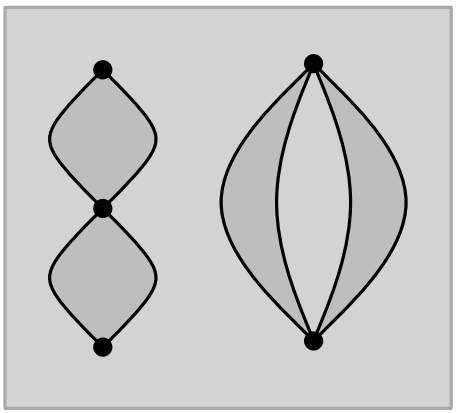
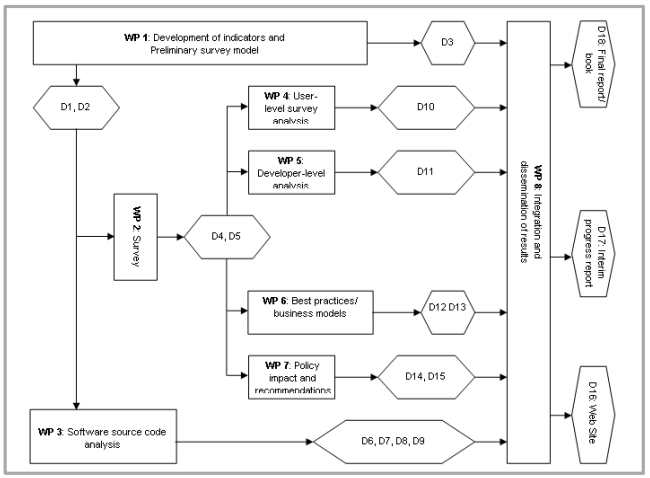
Trees and Series-Parallel Graphs

Lecture Graph Drawing Algorithms · 192.053

Martin Nöllenburg

10.04.2018



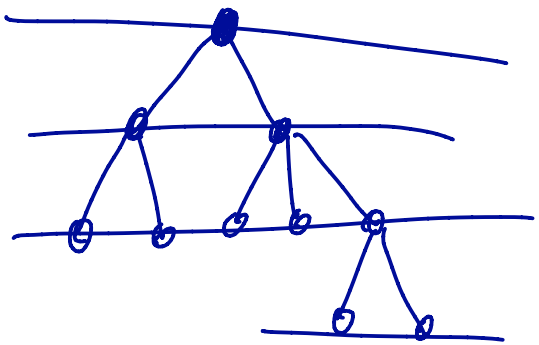


TU WIEN

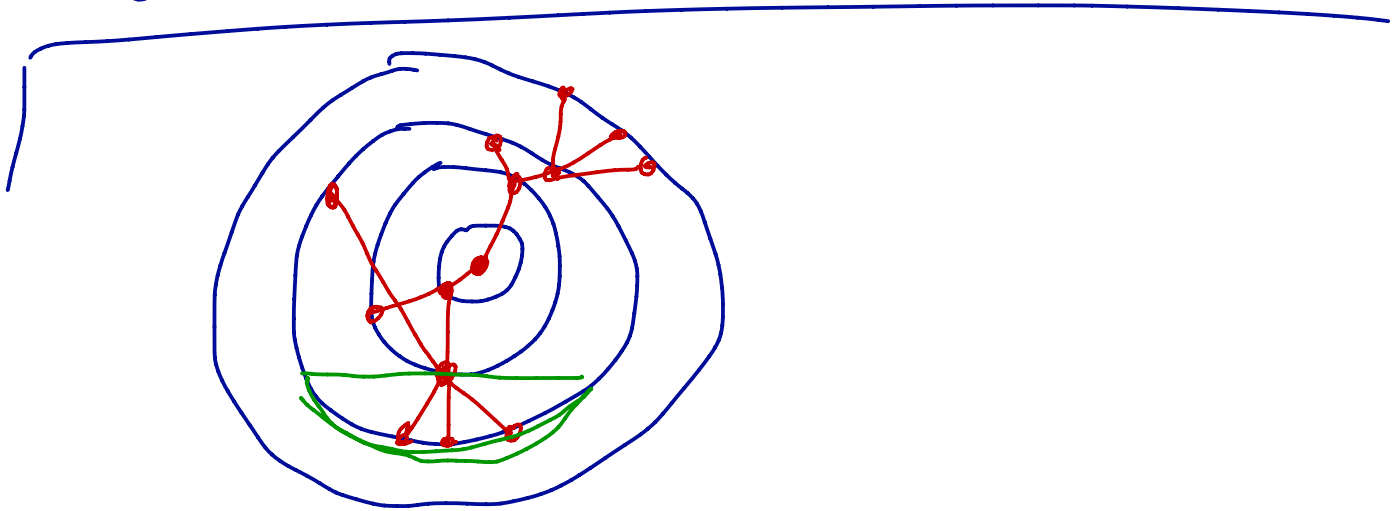
ALGORITHMS AND COMPLEXITY GROUP

Recall

What did we learn about tree drawings last time?



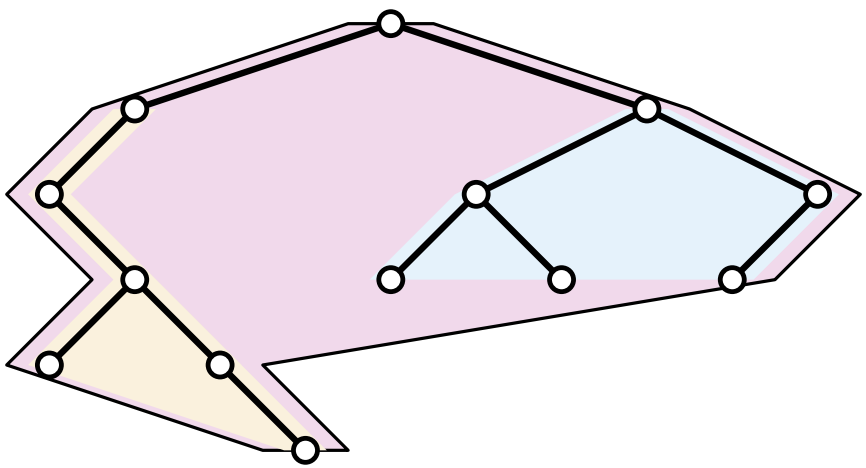
- symmetric placement
- small width

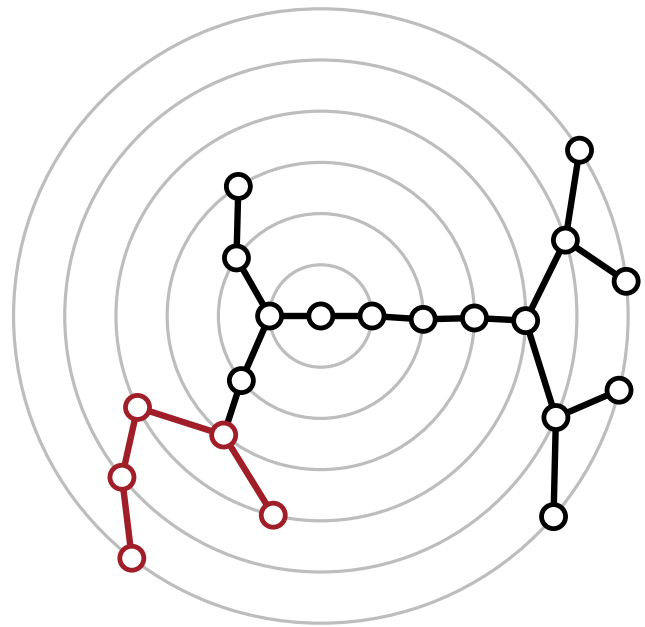


Recall

What did we learn about tree drawings last time?

- layered drawings: simple DFS-based algorithm
- improved contour-based 2-phase algorithm
- width minimization is generally NP-hard
- radial drawings (careful with crossings!)





Overview

Layered drawings of rooted trees

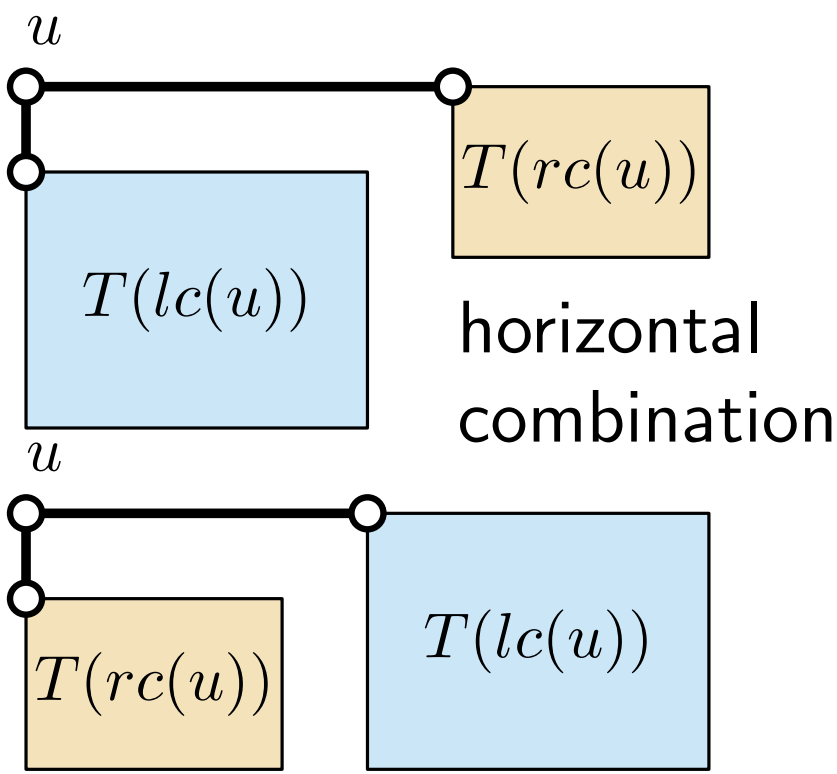
Radial tree drawings

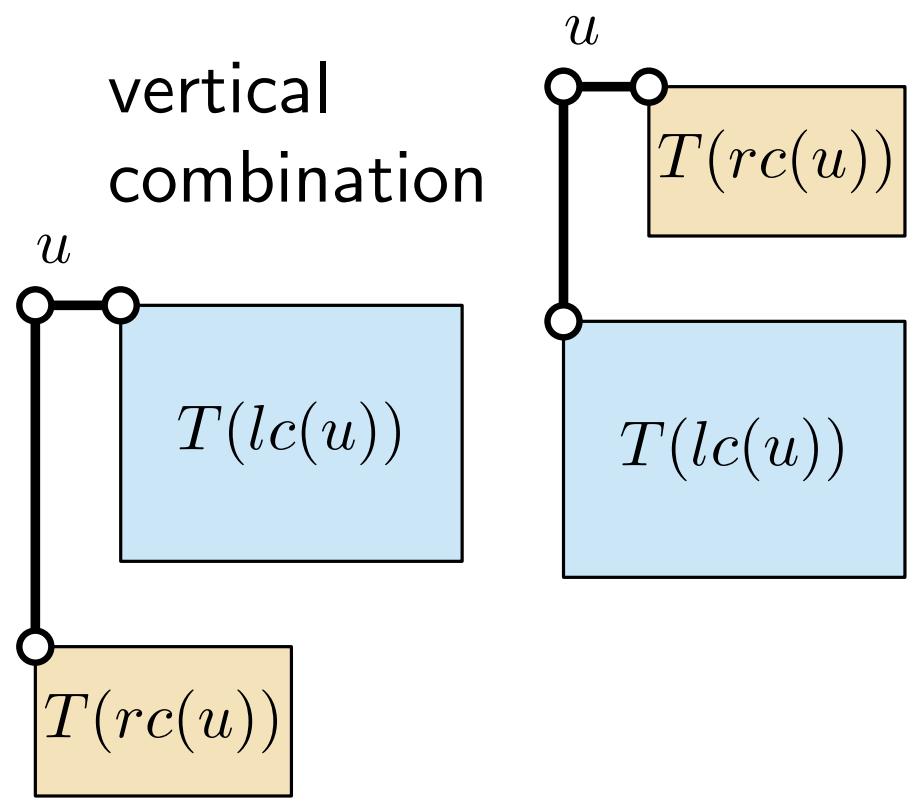
Compact orthogonal drawings of rooted trees

Compact Orthogonal Tree Drawings: HV Layout

Def: HV layout is a planar, straight-line grid drawing of a rooted binary tree with

- children either vertically aligned and below the parent or horizontally aligned and to the right
- recursively defined by placing bounding boxes of subtrees without intersections





Algorithm RightHeavyHVTreeDraw

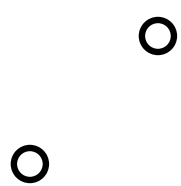
- Construct drawings of left and right subtrees recursively (postorder)

Algorithm RightHeavyHVTreeDraw

- Construct drawings of left and right subtrees recursively (postorder)
- Place heavier subtree as horizontal combination right of lighter subtree

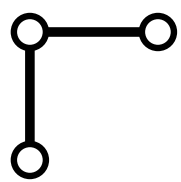
Algorithm RightHeavyHVTreeDraw

- Construct drawings of left and right subtrees recursively (postorder)
- Place heavier subtree as horizontal combination right of lighter subtree



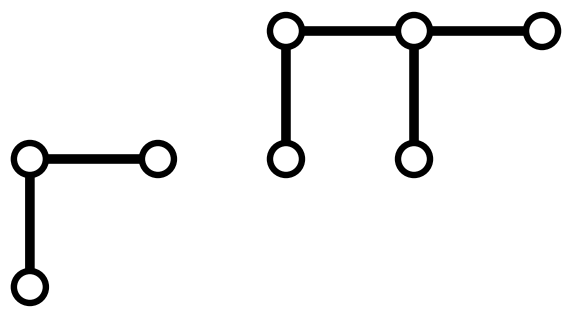
Algorithm RightHeavyHVTreeDraw

- Construct drawings of left and right subtrees recursively (postorder)
- Place heavier subtree as horizontal combination right of lighter subtree



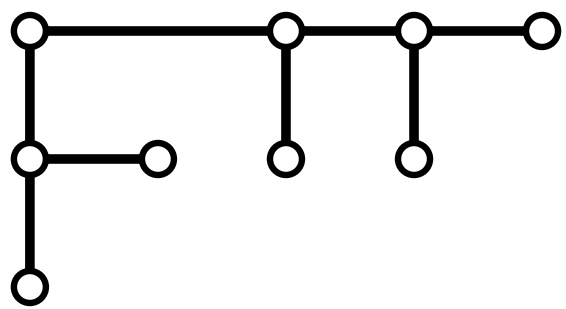
Algorithm RightHeavyHVTreeDraw

- Construct drawings of left and right subtrees recursively (postorder)
- Place heavier subtree as horizontal combination right of lighter subtree



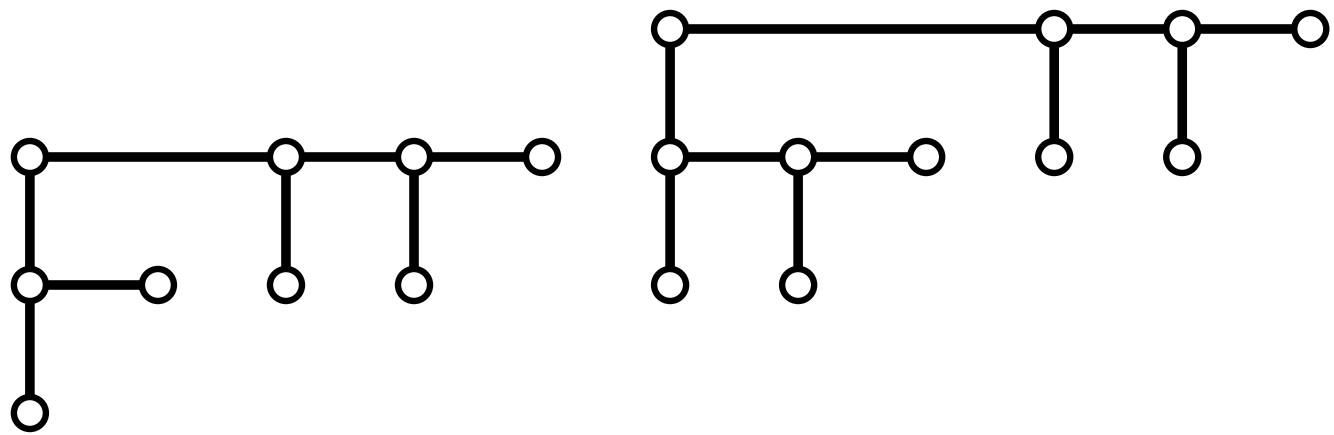
Algorithm RightHeavyHVTreeDraw

- Construct drawings of left and right subtrees recursively (postorder)
- Place heavier subtree as horizontal combination right of lighter subtree



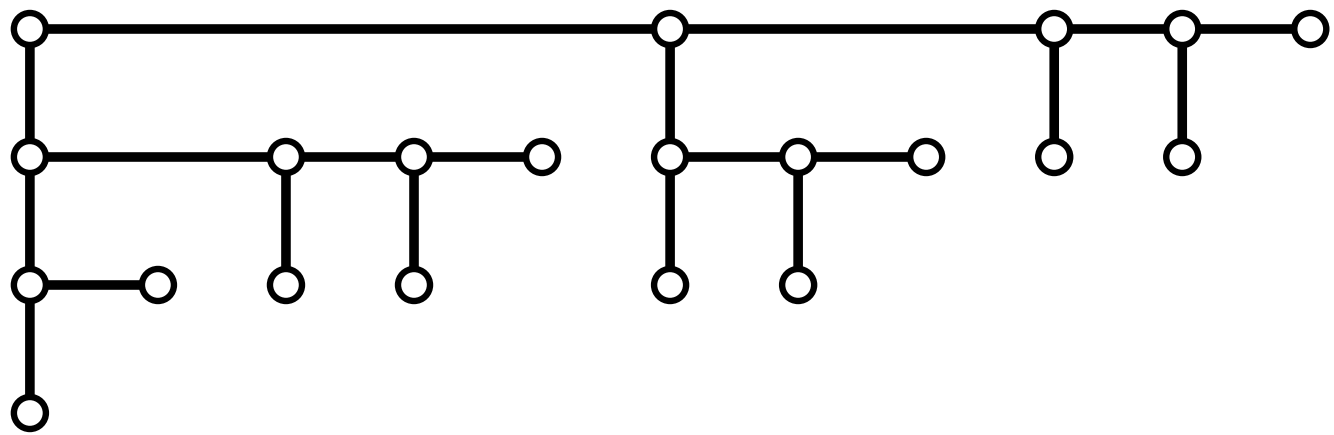
Algorithm RightHeavyHVTreeDraw

- Construct drawings of left and right subtrees recursively (postorder)
- Place heavier subtree as horizontal combination right of lighter subtree



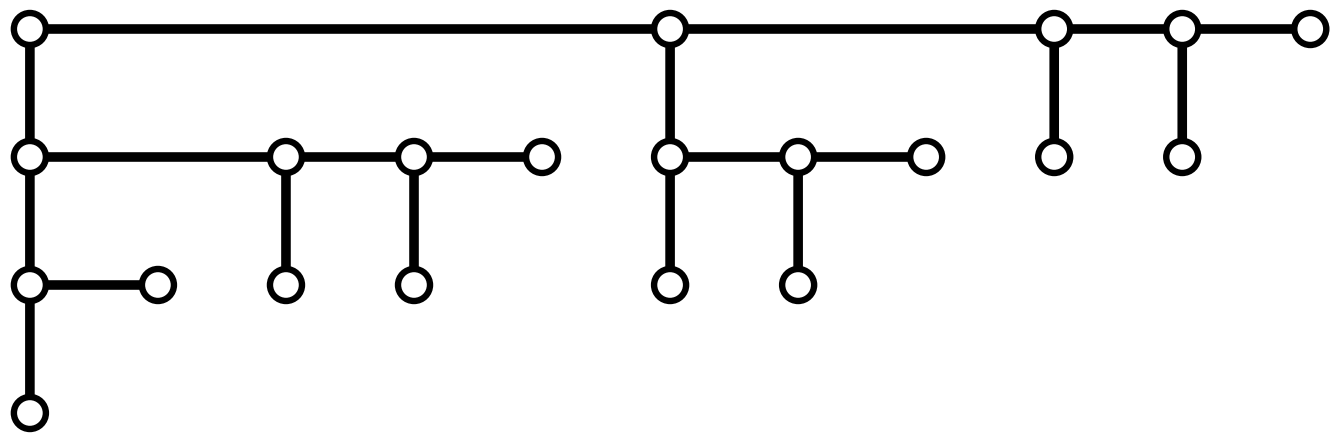
Algorithm RightHeavyHVTreeDraw

- Construct drawings of left and right subtrees recursively (postorder)
- Place heavier subtree as horizontal combination right of lighter subtree



Algorithm RightHeavyHVTreeDraw

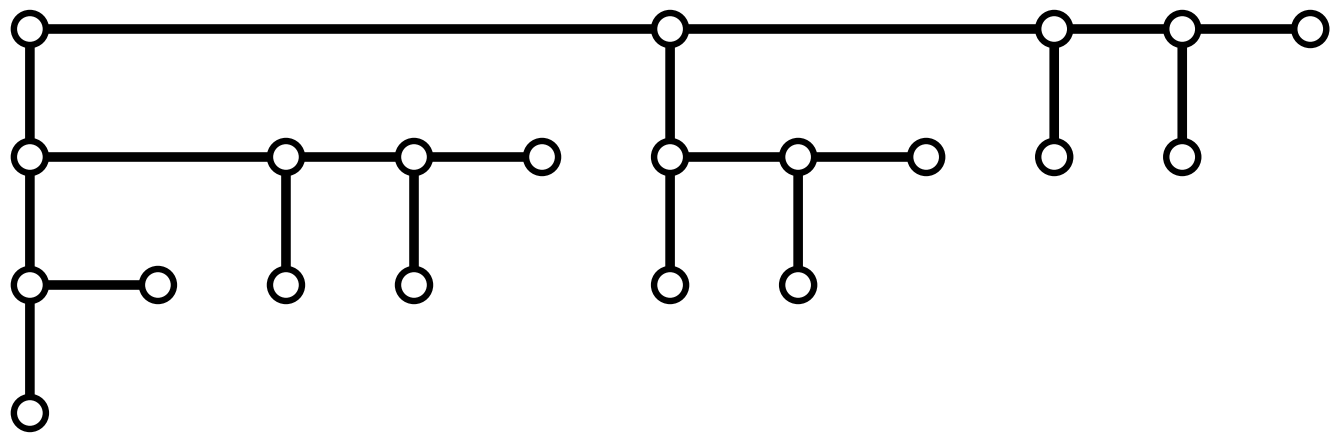
- Construct drawings of left and right subtrees recursively (postorder)
- Place heavier subtree as horizontal combination right of lighter subtree



Theorem: Drawing has width $\leq$

Algorithm RightHeavyHVTreeDraw

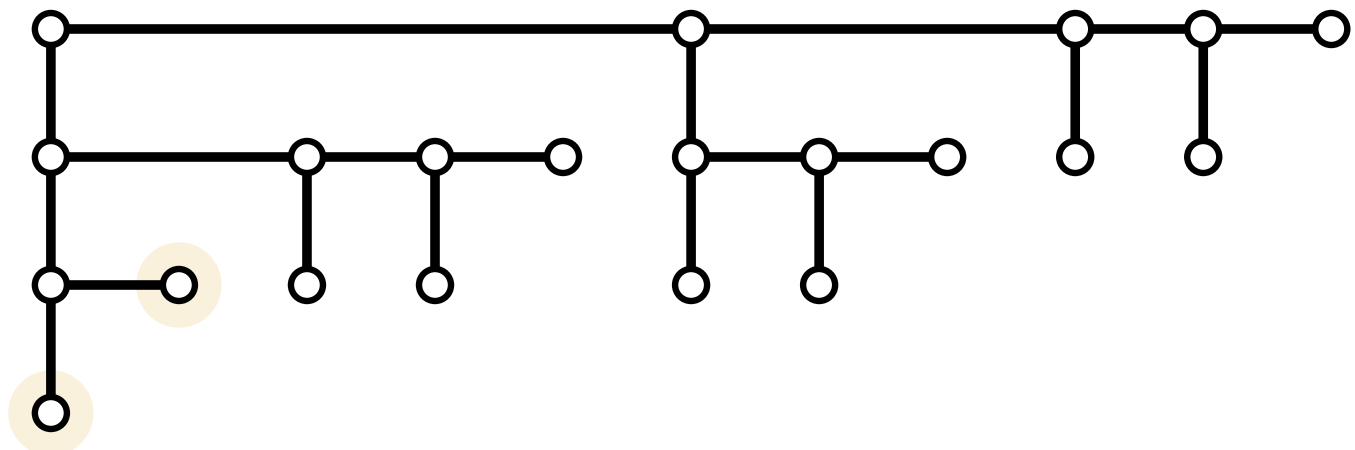
- Construct drawings of left and right subtrees recursively (postorder)
- Place heavier subtree as horizontal combination right of lighter subtree



