Characterization of Unlabeled Level Planar Trees

Alejandro Estrella-Balderrama, J. Joseph Fowler, and Stephen G. Kobourov

Department of Computer Science, The University of Arizona

The 14th International Symposium on Graph Drawing (GD 2006)

Planarity – Overview

- A graph $G(V, E)$ is *planar* if and only if

Planarity – Overview

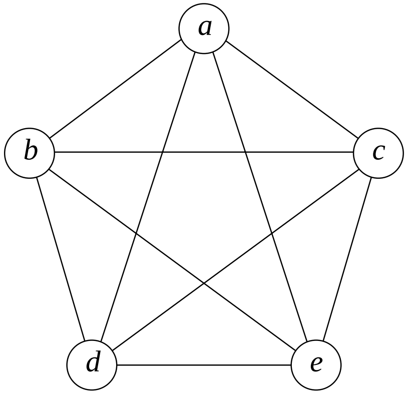
- A graph $G(V, E)$ is *planar* if and only if
  - G can be drawn in the plane without crossings

Planarity – Overview

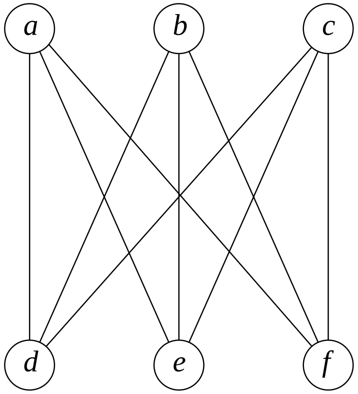
- A graph $G(V, E)$ is *planar* if and only if
  - G can be drawn in the plane without crossings
    - Edges can have curves or only be straight-line edges
    - Equivalent in that if a drawing of G with curved edges exist, then so does a straight-line drawing of G exists

Planarity – Overview

- A graph $G(V, E)$ is *planar* if and only if
  - G can be drawn in the plane without crossings
    - Edges can have curves or only be straight-line edges
    - Equivalent in that if a drawing of G with curved edges exist, then so does a straight-line drawing of G exists
  - Contains no copy of K_5 or $K_{3,3}$—Kuratowski's Theorem



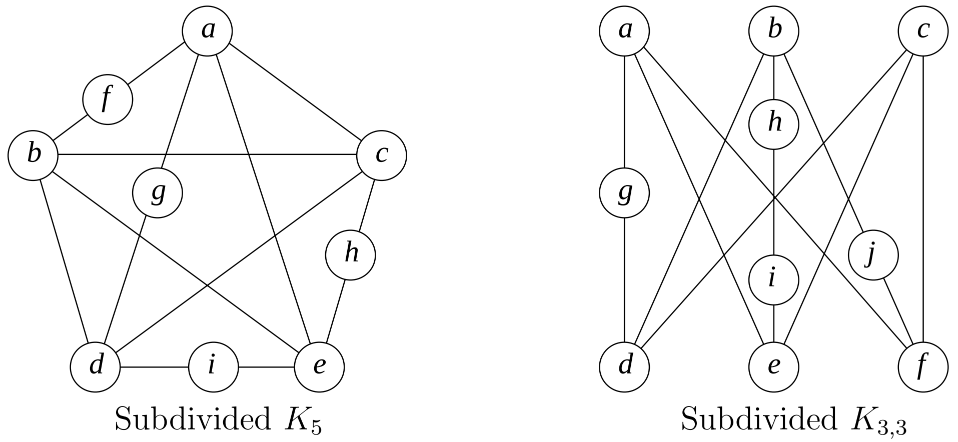
K_5



$K_{3,3}$

Planarity – Overview

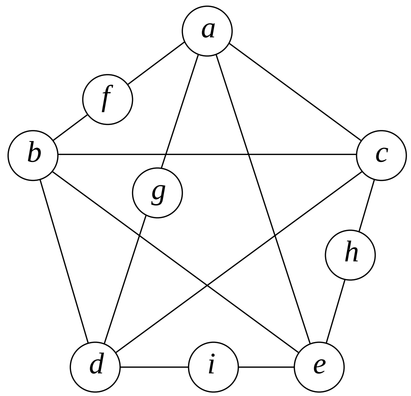
- A graph $G(V, E)$ is *planar* if and only if
  - ▶ G can be drawn in the plane without crossings
    - ♦ Edges can have curves or only be straight-line edges
    - ♦ Equivalent in that if a drawing of G with curved edges exist, then so does a straight-line drawing of G exists
  - ▶ Contains no copy of K_5 or $K_{3,3}$—Kuratowski's Theorem



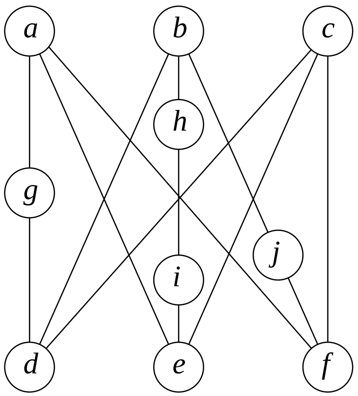
    - ♦ I.e., G does not contain a subgraph that is a subdivision of K_5 or $K_{3,3}$

Planarity – Overview

- A graph $G(V, E)$ is *planar* if and only if
  - G can be drawn in the plane without crossings
    - Edges can have curves or only be straight-line edges
    - Equivalent in that if a drawing of G with curved edges exist, then so does a straight-line drawing of G exists
  - Contains no copy of K_5 or $K_{3,3}$—Kuratowski's Theorem



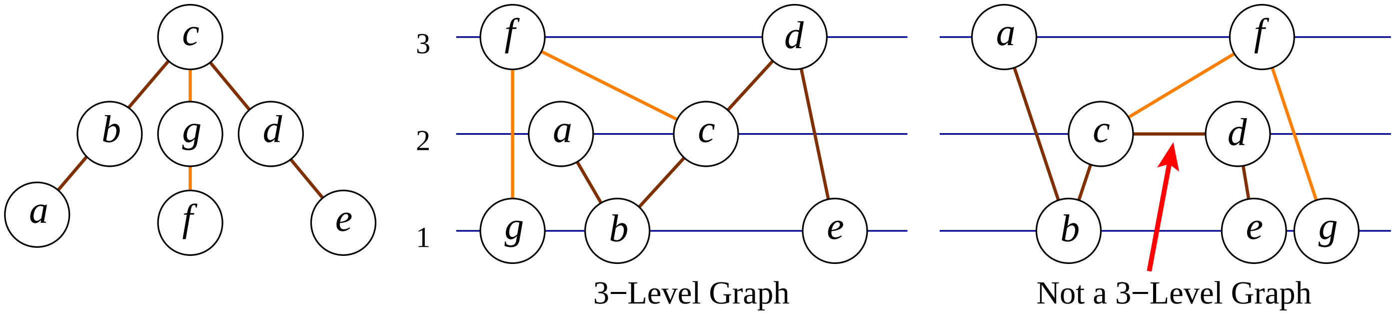
Subdivided K_5

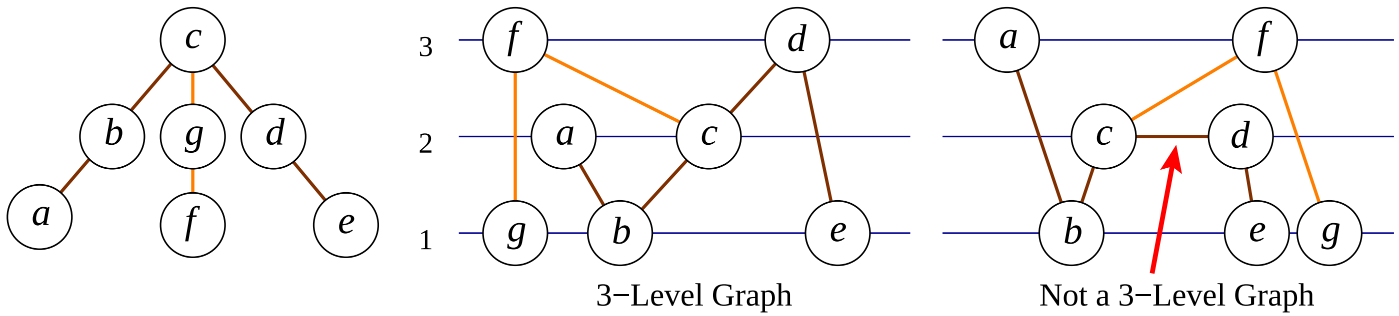


Subdivided $K_{3,3}$

    - I.e., G does not contain a subgraph that is a subdivision of K_5 or $K_{3,3}$
- Have developed similar forbidden subdivision characterization for ULP trees

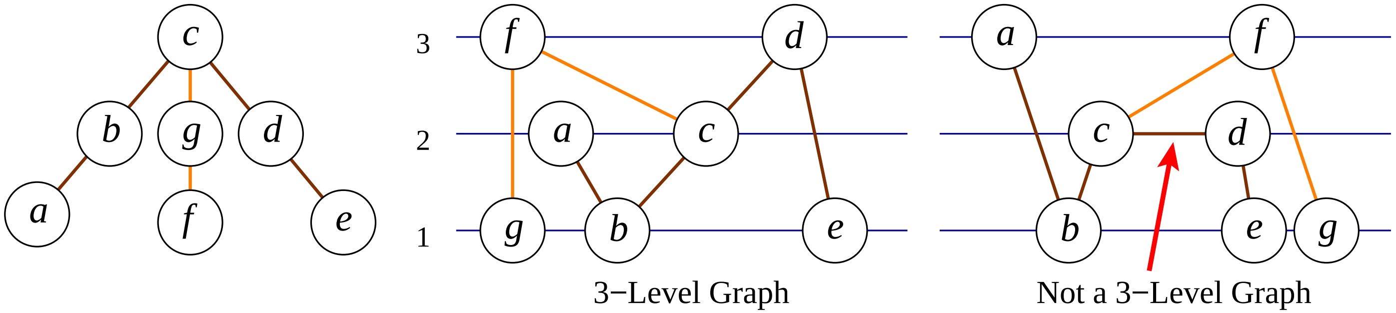
Level Planarity – Definitions





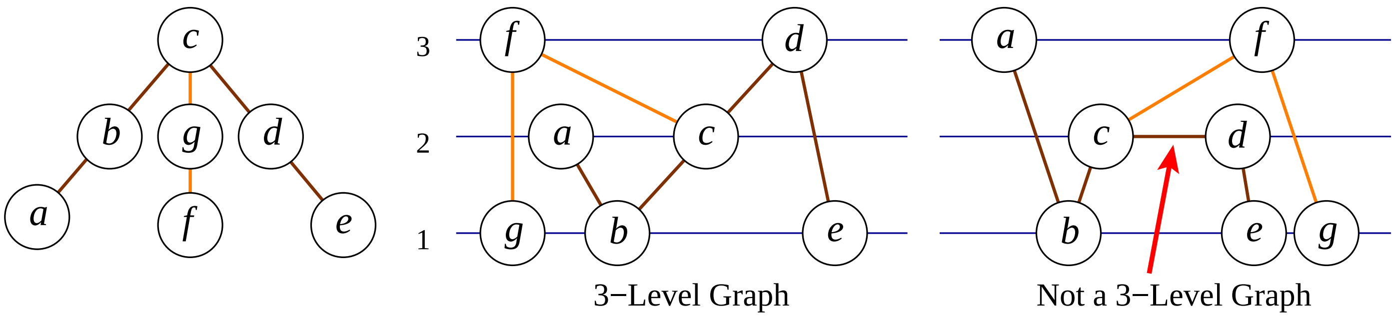
- A k-*level graph* $G(V, E, \phi)$
  - Has n vertices where $n \geq k$
  - Edges are drawn with straight-line segments
  - Has a *level assignment* $\phi : V \longrightarrow [1..k]$
    - Assigns each vertex to one of k equidistant horizontal levels
    - Cannot have an edge between two vertices on same level
    - I.e., $(u, v) \in E \Rightarrow \phi(u) \neq \phi(v)$

Level Planarity – Definitions



- A k-*level graph* $G(V, E, \phi)$
  - Has n vertices where $n \geq k$
  - Edges are drawn with straight-line segments
  - Has a *level assignment* $\phi : V \rightarrow [1..k]$
    - Assigns each vertex to one of k equidistant horizontal levels
    - Cannot have an edge between two vertices on same level
    - I.e., $(u, v) \in E \Rightarrow \phi(u) \neq \phi(v)$
  - Is *level planar* if there exists a plane drawing of G provided the y-coordinate of each $v \in V$ is $\phi(v)$
    - Placement of each vertex is restricted to its assigned level

Level Planarity – Definitions



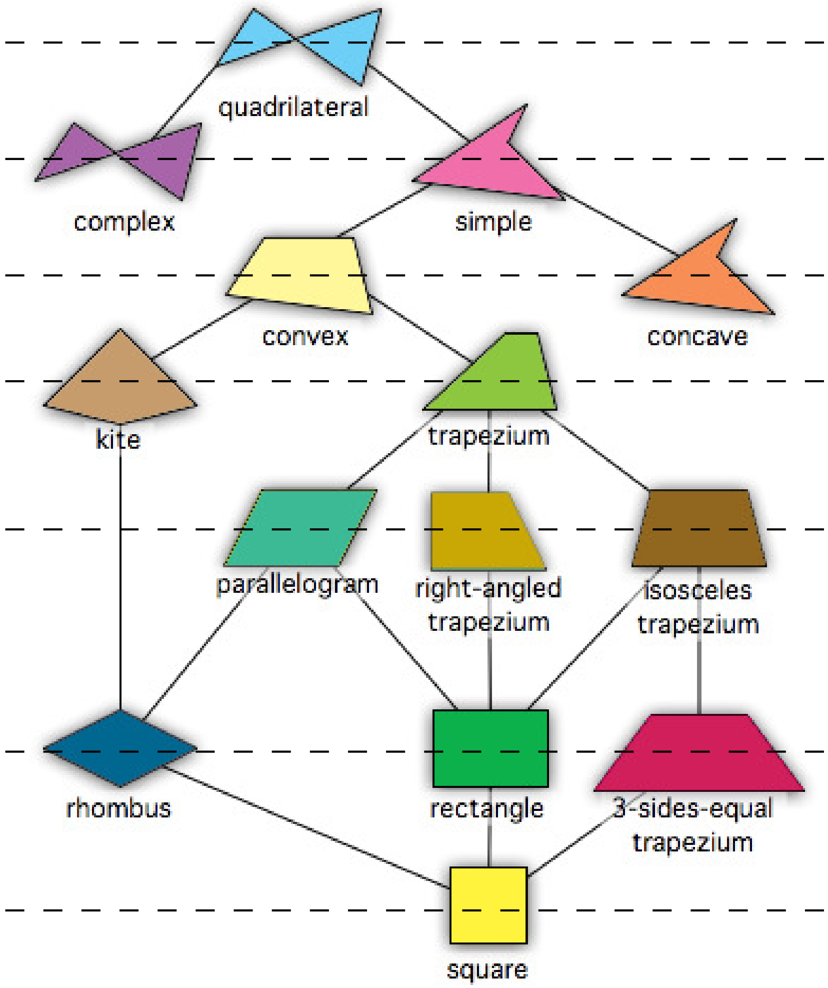
- A k-*level graph* $G(V, E, \phi)$
  - Has n vertices where $n \geq k$
  - Edges are drawn with straight-line segments
  - Has a *level assignment* $\phi : V \rightarrow [1..k]$
    - Assigns each vertex to one of k equidistant horizontal levels
    - Cannot have an edge between two vertices on same level
    - I.e., $(u, v) \in E \Rightarrow \phi(u) \neq \phi(v)$
  - Is *level planar* if there exists a plane drawing of G provided the y-coordinate of each $v \in V$ is $\phi(v)$
    - Placement of each vertex is restricted to its assigned level
  - Such a plane drawing forms a *realization* of G

Level Planarity – Motivation

- Useful in visualizing hierarchical models and relationships

Level Planarity – Motivation

- Useful in visualizing hierarchical models and relationships
  - Many natural examples of hierarchies



Level Planarity – Motivation

- Useful in visualizing hierarchical models and relationships
  - Many natural examples of hierarchies
    - Biological taxonomies
    - Software engineering drawings, e.g. flow charts
    - Social networks

Level Planarity – Motivation

- Useful in visualizing hierarchical models and relationships
  - Many natural examples of hierarchies
    - Biological taxonomies
    - Software engineering drawings, e.g. flow charts
    - Social networks
  - *Hierarchies* are level graphs

Level Planarity – Motivation

- Useful in visualizing hierarchical models and relationships
  - Many natural examples of hierarchies
    - Biological taxonomies
    - Software engineering drawings, e.g. flow charts
    - Social networks
  - *Hierarchies* are level graphs
    - Have a single source vertex on level 1
    - All the edges are directed from higher to lower levels
    - There exists a monotonic path from the source to every other vertex

Level Planarity – Motivation

- Useful in visualizing hierarchical models and relationships
  - Many natural examples of hierarchies
    - Biological taxonomies
    - Software engineering drawings, e.g. flow charts
    - Social networks
  - *Hierarchies* are level graphs
    - Have a single source vertex on level 1
    - All the edges are directed from higher to lower levels
    - There exists a monotonic path from the source to every other vertex
  - Any directed acyclic graph DAG can be visualized as a hierarchy

Level Planarity – Motivation

- Useful in visualizing hierarchical models and relationships
  - Many natural examples of hierarchies
    - Biological taxonomies
    - Software engineering drawings, e.g. flow charts
    - Social networks
  - *Hierarchies* are level graphs
    - Have a single source vertex on level 1
    - All the edges are directed from higher to lower levels
    - There exists a monotonic path from the source to every other vertex
  - Any directed acyclic graph DAG can be visualized as a hierarchy
- Application within automated graph drawing

Level Planarity – Motivation

- Useful in visualizing hierarchical models and relationships
  - Many natural examples of hierarchies
    - Biological taxonomies
    - Software engineering drawings, e.g. flow charts
    - Social networks
  - *Hierarchies* are level graphs
    - Have a single source vertex on level 1
    - All the edges are directed from higher to lower levels
    - There exists a monotonic path from the source to every other vertex
  - Any directed acyclic graph DAG can be visualized as a hierarchy
- Application within automated graph drawing
  - Sugiyama's algorithm draws DAG's in a top-down manner
    - Assigns compatible sets of vertices to the same rank or level

Level Planarity – Motivation

- Useful in visualizing hierarchical models and relationships
  - Many natural examples of hierarchies
    - Biological taxonomies
    - Software engineering drawings, e.g. flow charts
    - Social networks
  - *Hierarchies* are level graphs
    - Have a single source vertex on level 1
    - All the edges are directed from higher to lower levels
    - There exists a monotonic path from the source to every other vertex
  - Any directed acyclic graph DAG can be visualized as a hierarchy
- Application within automated graph drawing
  - Sugiyama's algorithm draws DAG's in a top-down manner
    - Assigns compatible sets of vertices to the same rank or level
  - Often the desire is to use as few levels as possible

Level Planarity – Motivation

- Useful in visualizing hierarchical models and relationships
  - Many natural examples of hierarchies
    - Biological taxonomies
    - Software engineering drawings, e.g. flow charts
    - Social networks
  - *Hierarchies* are level graphs
    - Have a single source vertex on level 1
    - All the edges are directed from higher to lower levels
    - There exists a monotonic path from the source to every other vertex
  - Any directed acyclic graph DAG can be visualized as a hierarchy
- Application within automated graph drawing
  - Sugiyama's algorithm draws DAG's in a top-down manner
    - Assigns compatible sets of vertices to the same rank or level
  - Often the desire is to use as few levels as possible
  - Finding a k-level assignment for which a graph is level planar is NP-hard

Level Planarity – Previous Work

- $O(n)$ time recognition, embedding and drawing algorithms for level graphs

Level Planarity – Previous Work

- $O(n)$ time recognition, embedding and drawing algorithms for level graphs
  - Jünger, Leipert, and Mutzel gave a linear time recognition algorithm at GD'98
    - Based on the level planarity test by Heath and Pemmaraju at GD'96
    - Extends PQ-tree planarity test of hierarchies by Di Battista and Nardelli in 1988

Level Planarity – Previous Work

- $O(n)$ time recognition, embedding and drawing algorithms for level graphs
  - Jünger, Leipert, and Mutzel gave a linear time recognition algorithm at GD'98
    - Based on the level planarity test by Heath and Pemmaraju at GD'96
    - Extends PQ-tree planarity test of hierarchies by Di Battista and Nardelli in 1988
  - Jünger and Leipert achieved linear time level planar embedding at GD'99
    - Outputs a set of linear orderings of the vertices on each level

Level Planarity – Previous Work

- $O(n)$ time recognition, embedding and drawing algorithms for level graphs
  - Jünger, Leipert, and Mutzel gave a linear time recognition algorithm at GD'98
    - Based on the level planarity test by Heath and Pemmaraju at GD'96
    - Extends PQ-tree planarity test of hierarchies by Di Battista and Nardelli in 1988
  - Jünger and Leipert achieved linear time level planar embedding at GD'99
    - Outputs a set of linear orderings of the vertices on each level
  - Eades, Feng, Lin, and Nagamochi Jünger and Leipert devised a straight-line level planar drawing algorithm if the input level graph is level planar
    - Initially ran in $O(|V|^2)$ time in 1997, improved to $O(|V|)$ time in 2006

