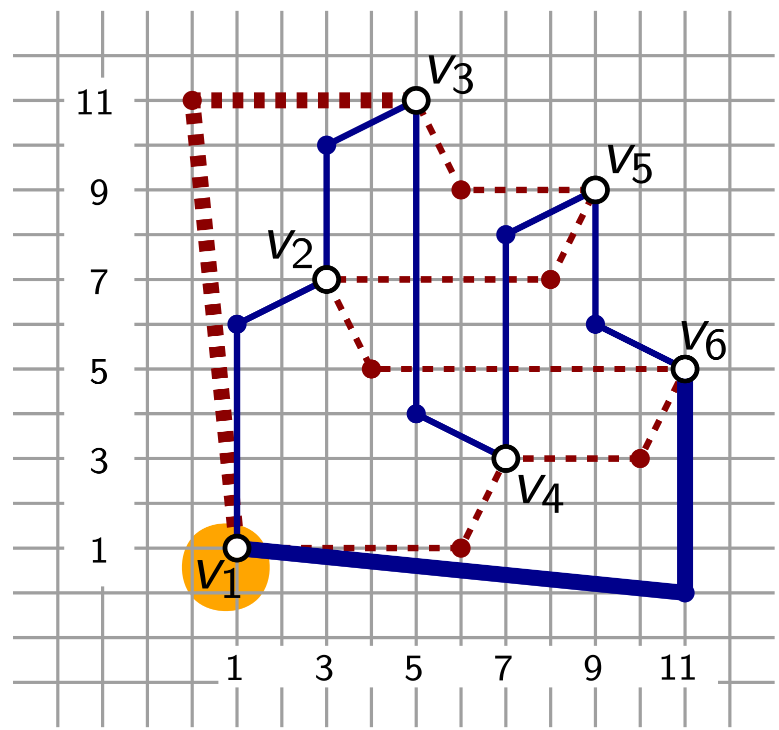




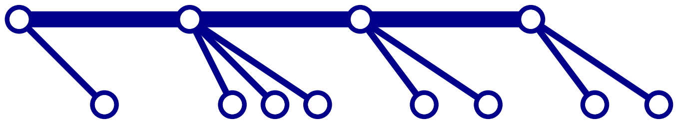
Bends: 1×1
Grid size: $(2n-1)^2$

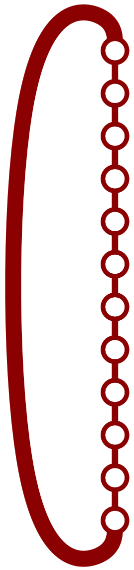
Edges:



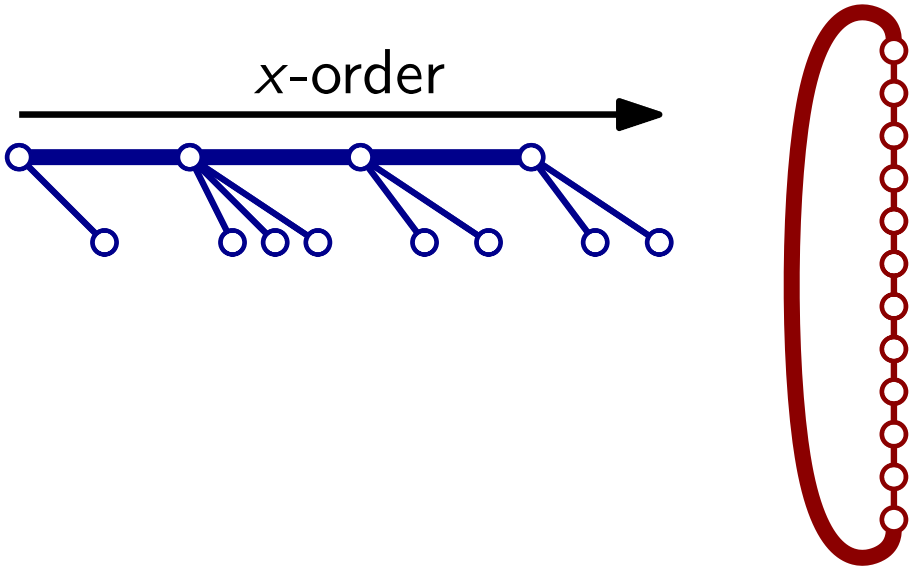


Caterpillar × Cycle

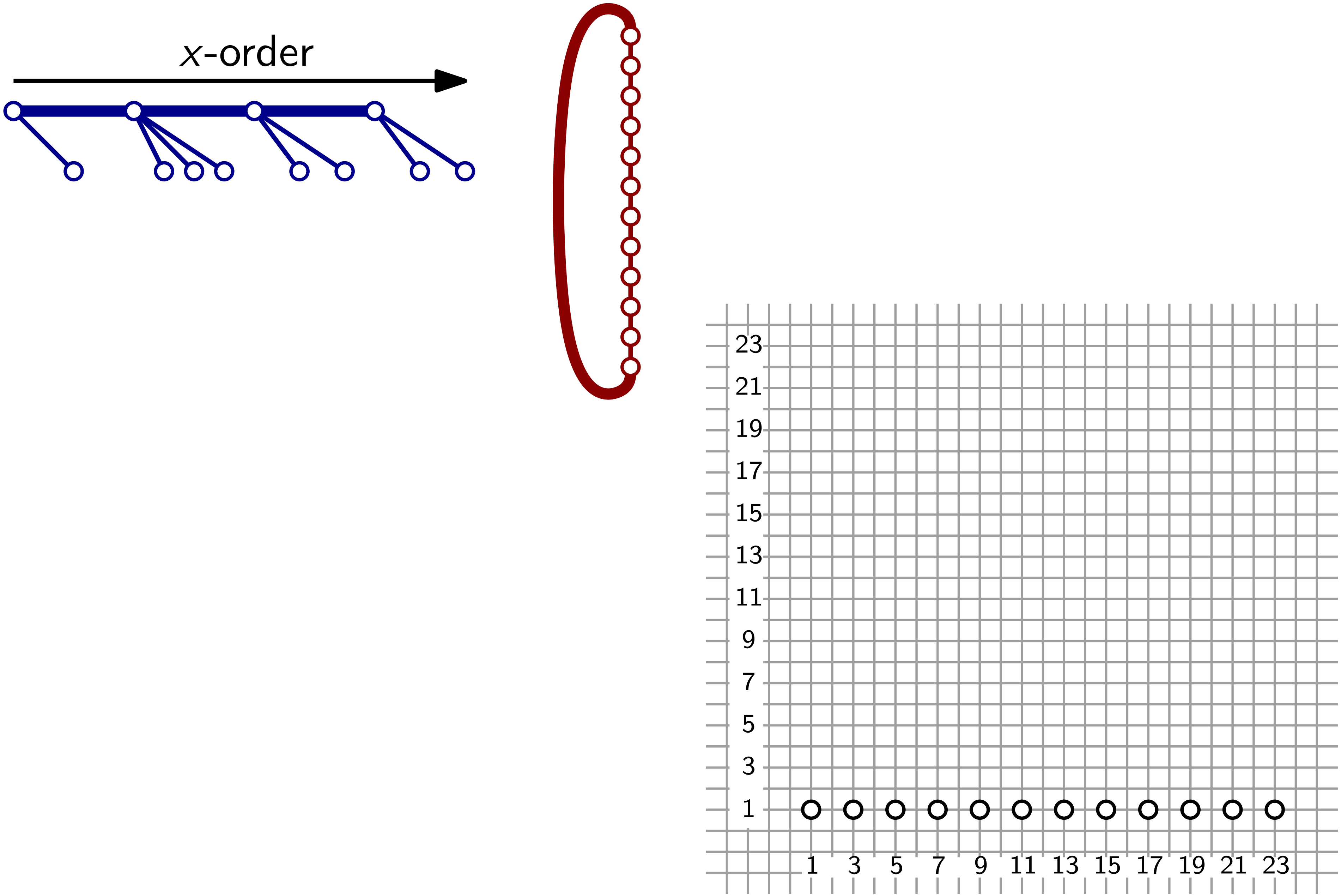




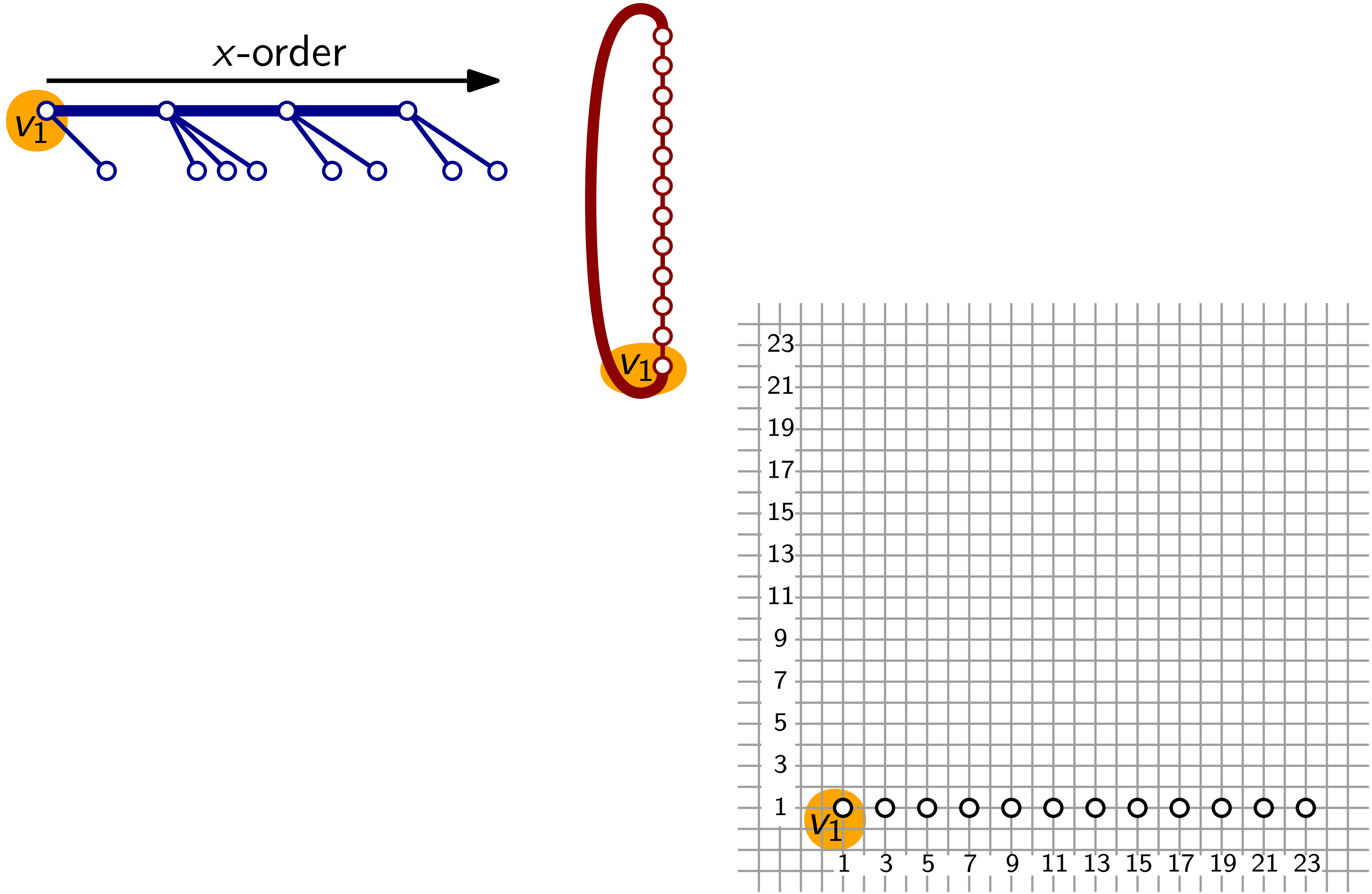
Caterpillar × Cycle



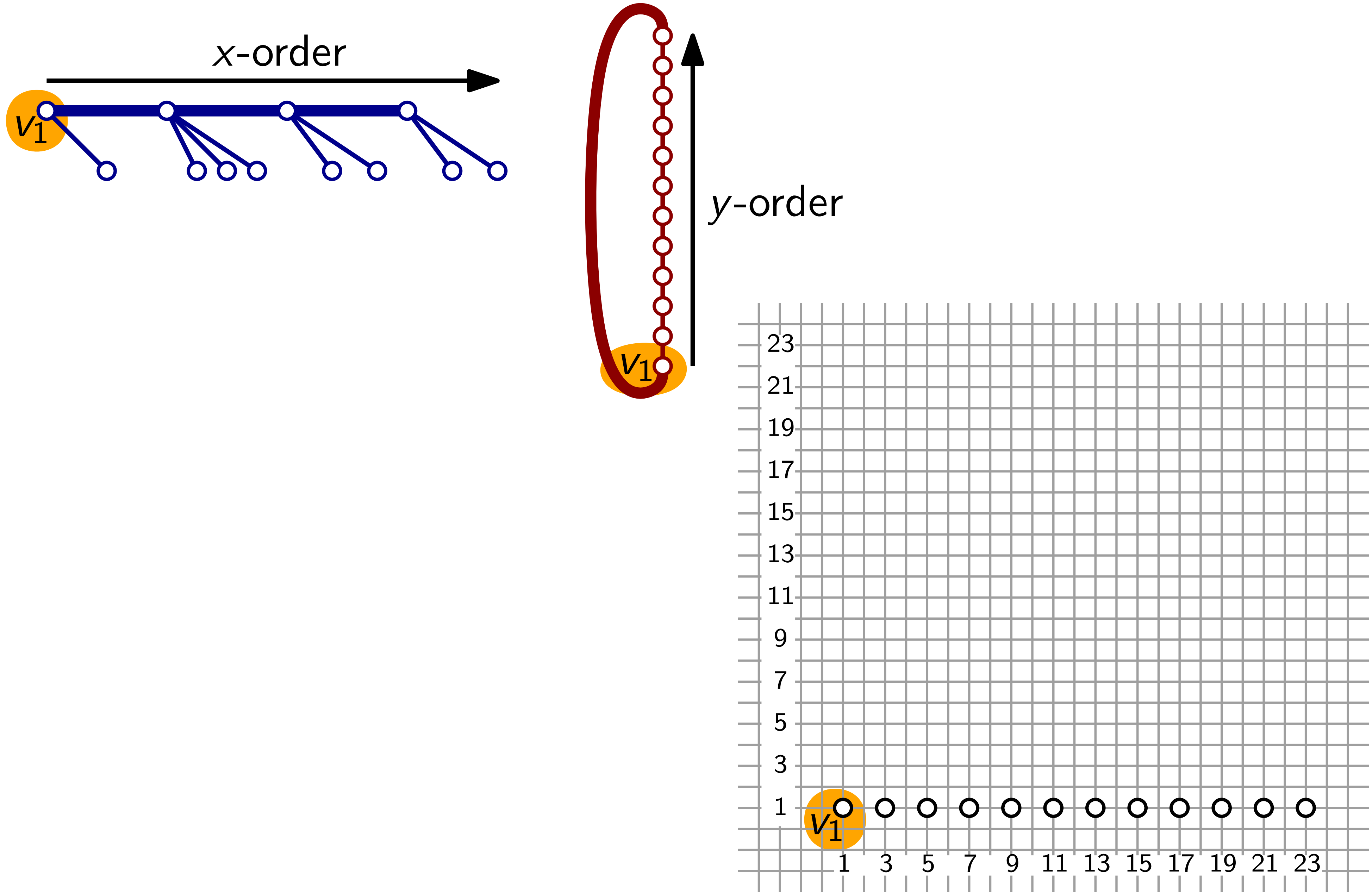
Caterpillar × Cycle



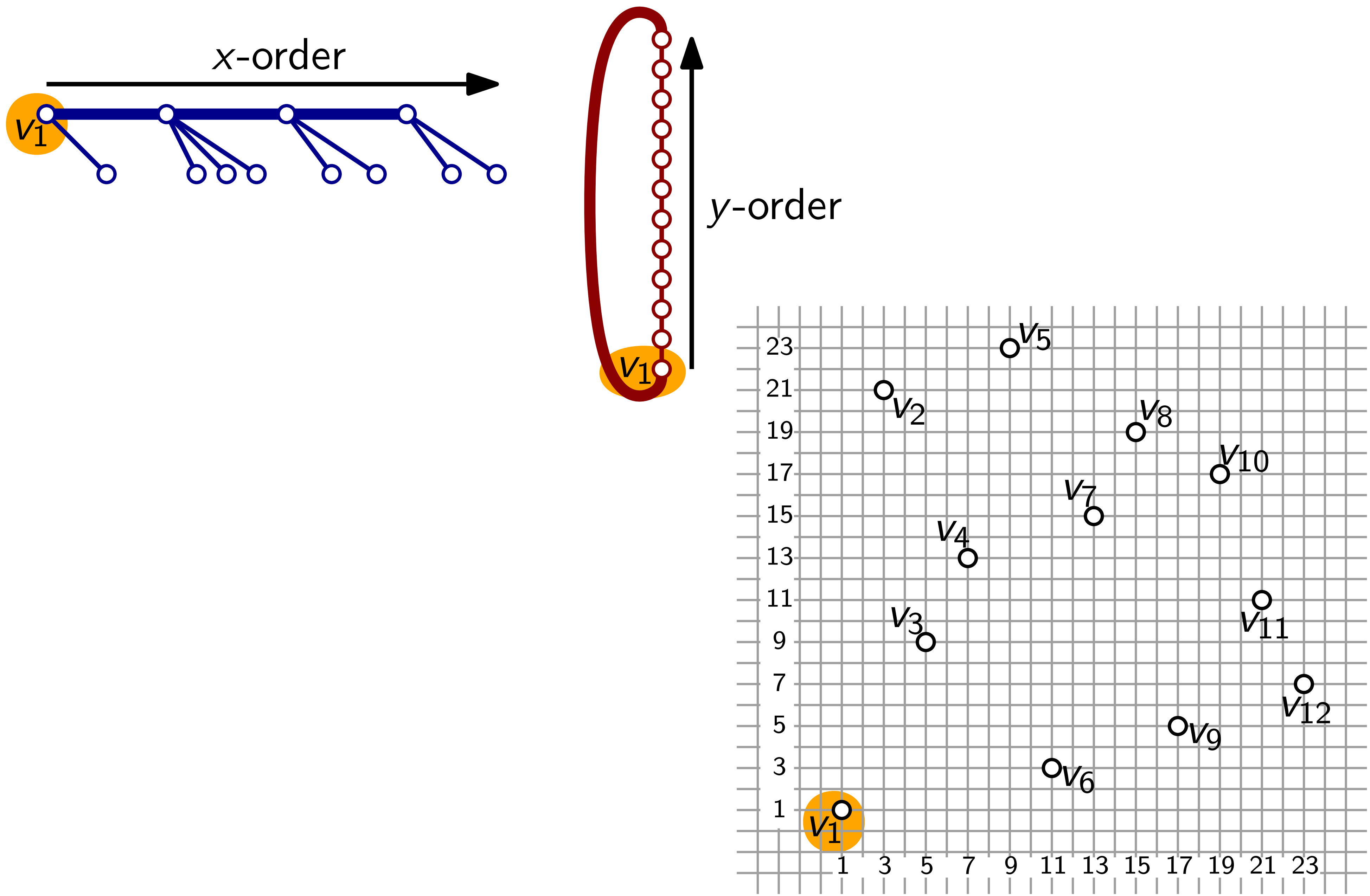
Caterpillar × Cycle



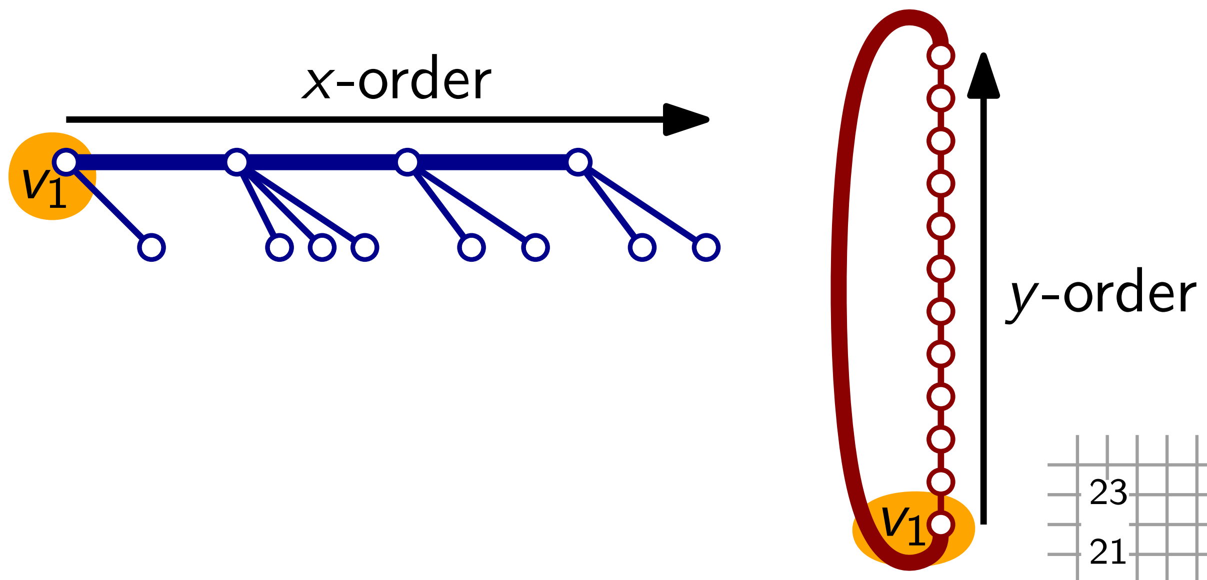
Caterpillar × Cycle



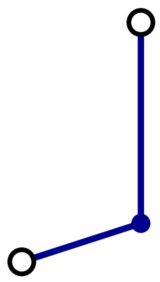
Caterpillar × Cycle

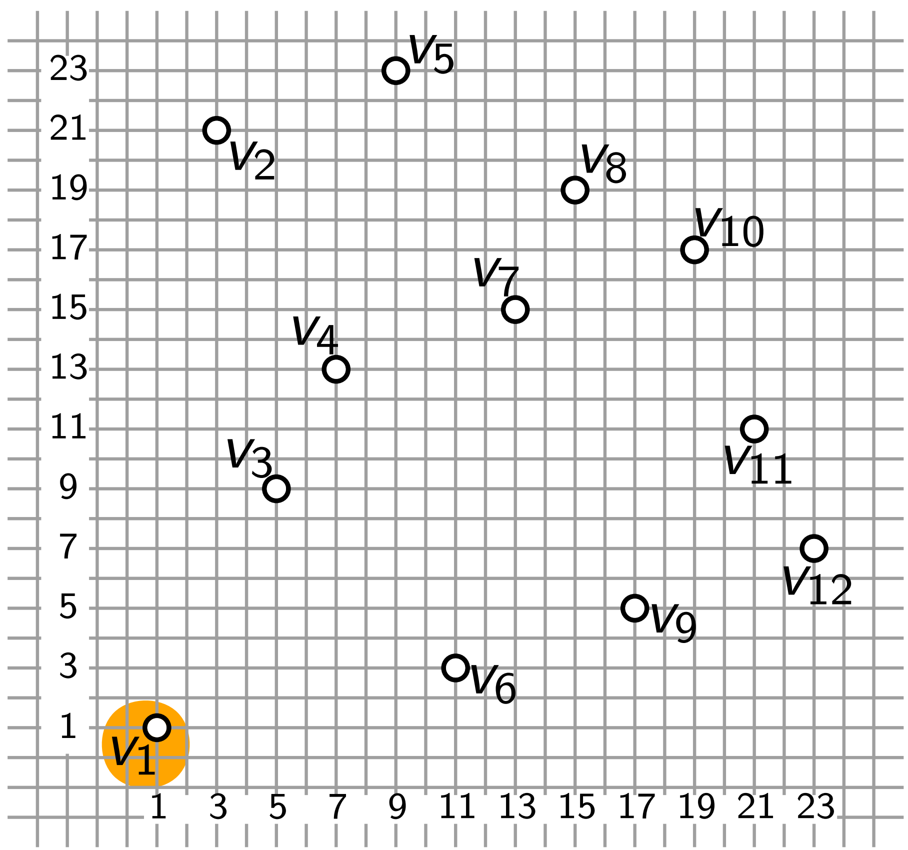


Caterpillar × Cycle

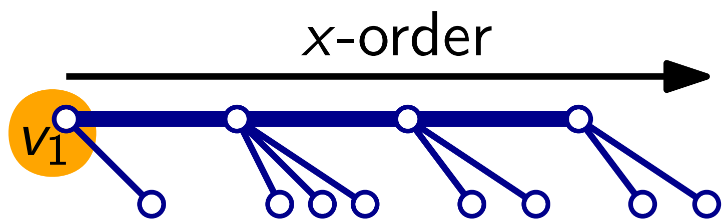


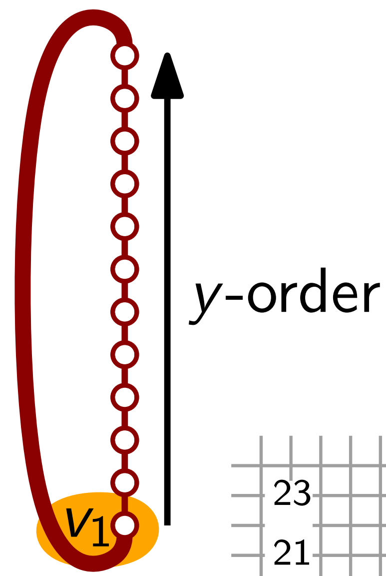
Edges:



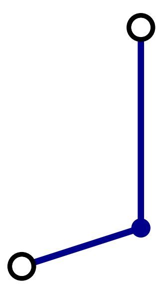


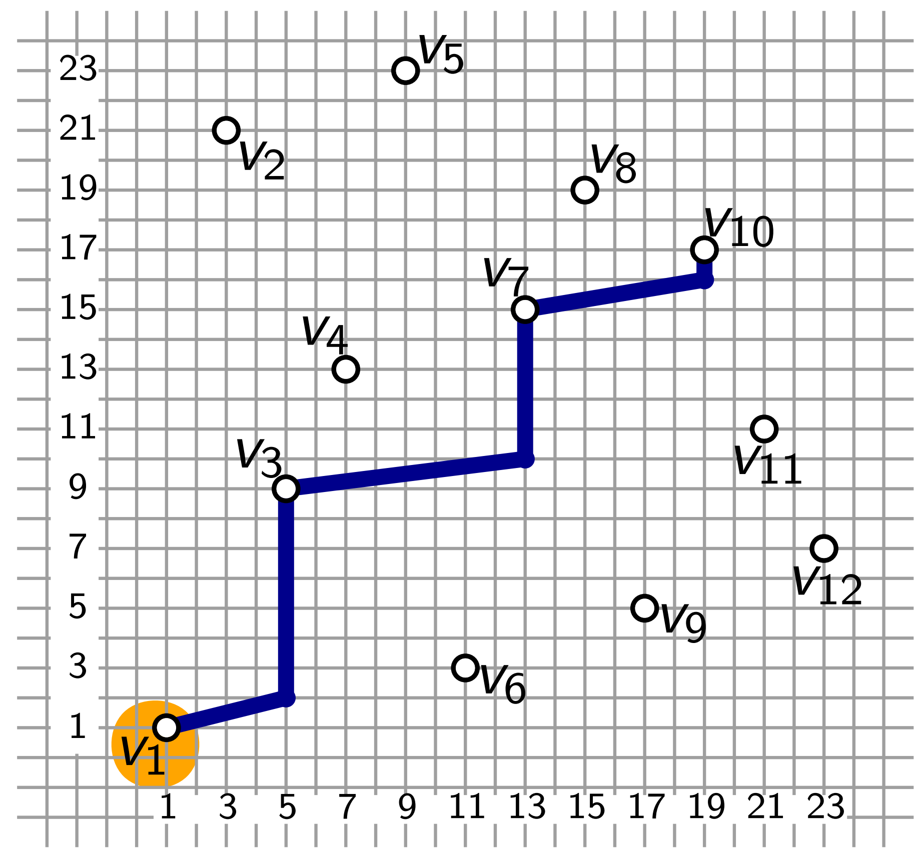
Caterpillar × Cycle



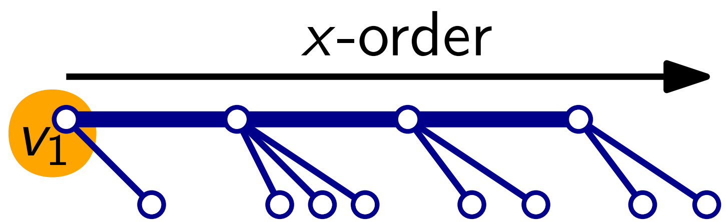


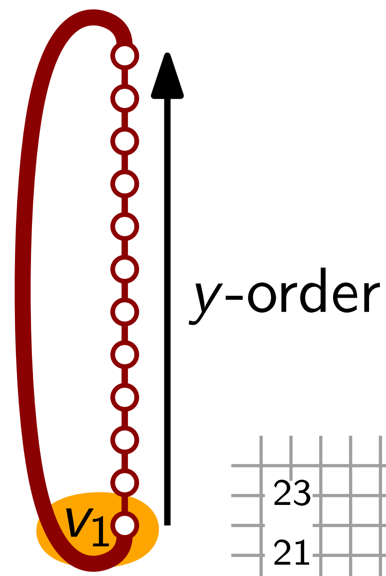
Edges:



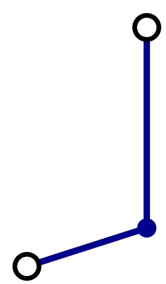


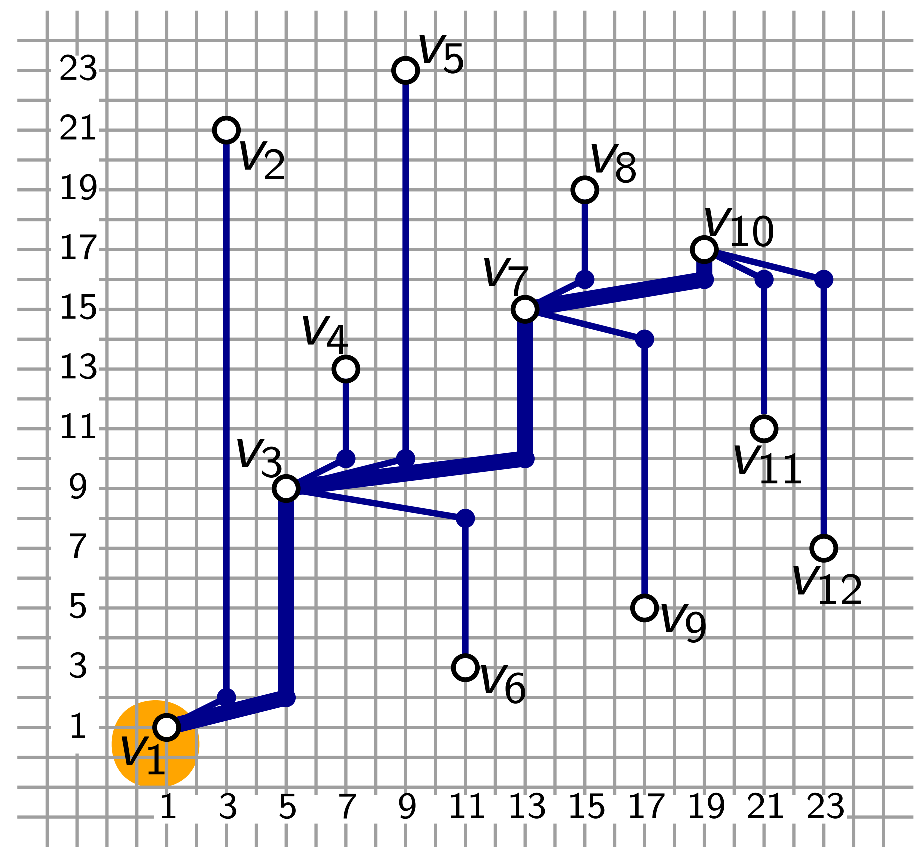
Caterpillar × Cycle



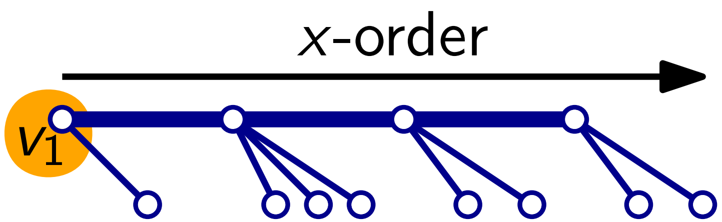


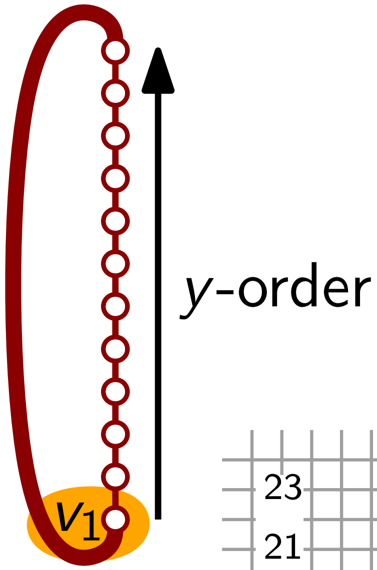
Edges:



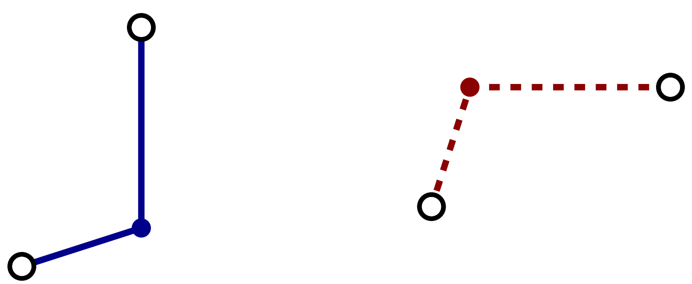


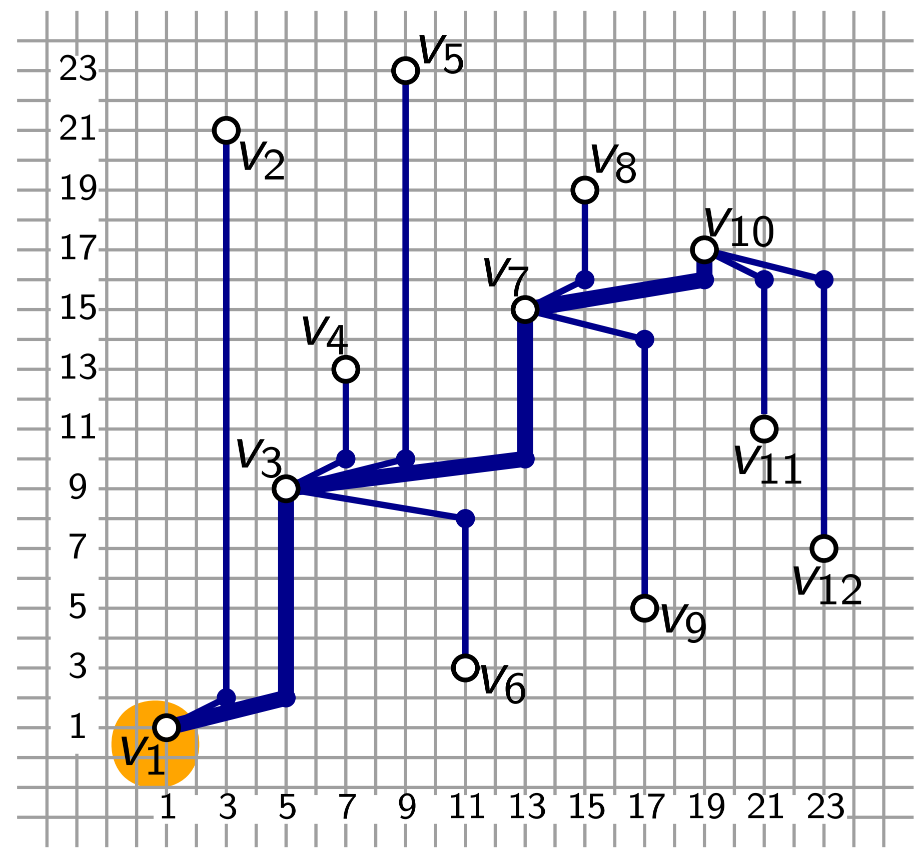
Caterpillar × Cycle



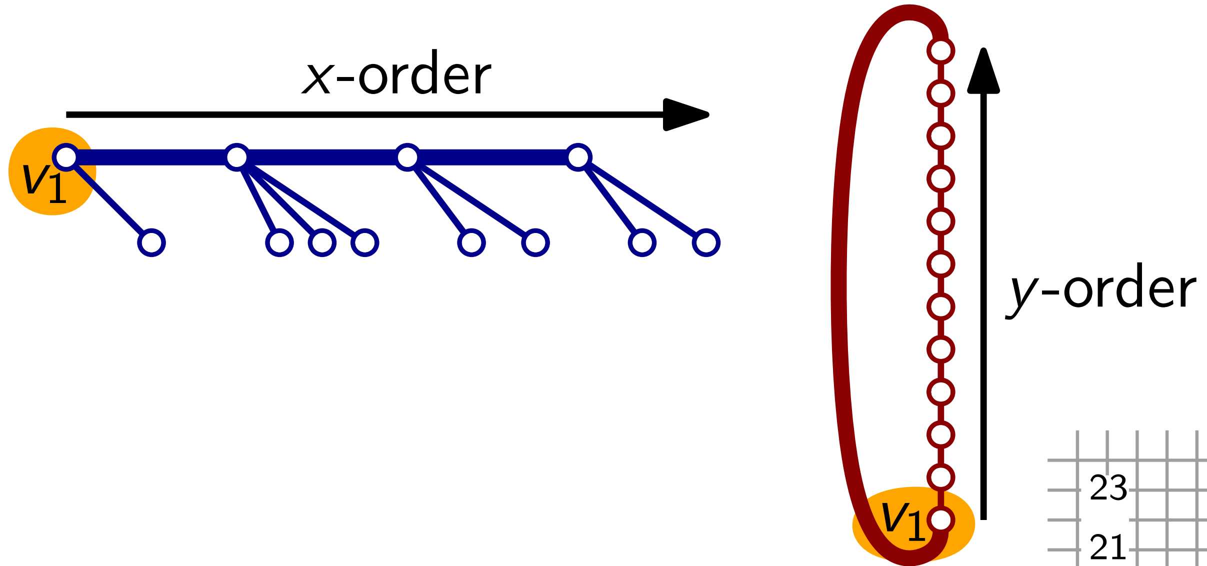


Edges:

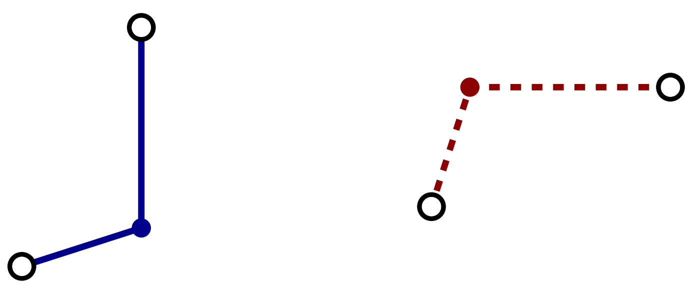


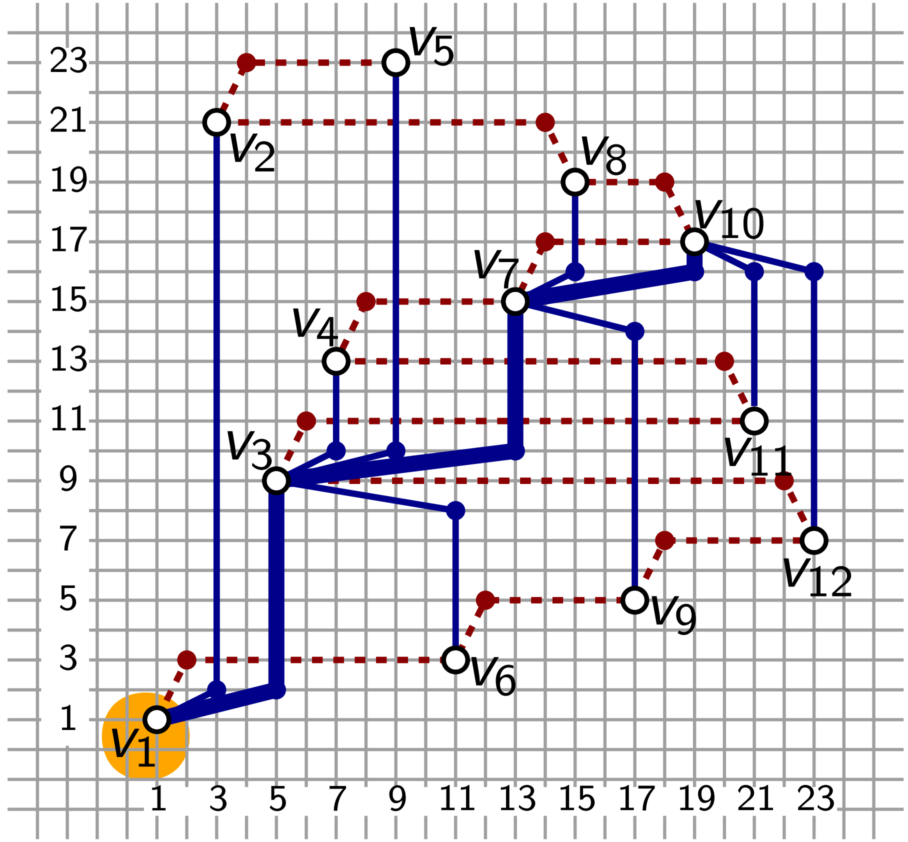


Caterpillar × Cycle

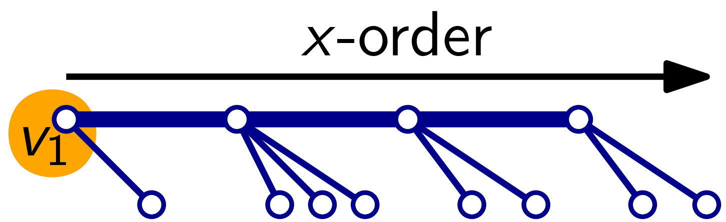


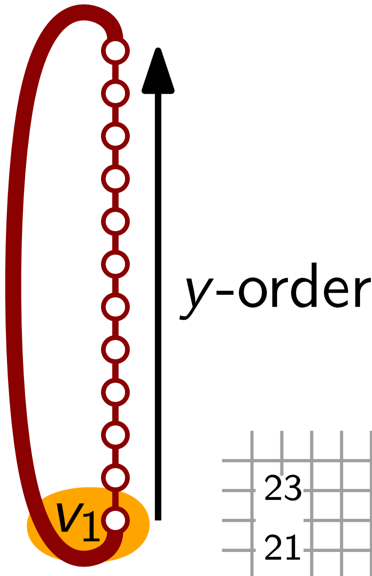
Edges:



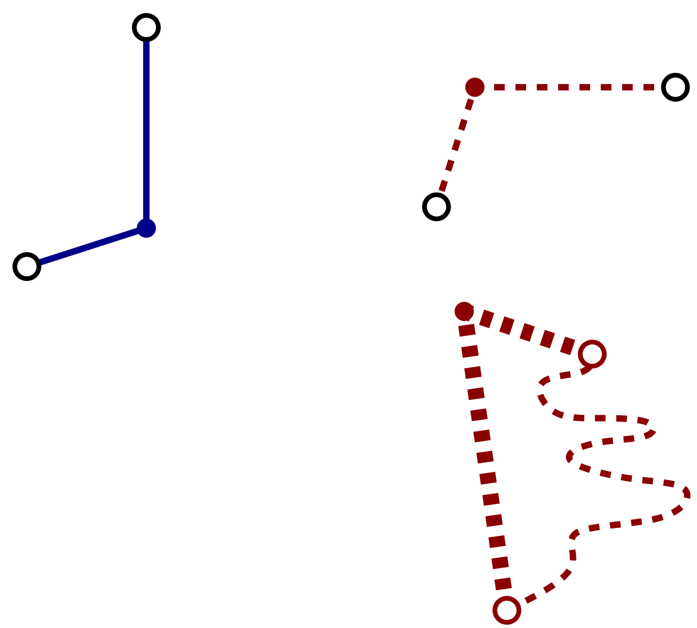


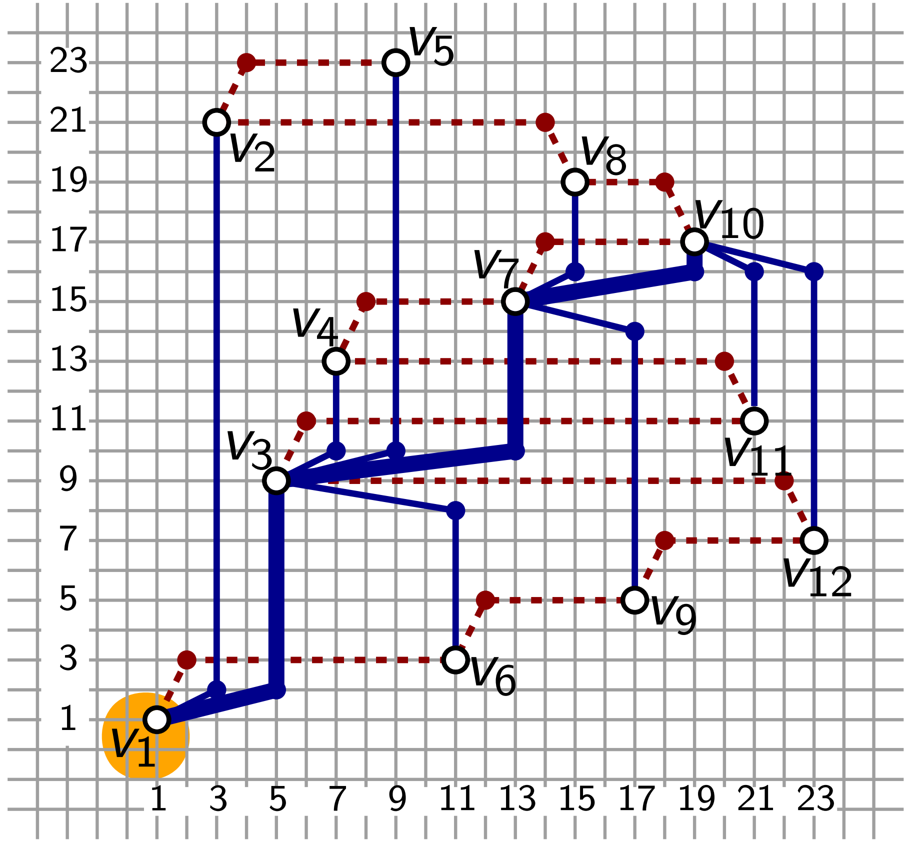
Caterpillar × Cycle



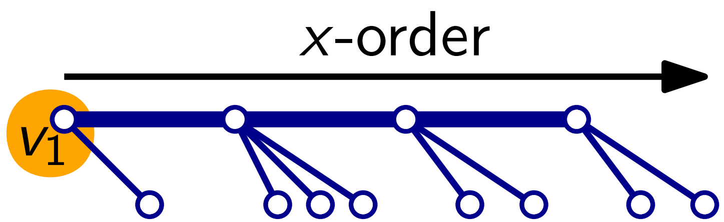


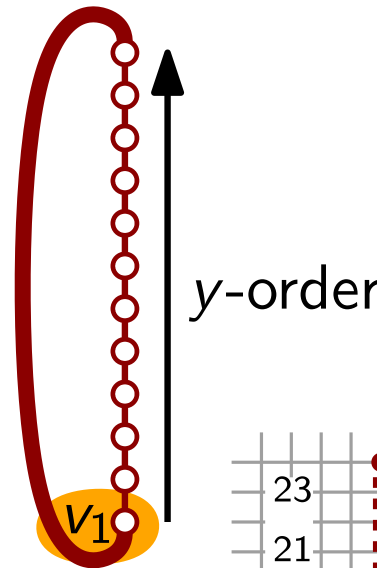
Edges:



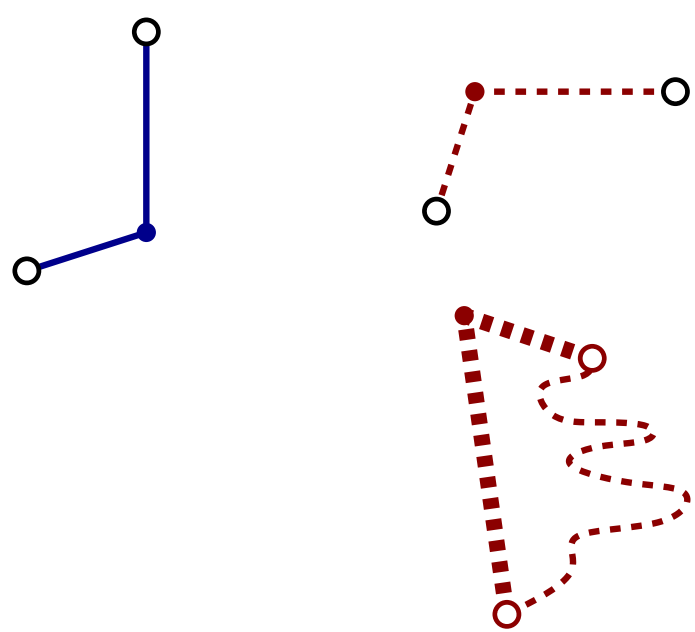


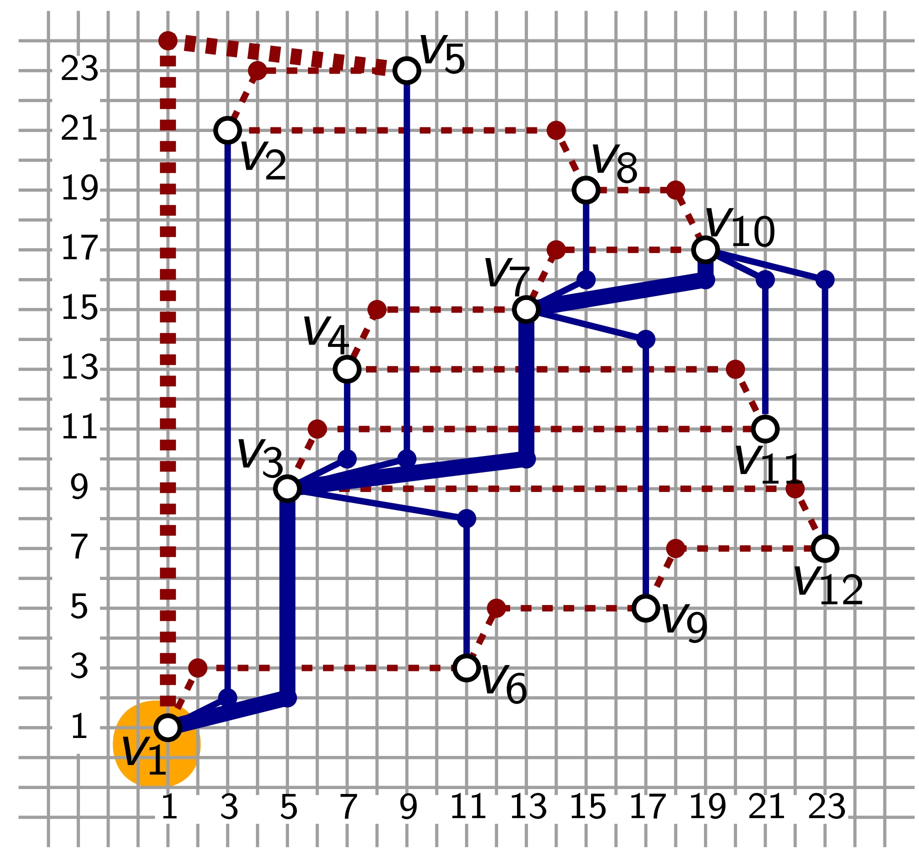
Caterpillar × Cycle



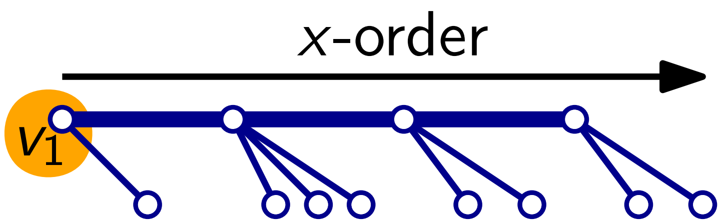


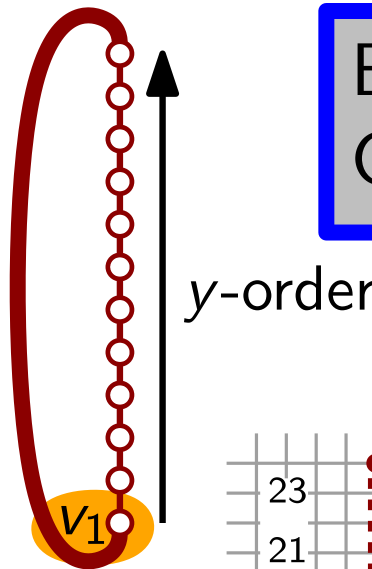
Edges:



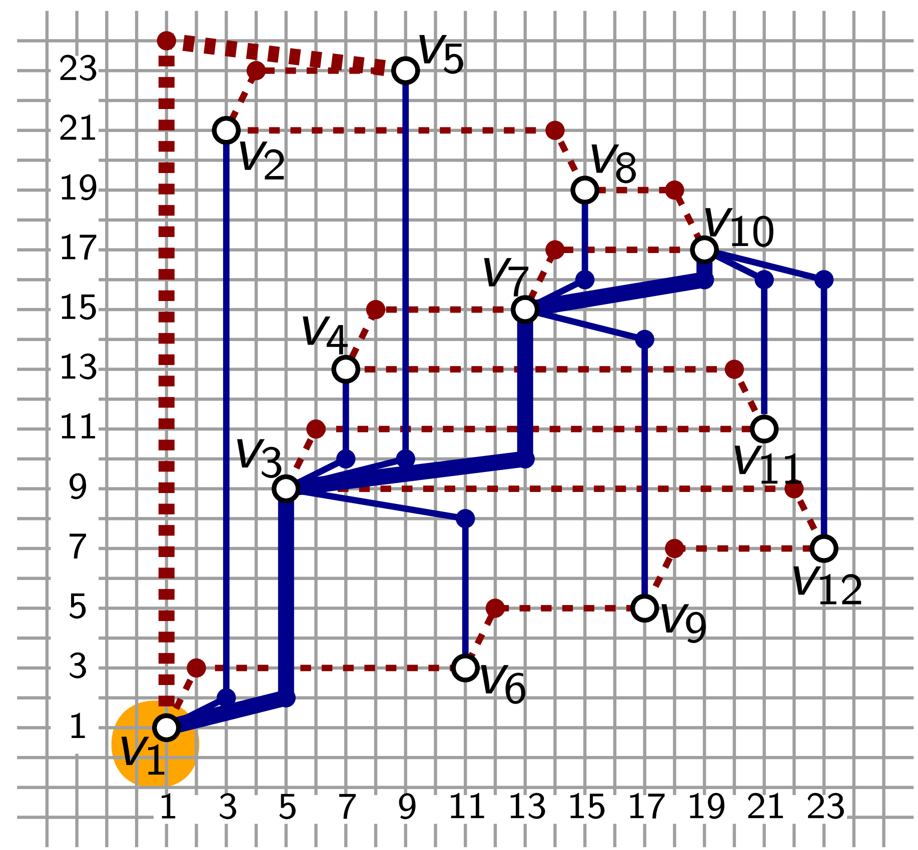


Caterpillar × Cycle

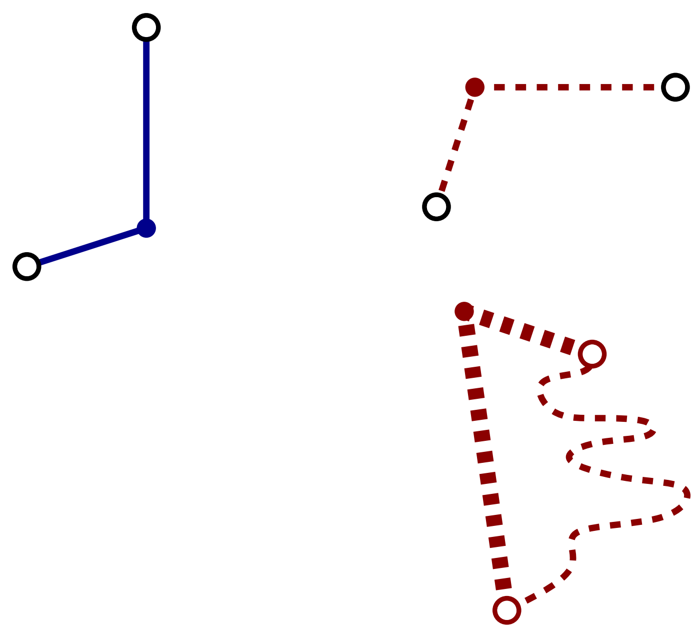




Bends: 1×1
Grid size: $(2n-1) \times 2n$



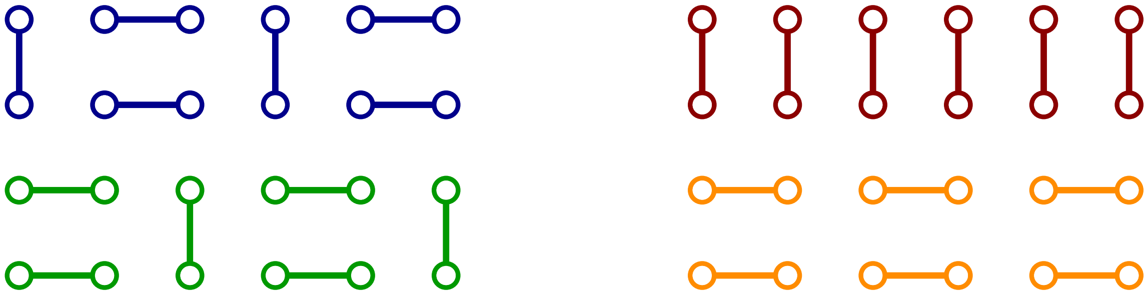
Edges:



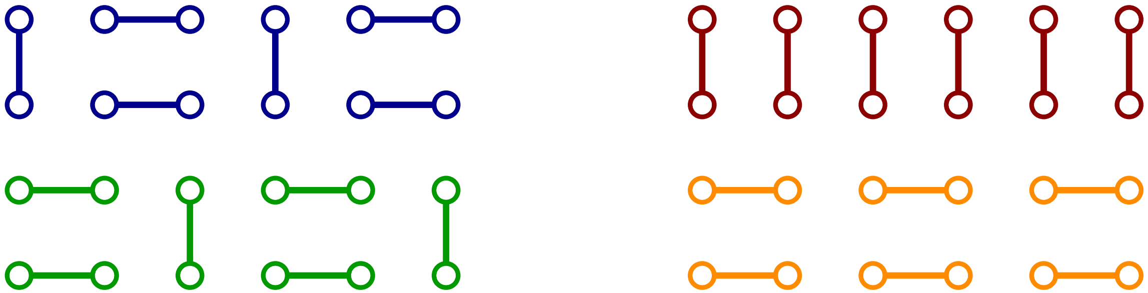
Overview

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



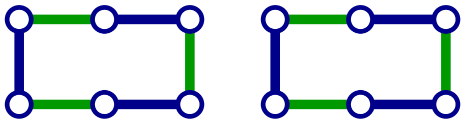
Combine $\Rightarrow$ two sets of cycles $\mathcal{C}_1$, $\mathcal{C}_2$

Four Matchings

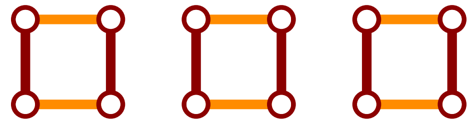
$\mathcal{C}_1$ $\mathcal{C}_2$

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

Four Matchings

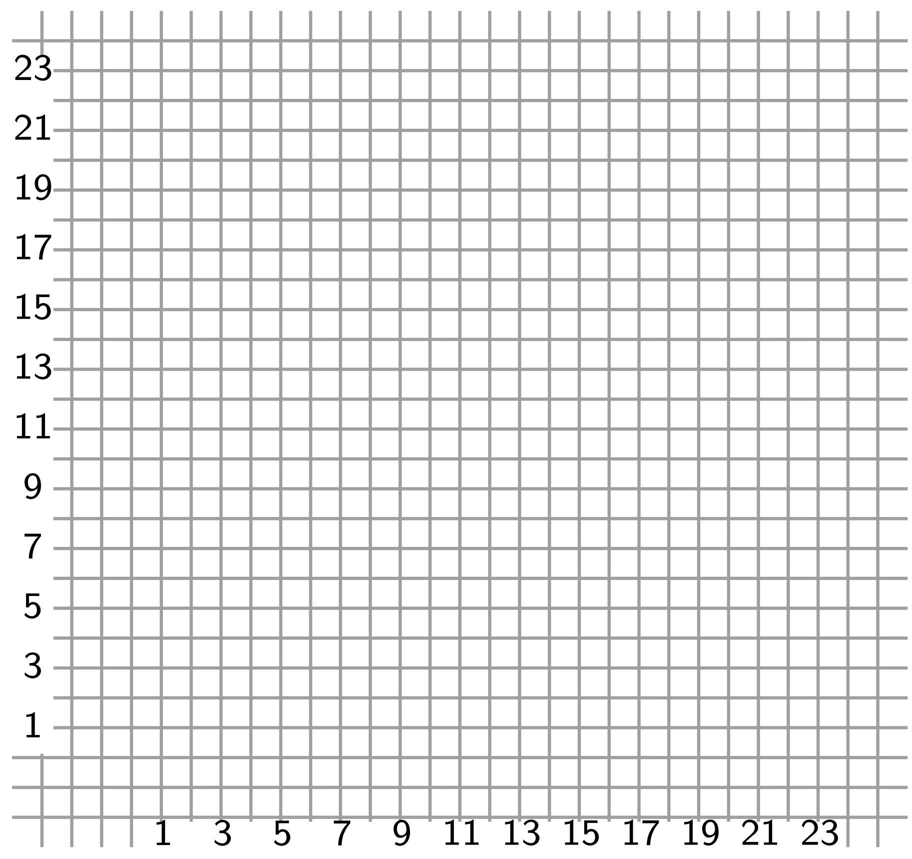


$\mathcal{C}_1$



$\mathcal{C}_2$

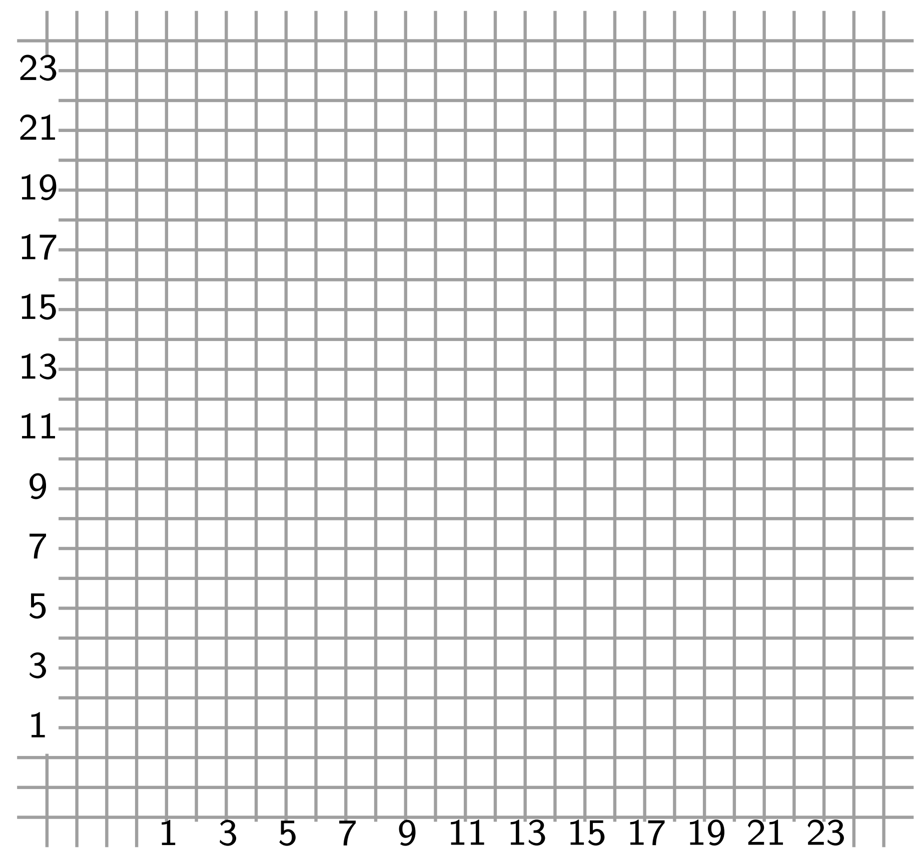
Placement algorithm:



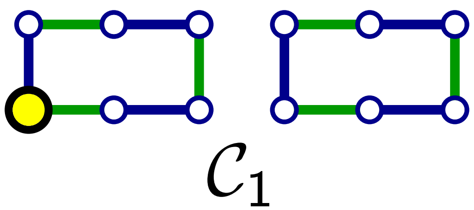
Four Matchings

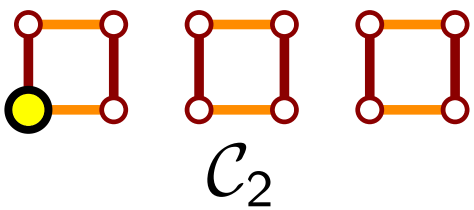
Placement algorithm:

- Pick v_1



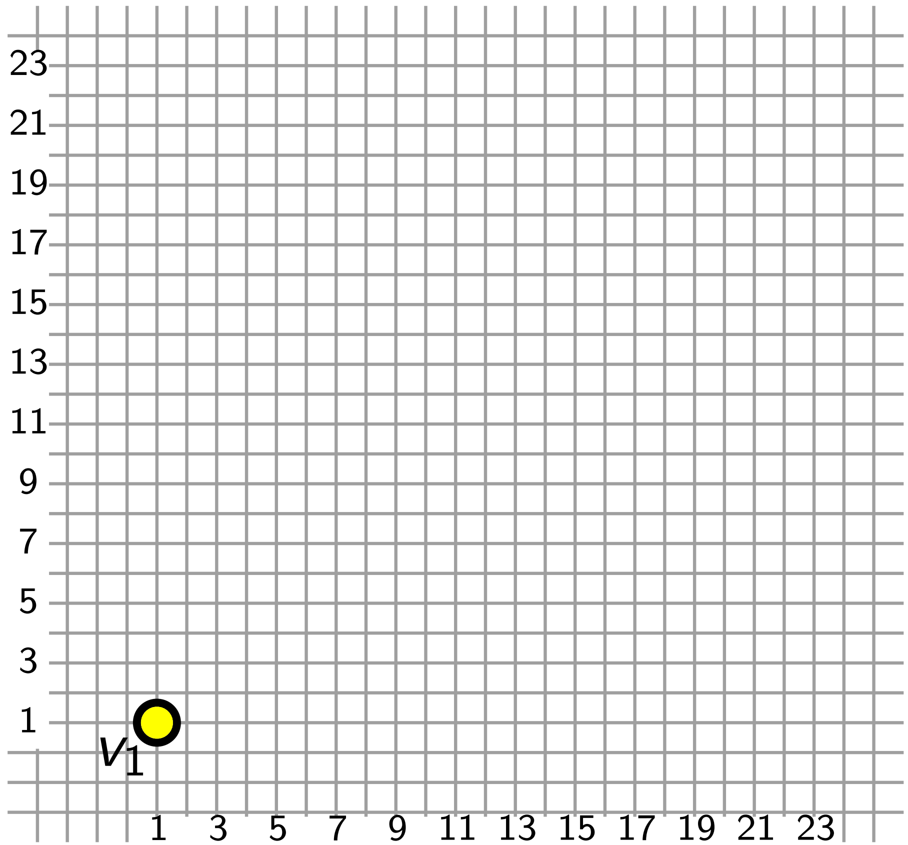
Four Matchings





Placement algorithm:

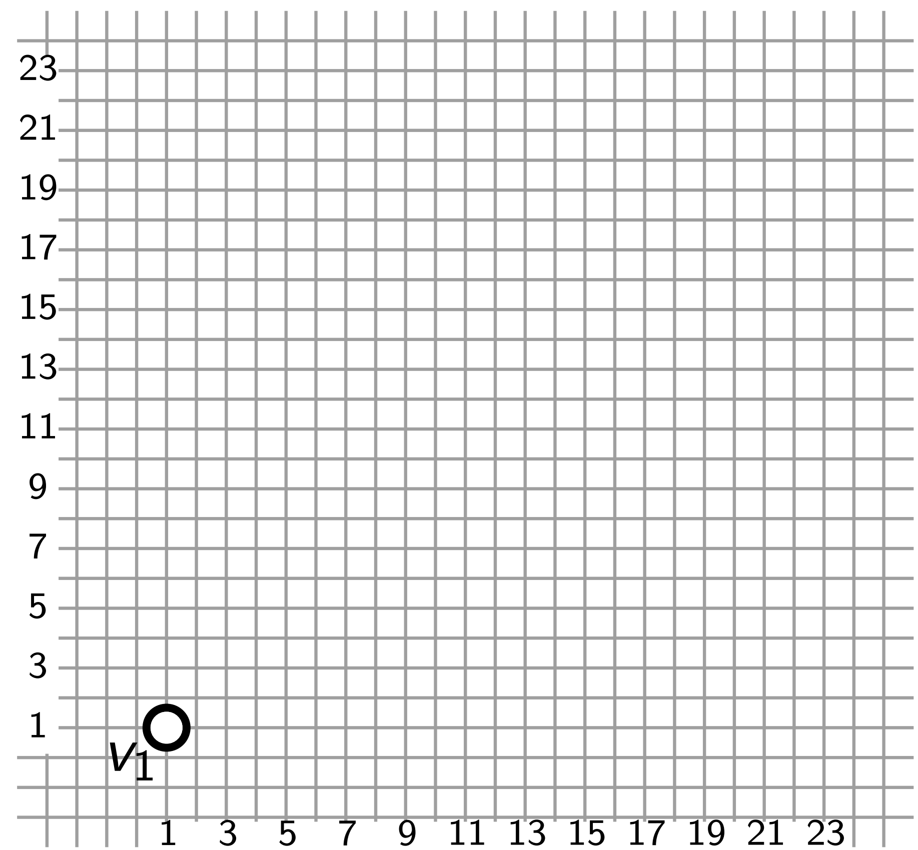
- Pick v_1, place it



Four Matchings

Placement algorithm:

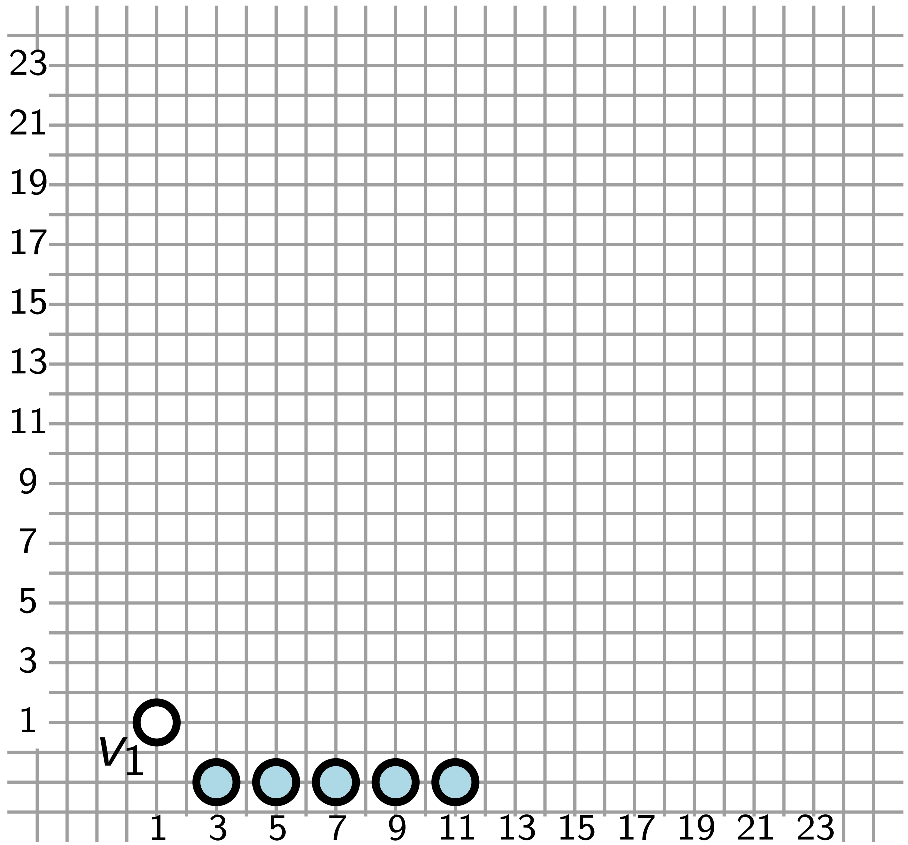
- Pick v_1, place it



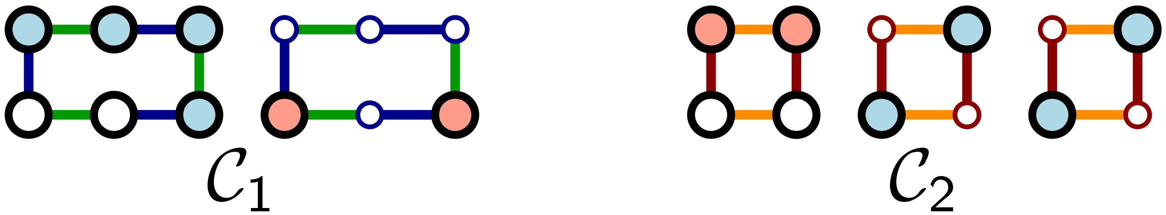
Four Matchings

Placement algorithm:

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

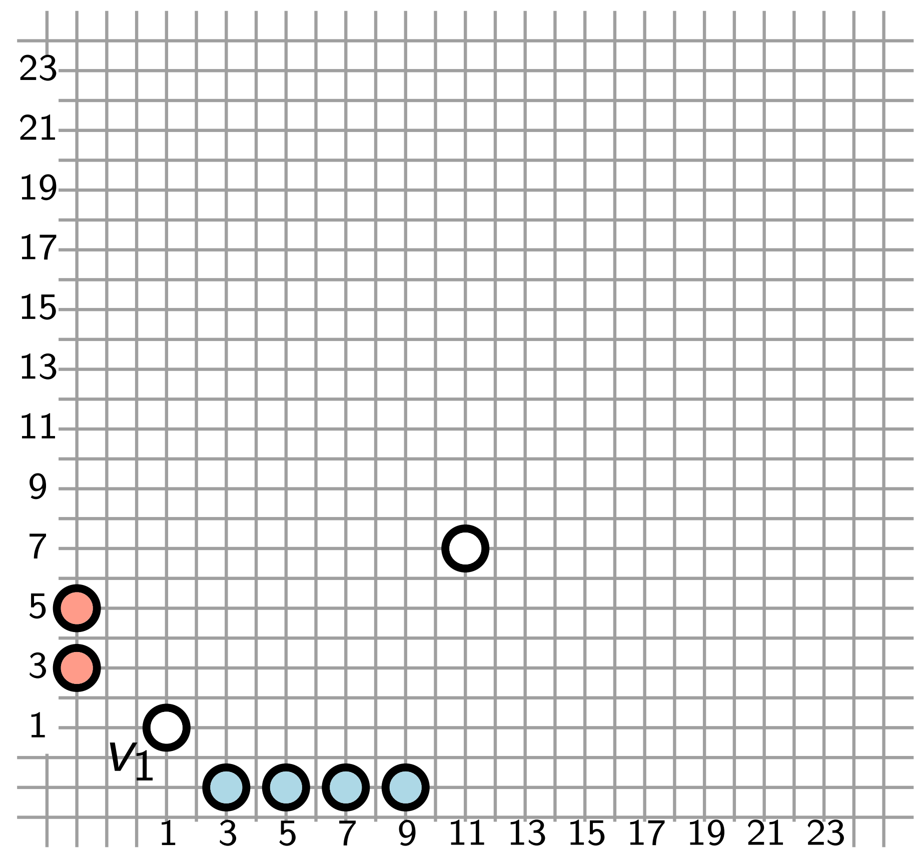


Four Matchings

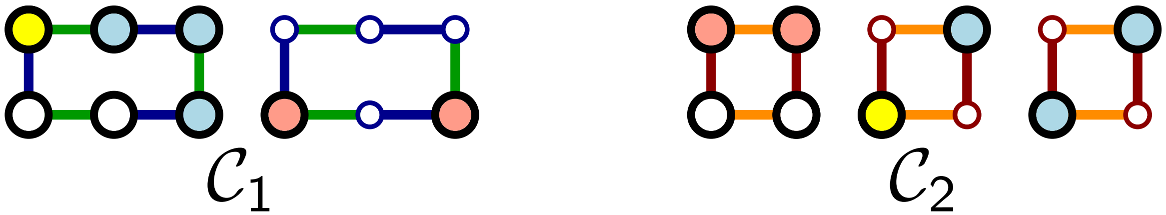


Placement algorithm:

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

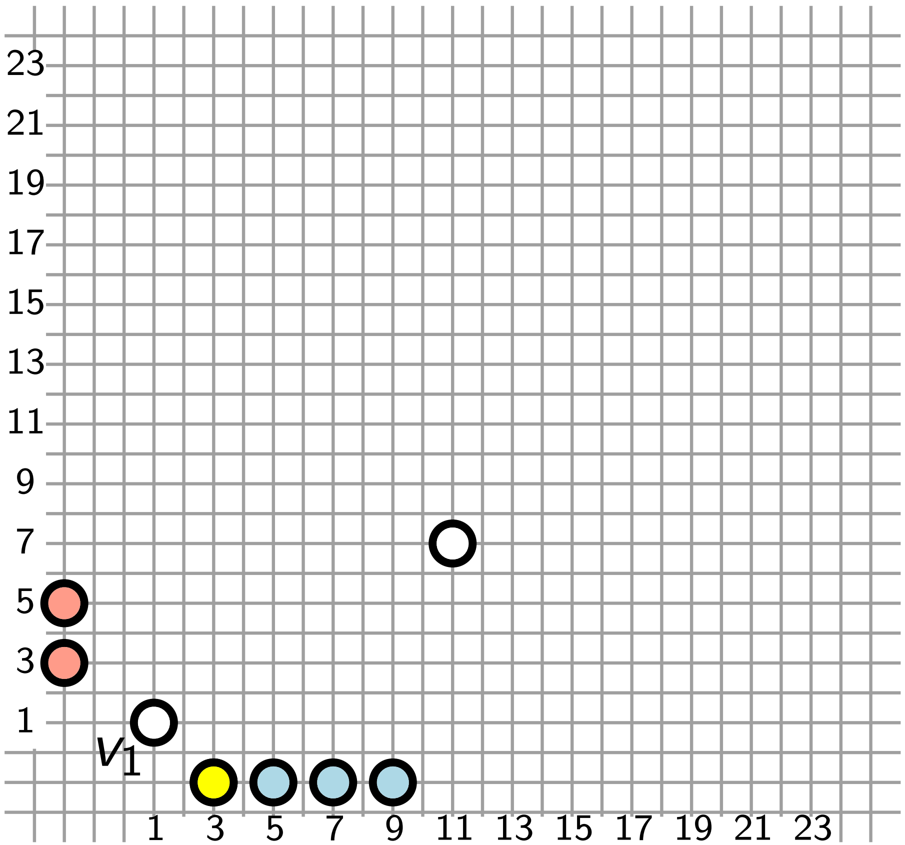


Four Matchings

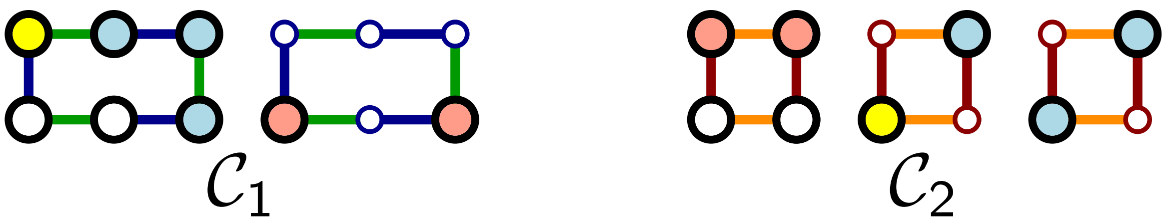


Placement algorithm:

- 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.

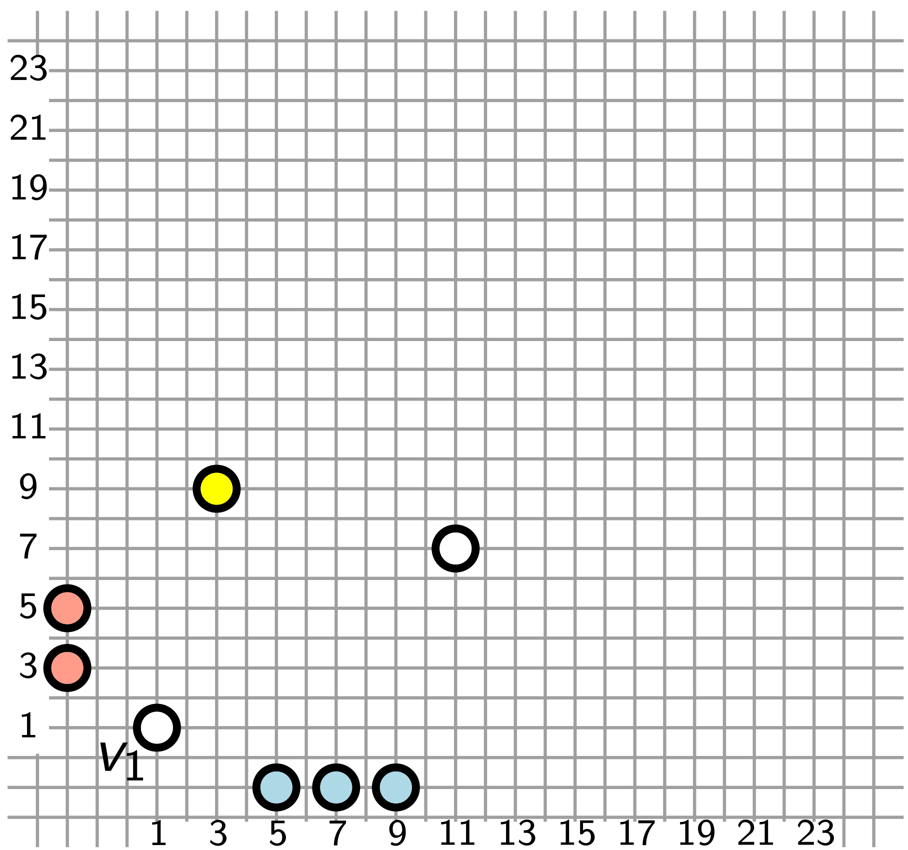


Four Matchings

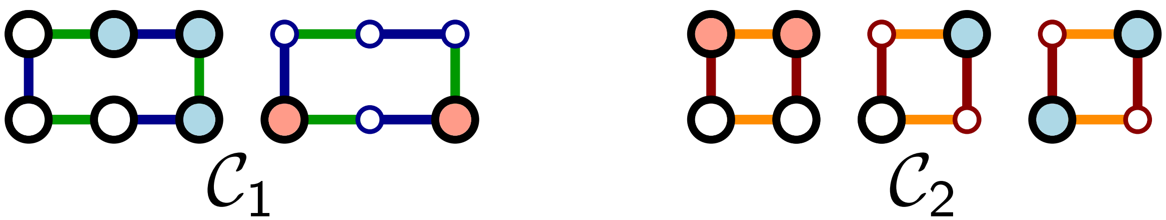


Placement algorithm:

- 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

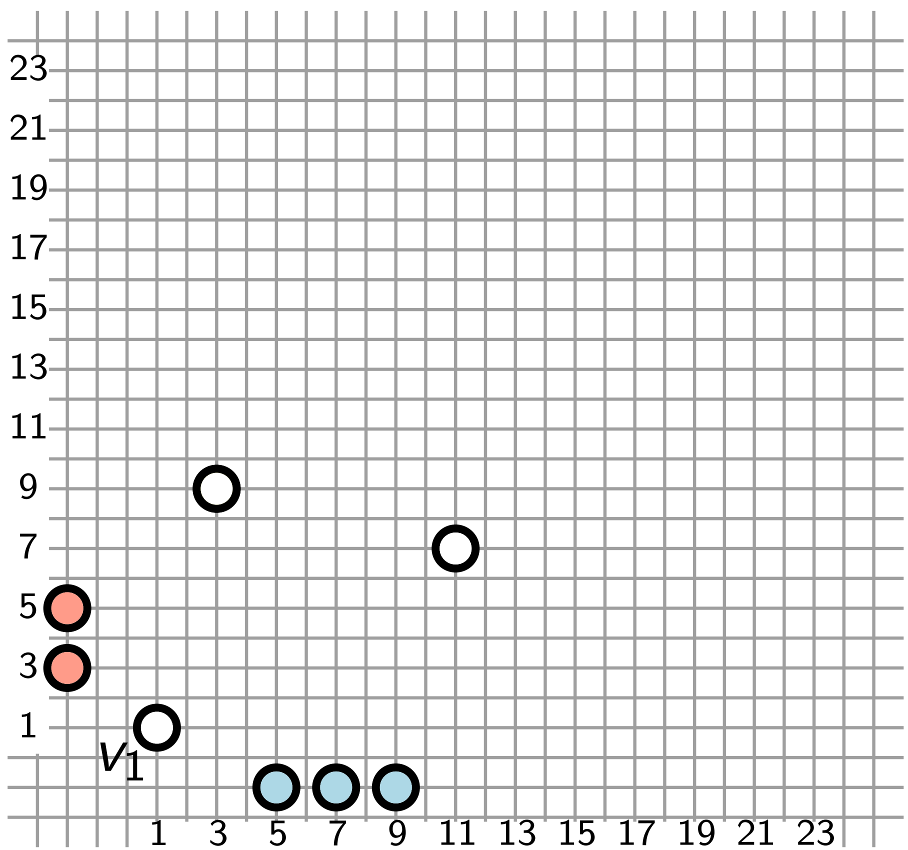


Four Matchings

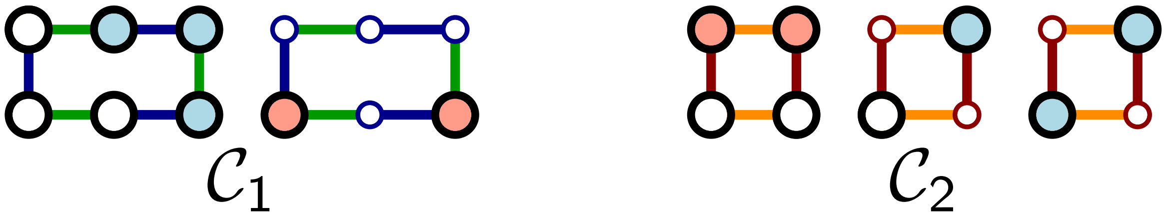


Placement algorithm:

- 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

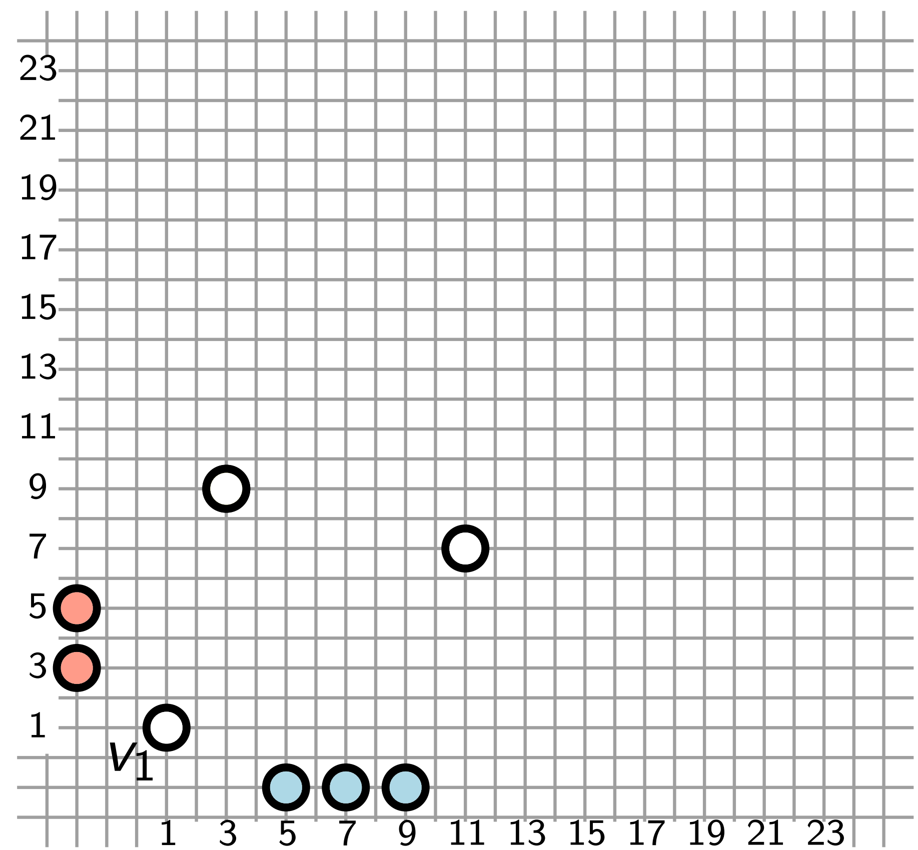


Four Matchings

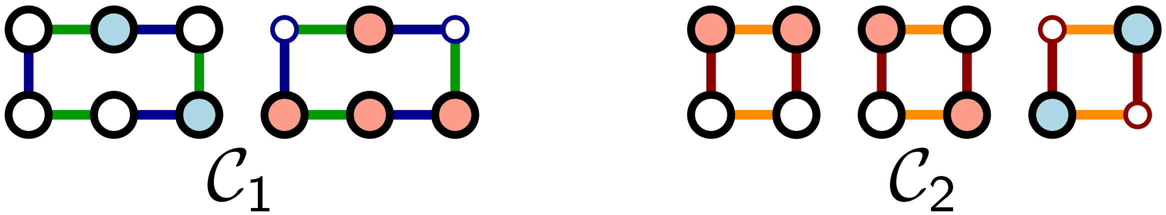


Placement algorithm:

- 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

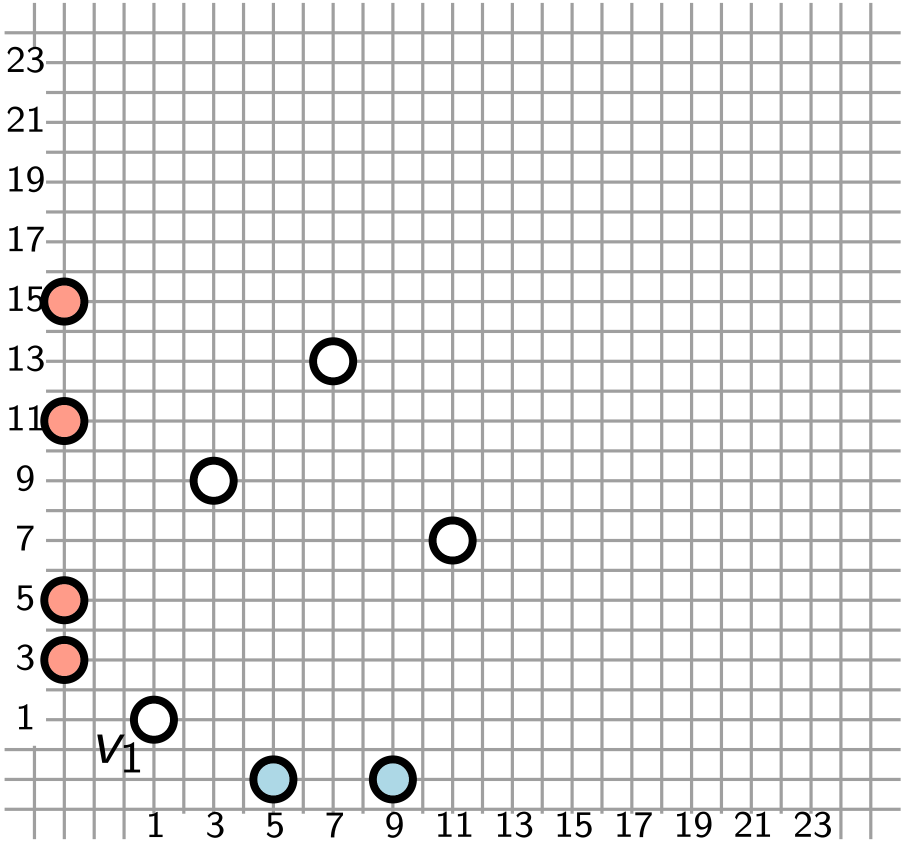


Four Matchings

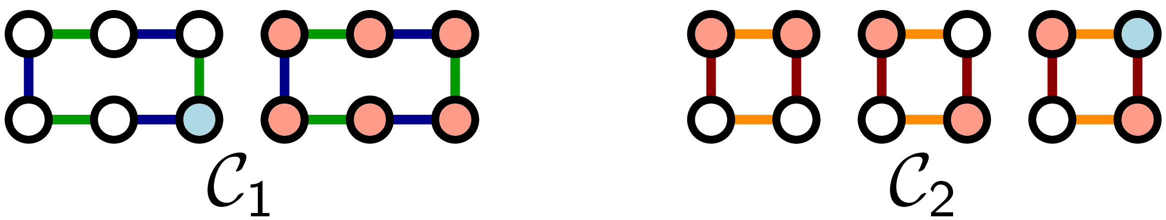


Placement algorithm:

- 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

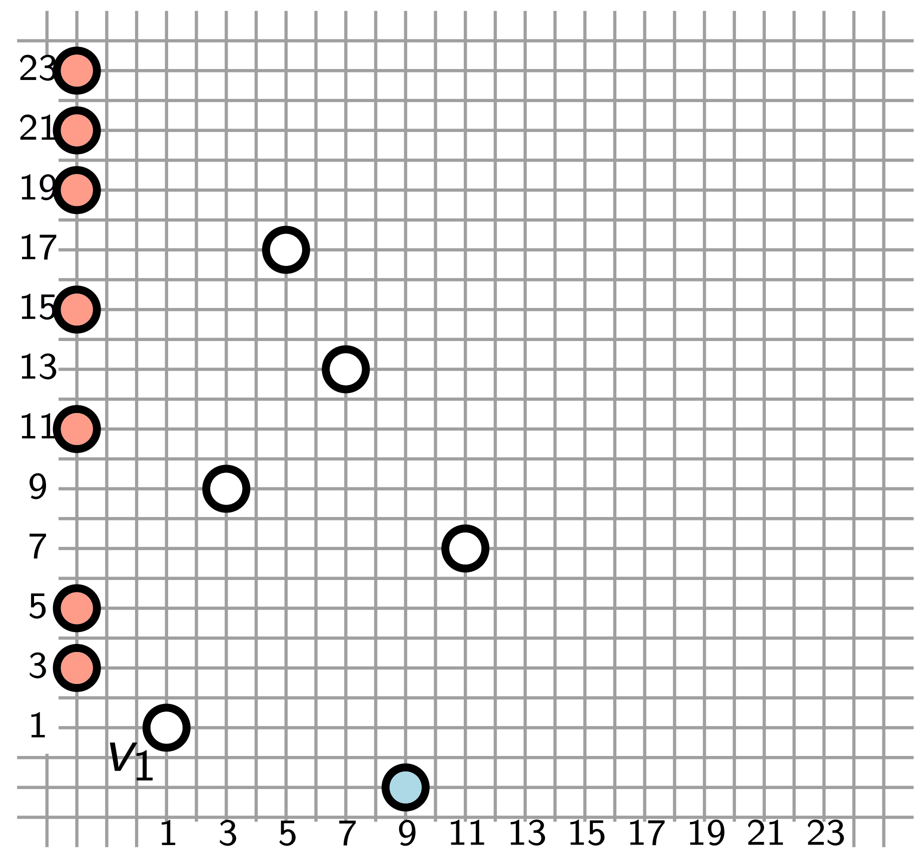


Four Matchings



Placement algorithm:

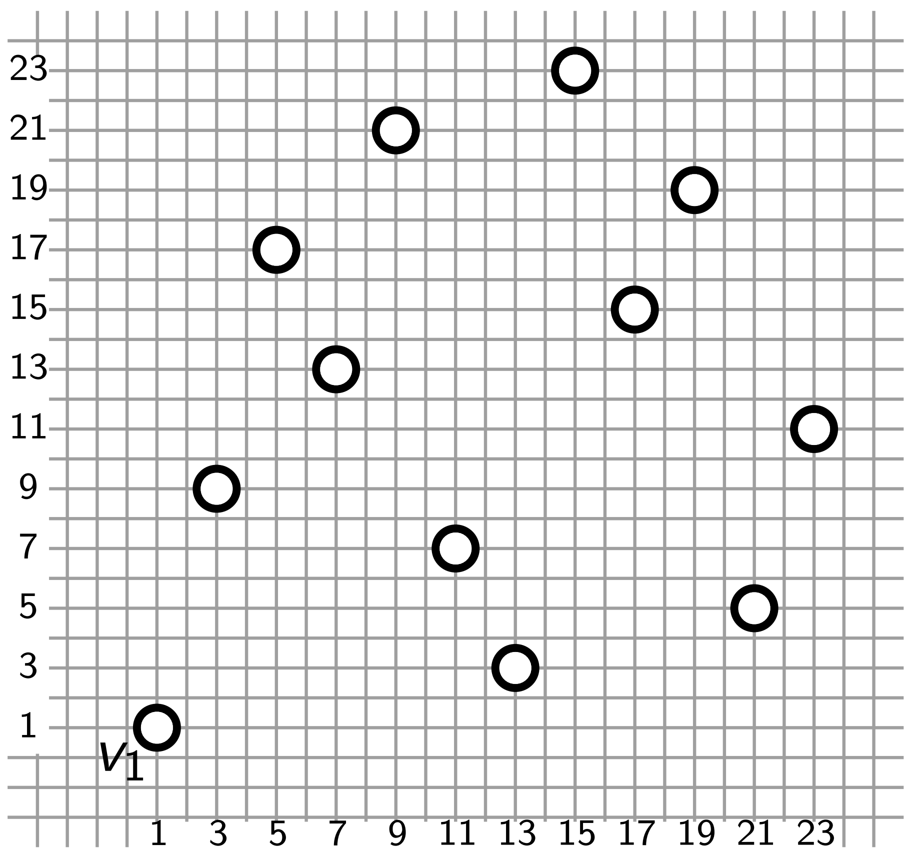
- Pick v_1, place it
- Assign x-coords. to cycle in $\mathcal{C}_1$
- Assign y-coords. to cycle in $\mathcal{C}_2$
- Pick v with 1 assigned coord.
- Place v



Four Matchings

Placement algorithm:

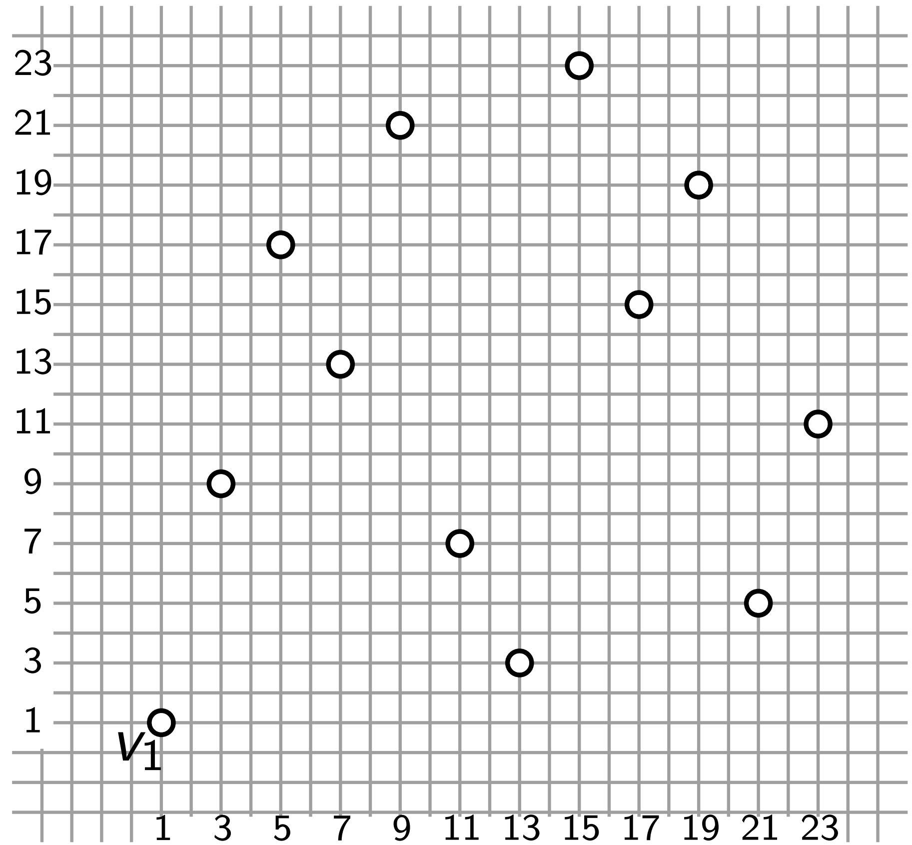
- Pick v_1, place it
- Assign x-coords. to cycle in $\mathcal{C}_1$
- Assign y-coords. to cycle in $\mathcal{C}_2$
- Pick v with 1 assigned coord.
- Place v



Four Matchings

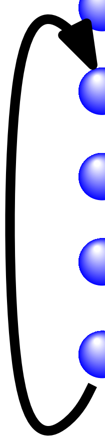
Placement algorithm:

- 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

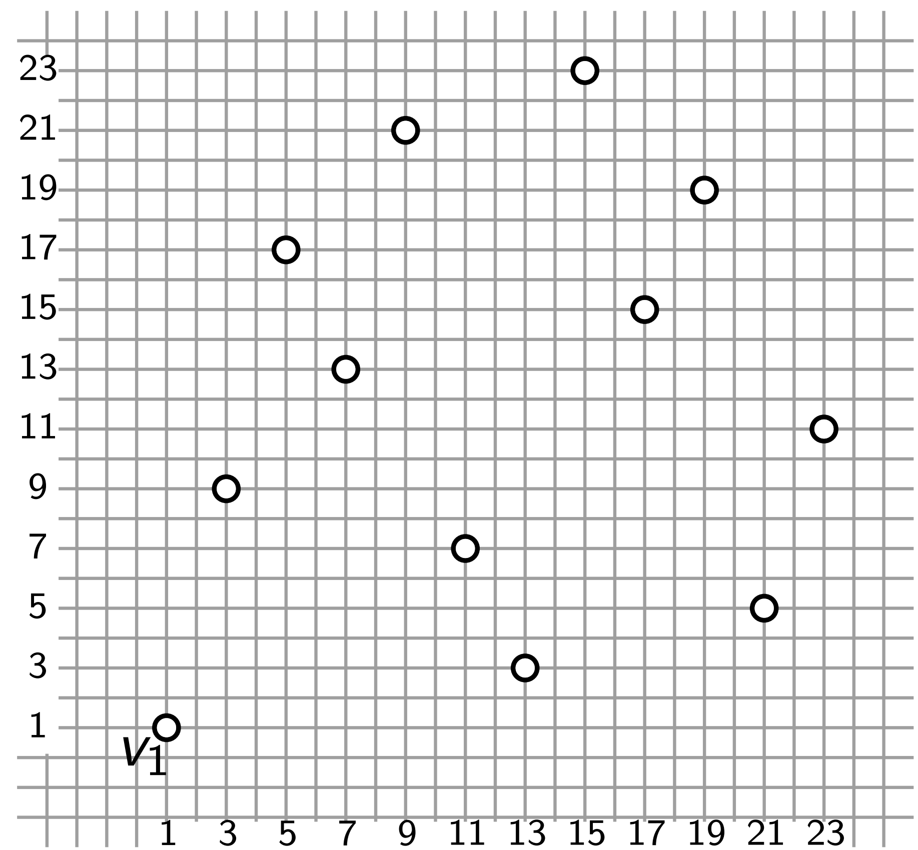


Four Matchings

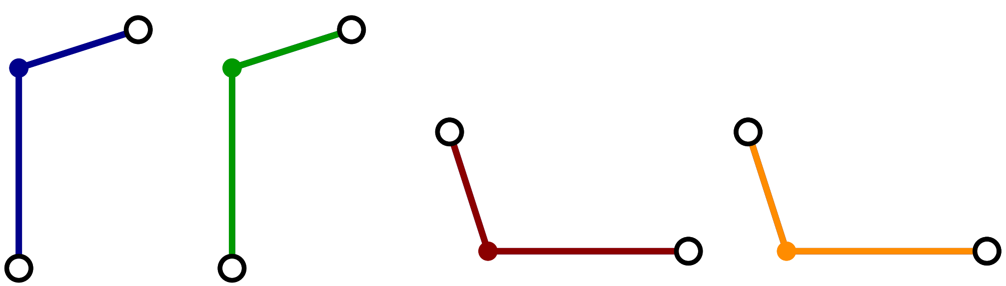
Placement algorithm:



- 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:

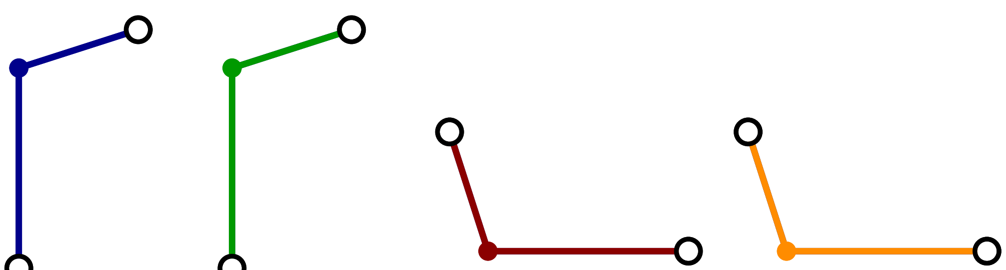


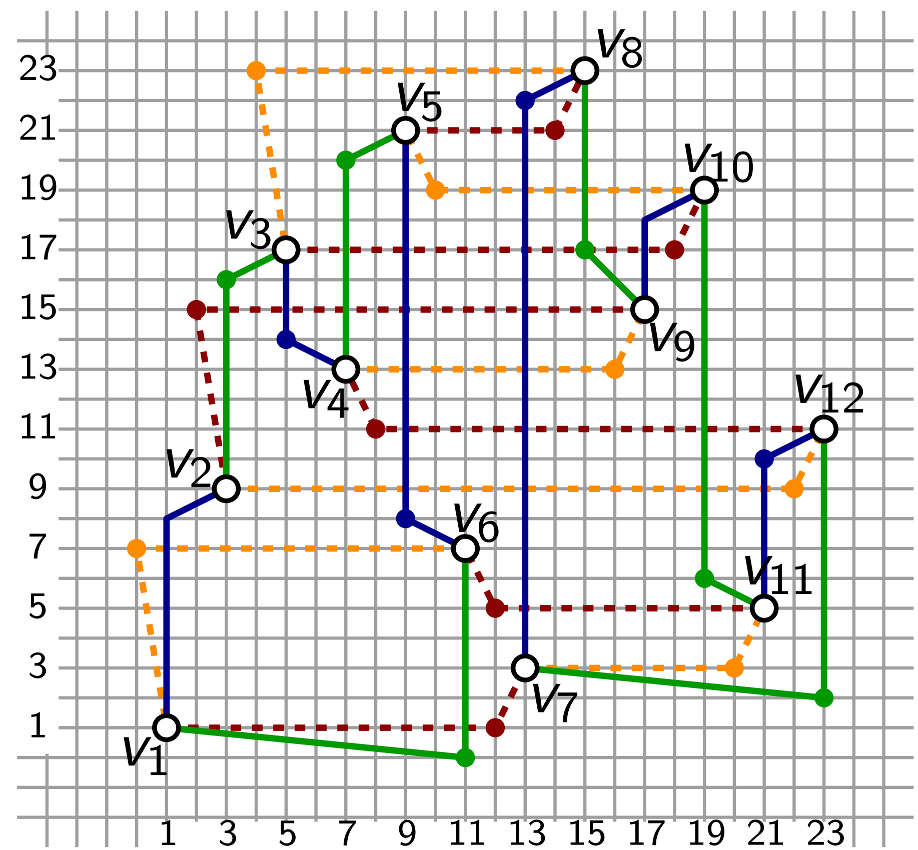
Four Matchings

Placement algorithm:

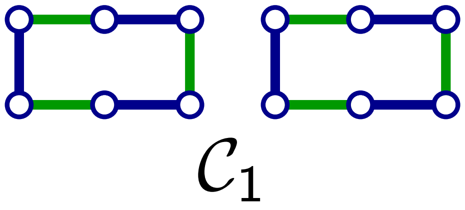
- Pick v_1, place it
- Assign x-coords. to cycle in $\mathcal{C}_1$
- Assign y-coords. to cycle in $\mathcal{C}_2$
- Pick v with 1 assigned coord.
- Place v

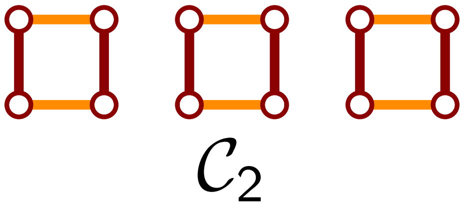
Edges:

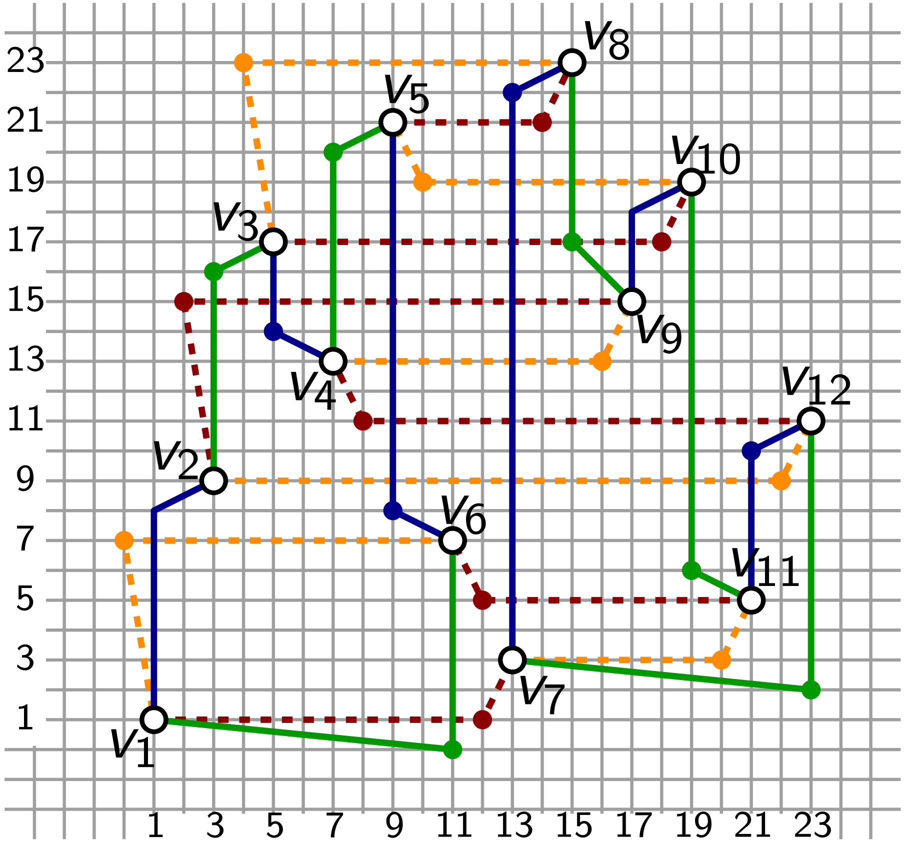




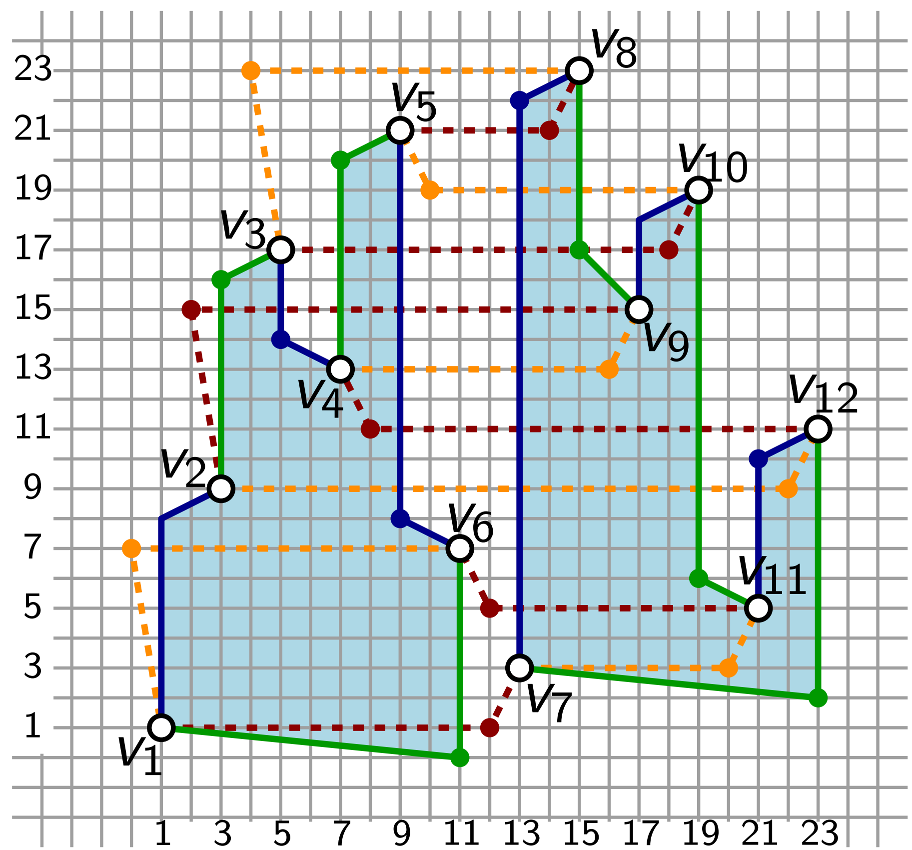
Four Matchings



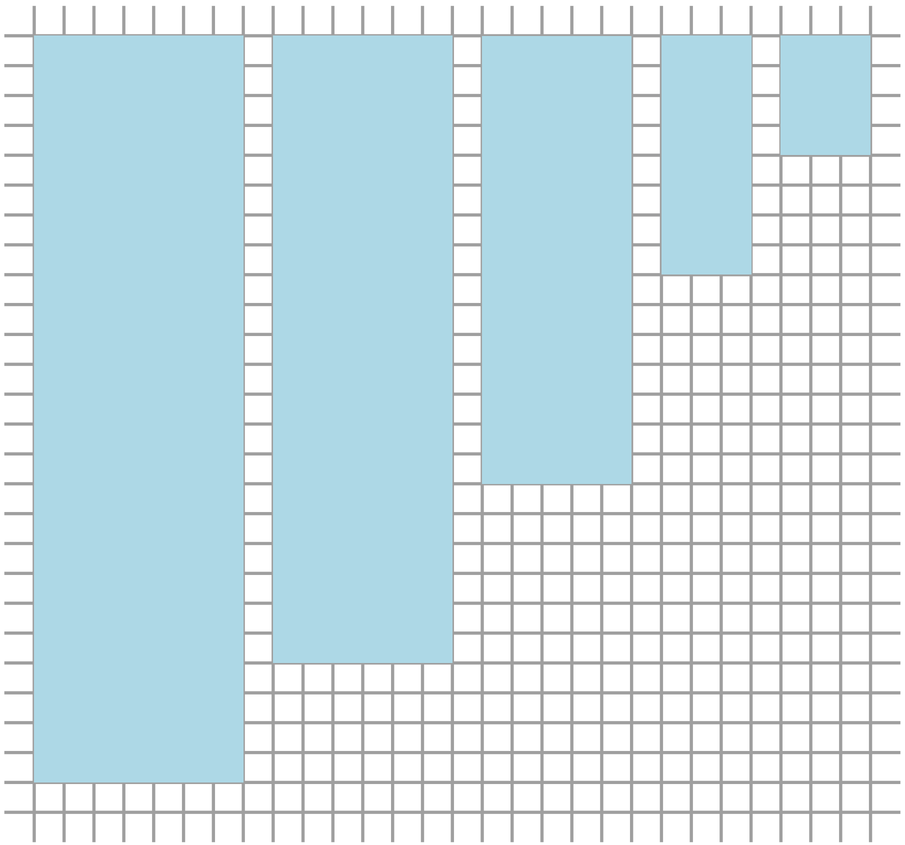


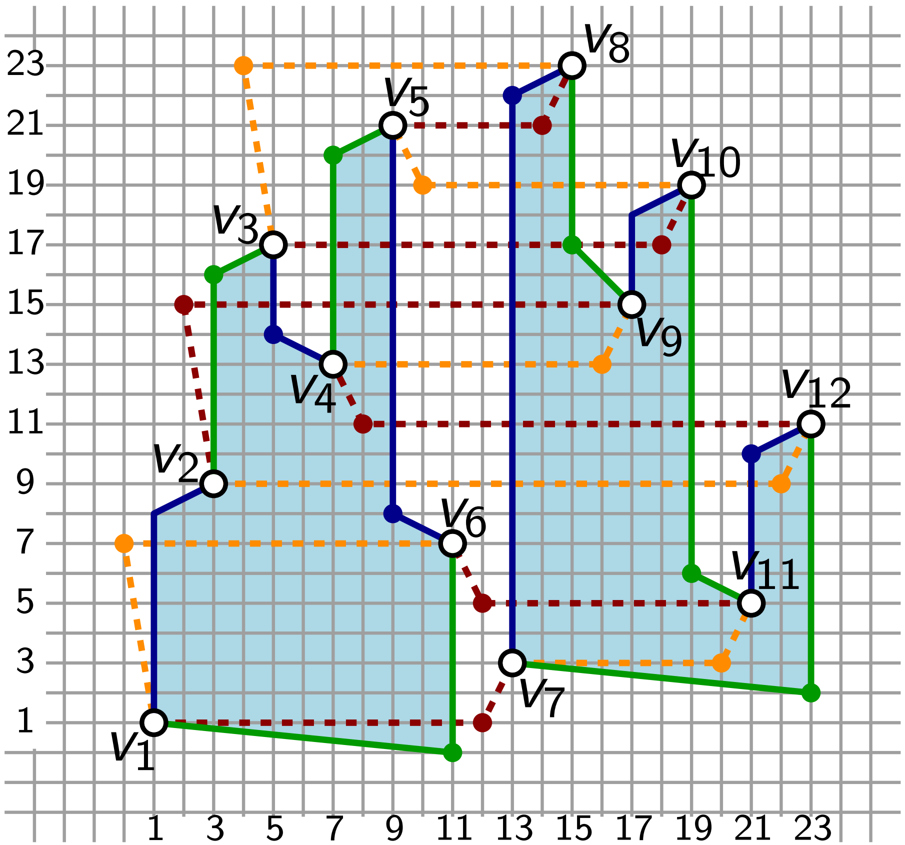


Four Matchings

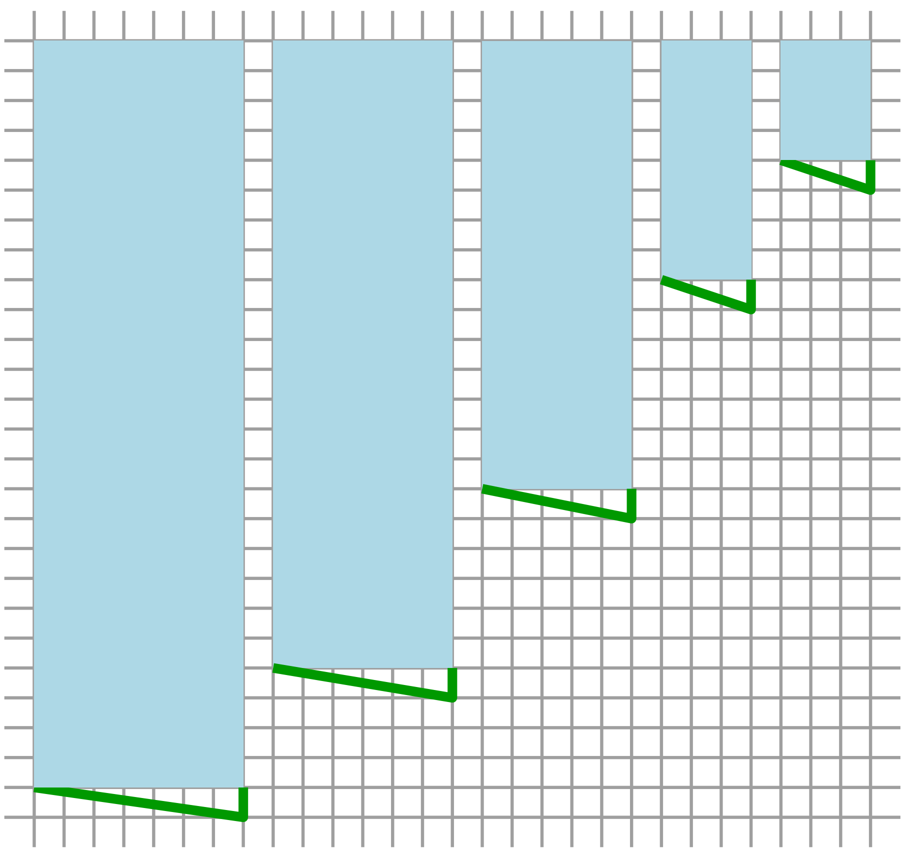


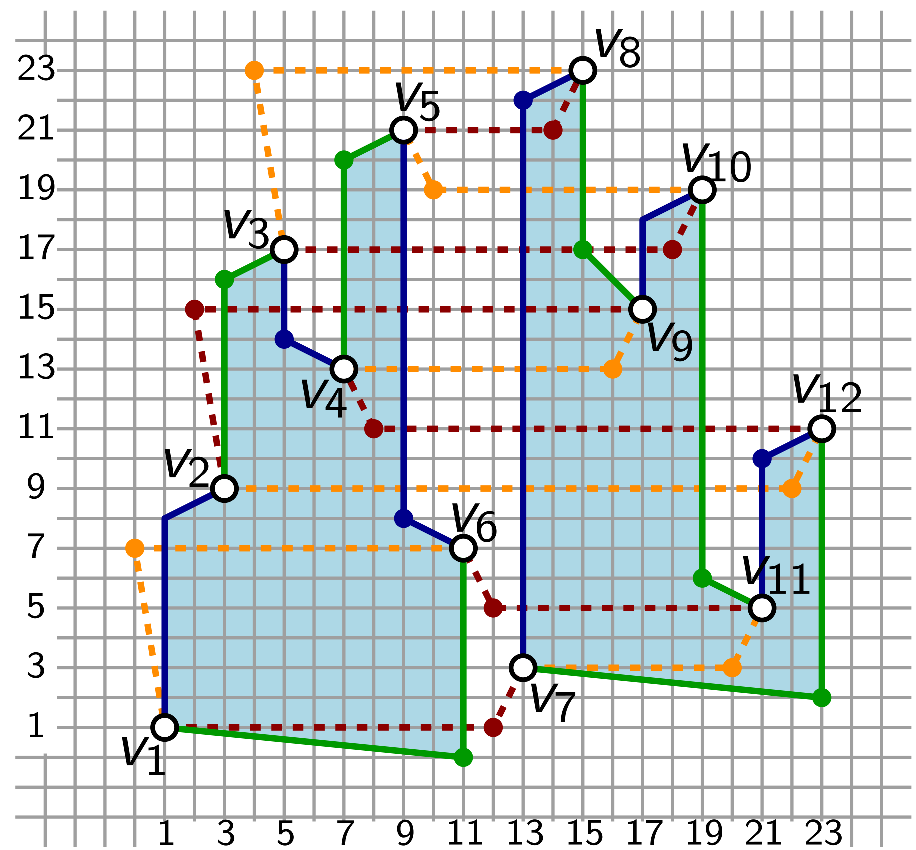
Four Matchings



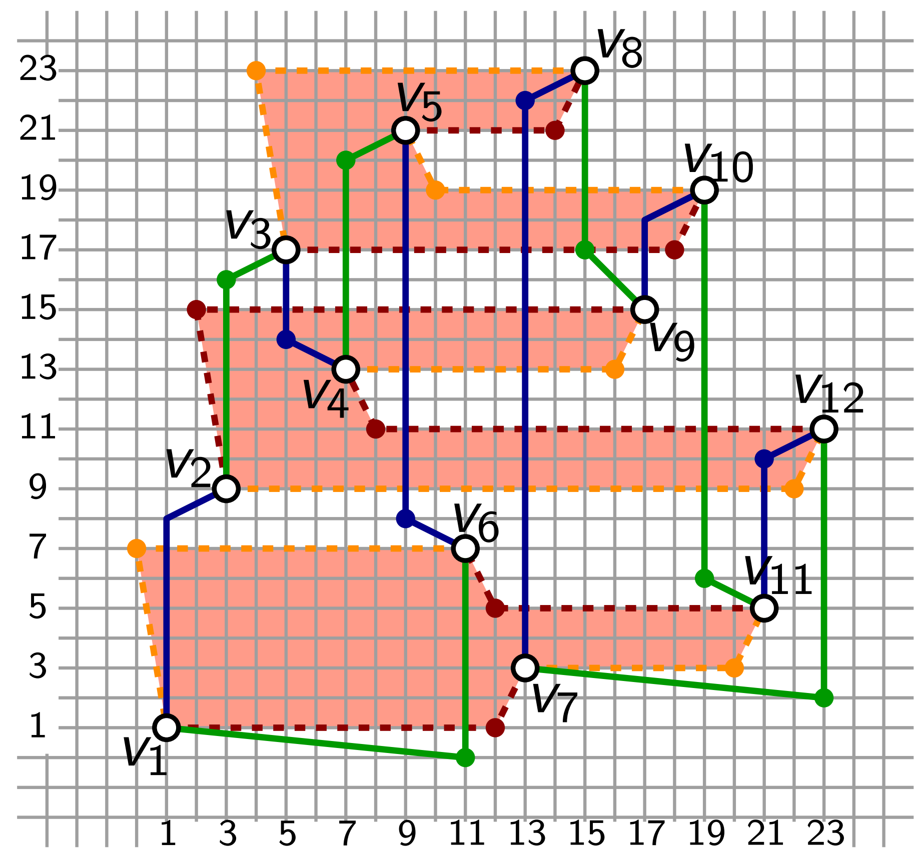


Four Matchings

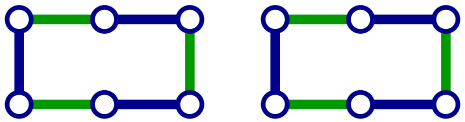




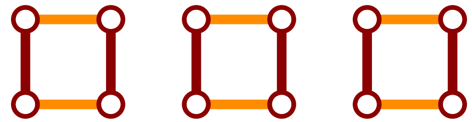
Four Matchings



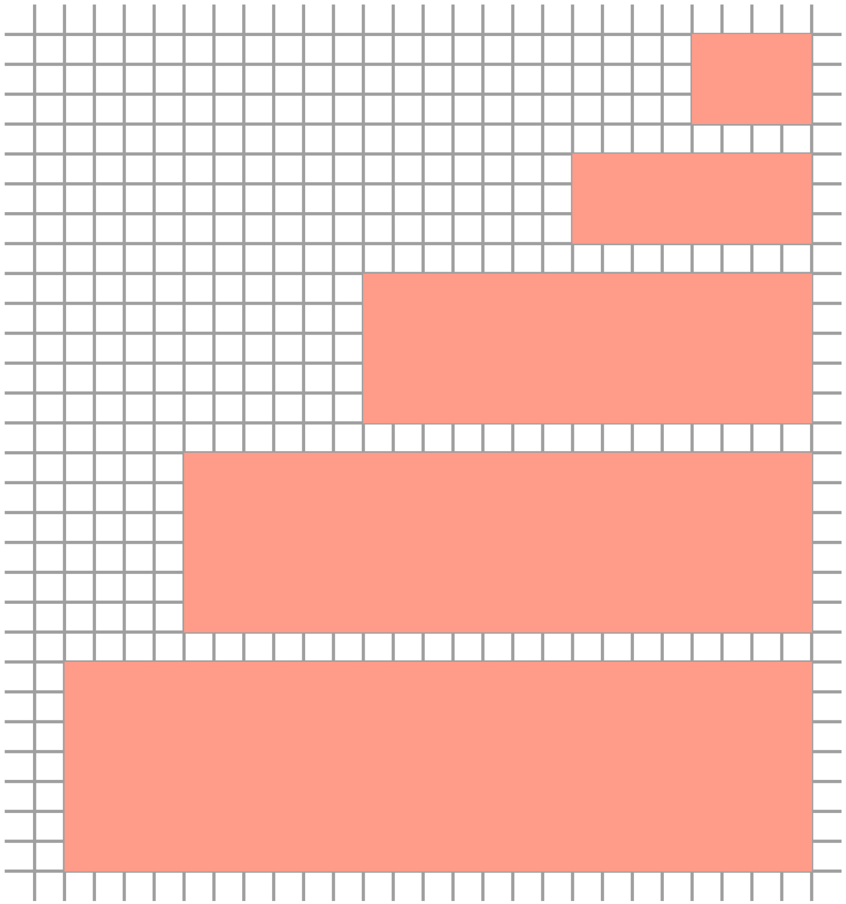
Four Matchings

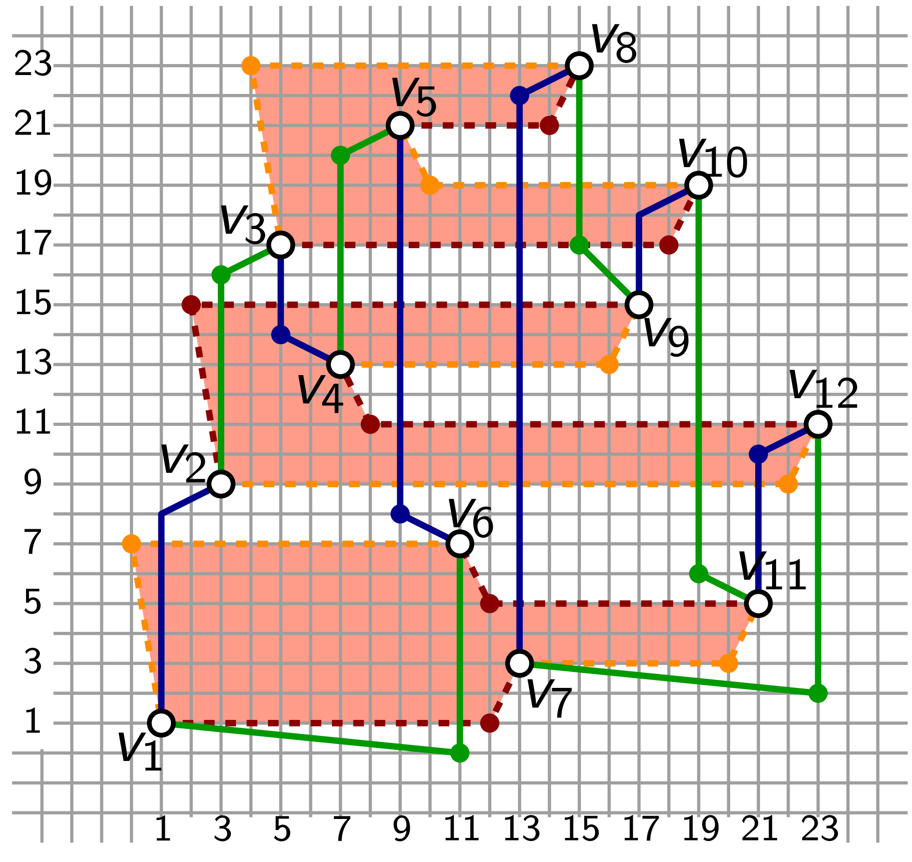


$\mathcal{C}_1$

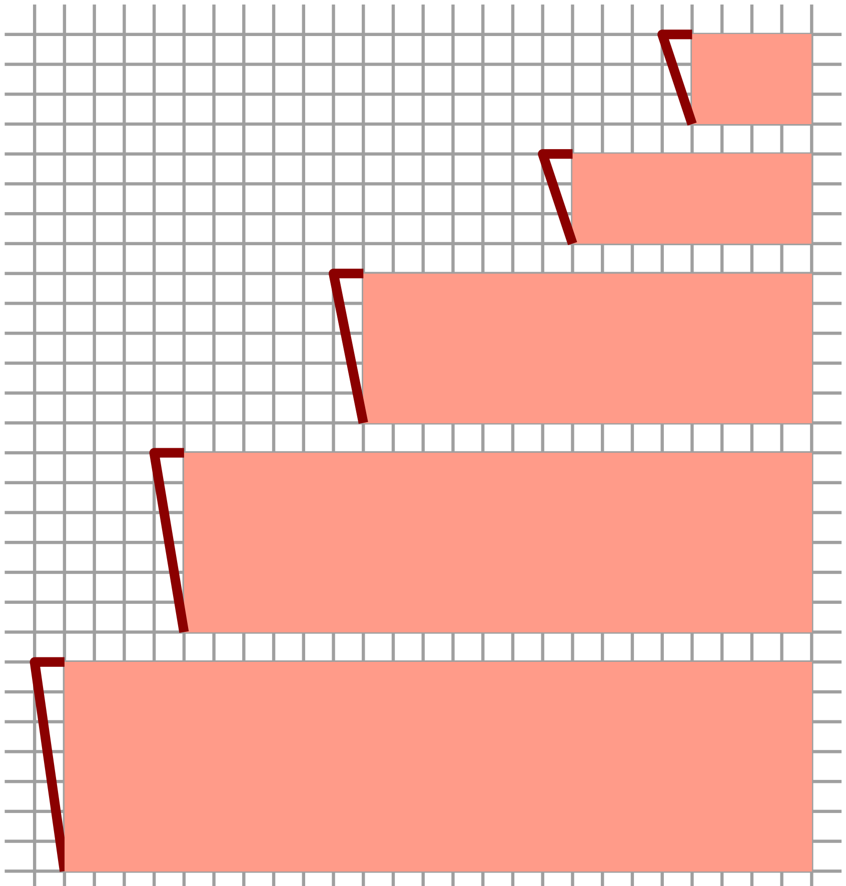


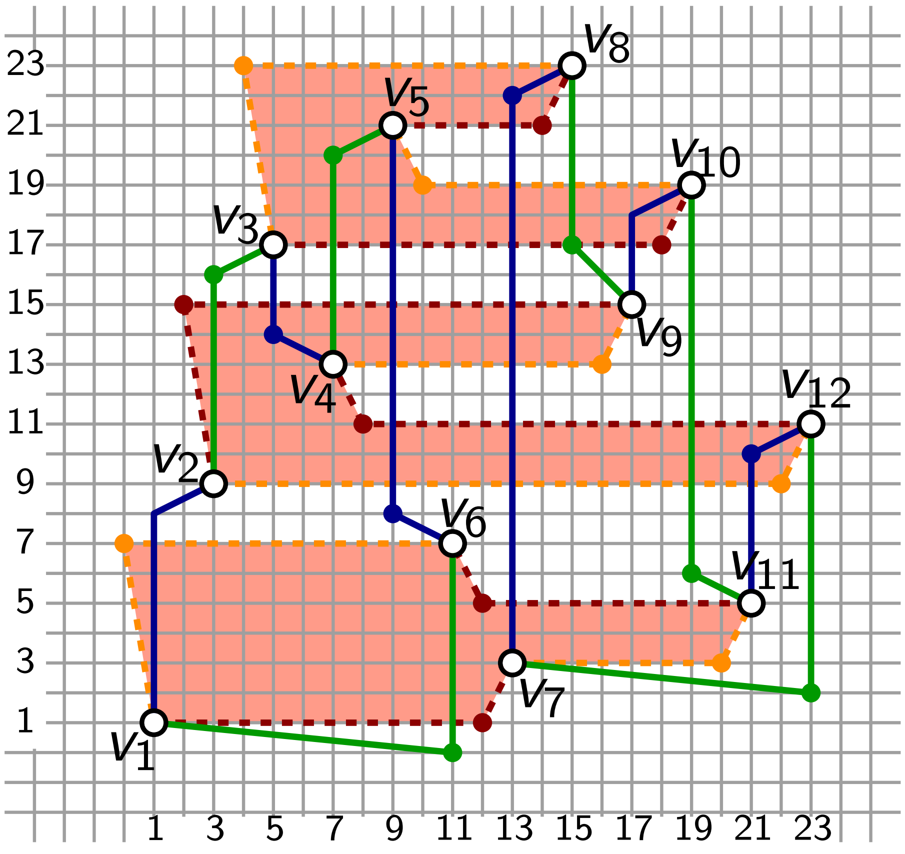
$\mathcal{C}_2$



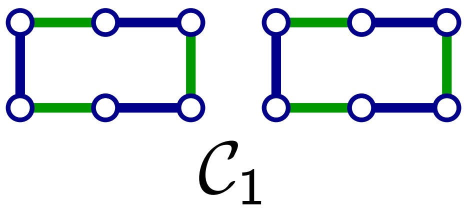


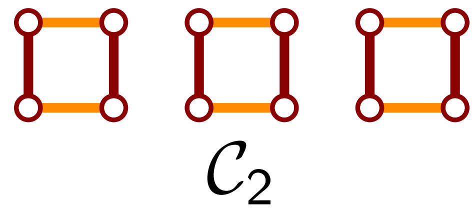
Four Matchings





Four Matchings

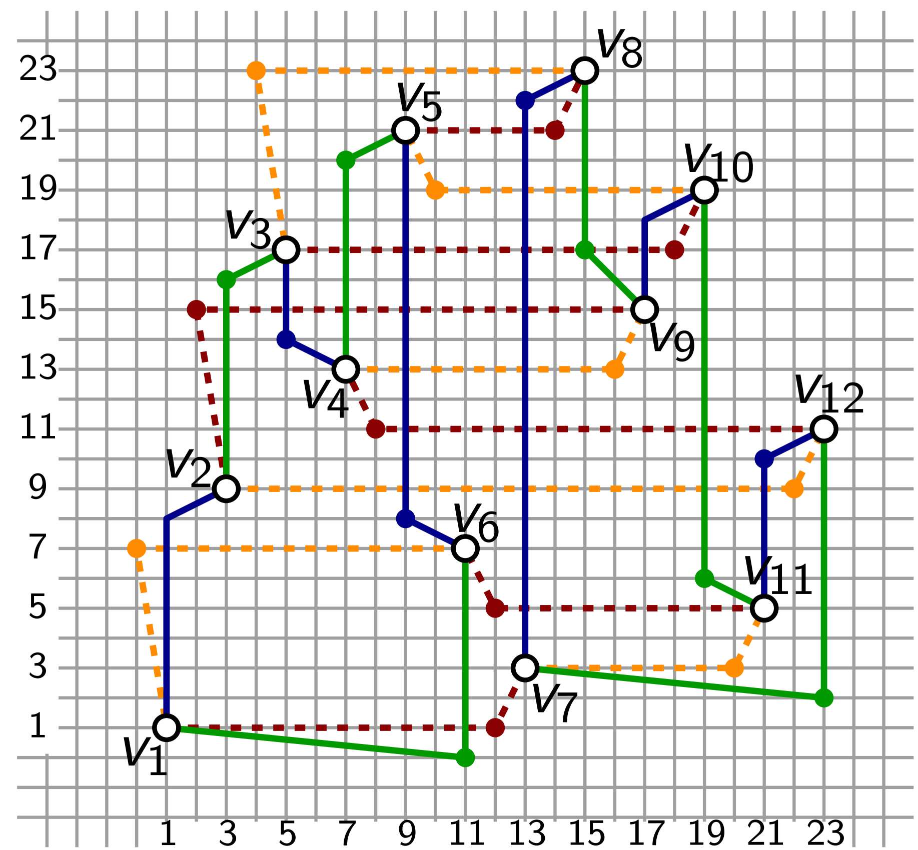




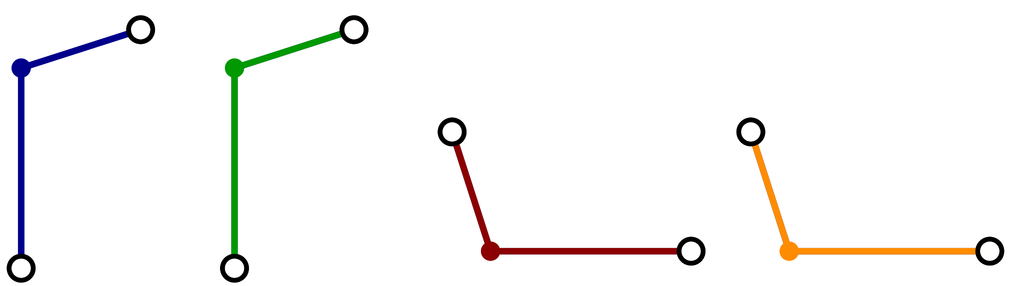
Bends:	1×1
Grid size:	$2n \times 2n$

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 × Matching

Idea: • Matching edges horizontal, tree edges with 1 bend

Tree $\times$ Matching

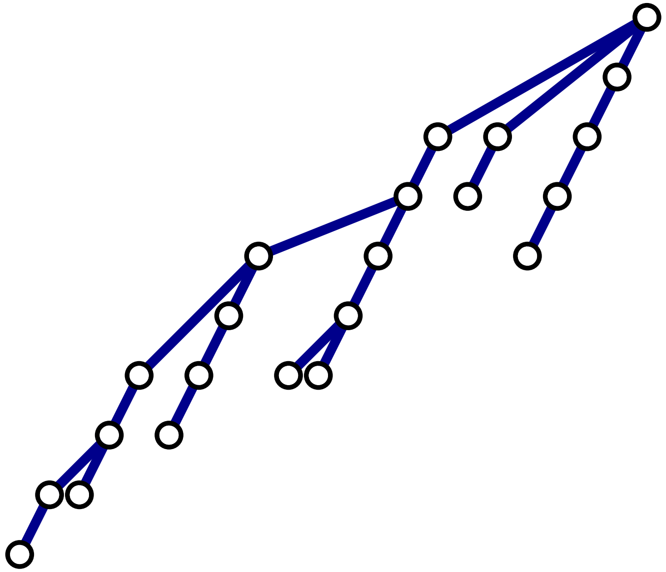
Idea:
- Matching edges horizontal, tree edges with 1 bend
- Place matching edges inductively $\Rightarrow$ y-coord.

Tree × Matching

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

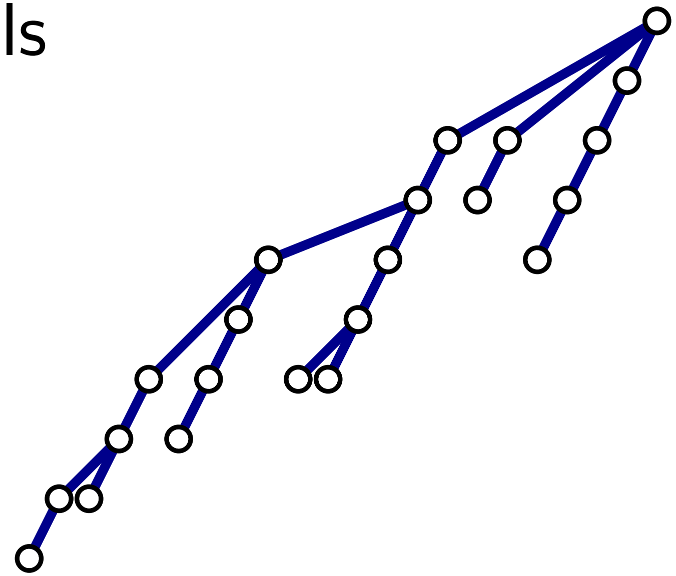
Tree × Matching

Idea:
- Matching edges horizontal, tree edges with 1 bend
- Place matching edges inductively ⇒ y-coord.
- Use post-order on tree ⇒ x-coord.



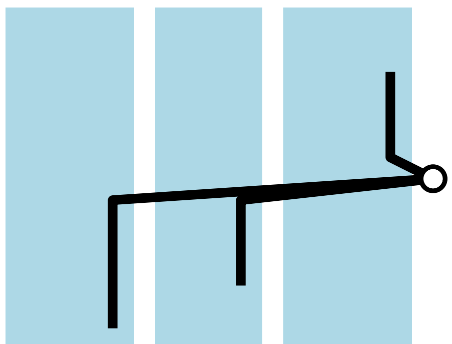
Tree × Matching

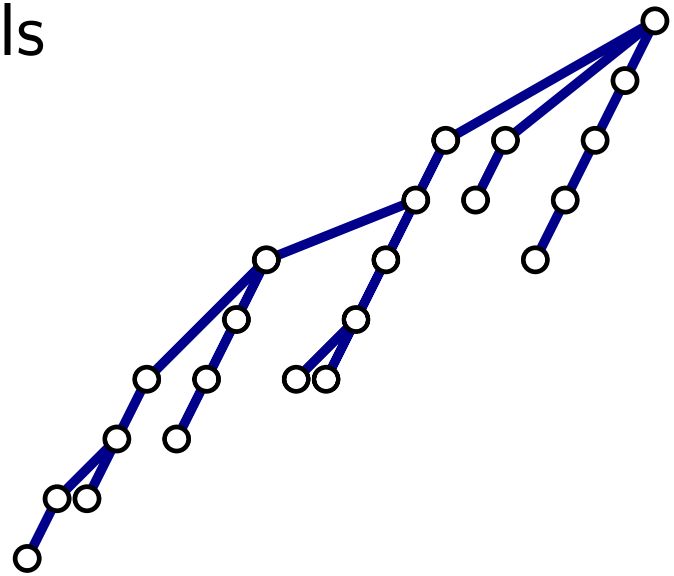
Idea:
- Matching edges horizontal, tree edges with 1 bend
- Place matching edges inductively ⇒ y-coord.
- Use post-order on tree ⇒ x-coord.
- ⇒ Subtrees in disjoint x-intervals



Tree × Matching

Idea:
- Matching edges horizontal, tree edges with 1 bend
- Place matching edges inductively ⇒ y-coord.
- Use post-order on tree ⇒ x-coord.
- ⇒ Subtrees in disjoint x-intervals

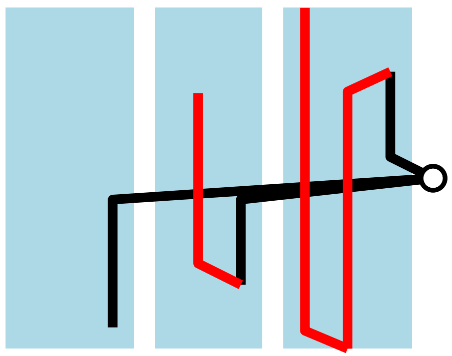


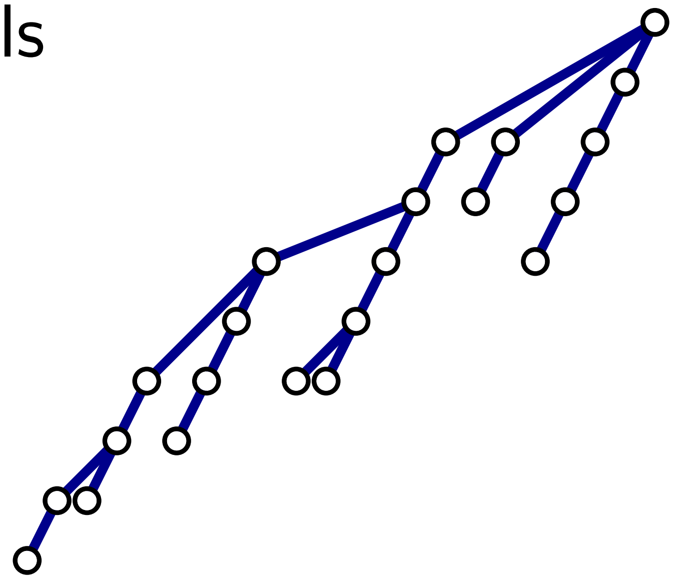


Tree × Matching

Idea:
- Matching edges horizontal, tree edges with 1 bend
- Place matching edges inductively ⇒ y-coord.
- Use post-order on tree ⇒ x-coord.
- ⇒ Subtrees in disjoint x-intervals

Problem:

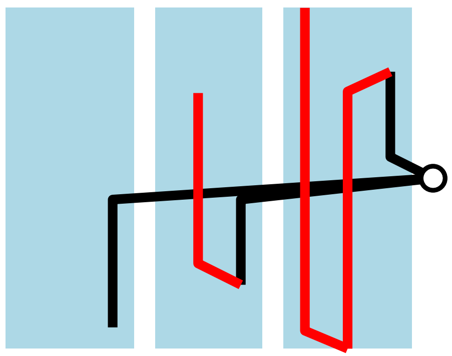




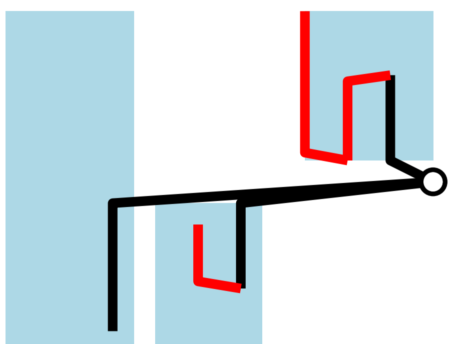
Tree × Matching

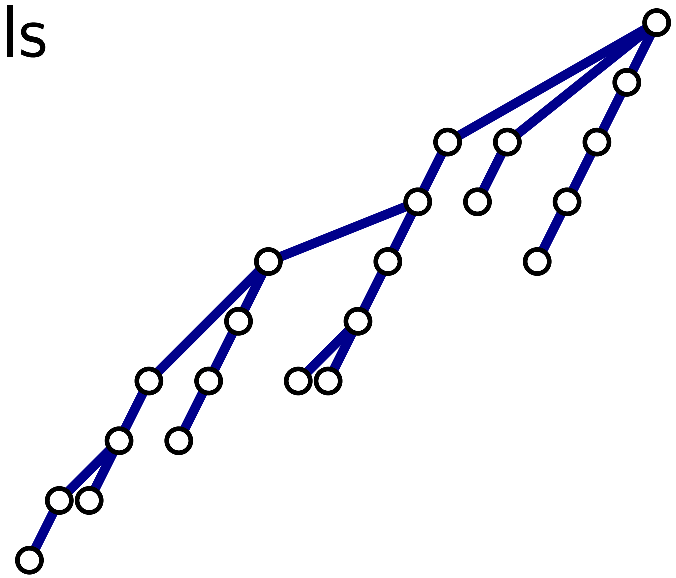
Idea:
- Matching edges horizontal, tree edges with 1 bend
- Place matching edges inductively ⇒ y-coord.
- Use post-order on tree ⇒ x-coord.
- ⇒ Subtrees in disjoint x-intervals

Problem:



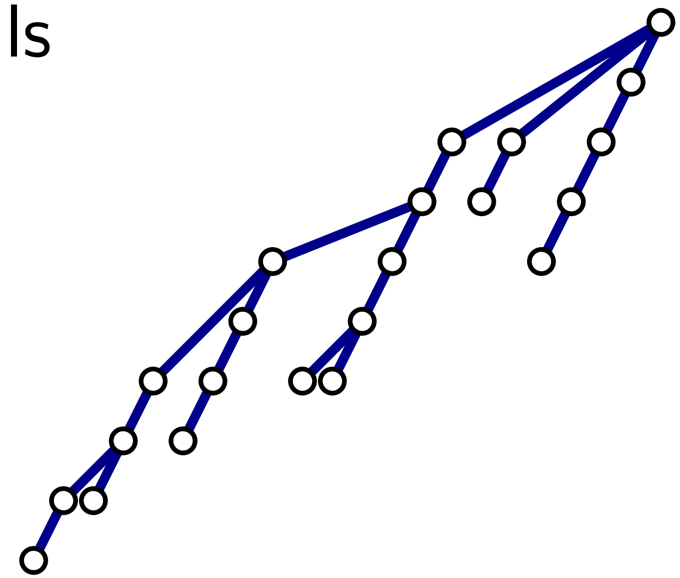
Solution:



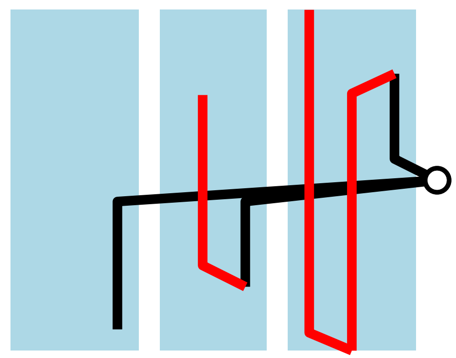


Tree × Matching

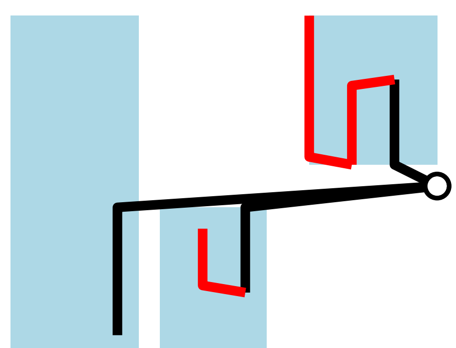
Idea:
- Matching edges horizontal, tree edges with 1 bend
- Place matching edges inductively ⇒ y-coord.
- Use post-order on tree ⇒ x-coord.
- ⇒ Subtrees in disjoint x-intervals



Problem:



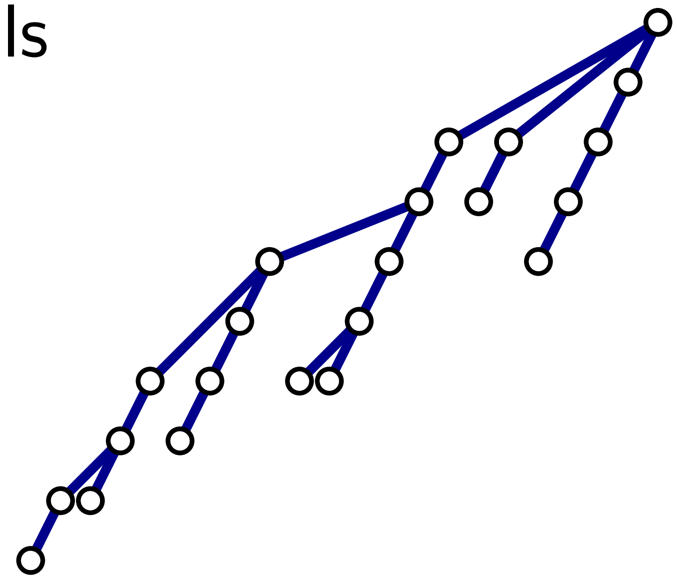
Solution:



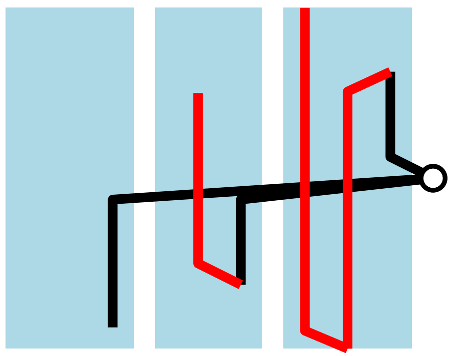
All but one subtree completely above or completely below.

Tree × Matching

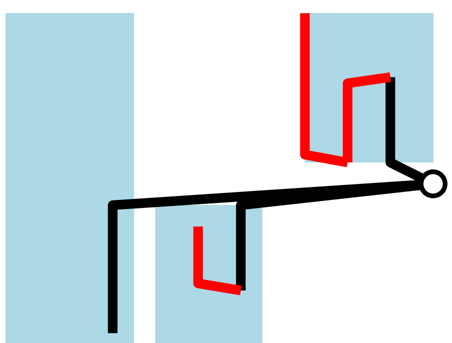
Idea:
- Matching edges horizontal, tree edges with 1 bend
- Place matching edges inductively ⇒ y-coord.
- Use post-order on tree ⇒ x-coord.
- ⇒ Subtrees in disjoint x-intervals



Problem:



Solution:

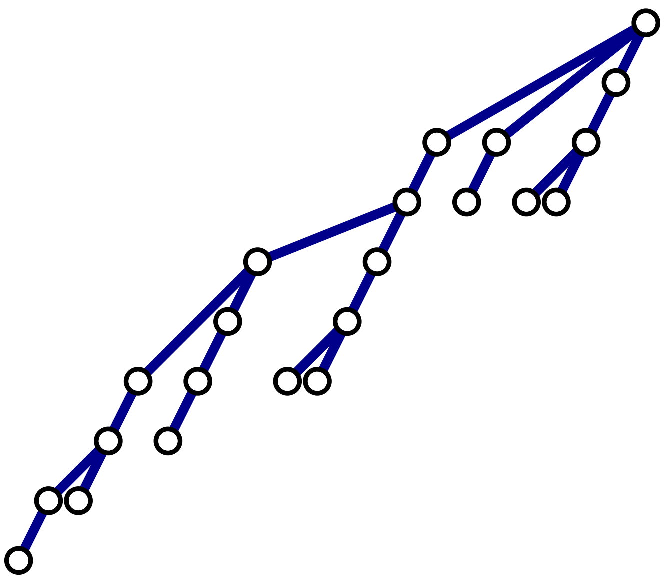


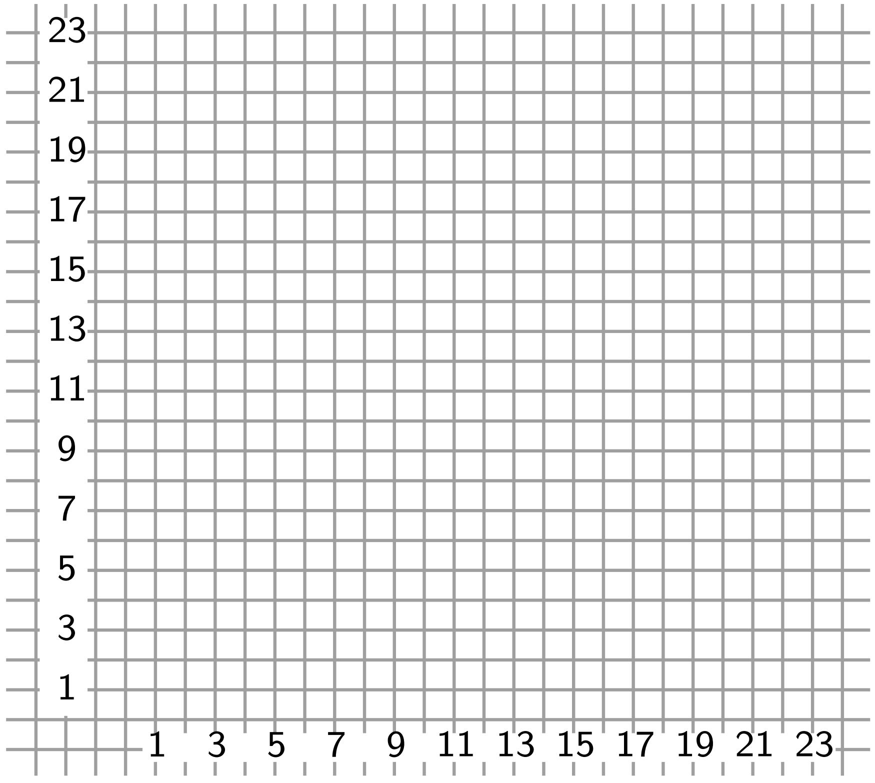
All but one subtree completely above or completely below.

Main ideas adopted from [Cabello et al. JGAA'11, Di Giacomo et al. JGAA'09].