Theorem: Drawing has width $\leq n$ and height $\leq$

Algorithm RightHeavyHVTreeDraw

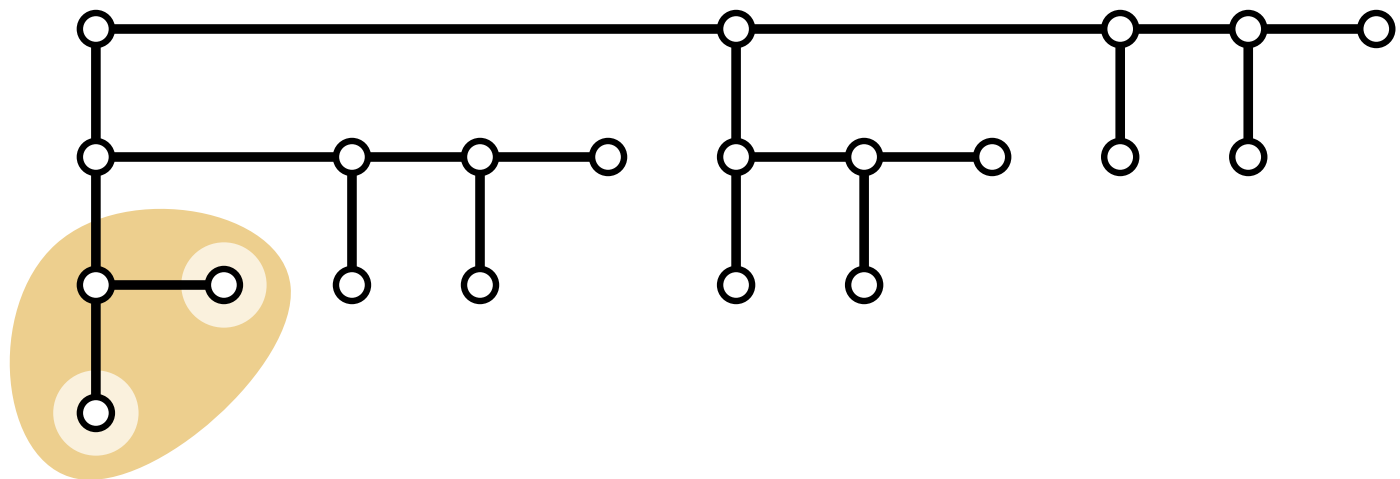
- Construct drawings of left and right subtrees recursively (postorder)
- Place heavier subtree as horizontal combination right of lighter subtree



Theorem: Drawing has width $\leq n$ and height $\leq$

Algorithm RightHeavyHVTreeDraw

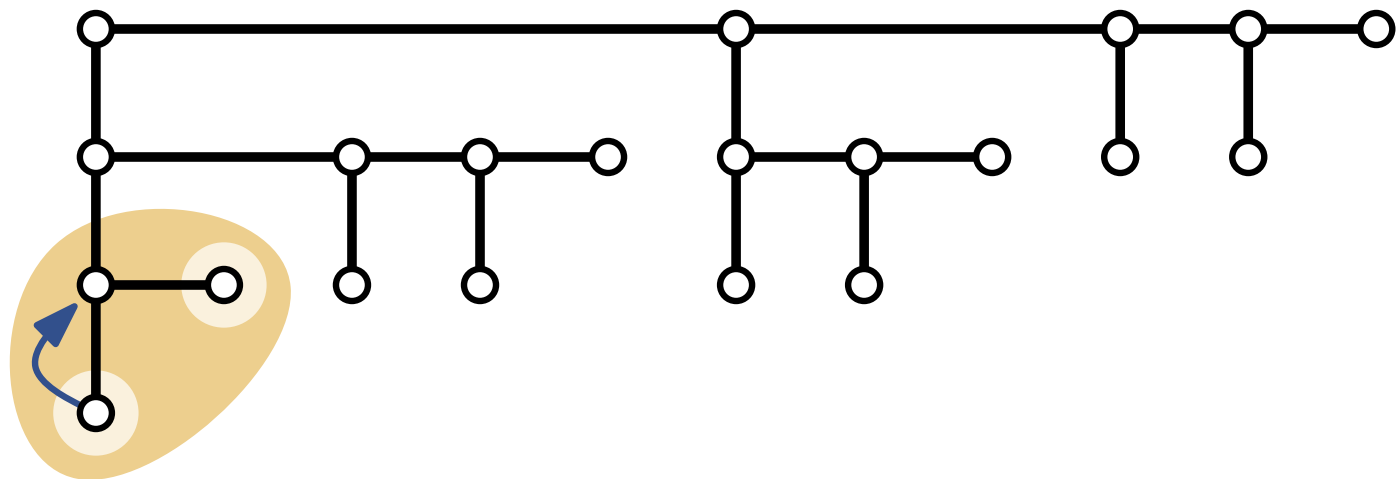
- Construct drawings of left and right subtrees recursively (postorder)
- Place heavier subtree as horizontal combination right of lighter subtree



Theorem: Drawing has width $\leq n$ and height $\leq$

Algorithm RightHeavyHVTreeDraw

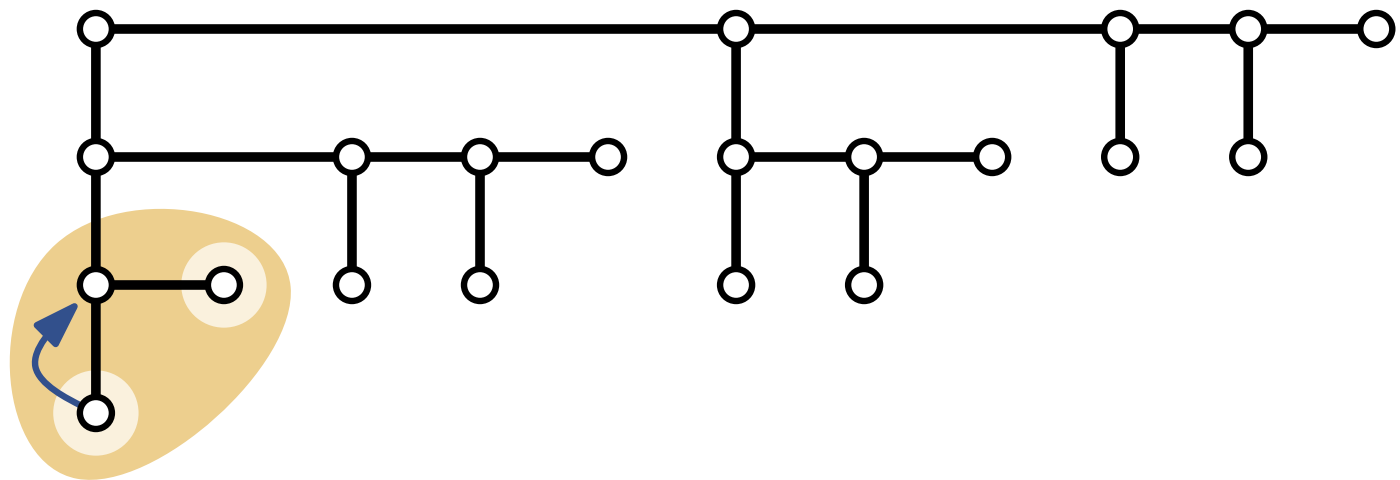
- Construct drawings of left and right subtrees recursively (postorder)
- Place heavier subtree as horizontal combination right of lighter subtree



Theorem: Drawing has width $\leq n$ and height $\leq$

Algorithm RightHeavyHVTreeDraw

- Construct drawings of left and right subtrees recursively (postorder)
- Place heavier subtree as horizontal combination right of lighter subtree



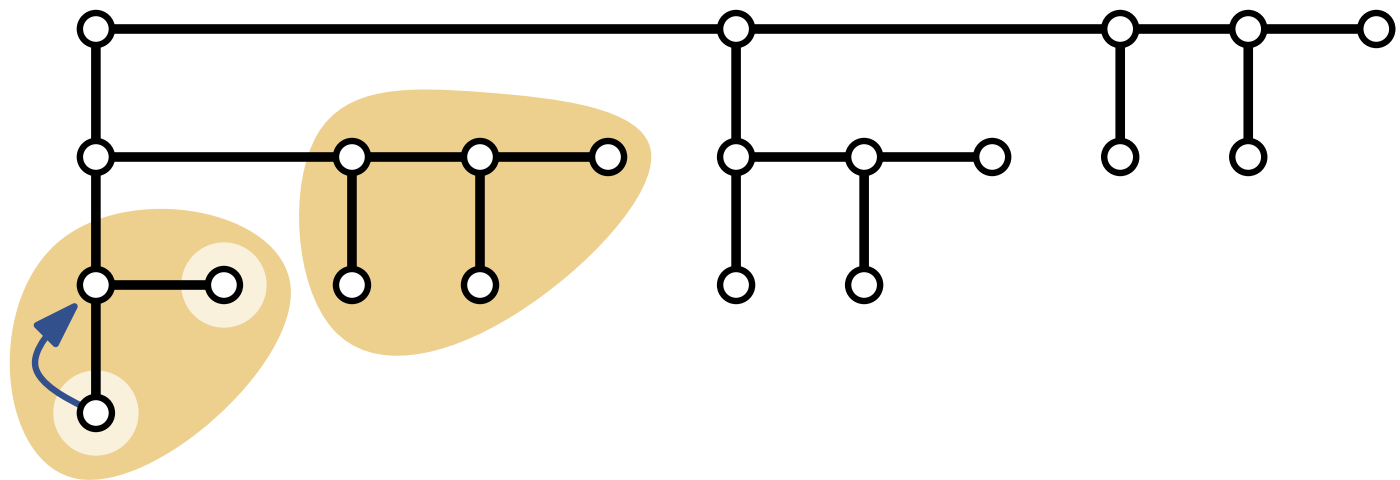
size at least doubles

Theorem: Drawing has width $\leq n$ and height $\leq$

Algorithm RightHeavyHVTreeDraw

- Construct drawings of left and right subtrees recursively (postorder)
- Place heavier subtree as horizontal combination right of lighter subtree

size at least doubles

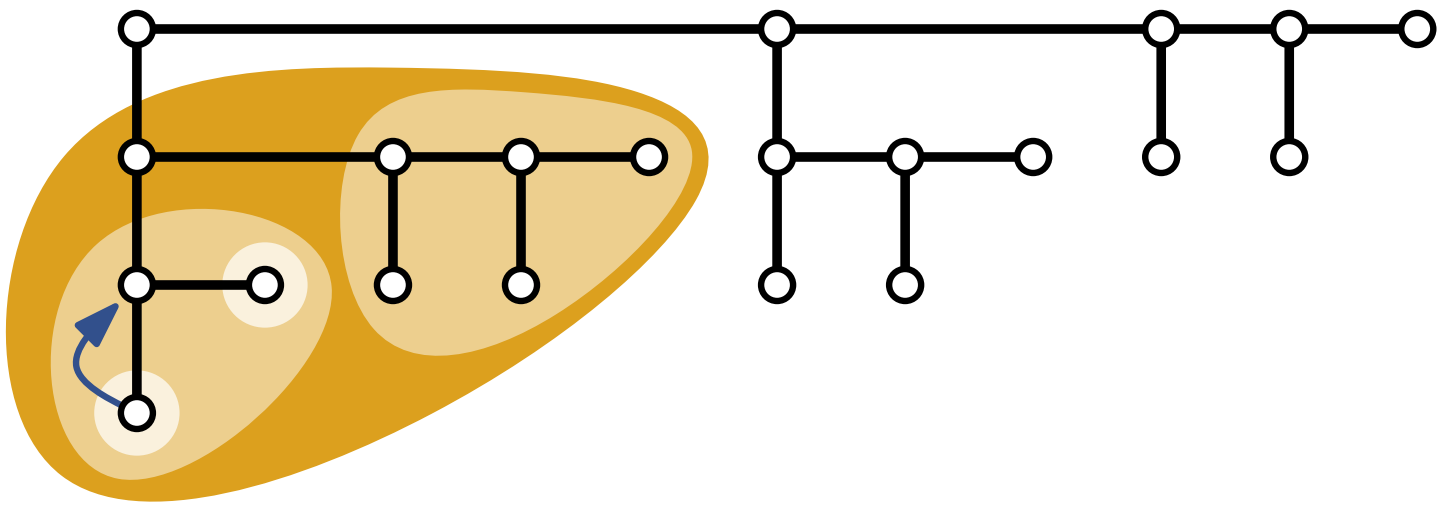


Theorem: Drawing has width $\leq n$ and height $\leq$

Algorithm RightHeavyHVTreeDraw

- Construct drawings of left and right subtrees recursively (postorder)
- Place heavier subtree as horizontal combination right of lighter subtree

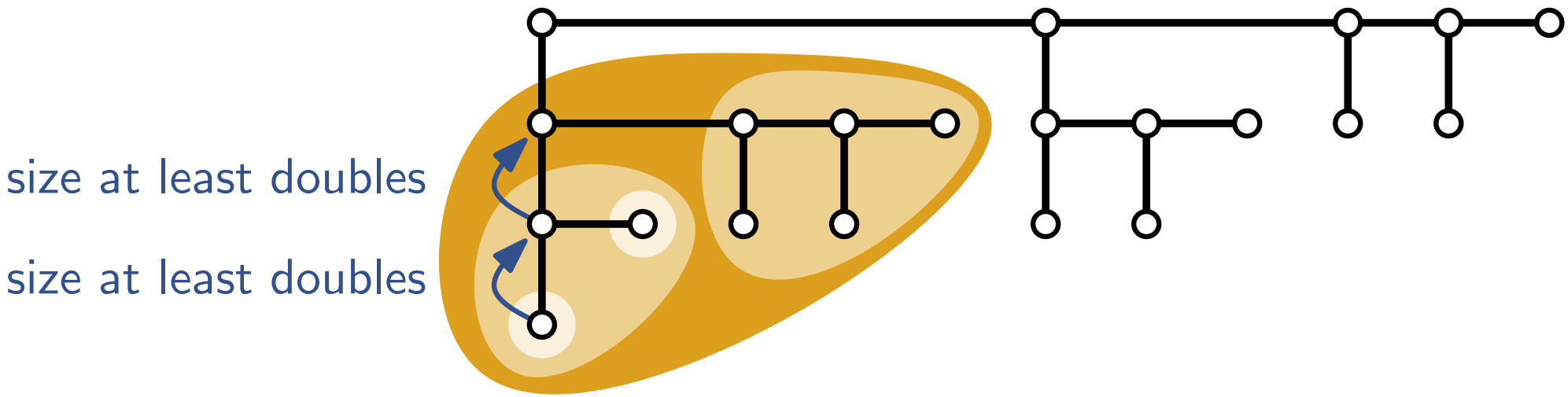
size at least doubles



Theorem: Drawing has width $\leq n$ and height $\leq$

Algorithm RightHeavyHVTreeDraw

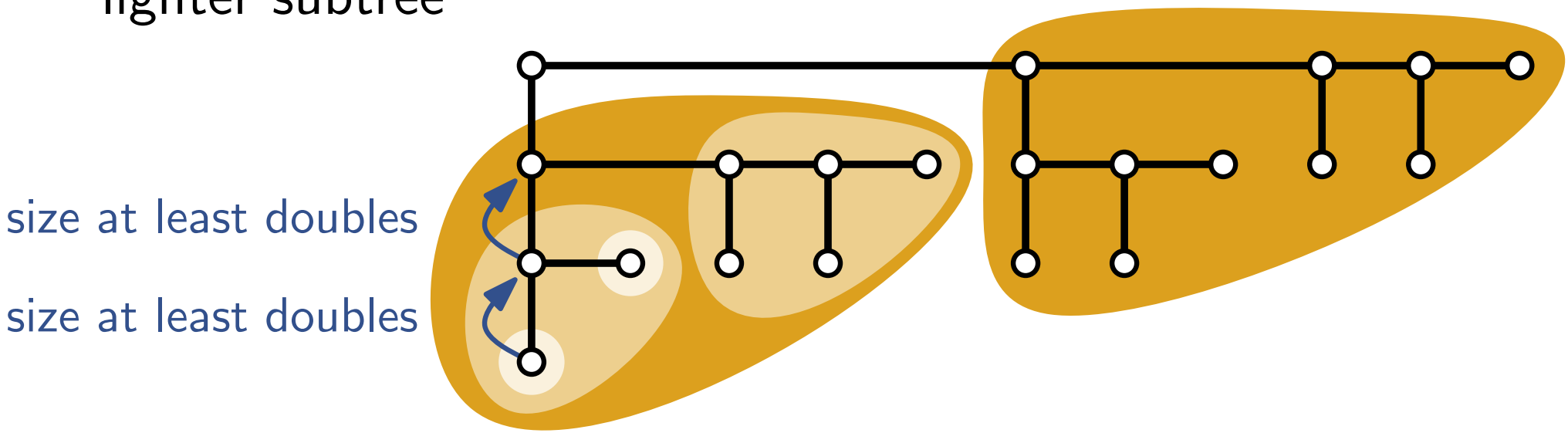
- Construct drawings of left and right subtrees recursively (postorder)
- Place heavier subtree as horizontal combination right of lighter subtree



Theorem: Drawing has width $\leq n$ and height $\leq$

Algorithm RightHeavyHVTreeDraw

- Construct drawings of left and right subtrees recursively (postorder)
- Place heavier subtree as horizontal combination right of lighter subtree



Theorem: Drawing has width $\leq n$ and height $\leq$

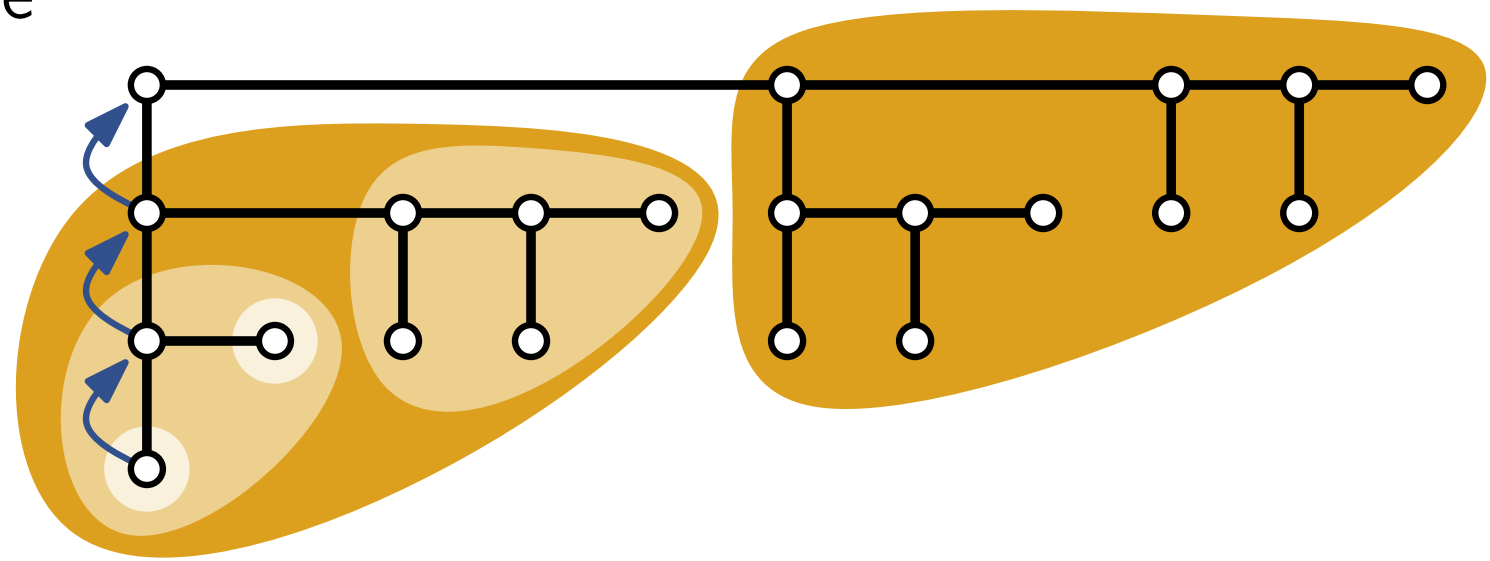
Algorithm RightHeavyHVTreeDraw

- Construct drawings of left and right subtrees recursively (postorder)
- Place heavier subtree as horizontal combination right of lighter subtree

size at least doubles

size at least doubles

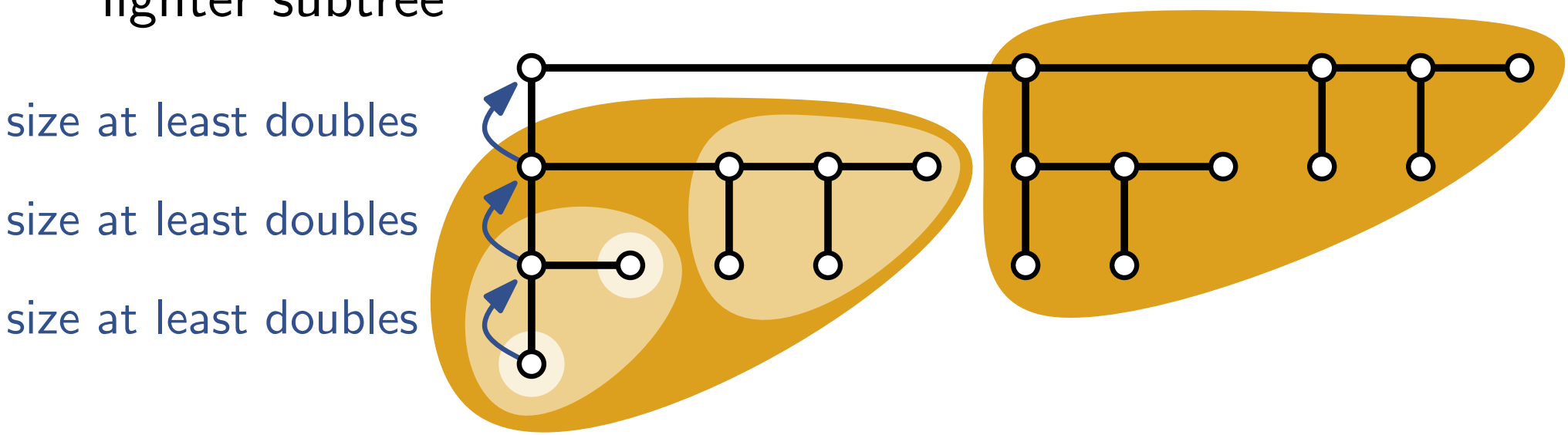
size at least doubles



Theorem: Drawing has width $\leq n$ and height $\leq$

Algorithm RightHeavyHVTreeDraw

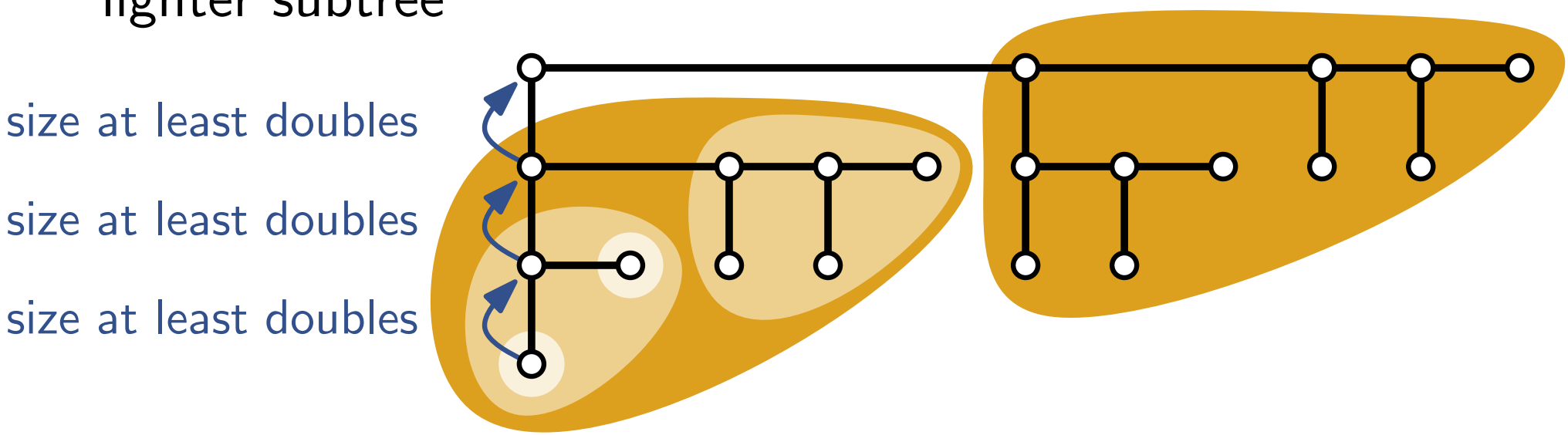
- Construct drawings of left and right subtrees recursively (postorder)
- Place heavier subtree as horizontal combination right of lighter subtree



Theorem: Drawing has width $\leq n$ and height $\leq \log_2 n$.