Level Planarity – Previous Work

- $O(n)$ time recognition, embedding and drawing algorithms for level graphs
  - Jünger, Leipert, and Mutzel gave a linear time recognition algorithm at GD'98
    - Based on the level planarity test by Heath and Pemmaraju at GD'96
    - Extends PQ-tree planarity test of hierarchies by Di Battista and Nardelli in 1988
  - Jünger and Leipert achieved linear time level planar embedding at GD'99
    - Outputs a set of linear orderings of the vertices on each level
  - Eades, Feng, Lin, and Nagamochi Jünger and Leipert devised a straight-line level planar drawing algorithm if the input level graph is level planar
    - Initially ran in $O(|V|^2)$ time in 1997, improved to $O(|V|)$ time in 2006
- Characterizations of level graphs

Level Planarity – Previous Work

- $O(n)$ time recognition, embedding and drawing algorithms for level graphs
  - Jünger, Leipert, and Mutzel gave a linear time recognition algorithm at GD'98
    - Based on the level planarity test by Heath and Pemmaraju at GD'96
    - Extends PQ-tree planarity test of hierarchies by Di Battista and Nardelli in 1988
  - Jünger and Leipert achieved linear time level planar embedding at GD'99
    - Outputs a set of linear orderings of the vertices on each level
  - Eades, Feng, Lin, and Nagamochi Jünger and Leipert devised a straight-line level planar drawing algorithm if the input level graph is level planar
    - Initially ran in $O(|V|^2)$ time in 1997, improved to $O(|V|)$ time in 2006
- Characterizations of level graphs
  - Di Battista and Nardelli provided a characterization of hierarchies in 1988
    - Uses level non-planar (LNP) patterns

Level Planarity – Previous Work

- $O(n)$ time recognition, embedding and drawing algorithms for level graphs
  - Jünger, Leipert, and Mutzel gave a linear time recognition algorithm at GD'98
    - Based on the level planarity test by Heath and Pemmaraju at GD'96
    - Extends PQ-tree planarity test of hierarchies by Di Battista and Nardelli in 1988
  - Jünger and Leipert achieved linear time level planar embedding at GD'99
    - Outputs a set of linear orderings of the vertices on each level
  - Eades, Feng, Lin, and Nagamochi Jünger and Leipert devised a straight-line level planar drawing algorithm if the input level graph is level planar
    - Initially ran in $O(|V|^2)$ time in 1997, improved to $O(|V|)$ time in 2006
- Characterizations of level graphs
  - Di Battista and Nardelli provided a characterization of hierarchies in 1988
    - Uses level non-planar (LNP) patterns
  - Healy, Kuusik, and Leipert found minimal LNP subgraph patterns at CC'00
    - Patterns analogous to Kuratowski's subgraphs of regular planar graphs

Level Planarity – Previous Work

- $O(n)$ time recognition, embedding and drawing algorithms for level graphs
  - Jünger, Leipert, and Mutzel gave a linear time recognition algorithm at GD'98
    - Based on the level planarity test by Heath and Pemmaraju at GD'96
    - Extends PQ-tree planarity test of hierarchies by Di Battista and Nardelli in 1988
  - Jünger and Leipert achieved linear time level planar embedding at GD'99
    - Outputs a set of linear orderings of the vertices on each level
  - Eades, Feng, Lin, and Nagamochi Jünger and Leipert devised a straight-line level planar drawing algorithm if the input level graph is level planar
    - Initially ran in $O(|V|^2)$ time in 1997, improved to $O(|V|)$ time in 2006
- Characterizations of level graphs
  - Di Battista and Nardelli provided a characterization of hierarchies in 1988
    - Uses level non-planar (LNP) patterns
  - Healy, Kuusik, and Leipert found minimal LNP subgraph patterns at CC'00
    - Patterns analogous to Kuratowski's subgraphs of regular planar graphs
  - All these characterizations are for a *single* level assignment

Unlabeled Level Planarity – Definition

- Consider level graphs with bijective level assignments

Unlabeled Level Planarity – Definition

- Consider level graphs with bijective level assignments
  - Each vertex lies on a distinct level, i.e. $k = n$

Unlabeled Level Planarity – Definition

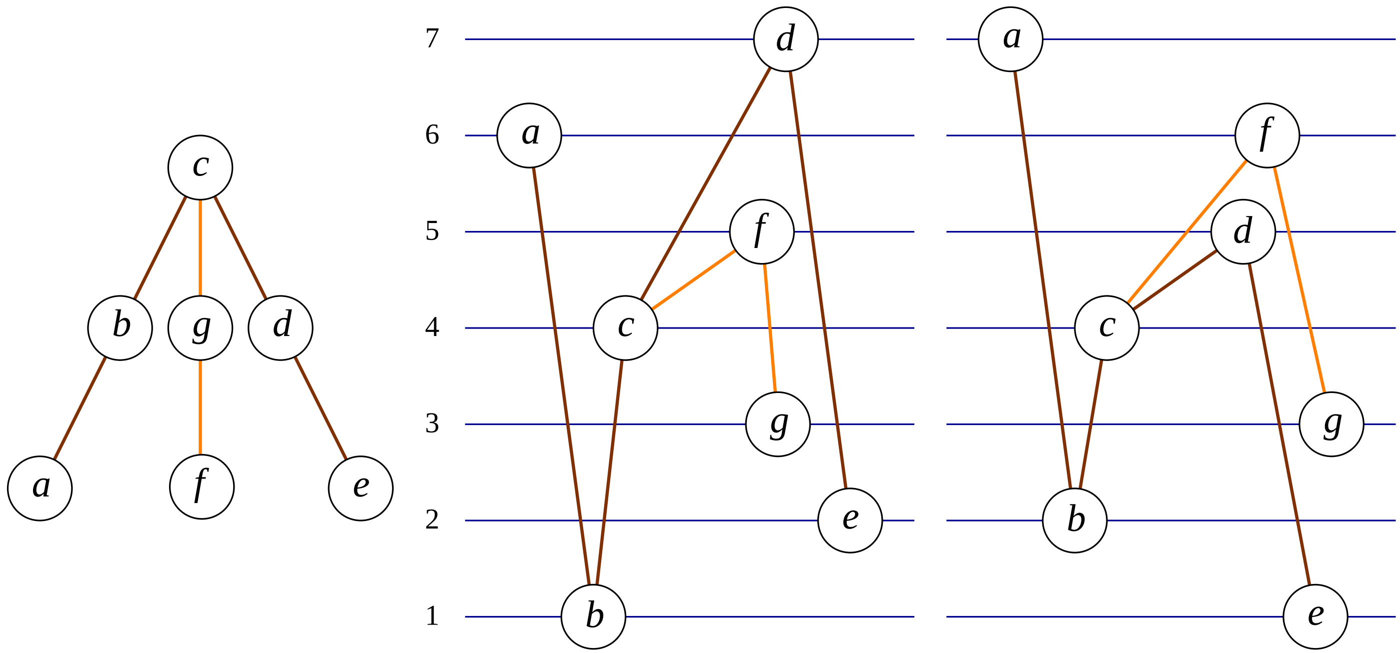
- Consider level graphs with bijective level assignments
  - Each vertex lies on a distinct level, i.e. $k = n$
  - *Every* planar graph G has *some* n-level assignment that is level planar
    - Perturb any plane drawing of G such that all the y-coordinates differ

Unlabeled Level Planarity – Definition

- Consider level graphs with bijective level assignments
  - Each vertex lies on a distinct level, i.e. $k = n$
  - *Every* planar graph G has *some* n-level assignment that is level planar
    - Perturb any plane drawing of G such that all the y-coordinates differ
  - Only *some* planar graphs are n-level planar over *every* level assignment
    - Such graphs are called Unlabeled Level Planar (ULP)

Unlabeled Level Planarity – Definition

- Consider level graphs with bijective level assignments
  - Each vertex lies on a distinct level, i.e. $k = n$
  - *Every* planar graph G has *some* n-level assignment that is level planar
    - Perturb any plane drawing of G such that all the y-coordinates differ
  - Only *some* planar graphs are n-level planar over *every* level assignment
    - Such graphs are called Unlabeled Level Planar (ULP)



Unlabeled Level Planarity – Motivation

- Application with *simultaneous embedding*

Unlabeled Level Planarity – Motivation

- Application with *simultaneous embedding*
  - Embedding of multiple planar graphs onto the same vertex set V

Unlabeled Level Planarity – Motivation

- Application with *simultaneous embedding*
  - Embedding of multiple planar graphs onto the same vertex set V
    - Has to work for *any* vertex mapping between graphs

Unlabeled Level Planarity – Motivation

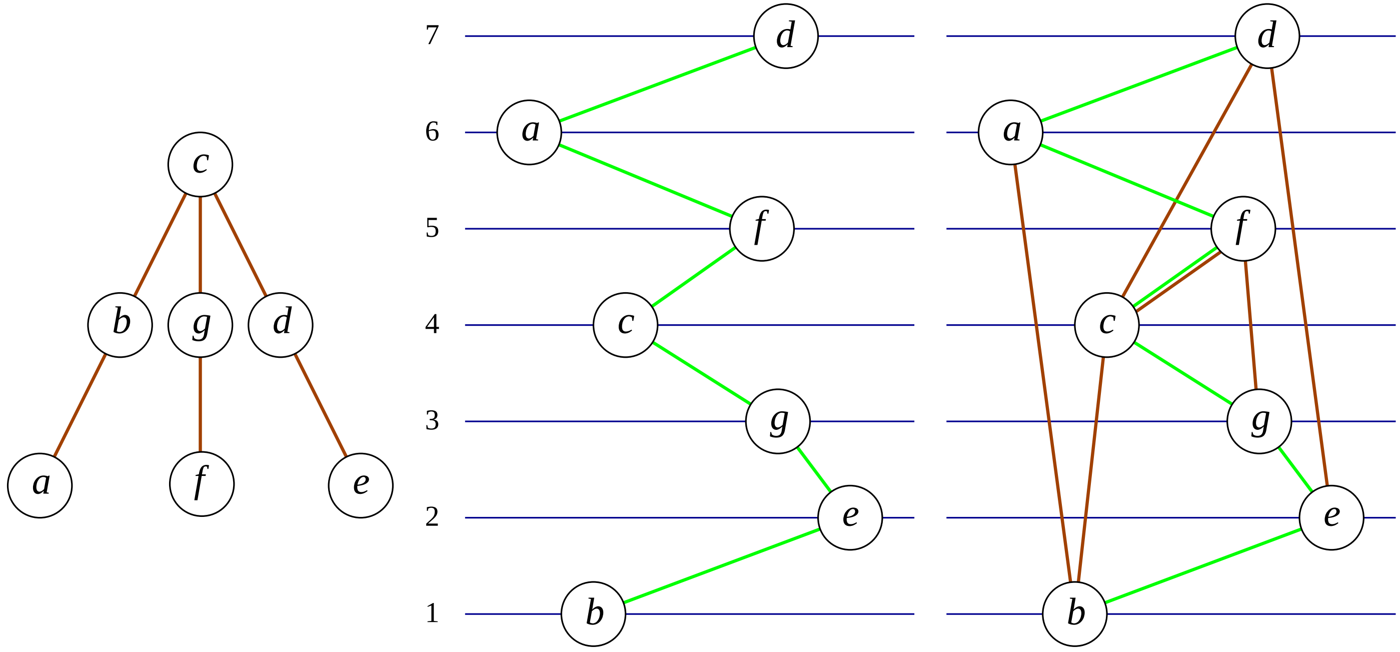
- Application with *simultaneous embedding*
  - Embedding of multiple planar graphs onto the same vertex set V
    - Has to work for *any* vertex mapping between graphs
    - Desire straight-line edges

Unlabeled Level Planarity – Motivation

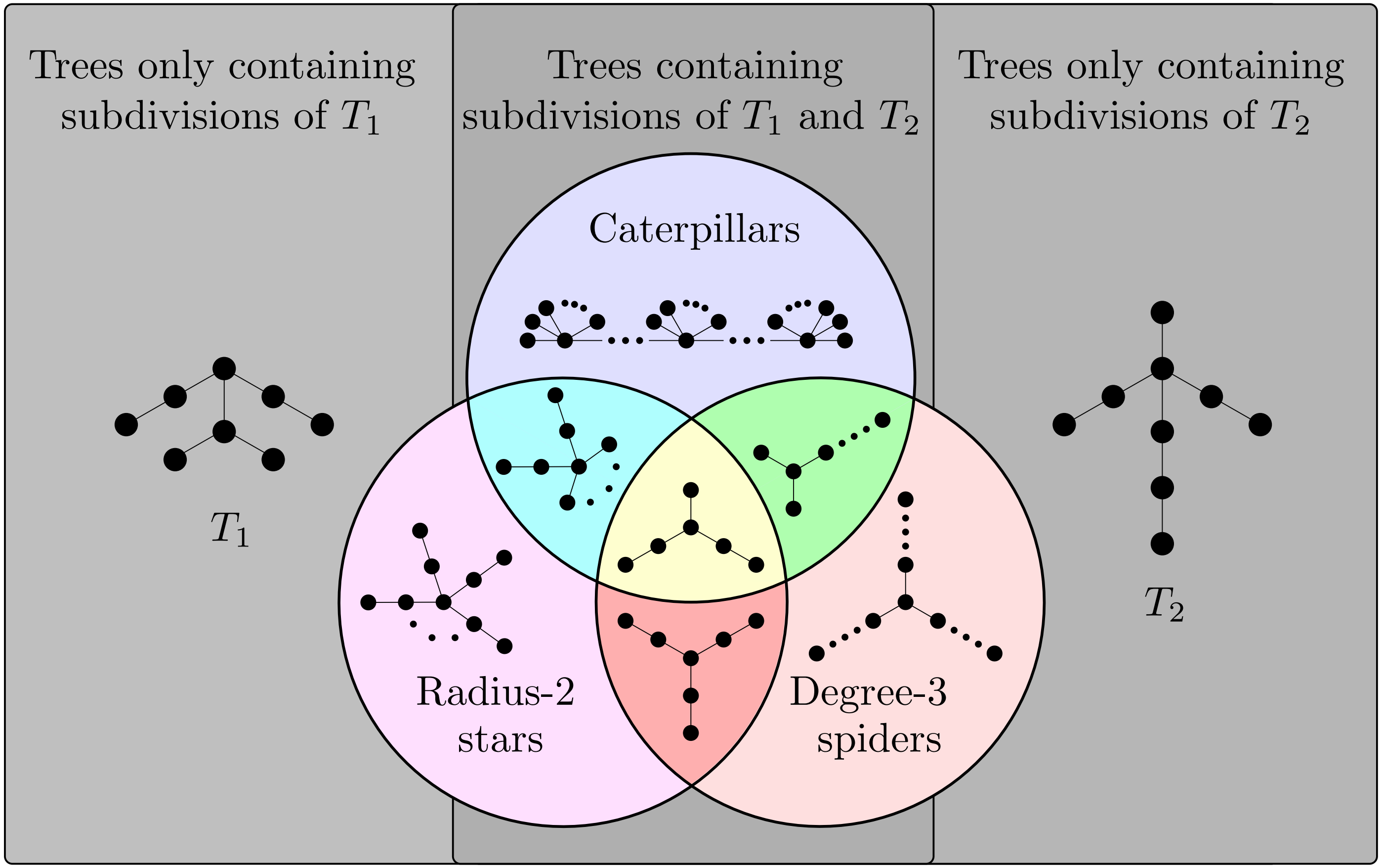
- Application with *simultaneous embedding*
  - Embedding of multiple planar graphs onto the same vertex set V
    - Has to work for *any* vertex mapping between graphs
    - Desire straight-line edges
    - No crossings allowed within each planar layer

Unlabeled Level Planarity – Motivation

- Application with *simultaneous embedding*
  - Embedding of multiple planar graphs onto the same vertex set V
    - Has to work for *any* vertex mapping between graphs
    - Desire straight-line edges
    - No crossings allowed within each planar layer
  - Can simultaneously embed a path P with any ULP graph G

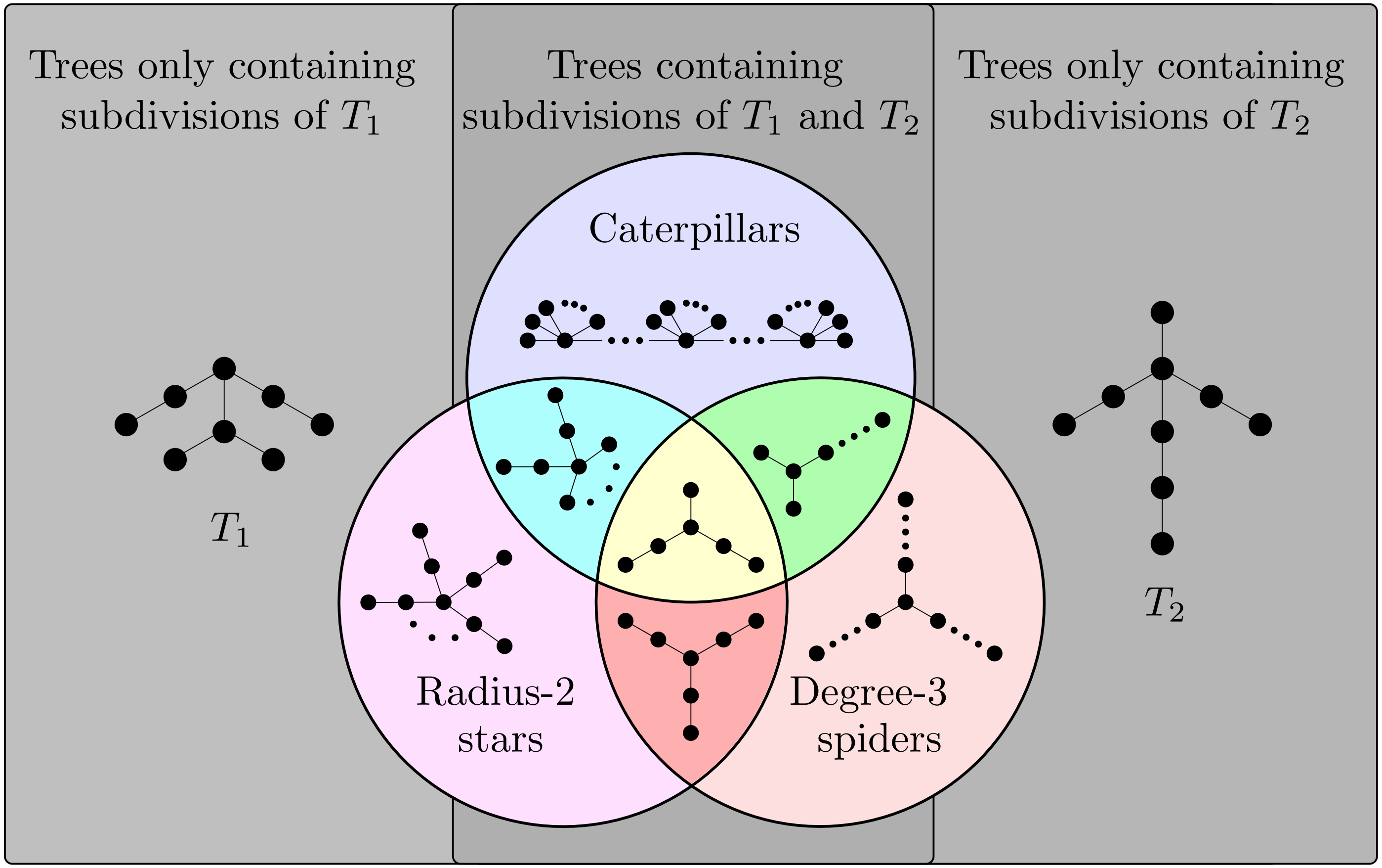


Unlabeled Level Planarity – Trees

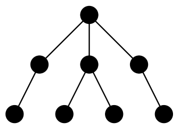


- Characterization of ULP trees by two forbidden subdivisions

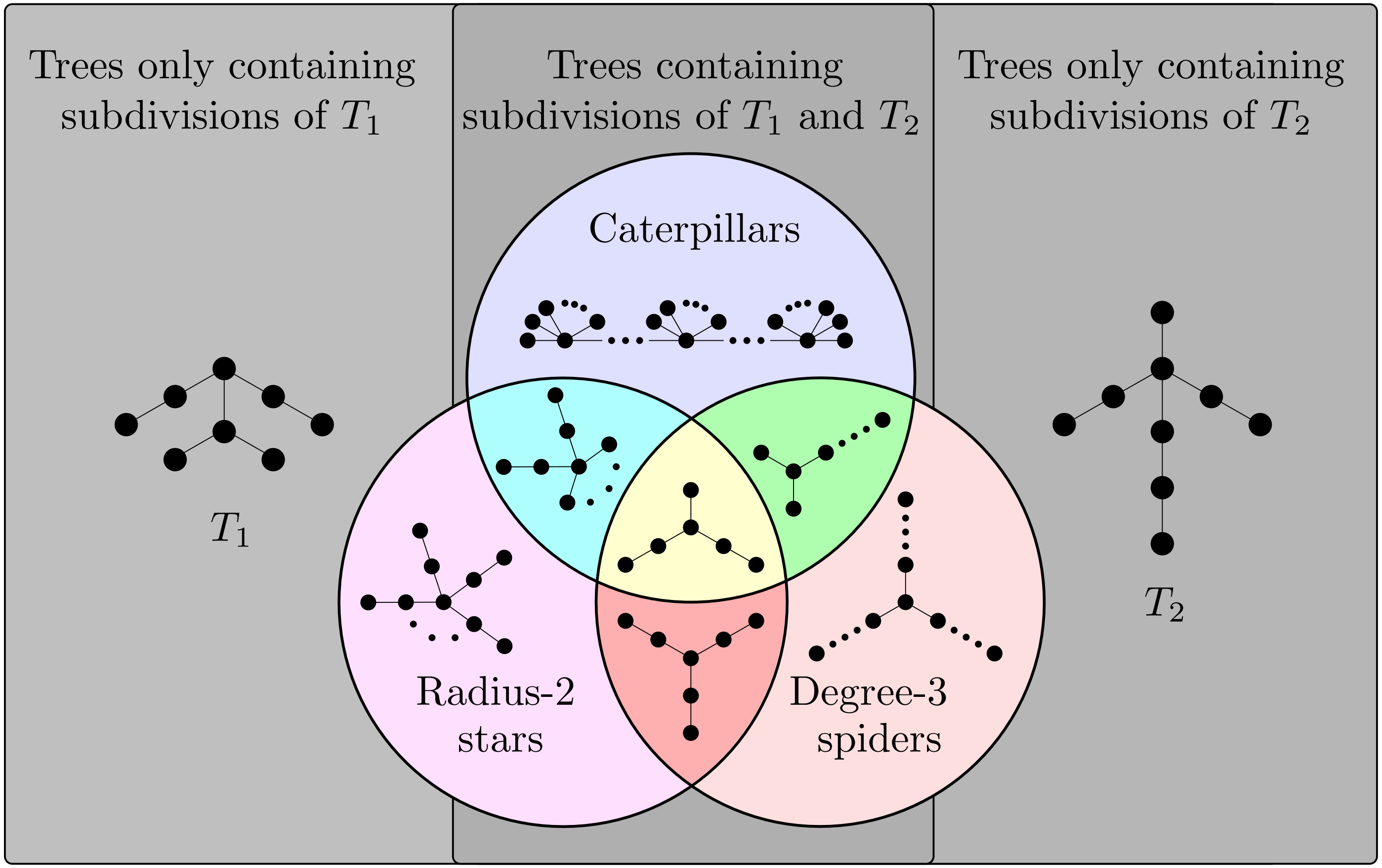
Unlabeled Level Planarity – Trees



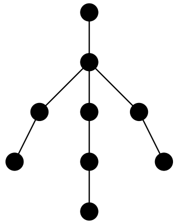
- Characterization of ULP trees by two forbidden subdivisions
  - Tree T_1 with 8 vertices and two nodes of degree 3