Tree × Matching

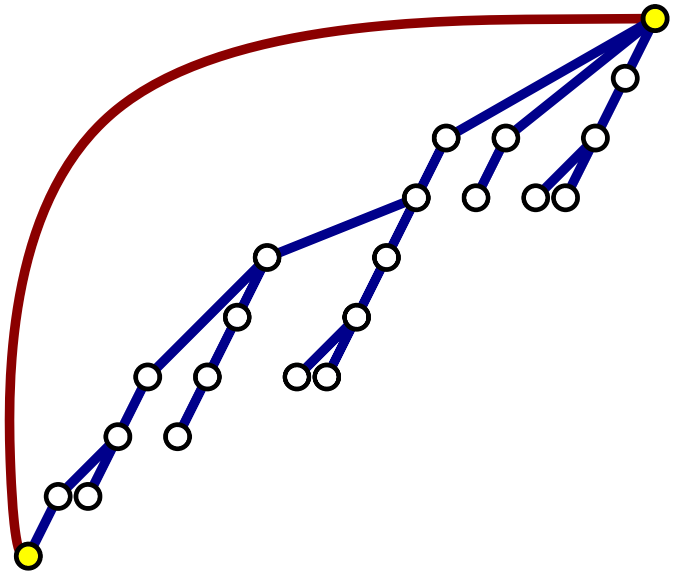
- Place root + matching at the top

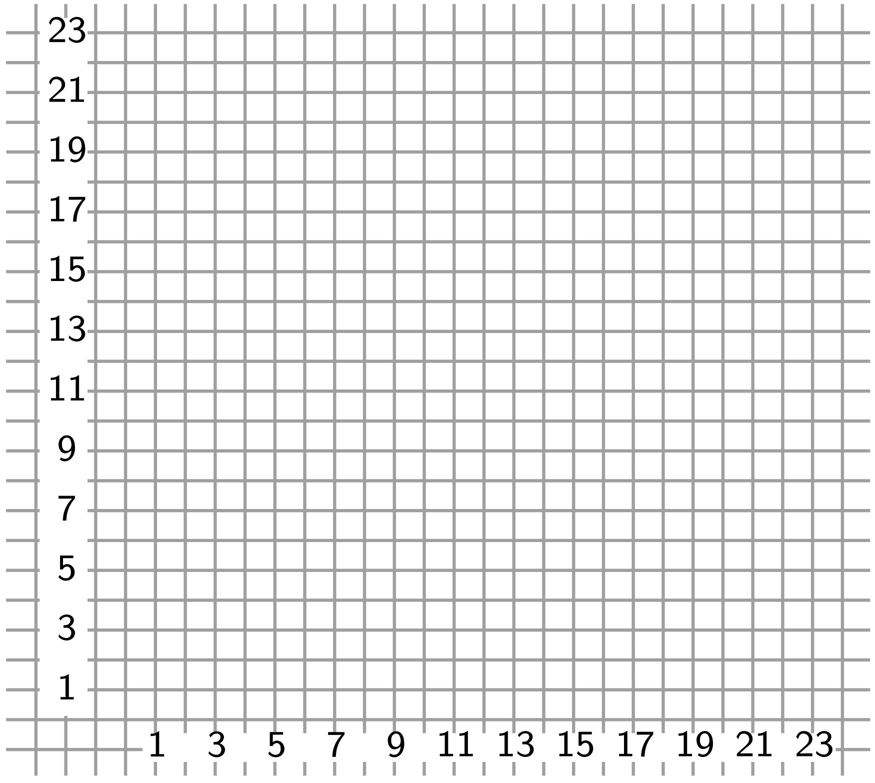




Tree × Matching

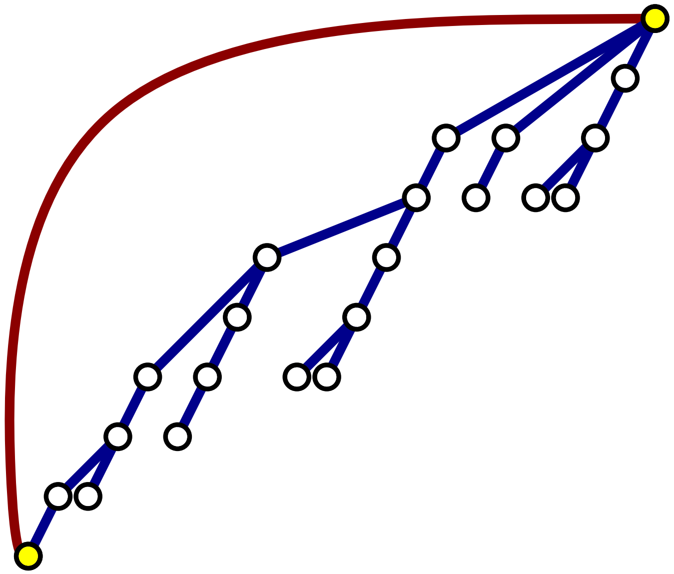
- Place root + matching at the top

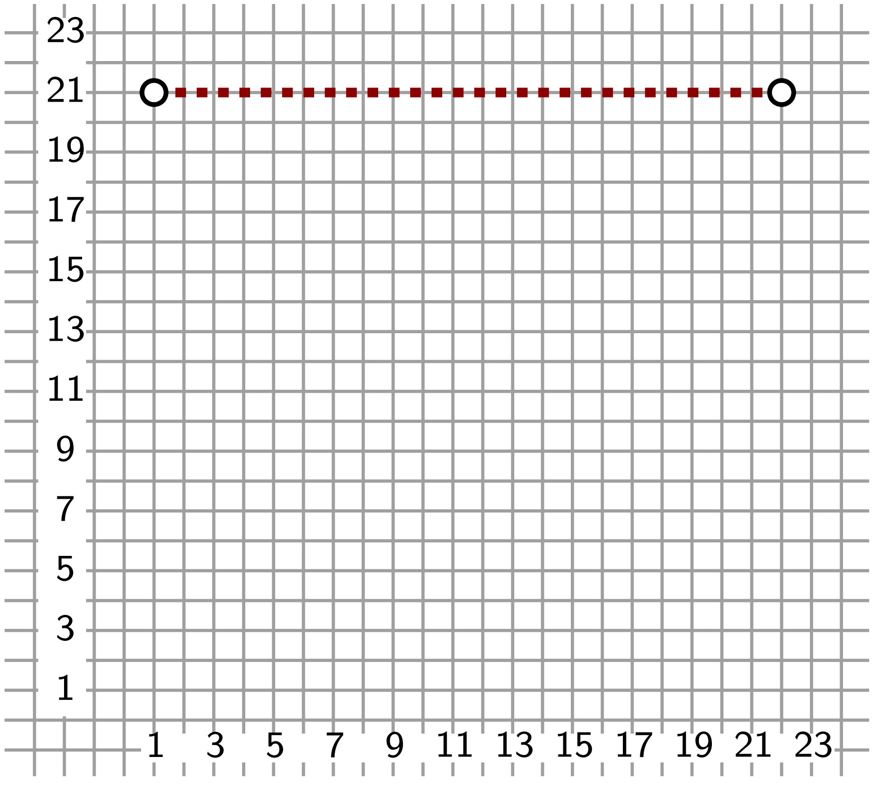




Tree × Matching

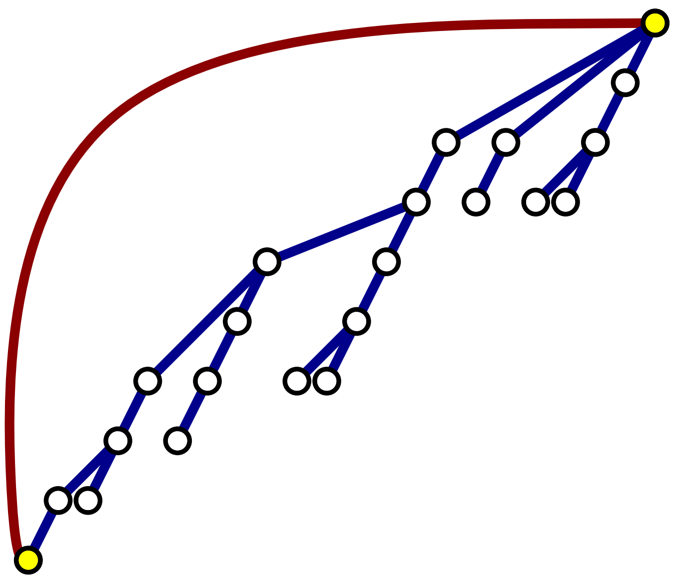
- Place root + matching at the top

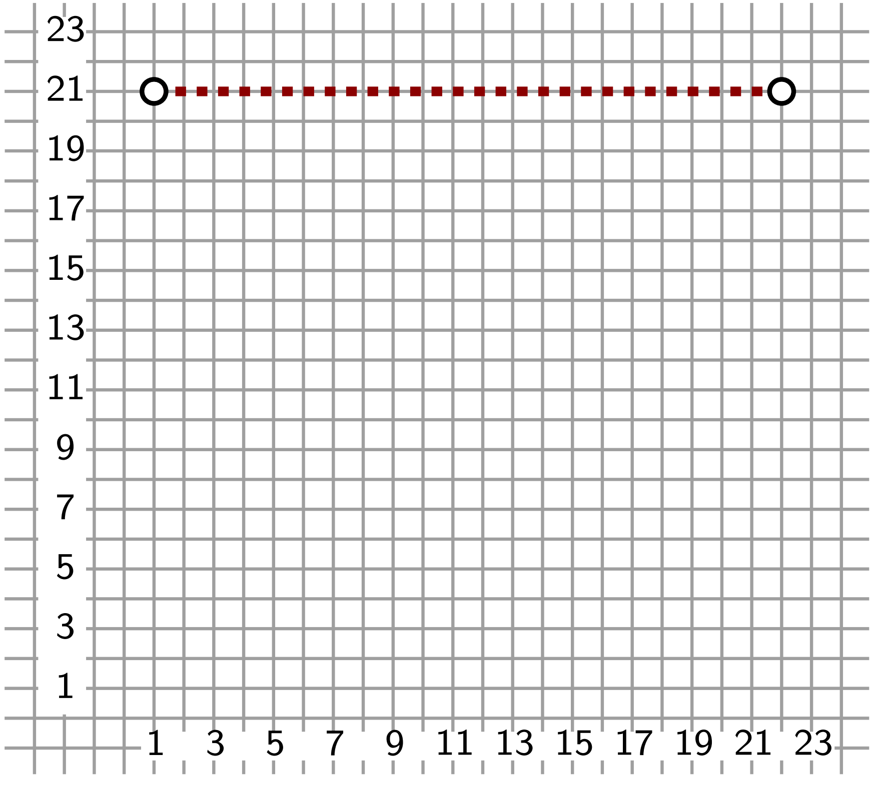




Tree × Matching

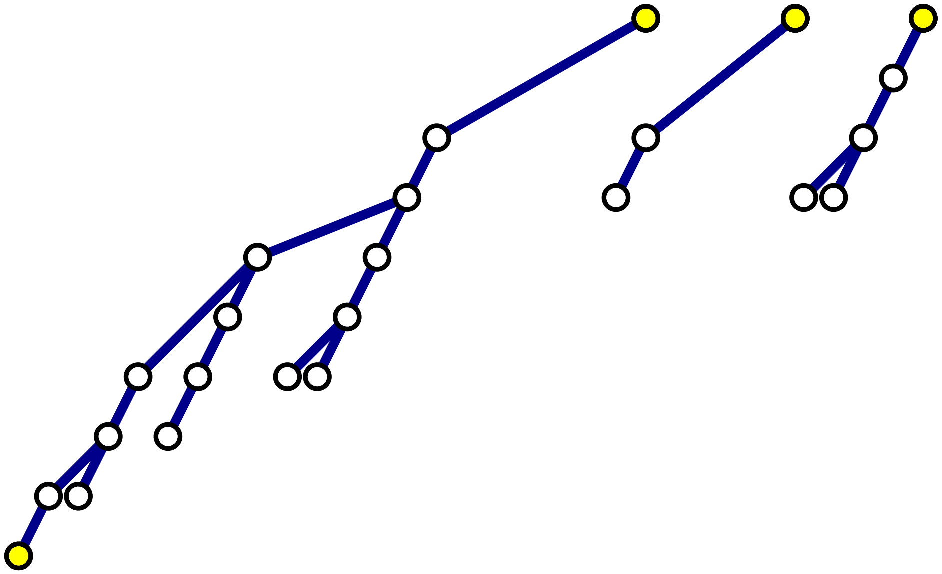
- Place root + matching at the top
- Split the tree

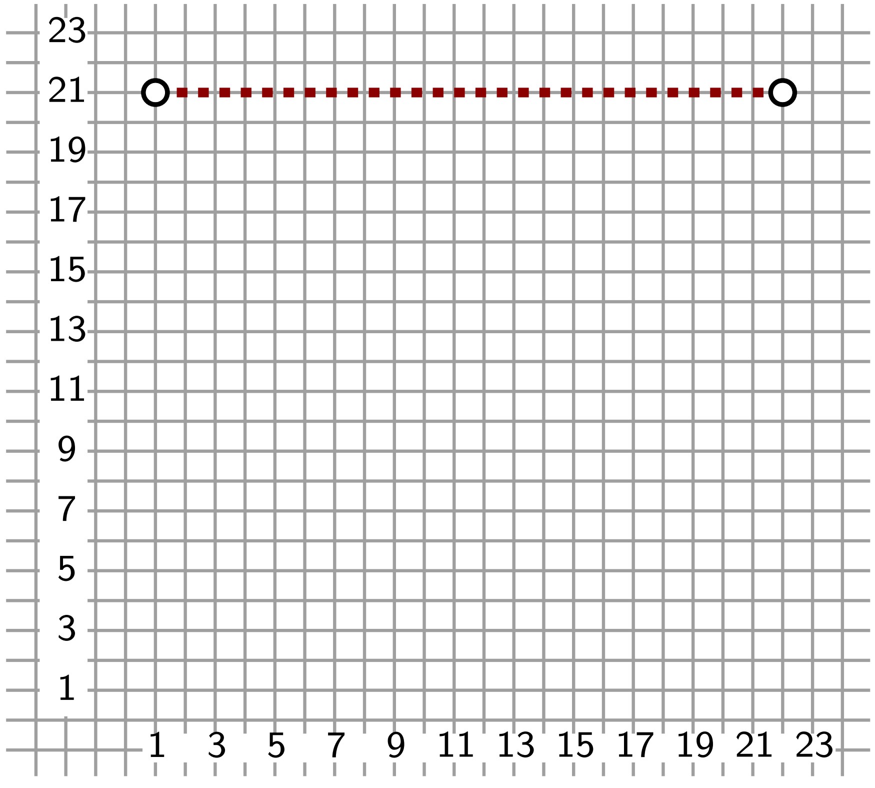




Tree × Matching

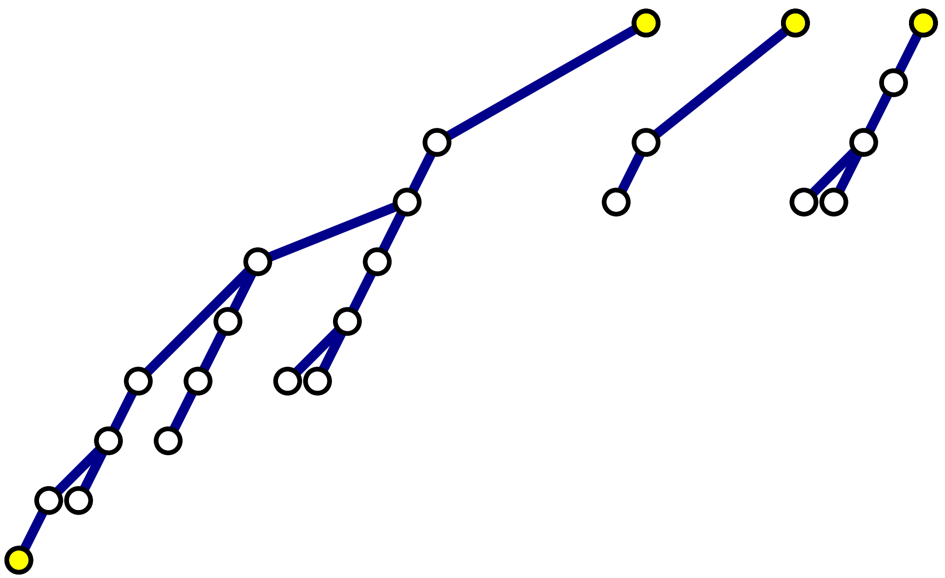
- Place root + matching at the top
- Split the tree

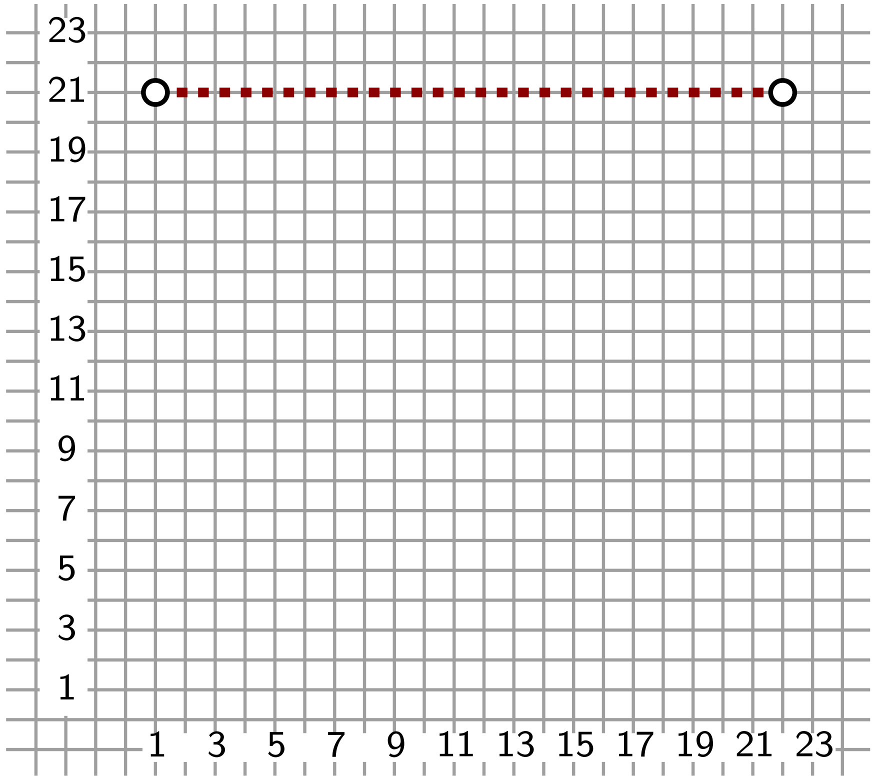




Tree × Matching

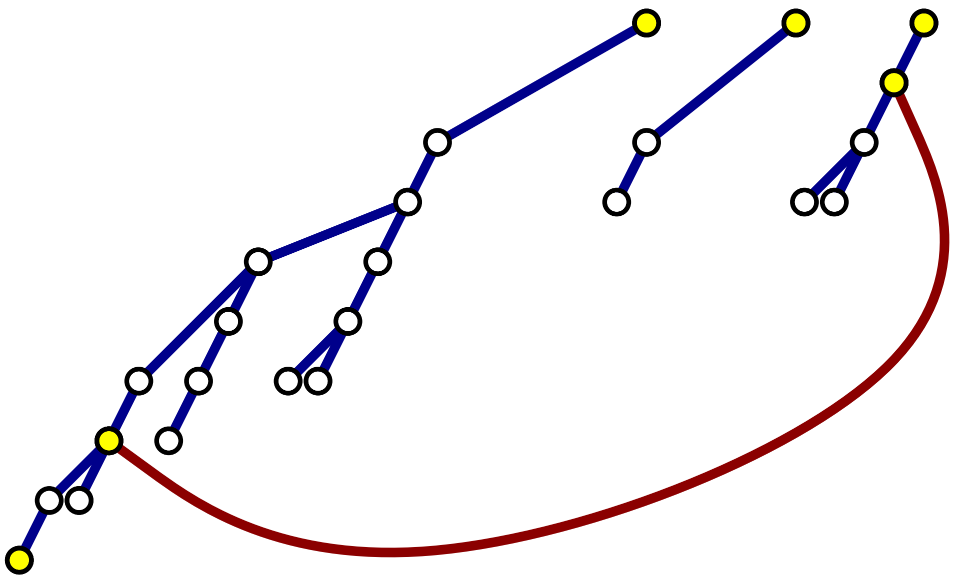
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top

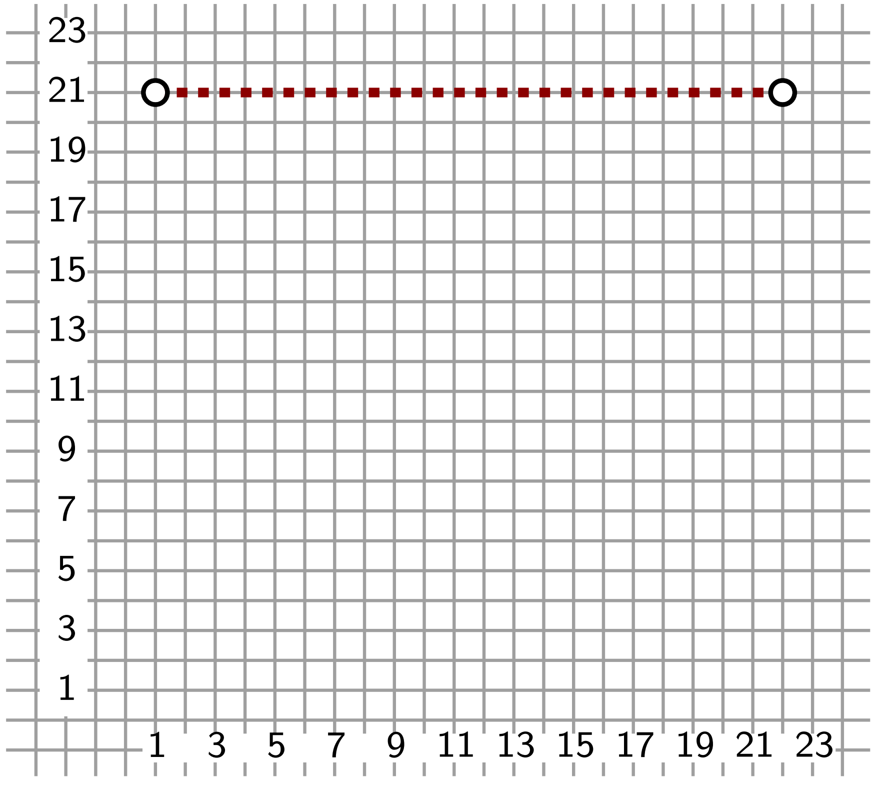




Tree × Matching

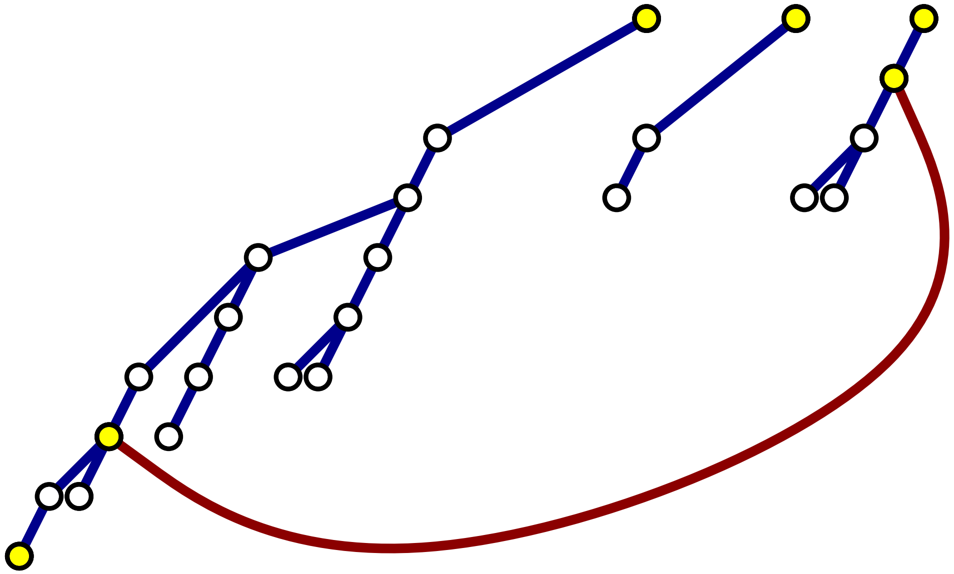
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top

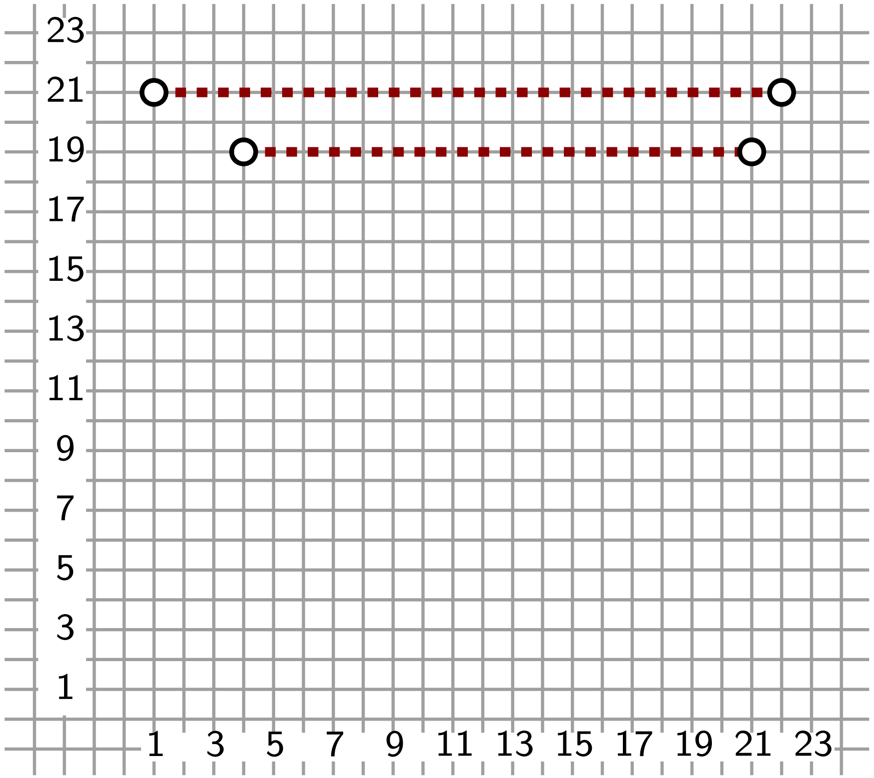




Tree × Matching

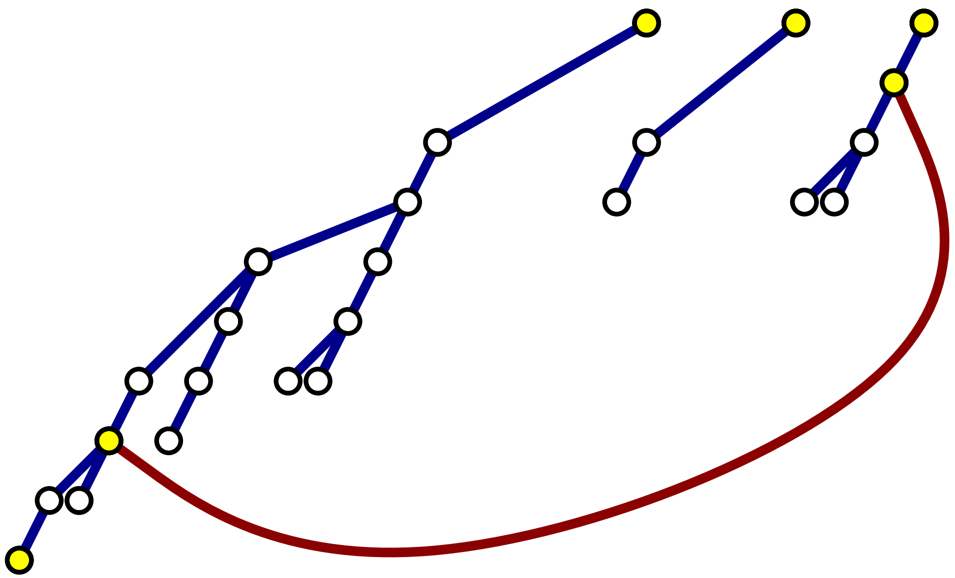
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top

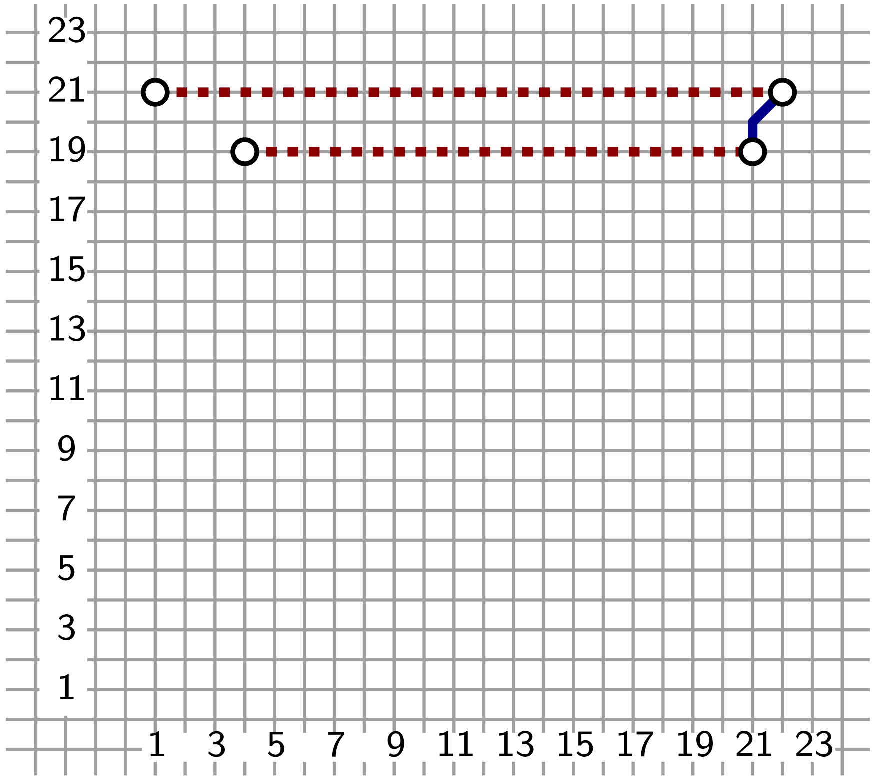




Tree × Matching

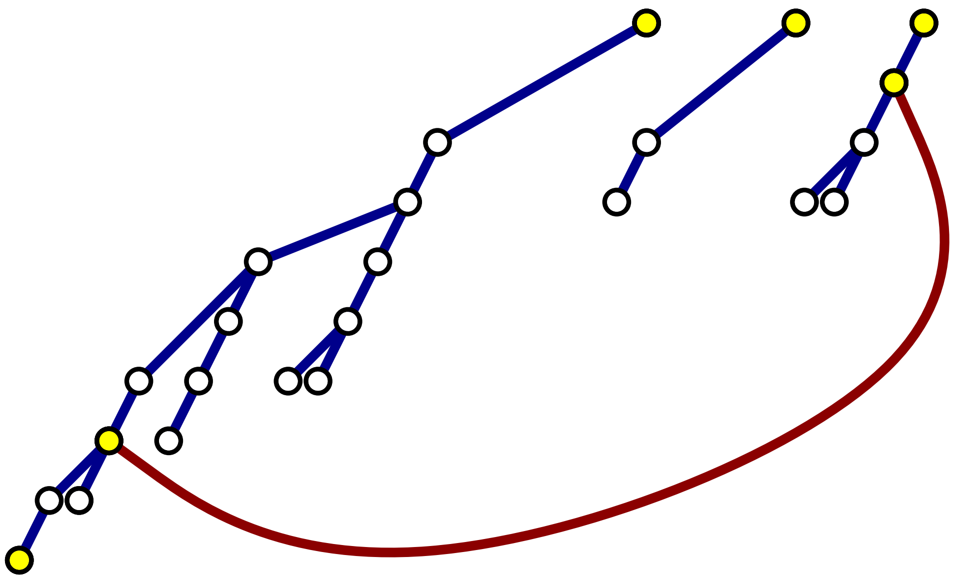
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top

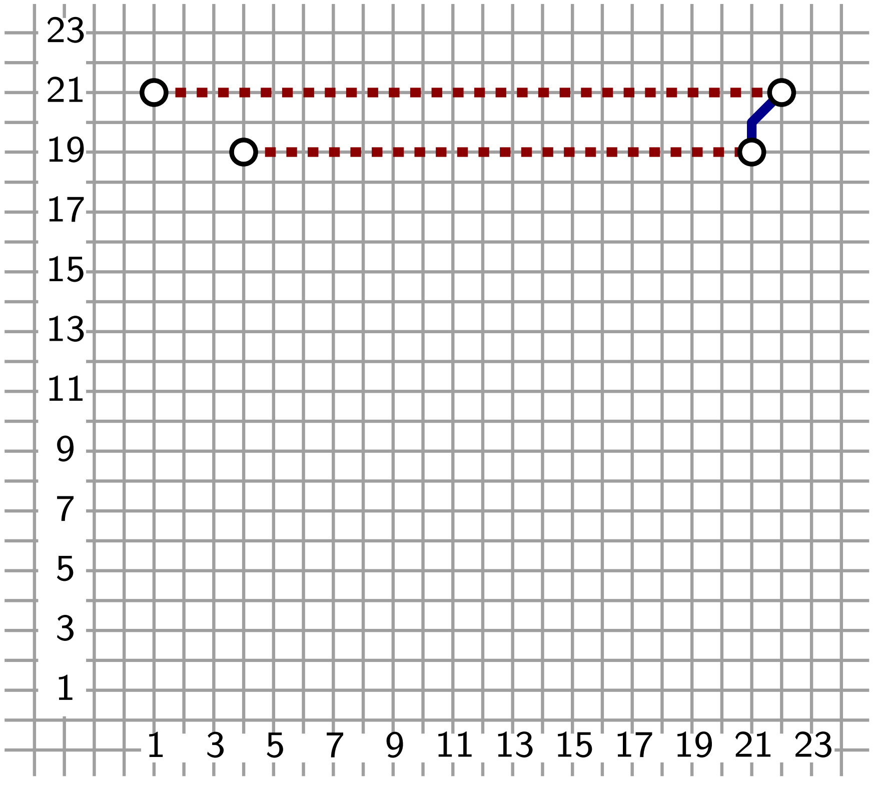




Tree × Matching

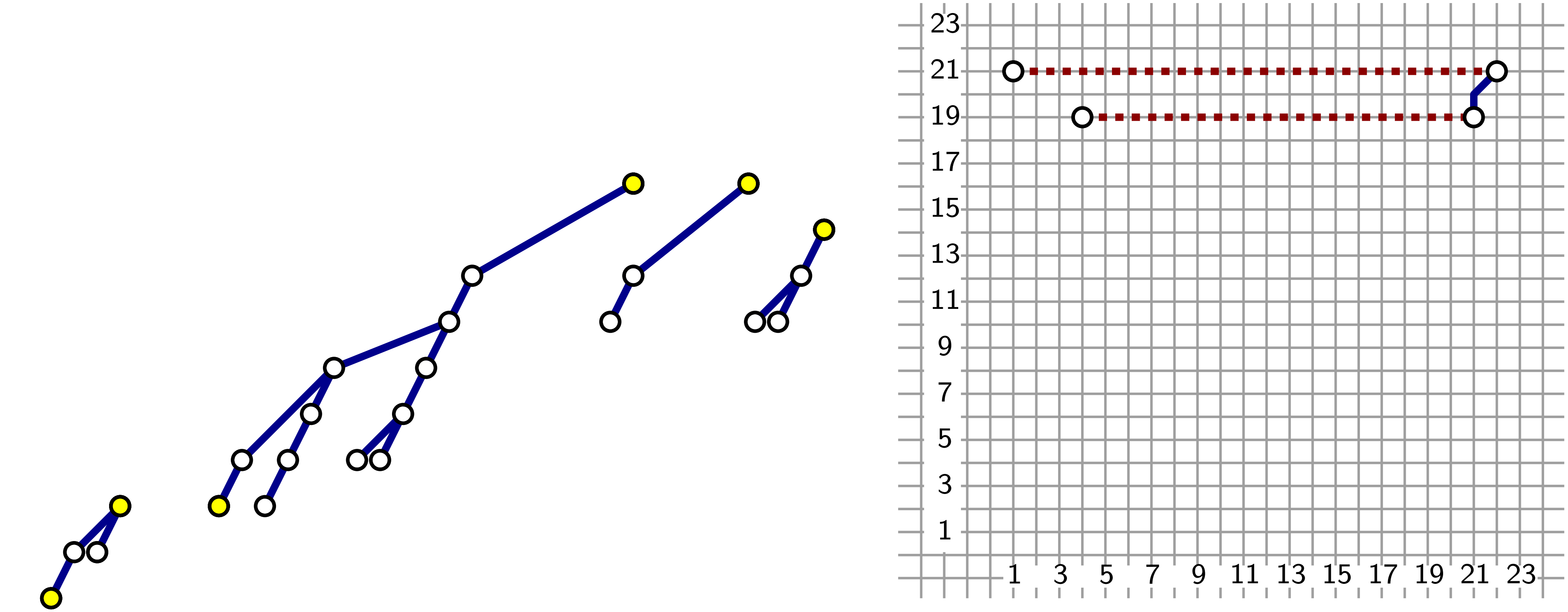
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top





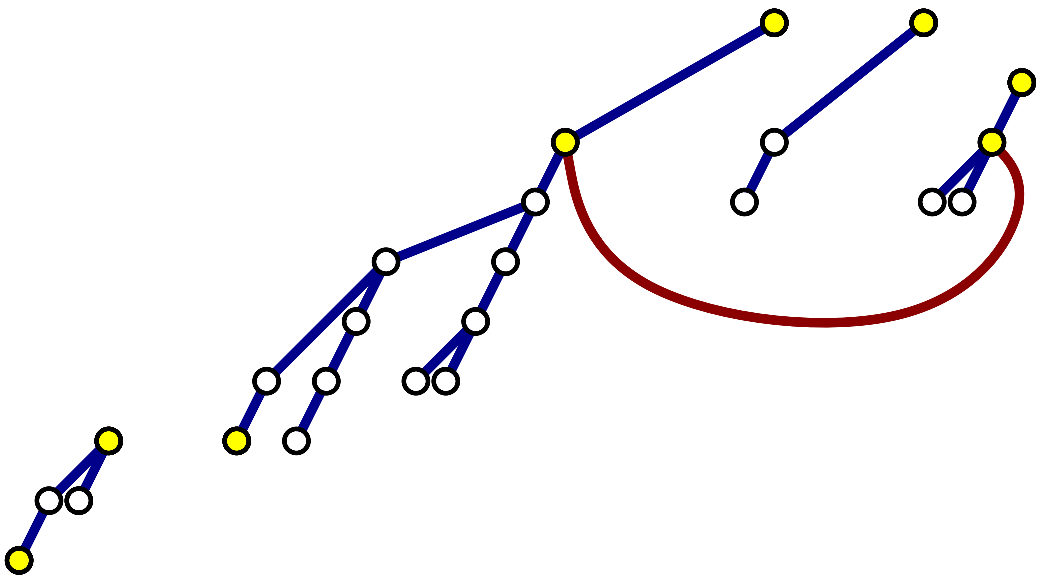
Tree × Matching

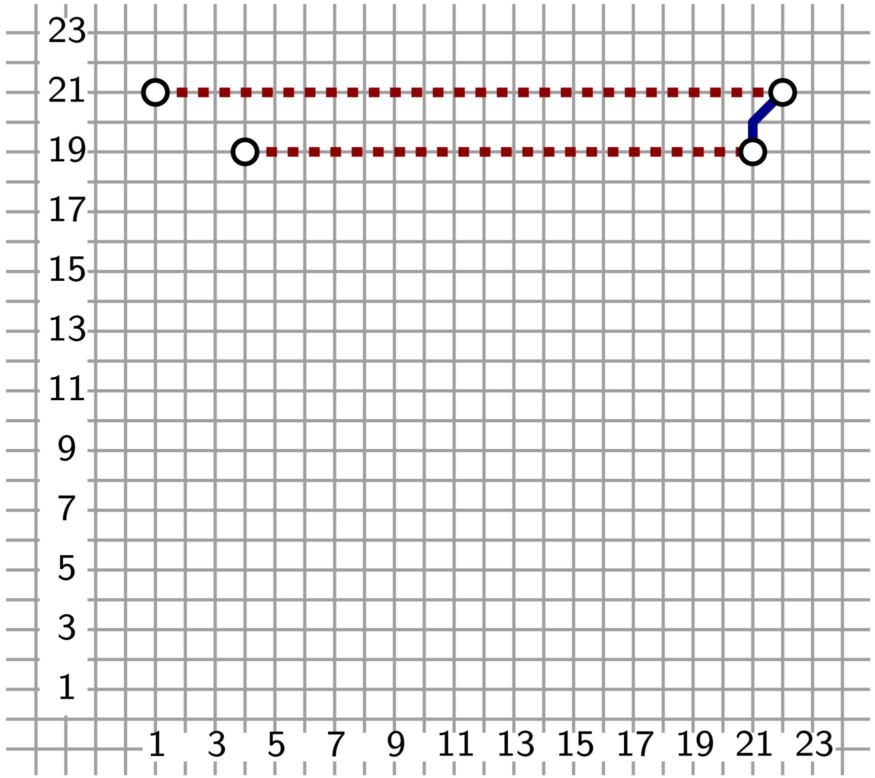
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top



Tree × Matching

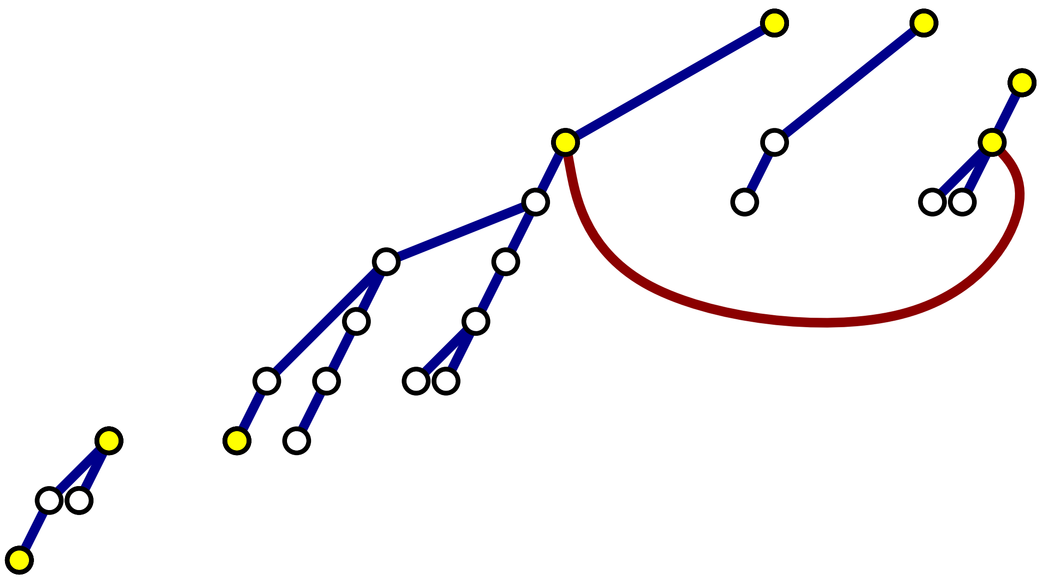
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top

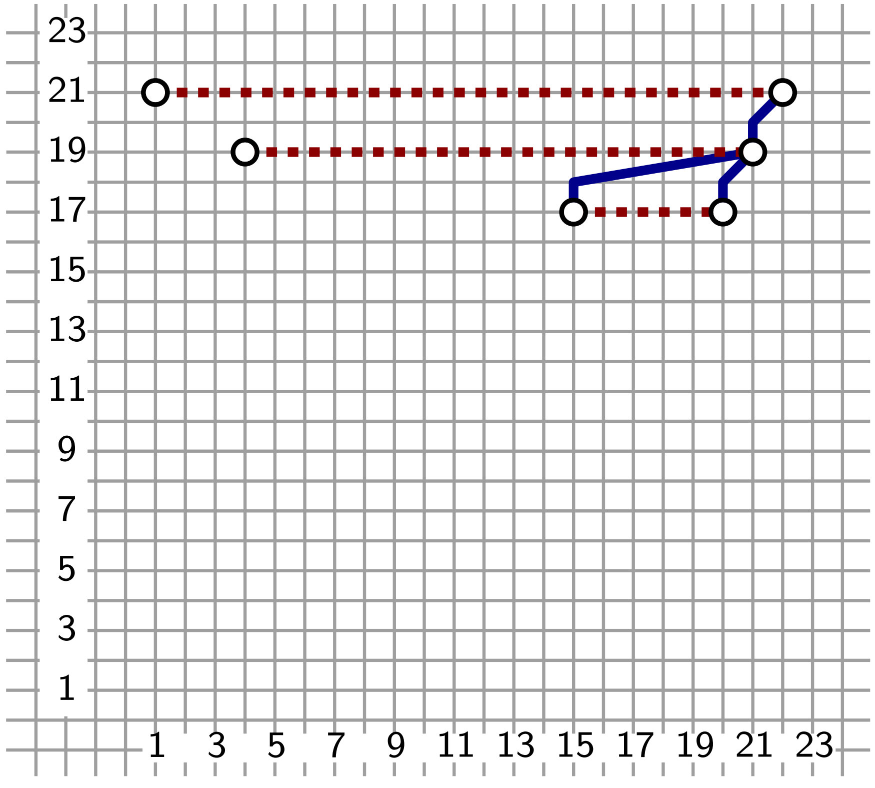




Tree × Matching

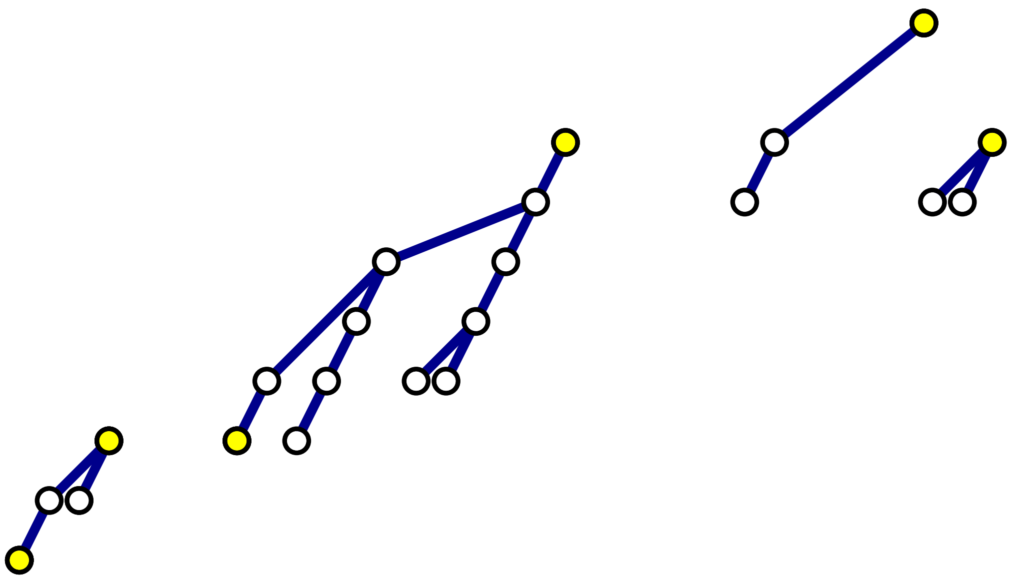
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top

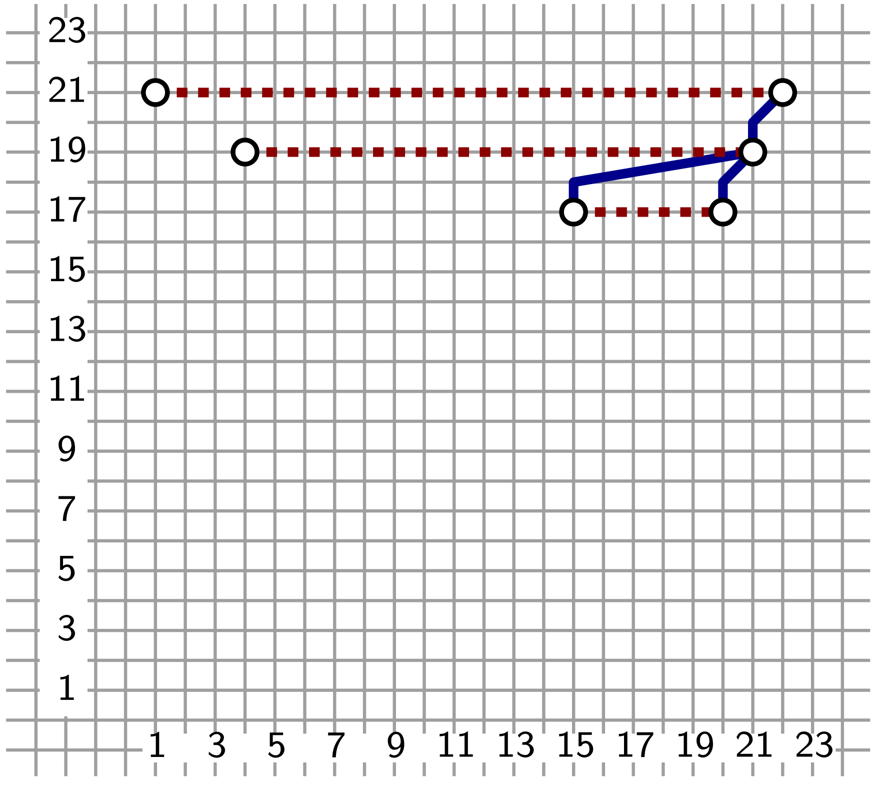




Tree × Matching

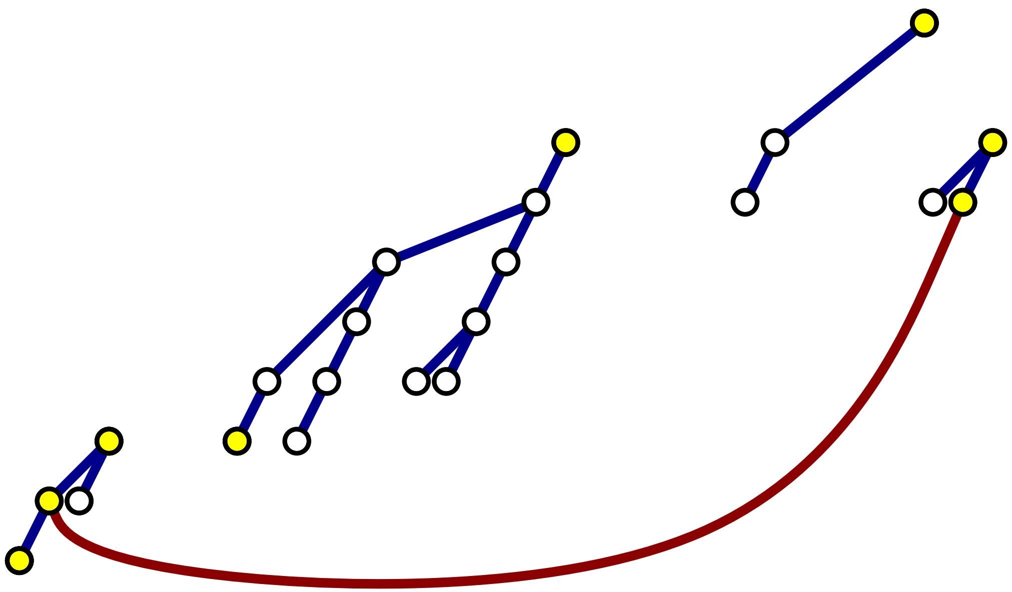
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top

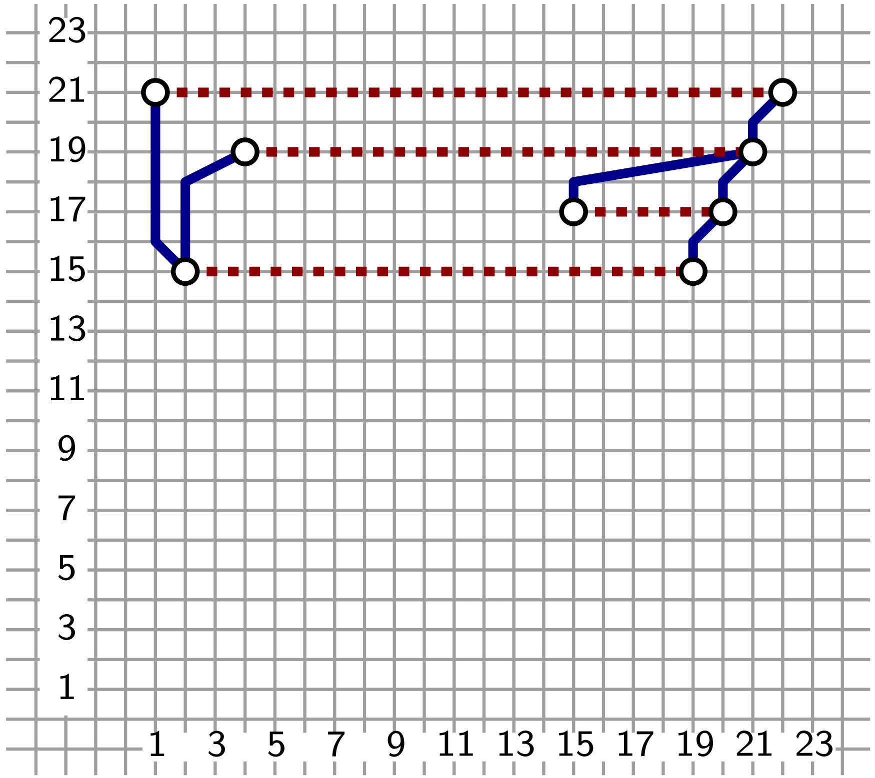




Tree × Matching

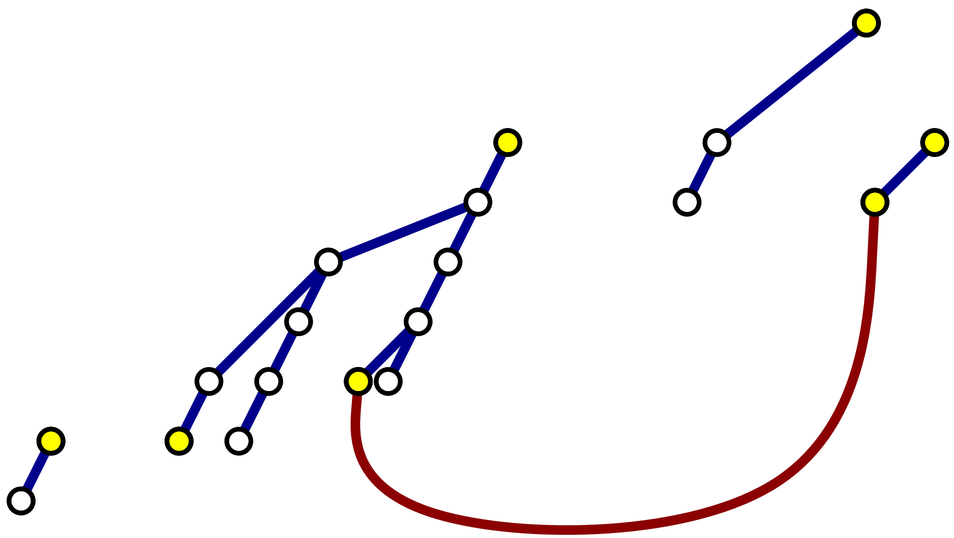
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top

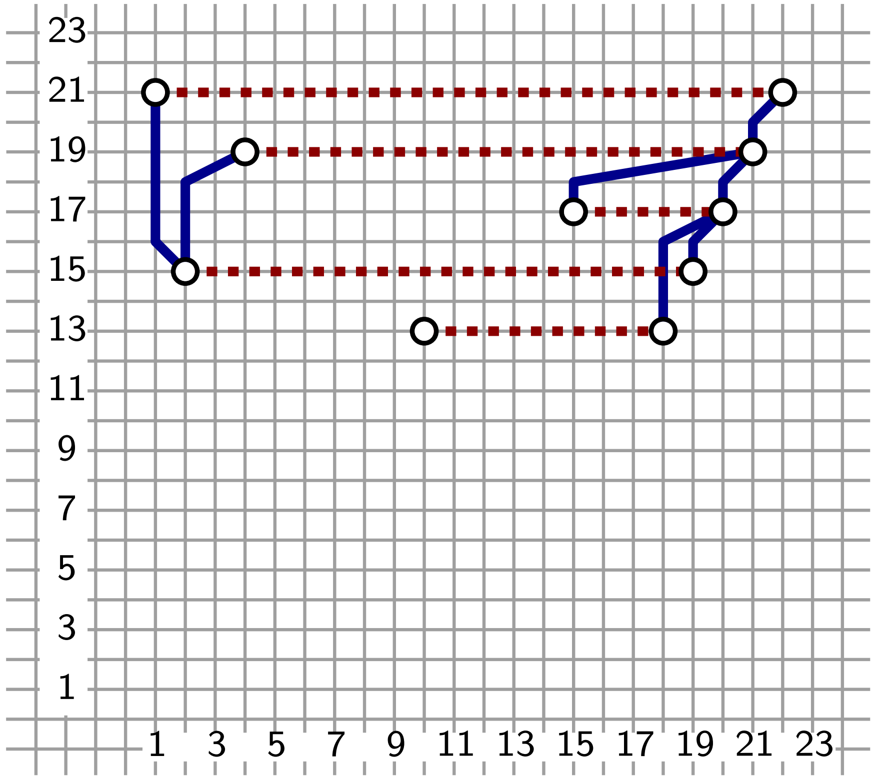




Tree × Matching

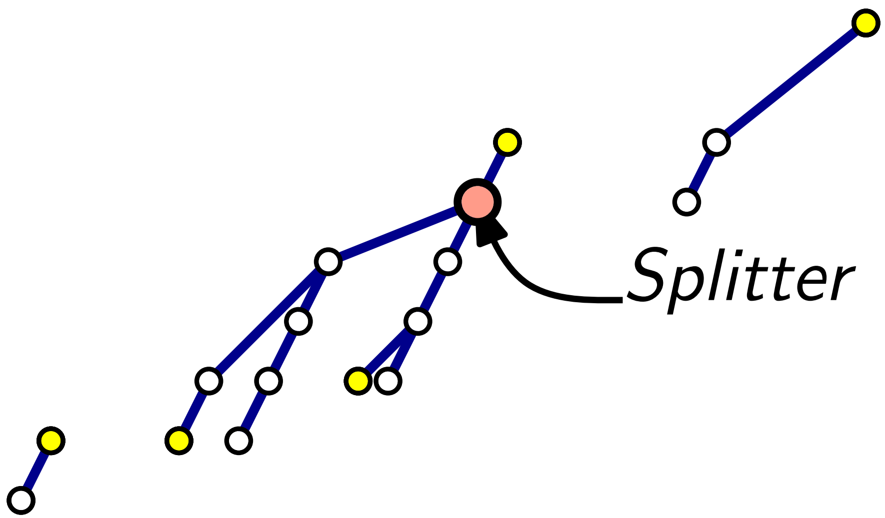
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top

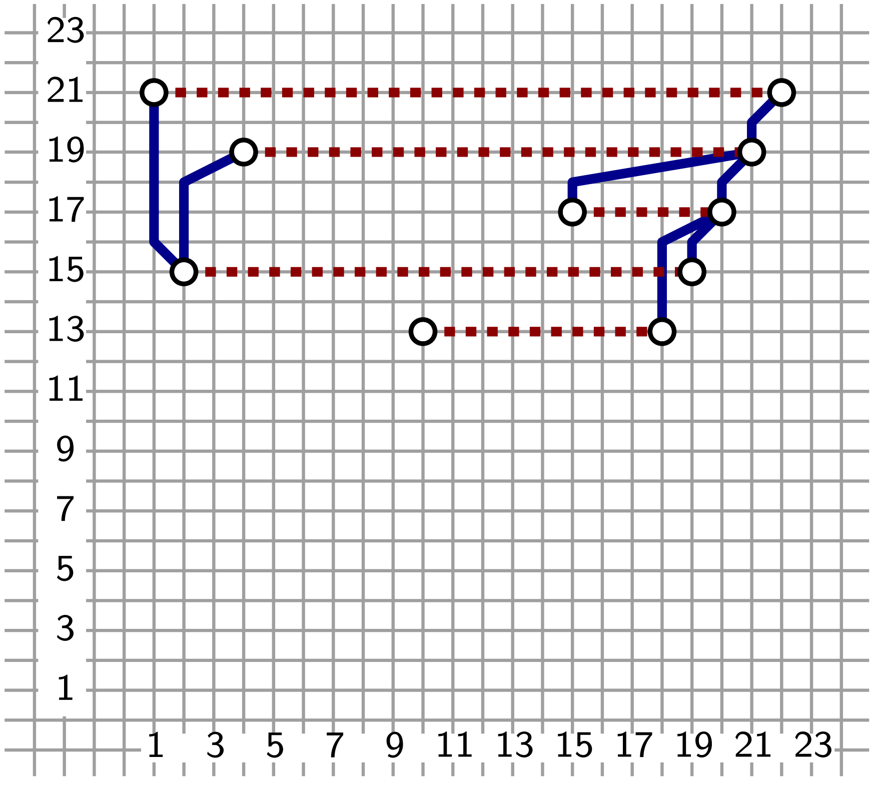




Tree × Matching

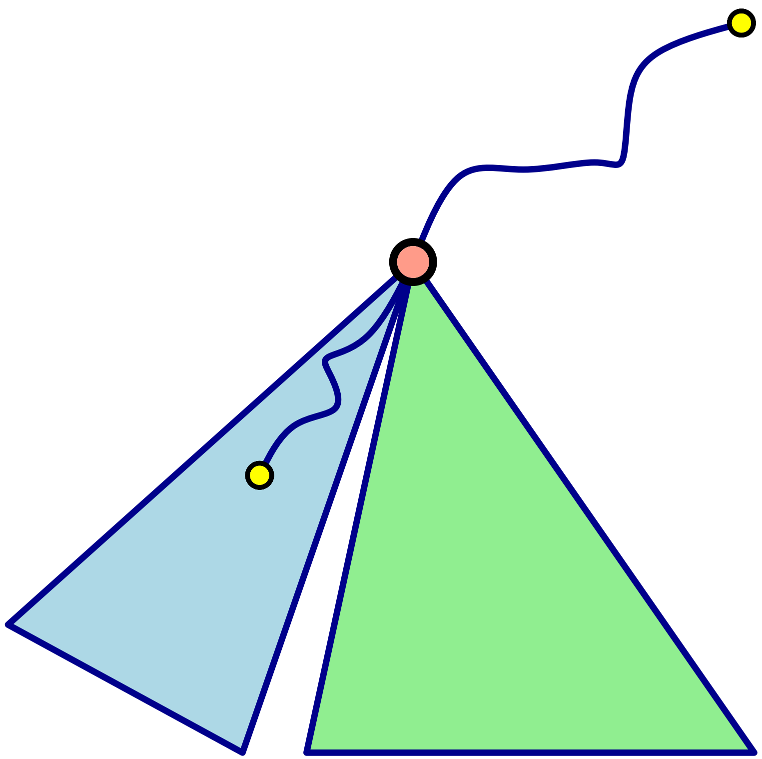
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top