Algorithm RightHeavyHVTreeDraw

- Construct drawings of left and right subtrees recursively (postorder)
- Place heavier subtree as horizontal combination right of lighter subtree

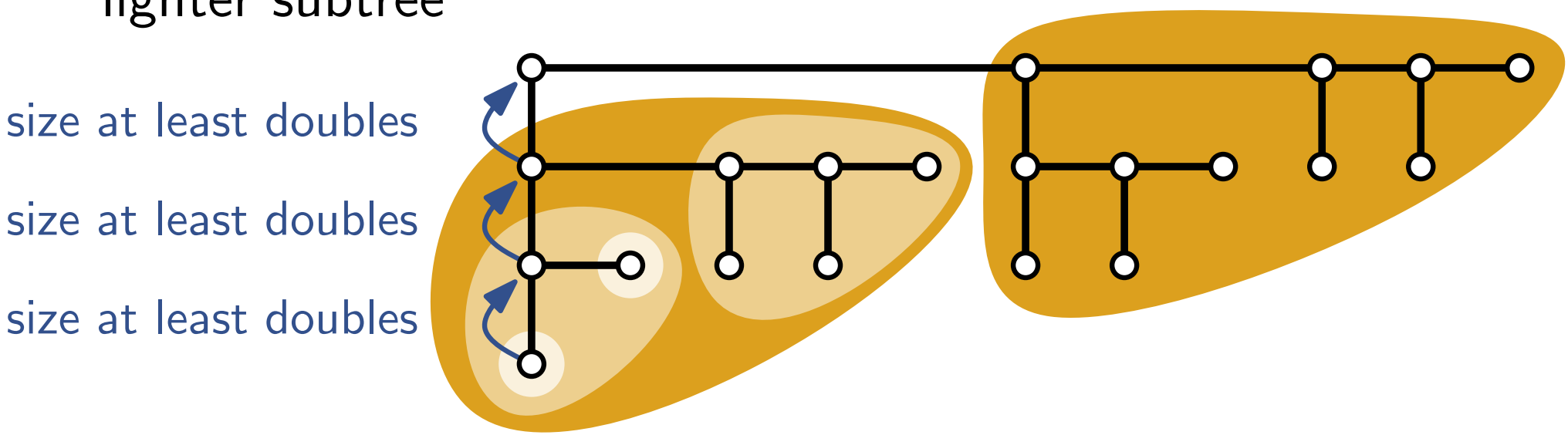


Theorem: Drawing has width $\leq n$ and height $\leq \log_2 n$.

In other words: A rooted binary tree can be drawn in $O(n)$ time on a grid of size $O(n \log n)$.

Algorithm RightHeavyHVTreeDraw

- Construct drawings of left and right subtrees recursively (postorder)
- Place heavier subtree as horizontal combination right of lighter subtree



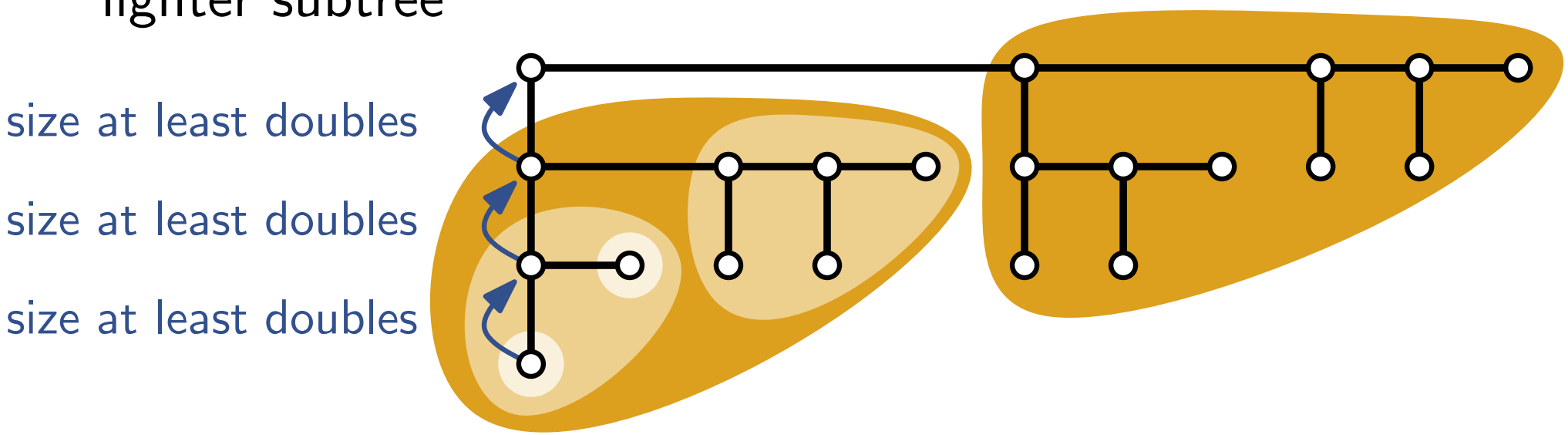
Theorem: Drawing has width $\leq n$ and height $\leq \log_2 n$.

In other words: A rooted binary tree can be drawn in $O(n)$ time on a grid of size $O(n \log n)$.

$\rightarrow$ using dynamic programming exact area minimization takes $O(n^2)$ time

Algorithm RightHeavyHVTreeDraw

- Construct drawings of left and right subtrees recursively (postorder)
- Place heavier subtree as horizontal combination right of lighter subtree



Theorem: Drawing has width $\leq n$ and height $\leq \log_2 n$.

In other words: A rooted binary tree can be drawn in $O(n)$ time on a grid of size $O(n \log n)$.

$\rightarrow$ using dynamic programming exact area minimization takes $O(n^2)$ time

But: we might have changed child order

Recall: Layout Problem

Graph visualization problem

given: graph $G = (V, E)$
find: drawing Γ of G that

- complies with the given drawing conventions
- optimizes the given aesthetics
- satisfies the partial/local constraints

$\rightarrow$ often lead to NP-hard optimization problems!
$\rightarrow$ often several competing criteria

Recall: Layout Problem

Graph visualization problem

given: graph $G = (V, E)$
find: drawing Γ of G that

- complies with the given drawing conventions
- optimizes the given aesthetics
- satisfies the partial/local constraints

$\rightarrow$ often lead to NP-hard optimization problems!
$\rightarrow$ often several competing criteria

we have seen:

- (binary/ordered) trees
- planar, straight-line (grid) drawings
- optional: centered parent, layered, upward, ordered, isomorphic subtrees
- optimization: area
- NP-hard: width minimization
- competing: width vs. isomorphic subtrees, area vs. ordered

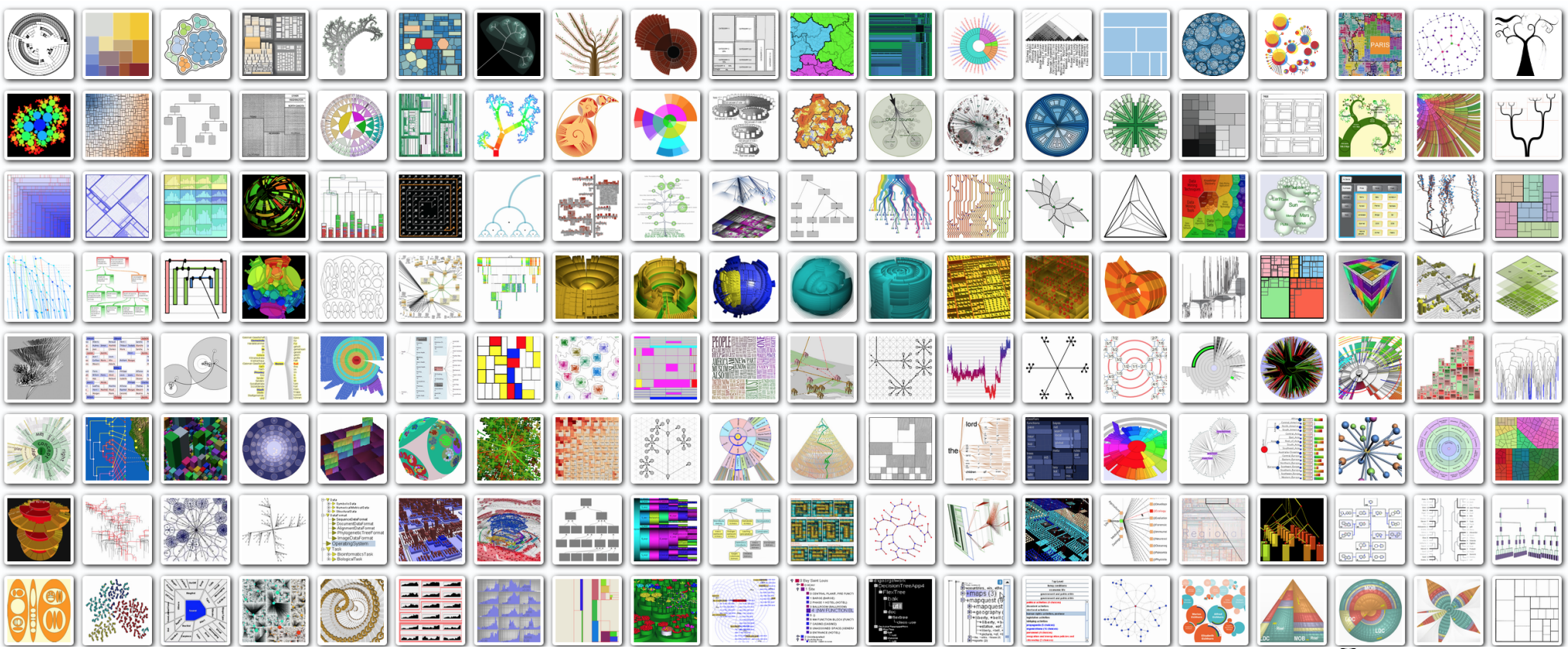
Summary and Outlook

- layered tree drawings
- radial tree drawings
- HV tree drawings

Summary and Outlook

- layered tree drawings
- radial tree drawings
- HV tree drawings

Trees are practically important graph class! Many layout algorithms exist and new ones are developed.



Further Reading: Area Minimization for Trees

Area minimization and area bounds of tree drawings in many flavors is still an active (theoretical) research topic.

Some examples:

- unordered, upward, binary: $O(n \log \log n)$ (1996)
- ordered, upward, binary: $O(n \log n)$ (2003)
- ordered, non-upward, orthogonal, binary: $O(nc^{\sqrt{\log n}})$ (2018)
- unordered, non-upward, binary: $\Theta(n)$ (2003)
- unordered, upward, general: $O(n\sqrt{\log n}(\log \log n)^k)$ (2018)
- ordered, upward, general: $O(nc^{\sqrt{\log n}})$ (1999)
- ...

see [Chan, 2018] for more

$\rightarrow$ unknown, whether most bounds are tight

Overview

Series-Parallel Graphs

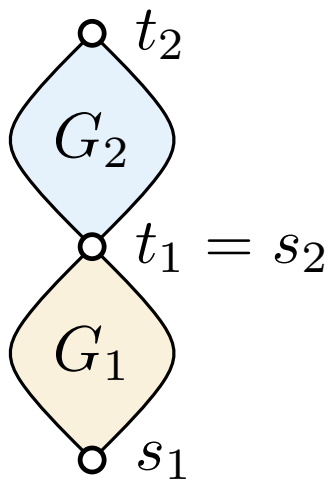
Series-Parallel Graphs

A simple (directed) graph G is **series-parallel**, if

- it consists of a single edge (s, t) with source s and sink t
- it is composed of two series-parallel graphs G_1, G_2 with sources s_1, s_2 and sinks t_1, t_2, which are combined using one of the following composition rules:

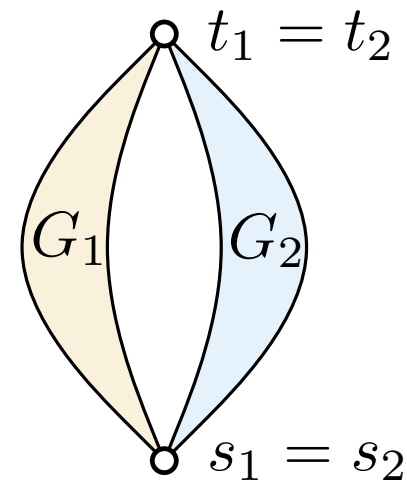
Series composition:

- identify t_1 and s_2,
- set s_1 as the source of G, t_2 as the sink of G



Parallel composition:

- identify s_1 and s_2 as new source
- identify t_1 and t_2 as new sink



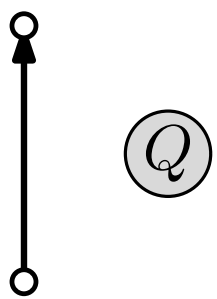
Decomposition Tree

A **decomposition tree** of G is a binary tree T with nodes of three types: S-nodes, P-nodes, and Q-nodes.

Decomposition Tree

A **decomposition tree** of G is a binary tree T with nodes of three types: S-nodes, P-nodes, and Q-nodes.

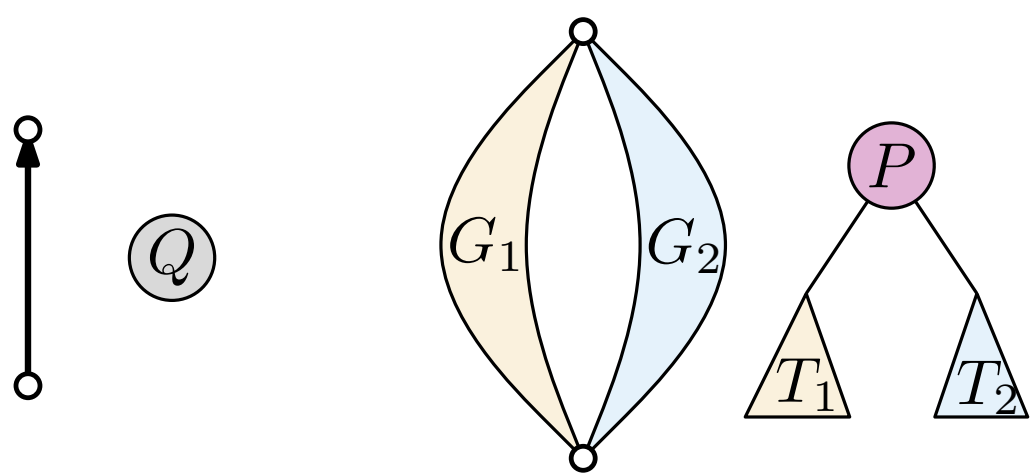
- If G is a **single edge**, then T is a single **Q-node**



Decomposition Tree

A **decomposition tree** of G is a binary tree T with nodes of three types: S-nodes, P-nodes, and Q-nodes.

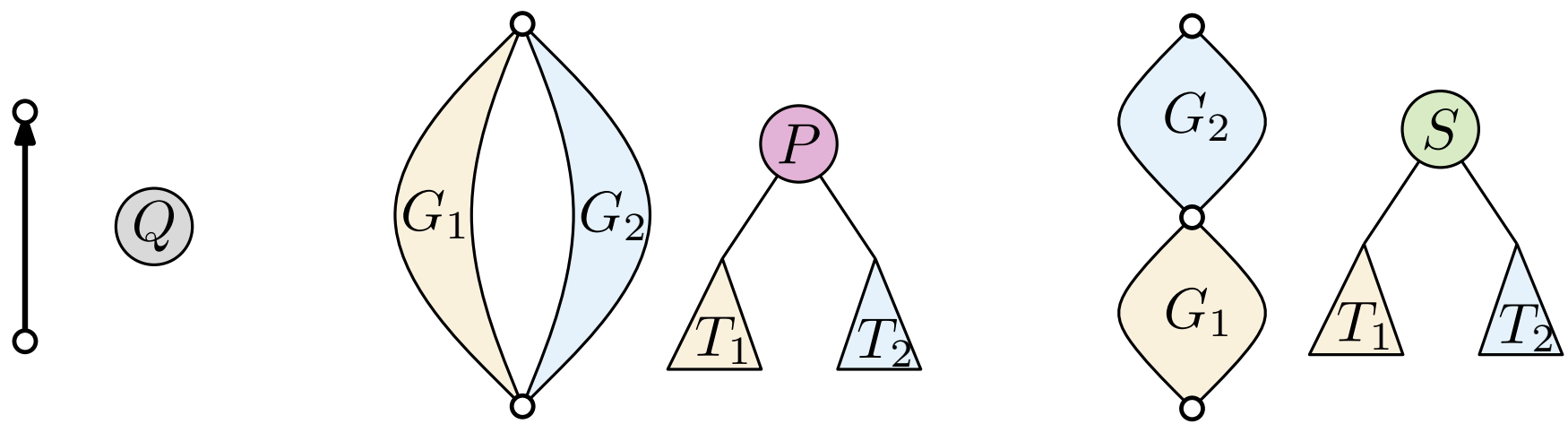
- If G is a **single edge**, then T is a single **Q-node**
- If G is a **parallel composition** of G_1 (with tree T_1) and G_2 (with tree T_2), then the root of T is a **P-node** with left and right subtrees T_1 and T_2



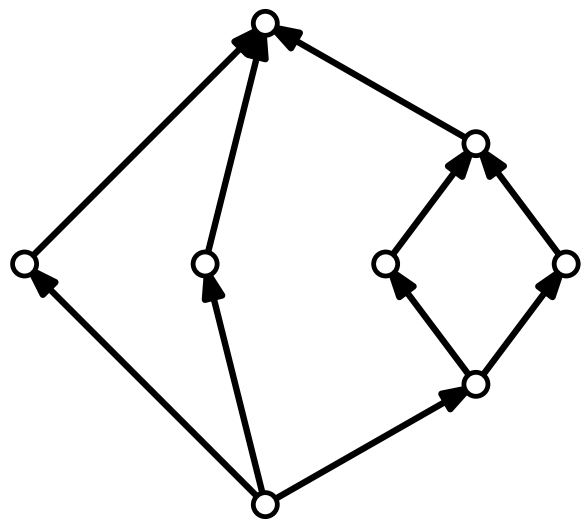
Decomposition Tree

A **decomposition tree** of G is a binary tree T with nodes of three types: S-nodes, P-nodes, and Q-nodes.

- If G is a **single edge**, then T is a single **Q-node**
- If G is a **parallel composition** of G_1 (with tree T_1) and G_2 (with tree T_2), then the root of T is a **P-node** with left and right subtrees T_1 and T_2
- If G is a **series composition** of G_1 (with tree T_1) and G_2 (with tree T_2), then the root of T is an **S-node** with left and right subtrees T_1 and T_2

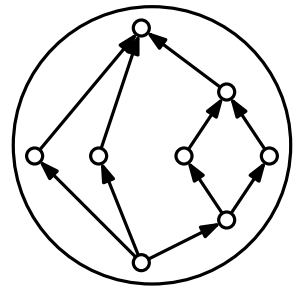


Decomposition Tree Example

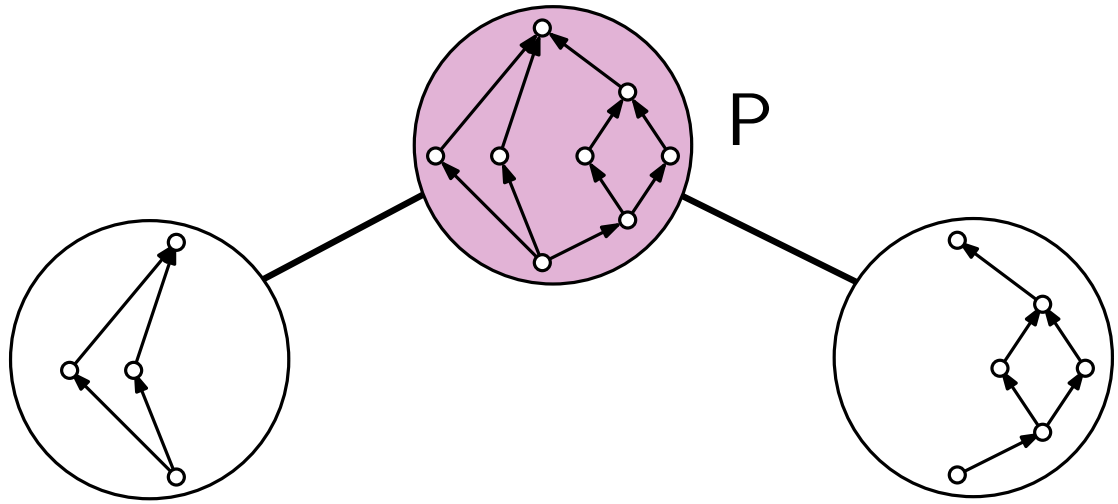


Is this a series-parallel graph?