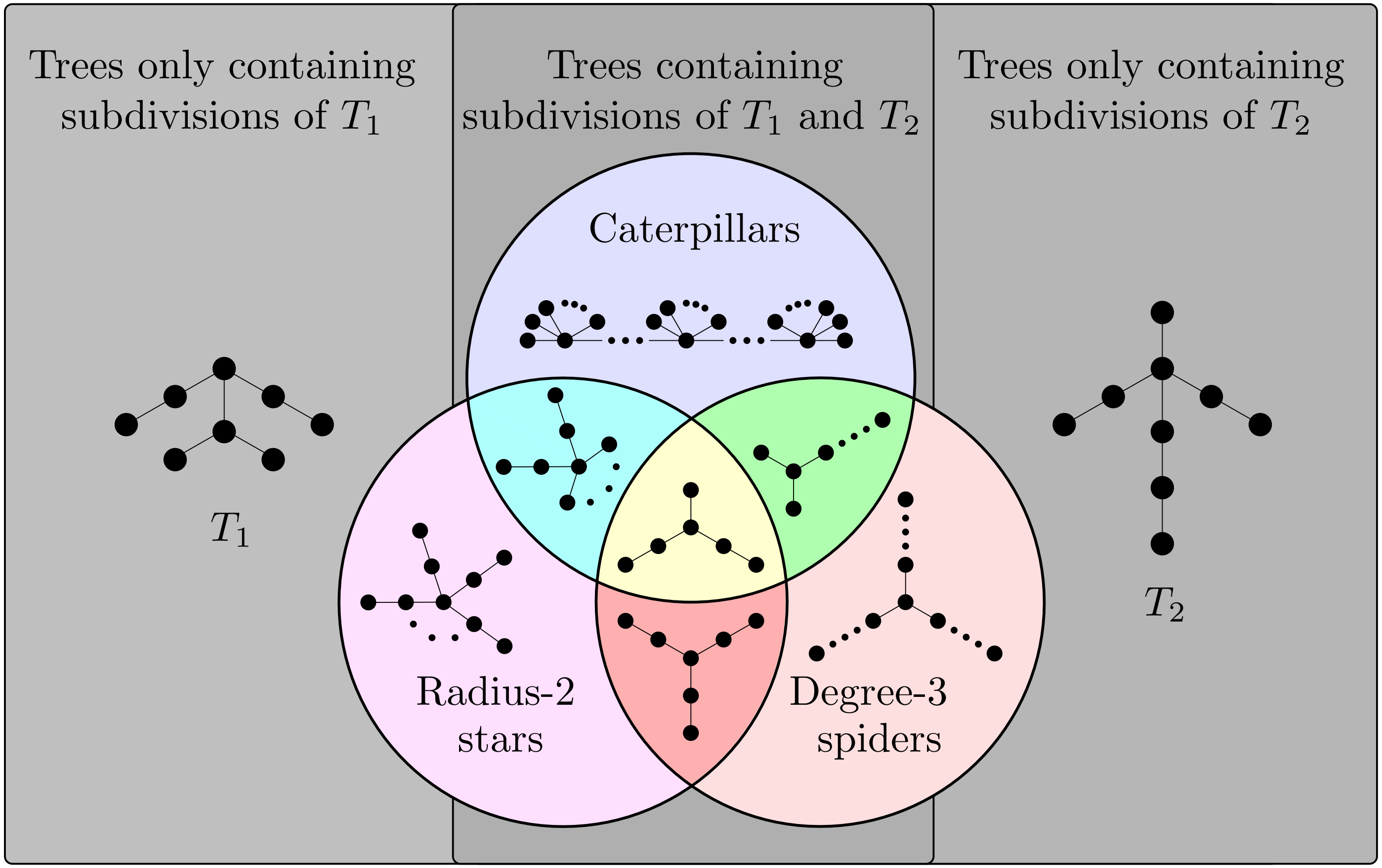
Unlabeled Level Planarity – Trees



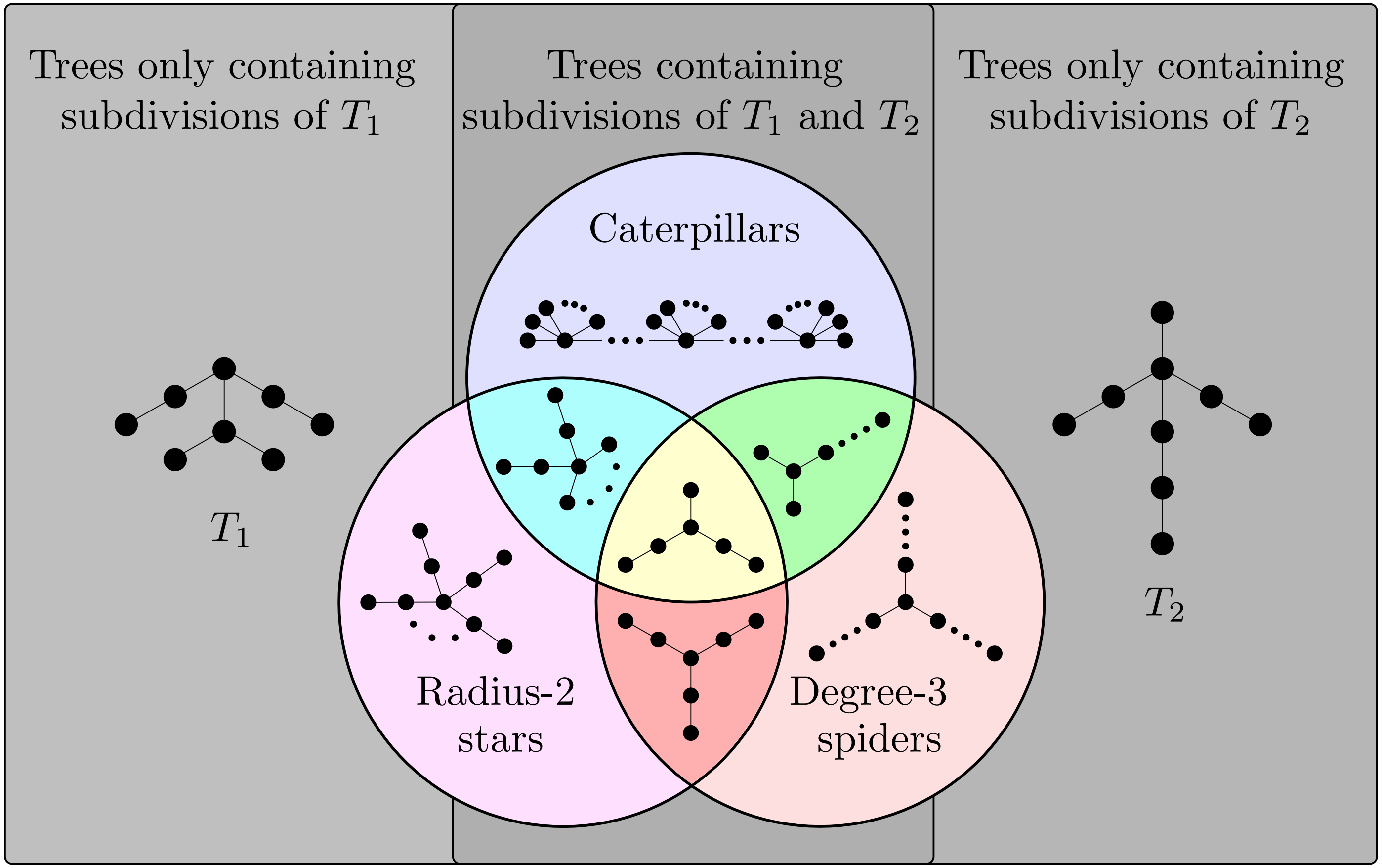
- Characterization of ULP trees by two forbidden subdivisions
  - Tree T_1 with 8 vertices and two nodes of degree 3
  - Tree T_2 with 9 vertices and one node of degree 4



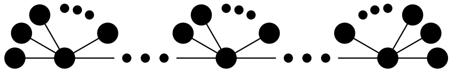
Unlabeled Level Planarity – Trees



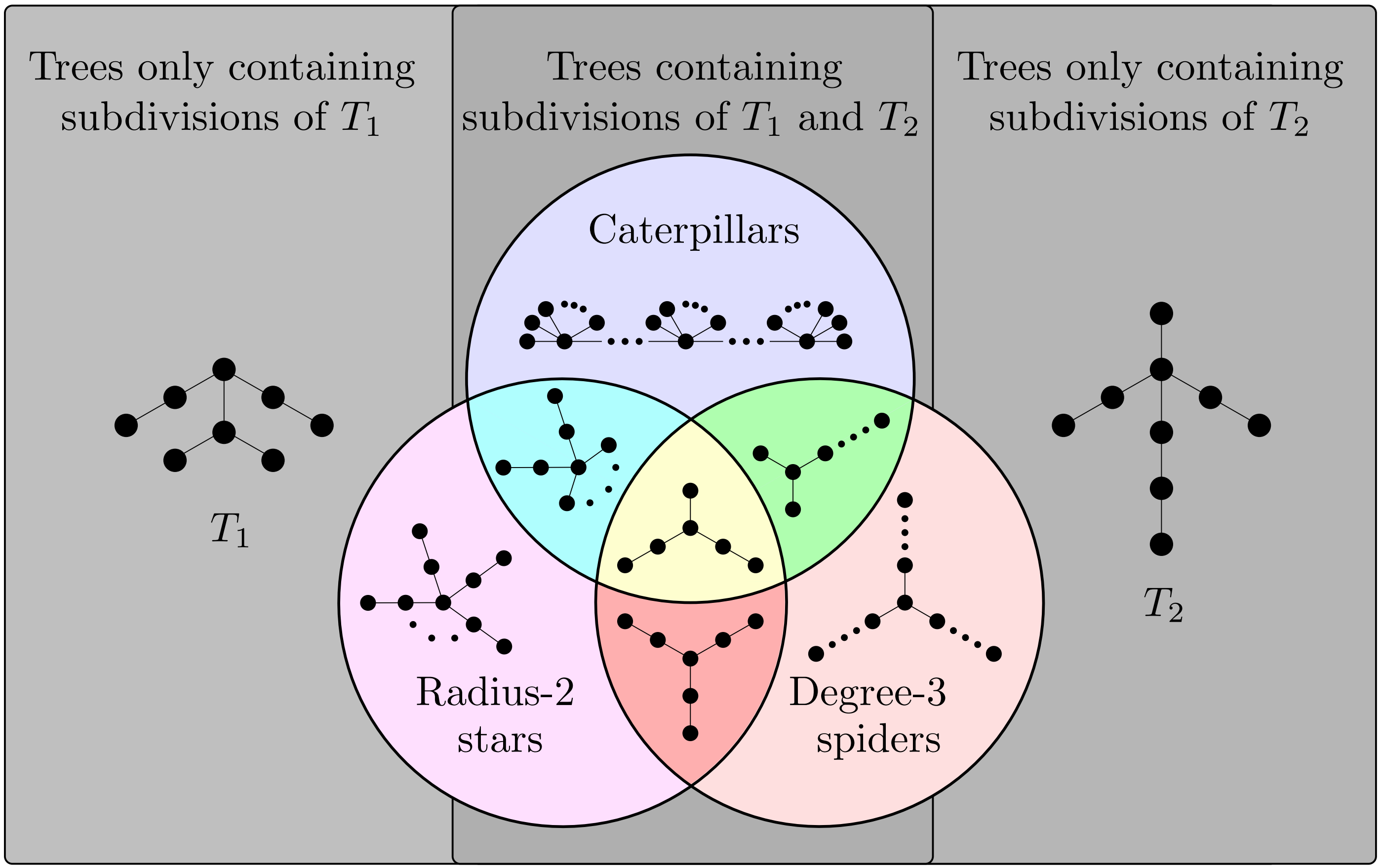
- All ULP trees fall into one of three categories:



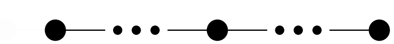
- All ULP trees fall into one of three categories:
  - **Caterpillars**



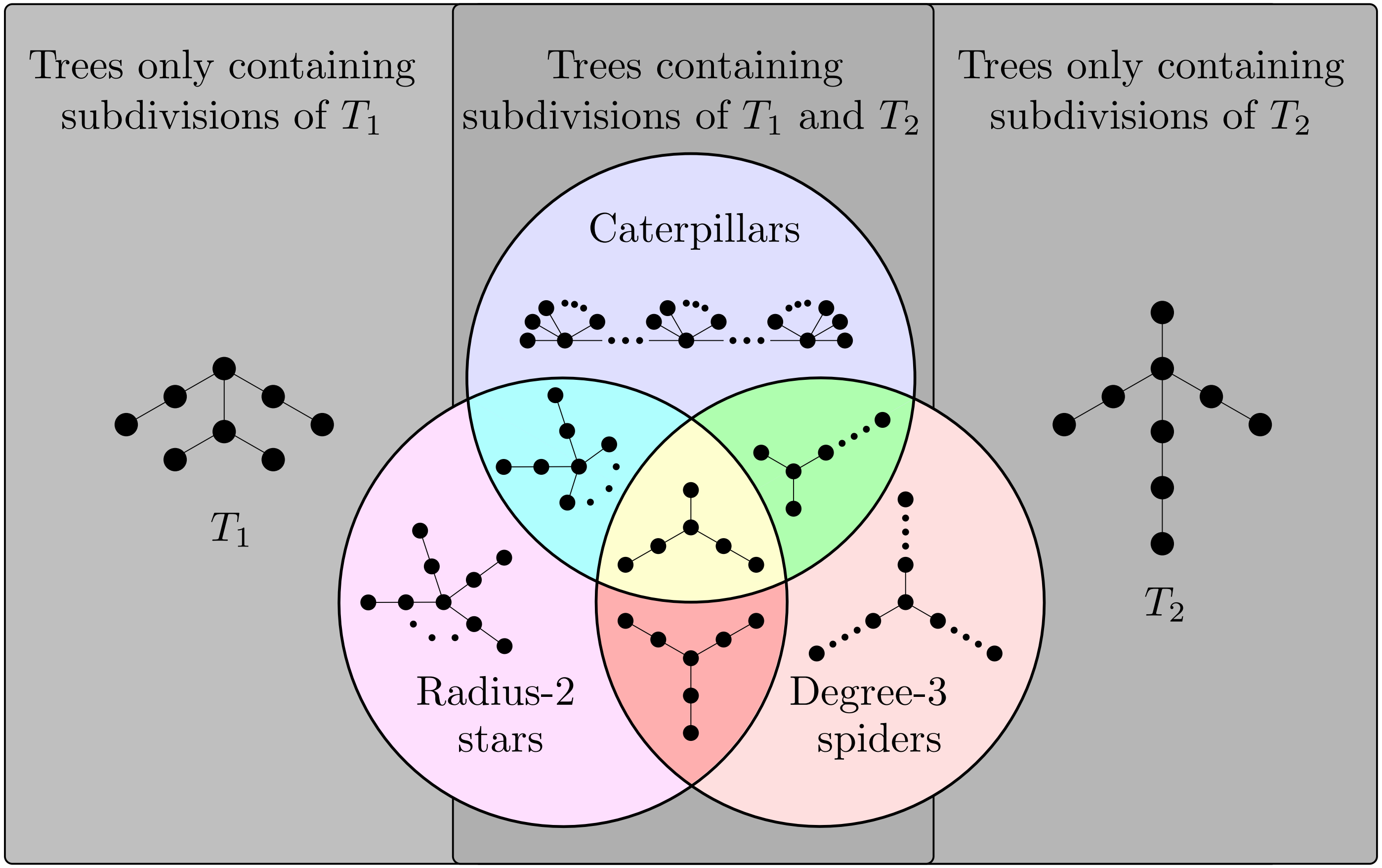
Unlabeled Level Planarity – Trees



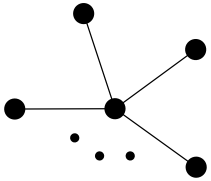
- All ULP trees fall into one of three categories:
  - **Caterpillars**



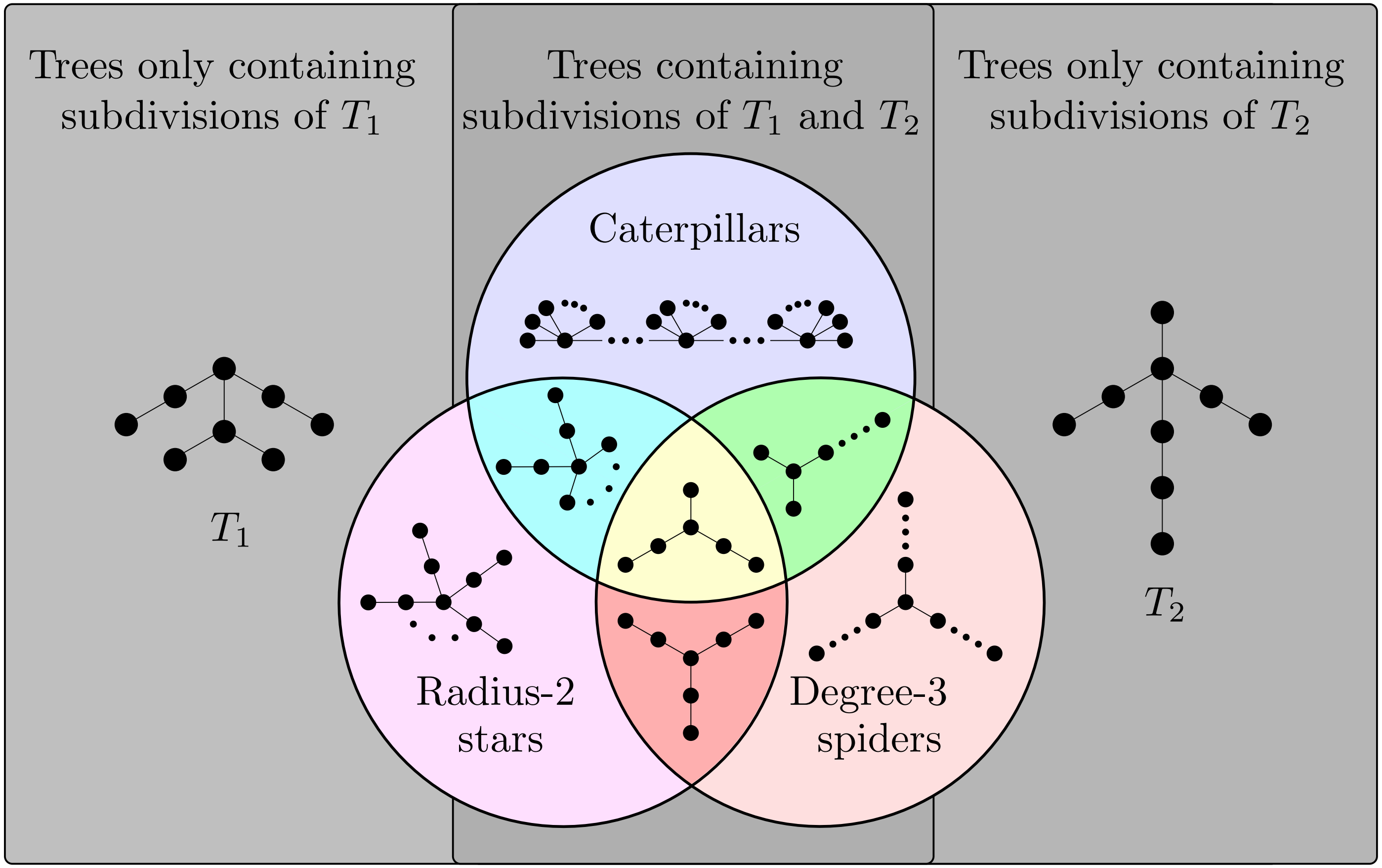
Unlabeled Level Planarity – Trees



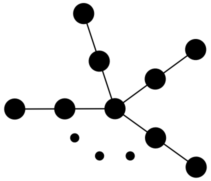
- All ULP trees fall into one of three categories:
  - Caterpillars
  - **Radius-2 stars**