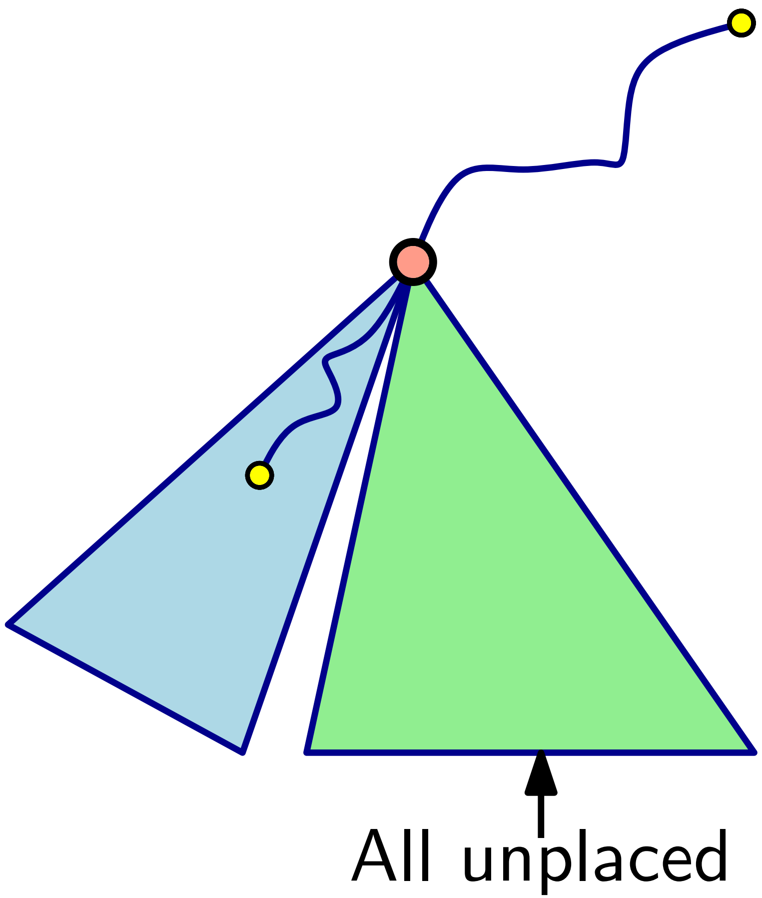
Tree × Matching

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



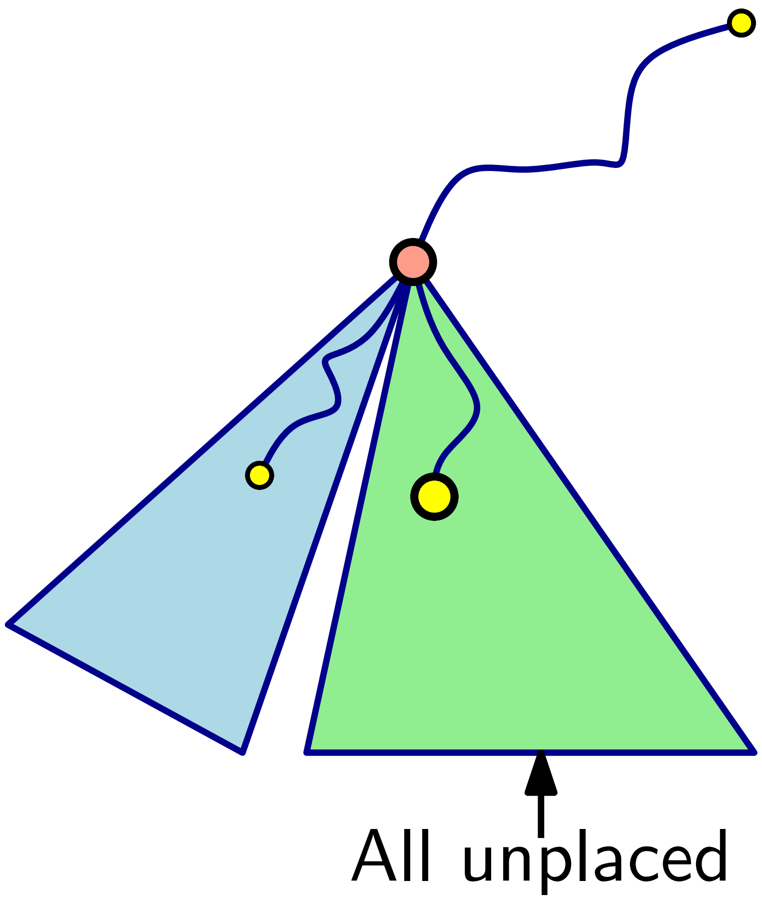
Tree × Matching

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



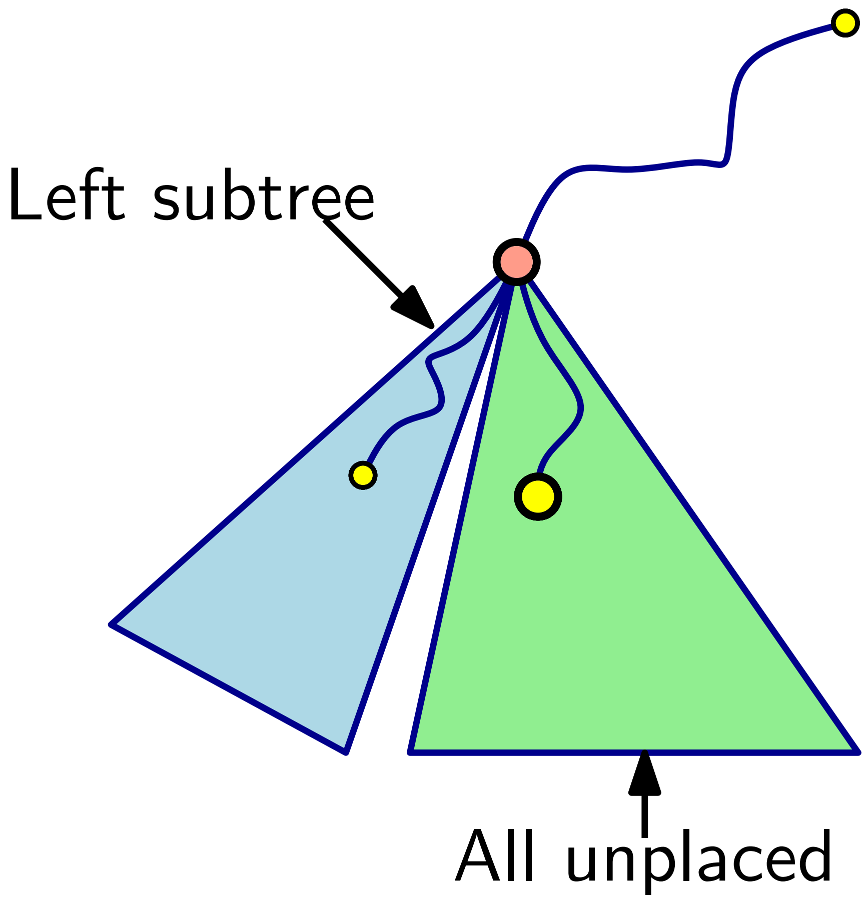
Tree × Matching

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



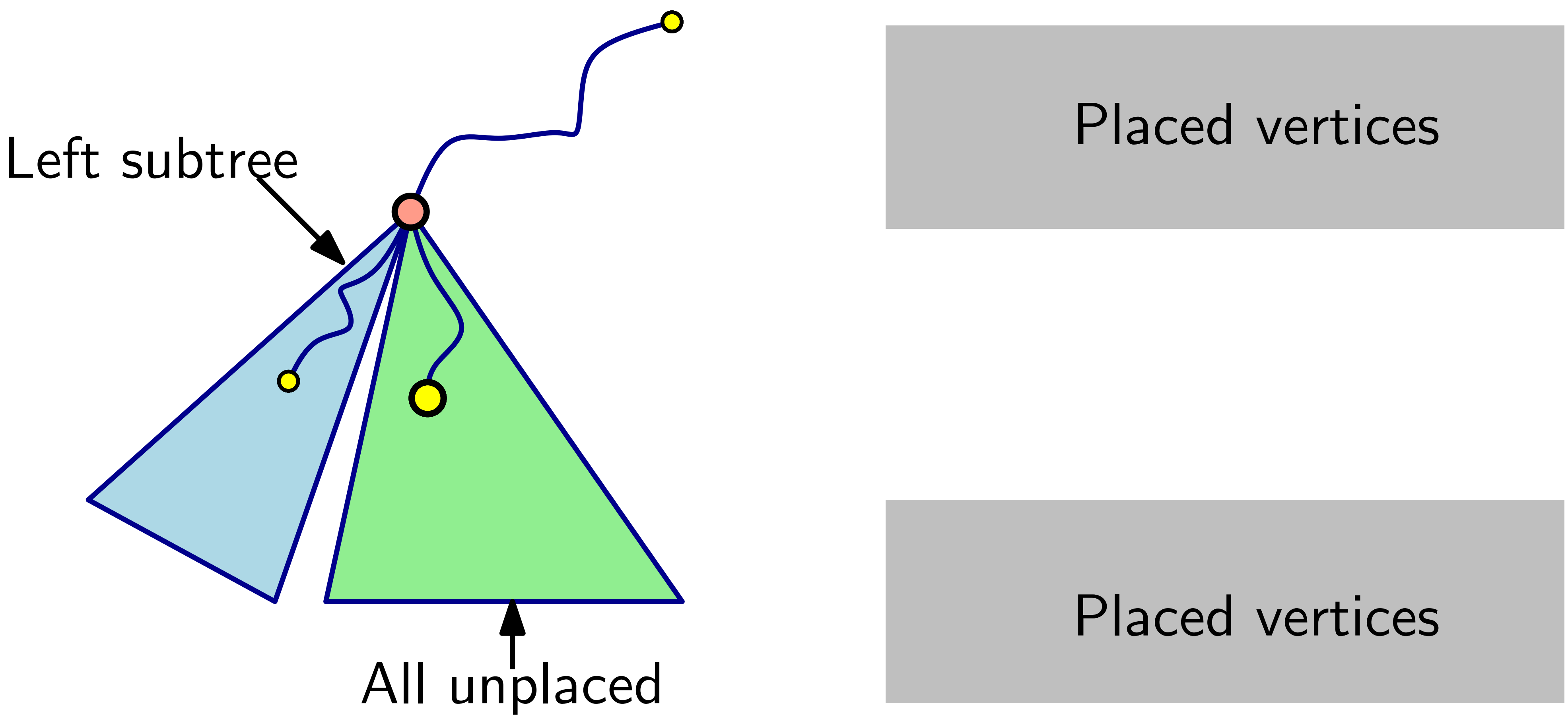
Tree × Matching

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



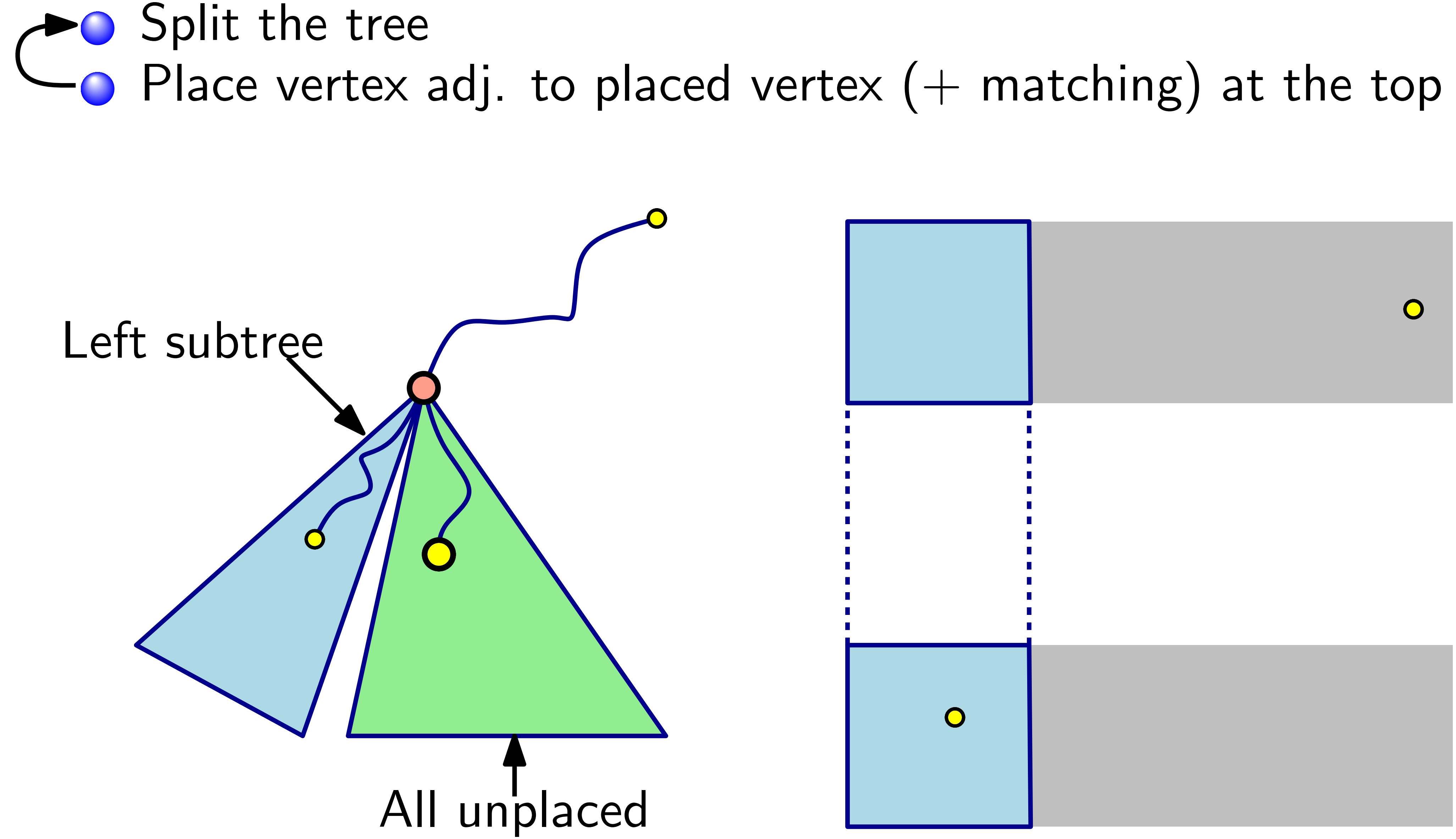
Tree × Matching

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



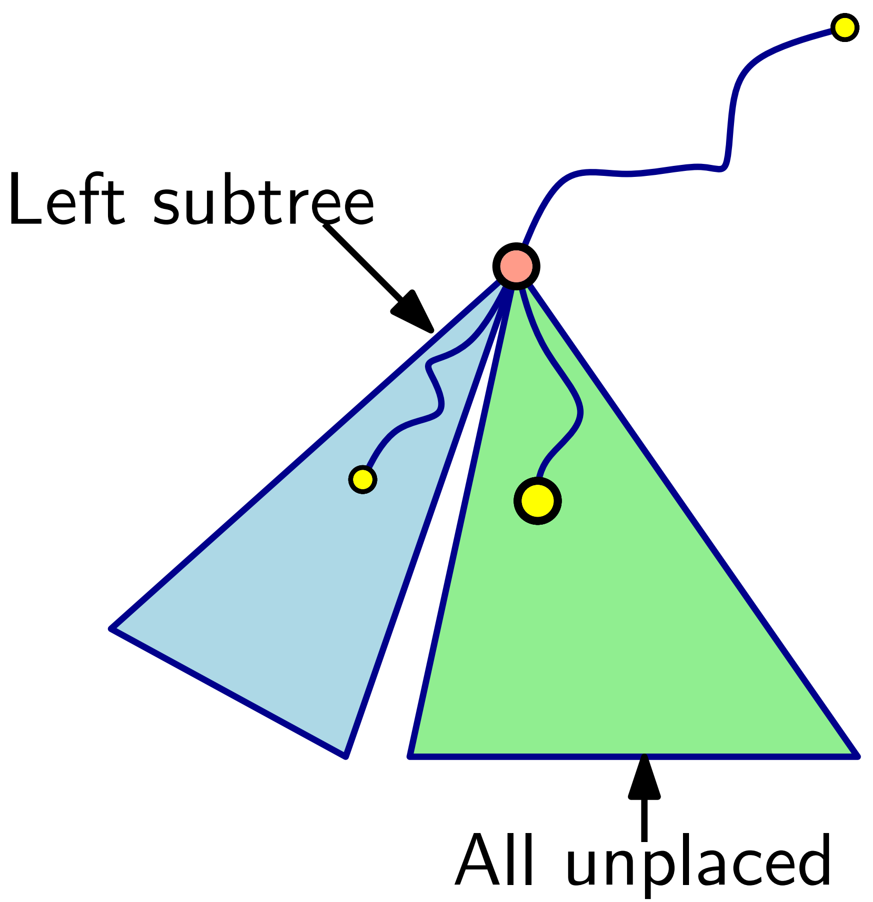
Tree × Matching

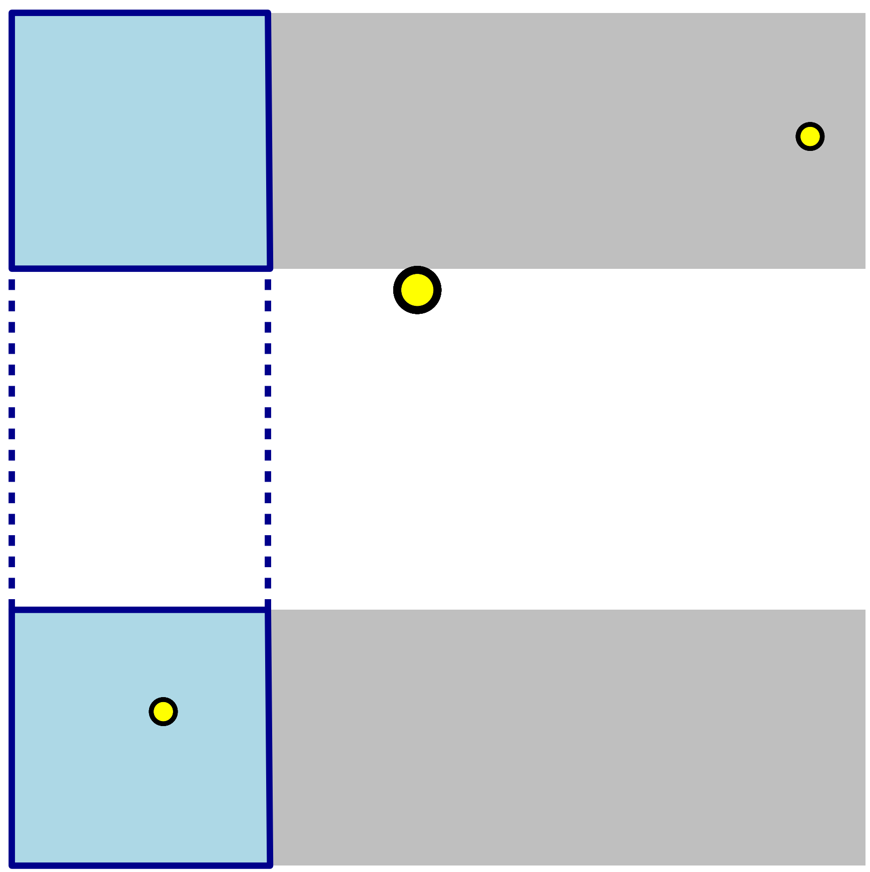
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top



Tree × Matching

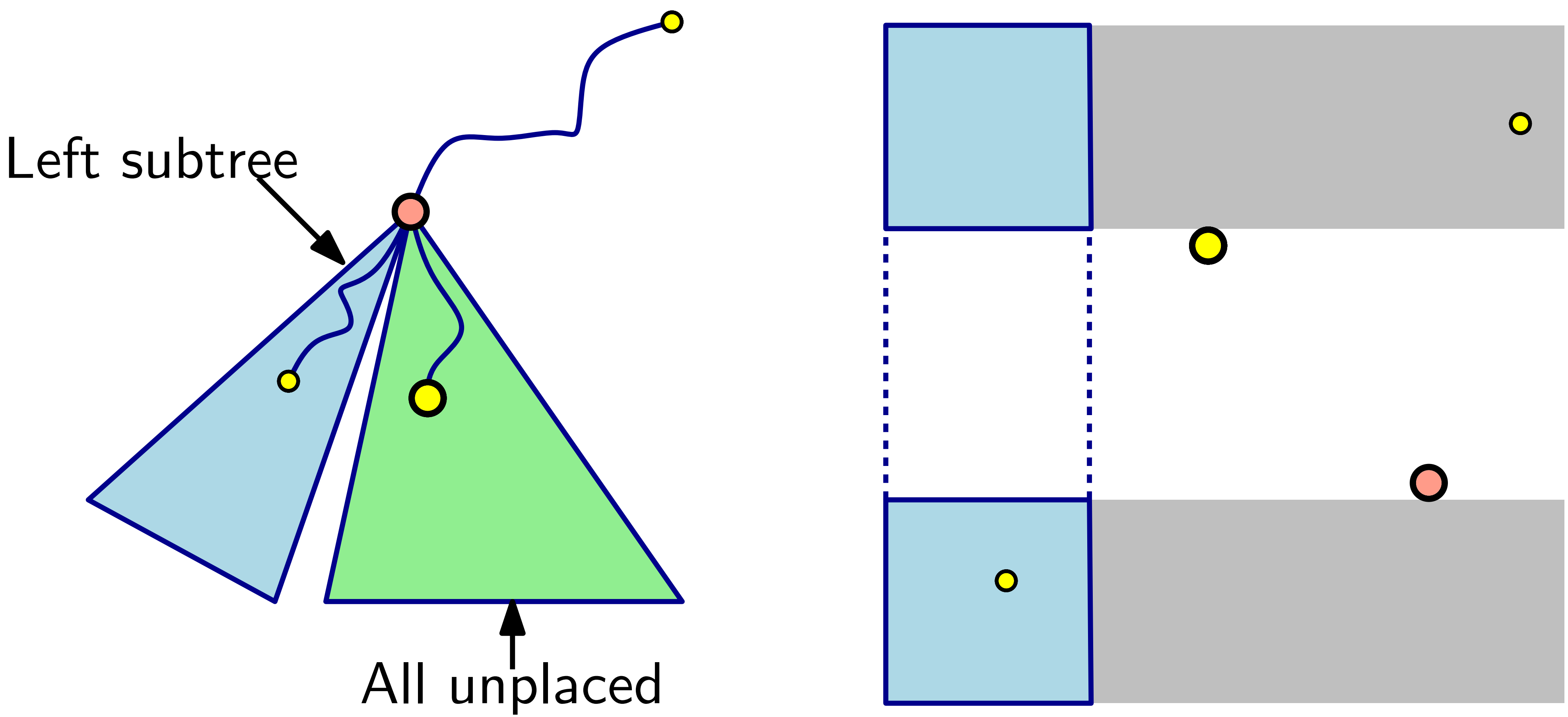
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top





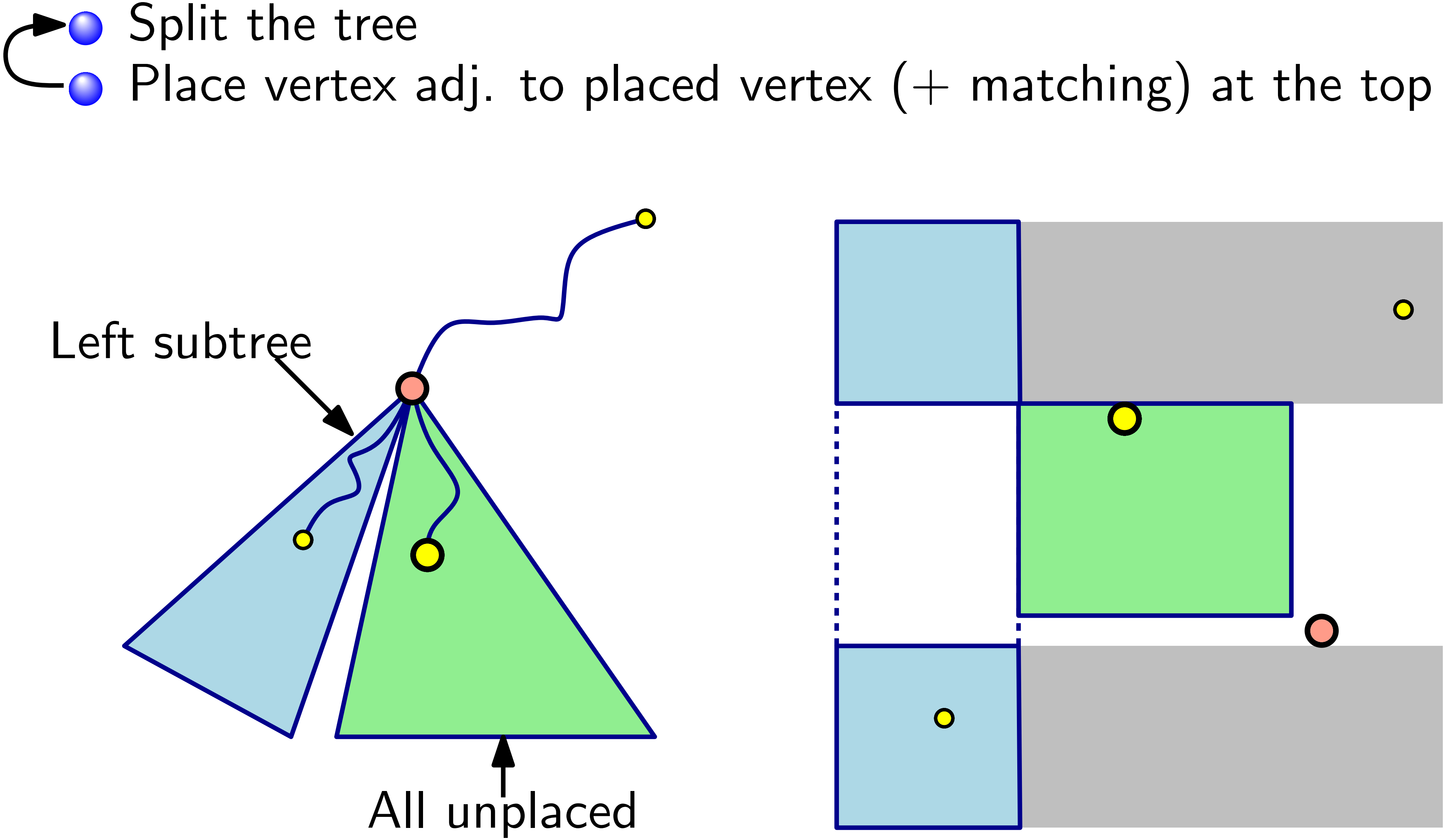
Tree × Matching

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



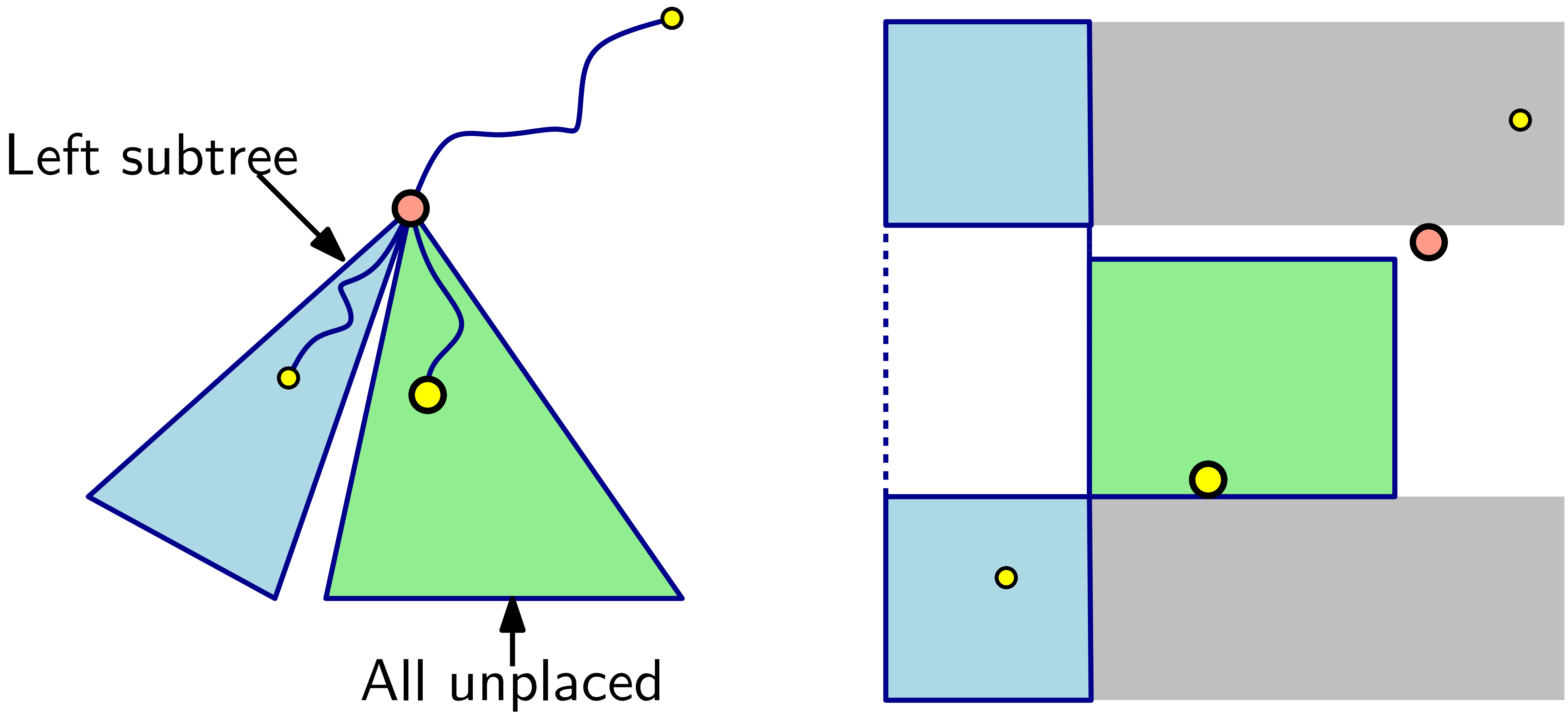
Tree × Matching

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



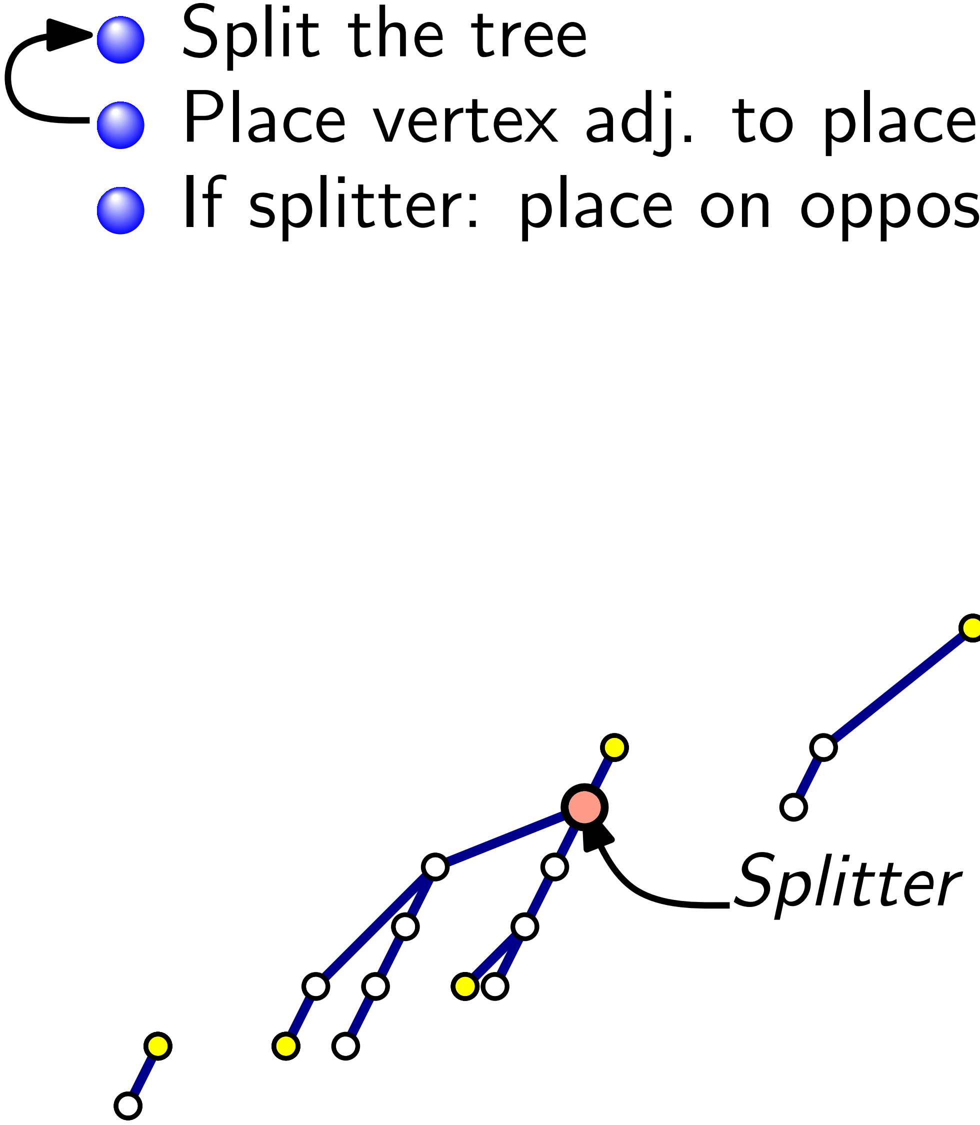
Tree × Matching

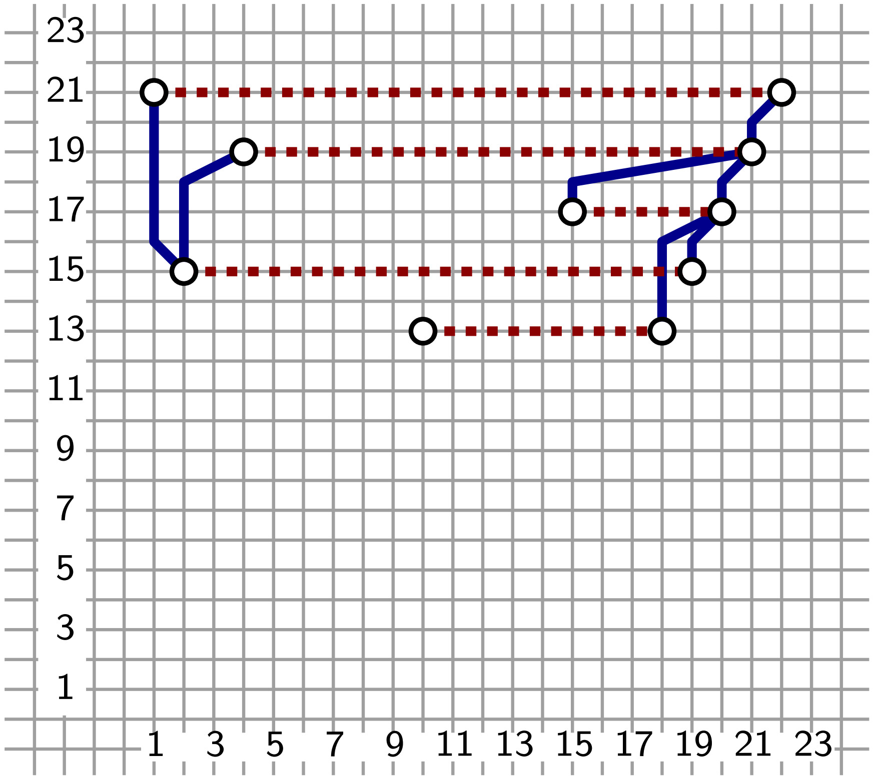
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top



Tree × Matching

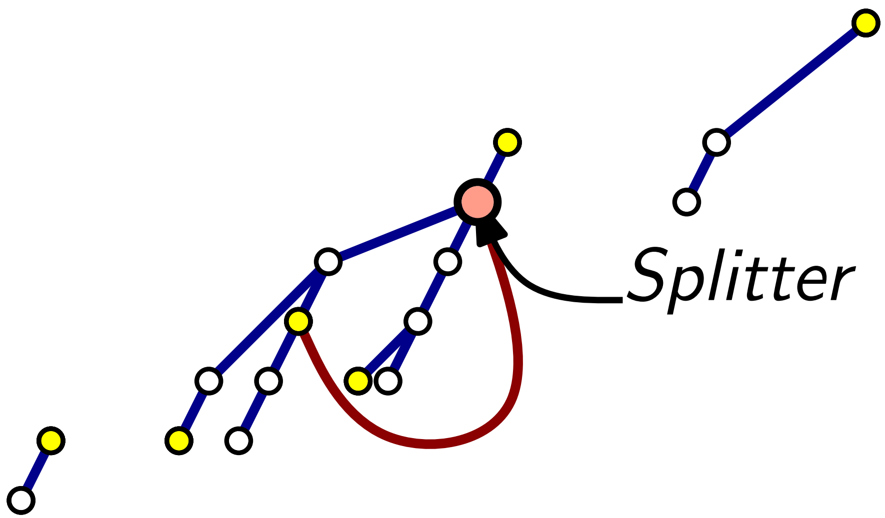
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top
- If splitter: place on opposite side

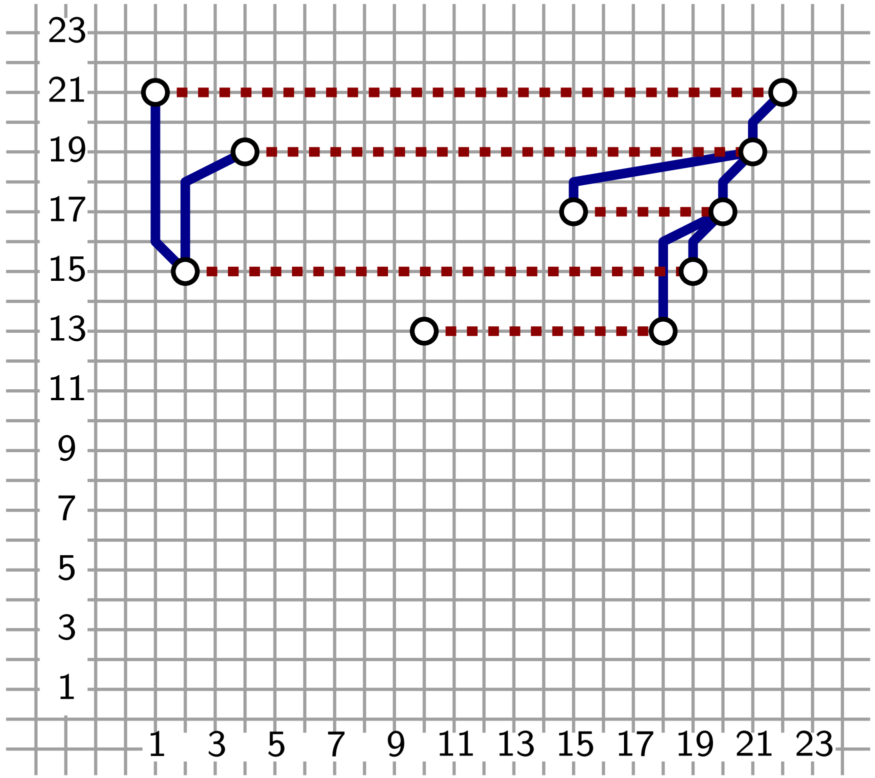




Tree × Matching

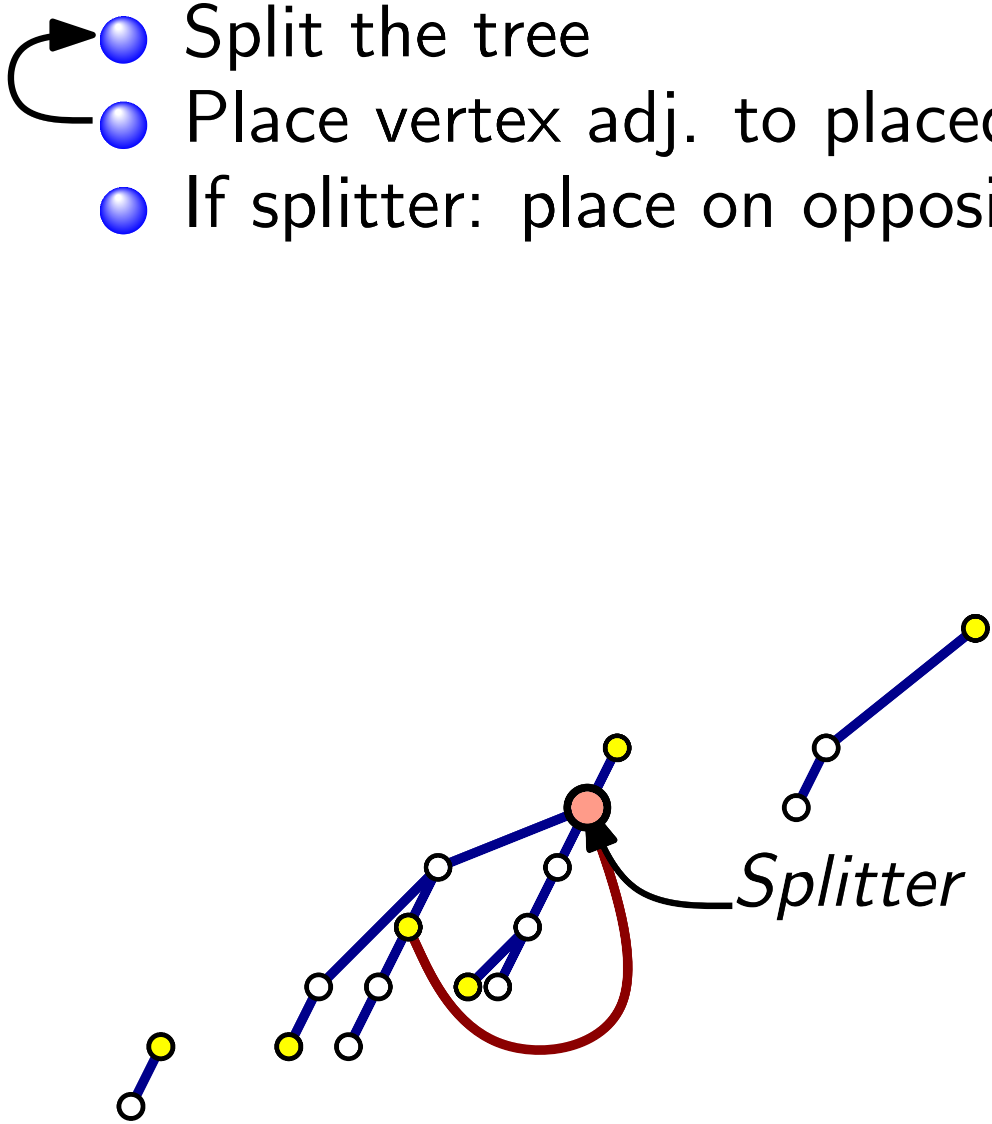
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top
- If splitter: place on opposite side

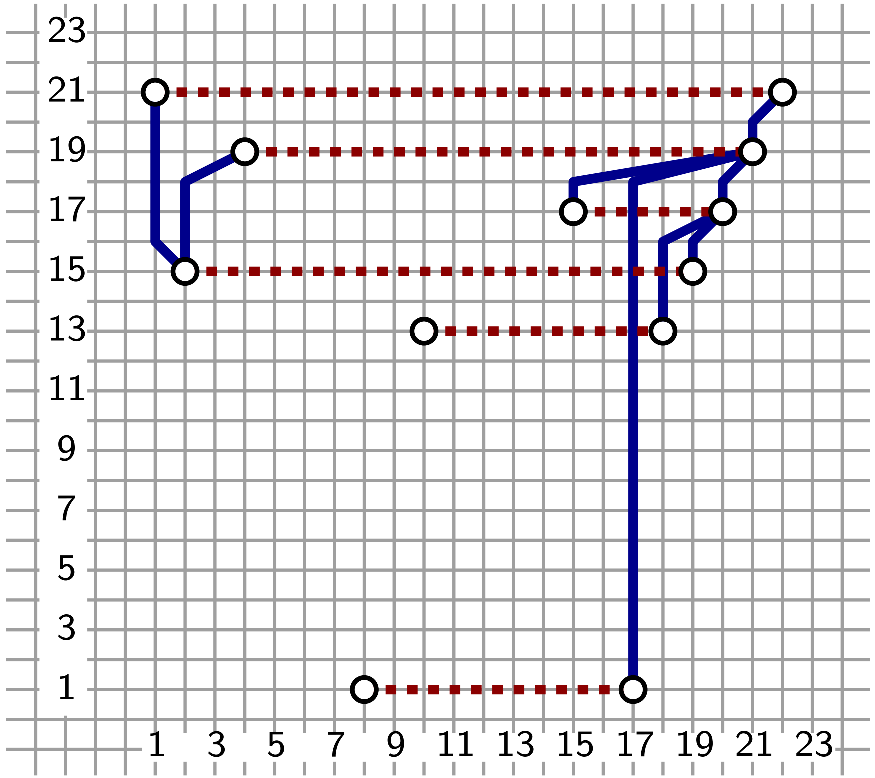




Tree × Matching

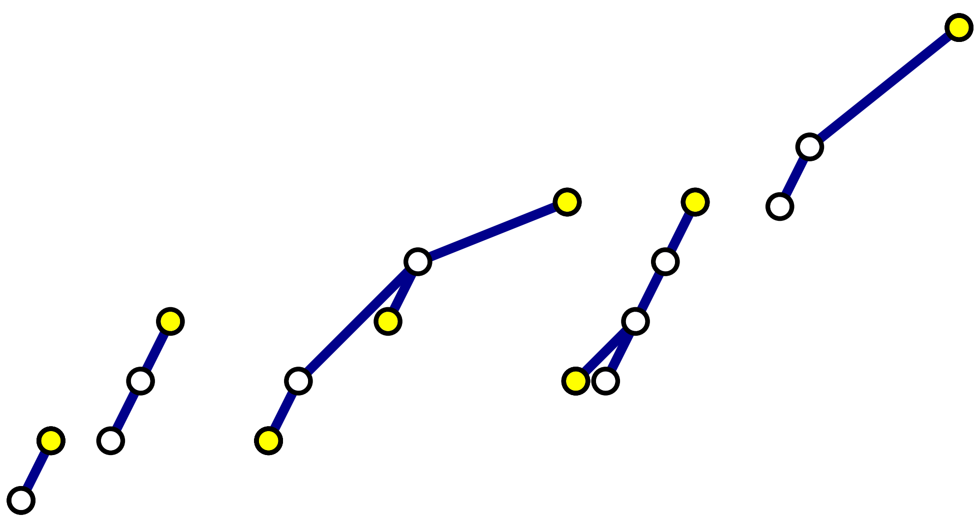
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top
- If splitter: place on opposite side

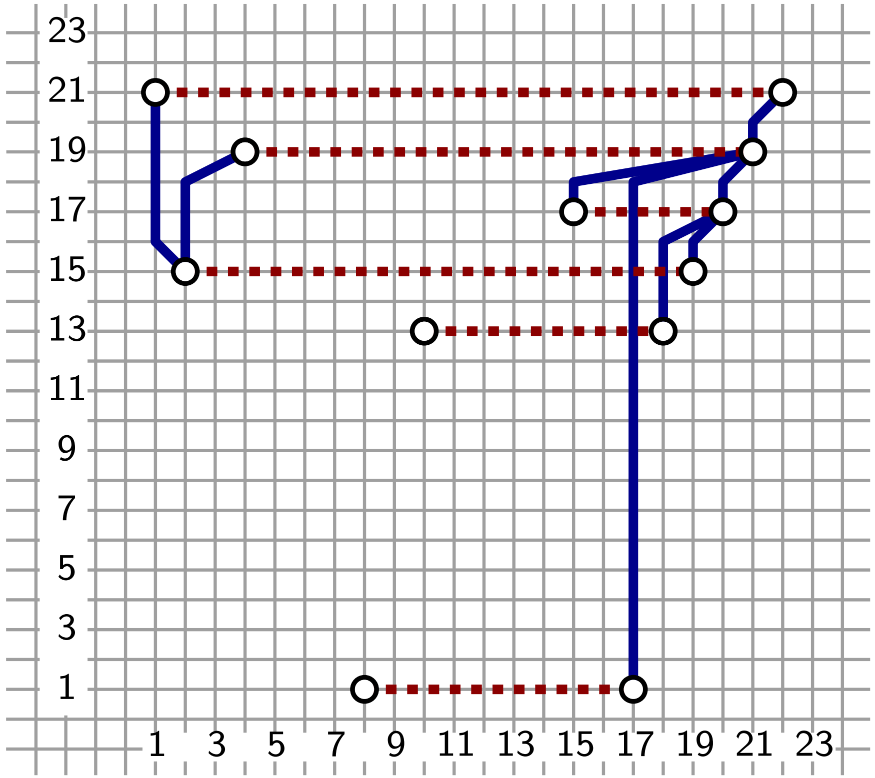




Tree × Matching

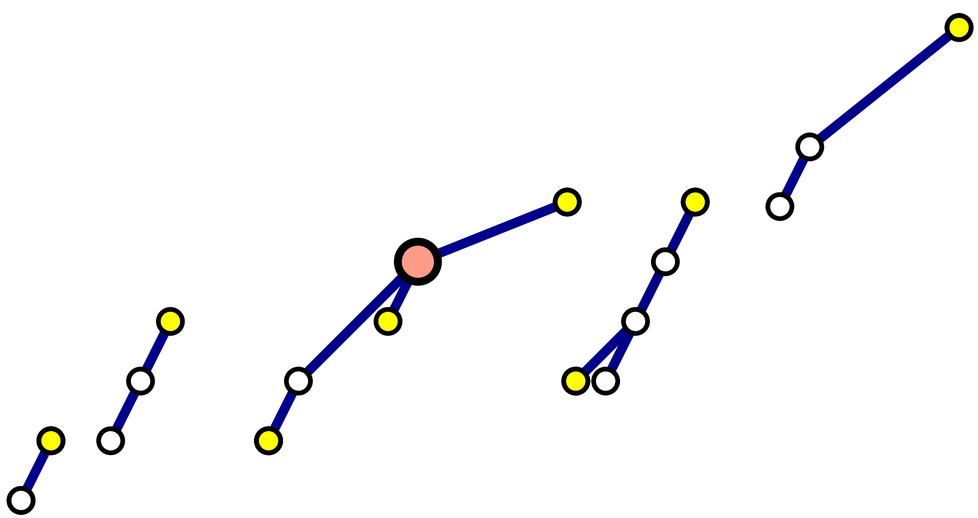
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top
- If splitter: place on opposite side

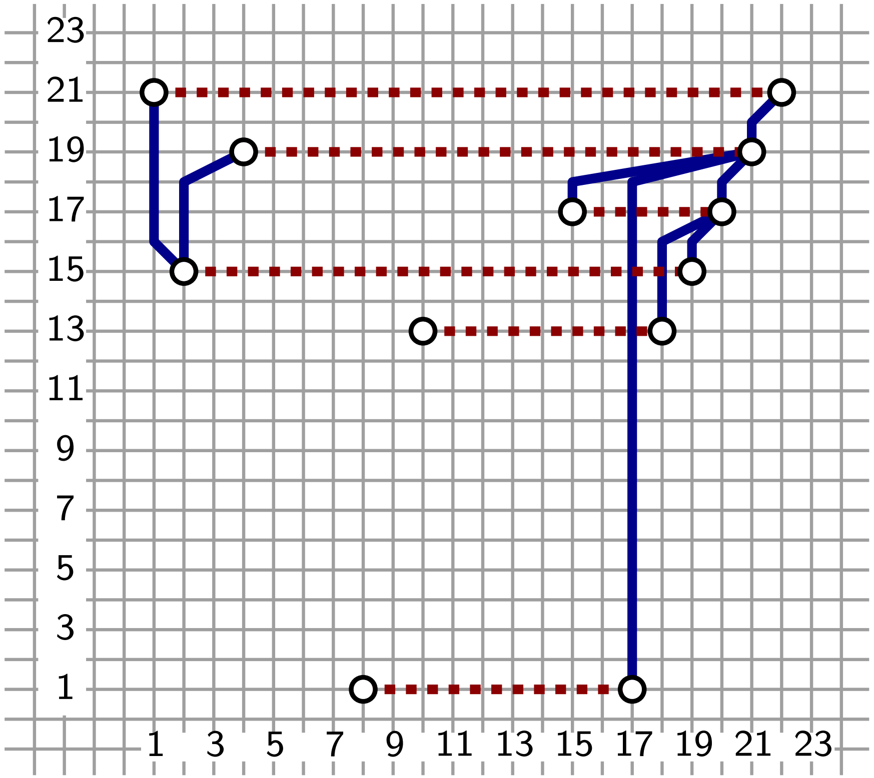




Tree × Matching

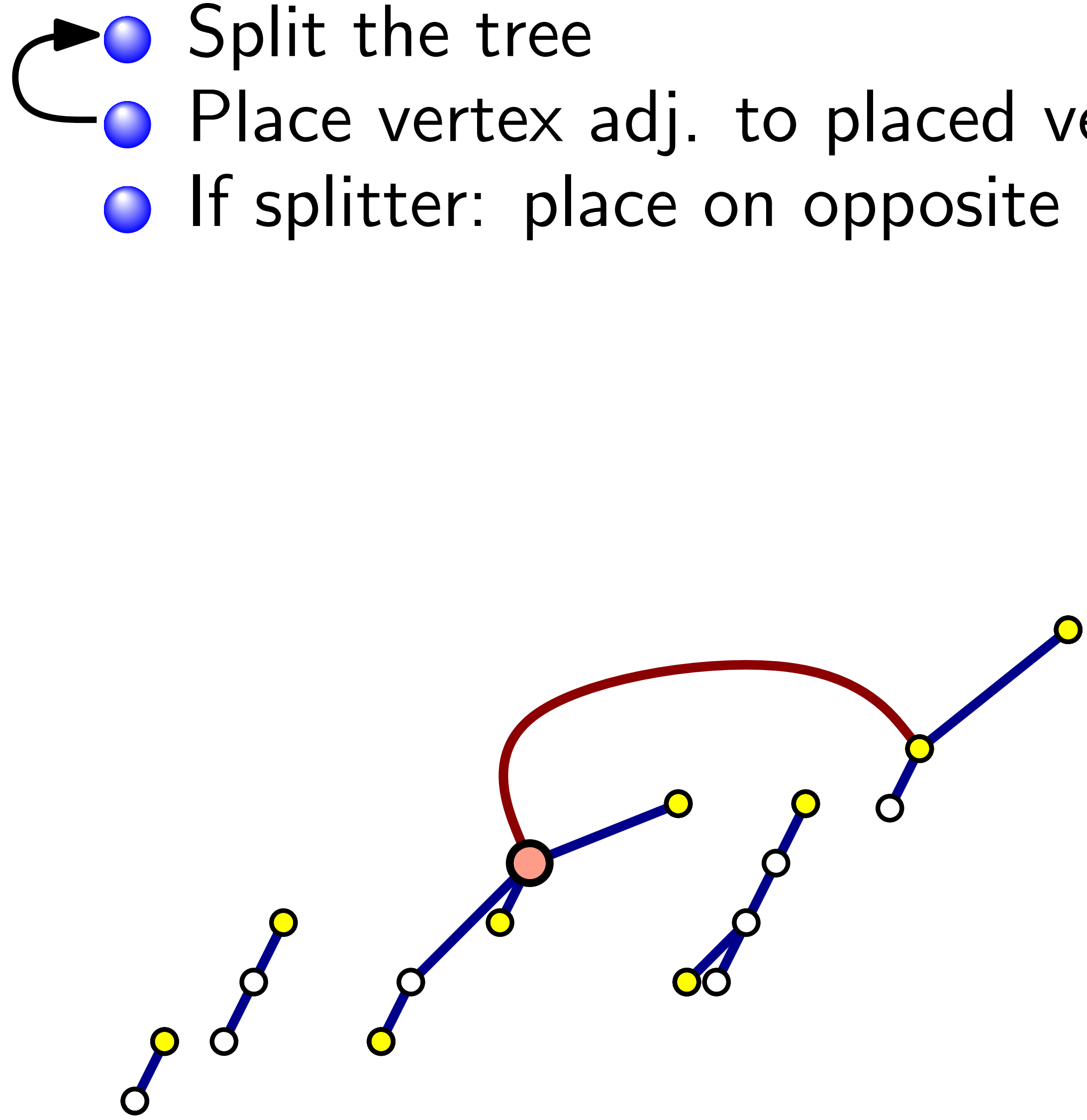
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top
- If splitter: place on opposite side