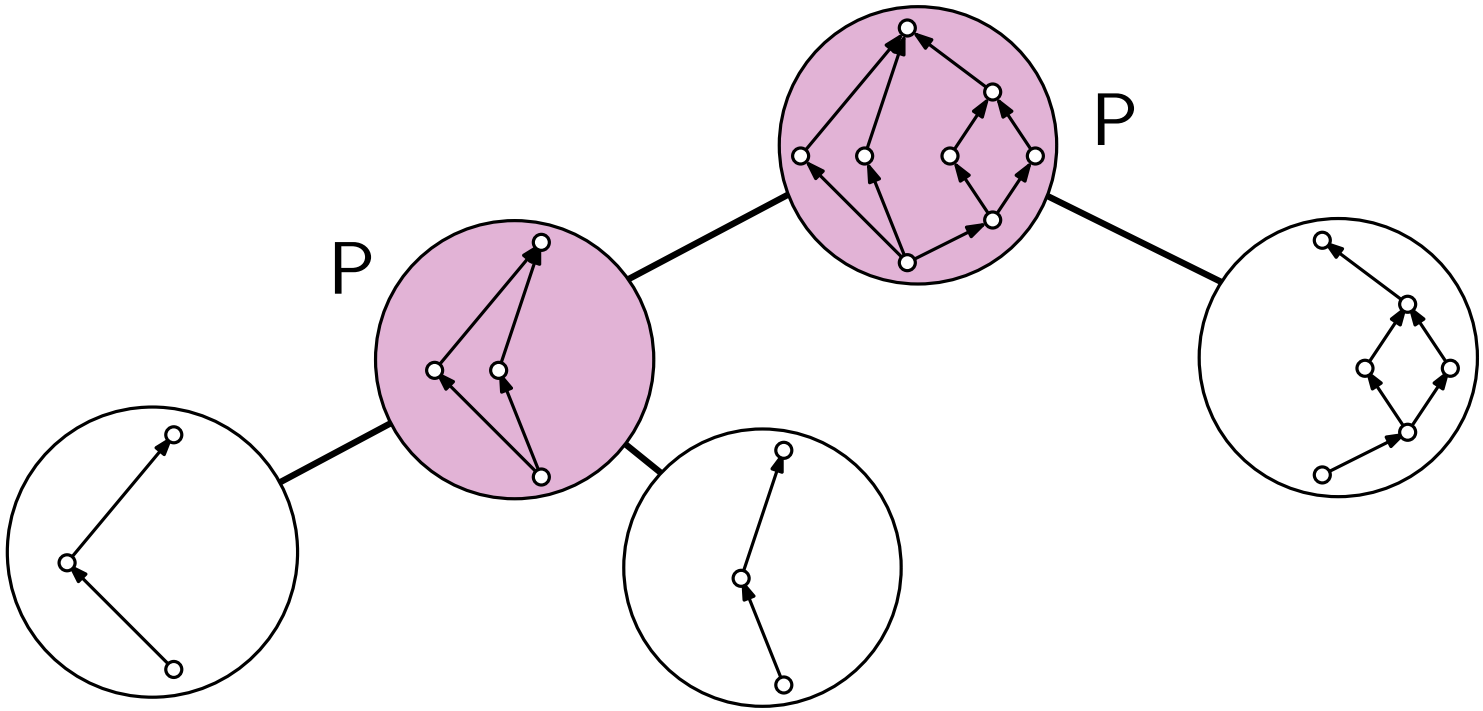
Decomposition Tree Example



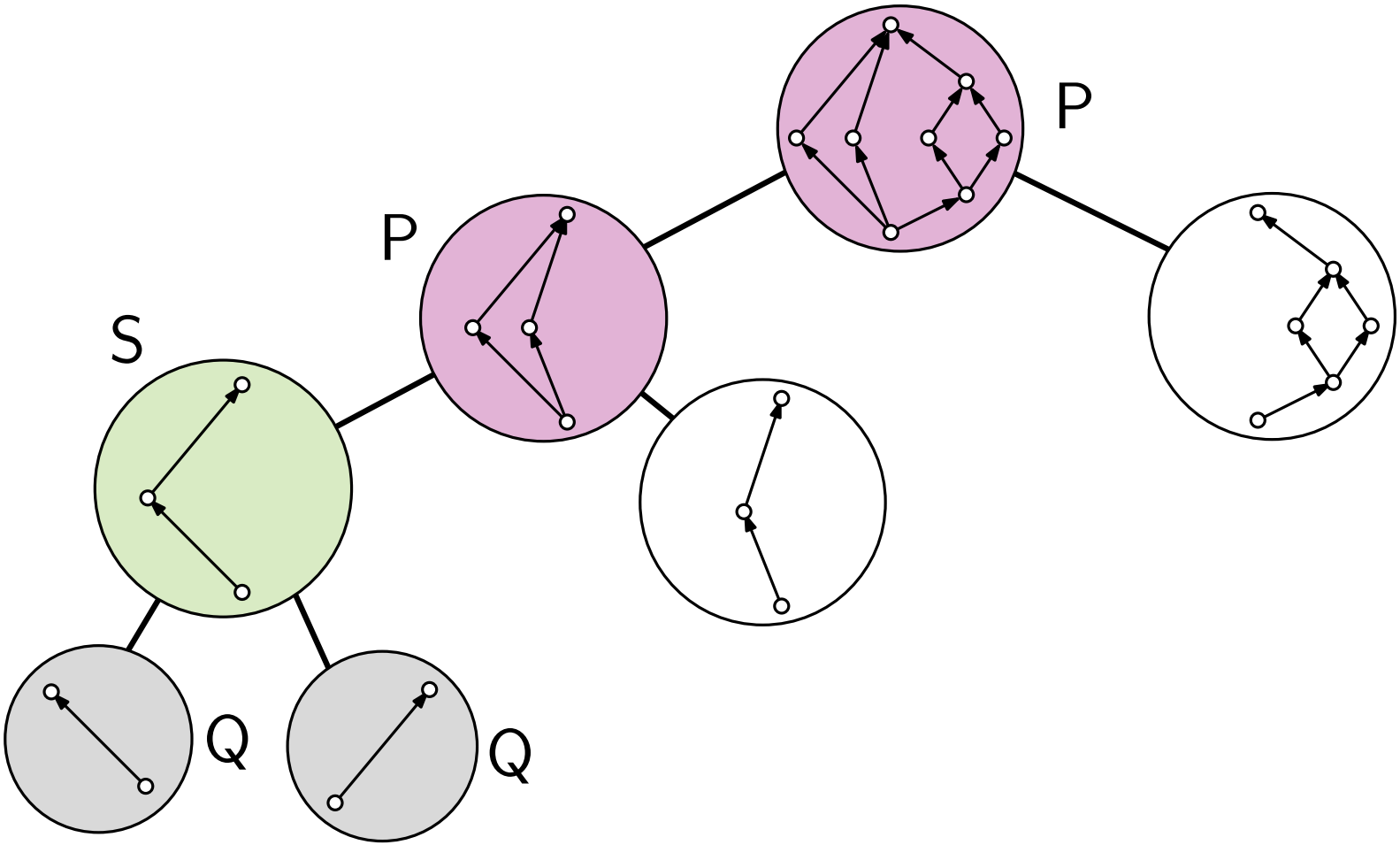
Decomposition Tree Example



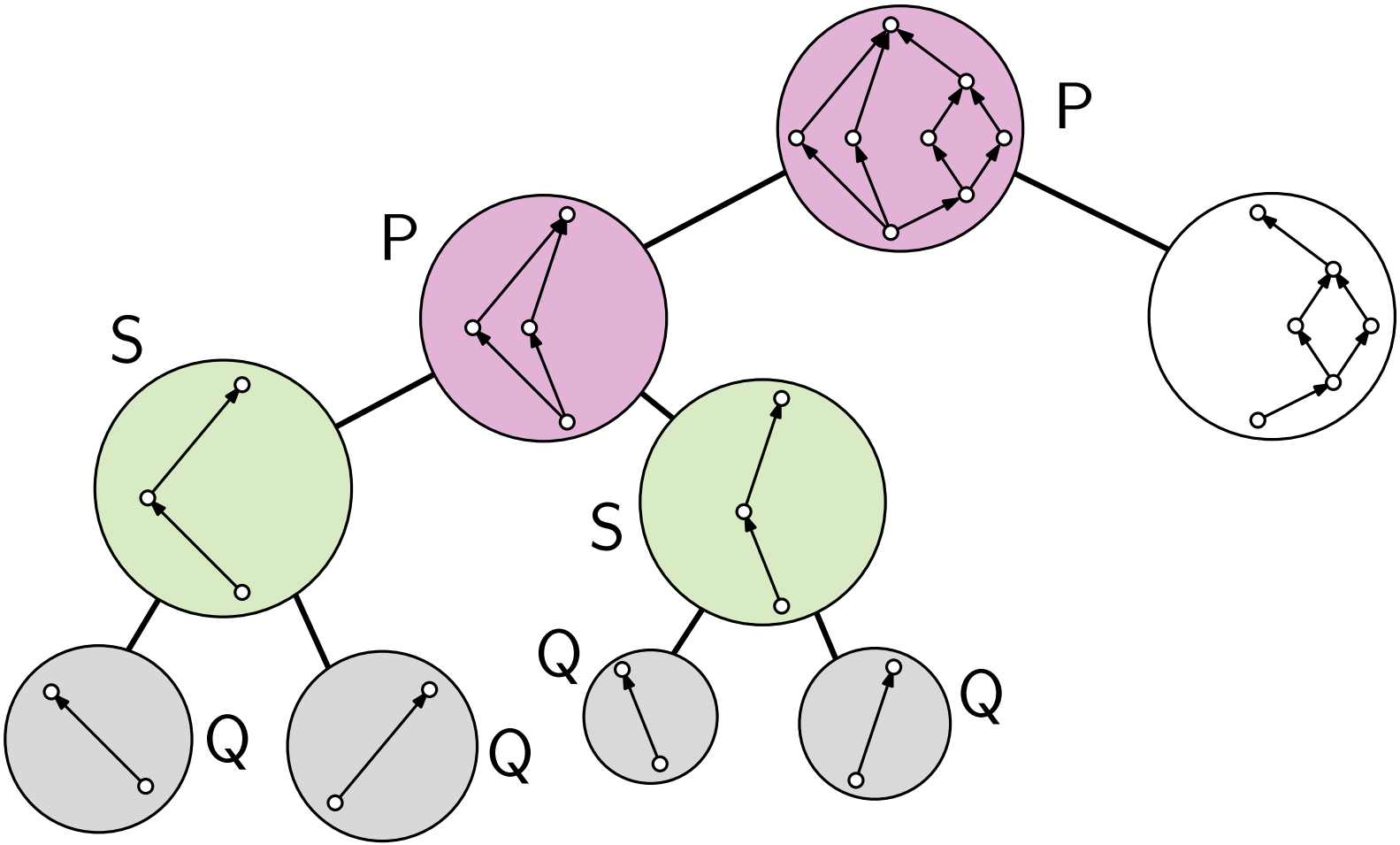
Decomposition Tree Example



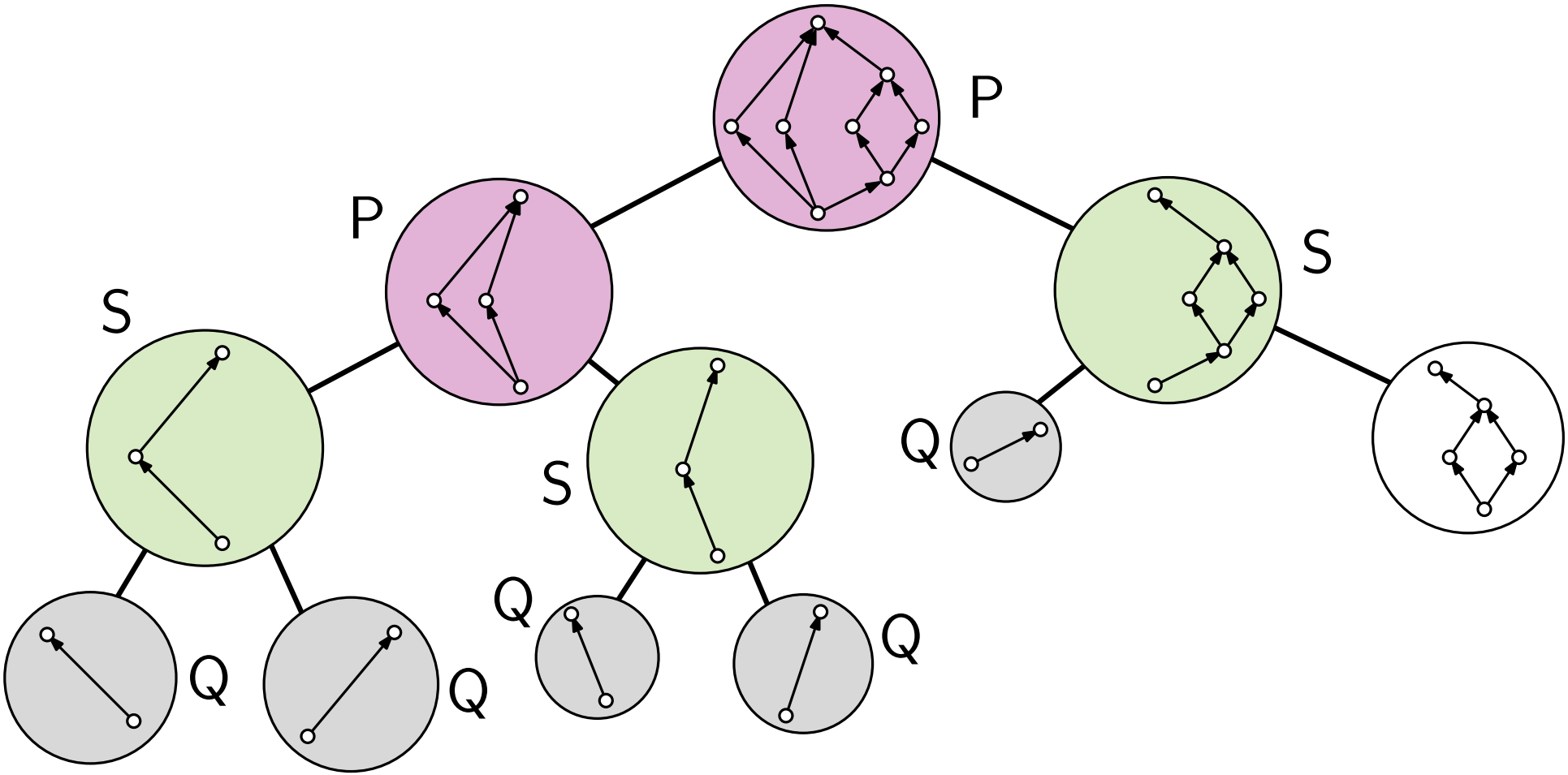
Decomposition Tree Example



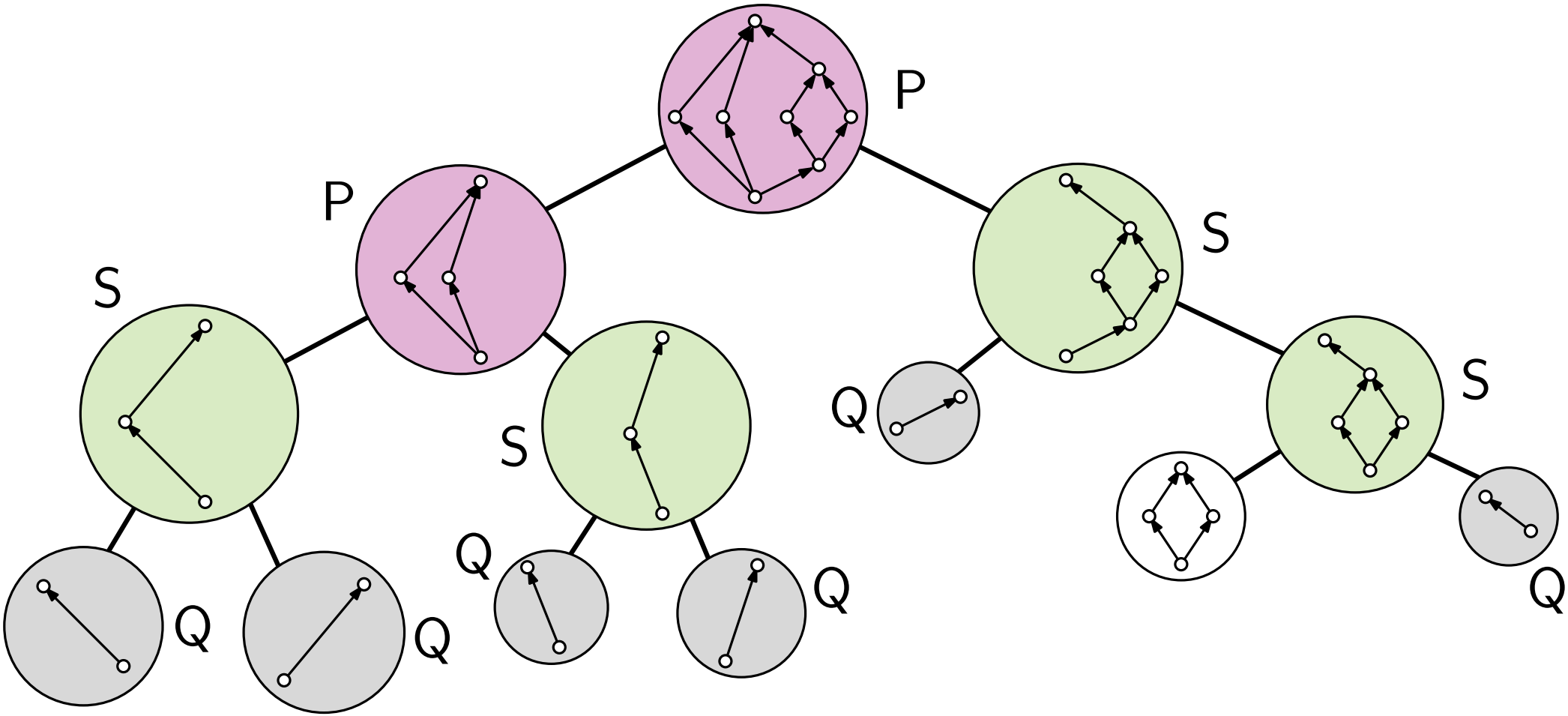
Decomposition Tree Example



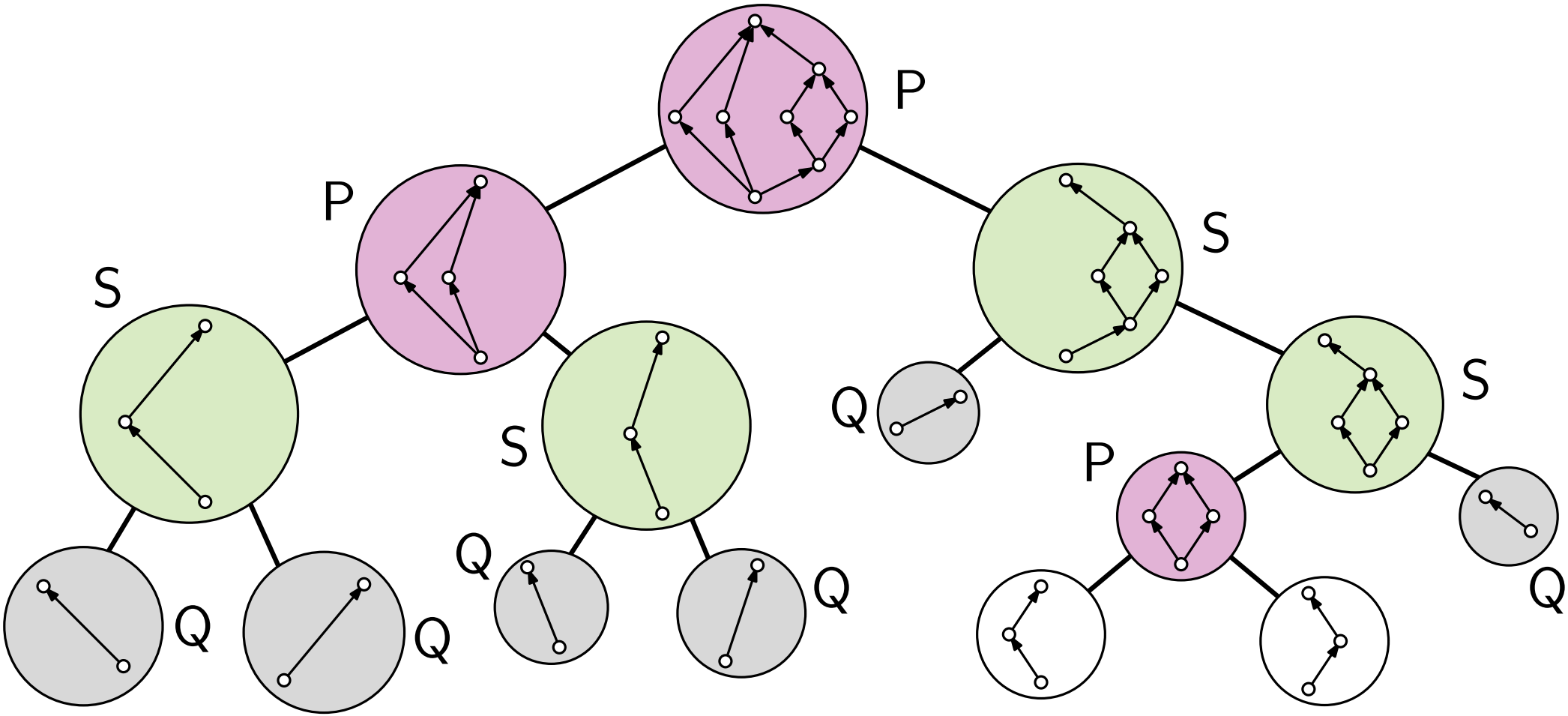
Decomposition Tree Example



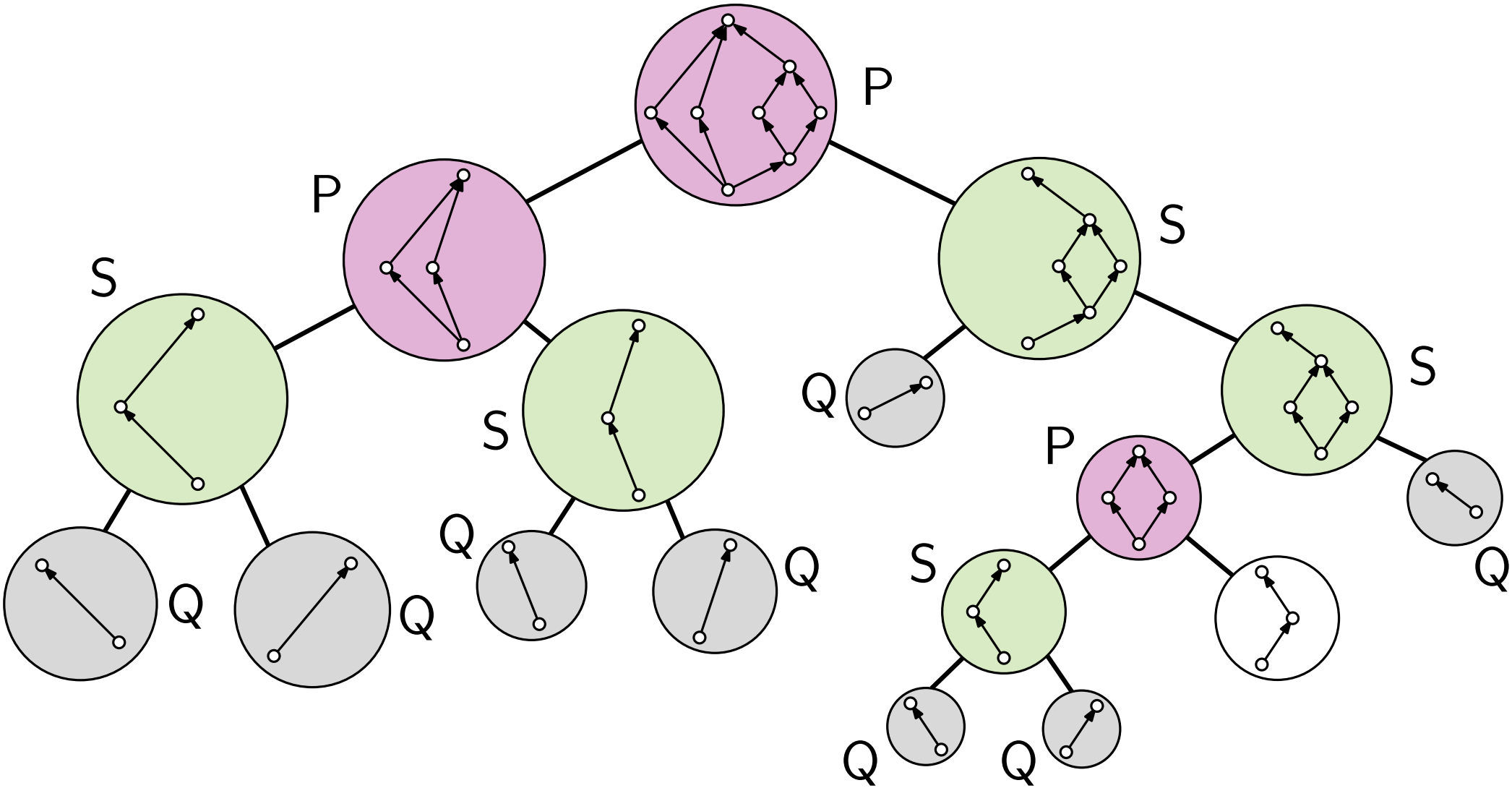
Decomposition Tree Example



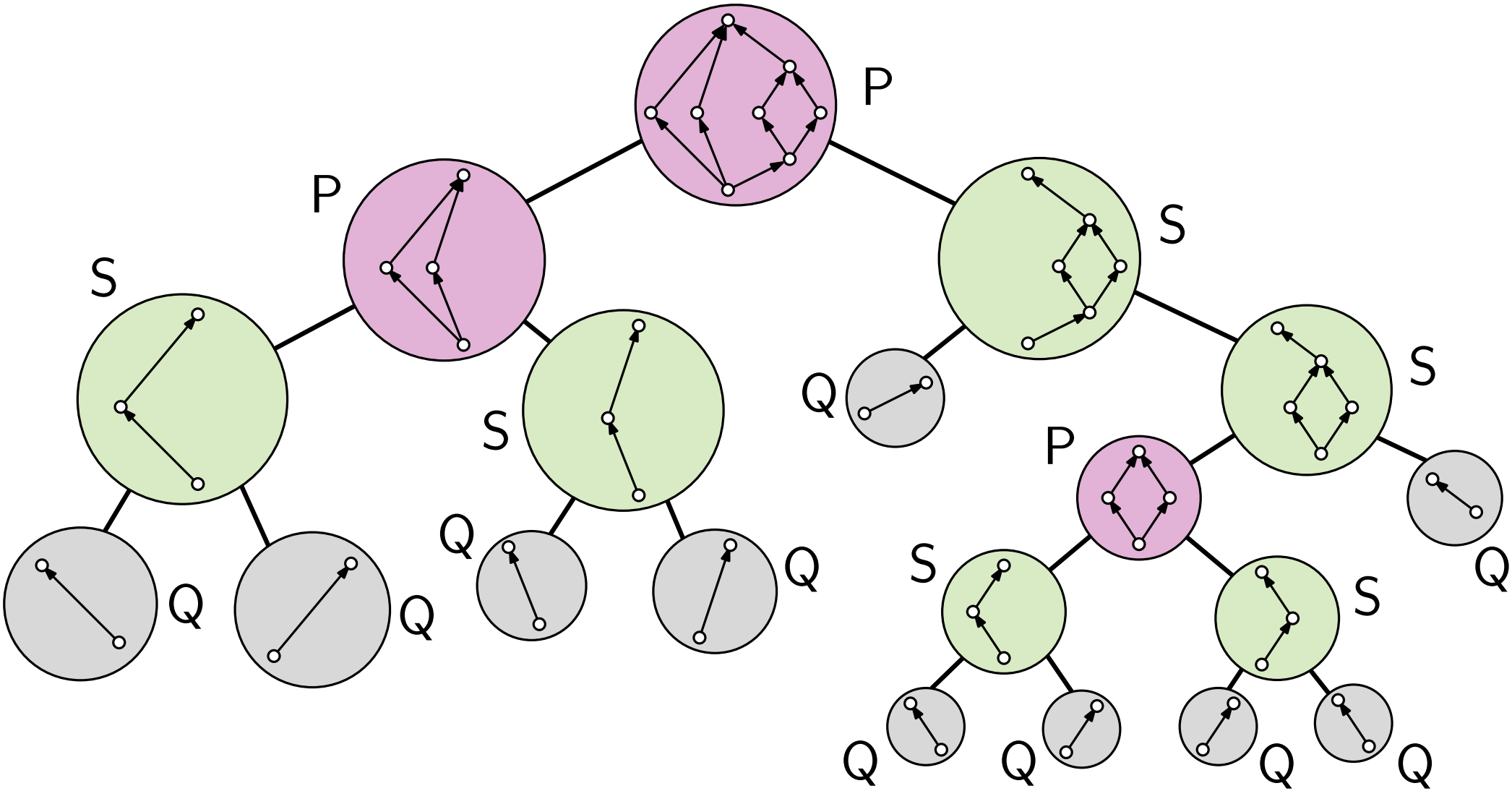
Decomposition Tree Example



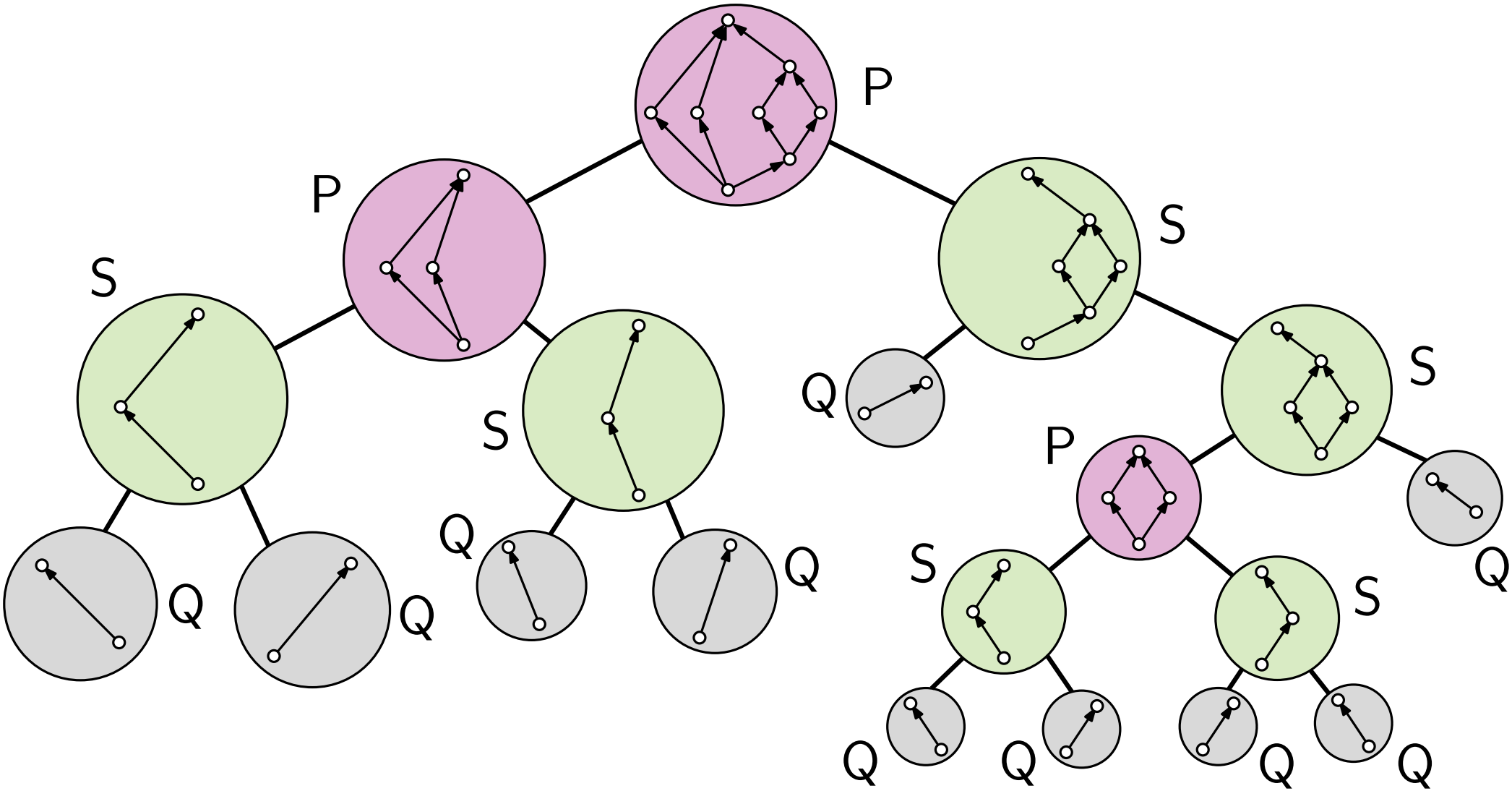
Decomposition Tree Example



Decomposition Tree Example



Decomposition Tree Example



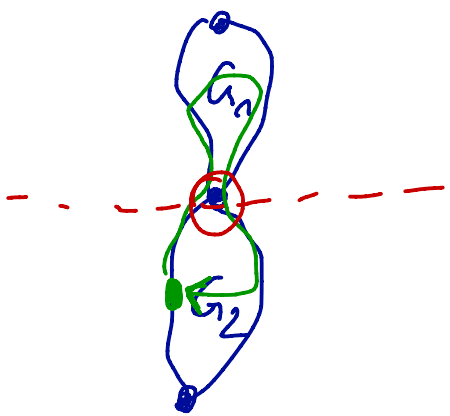
$\rightarrow$ A similar recursive decomposition tree, the *SPQR-Tree* exists for biconnected planar graphs and is frequently used as a tool in graph drawing.