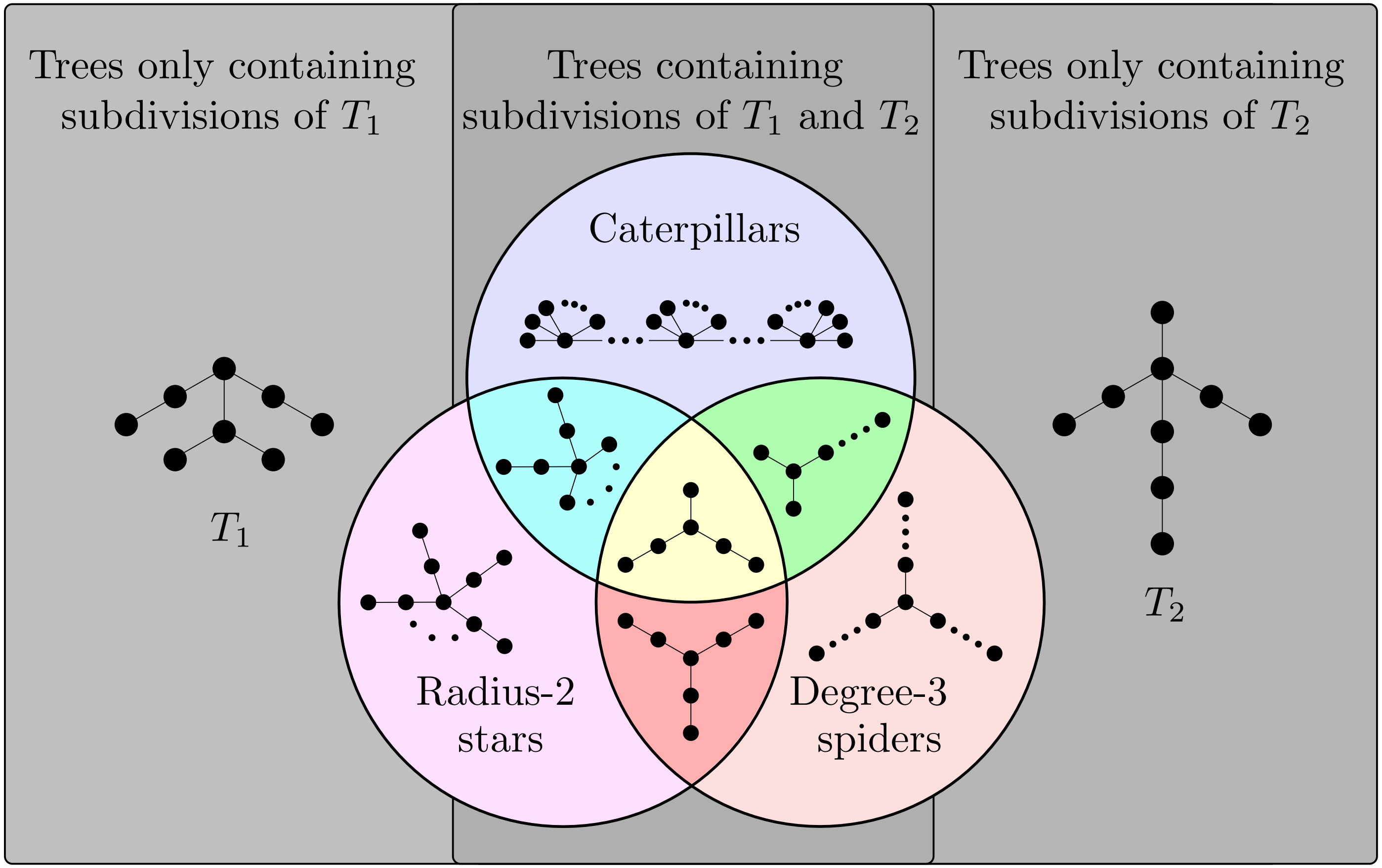
Unlabeled Level Planarity – Trees



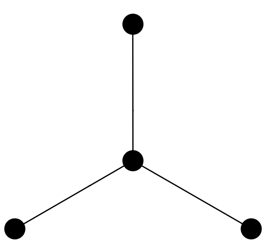
- All ULP trees fall into one of three categories:
  - Caterpillars
  - **Radius-2 stars**



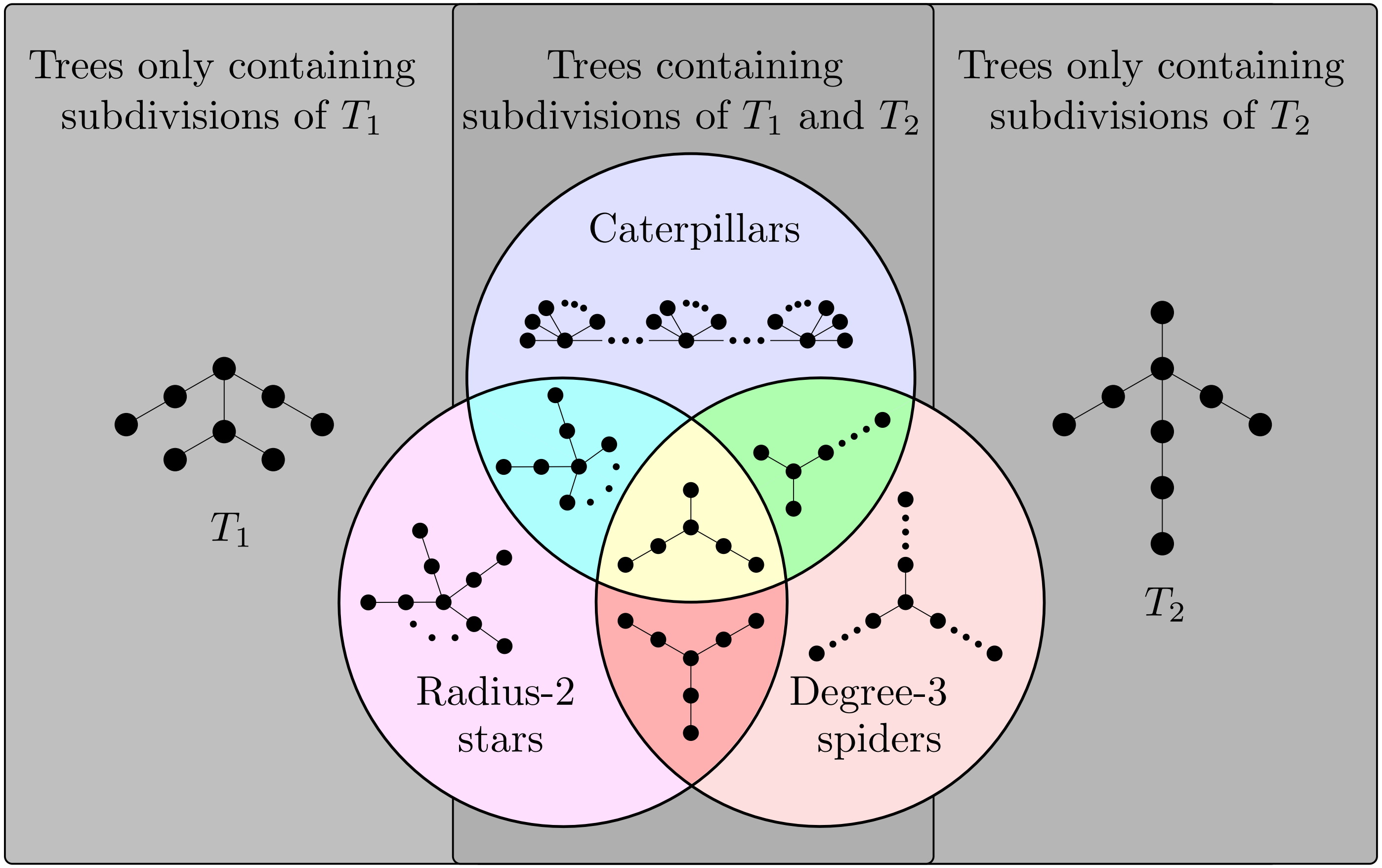
Unlabeled Level Planarity – Trees



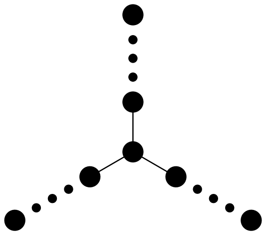
- All ULP trees fall into one of three categories:
  - Caterpillars
  - Radius-2 stars
  - **Degree-3 spiders**



Unlabeled Level Planarity – Trees



- All ULP trees fall into one of three categories:
  - Caterpillars
  - Radius-2 stars
  - **Degree-3 spiders**



Level Planarity vs. Standard Planarity

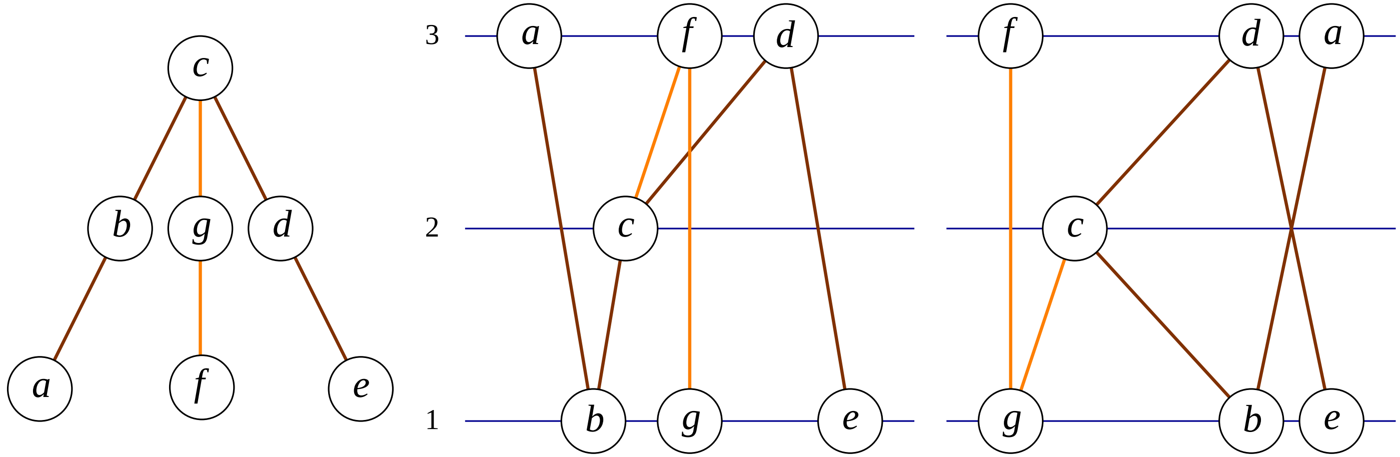
- More restrictive than standard planarity

Level Planarity vs. Standard Planarity

- More restrictive than standard planarity
  - All level planar graphs are planar

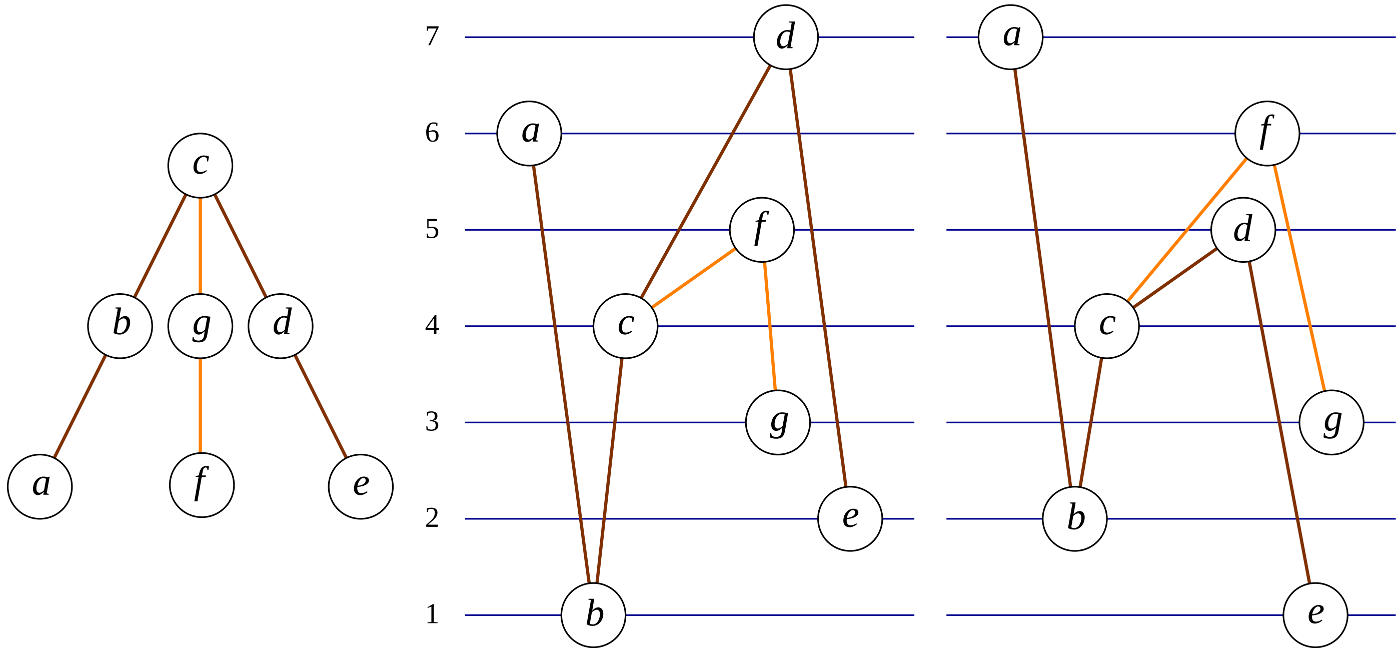
Level Planarity vs. Standard Planarity

- More restrictive than standard planarity
  - All level planar graphs are planar
  - But not all planar graphs are level planar for a given level assignment



Level Planarity vs. Standard Planarity

- More restrictive than standard planarity
  - All level planar graphs are planar
  - But not all planar graphs are level planar for a given level assignment
- A ULP graph can have a non-level planar assignment



Terminology

- For an n-level graph $G(V, E, \phi)$

Terminology

- For an n-level graph $G(V, E, \phi)$
  - A *chain* C is a path $v_1 - v_2 - \cdots - v_j$ in the underlying undirected graph

Terminology

- For an n-level graph $G(V, E, \phi)$
  - A *chain* C is a path $v_1 - v_2 - \cdots - v_j$ in the underlying undirected graph
  - $<_Y$ denotes the linear ordering of V induced by ϕ
    - $u <_Y v \iff \phi(u) < \phi(v) \iff u$ **lies below** v

Terminology

- For an n-level graph $G(V, E, \phi)$
  - A *chain* C is a path $v_1 - v_2 - \cdots - v_j$ in the underlying undirected graph
  - $<_Y$ denotes the linear ordering of V induced by ϕ
    - $u <_Y v \iff \phi(u) < \phi(v) \iff u$ **lies below** v
  - $<_X$ denotes the linear ordering of V induced by the x-coordinates of a level drawing of G
    - $u <_X v \iff$ u **lies to the right of** v

Terminology

- For an n-level graph $G(V, E, \phi)$
  - A *chain* C is a path $v_1 - v_2 - \cdots - v_j$ in the underlying undirected graph
  - $<_Y$ denotes the linear ordering of V induced by ϕ
    - $u <_Y v \iff \phi(u) < \phi(v) \iff u$ **lies below** v
  - $<_X$ denotes the linear ordering of V induced by the x-coordinates of a level drawing of G
    - $u <_X v \iff$ u **lies to the right of** v
  - Both $<_X$ and $<_Y$ can be extended to compare a vertex with a chain C

Terminology

- For an n-level graph $G(V, E, \phi)$
  - A *chain* C is a path $v_1 - v_2 - \cdots - v_j$ in the underlying undirected graph
  - $<_Y$ denotes the linear ordering of V induced by ϕ
    - $u <_Y v \iff \phi(u) < \phi(v) \iff u$ **lies below** v
  - $<_X$ denotes the linear ordering of V induced by the x-coordinates of a level drawing of G
    - $u <_X v \iff$ u **lies to the right of** v
  - Both $<_X$ and $<_Y$ can be extended to compare a vertex with a chain C
    - $u <_X v_1 - v_2 - \cdots - v_j$

      $\iff u$ **lies to the right of** every point at which C intersects level $\phi(u)$

Terminology

- For an n-level graph $G(V, E, \phi)$
  - A *chain* C is a path $v_1 - v_2 - \cdots - v_j$ in the underlying undirected graph
  - $<_Y$ denotes the linear ordering of V induced by ϕ
    - $u <_Y v \iff \phi(u) < \phi(v) \iff u$ **lies below** v
  - $<_X$ denotes the linear ordering of V induced by the x-coordinates of a level drawing of G
    - $u <_X v \iff$ u **lies to the right of** v
  - Both $<_X$ and $<_Y$ can be extended to compare a vertex with a chain C
    - $u <_X v_1 - v_2 - \cdots - v_j$

      $\iff u$ **lies to the right of** every point at which C intersects level $\phi(u)$
    - $u <_Y v_1 - v_2 - \cdots - v_j$

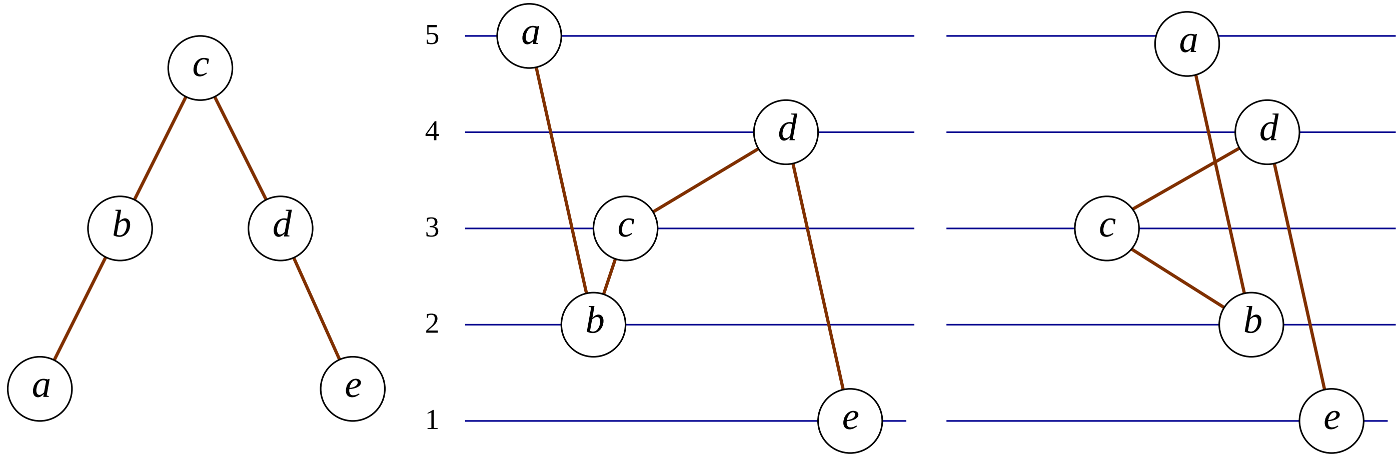
      $\iff u$ **lies below** every point that C shares the same x-coordinate as u