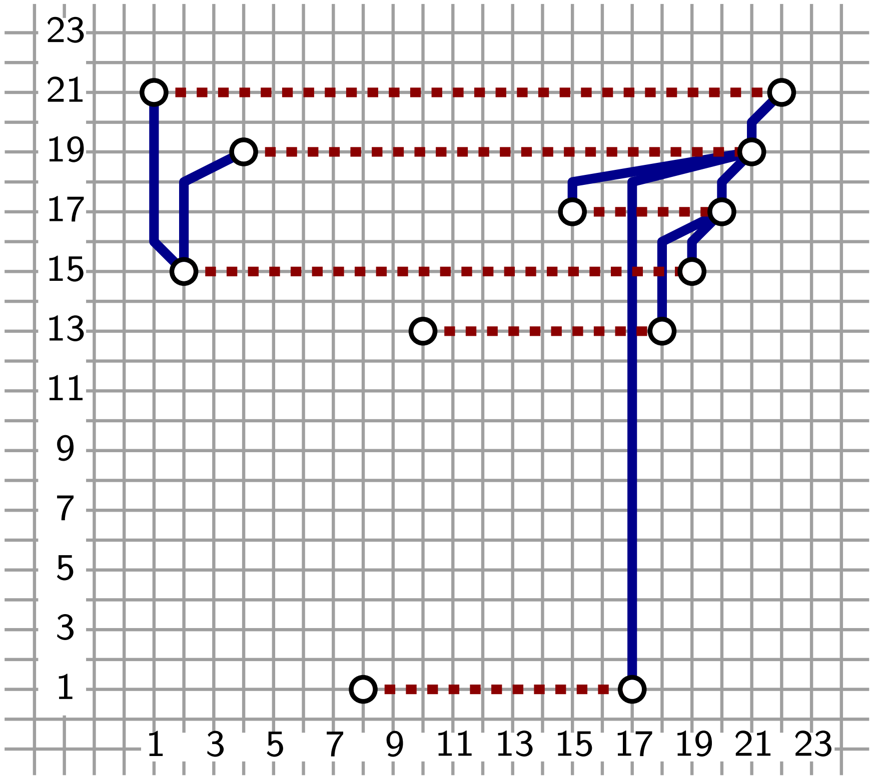




Tree × Matching

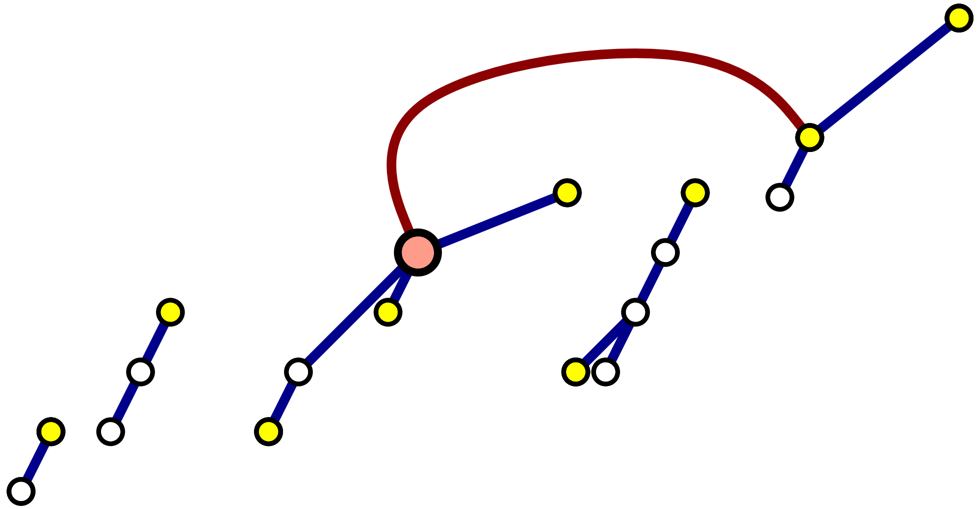
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top
- If splitter: place on opposite side

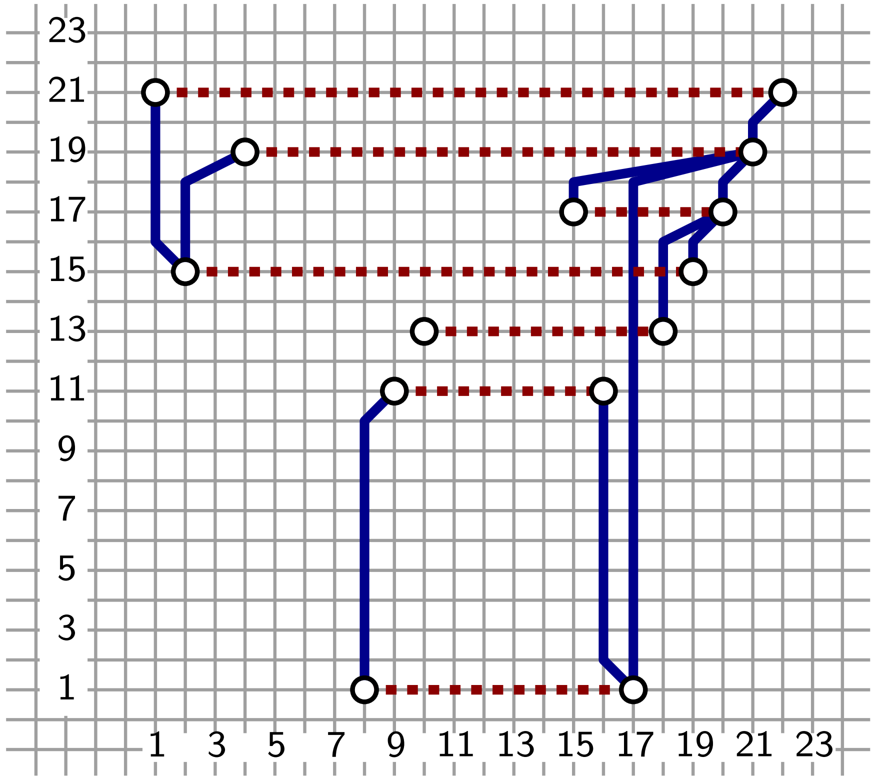




Tree × Matching

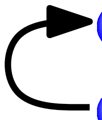
- Place root + matching at the top
- Split the tree
- Place vertex adj. to placed vertex (+ matching) at the top
- If splitter: place on opposite side

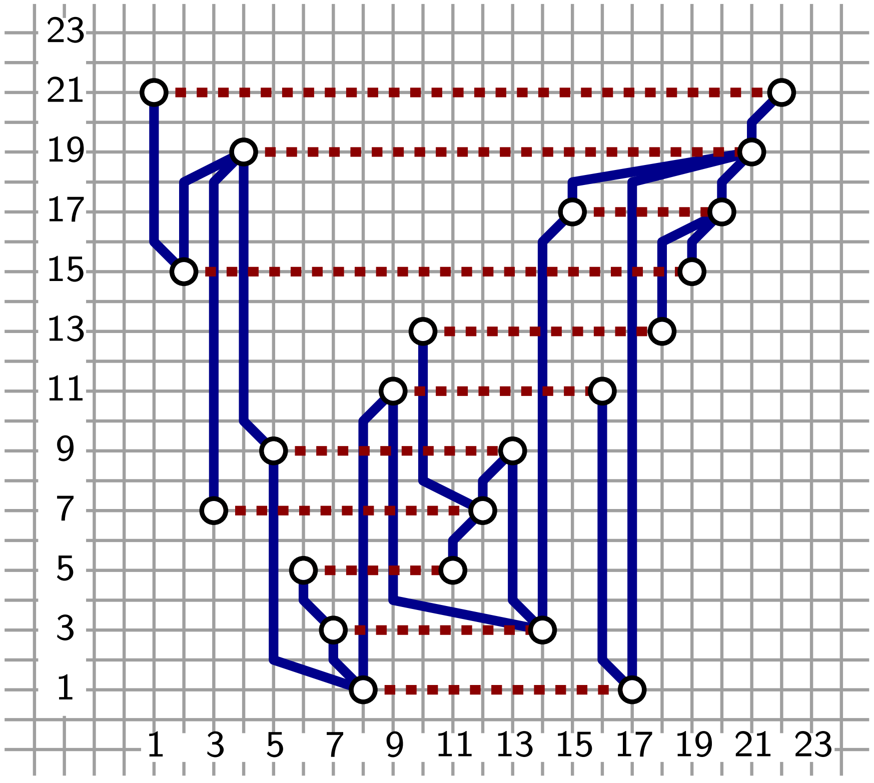




Tree × Matching

- 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

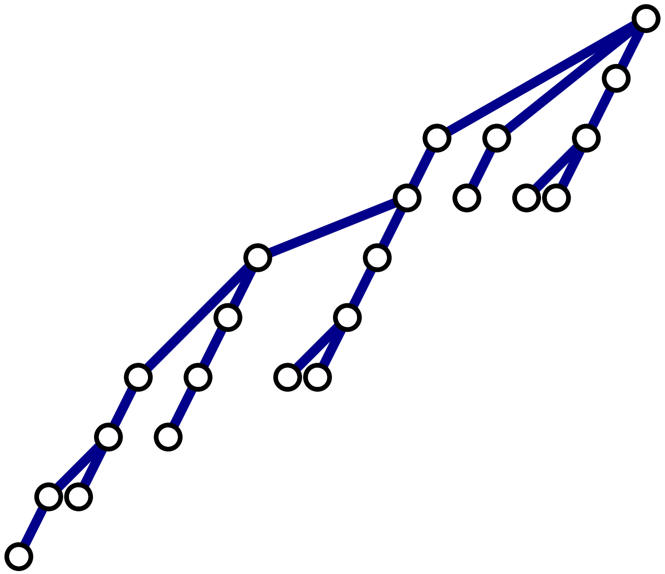


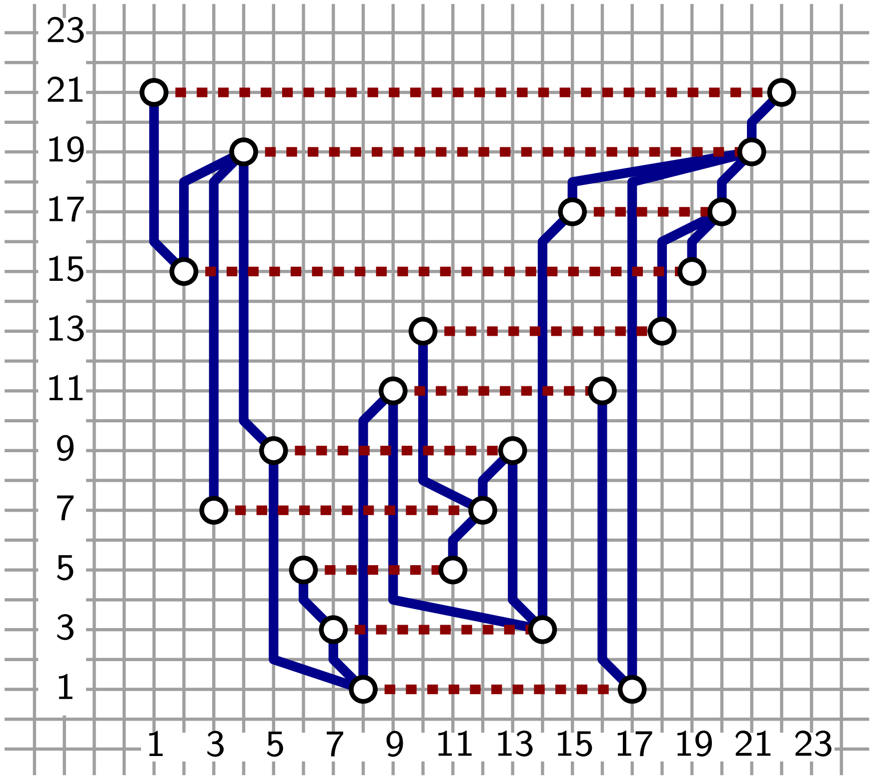


Tree × Matching

- 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:	$n \times (n-1)$

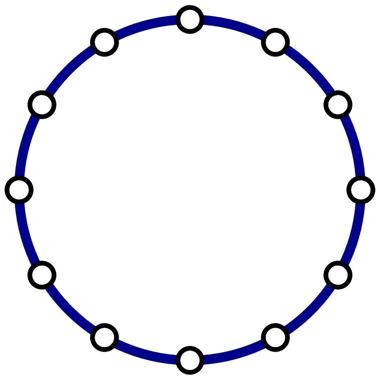




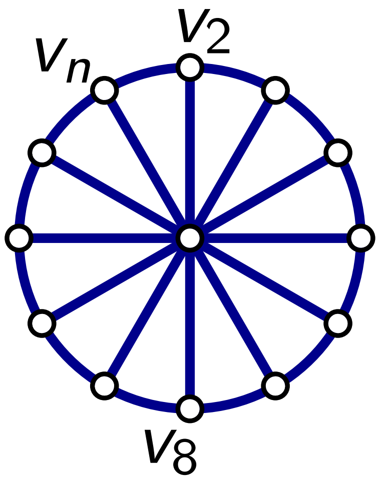
Overview

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

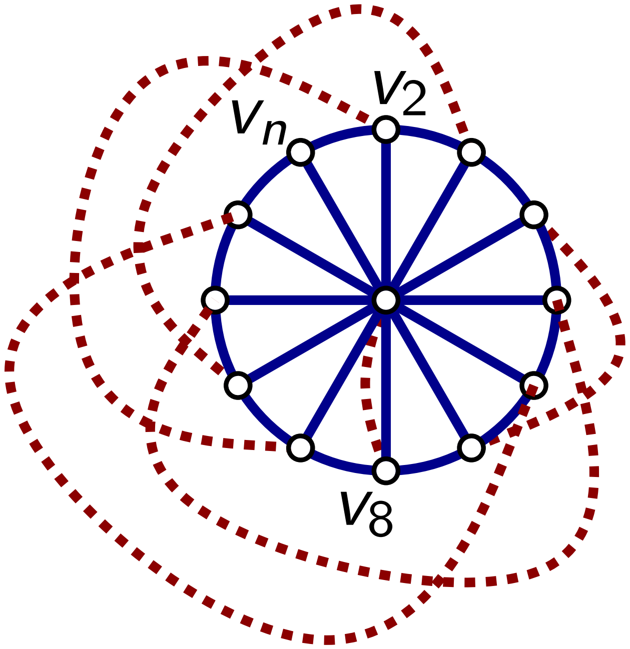
Wheel $\times$ Matching



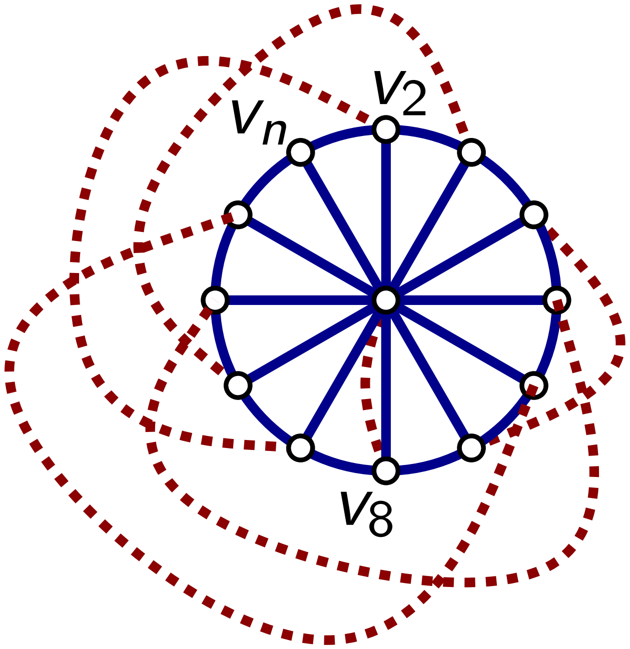
Wheel × Matching

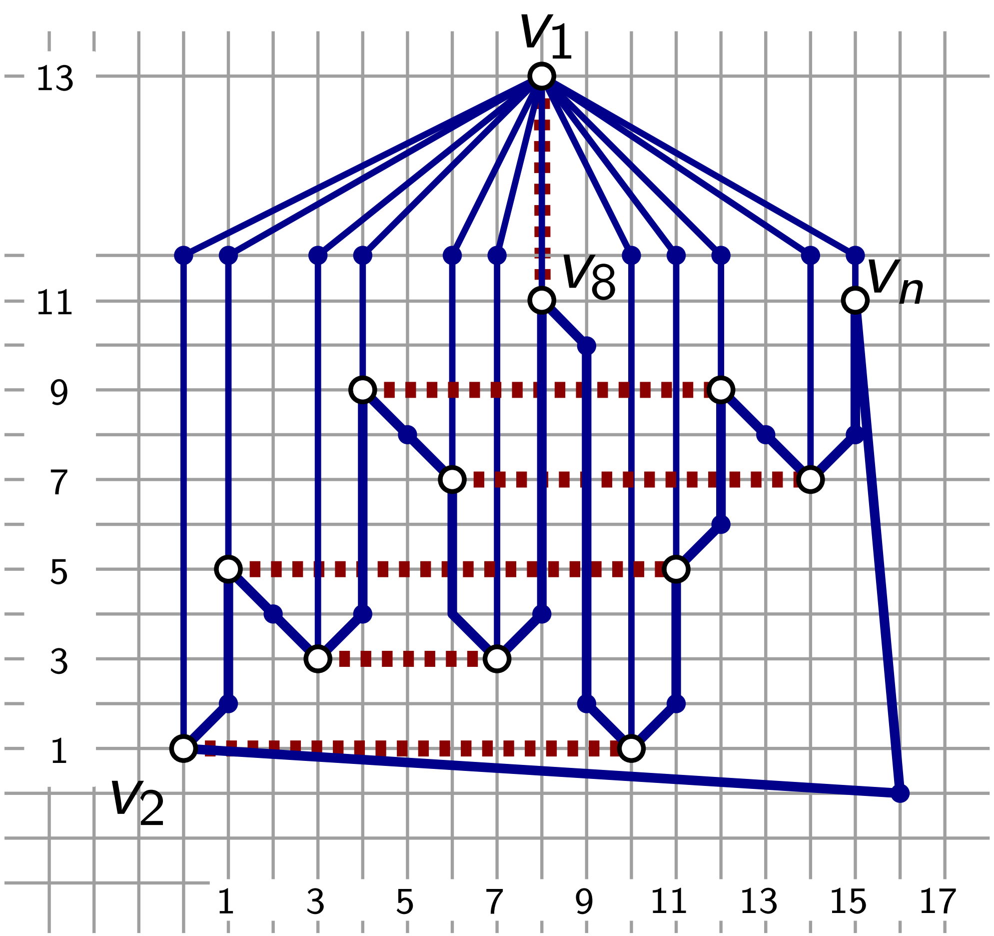


Wheel × Matching

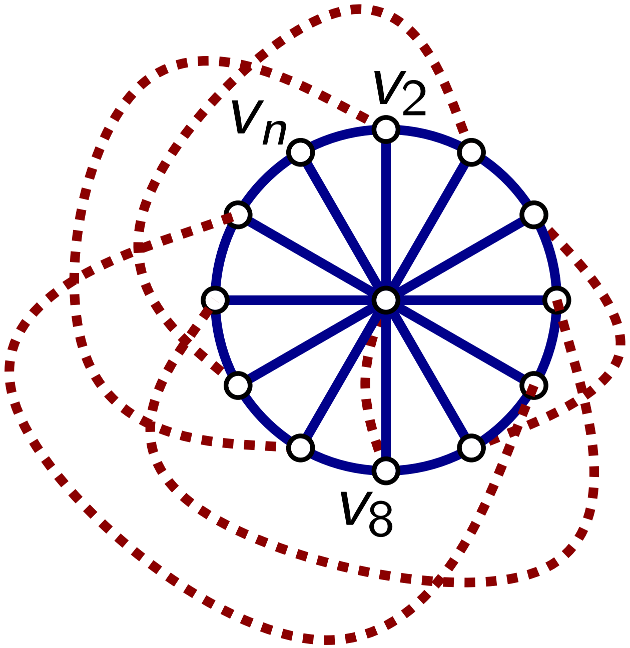


Wheel × Matching

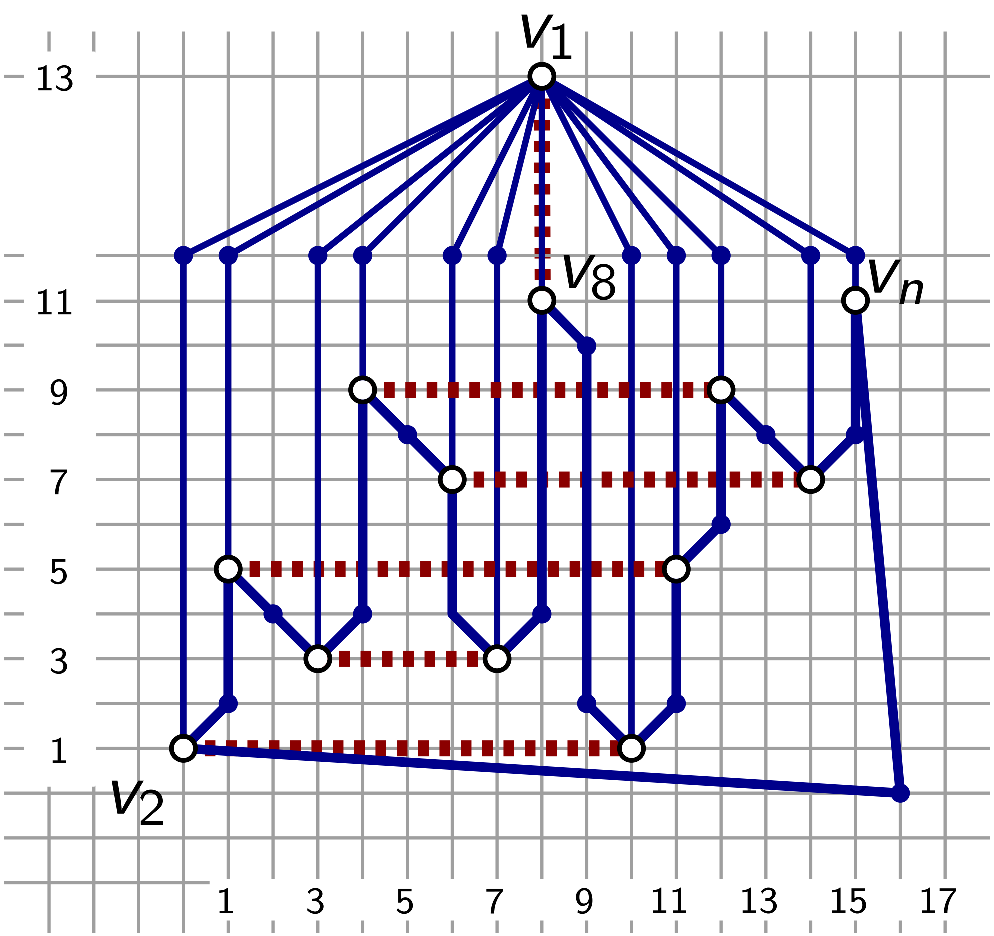




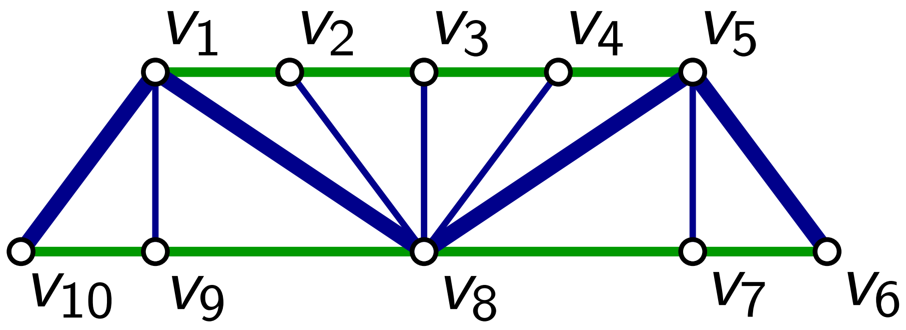
Wheel × Matching



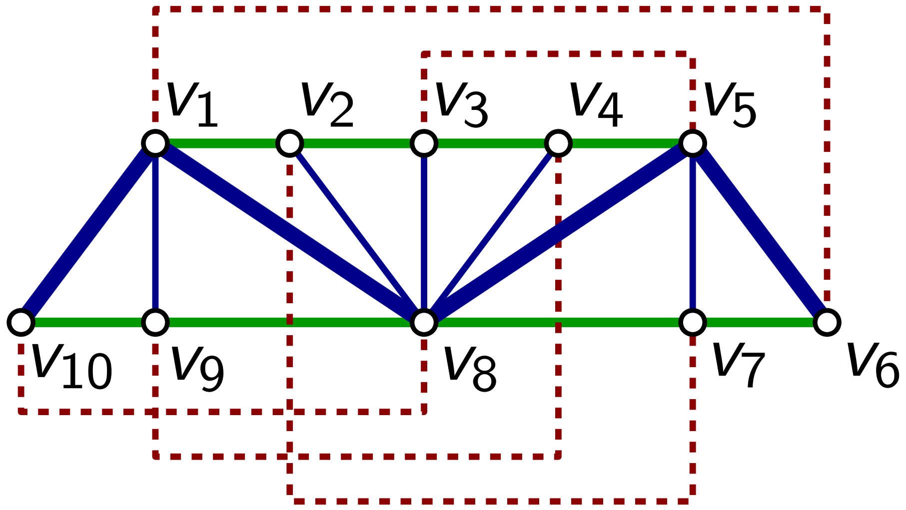
Bends: 2×0
Grid size: $(1.5n - 1) \times (n + 2)$