Some Properties

Lemma: Series-parallel graphs are planar and acyclic.

How could you prove this?

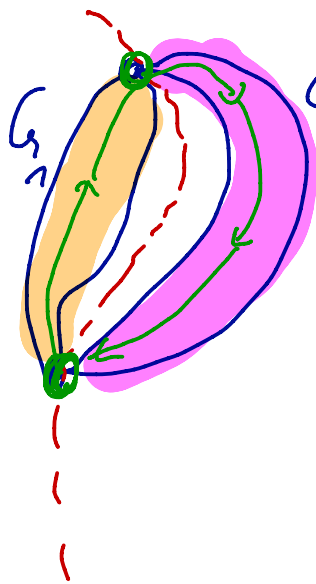
• S-composition:



planarity by design ✓

acyclic ✓

• P-composition



planar ✓

acyclic:
we identify two sources and two sinks
→ new sink has no outgoing edges

Some Properties

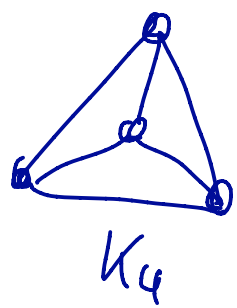
Lemma: Series-parallel graphs are planar and acyclic.

- Given a graph G with n vertices, it can be tested in $O(n)$ time whether G is series-parallel. If so, the decomposition tree can be constructed in $O(n)$ time. [Valdes et al. '82]

Some Properties

Lemma: Series-parallel graphs are planar and acyclic.

- Given a graph G with n vertices, it can be tested in $O(n)$ time whether G is series-parallel. If so, the decomposition tree can be constructed in $O(n)$ time. [Valdes et al. '82]
- Series-parallel graphs have treewidth ≤ 2 and are K_4-minor-free.



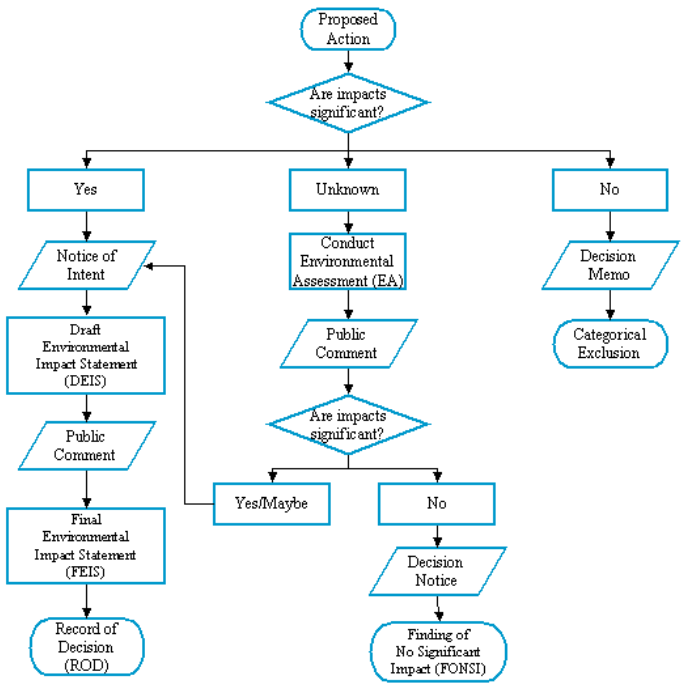
Some Properties

Lemma: Series-parallel graphs are planar and acyclic.

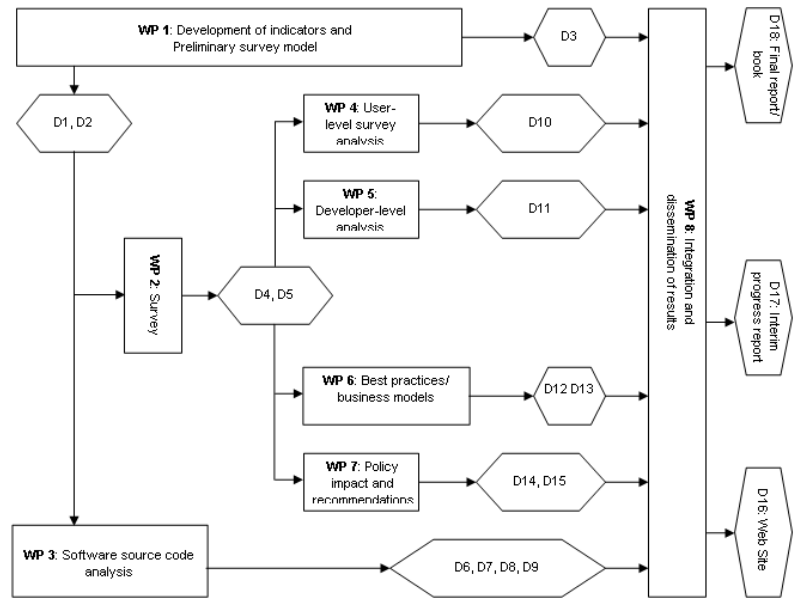
- Given a graph G with n vertices, it can be tested in $O(n)$ time whether G is series-parallel. If so, the decomposition tree can be constructed in $O(n)$ time. [Valdes et al. '82]
- Series-parallel graphs have treewidth ≤ 2 and are K_4-minor-free.
- Many NP-hard problems can be solved in linear time on series-parallel graphs, e.g., [Takamizawa et al. '82]
  - maximum independent set
  - maximum matching
  - minimum dominating set

Some Properties

Lemma: Series-parallel graphs are planar and acyclic.



flow chart



PERT diagram

(Program Evaluation and Review Technique)

- Series-parallel graphs are used to model electrical circuits, flow charts, PERT diagrams etc.

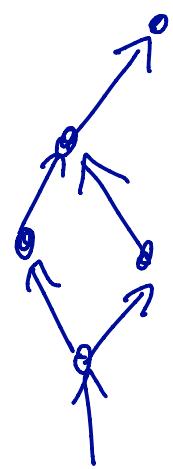
Drawing Series-Parallel Graphs

What are desirable properties of drawings of series-parallel graphs?

- what are reasonable drawing conventions?
- which aesthetics shall be optimized?

Let's collect some properties!

- straight edges
- upward drawings
- no crossings



optimize?

- area
- minimize longest edge
- similar edge lengths
- isomorphic structures drawn the same
- symmetries

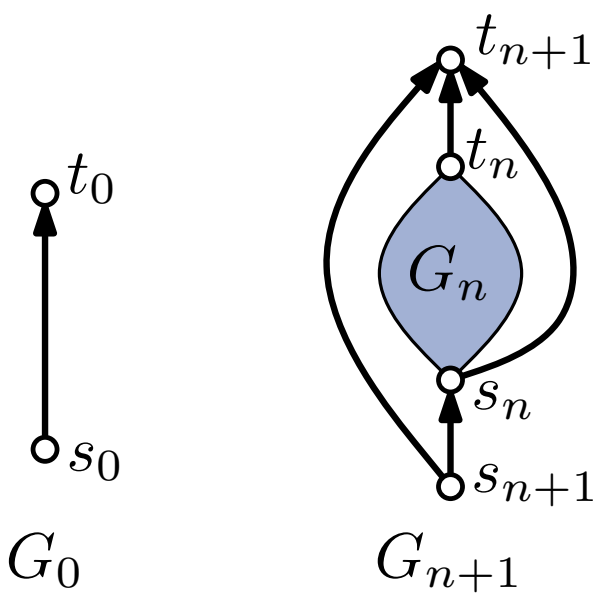
Bad News: Exponential Area Lower Bound

Theorem: There exists a family of series-parallel graphs G_n ($n \in \mathbb{N}$) such that any order-preserving upward planar straight-line drawing of G_n requires area $\Omega(4^n)$. [Bertolazzi et al. '94]

Bad News: Exponential Area Lower Bound

Theorem: There exists a family of series-parallel graphs G_n ($n \in \mathbb{N}$) such that any order-preserving upward planar drawing of G_n requires area $\Omega(4^n)$. [Bertolazzi et al. '94]

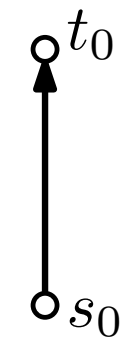
Proof (sketch):



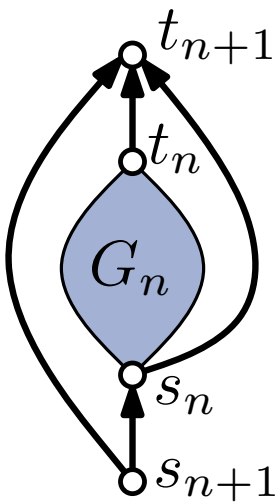
Bad News: Exponential Area Lower Bound

Theorem: There exists a family of series-parallel graphs G_n ($n \in \mathbb{N}$) such that any order-preserving upward planar drawing of G_n requires area $\Omega(4^n)$. [Bertolazzi et al. '94]

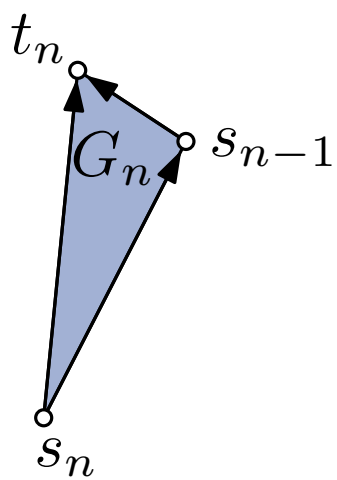
Proof (sketch):



G_0



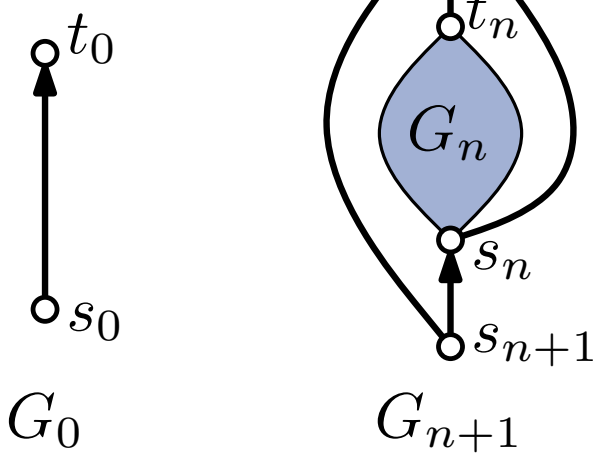
G_{n+1}

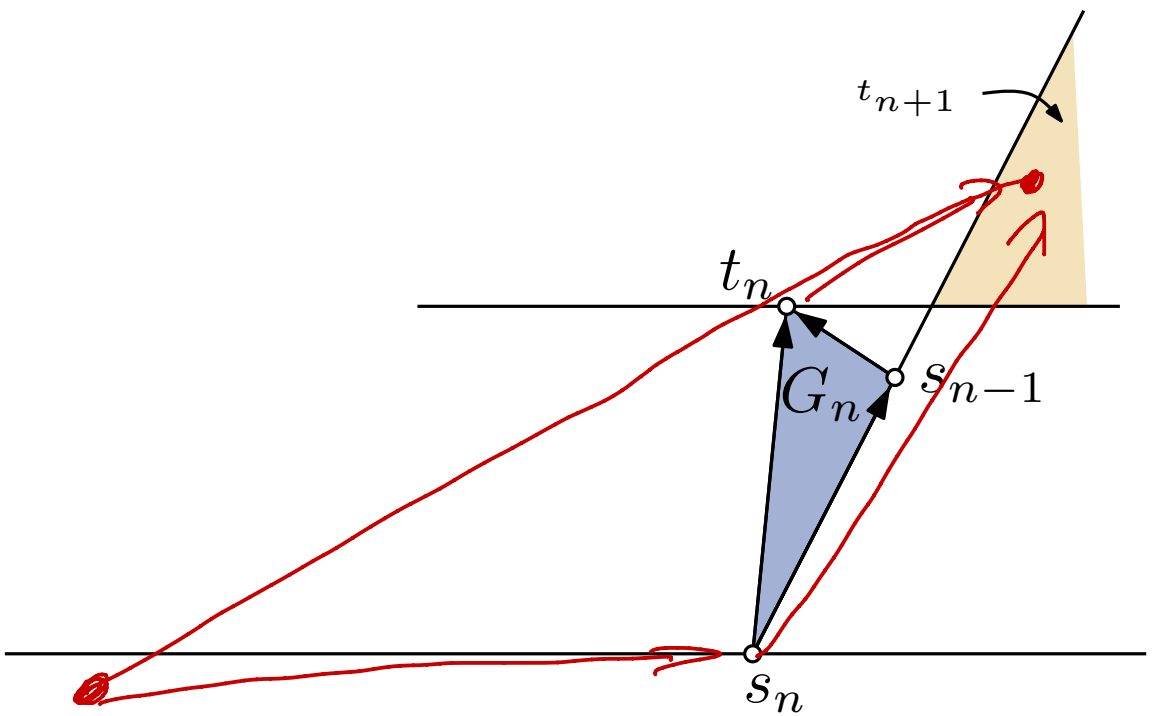


Bad News: Exponential Area Lower Bound

Theorem: There exists a family of series-parallel graphs G_n ($n \in \mathbb{N}$) such that any order-preserving upward planar drawing of G_n requires area $\Omega(4^n)$. [Bertolazzi et al. '94]

Proof (sketch):

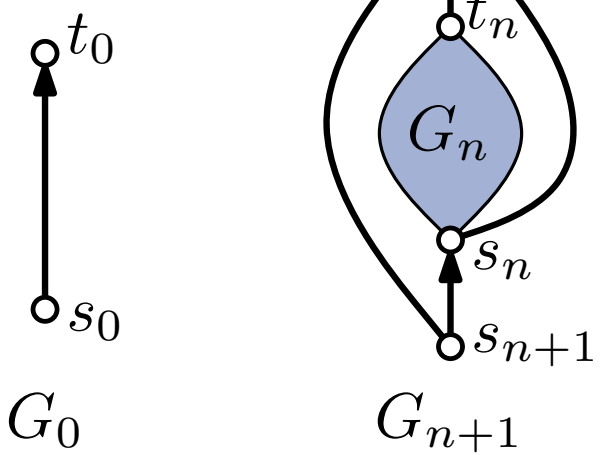


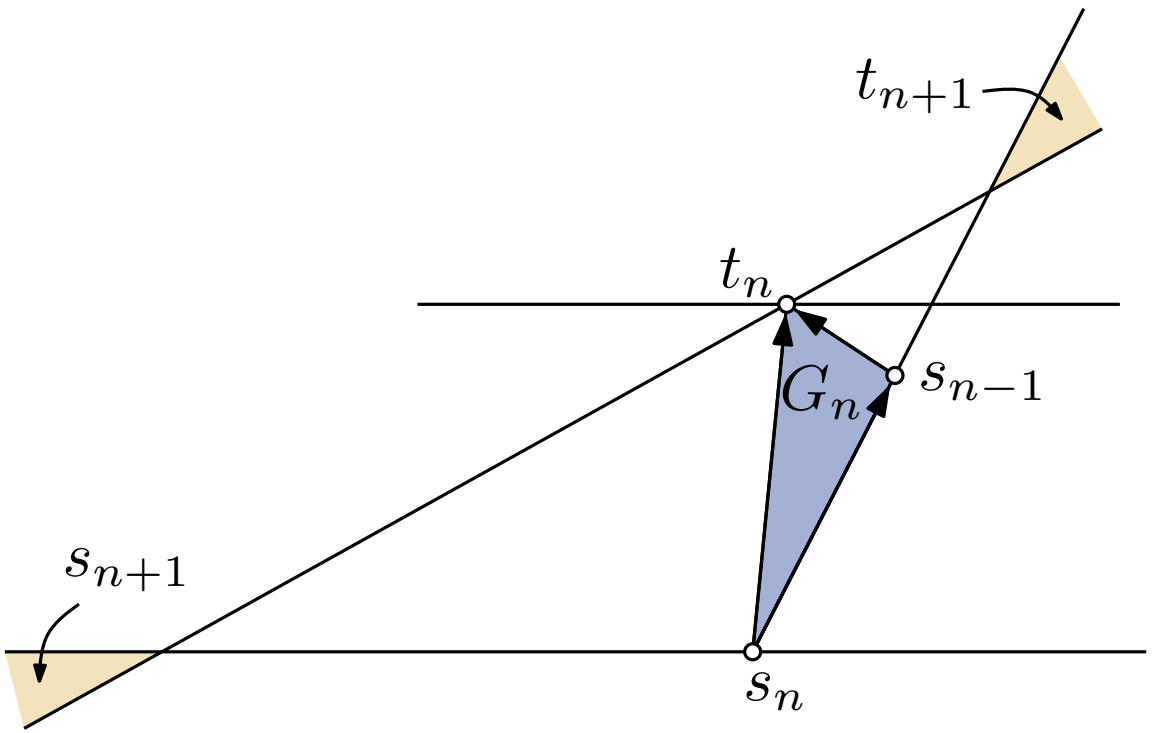


Bad News: Exponential Area Lower Bound

Theorem: There exists a family of series-parallel graphs G_n ($n \in \mathbb{N}$) such that any order-preserving upward planar drawing of G_n requires area $\Omega(4^n)$. [Bertolazzi et al. '94]

Proof (sketch):

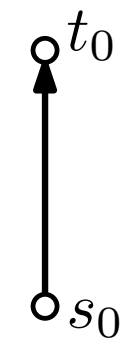




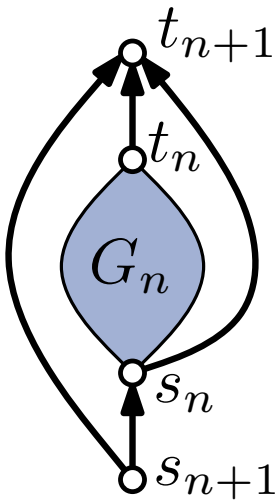
Bad News: Exponential Area Lower Bound

Theorem: There exists a family of series-parallel graphs G_n ($n \in \mathbb{N}$) such that any order-preserving upward planar drawing of G_n requires area $\Omega(4^n)$. [Bertolazzi et al. '94]

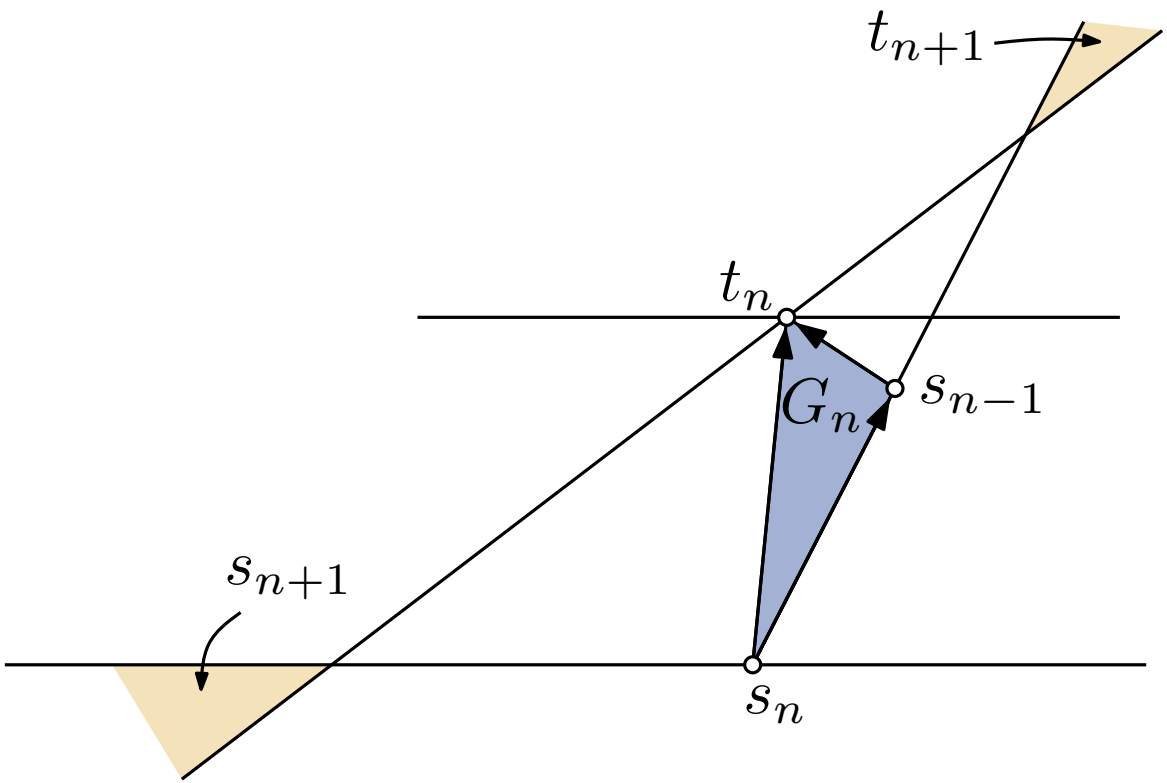
Proof (sketch):



G_0



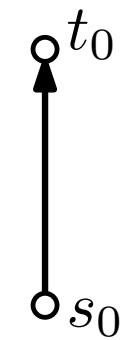
G_{n+1}



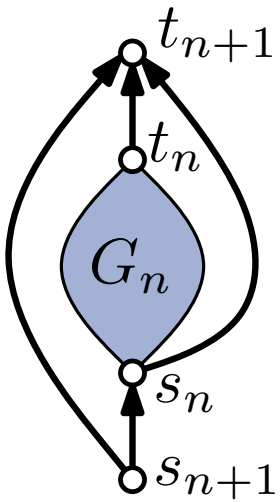
Bad News: Exponential Area Lower Bound

Theorem: There exists a family of series-parallel graphs G_n ($n \in \mathbb{N}$) such that any order-preserving upward planar drawing of G_n requires area $\Omega(4^n)$. [Bertolazzi et al. '94]