Key Observation

- Let C be some chain $a{-}b{-}c{-}d{-}e$

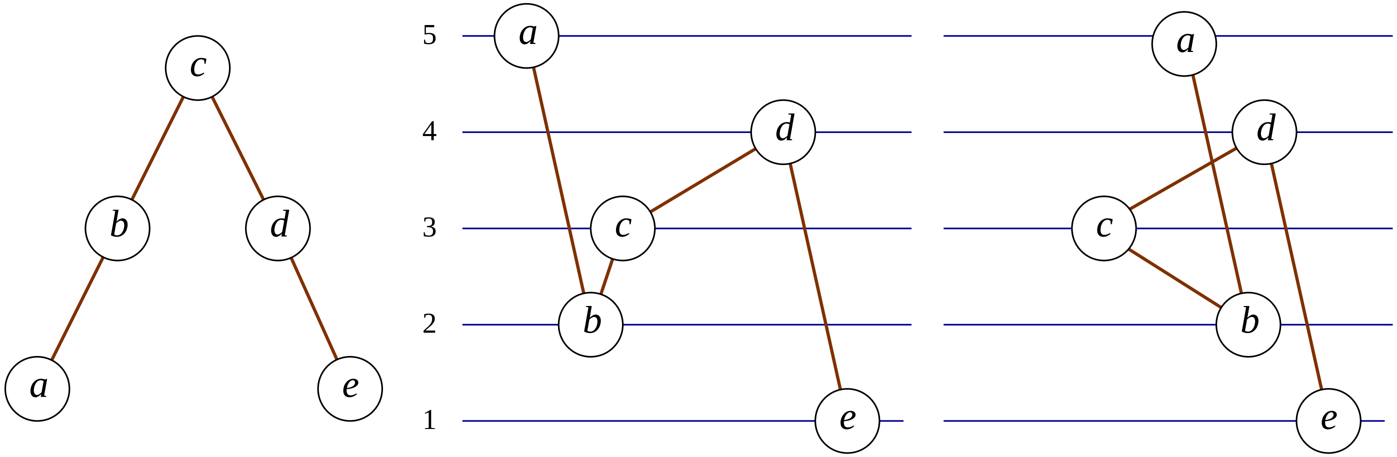
Key Observation

- Let C be some chain a—b—c—d—e
  - If $a <_Y d <_Y c <_Y b <_Y e$



Key Observation

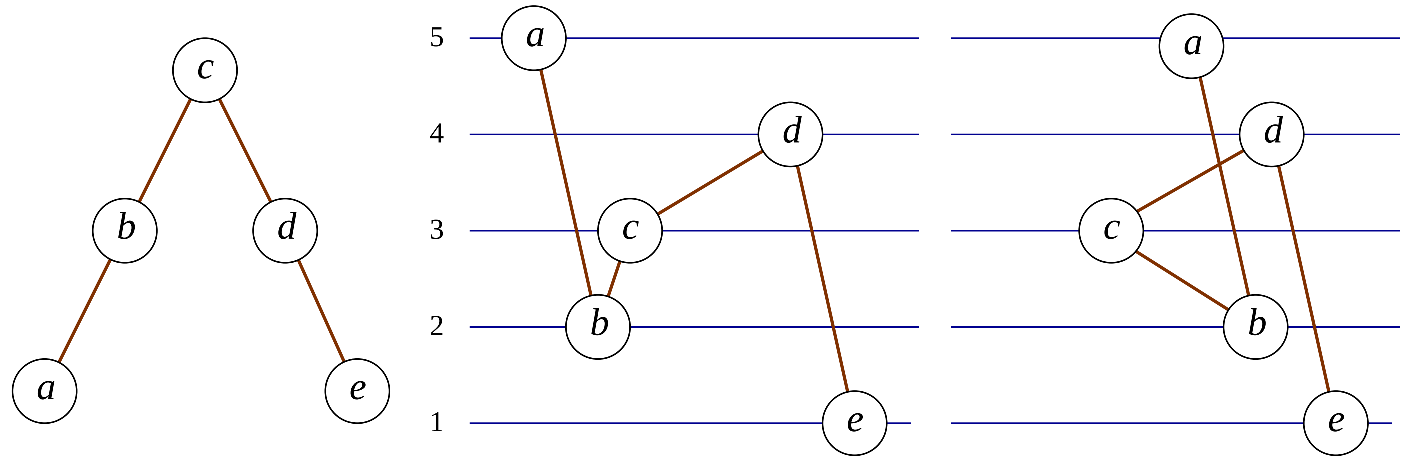
- Let C be some chain a—b—c—d—e
  - If $a <_Y d <_Y c <_Y b <_Y e$



  - Then either
    - a—$b <_X c <_X d$—e or
    - d—$e <_X c <_X a$—b, i.e, c is between a—b and d—e

Key Observation

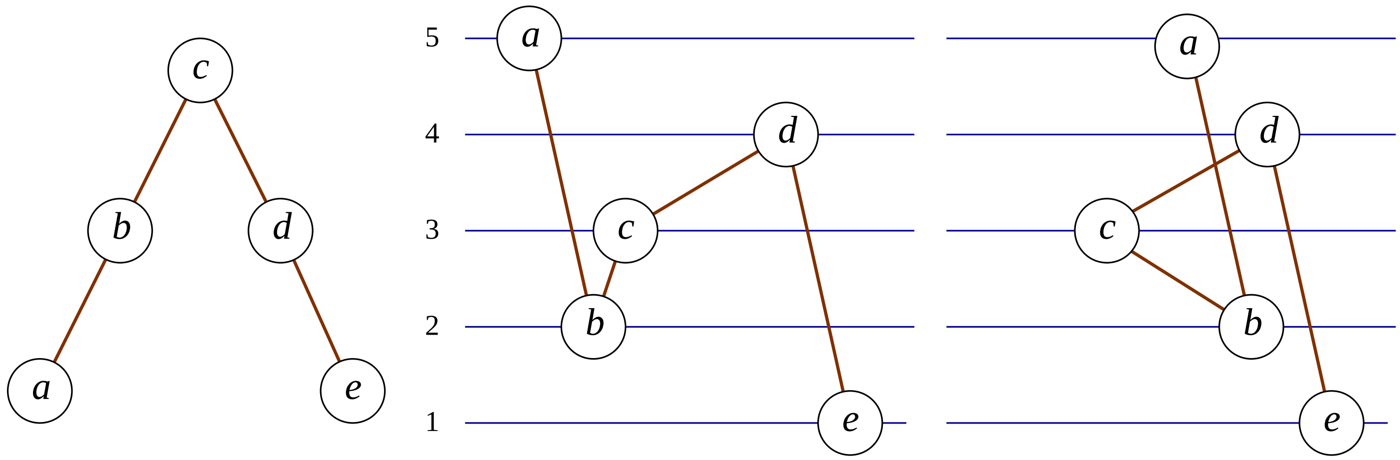
- Let C be some chain $a{-}b{-}c{-}d{-}e$
  - If $a <_Y d <_Y c <_Y b <_Y e$



  - Then either
    - $a{-}b <_X c <_X d{-}e$ or
    - $d{-}e <_X c <_X a{-}b$, i.e, c is between $a{-}b$ and $d{-}e$
  - Since otherwise $c{-}b{-}a$ will cross $c{-}d{-}e$

Key Observation

- Let C be some chain $a—b—c—d—e$
  - If $a <_Y d <_Y c <_Y b <_Y e$



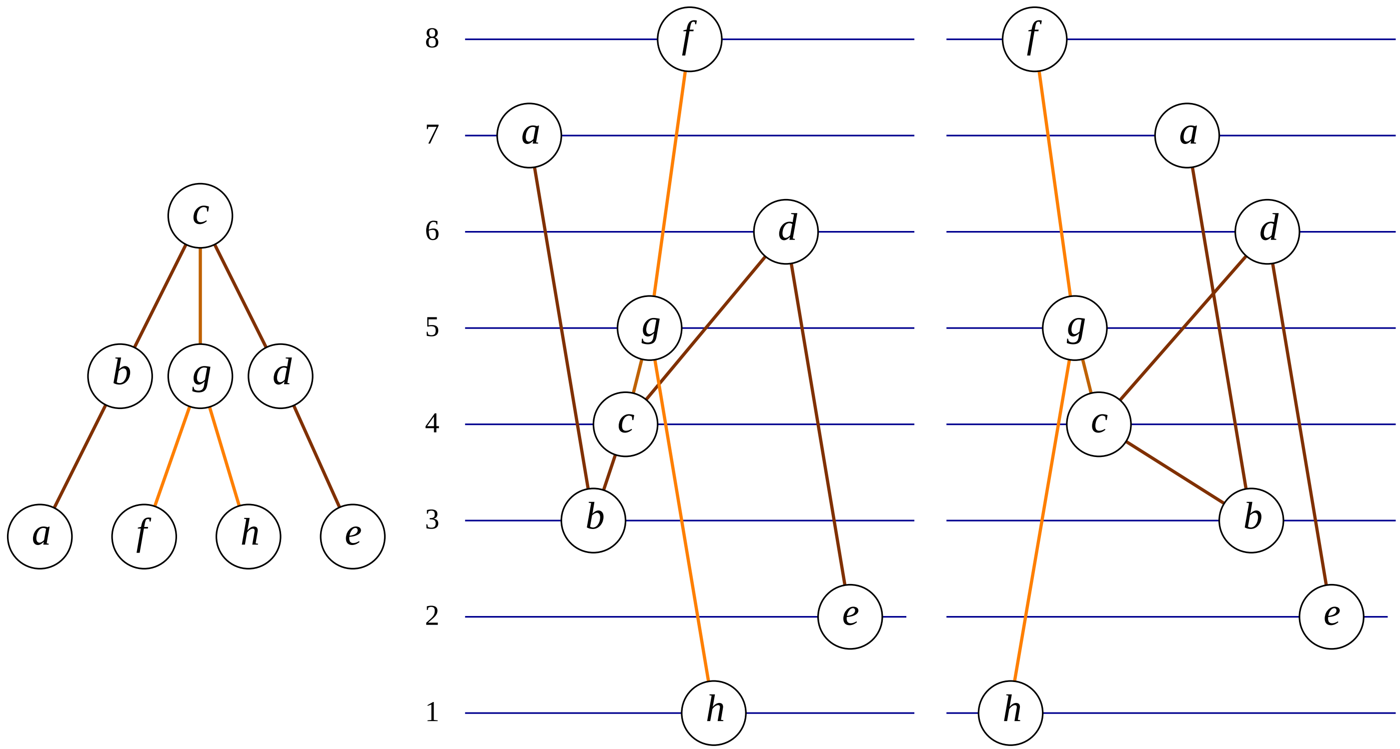
  - Then either
    - $a—b <_X c <_X d—e$ or
    - $d—e <_X c <_X a—b$, i.e, c is between $a—b$ and $d—e$
  - Since otherwise $c—b—a$ will cross $c—d—e$
- So c cannot be leftmost or rightmost without forcing a crossing
  - Can use this property to prove T_1 and T_2 are not ULP

Forbidden Trees – Proof T_1 is not ULP

- Let C be the chain a–b–c–d–e

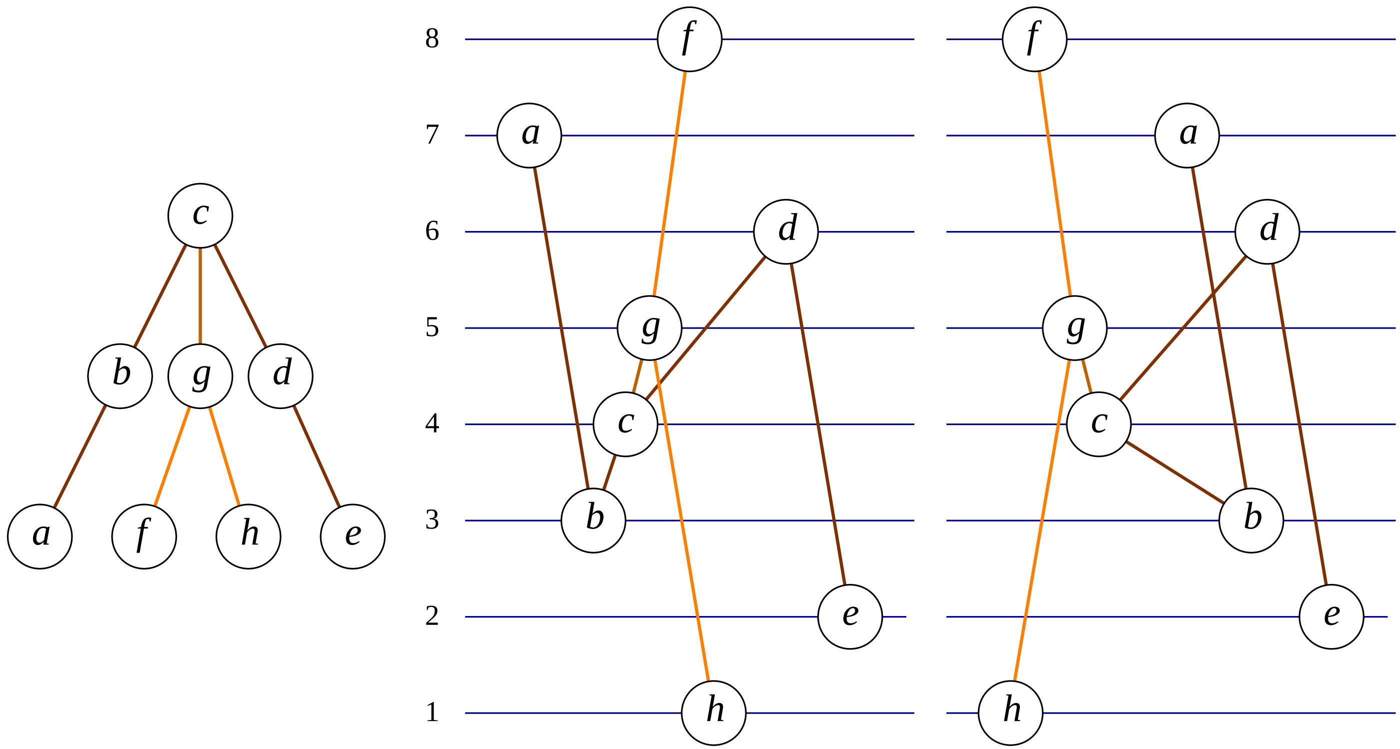
Forbidden Trees – Proof T_1 is not ULP

- Let C be the chain a—b—c—d—e
  - Where $\{a, f\} <_Y d <_Y \{c, g\} <_Y b <_Y \{e, h\}$



Forbidden Trees – Proof T_1 is not ULP

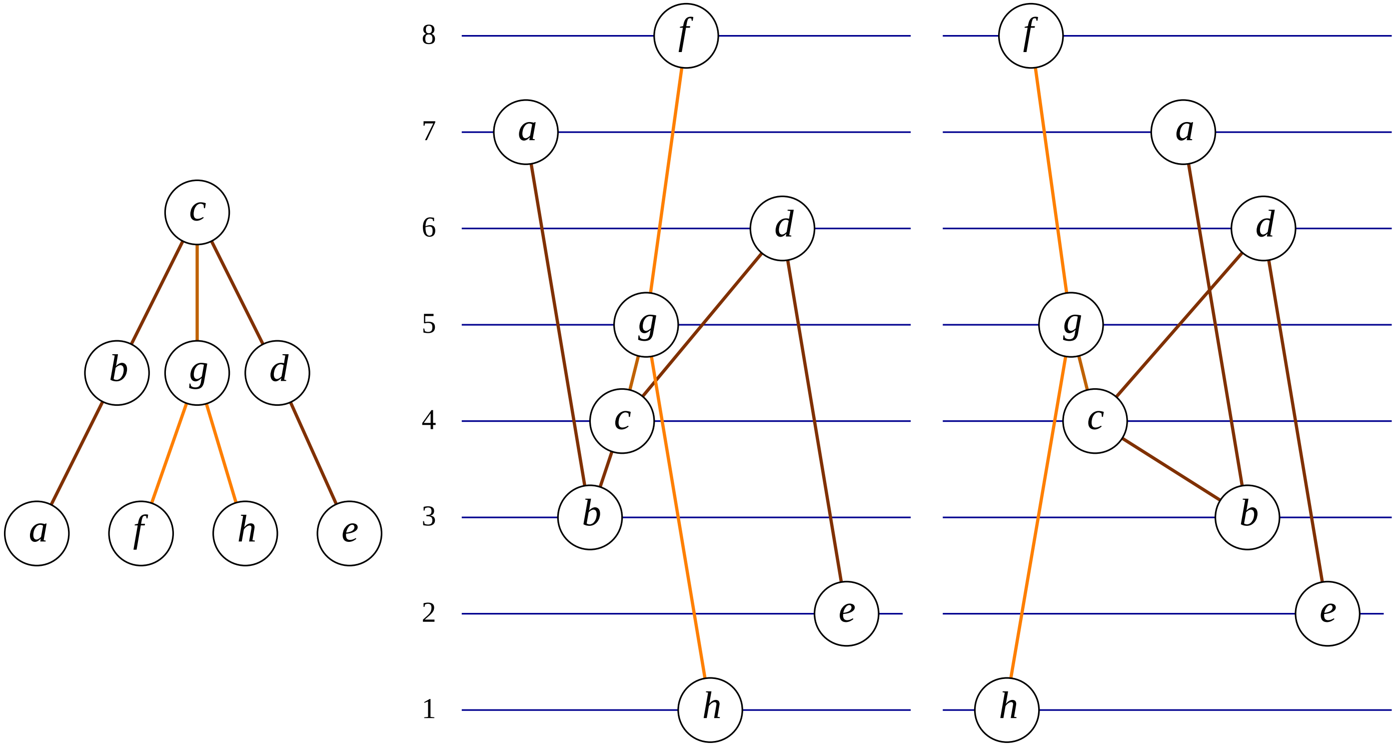
- Let C be the chain $a–b–c–d–e$
  - Where $\{a, f\} <_Y d <_Y \{c, g\} <_Y b <_Y \{e, h\}$



  - Can assume without loss of generality that
    - $a–b <_X c <_X d–e$
    - I.e., c lies between $a–b$ and $d–e$

Forbidden Trees – Proof T_1 is not ULP

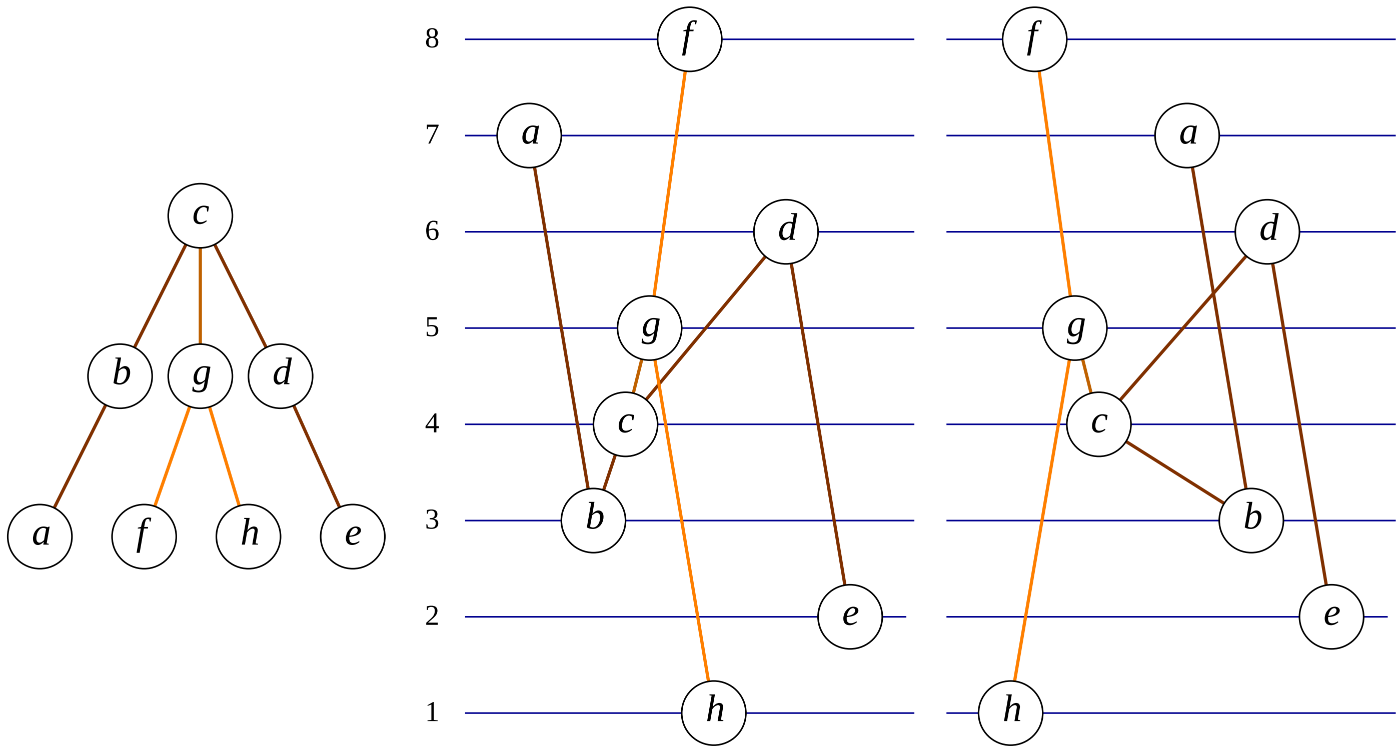
- Let C be the chain a–b–c–d–e
  - Where $\{a, f\} <_Y d <_Y \{c, g\} <_Y b <_Y \{e, h\}$