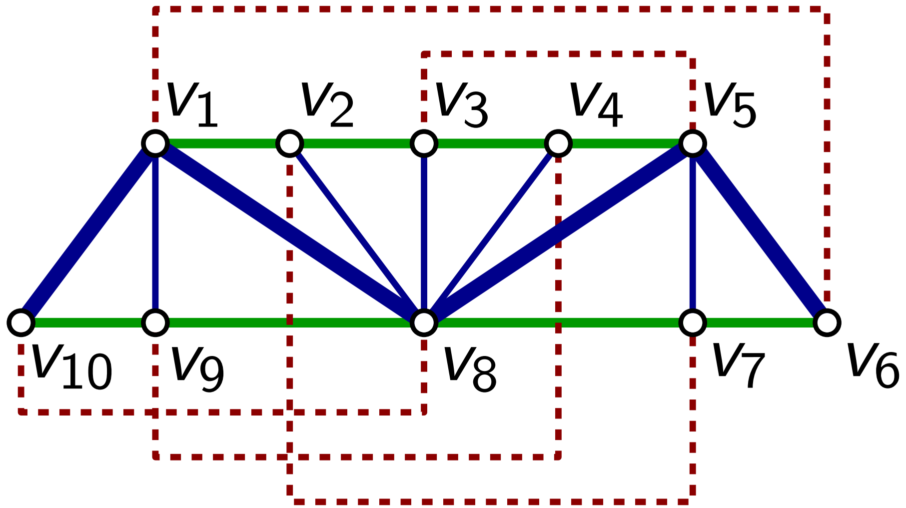
Outerpath × Matching

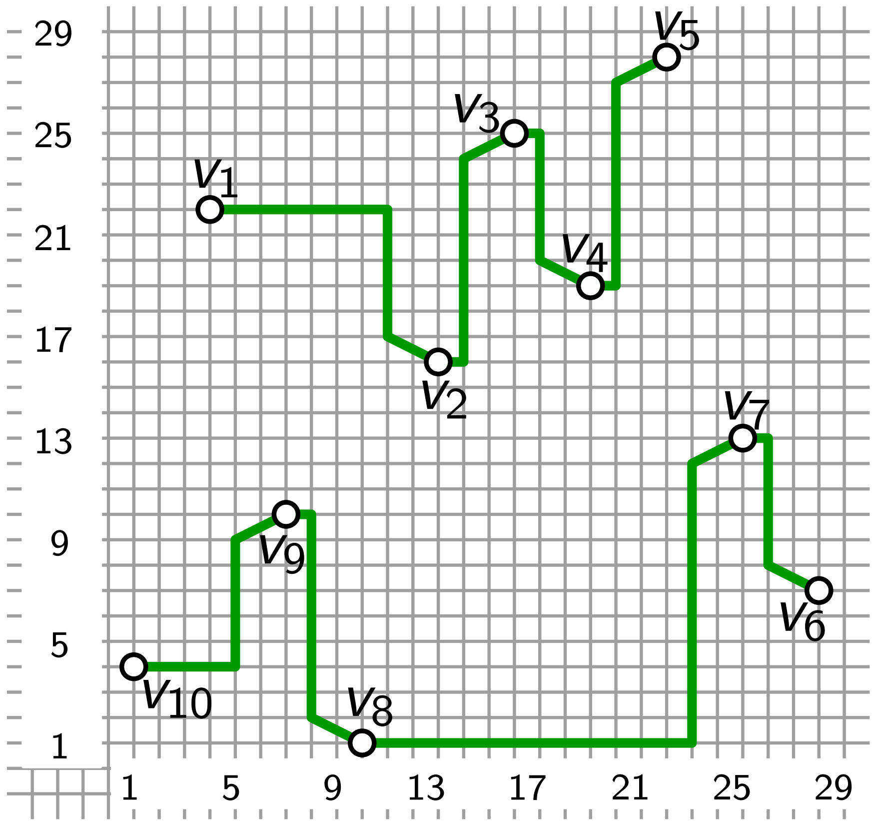


Outerpath × Matching

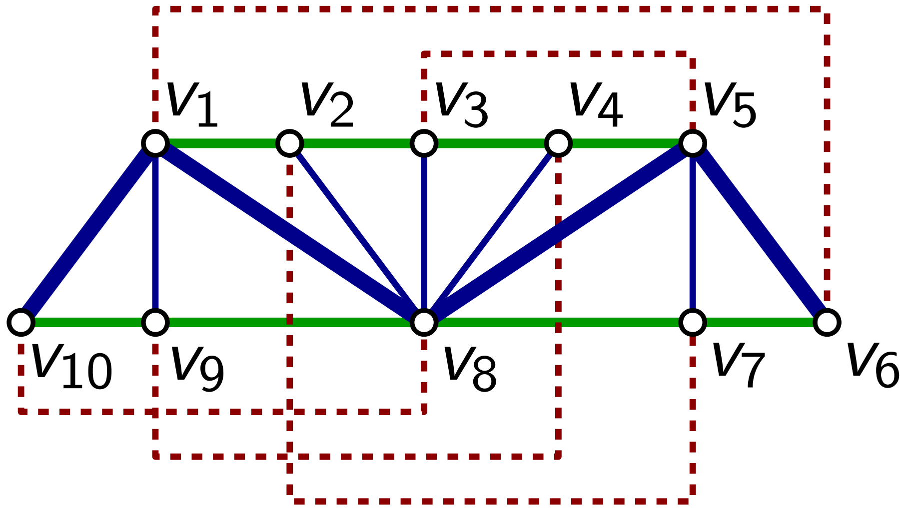


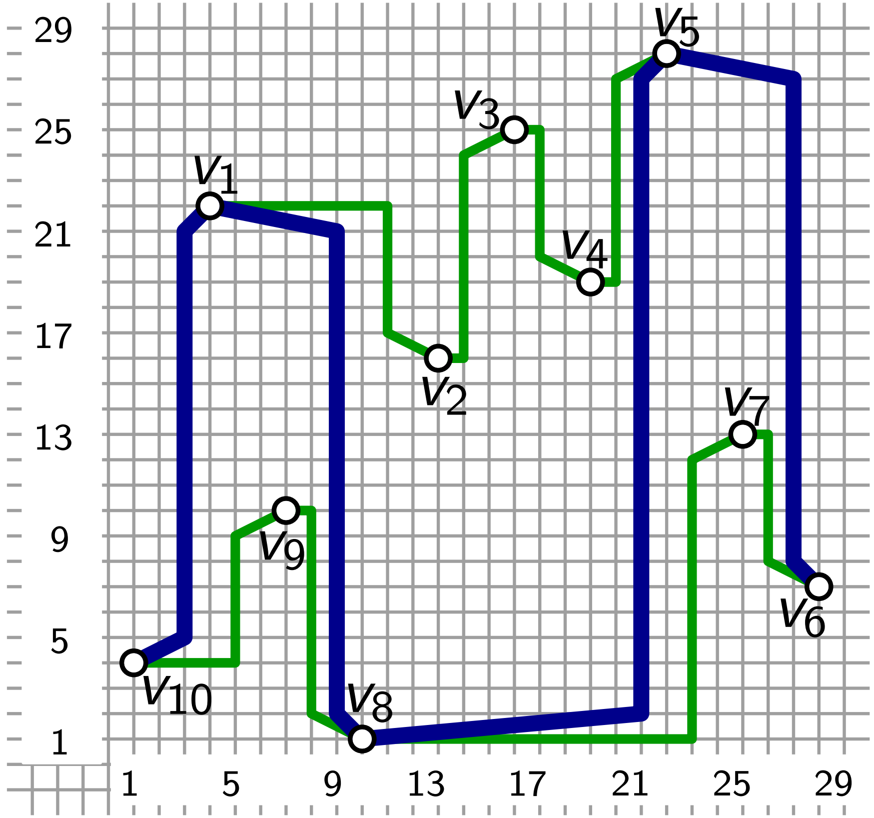
Outerpath × Matching



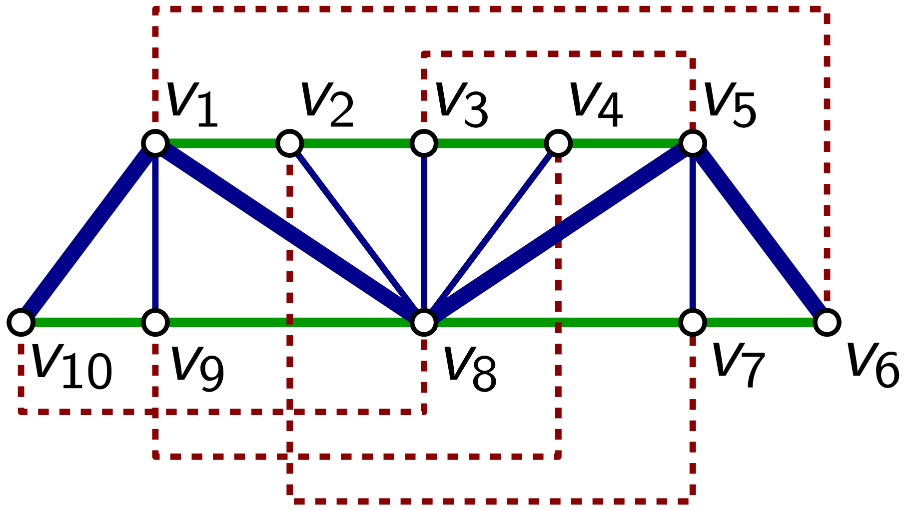


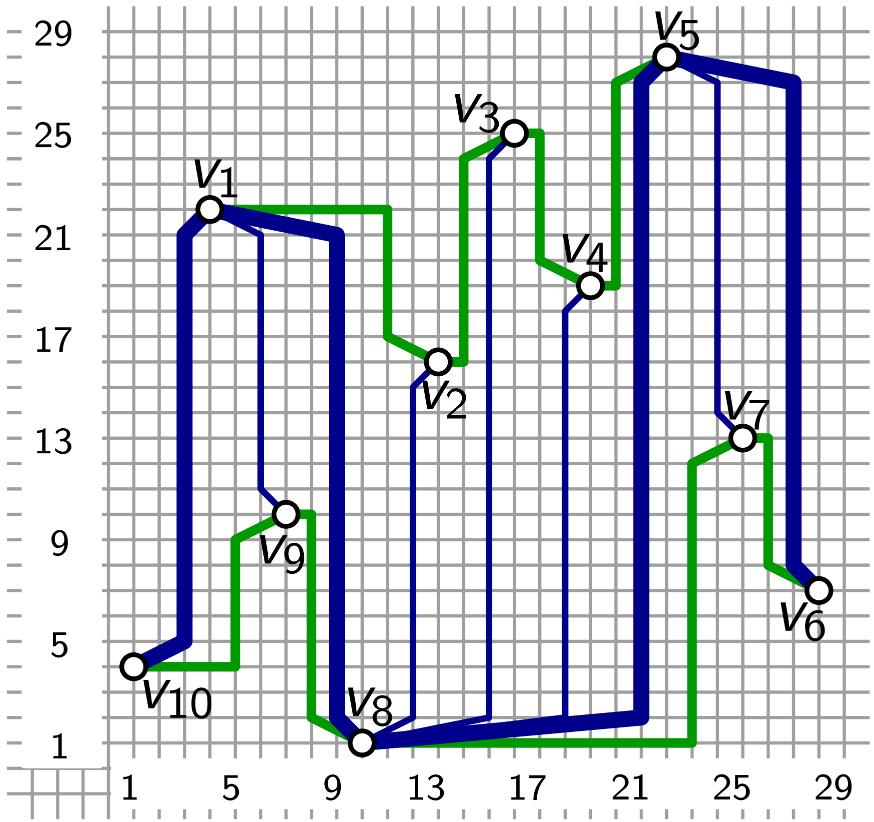
Outerpath × Matching



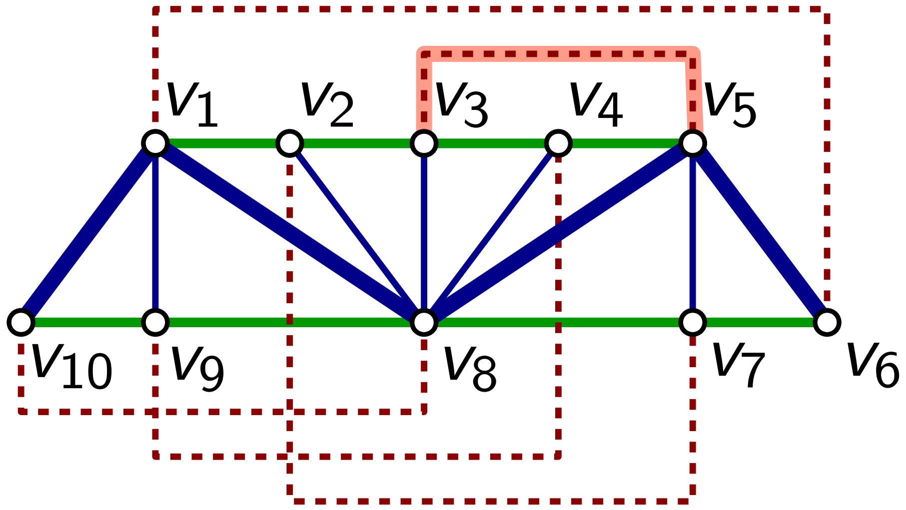


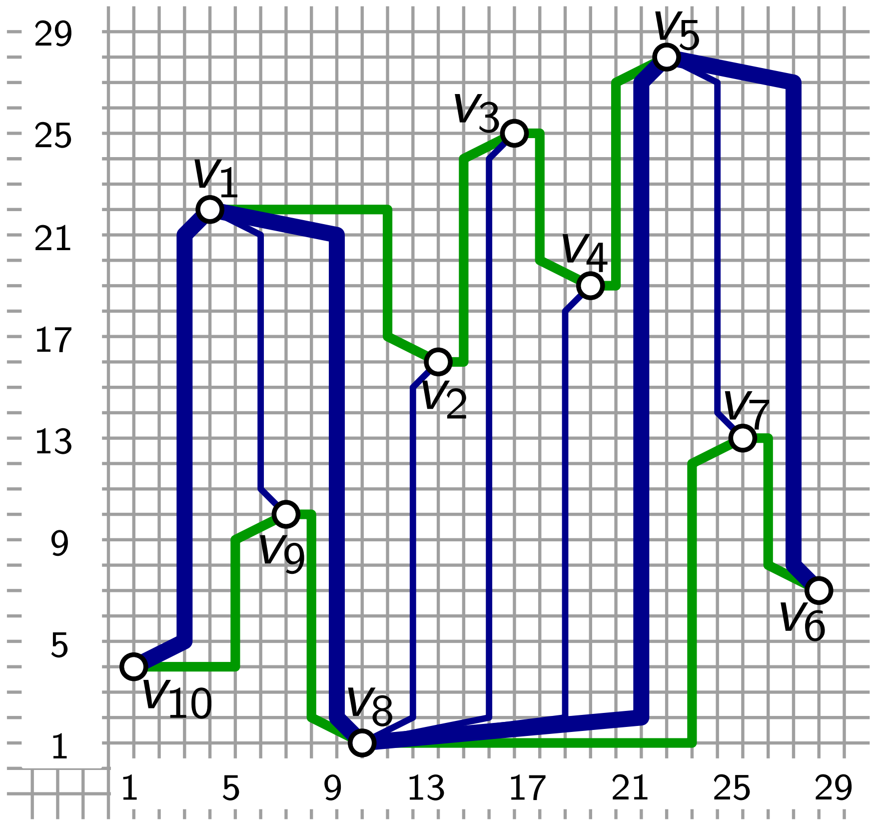
Outerpath × Matching



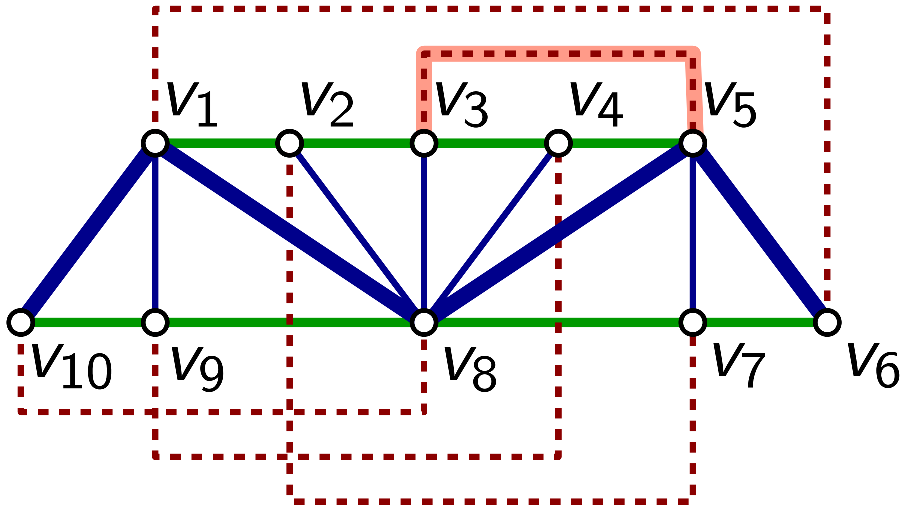


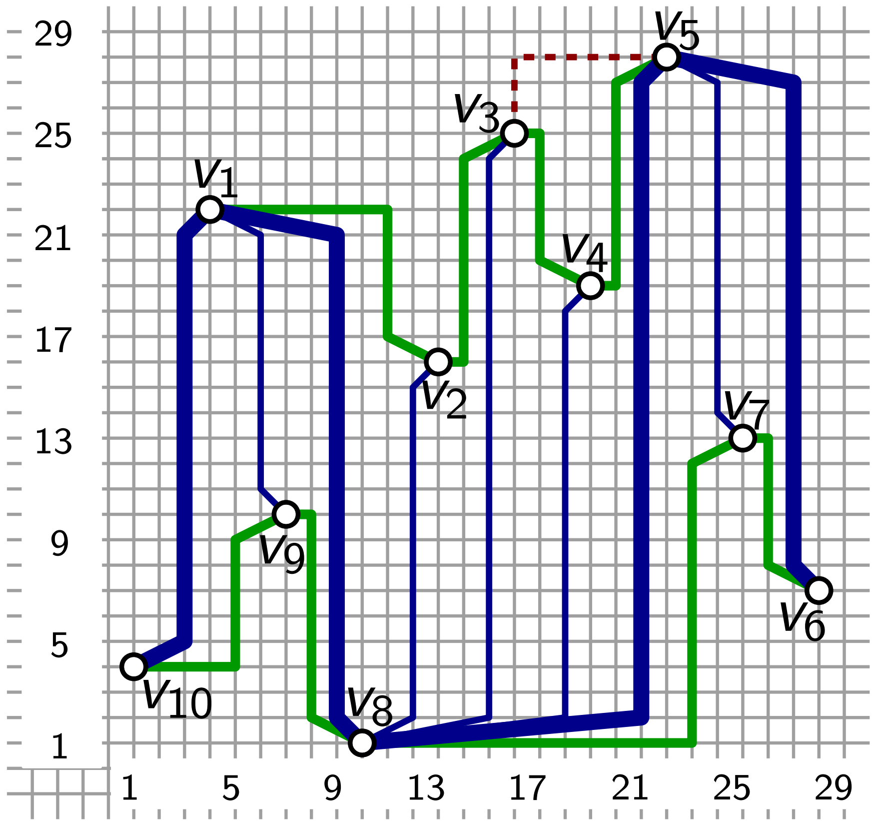
Outerpath × Matching



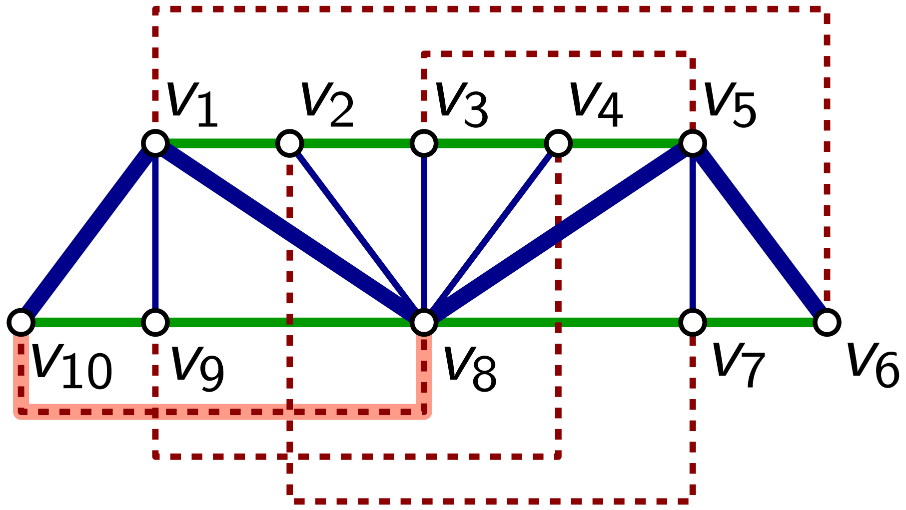


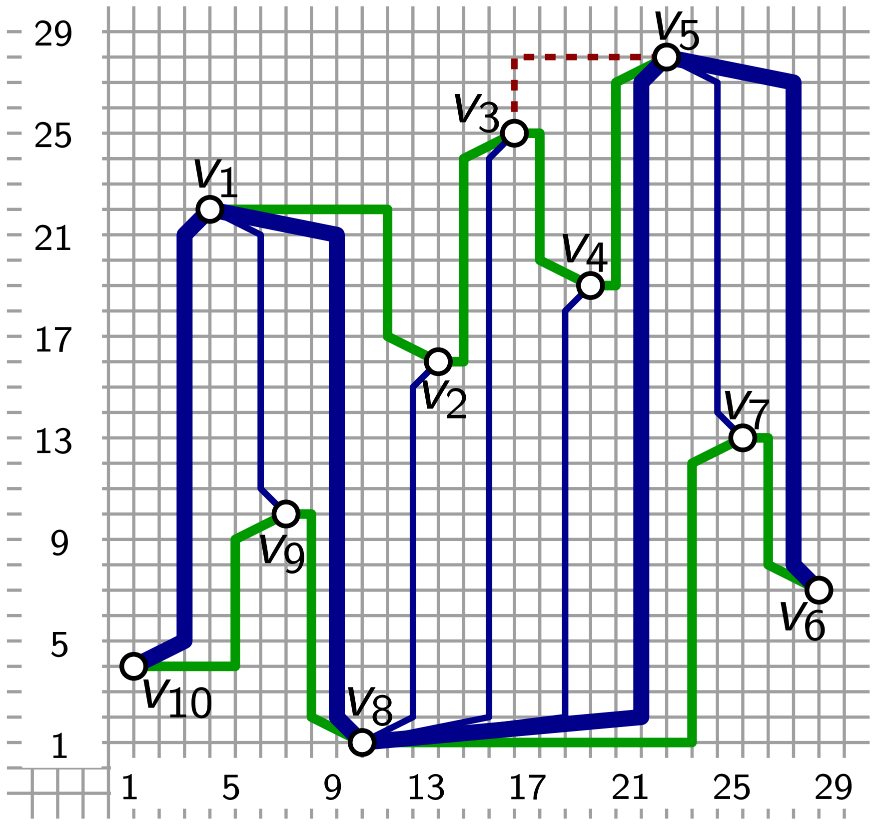
Outerpath × Matching



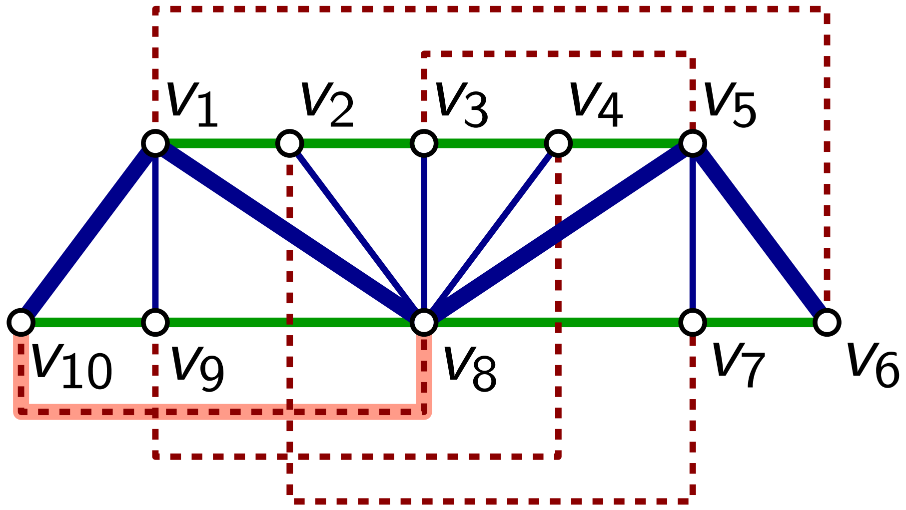


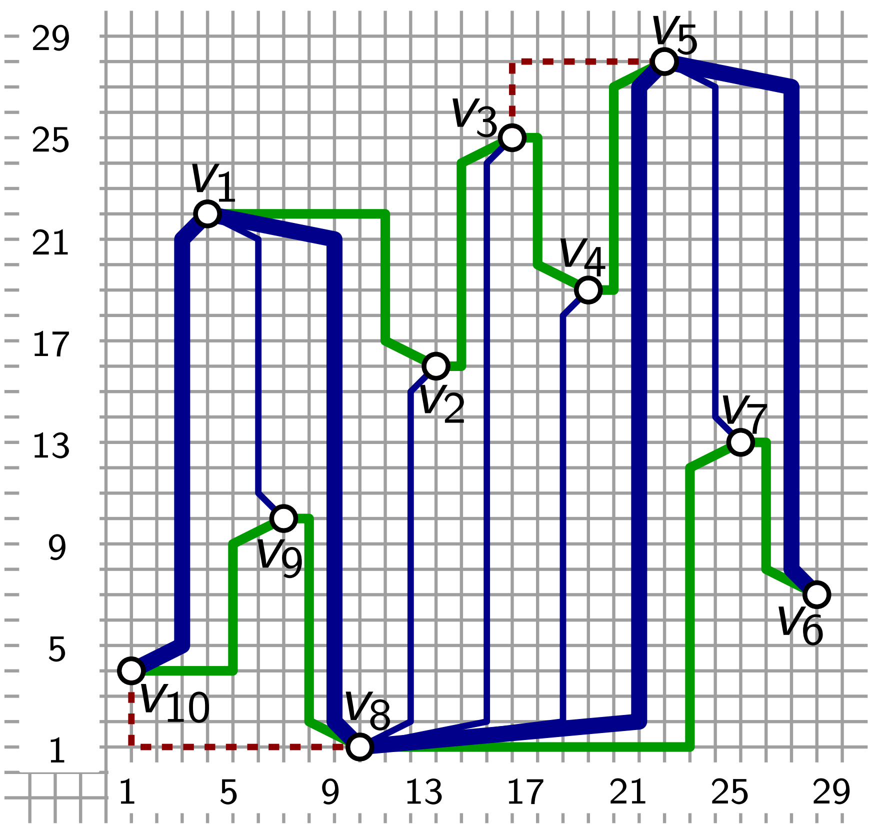
Outerpath × Matching



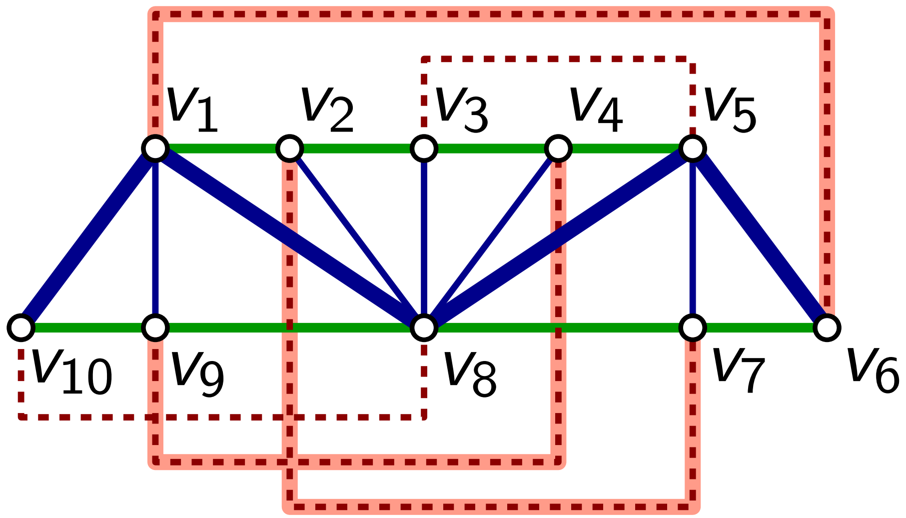


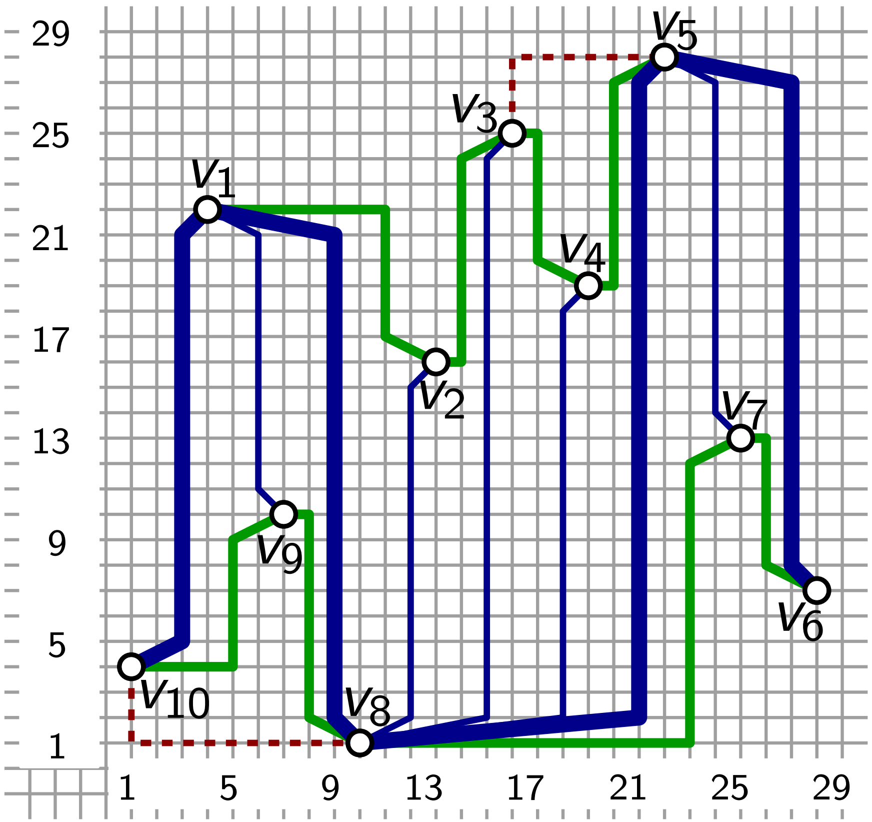
Outerpath × Matching



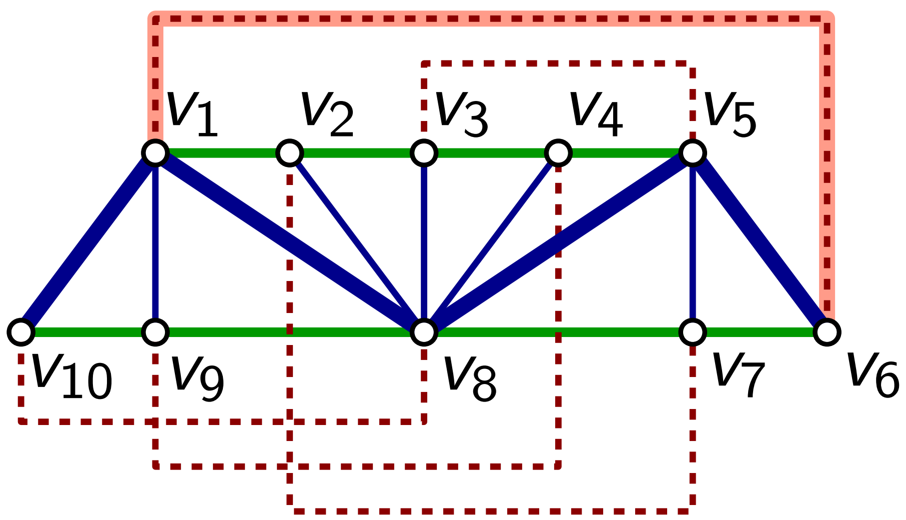


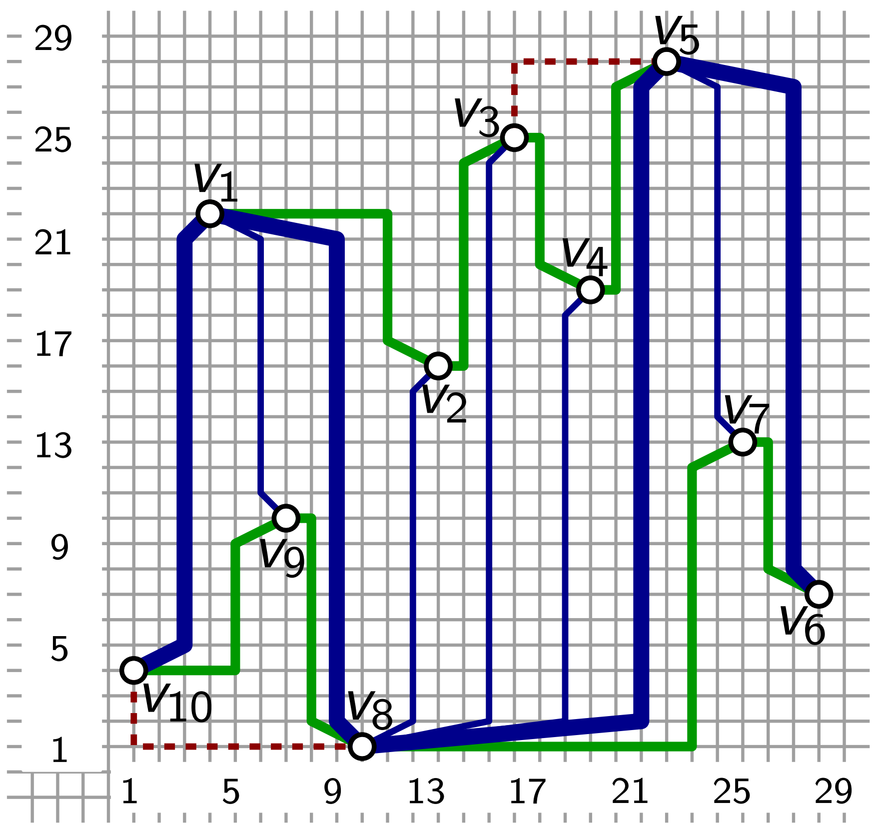
Outerpath × Matching



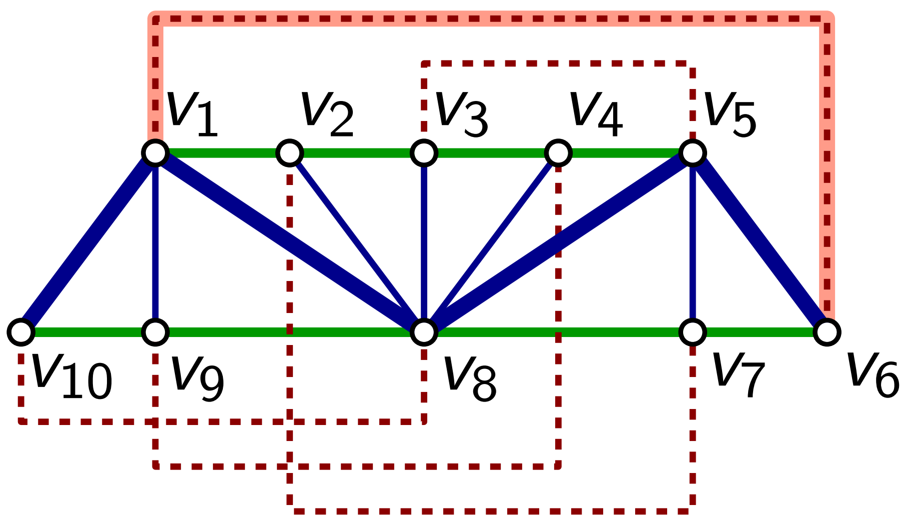


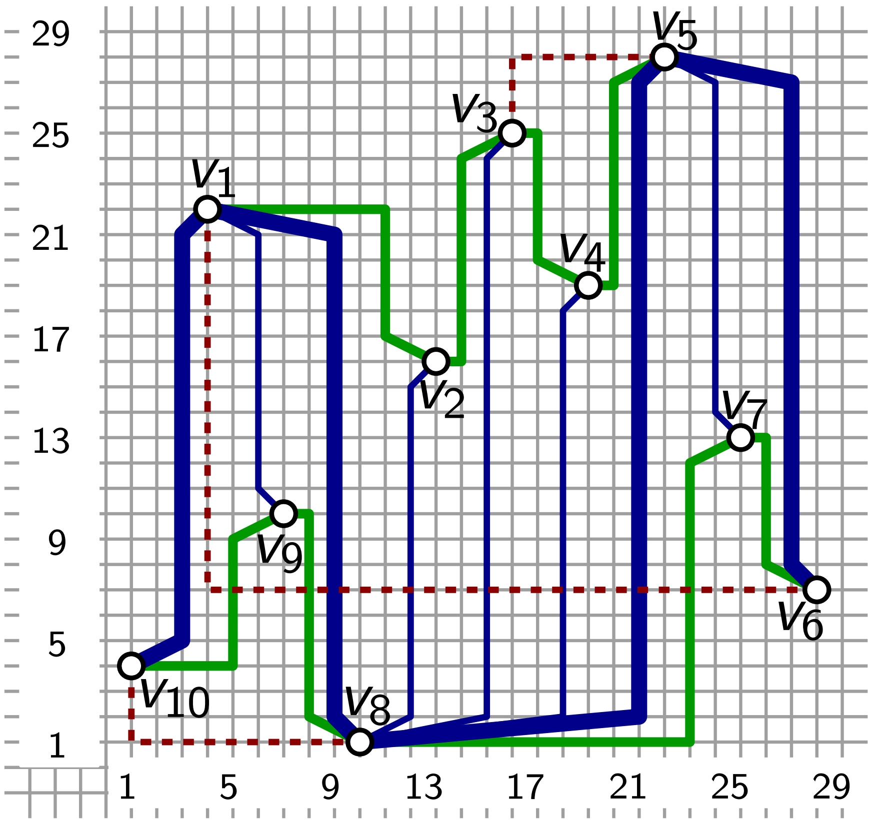
Outerpath × Matching



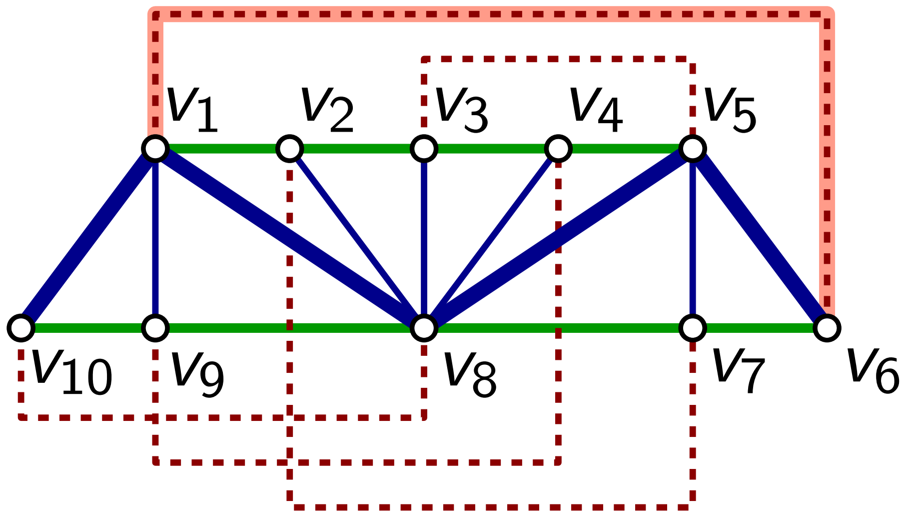


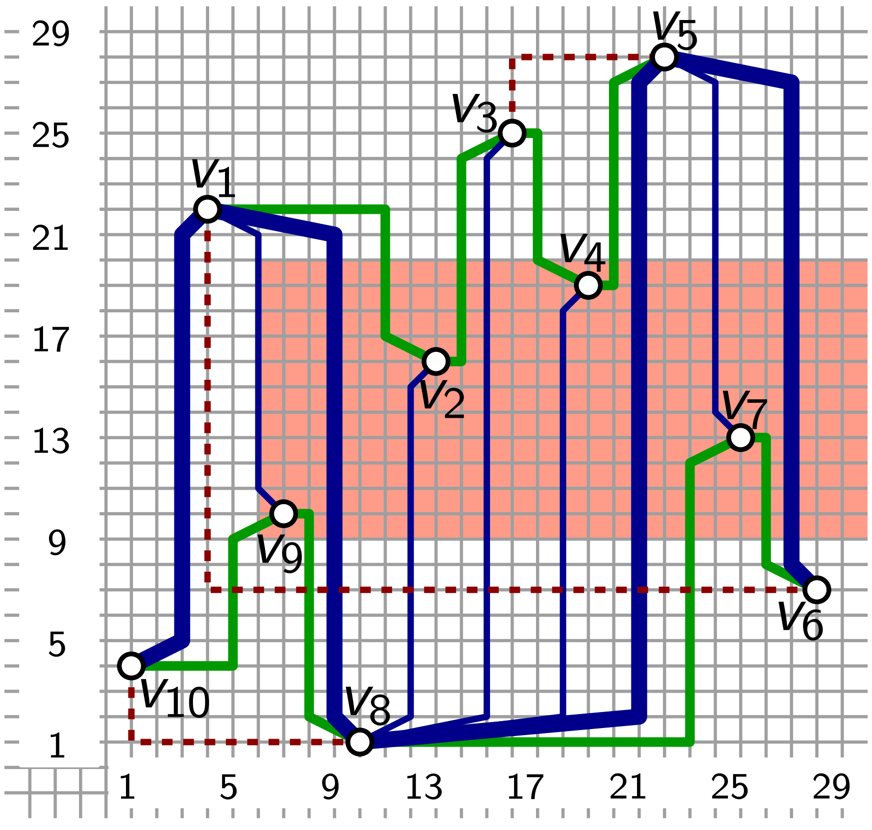
Outerpath × Matching



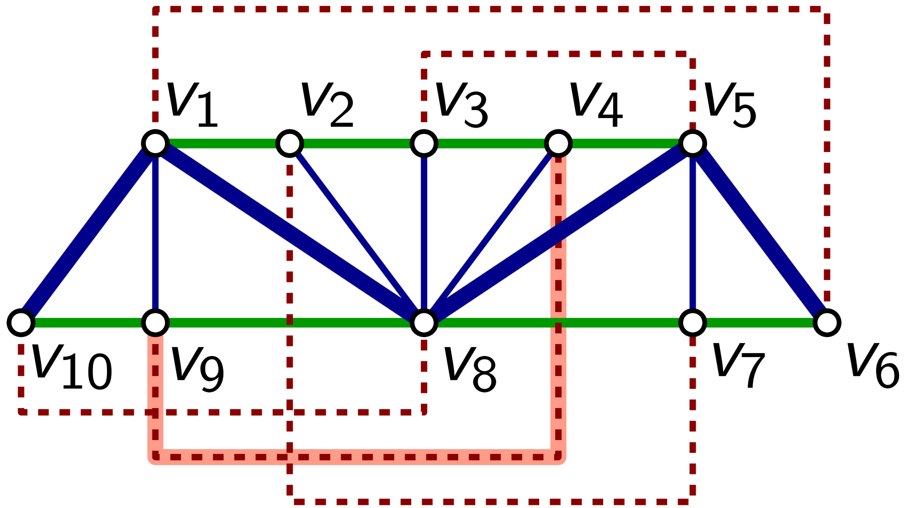


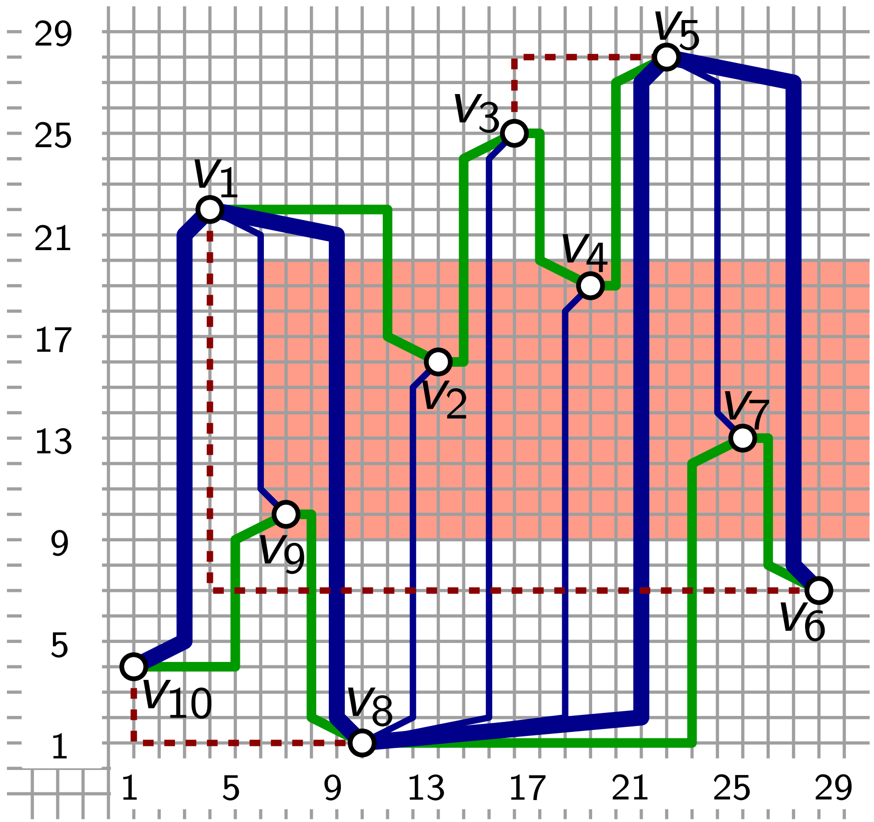
Outerpath × Matching



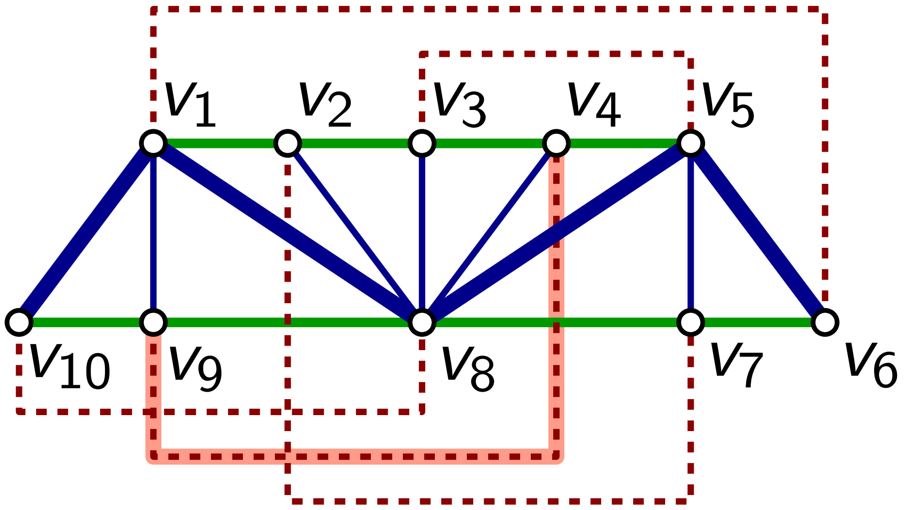


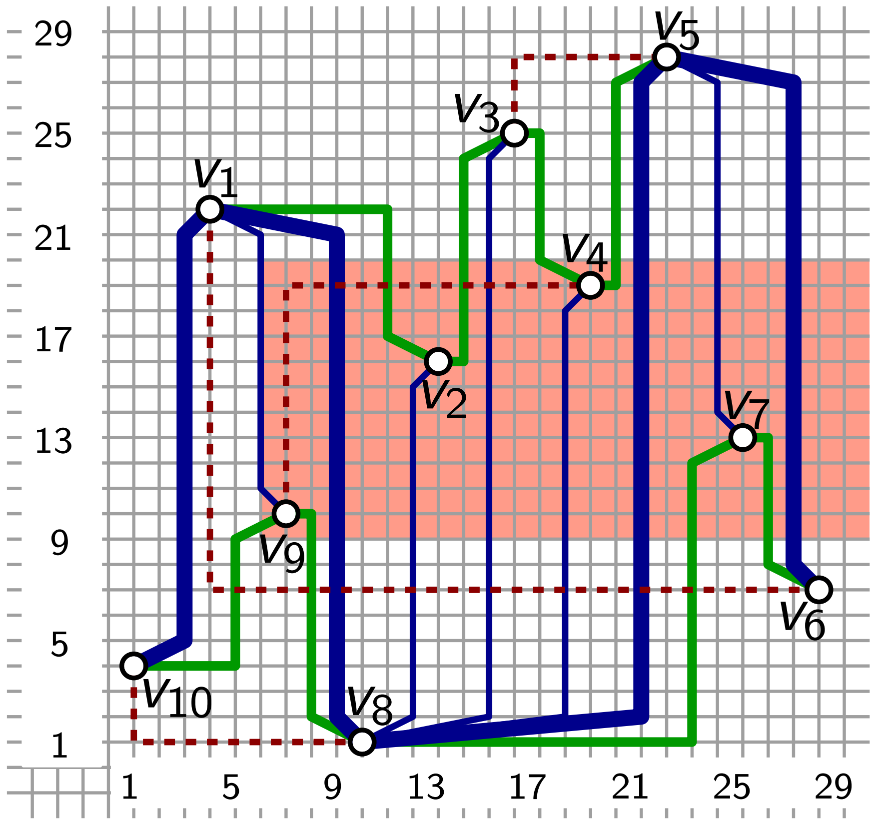
Outerpath × Matching



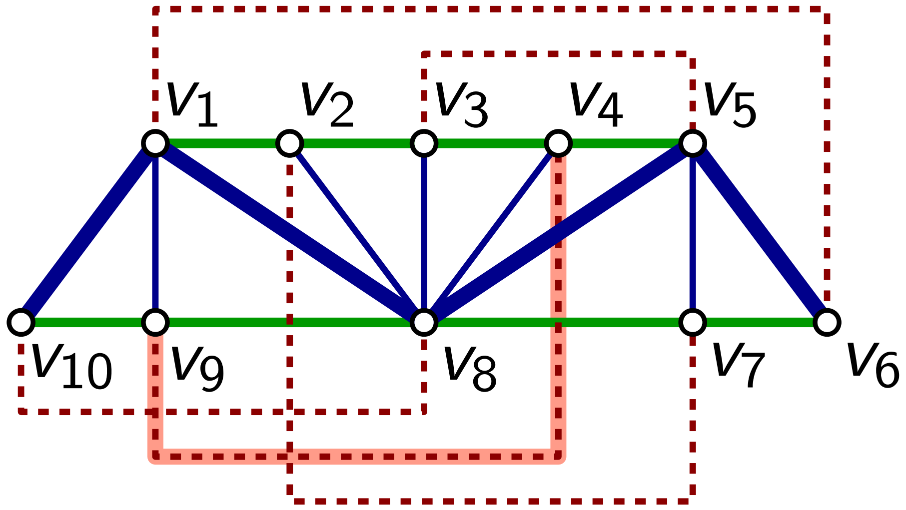


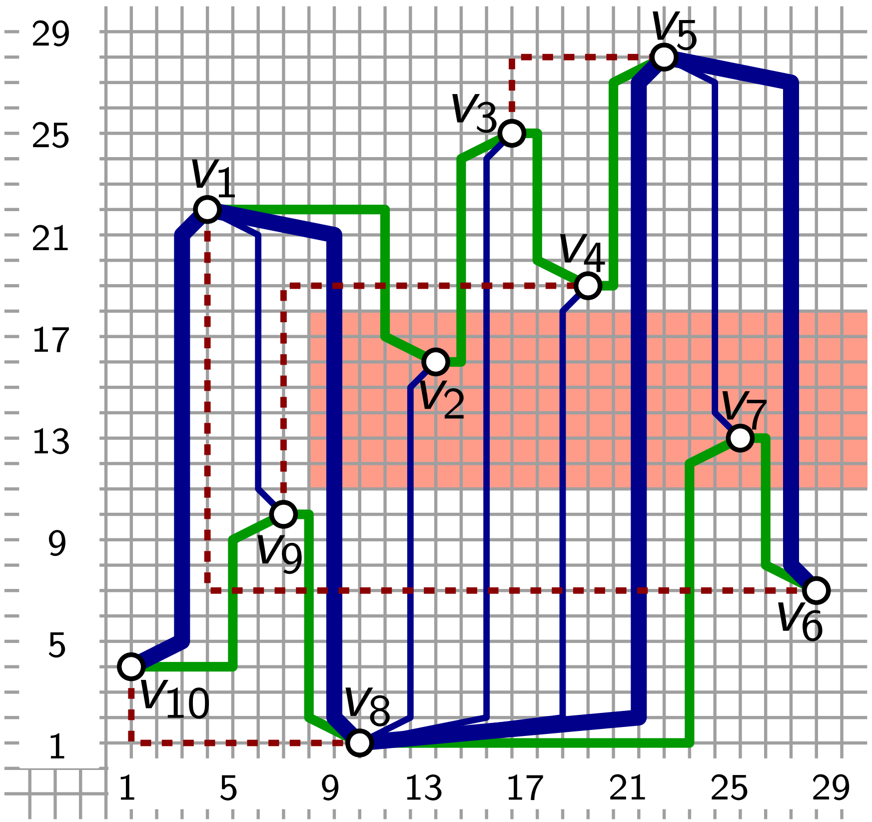
Outerpath × Matching



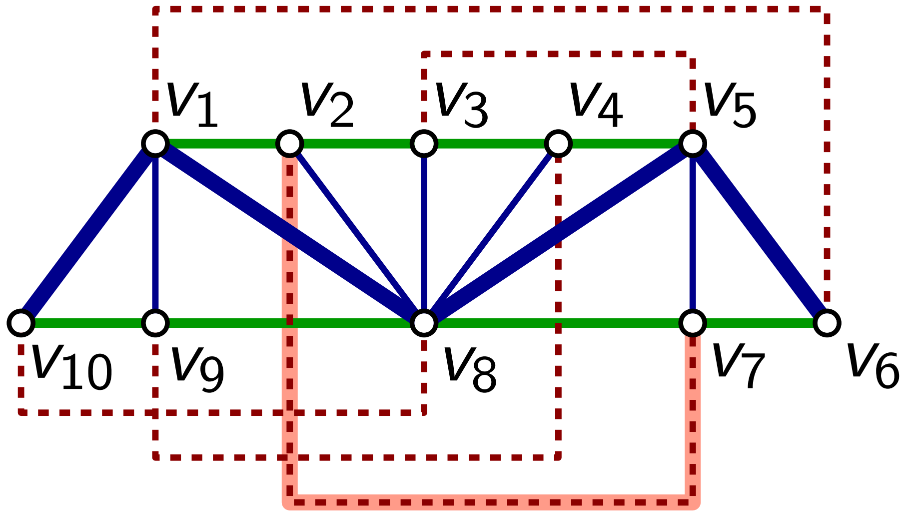


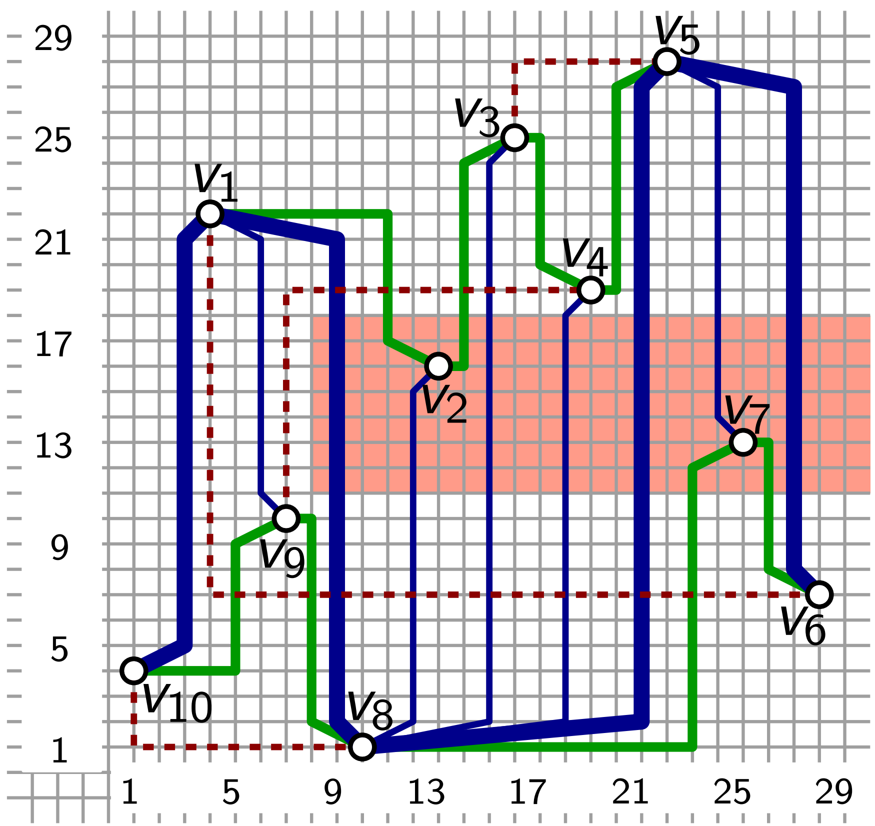
Outerpath × Matching



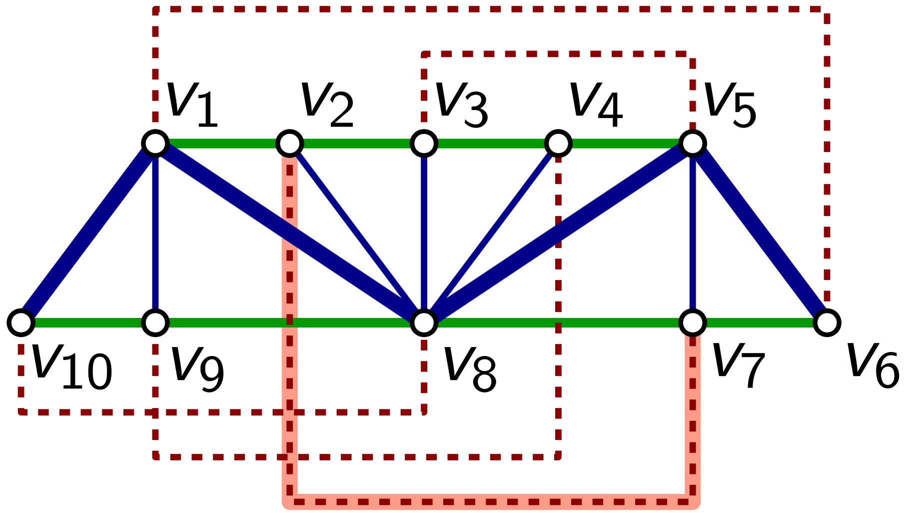


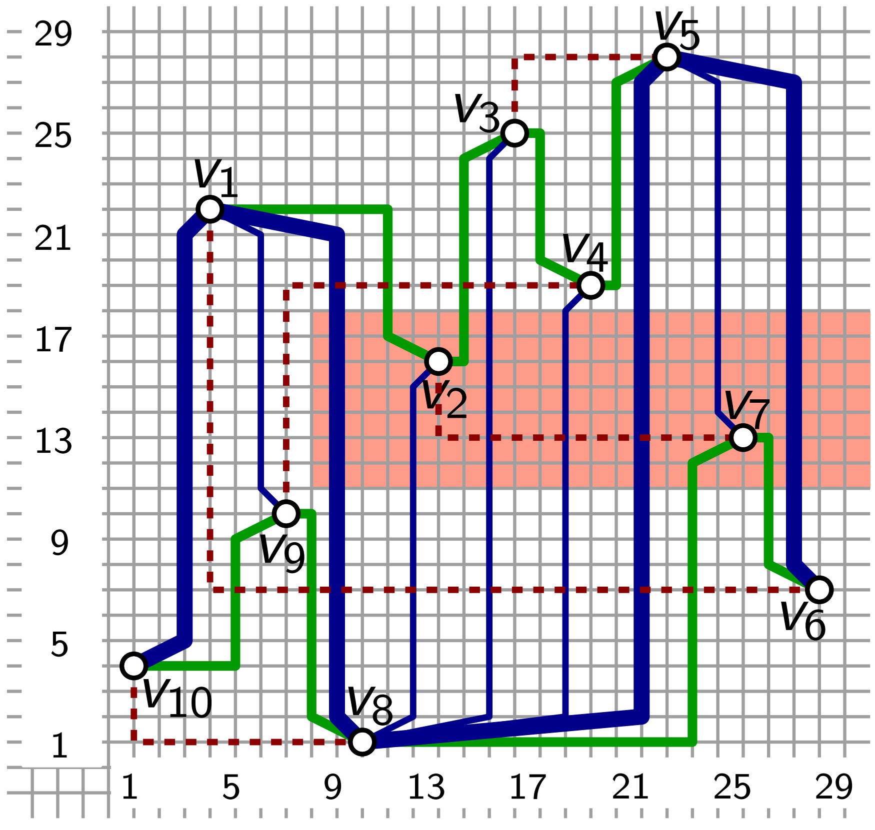
Outerpath × Matching



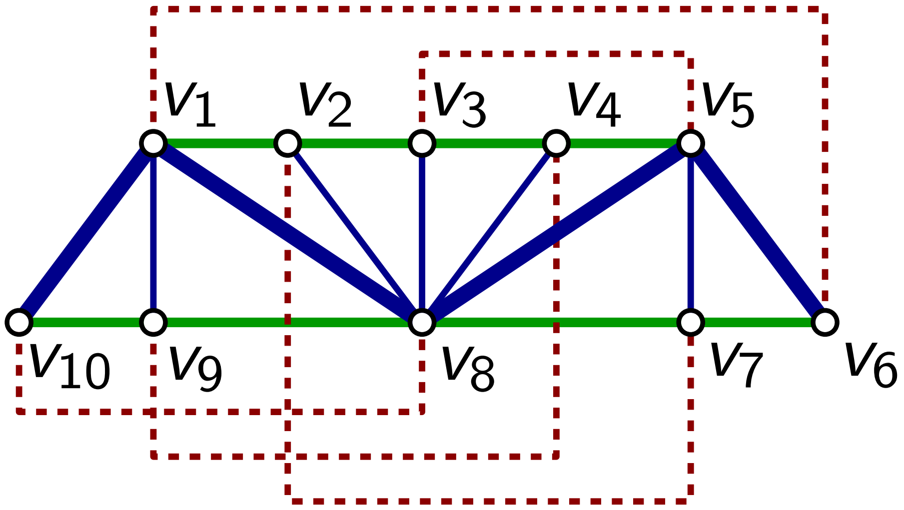


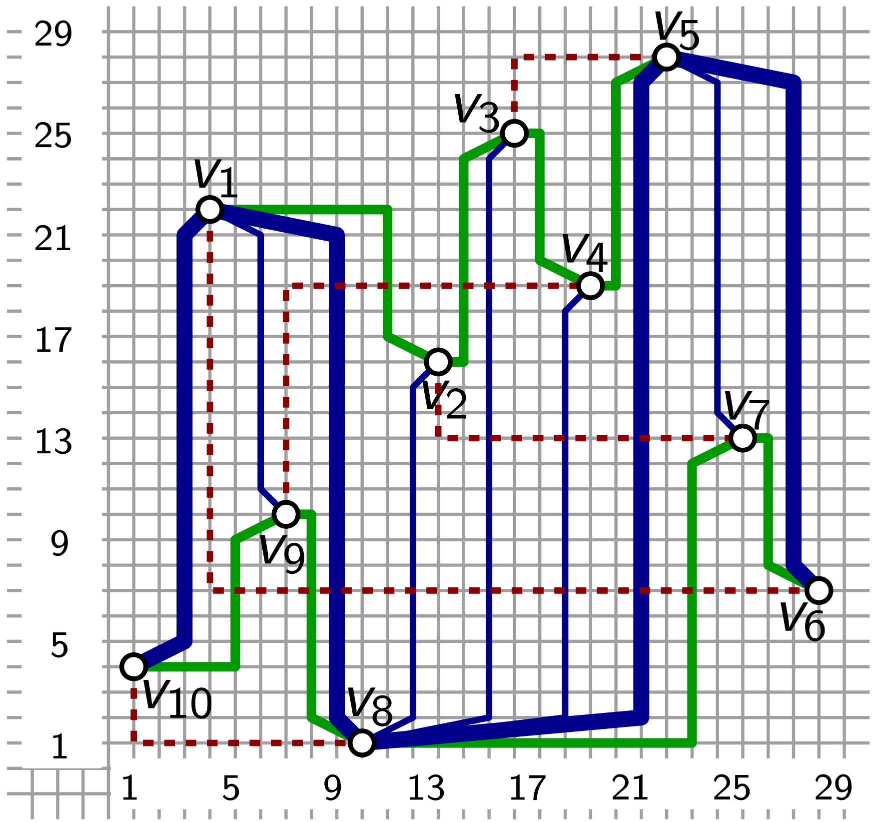
Outerpath × Matching



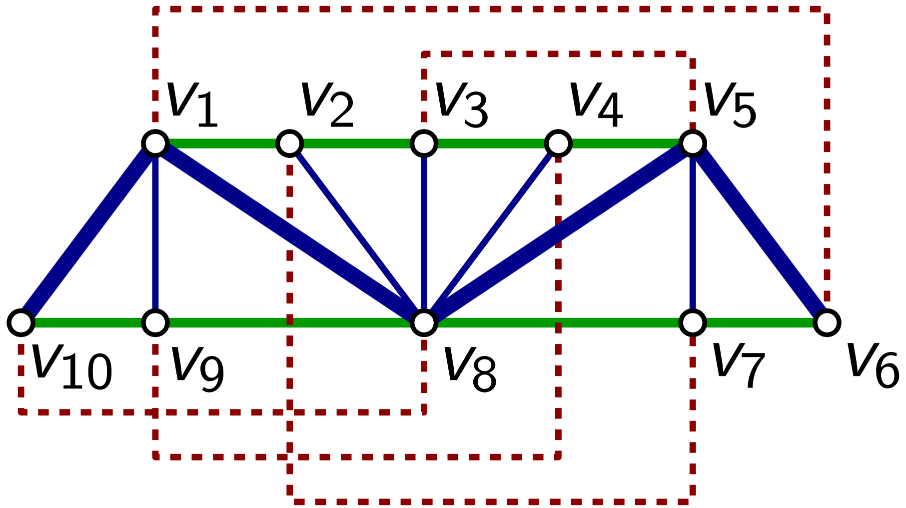


Outerpath × Matching

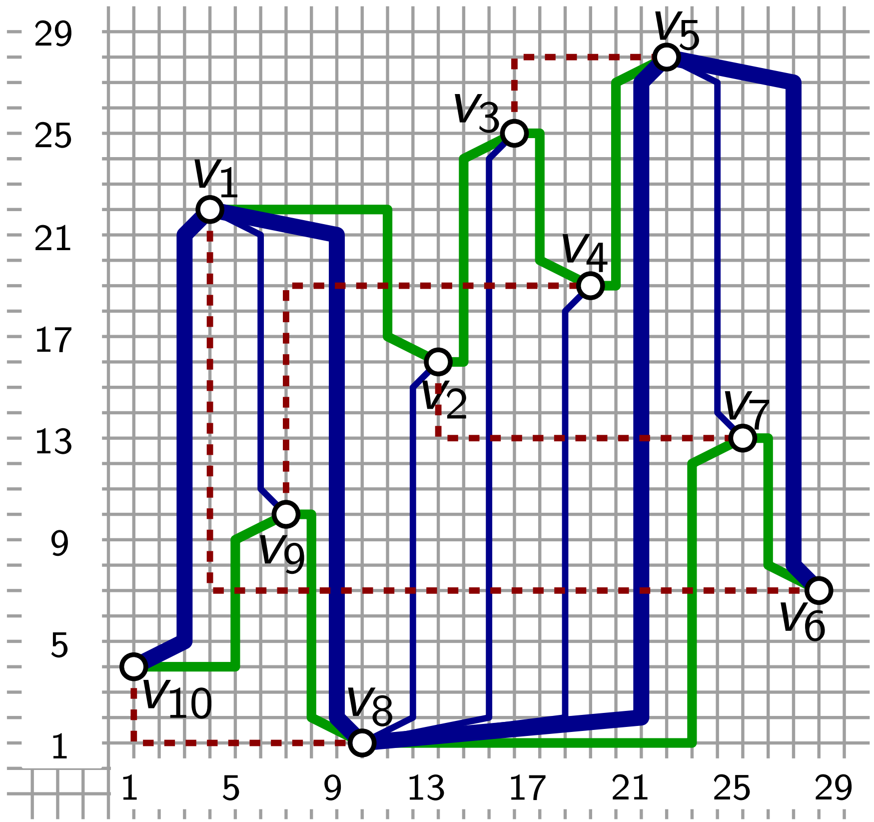




Outerpath × Matching



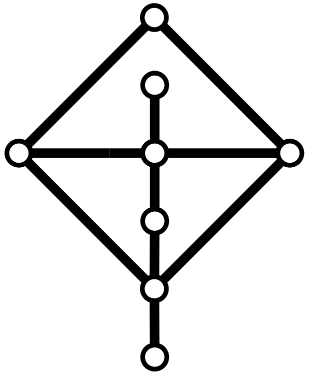
Bends: 2×1
Grid size: $(3n-2)^2$



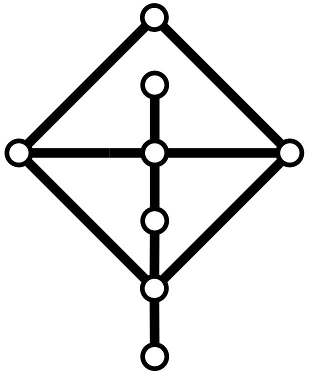
Overview

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

Planar × Planar

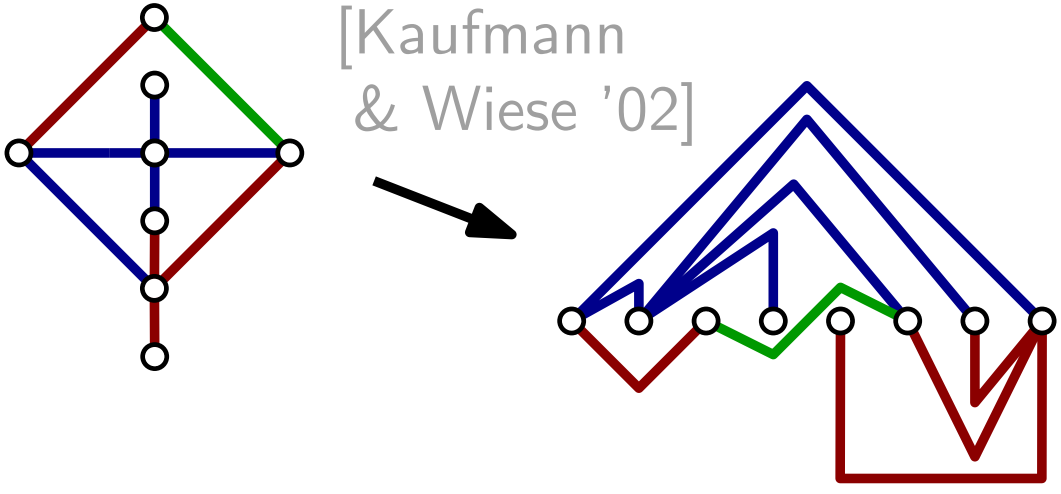


Planar × Planar



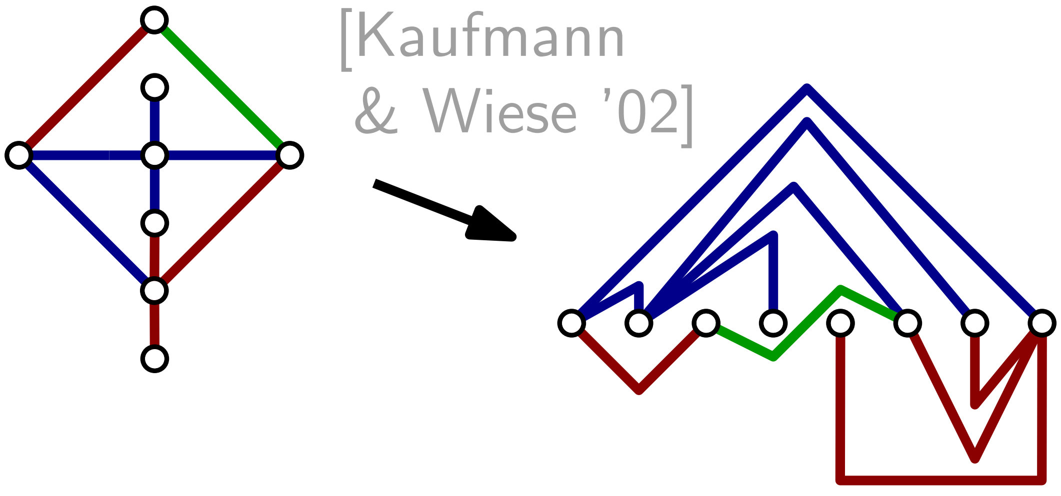
[Kaufmann & Wiese '02]

Planar × Planar



[Kaufmann & Wiese '02]

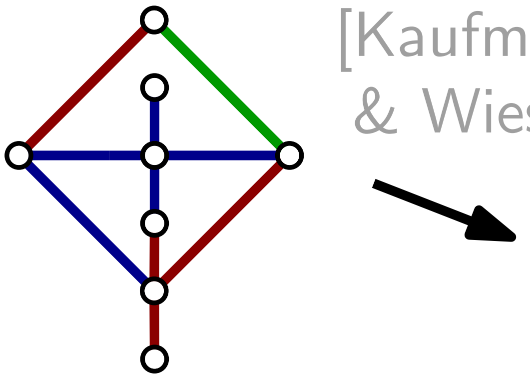
Planar × Planar



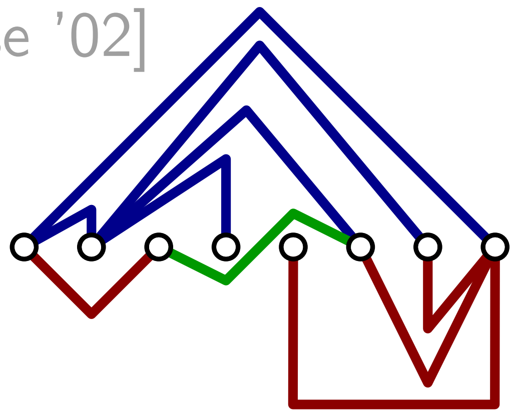
Graph 1: x-coordinates

Graph 2: y-coordinates

Planar × Planar

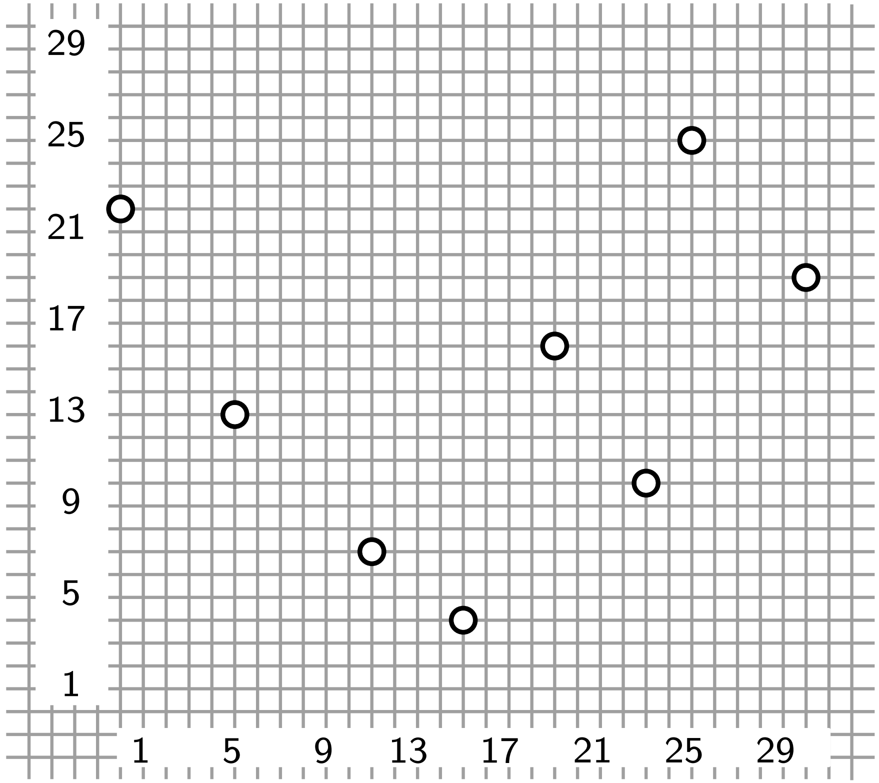


[Kaufmann & Wiese '02]

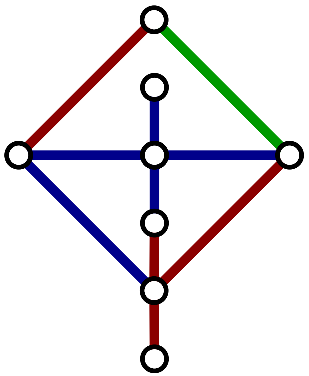


Graph 1: x-coordinates

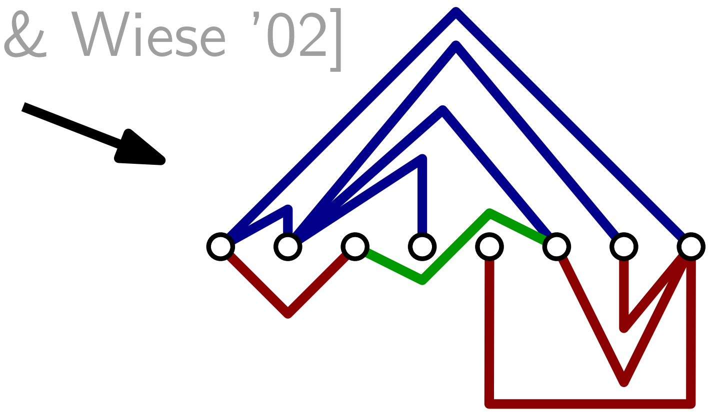
Graph 2: y-coordinates



Planar × Planar

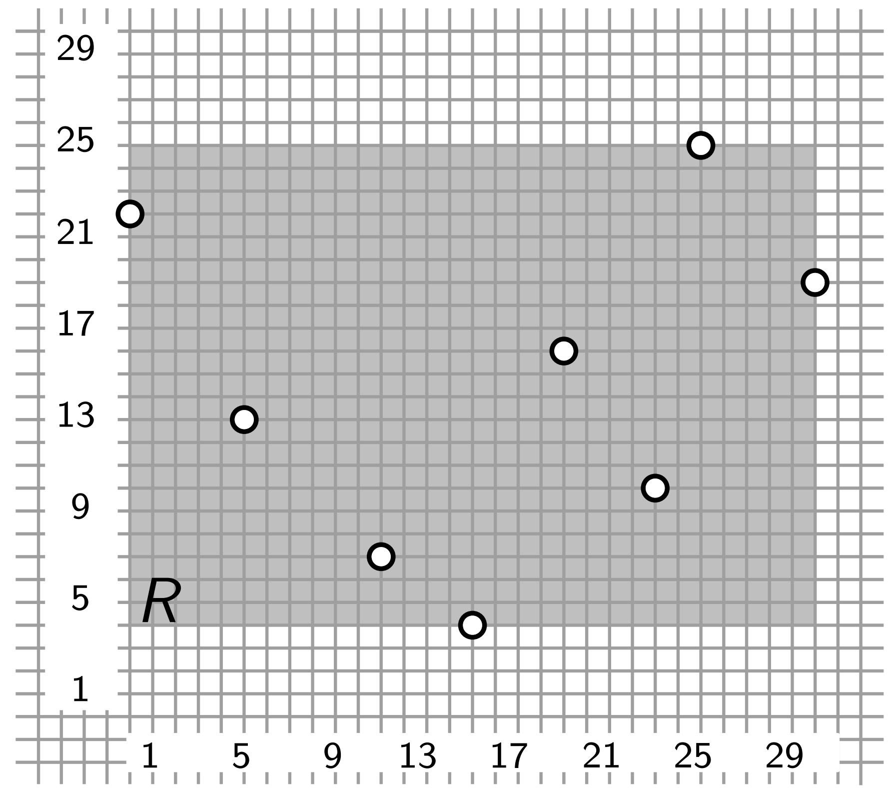


[Kaufmann & Wiese '02]

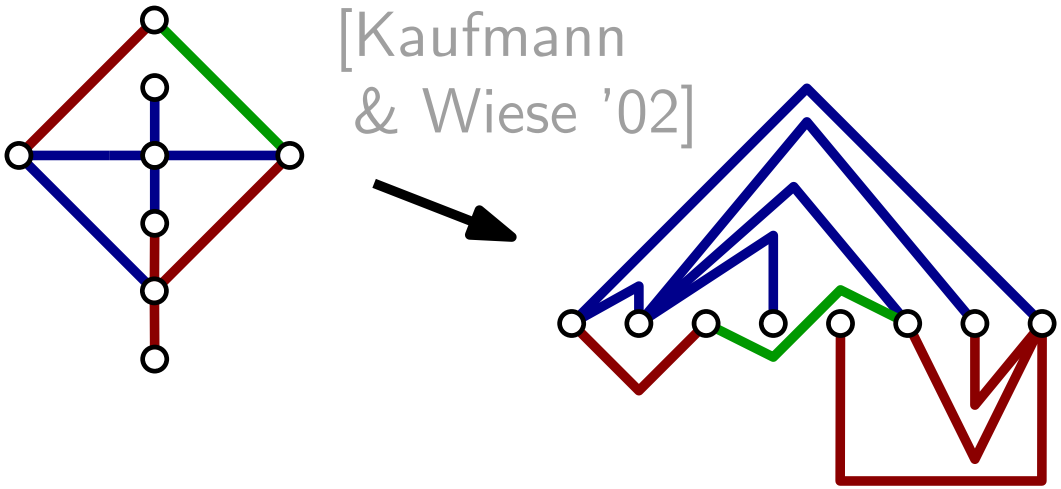


Graph 1: x-coordinates

Graph 2: y-coordinates



Planar × Planar

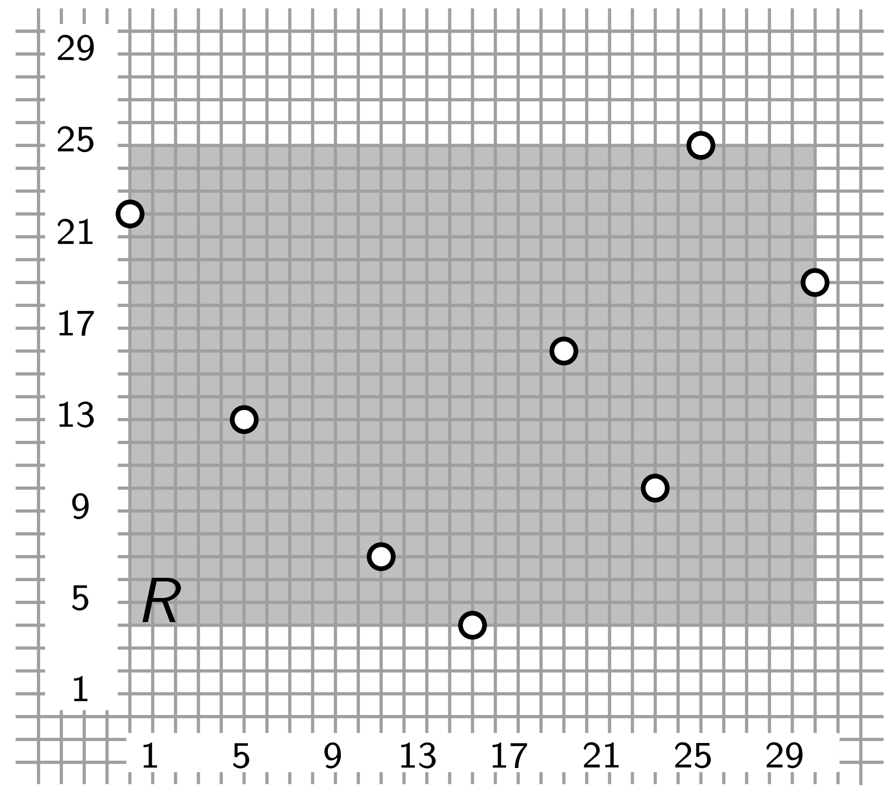


[Kaufmann & Wiese '02]

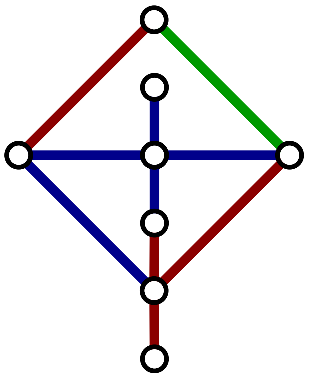
Graph 1: x-coordinates

In R: All segments vertical or slanted of y-length 1.