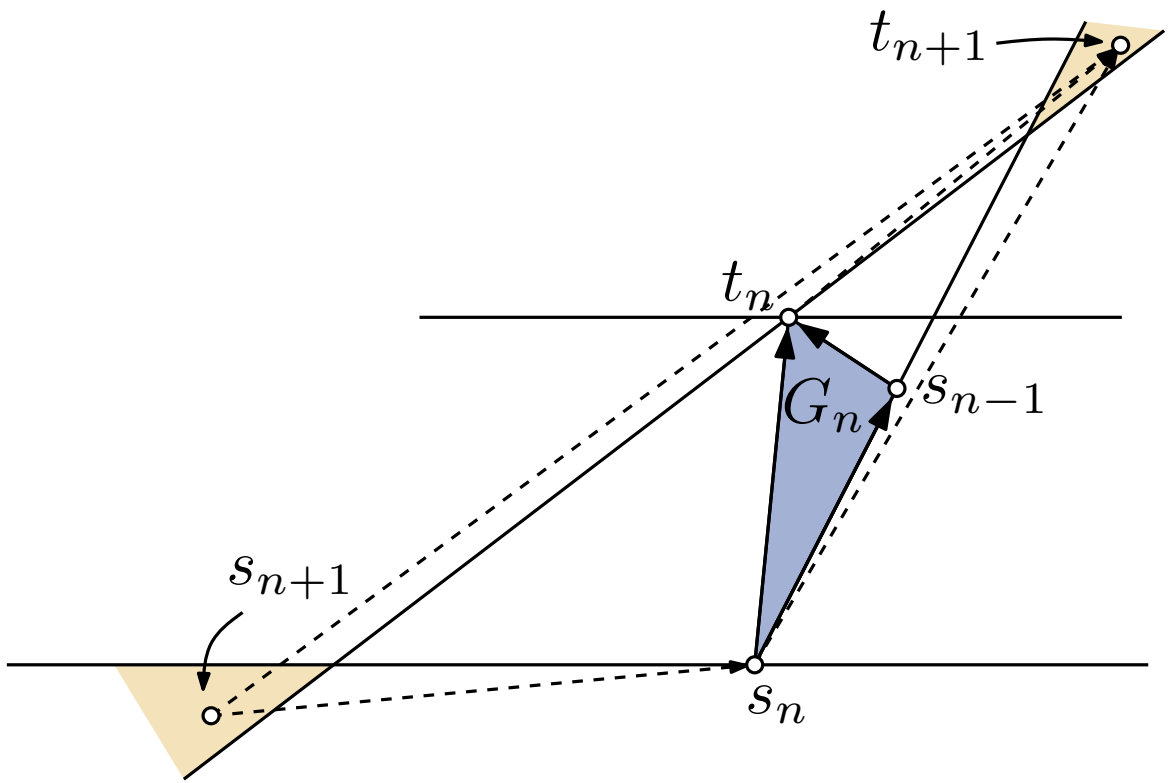
Proof (sketch):



G_0



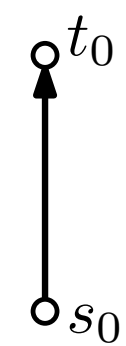
G_{n+1}



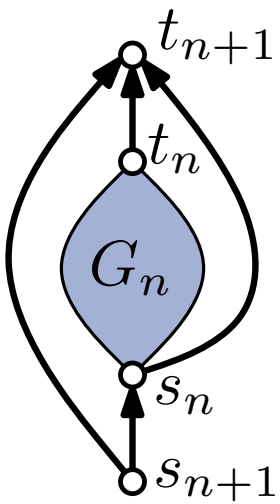
Bad News: Exponential Area Lower Bound

Theorem: There exists a family of series-parallel graphs G_n ($n \in \mathbb{N}$) such that any order-preserving upward planar drawing of G_n requires area $\Omega(4^n)$. [Bertolazzi et al. '94]

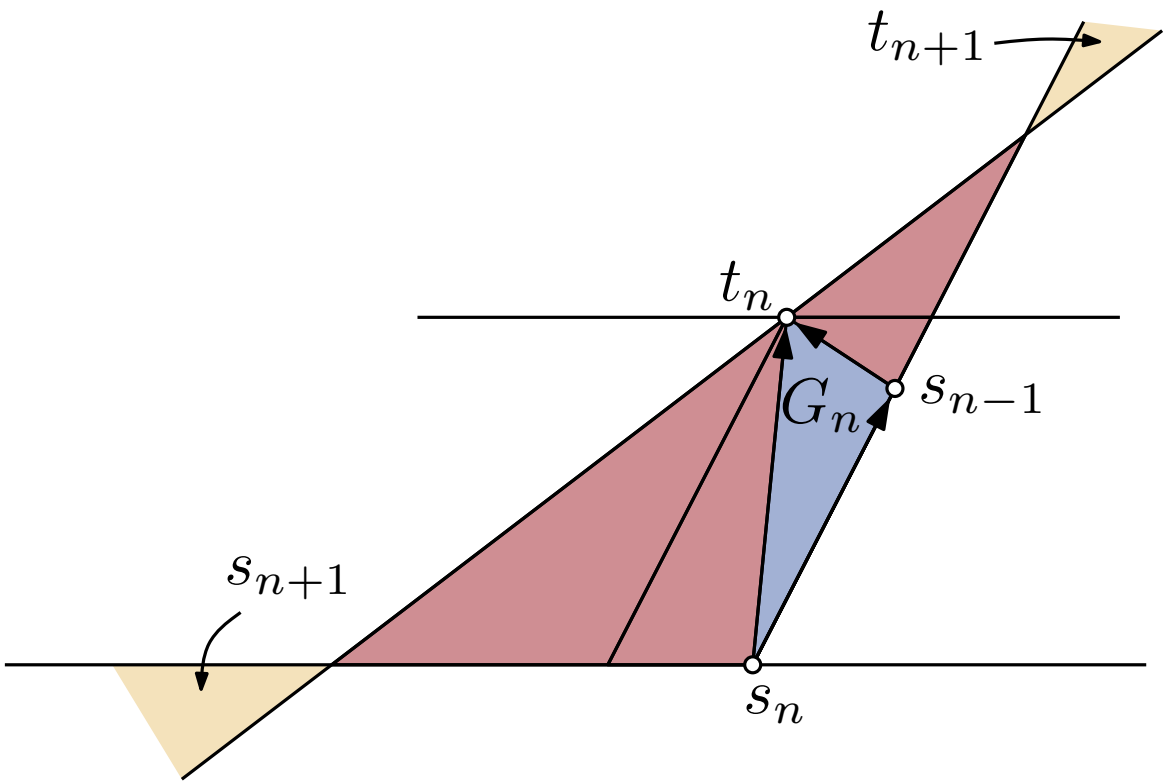
Proof (sketch):



G_0



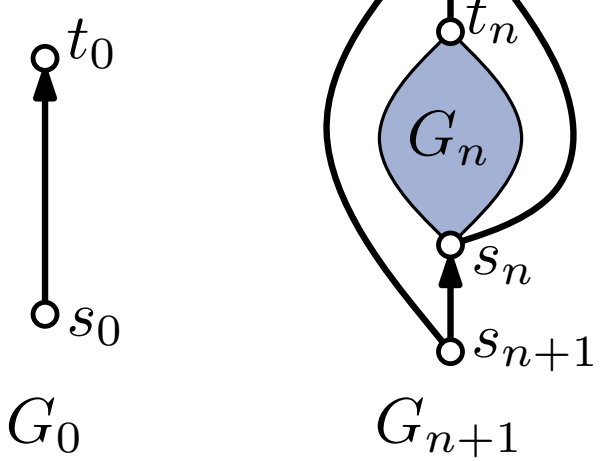
G_{n+1}

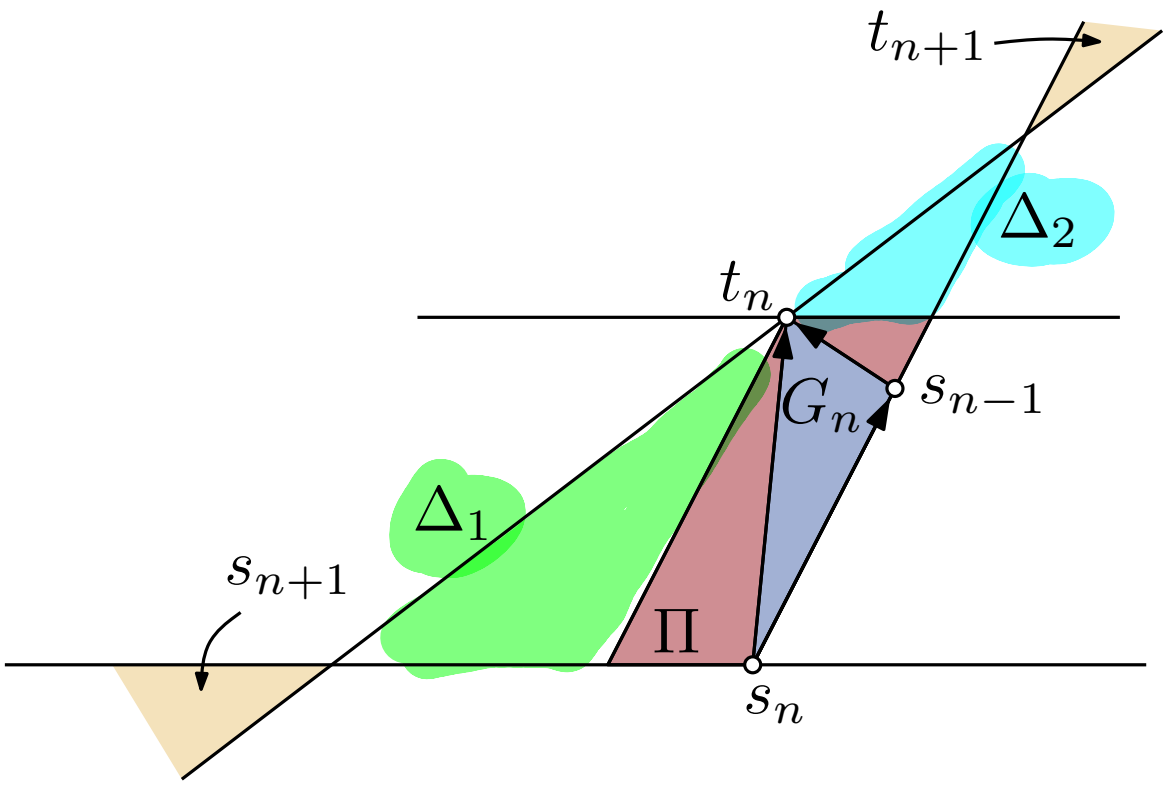


Bad News: Exponential Area Lower Bound

Theorem: There exists a family of series-parallel graphs G_n ($n \in \mathbb{N}$) such that any order-preserving upward planar drawing of G_n requires area $\Omega(4^n)$. [Bertolazzi et al. '94]

Proof (sketch):

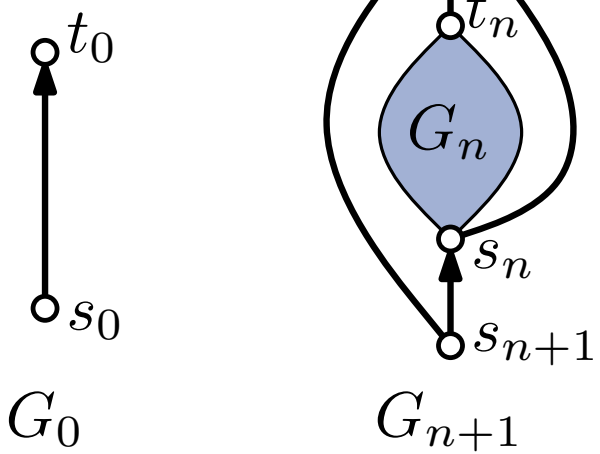


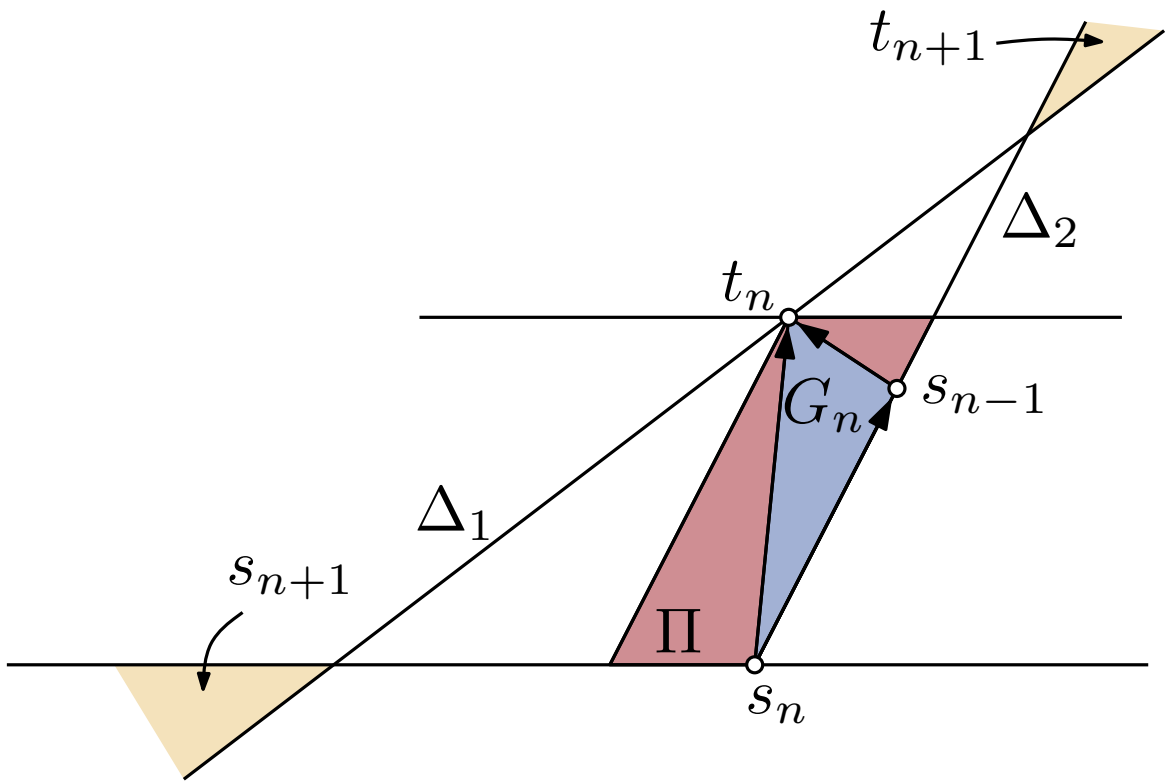


Bad News: Exponential Area Lower Bound

Theorem: There exists a family of series-parallel graphs G_n ($n \in \mathbb{N}$) such that any order-preserving upward planar drawing of G_n requires area $\Omega(4^n)$. [Bertolazzi et al. '94]

Proof (sketch):



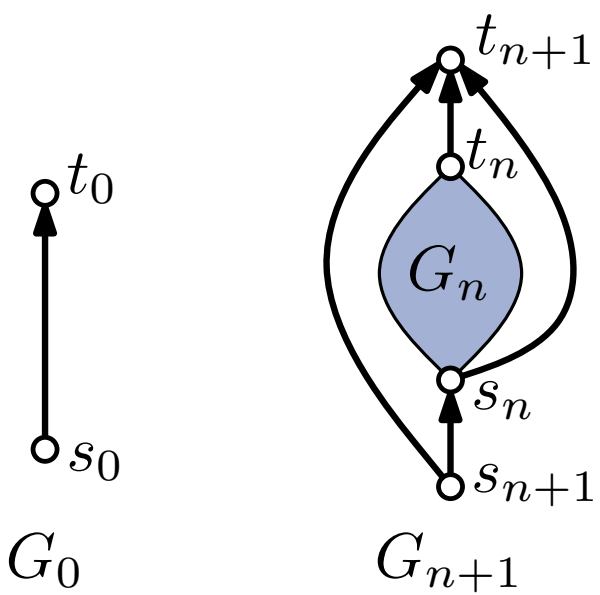


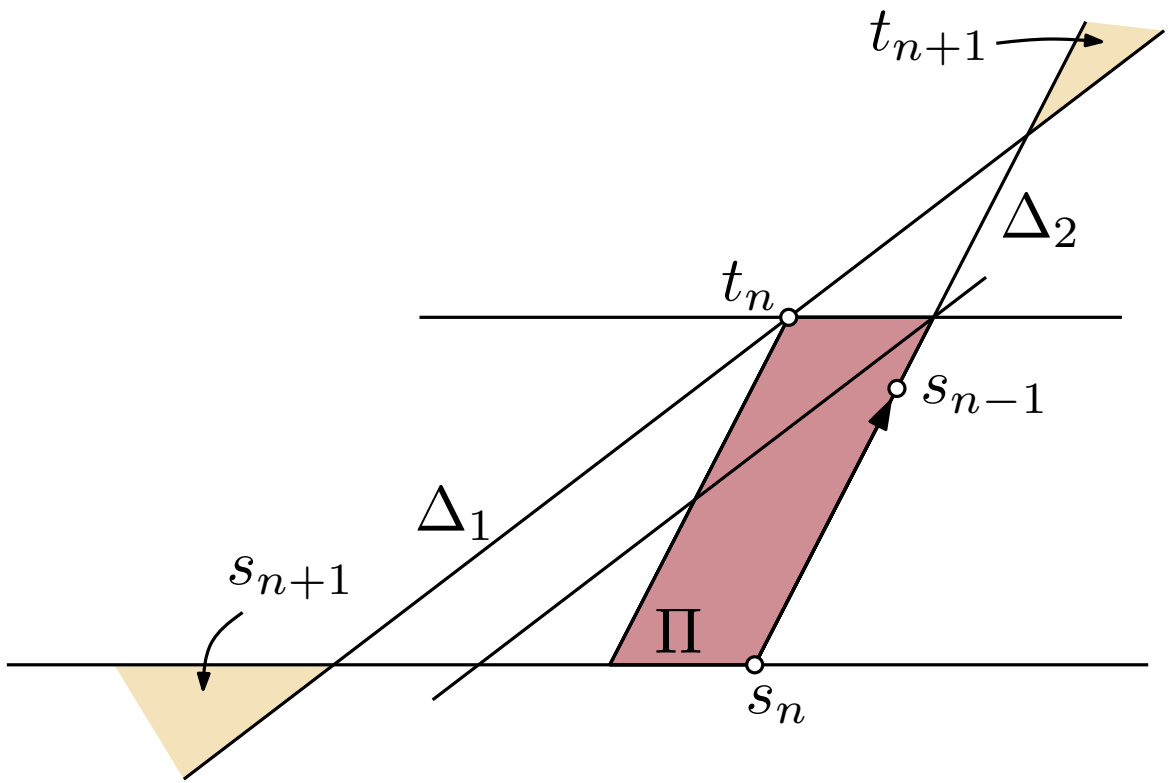
we have: $a(G_{n+1}) \geq a(\Pi) + a(\Delta_1) + a(\Delta_2)$ and $a(G_n) \leq 1/2 \cdot a(\Pi)$

Bad News: Exponential Area Lower Bound

Theorem: There exists a family of series-parallel graphs G_n ($n \in \mathbb{N}$) such that any order-preserving upward planar drawing of G_n requires area $\Omega(4^n)$. [Bertolazzi et al. '94]

Proof (sketch):



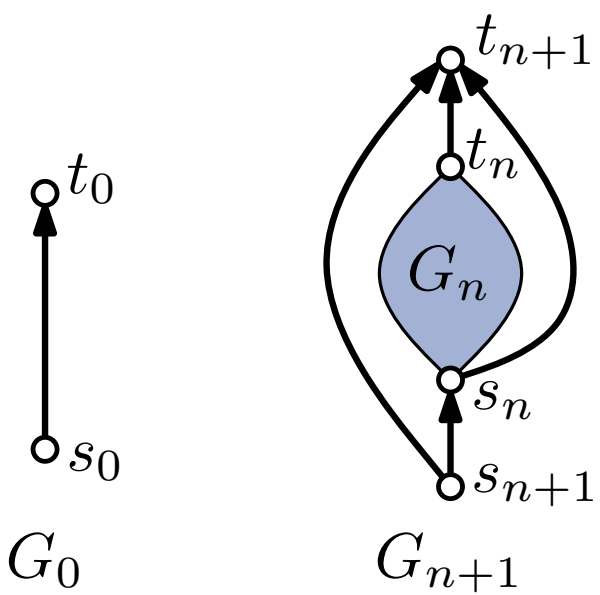


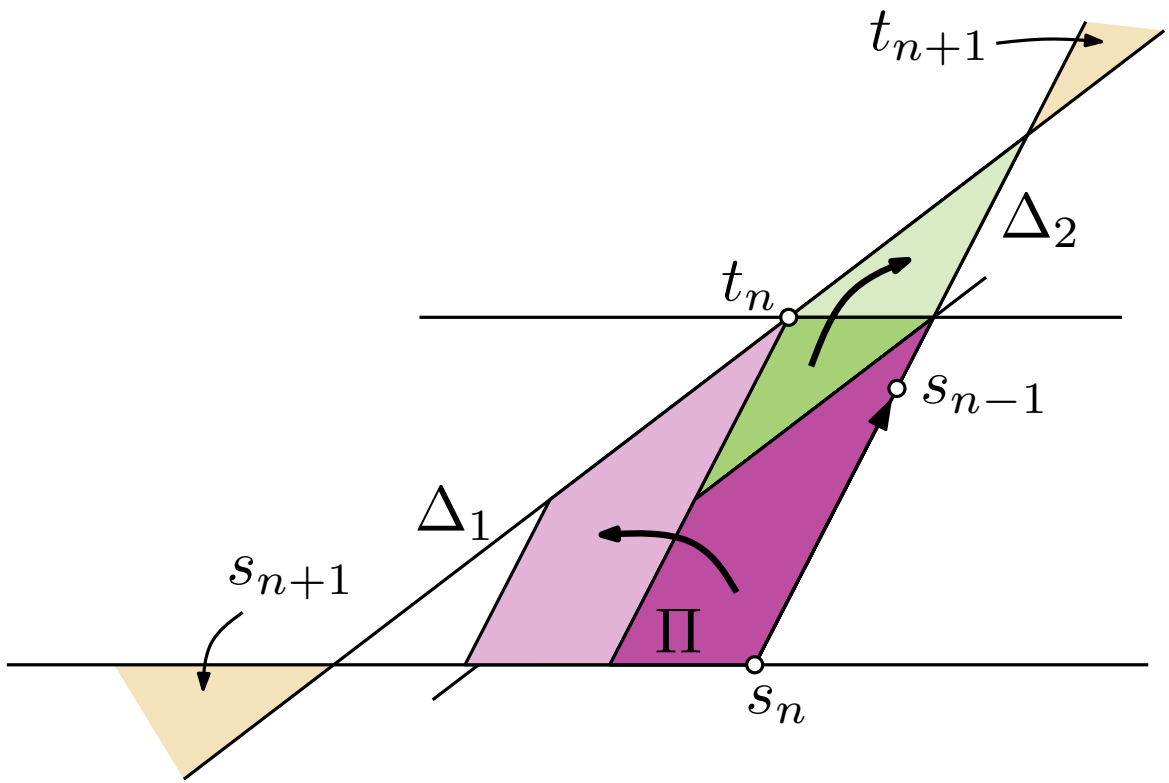
we have: $a(G_{n+1}) \geq a(\Pi) + a(\Delta_1) + a(\Delta_2)$ and $a(G_n) \leq 1/2 \cdot a(\Pi)$

Bad News: Exponential Area Lower Bound

Theorem: There exists a family of series-parallel graphs G_n ($n \in \mathbb{N}$) such that any order-preserving upward planar drawing of G_n requires area $\Omega(4^n)$. [Bertolazzi et al. '94]

Proof (sketch):