  - Implies that a–$b <_X g <_X d$–e
    - Otherwise, c–g will cross a–b or d–e

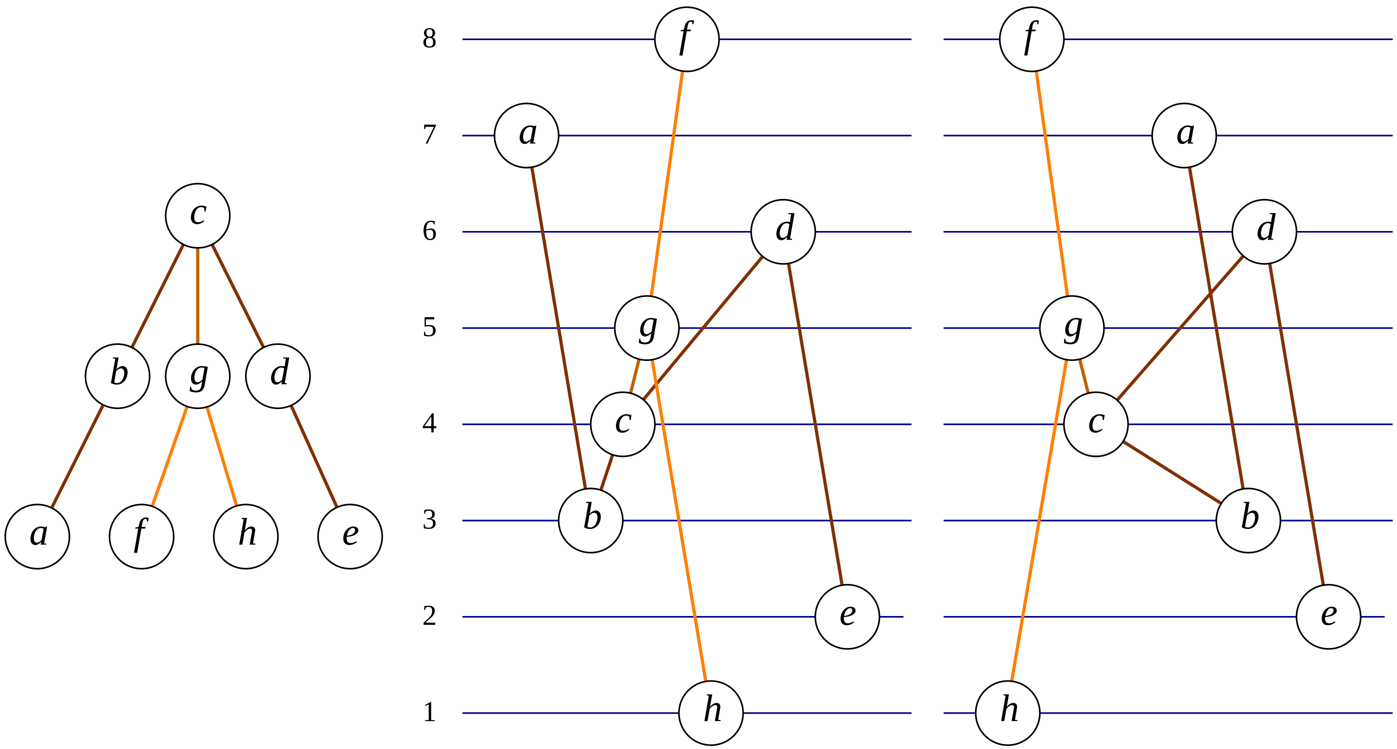
Forbidden Trees – Proof T_1 is not ULP

- Let C be the chain a—b—c—d—e
  - Where $\{a, f\} <_Y d <_Y \{c, g\} <_Y b <_Y \{e, h\}$



  - Then either $g >_Y a$—b—c—d
    - In which case g—h crosses a—b—c—d

- Let C be the chain a–b–c–d–e
  - Where $\{a, f\} <_Y d <_Y \{c, g\} <_Y b <_Y \{e, h\}$



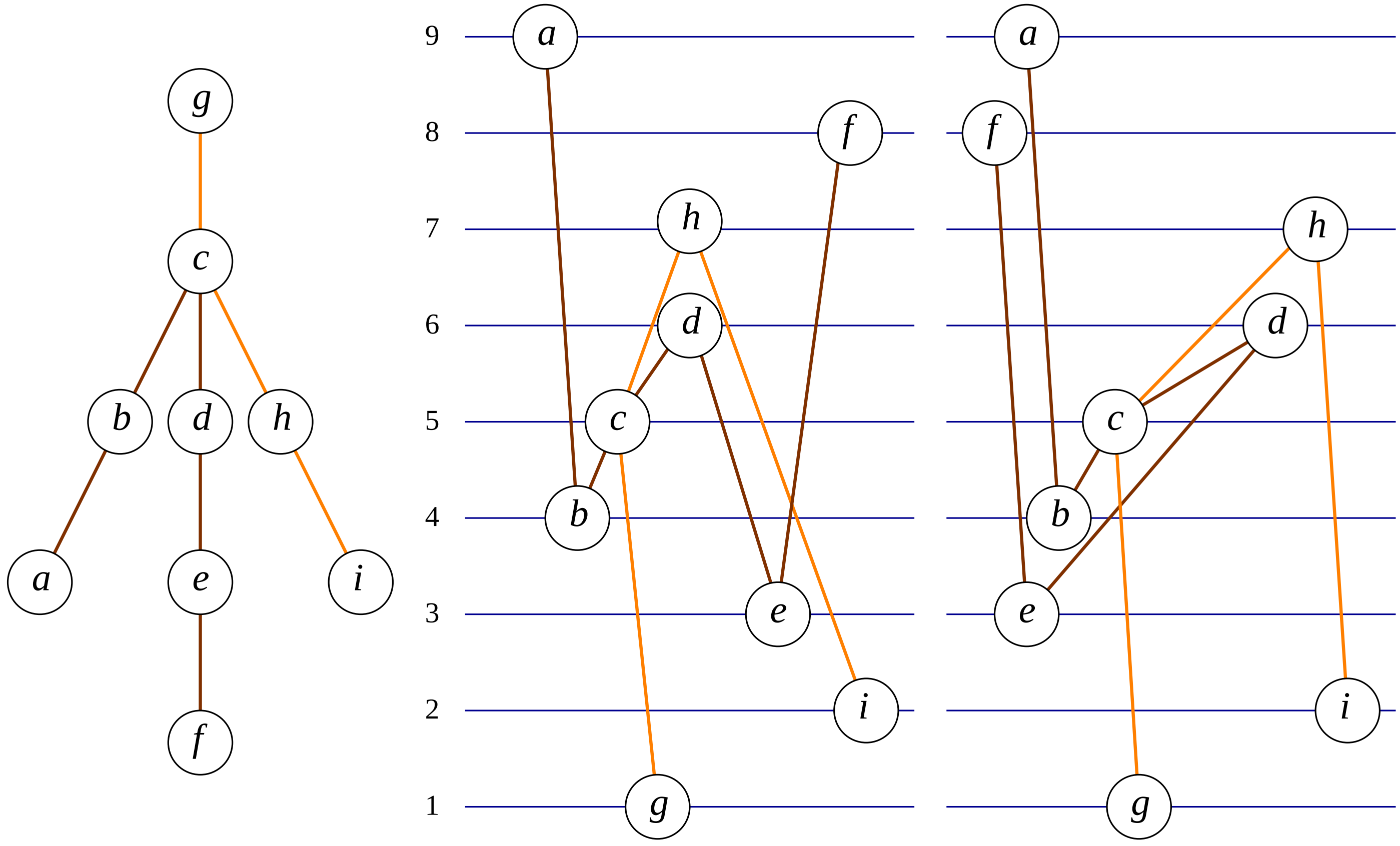
  - Or $g <_Y b$–c–d–e
    - In which case g–f crosses b–c–d–e

Forbidden Trees – Proof T_2 is not ULP

- Let C be the chain $a{-}b{-}c{-}d{-}e$

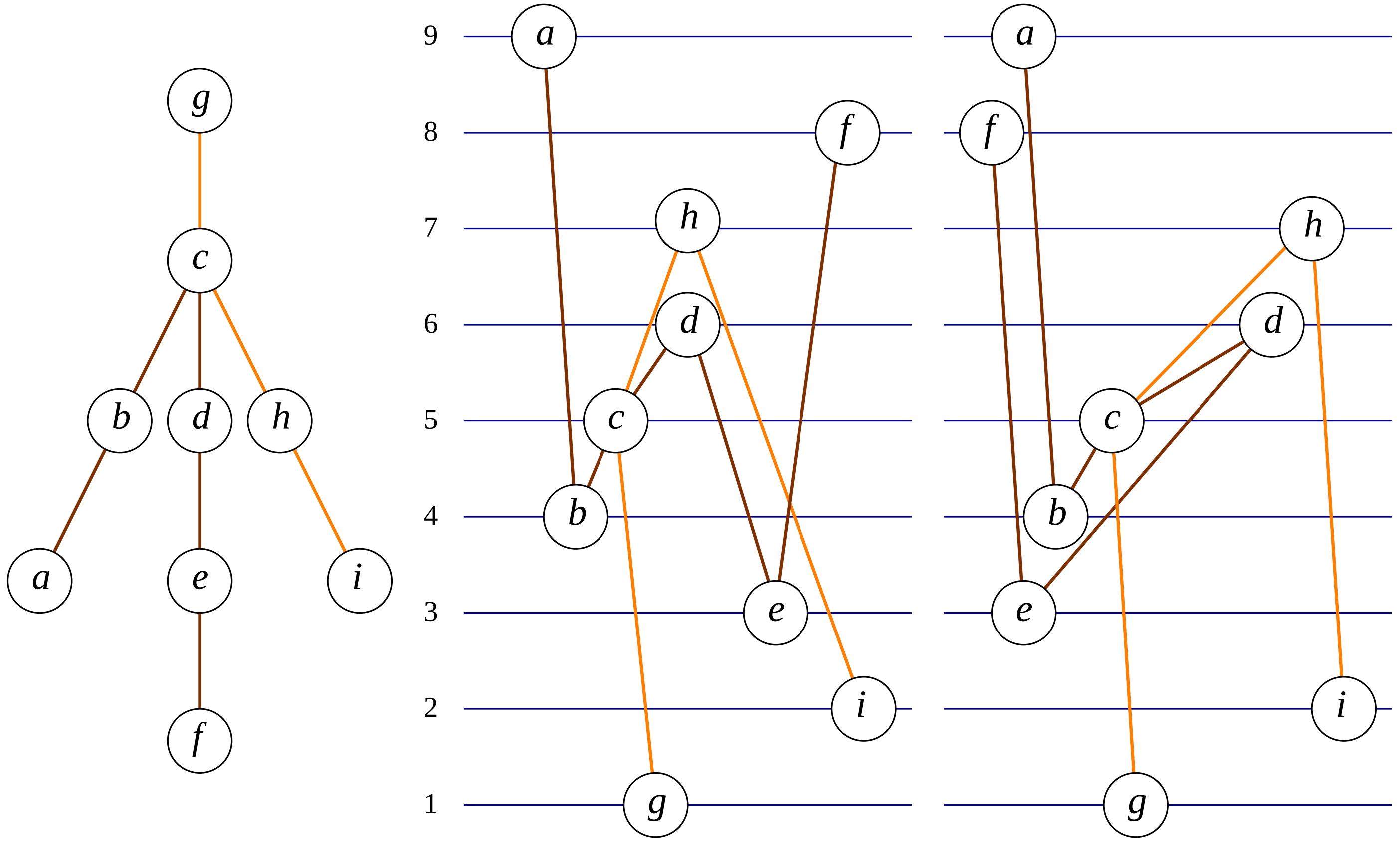
Forbidden Trees – Proof T_2 is not ULP

- Let C be the chain a–b–c–d–e
  - Where $\{a, f\} <_Y h <_Y d <_Y c <_Y b <_Y e <_Y \{g, i\}$



Forbidden Trees – Proof T_2 is not ULP

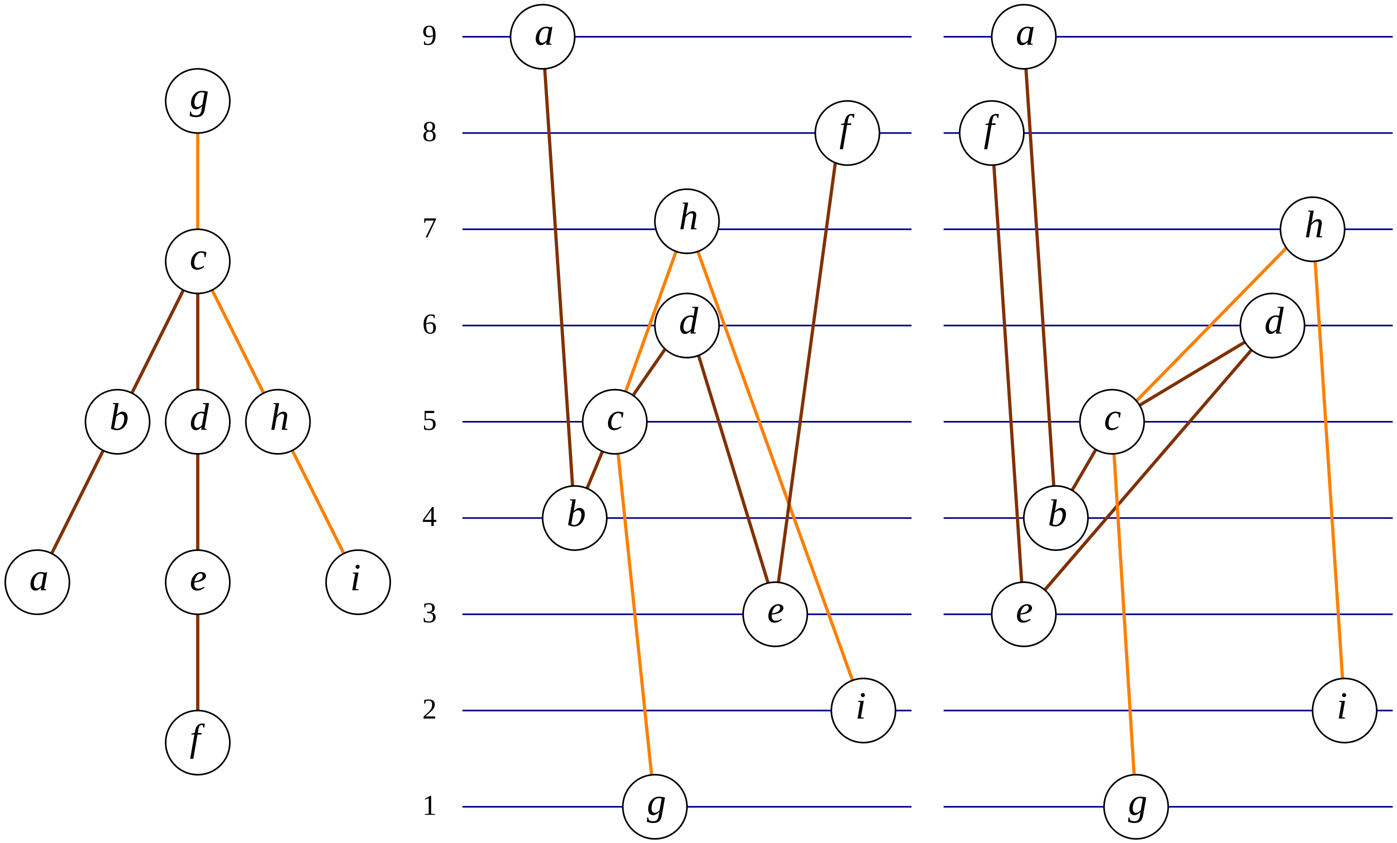
- Let C be the chain $a–b–c–d–e$
  - Where $\{a, f\} <_Y h <_Y d <_Y c <_Y b <_Y e <_Y \{g, i\}$



  - One can assume without loss of generality that
    - $a–b <_X c <_X d–e$ since $a <_Y d <_Y c <_Y b <_Y e$

Forbidden Trees – Proof T_2 is not ULP

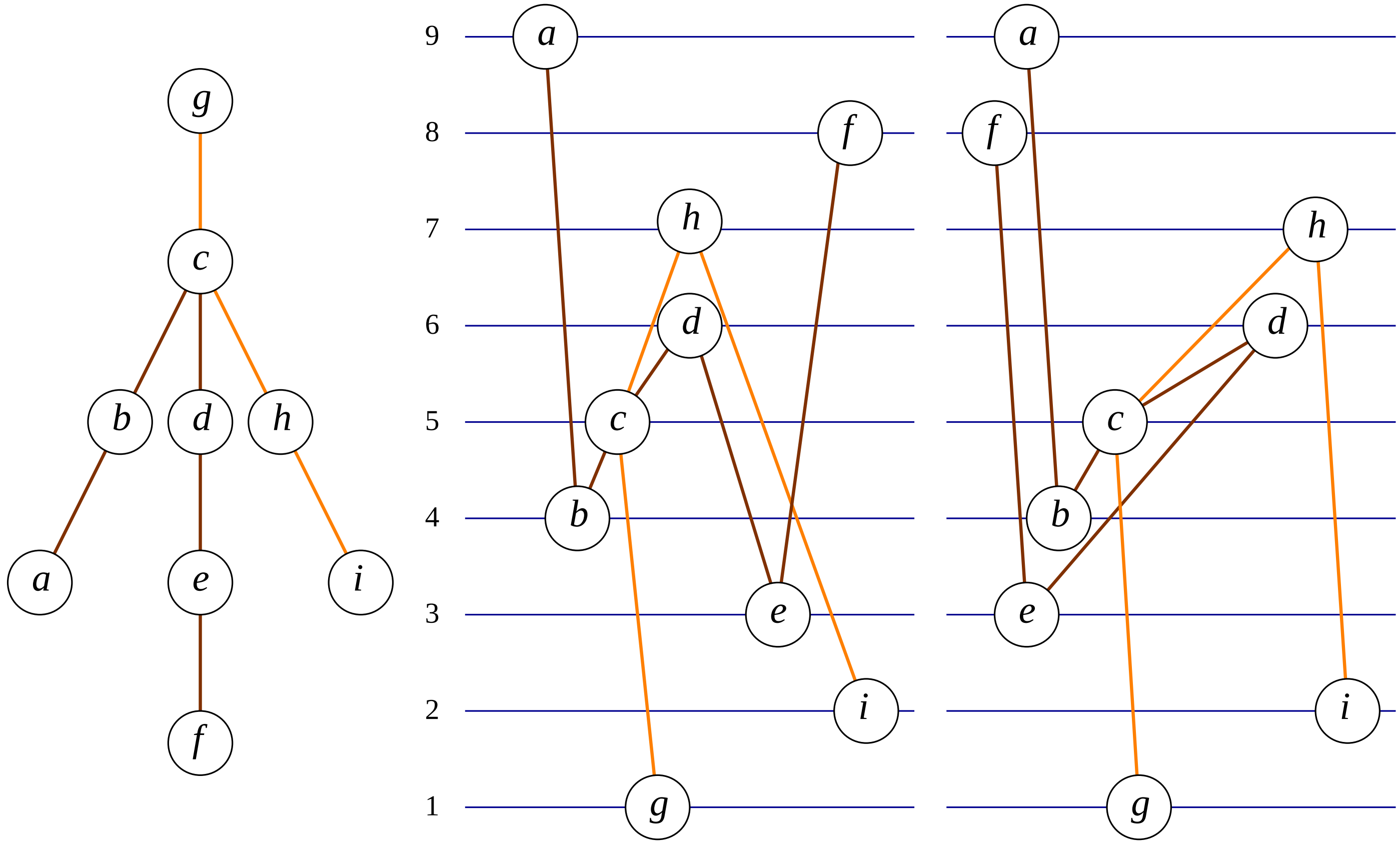
- Let C be the chain a–b–c–d–e
  - Where $\{a, f\} <_Y h <_Y d <_Y c <_Y b <_Y e <_Y \{g, i\}$



  - AND one can also assume that
    - a–$b <_X c <_X h$–i since $a <_Y h <_Y c <_Y b <_Y i$

Forbidden Trees – Proof T_2 is not ULP

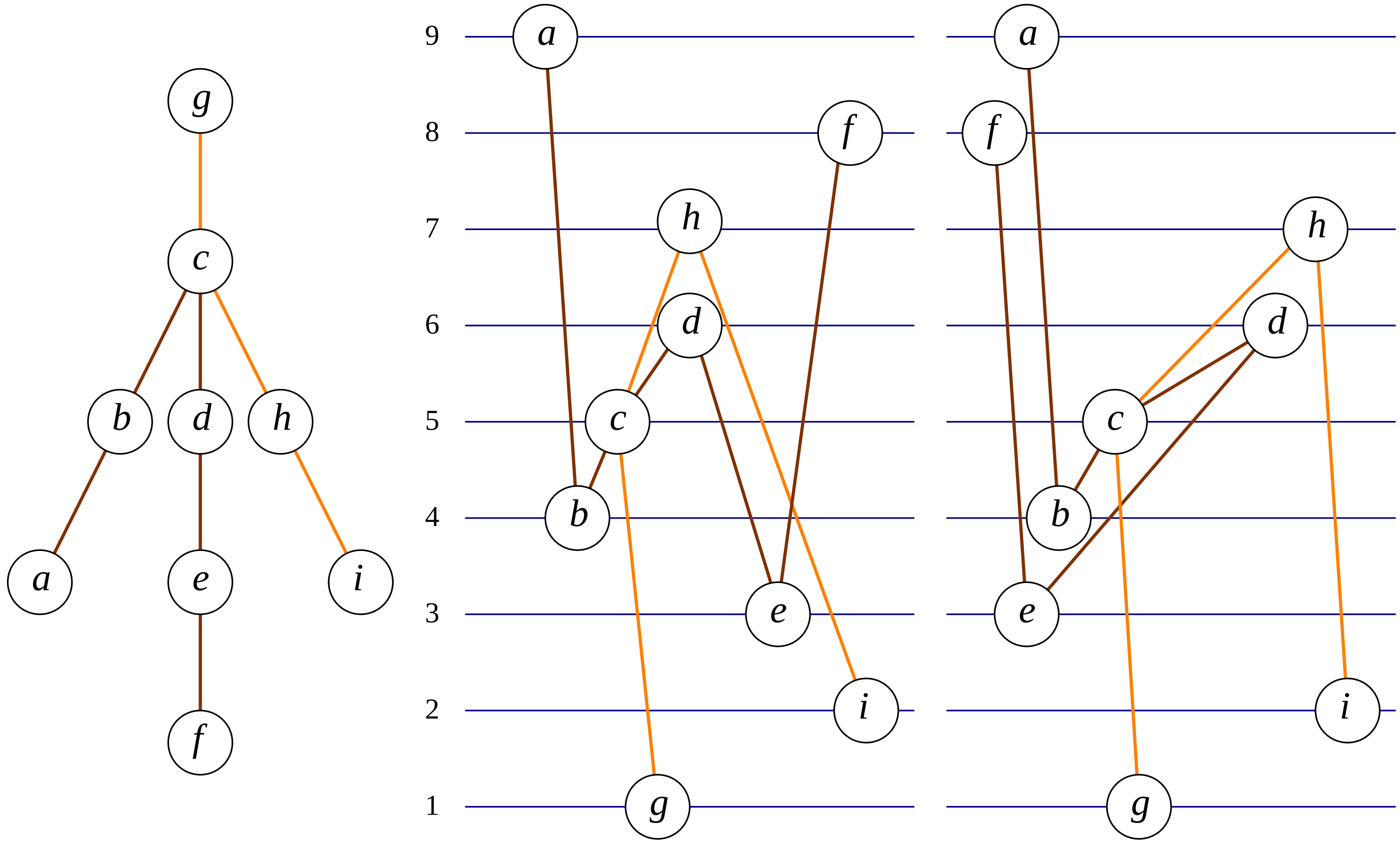
- Let C be the chain a–b–c–d–e
  - Where $\{a, f\} <_Y h <_Y d <_Y c <_Y b <_Y e <_Y \{g, i\}$