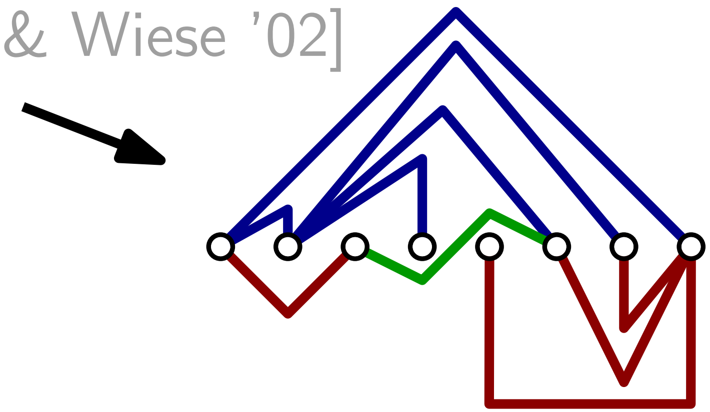
Graph 2: y-coordinates



Planar × Planar



[Kaufmann & Wiese '02]

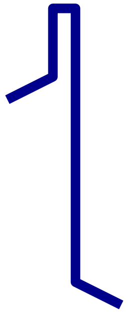


Graph 1: *x*-coordinates

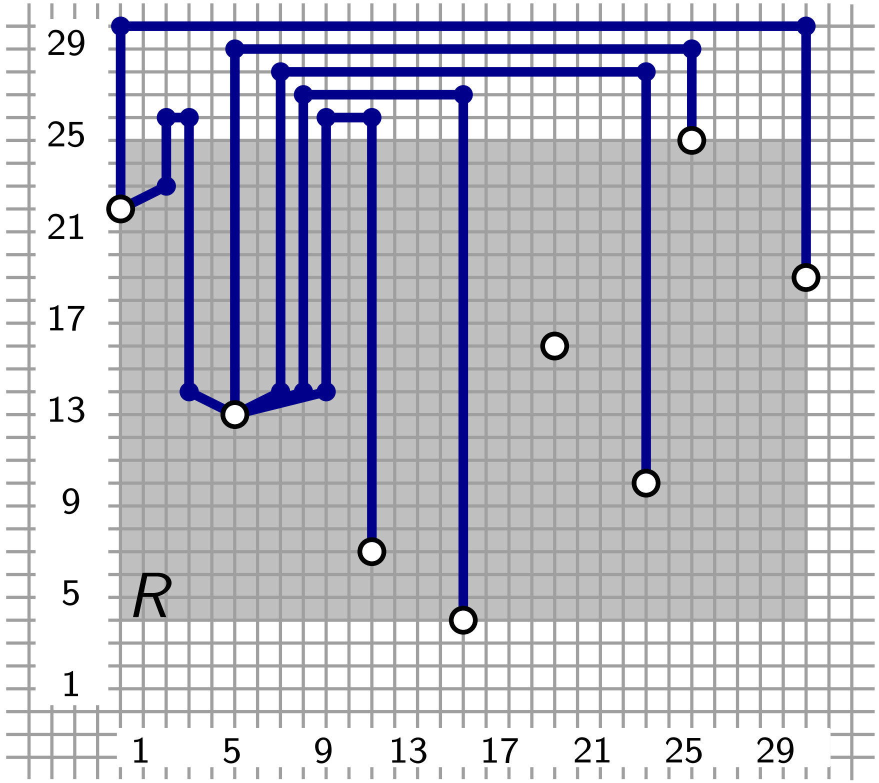
In *R*: All segments vertical or slanted of *y*-length 1.

Graph 2: *y*-coordinates

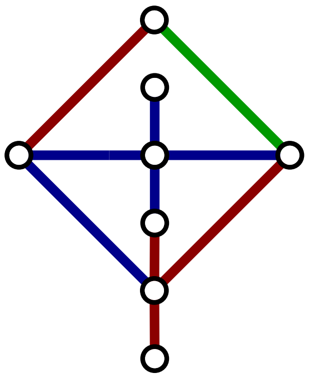
Edges:



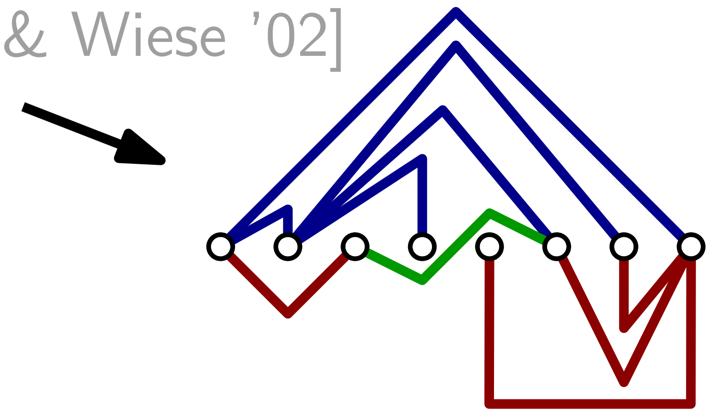
4



Planar × Planar



[Kaufmann & Wiese '02]

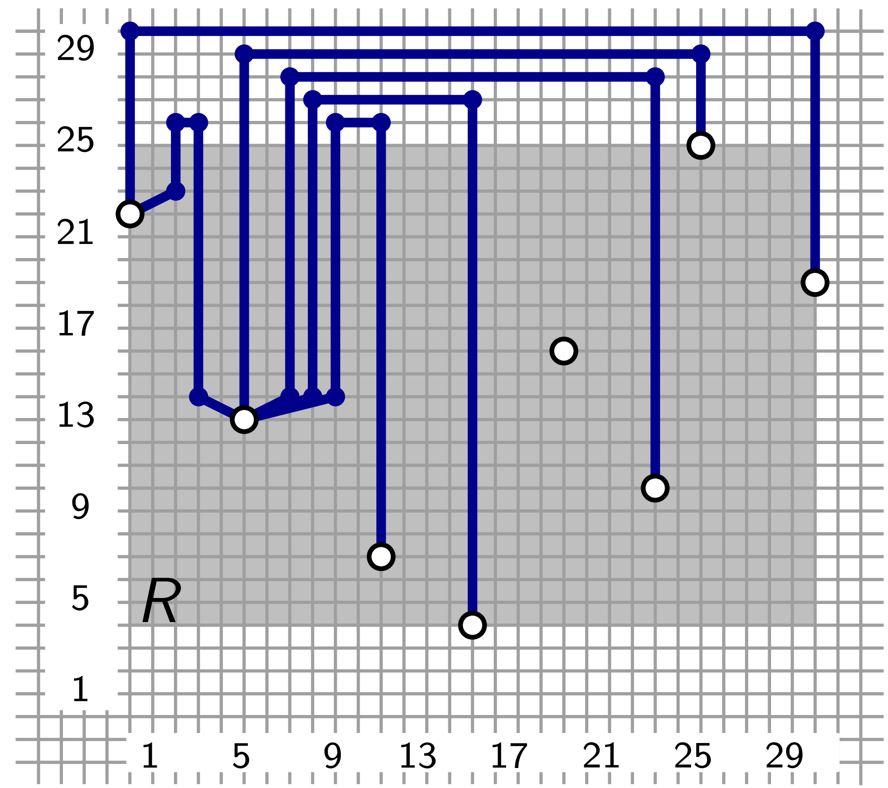


New idea:
Place turns outside of R!

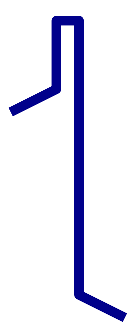
Graph 1: x-coordinates

In R: All segments vertical or slanted of y-length 1.

Graph 2: y-coordinates

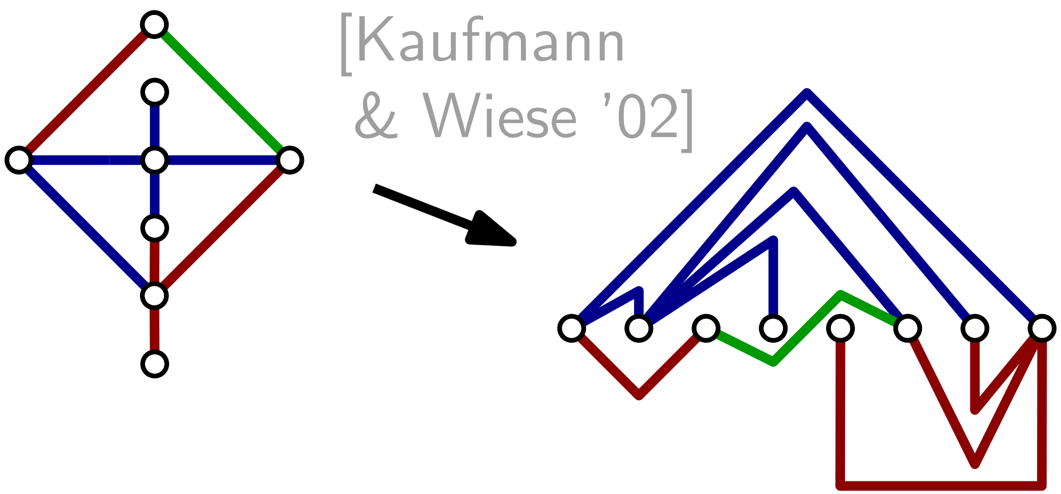


Edges:



4

Planar × Planar

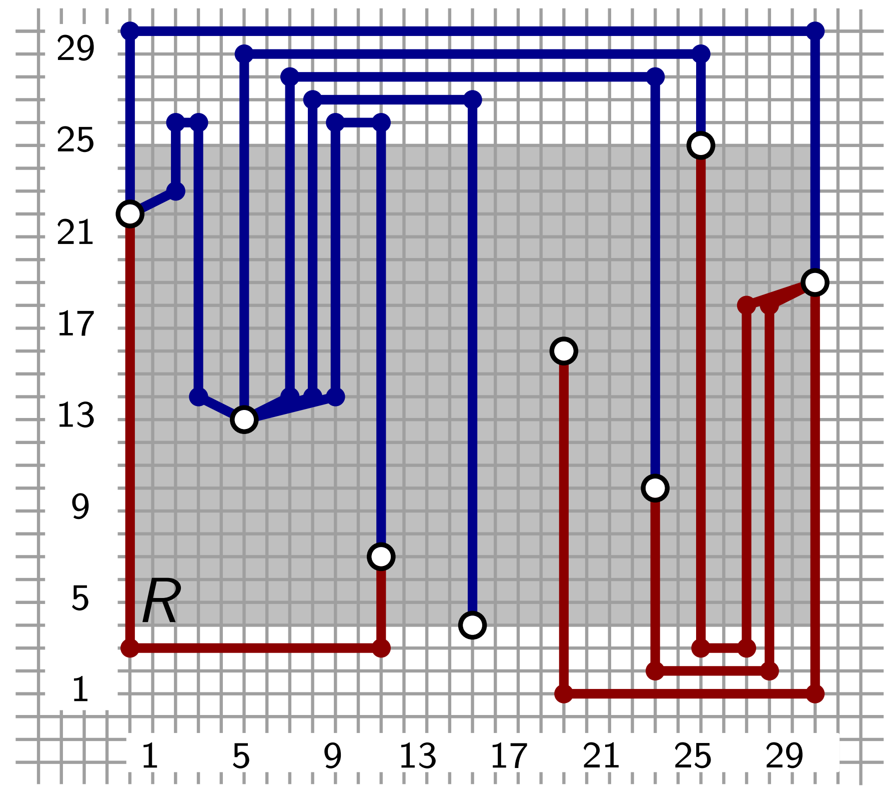


New idea:
Place turns outside of *R*!

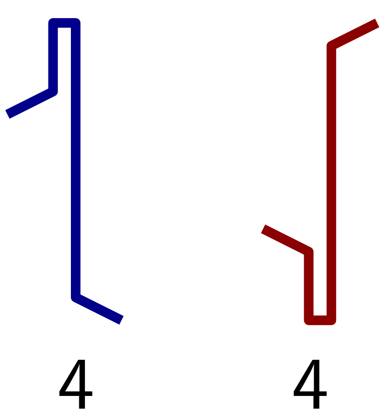
Graph 1: *x*-coordinates

In *R*: All segments vertical or slanted of *y*-length 1.

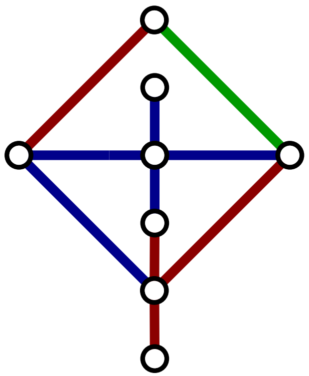
Graph 2: *y*-coordinates



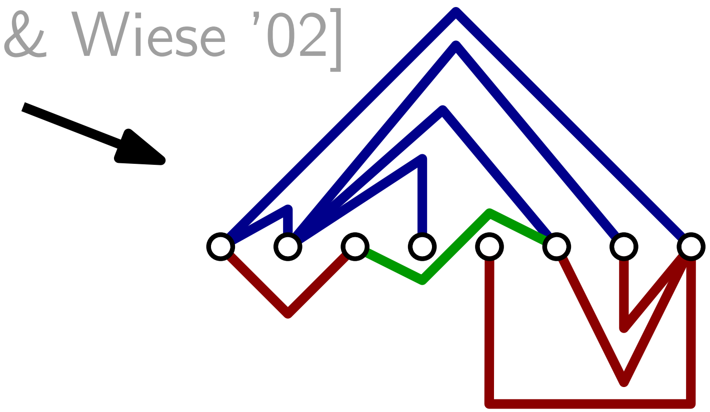
Edges:



Planar × Planar



[Kaufmann & Wiese '02]



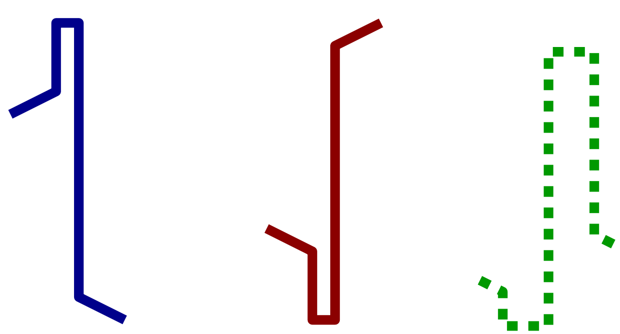
New idea:
Place turns outside of *R*!

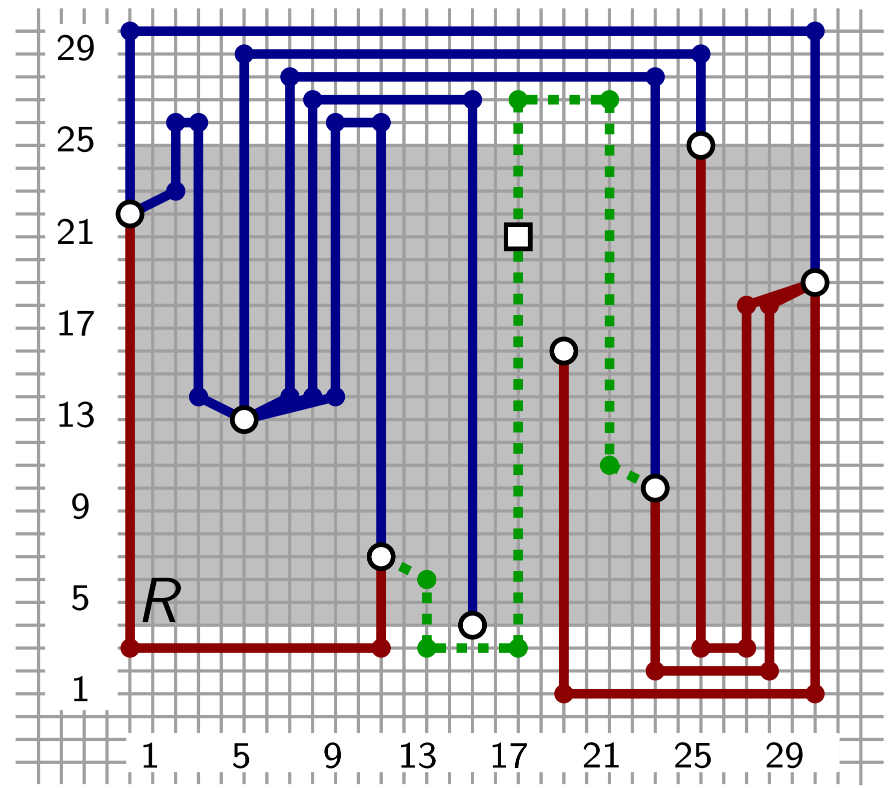
Graph 1: x-coordinates

In *R*: All segments vertical or slanted of y-length 1.

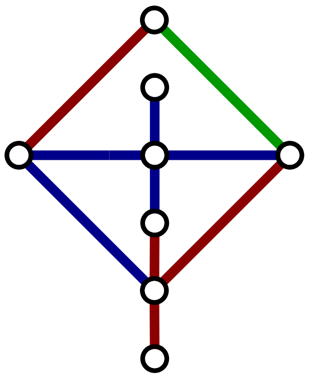
Graph 2: y-coordinates

Edges:

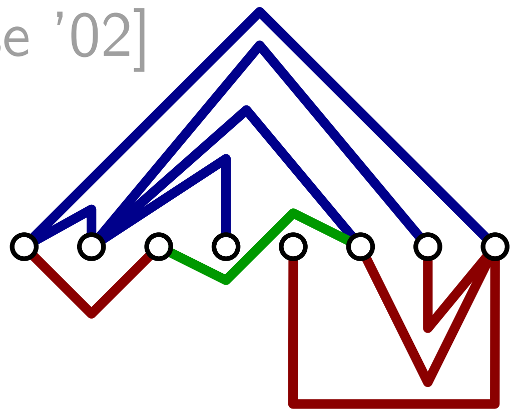




Planar × Planar



[Kaufmann & Wiese '02]



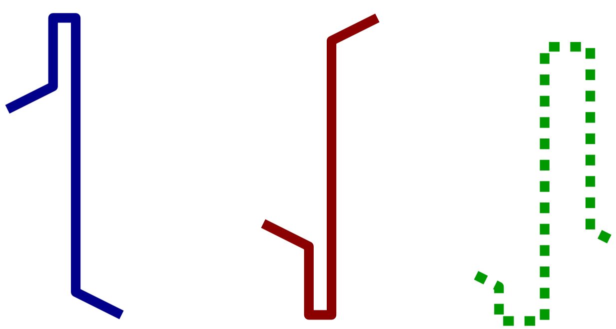
New idea:
Place turns outside of *R*!

Graph 1: *x*-coordinates

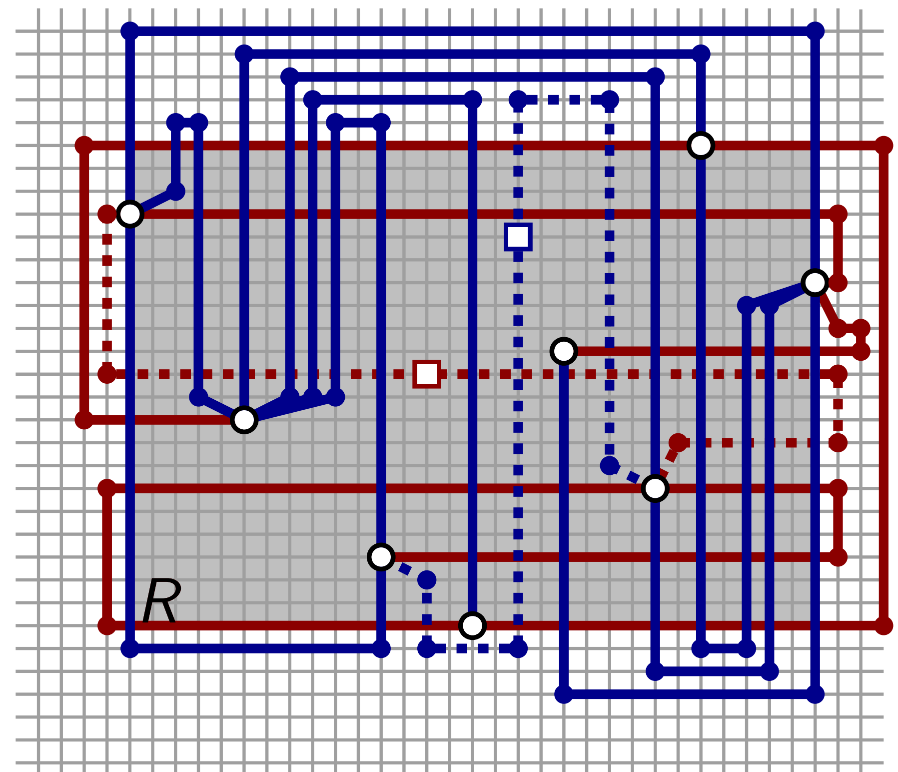
In *R*: All segments vertical or slanted of *y*-length 1.

Graph 2: *y*-coordinates

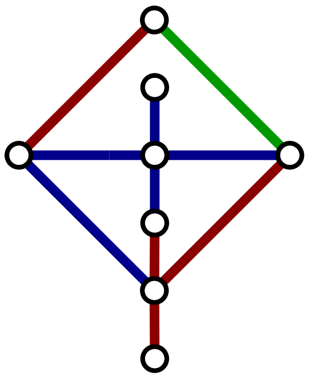
Edges:



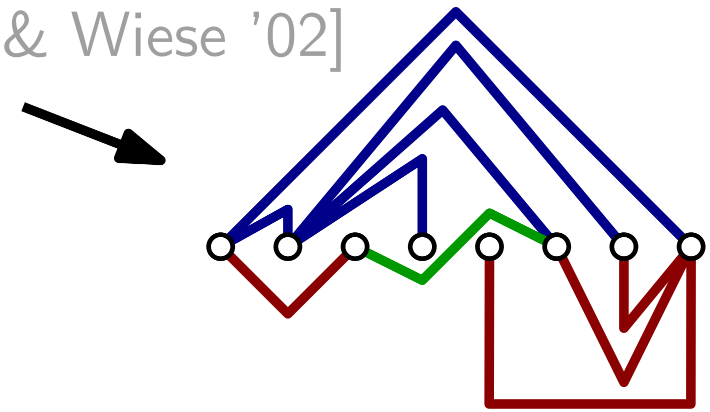
4 4 6



Planar × Planar



[Kaufmann & Wiese '02]

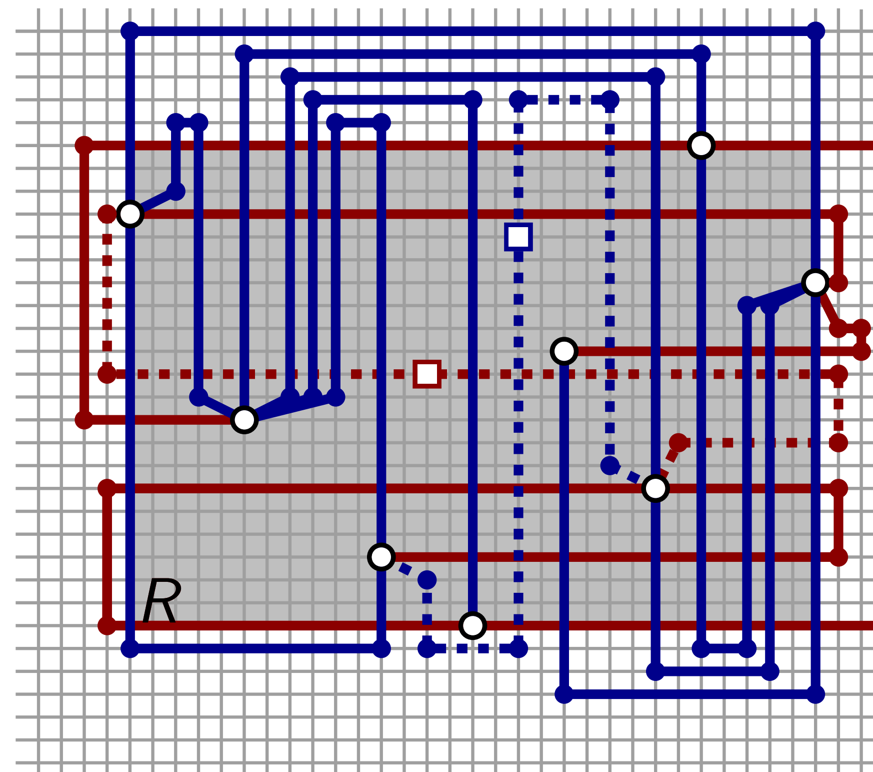


Bends: 6×6
Grid size: $(14n - 26)^2$

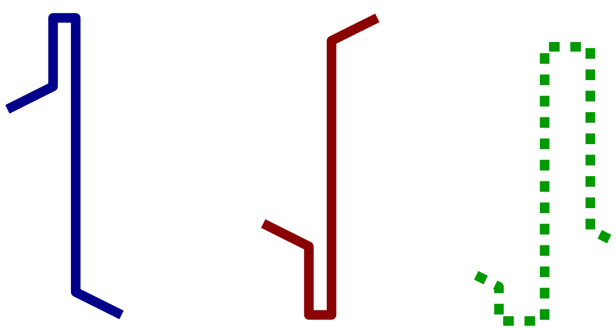
Graph 1: x-coordinates

In R: All segments vertical or slanted of y-length 1.

Graph 2: y-coordinates

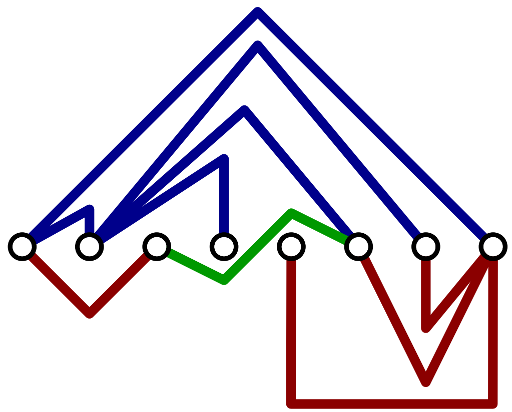


Edges:



4 4 6

2-Page Book Embed. × 2-Page Book Embed.

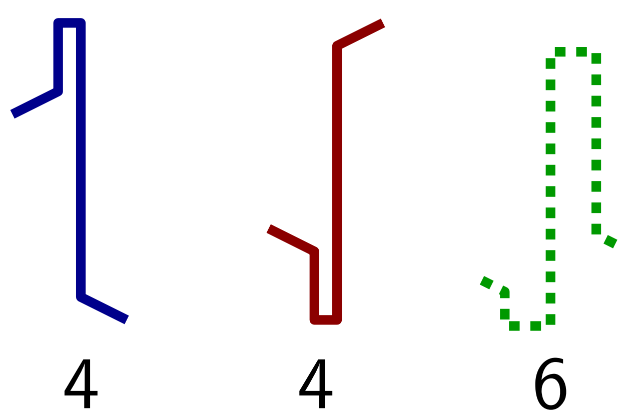


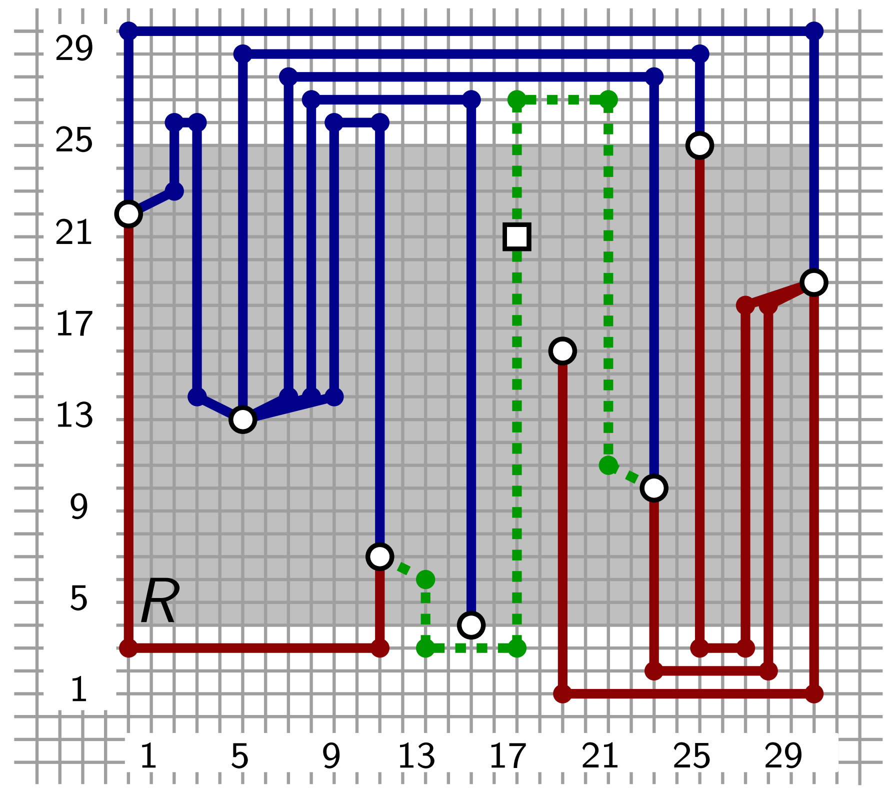
Graph 1: x-coordinates

In R: All segments vertical or slanted of y-length 1.

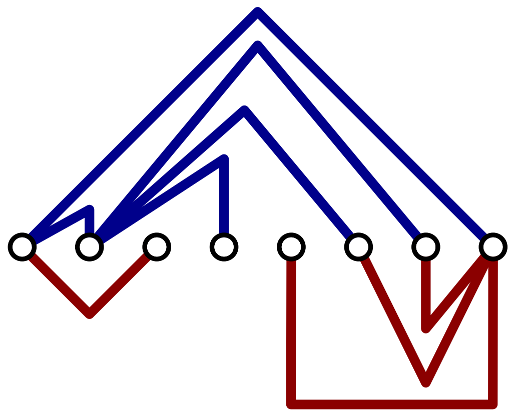
Graph 2: y-coordinates

Edges:





2-Page Book Embed. × 2-Page Book Embed.

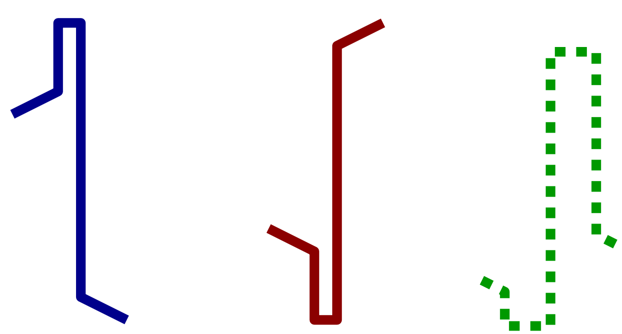


Graph 1: x-coordinates

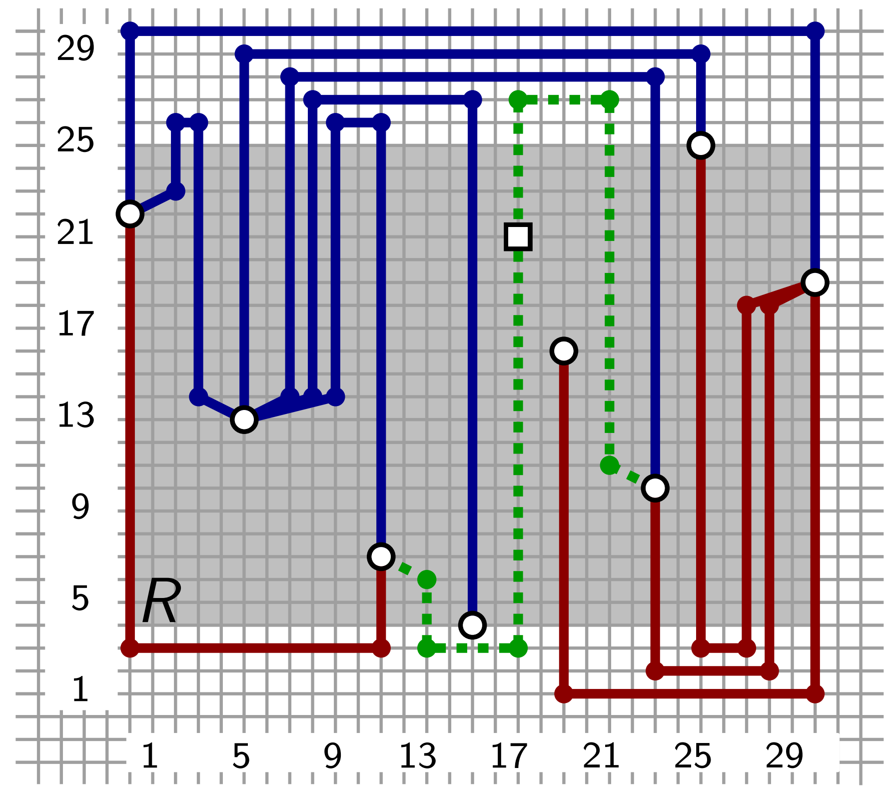
In R: All segments vertical or slanted of y-length 1.

Graph 2: y-coordinates

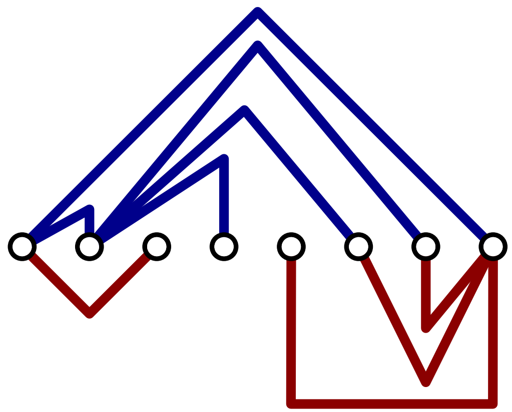
Edges:



4 4 6



2-Page Book Embed. × 2-Page Book Embed.

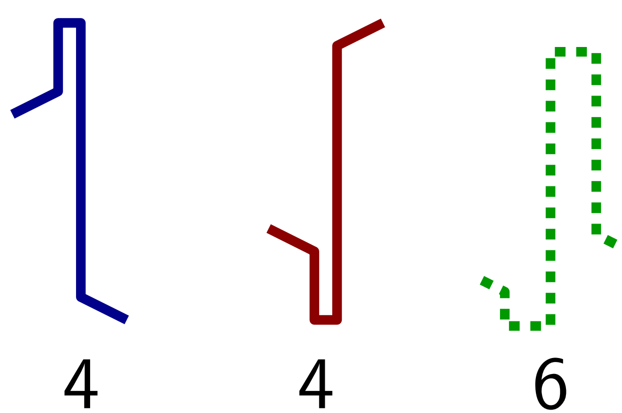


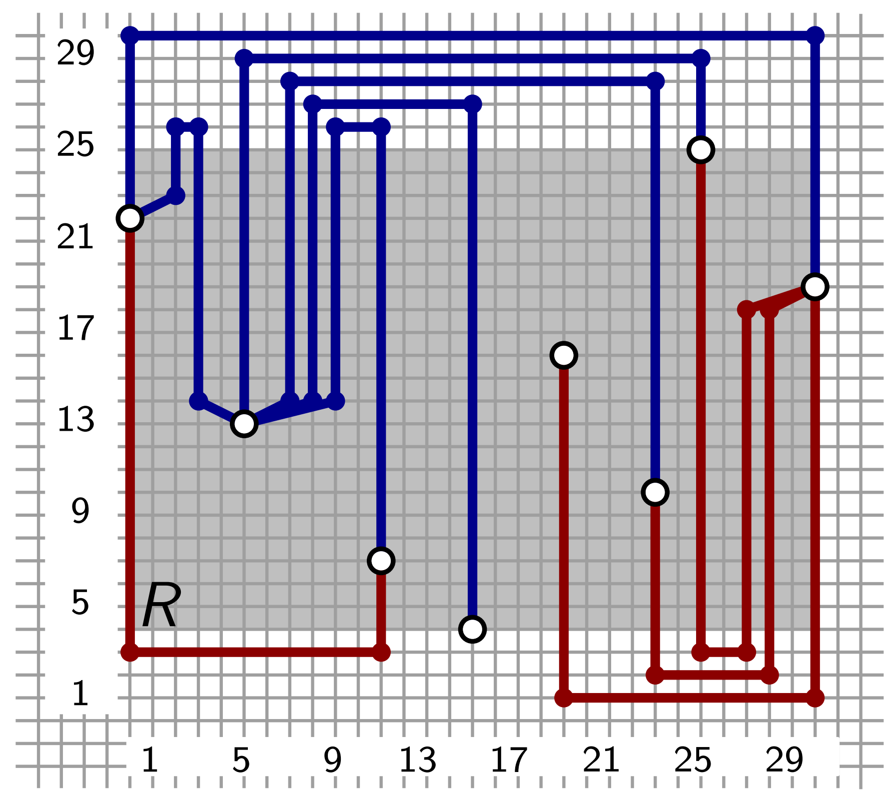
Graph 1: *x*-coordinates

In *R*: All segments vertical or slanted of *y*-length 1.

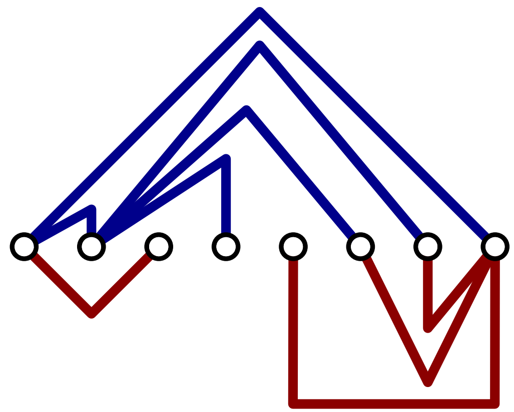
Graph 2: *y*-coordinates

Edges:





2-Page Book Embed. × 2-Page Book Embed.

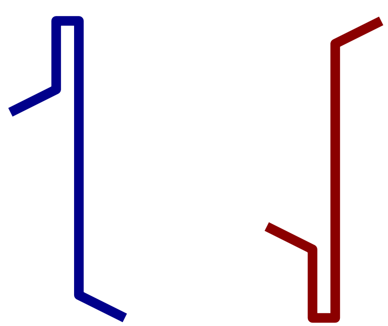


Graph 1: *x*-coordinates

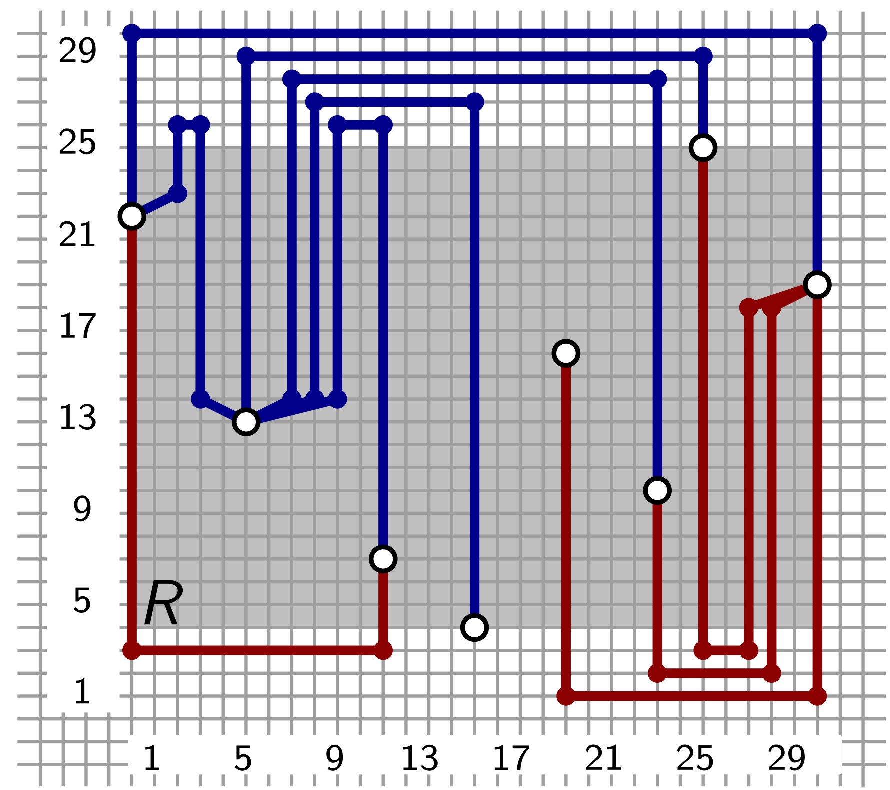
In *R*: All segments vertical or slanted of *y*-length 1.

Graph 2: *y*-coordinates

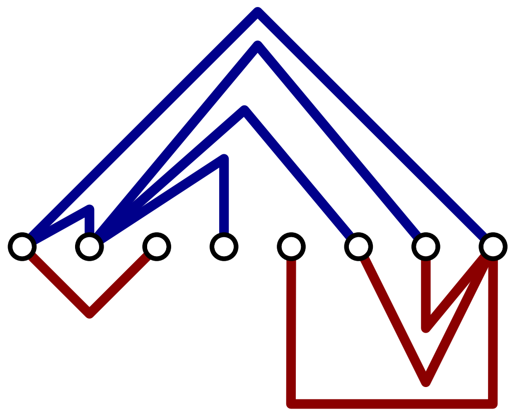
Edges:



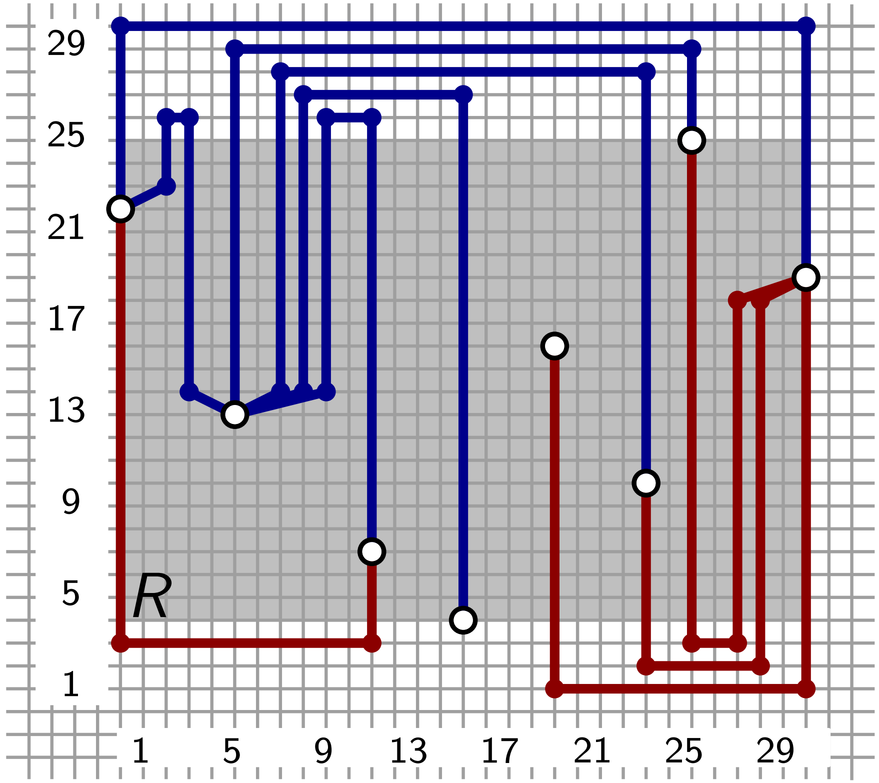
4 4



2-Page Book Embed. × 2-Page Book Embed.



Bends: 4 × 4
Grid size: $(11n - 32)^2$

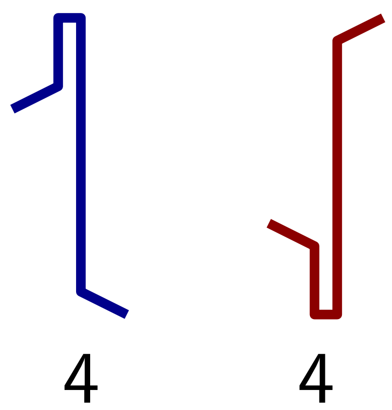


Graph 1: *x*-coordinates

In *R*: All segments vertical or slanted of *y*-length 1.

Graph 2: *y*-coordinates

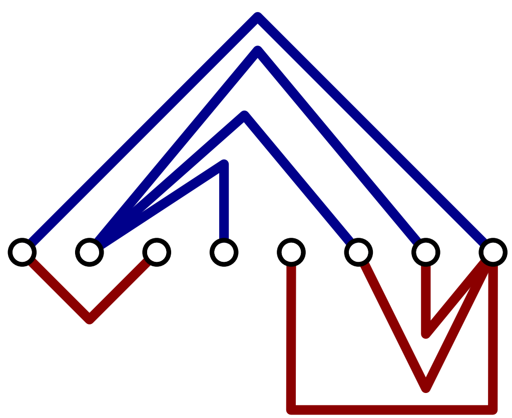
Edges:



~~2~~ 1-Page Book Embed. × ~~2~~ 1-Page Book Embed.

Bends: 4 × 4
Grid size: $(11n - 32)^2$

?

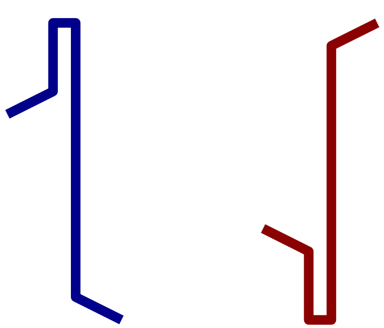


Graph 1: x-coordinates

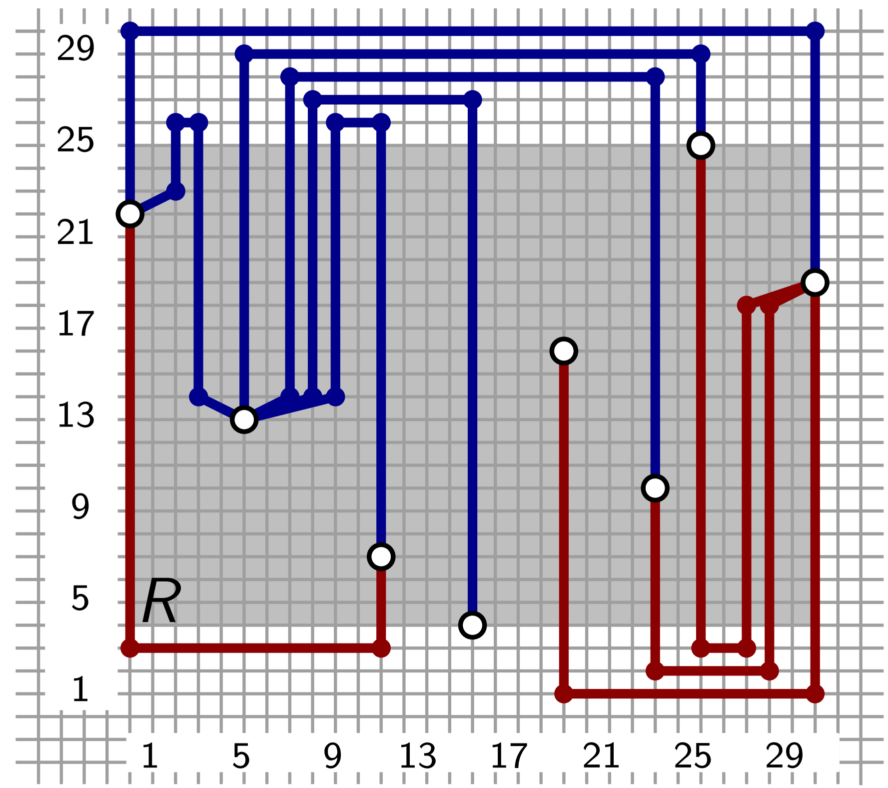
In R: All segments vertical or slanted of y-length 1.

Graph 2: y-coordinates

Edges:



4 4



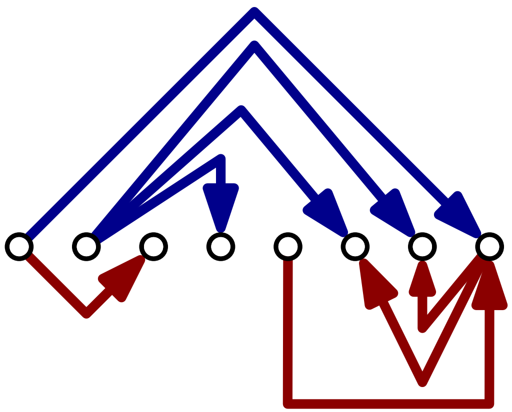
Outerplanar × Outerplanar

Bends: 3×3
Grid size: $(7n - 10)^2$

Decompose into two forests. . .

[Nash-Williams '64]

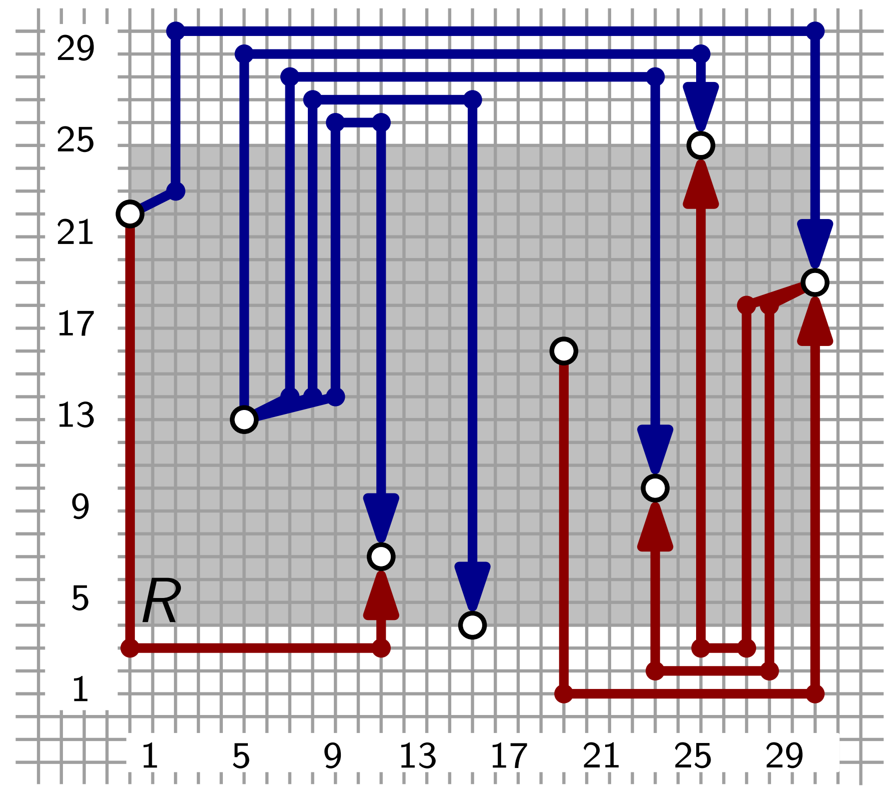
and direct them!



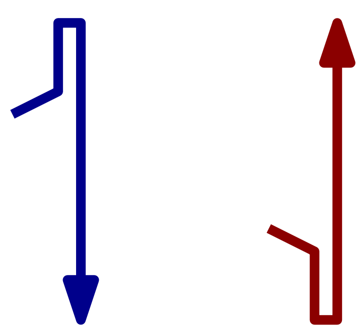
Graph 1: x-coordinates

In R: All segments vertical or slanted of y-length 1.

Graph 2: y-coordinates



Edges:



3 3

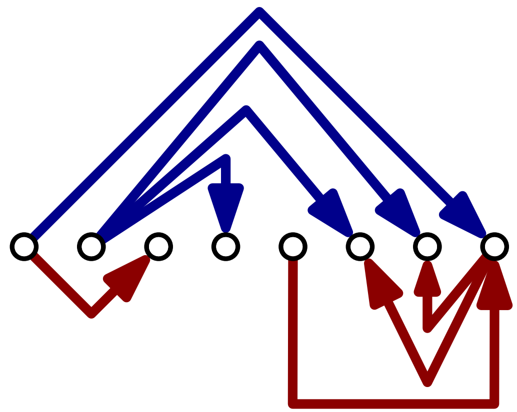
Outerplanar × Outerplanar

Bends: 3×3
Grid size: $(7n - 10)^2$

Decompose into two forests. . .

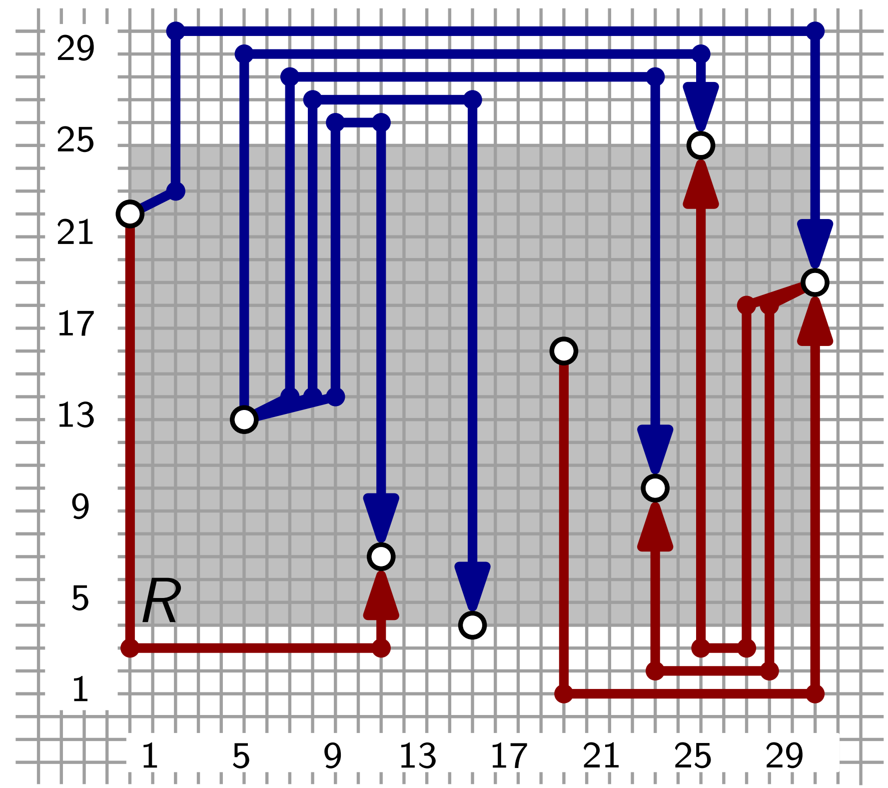
[Nash-Williams '64]

and direct them!



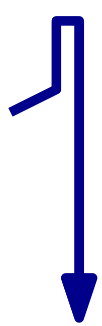
Graph 1: x-coordinates

In R: All segments vertical or slanted of y-length 1.

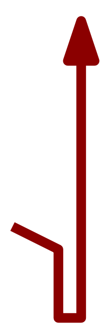


Graph 2: y-coordinates

Edges:



3



3

Every vertex has ≤ 1 incoming edge from above and ≤ 1 from below.

Conclusions

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	✓

Conclusions

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

- Reduce bend numbers!

Open Problems

- Reduce bend numbers!
- Relax constraints on crossing resolution
  $\Rightarrow$ LacSim
  (*Simultaneous Embedding with Large-Angle Crossings*)

Open Problems

- Reduce bend numbers!
- Relax constraints on crossing resolution
  $\Rightarrow$ LacSim
  (*Simultaneous Embedding with Large-Angle Crossings*)
- Computational complexity:
  Given n points and two n-vertex planar graphs,
  do they admit a RacSim drawing with $\leq k$ bends per edge?

Open Problems

- Reduce bend numbers!
- Relax constraints on crossing resolution
  $\Rightarrow$ LacSim
  (*Simultaneous Embedding with Large-Angle Crossings*)
- Computational complexity:
  Given n points and two n-vertex planar graphs,
  do they admit a RacSim drawing with $\leq k$ bends per edge?
- SimRac instead of RacSim:
  Draw each graph Rac (but ignore crossing angles).

Open Problems

- Reduce bend numbers!
- Relax constraints on crossing resolution
  $\Rightarrow$ LacSim
  (*Simultaneous Embedding with Large-Angle Crossings*)
- Computational complexity:
  Given *n* points and two *n*-vertex planar graphs,
  do they admit a RacSim drawing with $\leq k$ bends per edge?
- SimRac instead of RacSim:
  Draw each graph Rac (but ignore crossing angles).
- RacSefe instead of RacSim:
  Cross at right angles *and* fix common edges!

Open Problems

- Reduce bend numbers!
- Relax constraints on crossing resolution
  $\Rightarrow$ LacSim
  (*Simultaneous Embedding with Large-Angle Crossings*)
- Computational complexity:
  Given n points and two n-vertex planar graphs,
  do they admit a RacSim drawing with $\leq k$ bends per edge?
- SimRac instead of RacSim:
  Draw each graph Rac (but ignore crossing angles).
- RacSefe instead of RacSim:
  Cross at right angles *and* fix common edges!
- Can we do (const, const)-Sefe in polynomial area? [FHK'15]

Open Problems

- Reduce bend numbers!
- Relax constraints on crossing resolution
  $\Rightarrow$ LacSim
  (*Simultaneous Embedding with Large-Angle Crossings*)
- Computational complexity:
  Given *n* points and two *n*-vertex planar graphs,
  do they admit a RacSim drawing with $\leq k$ bends per edge?
- SimRac instead of RacSim:
  Draw each graph Rac (but ignore crossing angles).
- RacSefe instead of RacSim:
  Cross at right angles *and* fix common edges!
- Can we do (const, const)-Sefe in polynomial area? [FHK'15]

Our Results (Journal Version)

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

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