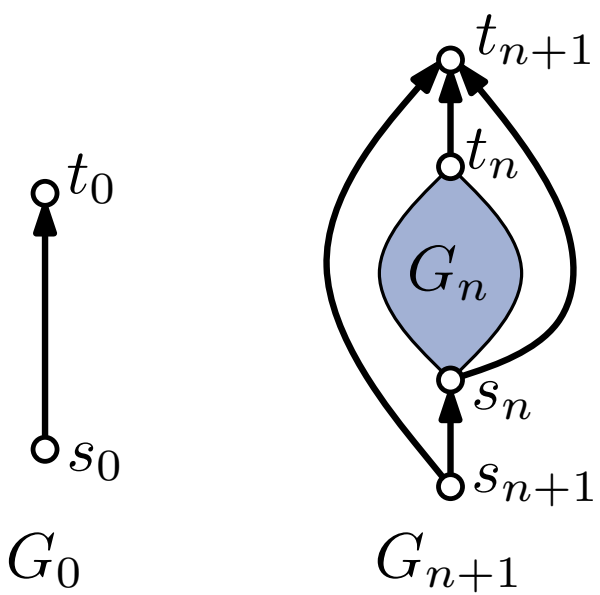


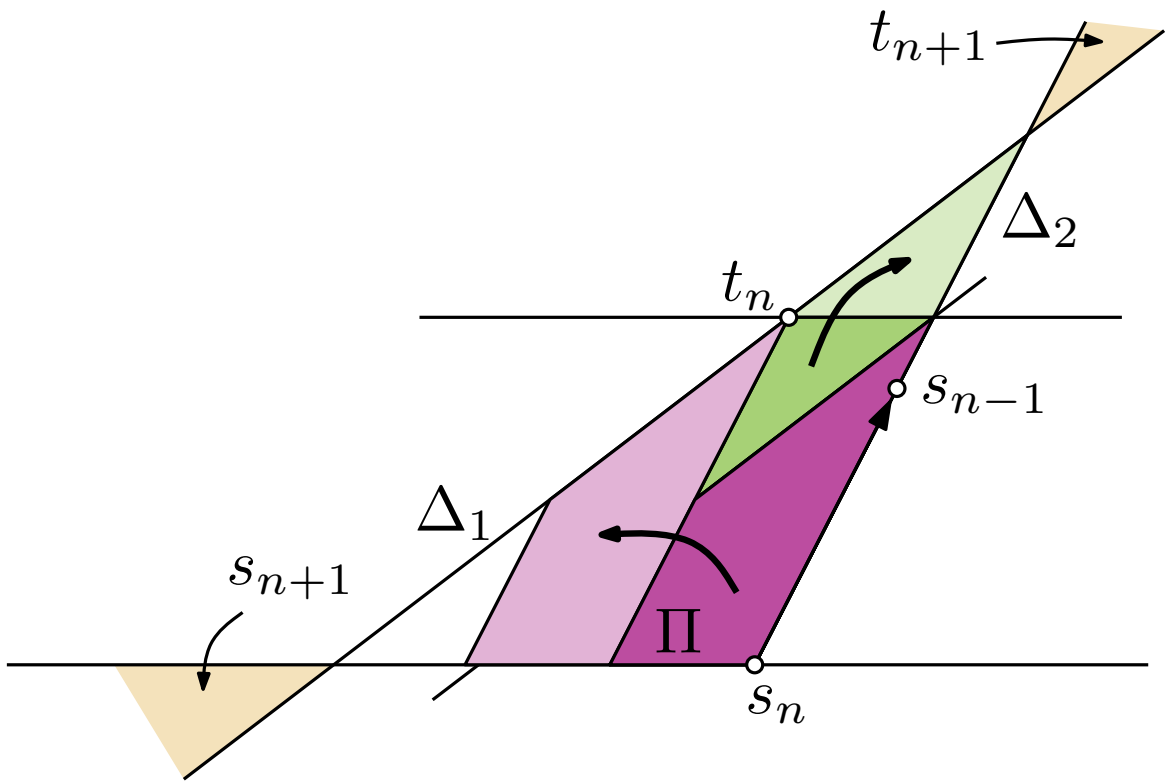
we have: $a(G_{n+1}) \geq a(\Pi) + a(\Delta_1) + a(\Delta_2)$ and $a(G_n) \leq 1/2 \cdot a(\Pi)$

Bad News: Exponential Area Lower Bound

Theorem: There exists a family of series-parallel graphs G_n ($n \in \mathbb{N}$) such that any order-preserving upward planar drawing of G_n requires area $\Omega(4^n)$. [Bertolazzi et al. '94]

Proof (sketch):





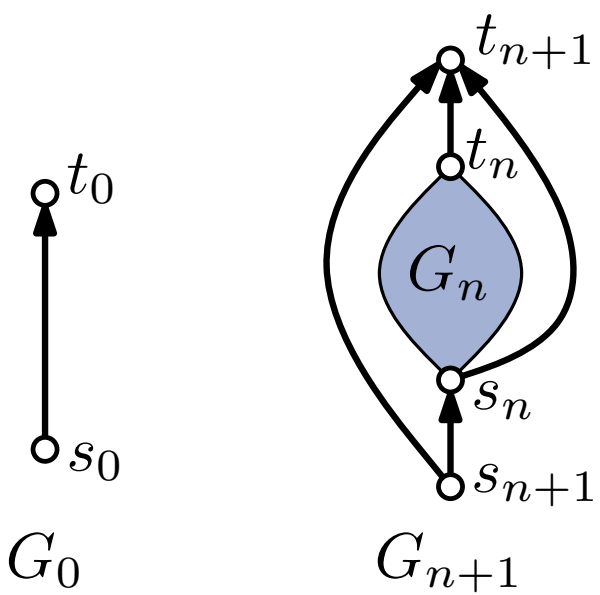
we have: $a(G_{n+1}) \geq a(\Pi) + a(\Delta_1) + a(\Delta_2)$ and $a(G_n) \leq 1/2 \cdot a(\Pi)$

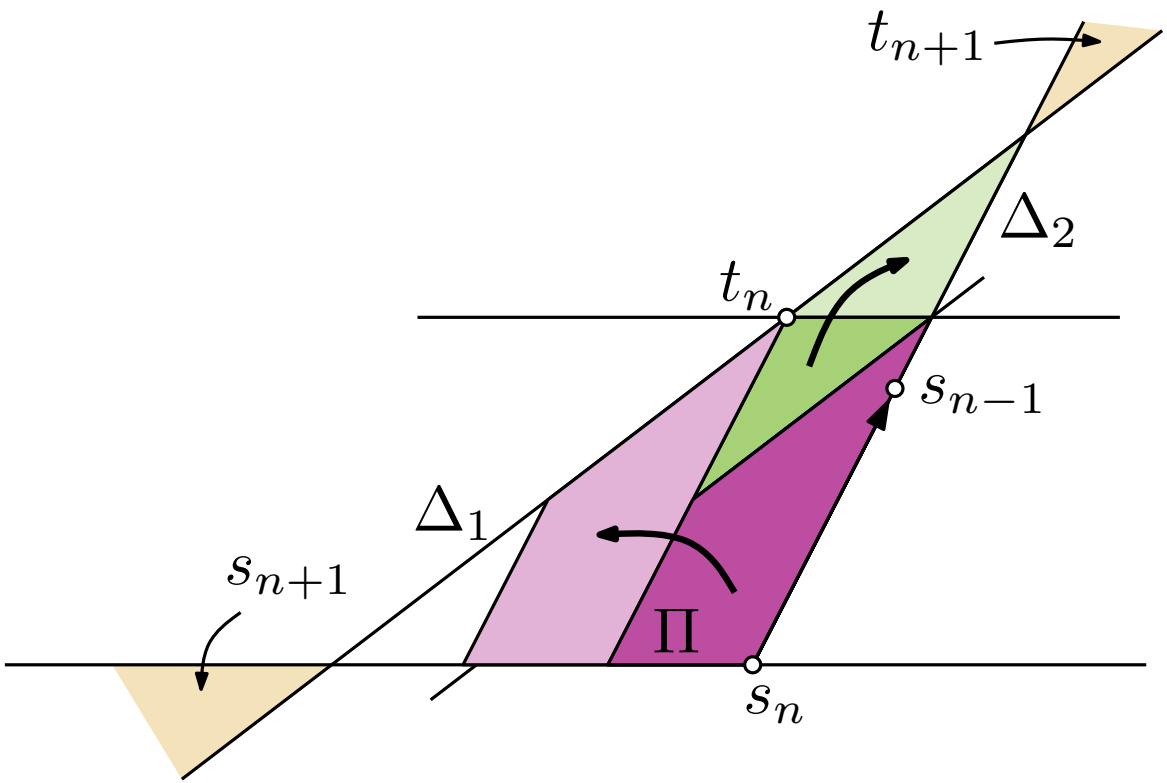
further: $a(\Delta_1) + a(\Delta_2) \geq a(\Pi) \Rightarrow a(G_{n+1}) \geq 2a(\Pi) \geq 4a(G_n)$

Bad News: Exponential Area Lower Bound

Theorem: There exists a family of series-parallel graphs G_n ($n \in \mathbb{N}$) such that any order-preserving upward planar straight-line drawing of G_n requires area $\Omega(4^n)$. [Bertolazzi et al. '94]

Proof (sketch):





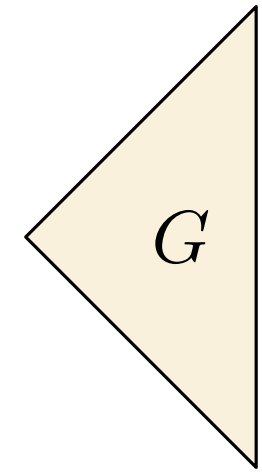
we have: $a(G_{n+1}) \geq a(\Pi) + a(\Delta_1) + a(\Delta_2)$ and $a(G_n) \leq 1/2 \cdot a(\Pi)$

further: $a(\Delta_1) + a(\Delta_2) \geq a(\Pi) \Rightarrow a(G_{n+1}) \geq 2a(\Pi) \geq 4a(G_n)$ □

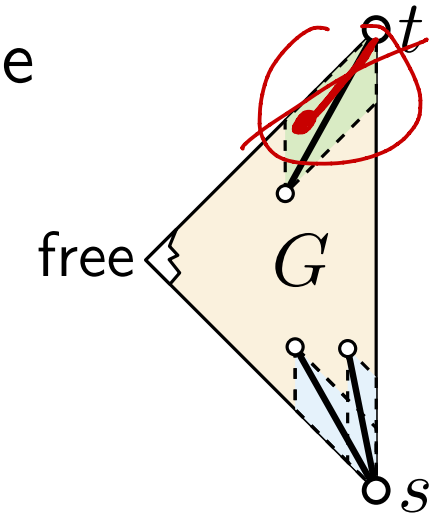
Good News: Polynomial-Area Drawings

Divide & conquer algorithm using decomposition tree

- draw G in an isosceles, right-angled triangle Δ



Good News: Polynomial-Area Drawings



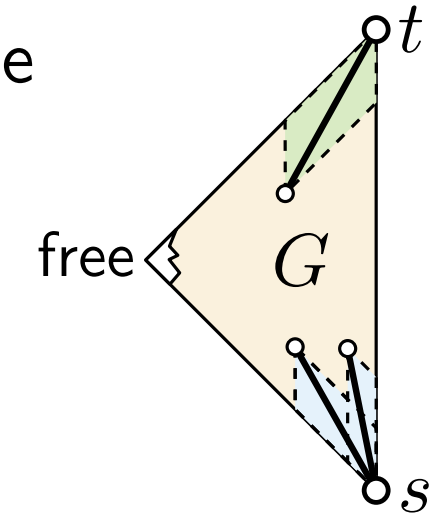
Divide & conquer algorithm using decomposition tree

- draw G in an isosceles, right-angled triangle Δ

Invariants:

- s and t in lower/upper corner of Δ
- left corner of Δ remains free (no vertex)
- for each neighbor v of s or t, the triangle-aligned parallelogram from v contains no vertex (except s or t)

Good News: Polynomial-Area Drawings

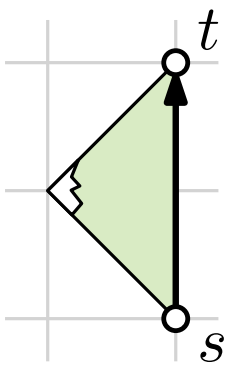


Divide & conquer algorithm using decomposition tree

- draw G in an isosceles, right-angled triangle Δ

Invariants:

- s and t in lower/upper corner of Δ
- left corner of Δ remains free (no vertex)
- for each neighbor v of s or t, the triangle-aligned parallelogram from v contains no vertex (except s or t)

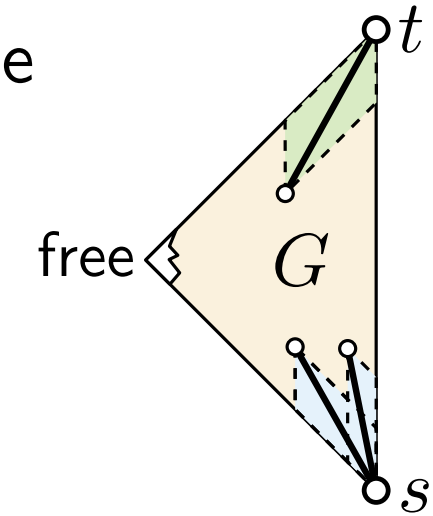


Q-node

Good News: Polynomial-Area Drawings

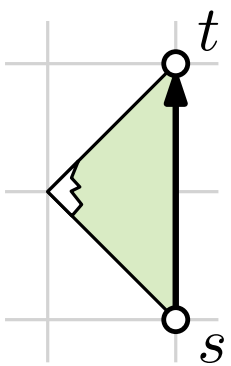
Divide & conquer algorithm using decomposition tree

- draw G in an isosceles, right-angled triangle Δ

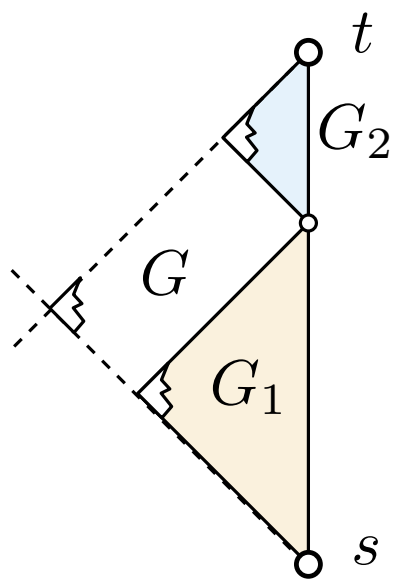


Invariants:

- s and t in lower/upper corner of Δ
- left corner of Δ remains free (no vertex)
- for each neighbor v of s or t, the triangle-aligned parallelogram from v contains no vertex (except s or t)



Q-node

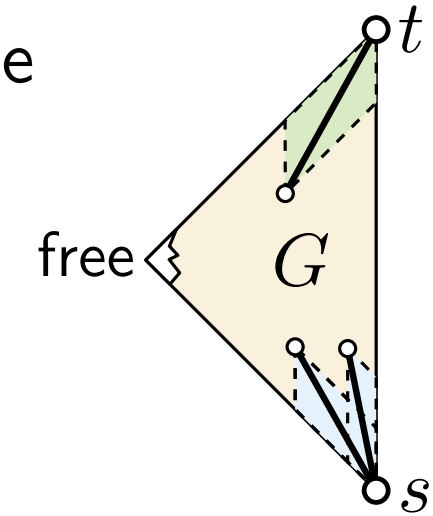


S-node

Good News: Polynomial-Area Drawings

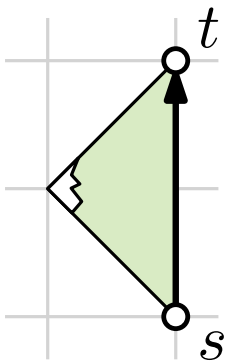
Divide & conquer algorithm using decomposition tree

- draw G in an isosceles, right-angled triangle Δ

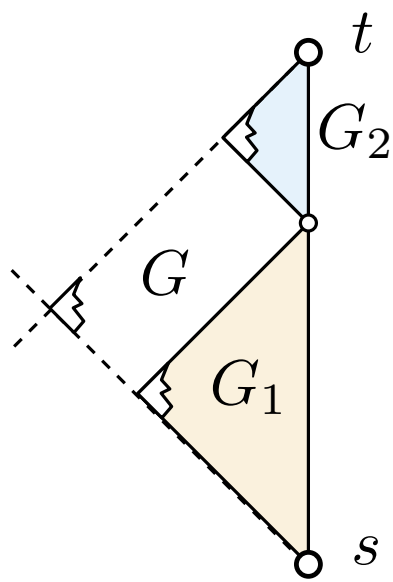


Invariants:

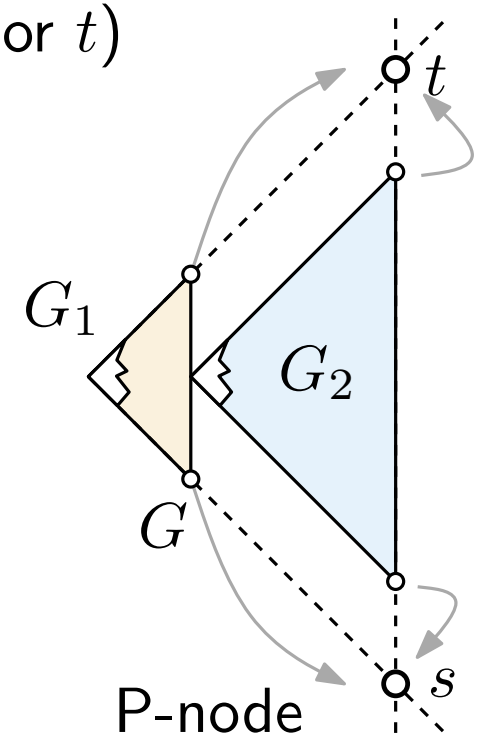
- s and t in lower/upper corner of Δ
- left corner of Δ remains free (no vertex)
- for each neighbor v of s or t, the triangle-aligned parallelogram from v contains no vertex (except s or t)



Q-node



S-node

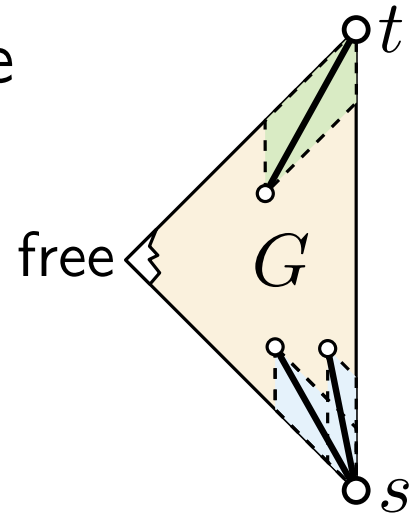


P-node

Good News: Polynomial-Area Drawings

Divide & conquer algorithm using decomposition tree

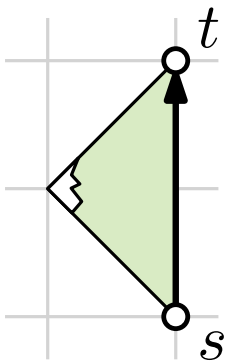
- draw G in an isosceles, right-angled triangle Δ



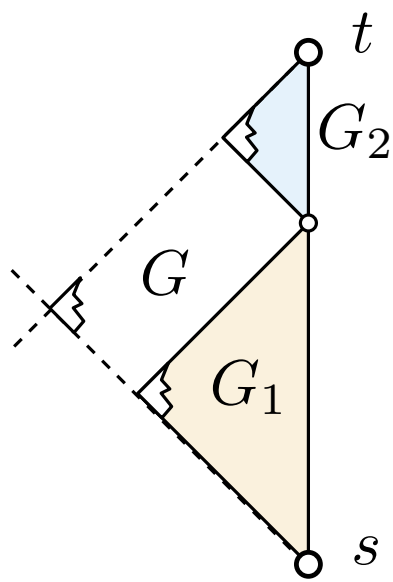
- simple graphs only (no multi-edges)

Invariants:

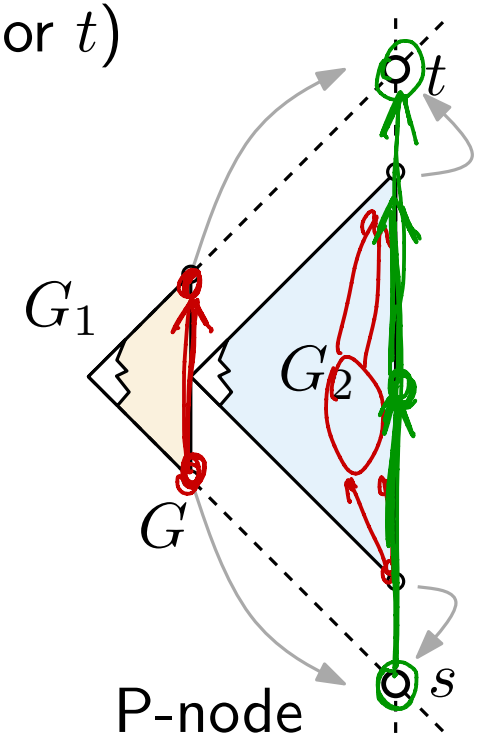
- s and t in lower/upper corner of Δ
- left corner of Δ remains free (no vertex)
- for each neighbor v of s or t, the triangle-aligned parallelogram from v contains no vertex (except s or t)



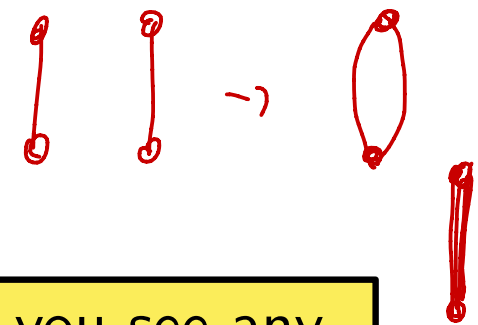
Q-node



S-node



P-node

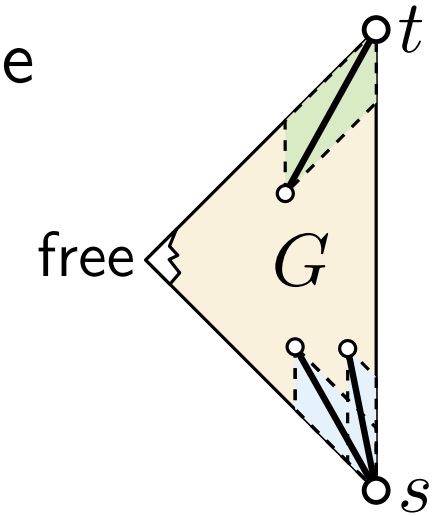


Do you see any problems?

Good News: Polynomial-Area Drawings

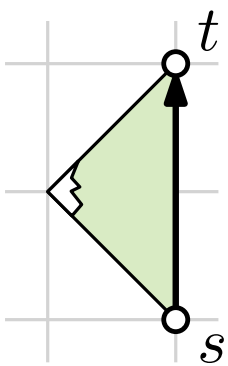
Divide & conquer algorithm using decomposition tree

- draw G in an isosceles, right-angled triangle Δ

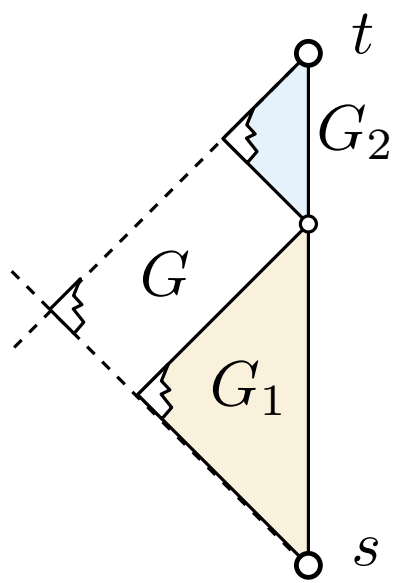


Invariants:

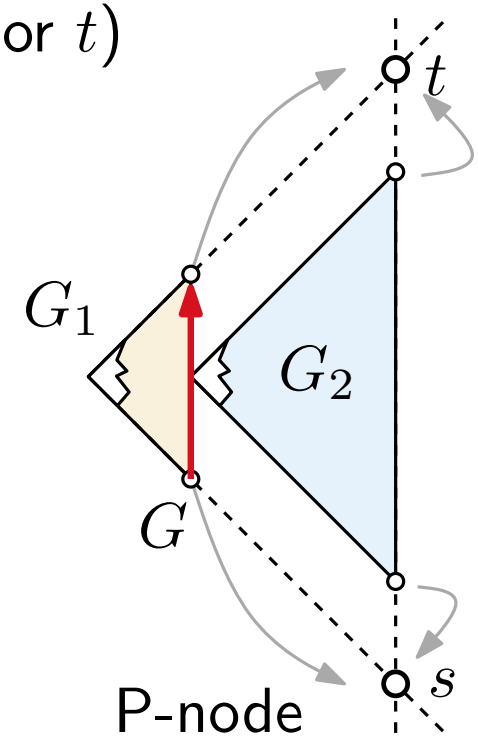
- s and t in lower/upper corner of Δ
- left corner of Δ remains free (no vertex)
- for each neighbor v of s or t, the triangle-aligned parallelogram from v contains no vertex (except s or t)



Q-node



S-node



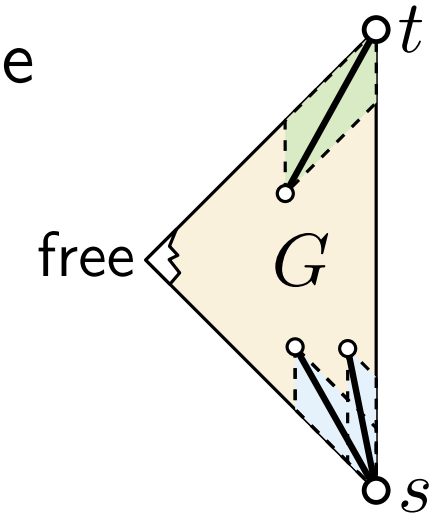
P-node

- in P-composition, G_1 cannot be a Q-node
- change subtree order (embedding) to have Q-nodes rightmost

Good News: Polynomial-Area Drawings

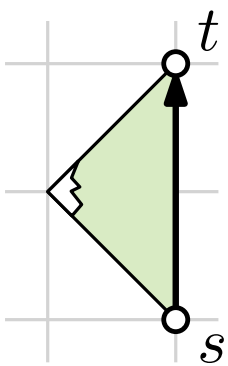
Divide & conquer algorithm using decomposition tree

- draw G in an isosceles, right-angled triangle Δ

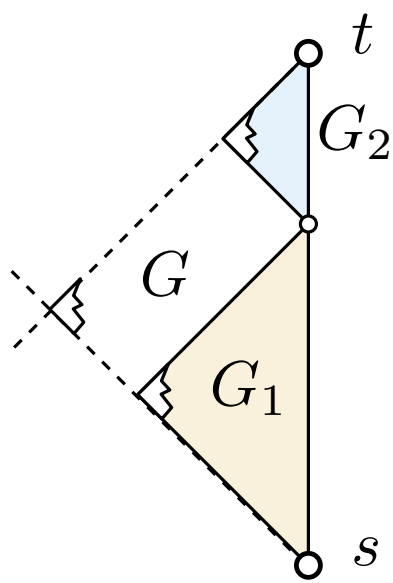


Invariants:

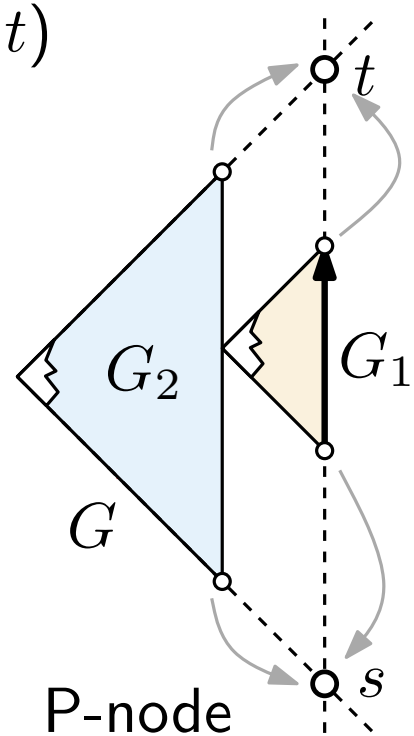
- s and t in lower/upper corner of Δ
- left corner of Δ remains free (no vertex)
- for each neighbor v of s or t, the triangle-aligned parallelogram from v contains no vertex (except s or t)



Q-node



S-node



P-node

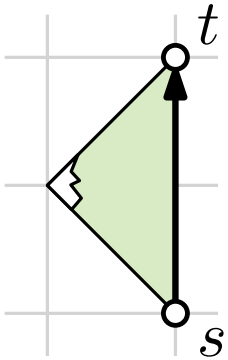
- in P-composition, G_1 cannot be a Q-node
- change subtree order (embedding) to have Q-nodes rightmost

→ **right-pushed** drawing

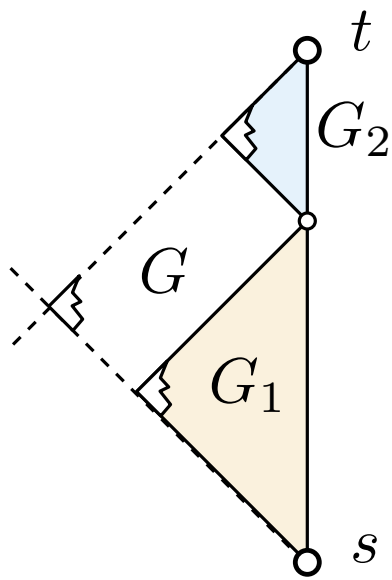
Properties of Right-Pushed Drawings

Verify Invariants, planarity, and compute area:

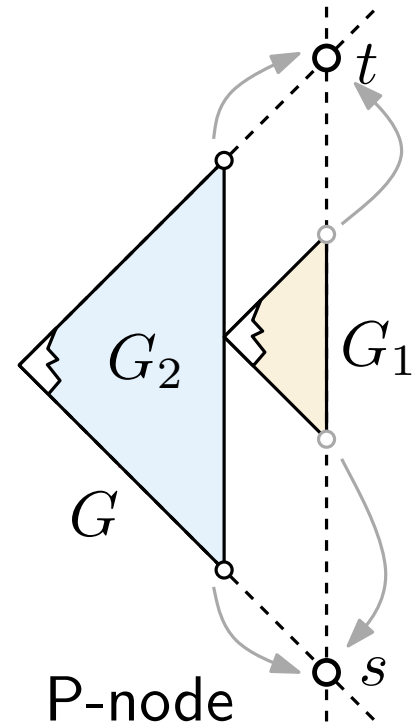
- s and t in lower/upper corner of Δ
- left corner of Δ remains free (no vertex)
- for each neighbor v of s or t, the triangle-aligned parallelogram from v contains no vertex (except s or t)



Q-node



S-node

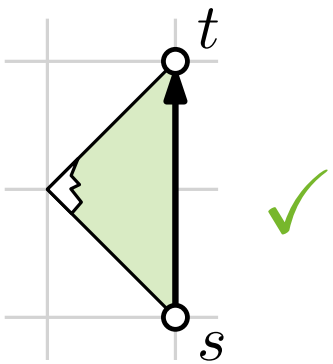


P-node

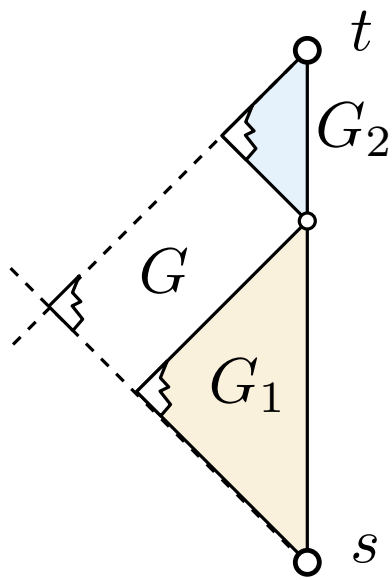
Properties of Right-Pushed Drawings

Verify Invariants, planarity, and compute area:

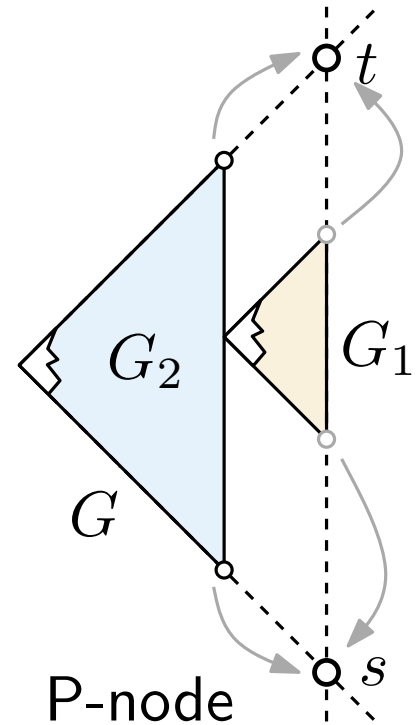
- s and t in lower/upper corner of Δ
- left corner of Δ remains free (no vertex)
- for each neighbor v of s or t, the triangle-aligned parallelogram from v contains no vertex (except s or t)



Q-node



S-node

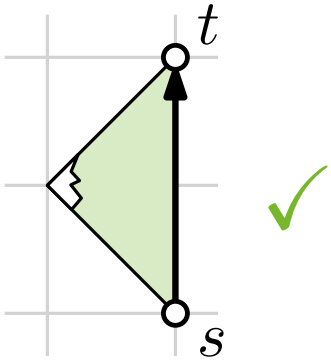


P-node

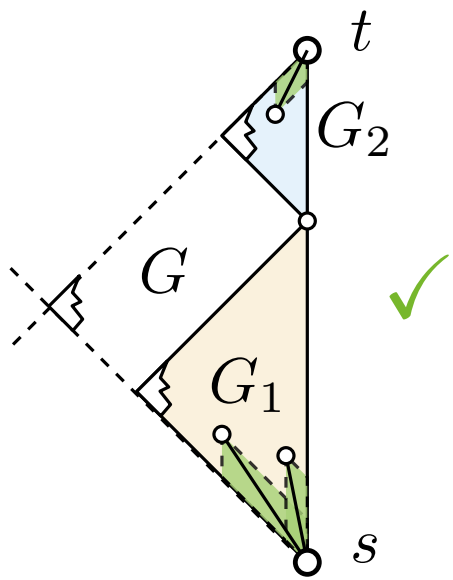
Properties of Right-Pushed Drawings

Verify Invariants, planarity, and compute area:

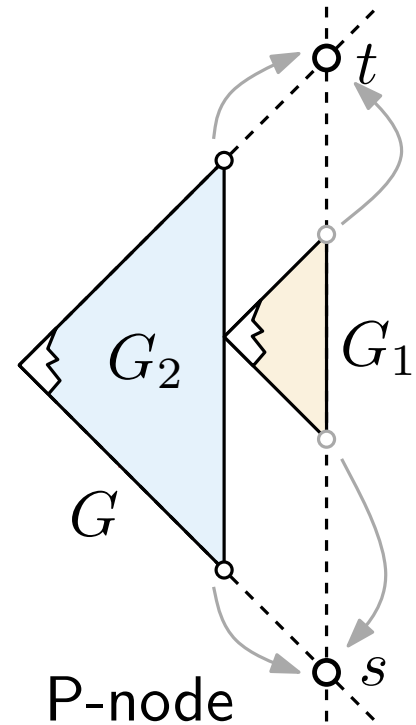
- s and t in lower/upper corner of Δ
- left corner of Δ remains free (no vertex)
- for each neighbor v of s or t, the triangle-aligned parallelogram from v contains no vertex (except s or t)



Q-node



S-node

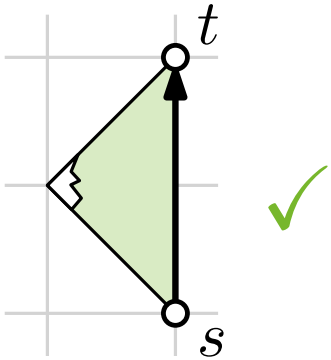


P-node

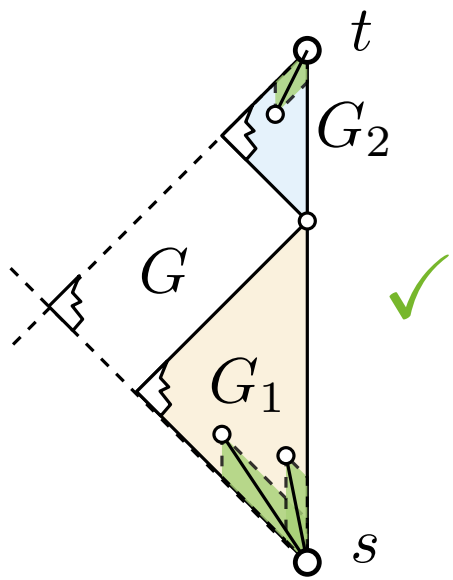
Properties of Right-Pushed Drawings

Verify Invariants, planarity, and compute area:

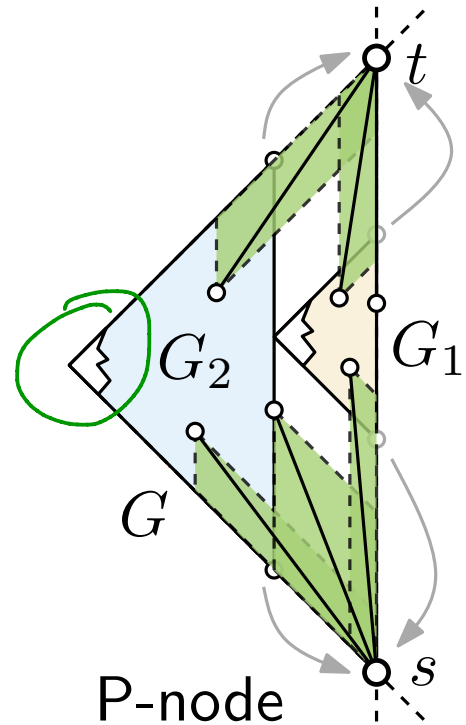
- s and t in lower/upper corner of Δ
- left corner of Δ remains free (no vertex)
- for each neighbor v of s or t, the triangle-aligned parallelogram from v contains no vertex (except s or t)



Q-node



S-node

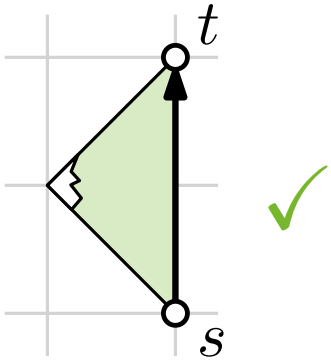


P-node

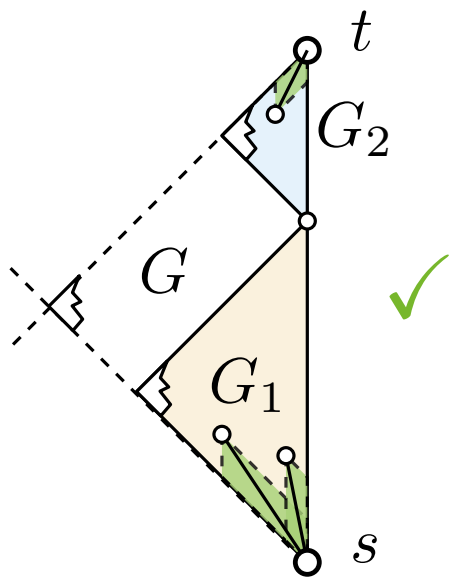
Properties of Right-Pushed Drawings

Verify Invariants, planarity, and compute area:

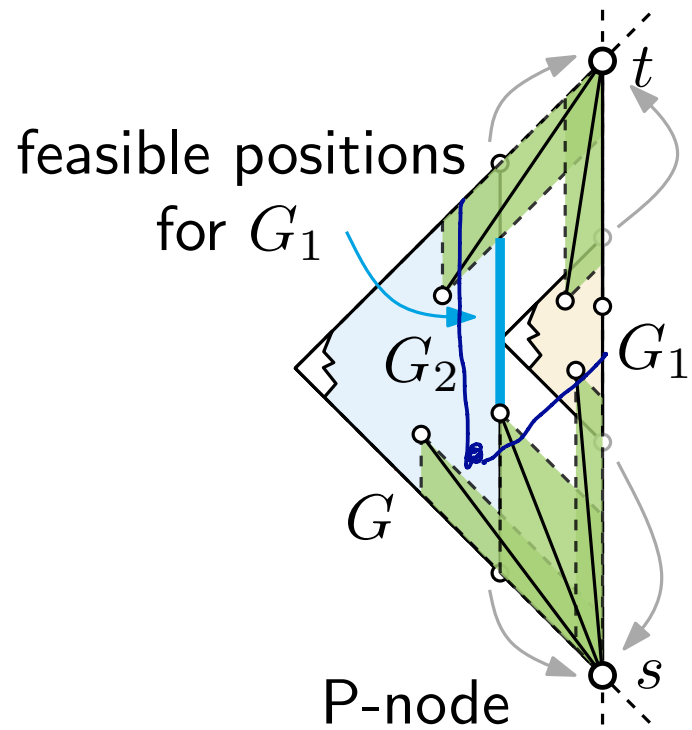
- s and t in lower/upper corner of Δ
- left corner of Δ remains free (no vertex)
- for each neighbor v of s or t, the triangle-aligned parallelogram from v contains no vertex (except s or t)



Q-node



S-node

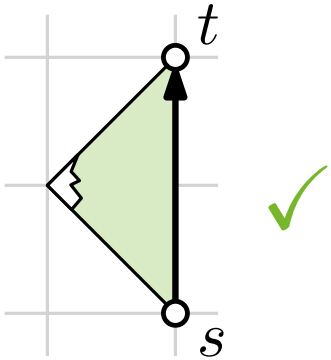


P-node

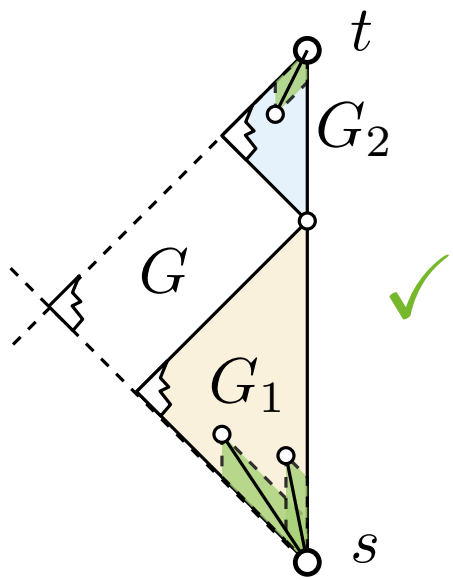
Properties of Right-Pushed Drawings

Verify Invariants, planarity, and compute area:

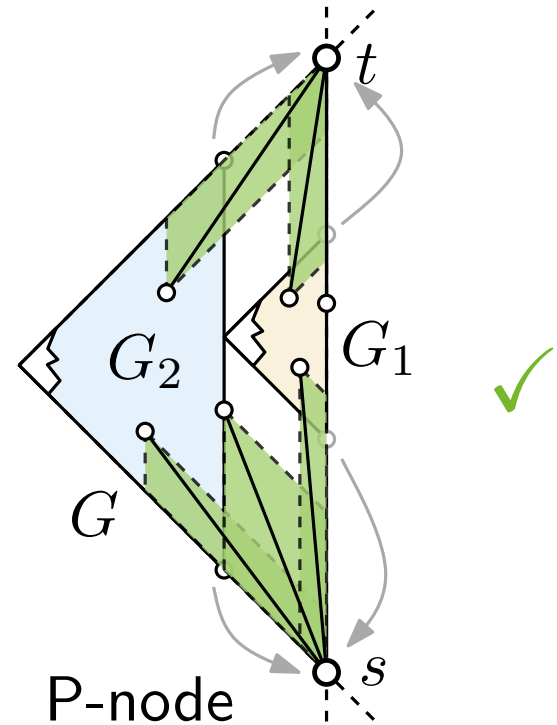
- s and t in lower/upper corner of Δ
- left corner of Δ remains free (no vertex)
- for each neighbor v of s or t, the triangle-aligned parallelogram from v contains no vertex (except s or t)



Q-node



S-node

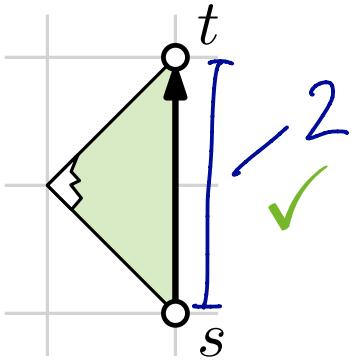


P-node

Properties of Right-Pushed Drawings

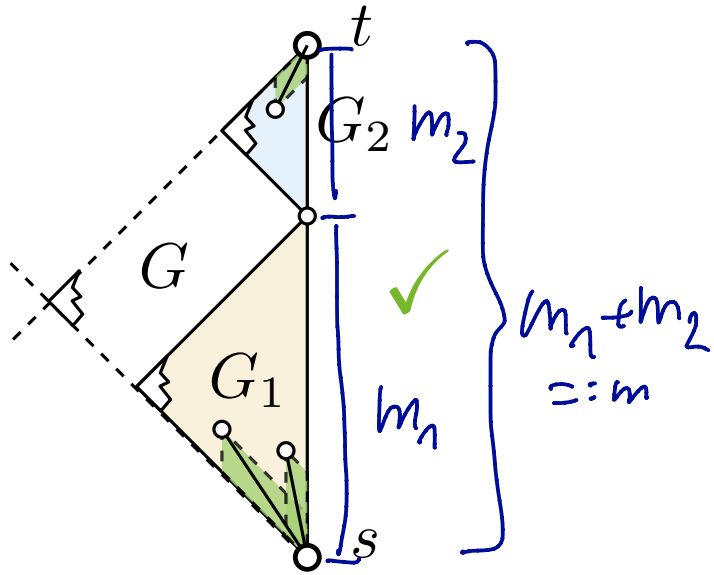
Verify Invariants, planarity, and compute area:

- s and t in lower/upper corner of Δ
- left corner of Δ remains free (no vertex)
- for each neighbor v of s or t, the triangle-aligned parallelogram from v contains no vertex (except s or t)



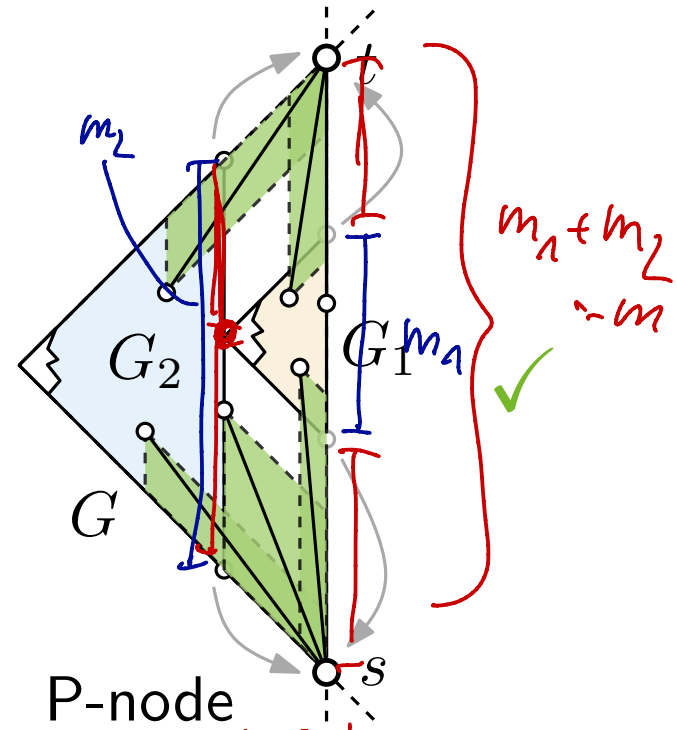
Q-node

area: 1



S-node

area: $O(m^2)$



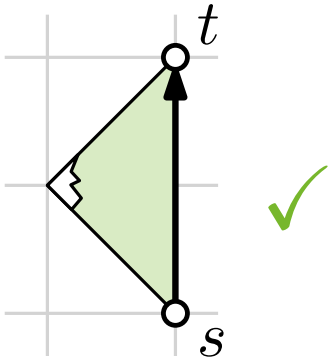
P-node

area: $O(m^2)$

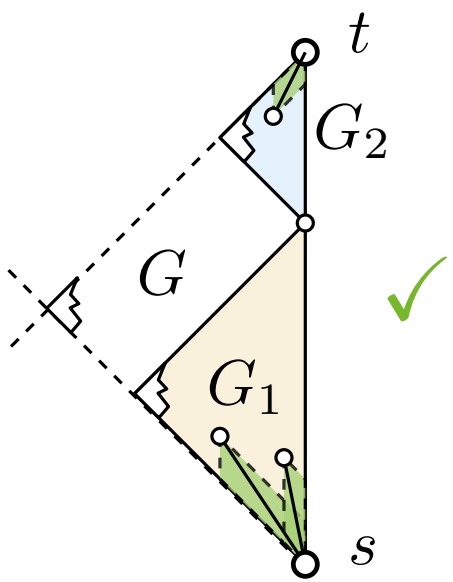
Properties of Right-Pushed Drawings

Verify Invariants, planarity, and compute area:

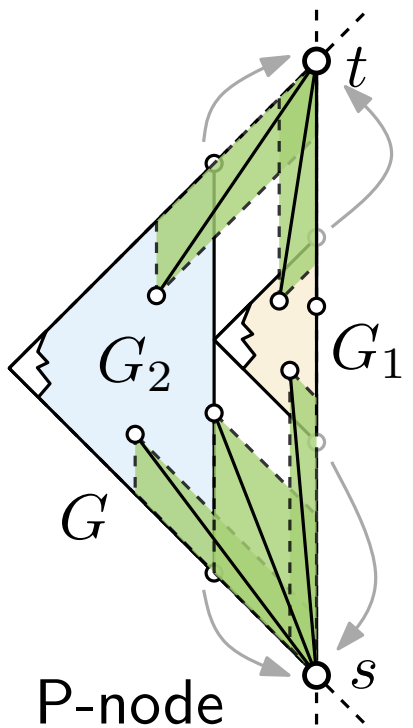
- s and t in lower/upper corner of Δ
- left corner of Δ remains free (no vertex)
- for each neighbor v of s or t, the triangle-aligned parallelogram from v contains no vertex (except s or t)



Q-node



S-node



P-node

Theorem: A simple, series-parallel graph G with m edges admits an (unordered) right-pushed upward planar drawing of area $O(m^2)$.

Summary

given: graph $G = (V, E)$
find: drawing Γ of G that

- complies with the given drawing conventions
- optimizes the given aesthetics
- satisfies the partial/local constraints

$\rightarrow$ often lead to NP-hard optimization problems!
$\rightarrow$ often several competing criteria

Summary

given: graph $G = (V, E)$
find: drawing Γ of G that

- complies with the given drawing conventions
- optimizes the given aesthetics
- satisfies the partial/local constraints

$\rightarrow$ often lead to NP-hard optimization problems!
$\rightarrow$ often several competing criteria

today:

- series-parallel (directed) graphs
- planar, straight-line upward drawings
- optimization goal: compactness/small area
- exponential lower bound for ordered (embedded) graphs
- divide & conquer algorithm for unordered graphs: $O(m^2)$ area
- competing: small area vs. ordered/preserve embedding
- *further reading:* maximize displayed symmetries via upward planar automorphism groups [Hong et al. '00]