  - Implies that c–$g <_X e <_X h$–i
    - Since otherwise c–g will cross d–e OR d–e will cross h–i

Forbidden Trees – Proof T_2 is not ULP

- Let C be the chain $a{-}b{-}c{-}d{-}e$
  - Where $\{a, f\} <_Y h <_Y d <_Y c <_Y b <_Y e <_Y \{g, i\}$



  - However, this implies that $e <_Y g{-}c{-}h{-}i$
    - In which case $e{-}f$ crosses $g{-}c{-}h{-}i$

Forbidden Trees – Lemma and Corollary

- Existence of labelings in which T_1 and T_2 are not level planar gives the following lemma:

  Lemma 1 *There exist labelings that prevent T_1 and T_2 from being level planar.*

Forbidden Trees – Lemma and Corollary

- Existence of labelings in which T_1 and T_2 are not level planar gives the following lemma:

  Lemma 1 *There exist labelings that prevent T_1 and T_2 from being level planar.*

- Considering subdivisions gives the next corollary:

  Corollary 2 *If a tree $T(V, E)$ contains a subdivision of T_1 or T_2, then it cannot be unlabeled level planar.*

Forbidden Trees – Lemma and Corollary

- Existence of labelings in which T_1 and T_2 are not level planar gives the following lemma:

  Lemma 1 *There exist labelings that prevent T_1 and T_2 from being level planar.*

- Considering subdivisions gives the next corollary:

  Corollary 2 *If a tree $T(V, E)$ contains a subdivision of T_1 or T_2, then it cannot be unlabeled level planar.*

- Proof idea:
  - Assign intermediate levels to vertices of subdivided edges

Caterpillars – Linear Time Realization

- An n-level realization of a caterpillar in linear time gives the next lemma:

Caterpillars – Linear Time Realization

- An n-level realization of a caterpillar in linear time gives the next lemma:

  Lemma 3 *(Brass* et al.*, 2003) An n-vertex caterpillar $T(V, E)$ with an m-vertex spine can be n-level realized in $O(n)$ time on a $2m \times n$ grid for any vertex labeling $\phi : V \xrightarrow[\text{onto}]{1:1} \{1, 2, \ldots, n\}$.*

Caterpillars – Linear Time Realization

- An n-level realization of a caterpillar in linear time gives the next lemma:

  Lemma 3 *(Brass* et al.*, 2003) An n-vertex caterpillar $T(V, E)$ with an m-vertex spine can be n-level realized in $O(n)$ time on a $2m \times n$ grid for any vertex labeling $\phi : V \xrightarrow[\textit{onto}]{1:1} \{1, 2, \ldots, n\}$.*

- Proof idea:
  - Embed spine vertices left to right on even x-coordinates

Caterpillars – Linear Time Realization

- An n-level realization of a caterpillar in linear time gives the next lemma:

  Lemma 3 *(Brass* et al.*, 2003) An n-vertex caterpillar $T(V, E)$ with an m-vertex spine can be n-level realized in $O(n)$ time on a $2m \times n$ grid for any vertex labeling $\phi : V \xrightarrow[\text{onto}]{1:1} \{1, 2, \ldots, n\}$.*

- Proof idea:
  - Embed spine vertices left to right on even x-coordinates
  - Then embed adjacent leaf vertices directly to the right one unit on odd x-coordinates

Caterpillars – Linear Time Realization

- An n-level realization of a caterpillar in linear time gives the next lemma:

  Lemma 3 *(Brass* et al.*, 2003) An n-vertex caterpillar $T(V, E)$ with an m-vertex spine can be n-level realized in $O(n)$ time on a $2m \times n$ grid for any vertex labeling $\phi : V \xrightarrow[\text{onto}]{1:1} \{1, 2, \ldots, n\}$.*

- Proof idea:
  - Embed spine vertices left to right on even x-coordinates
  - Then embed adjacent leaf vertices directly to the right one unit on odd x-coordinates
    - If a leaf vertex would lie on an edge, embed it directly below its spine vertex

Caterpillars – Linear Time Realization

- An n-level realization of a caterpillar in linear time gives the next lemma:

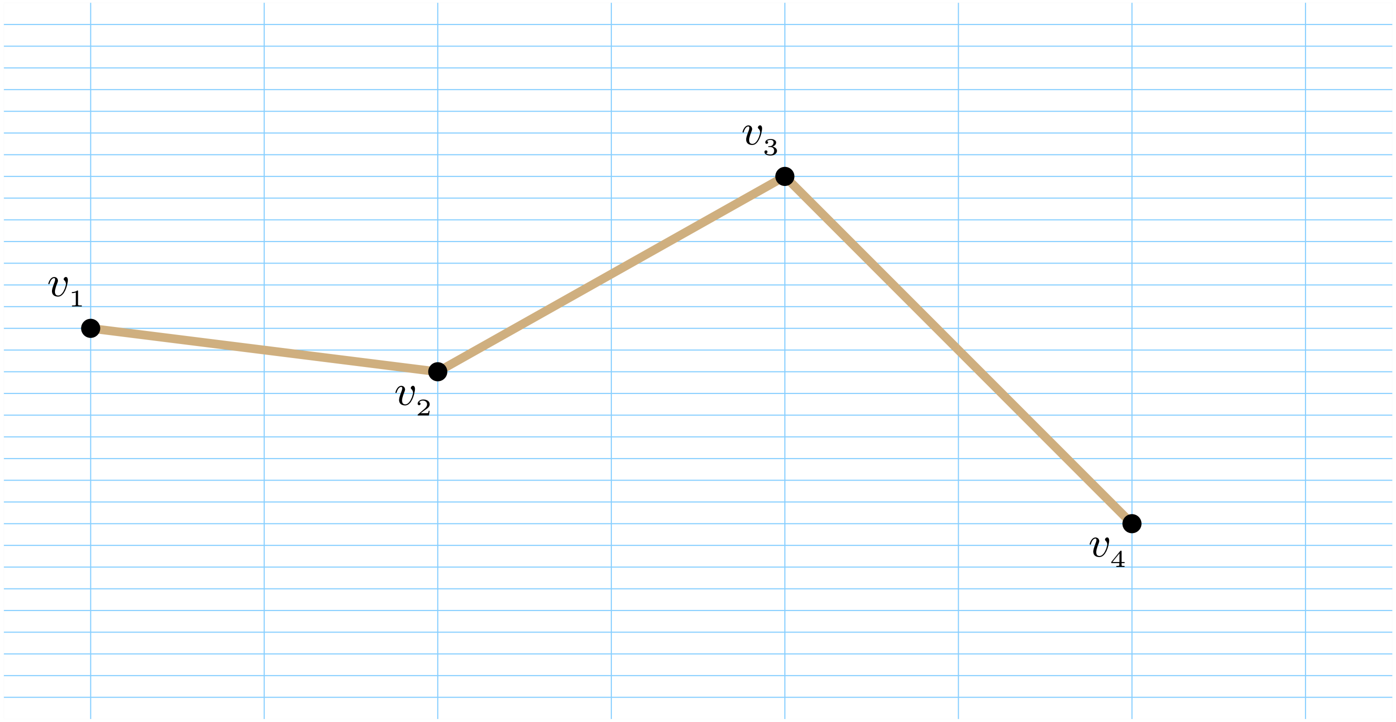
  Lemma 3 *(Brass* et al.*, 2003) An n-vertex caterpillar $T(V, E)$ with an m-vertex spine can be n-level realized in $O(n)$ time on a $2m \times n$ grid for any vertex labeling $\phi : V \xrightarrow[\text{onto}]{1:1} \{1, 2, \ldots, n\}$.*

- Proof idea:
  - Embed spine vertices left to right on even x-coordinates
  - Then embed adjacent leaf vertices directly to the right one unit on odd x-coordinates
    - If a leaf vertex would lie on an edge, embed it directly below its spine vertex
    - Can only happen for at most one leaf vertex per spine edge

Caterpillars – Linear Time Realization

- An n-level realization of a caterpillar in linear time gives the next lemma:

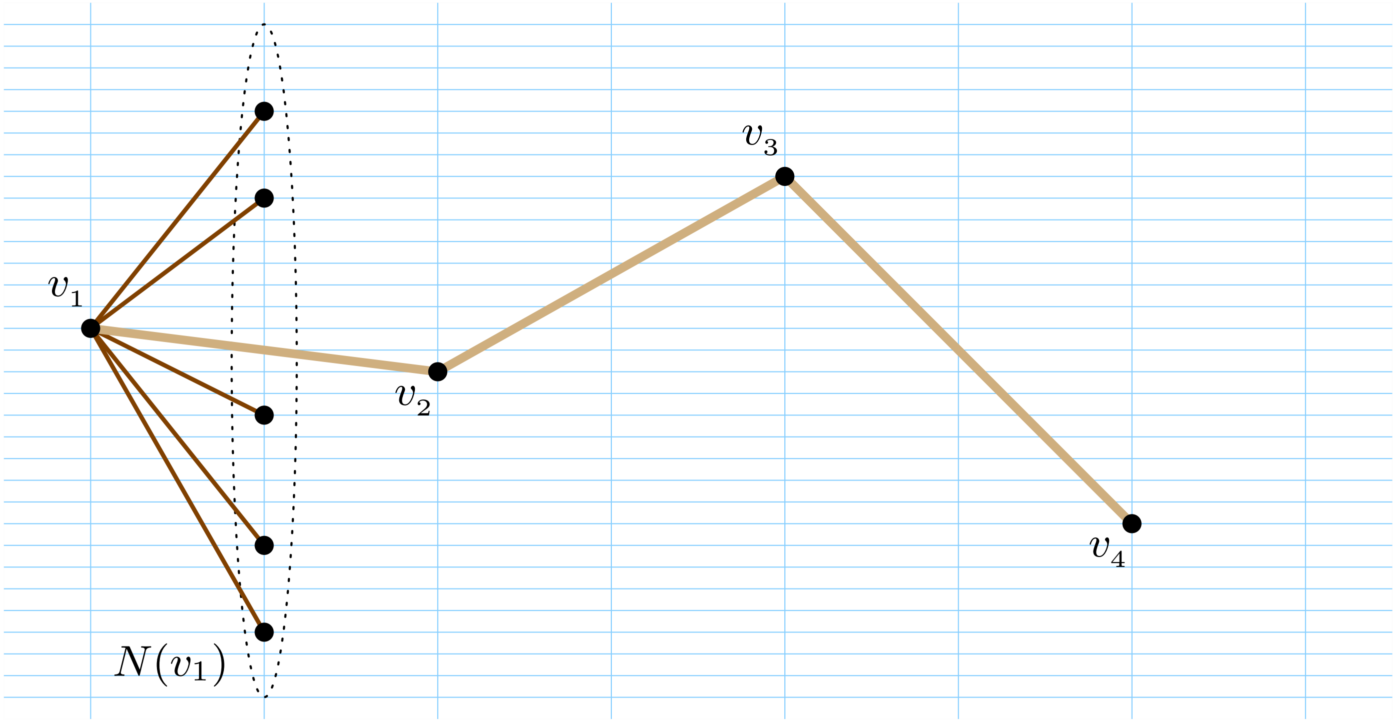
Lemma 3 *(Brass et al., 2003) An n-vertex caterpillar $T(V, E)$ with an m-vertex spine can be n-level realized in $O(n)$ time on a $2m \times n$ grid for any vertex labeling $\phi : V \xrightarrow[\text{onto}]{1:1} \{1, 2, \ldots, n\}$.*



Caterpillars – Linear Time Realization

- An n-level realization of a caterpillar in linear time gives the next lemma:

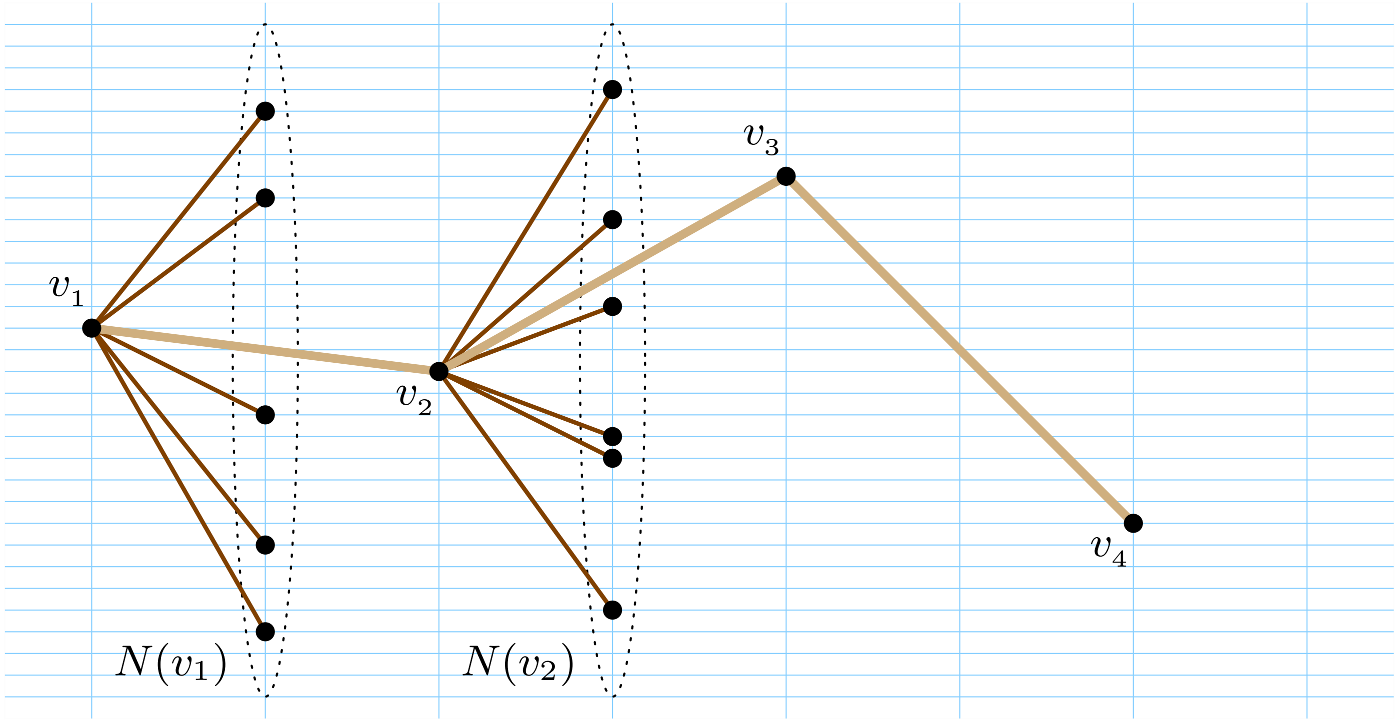
Lemma 3 *(Brass et al., 2003) An n-vertex caterpillar $T(V, E)$ with an m-vertex spine can be n-level realized in $O(n)$ time on a $2m \times n$ grid for any vertex labeling $\phi : V \xrightarrow[\text{onto}]{1:1} \{1, 2, \ldots, n\}$.*



Caterpillars – Linear Time Realization

- An n-level realization of a caterpillar in linear time gives the next lemma:

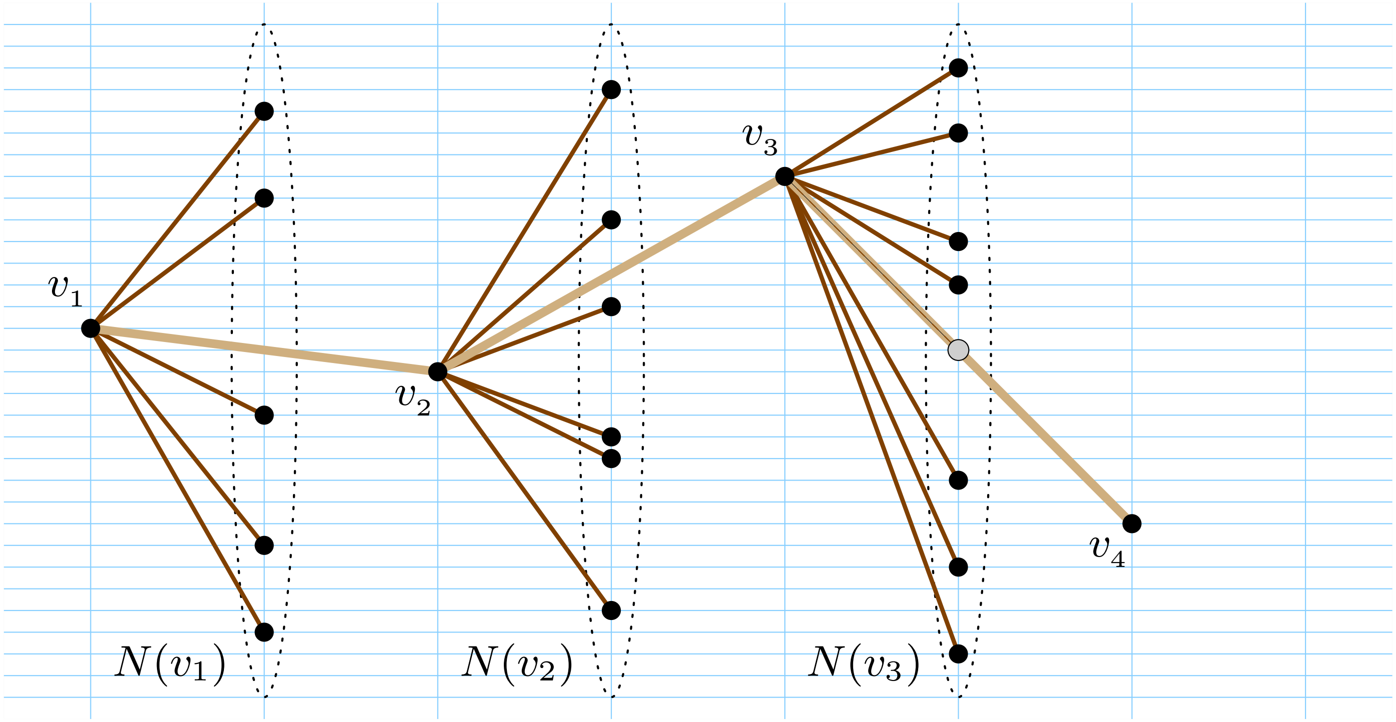
Lemma 3 *(Brass* et al.*, 2003) An n-vertex caterpillar $T(V, E)$ with an m-vertex spine can be n-level realized in $O(n)$ time on a $2m \times n$ grid for any vertex labeling $\phi : V \xrightarrow[\text{onto}]{1:1} \{1, 2, \ldots, n\}$.*



Caterpillars – Linear Time Realization

- An n-level realization of a caterpillar in linear time gives the next lemma:

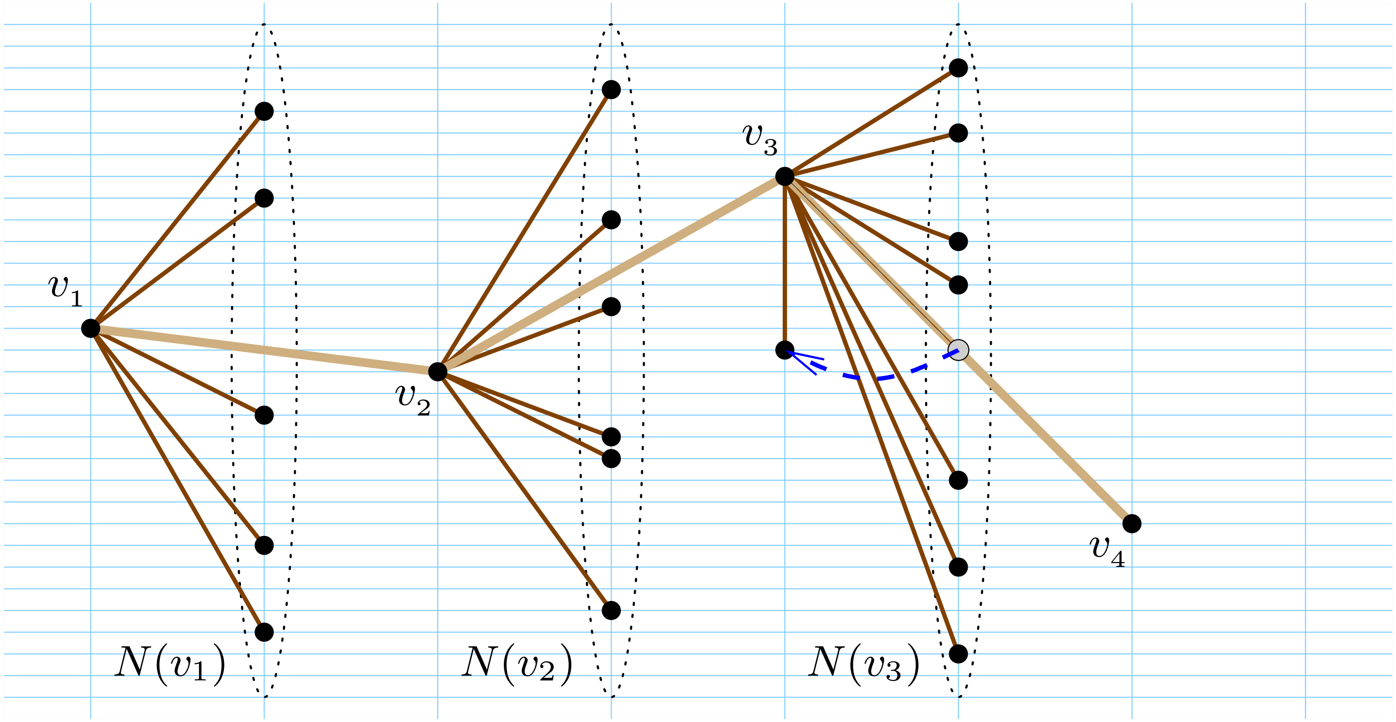
Lemma 3 *(Brass et al., 2003) An n-vertex caterpillar $T(V, E)$ with an m-vertex spine can be n-level realized in $O(n)$ time on a $2m \times n$ grid for any vertex labeling $\phi : V \xrightarrow[\text{onto}]{1:1} \{1, 2, \ldots, n\}$.*



Caterpillars – Linear Time Realization

- An n-level realization of a caterpillar in linear time gives the next lemma:

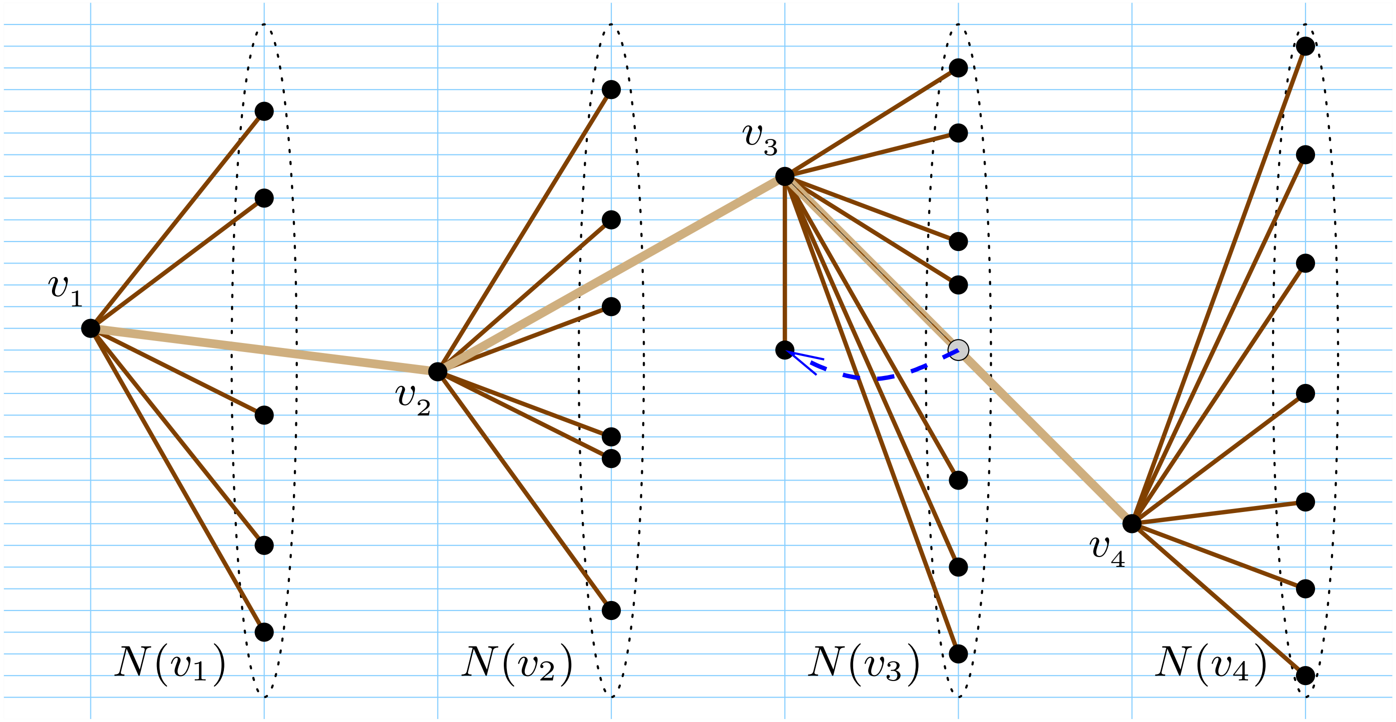
Lemma 3 *(Brass et al., 2003) An n-vertex caterpillar $T(V, E)$ with an m-vertex spine can be n-level realized in $O(n)$ time on a $2m \times n$ grid for any vertex labeling $\phi : V \xrightarrow[\text{onto}]{1:1} \{1, 2, \ldots, n\}$.*



Caterpillars – Linear Time Realization

- An n-level realization of a caterpillar in linear time gives the next lemma:

Lemma 3 *(Brass et al., 2003) An n-vertex caterpillar $T(V, E)$ with an m-vertex spine can be n-level realized in $O(n)$ time on a $2m \times n$ grid for any vertex labeling $\phi : V \xrightarrow[\text{onto}]{1:1} \{1, 2, \ldots, n\}$.*



Radius-2 Stars – Linear Time Realization

- An n-level realization of a radius-2 star in linear time yields the next lemma:

Radius-2 Stars – Linear Time Realization

- An n-level realization of a radius-2 star in linear time yields the next lemma:

Lemma 4 *An n-vertex radius-2 star $T(V, E)$ can be n-level realized in $O(n)$ time on a $(2n+3) \times n$ grid for any vertex labeling $\phi : V \xrightarrow[\text{onto}]{1:1} \{1, 2, \ldots, n\}$.*

Radius-2 Stars – Linear Time Realization

- An n-level realization of a radius-2 star in linear time yields the next lemma:

  Lemma 4 *An n-vertex radius-2 star $T(V, E)$ can be n-level realized in $O(n)$ time on a $(2n+3) \times n$ grid for any vertex labeling $\phi : V \xrightarrow[\text{onto}]{1:1} \{1,\, 2,\, \ldots,\, n\}$.*

- Proof idea:
  - Embed root vertex in middle of the x-coordinates

Radius-2 Stars – Linear Time Realization

- An n-level realization of a radius-2 star in linear time yields the next lemma:

  Lemma 4 *An n-vertex radius-2 star $T(V, E)$ can be n-level realized in $O(n)$ time on a $(2n+3) \times n$ grid for any vertex labeling $\phi : V \xrightarrow[\text{onto}]{1:1} \{1,\, 2,\, \ldots,\, n\}$.*

- Proof idea:
  - Embed root vertex in middle of the x-coordinates
  - Then embed adjacent vertices that have a leaf vertex below to the left, otherwise to the right

Radius-2 Stars – Linear Time Realization

- An n-level realization of a radius-2 star in linear time yields the next lemma:

  Lemma 4 *An n-vertex radius-2 star $T(V, E)$ can be n-level realized in $O(n)$ time on a $(2n + 3) \times n$ grid for any vertex labeling $\phi : V \xrightarrow[\text{onto}]{1:1} \{1, 2, \ldots, n\}$.*

- Proof idea:
  - Embed root vertex in middle of the x-coordinates
  - Then embed adjacent vertices that have a leaf vertex below to the left, otherwise to the right
  - Embed leaf vertices so that their incident edge segment has a slope of 1

Radius-2 Stars – Linear Time Realization

- An n-level realization of a radius-2 star in linear time yields the next lemma:

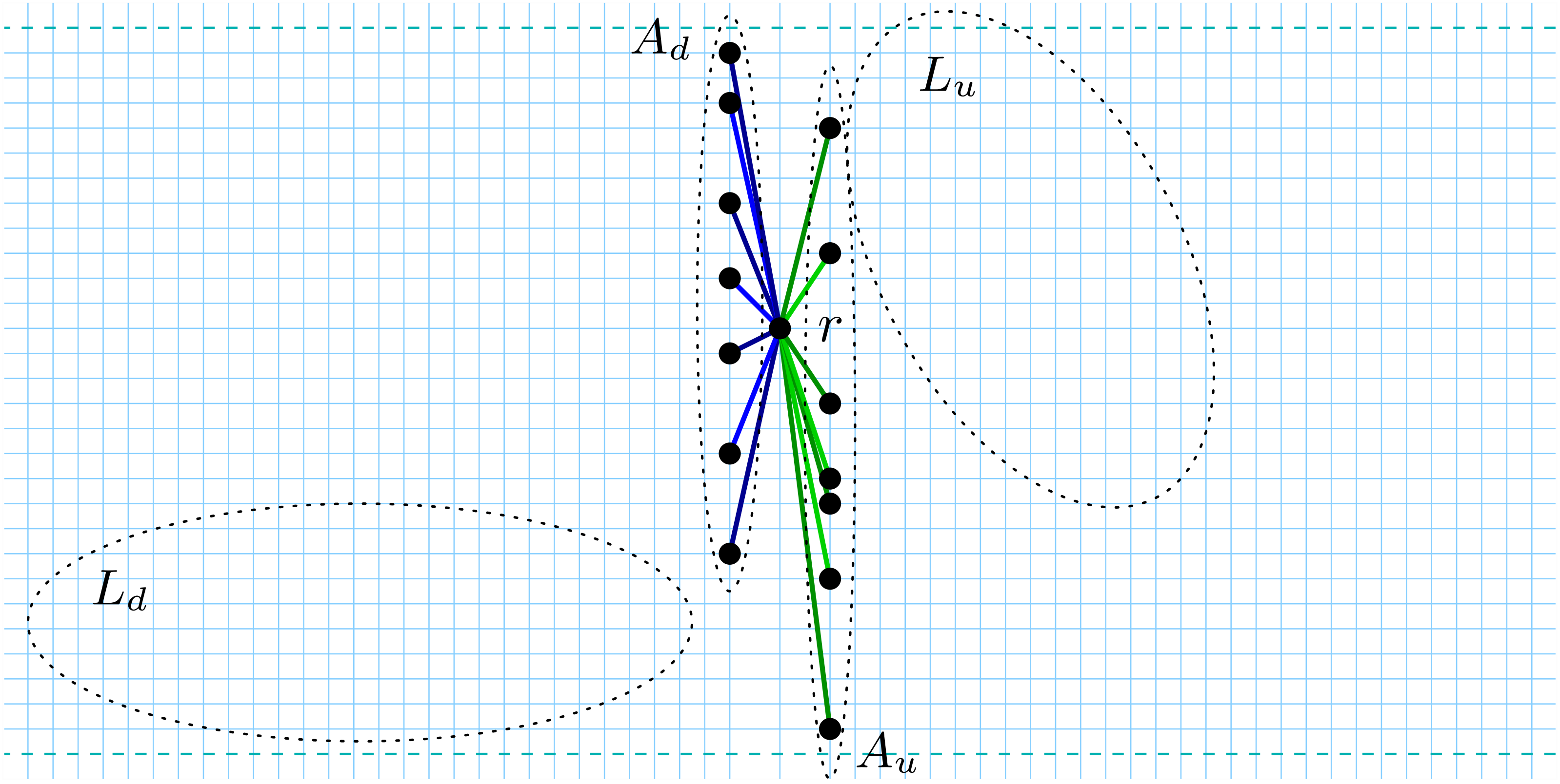
  Lemma 4 *An n-vertex radius-2 star $T(V, E)$ can be n-level realized in $O(n)$ time on a $(2n+3) \times n$ grid for any vertex labeling $\phi : V \xrightarrow[\text{onto}]{1:1} \{1, 2, \ldots, n\}$.*

- Proof idea:
  - Embed root vertex in middle of the x-coordinates
  - Then embed adjacent vertices that have a leaf vertex below to the left, otherwise to the right
  - Embed leaf vertices so that their incident edge segment has a slope of 1
    - Use imaginary levels above and below to determine x-coordinate

Radius-2 Stars – Linear Time Realization

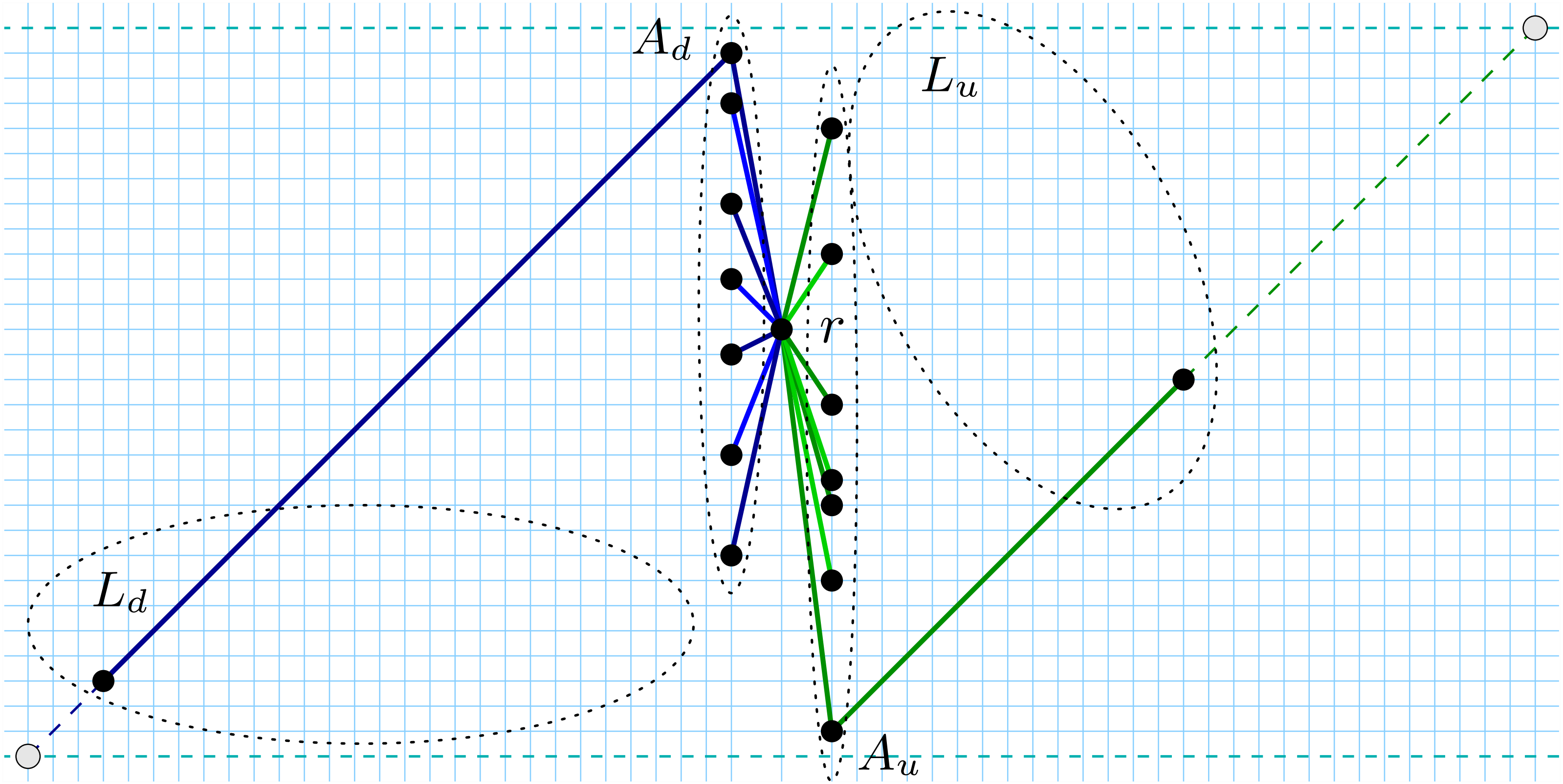
- An n-level realization of a radius-2 star in linear time yields the next lemma:

Lemma 4 *An n-vertex radius-2 star $T(V, E)$ can be n-level realized in $O(n)$ time on a $(2n+3) \times n$ grid for any vertex labeling $\phi : V \xrightarrow[\text{onto}]{1:1} \{1, 2, \ldots, n\}$.*



- An n-level realization of a radius-2 star in linear time yields the next lemma:

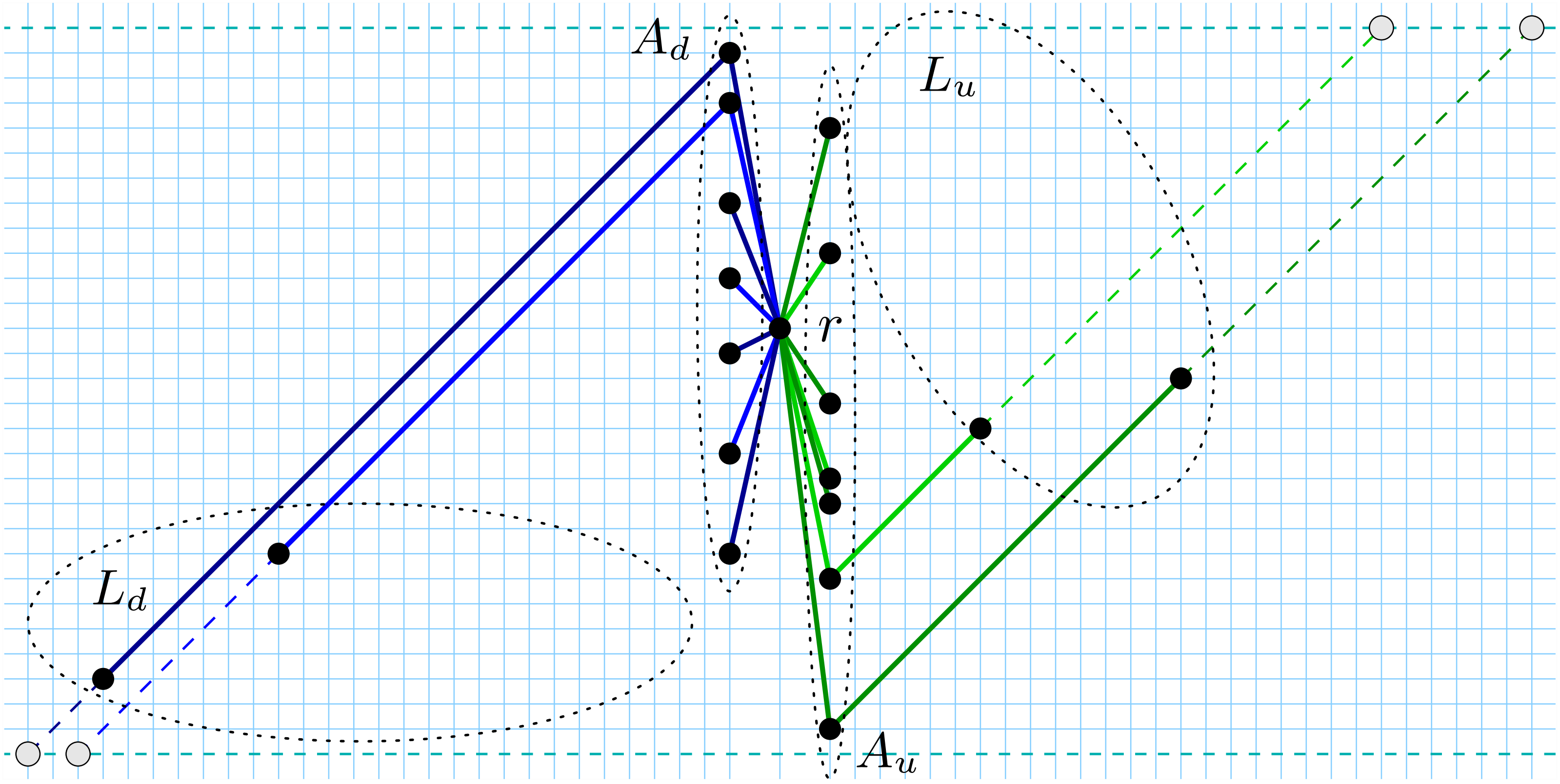
Lemma 4 *An n-vertex radius-2 star $T(V, E)$ can be n-level realized in $O(n)$ time on a $(2n + 3) \times n$ grid for any vertex labeling $\phi : V \xrightarrow[\text{onto}]{1:1} \{1, 2, \ldots, n\}$.*



Radius-2 Stars – Linear Time Realization

- An n-level realization of a radius-2 star in linear time yields the next lemma:

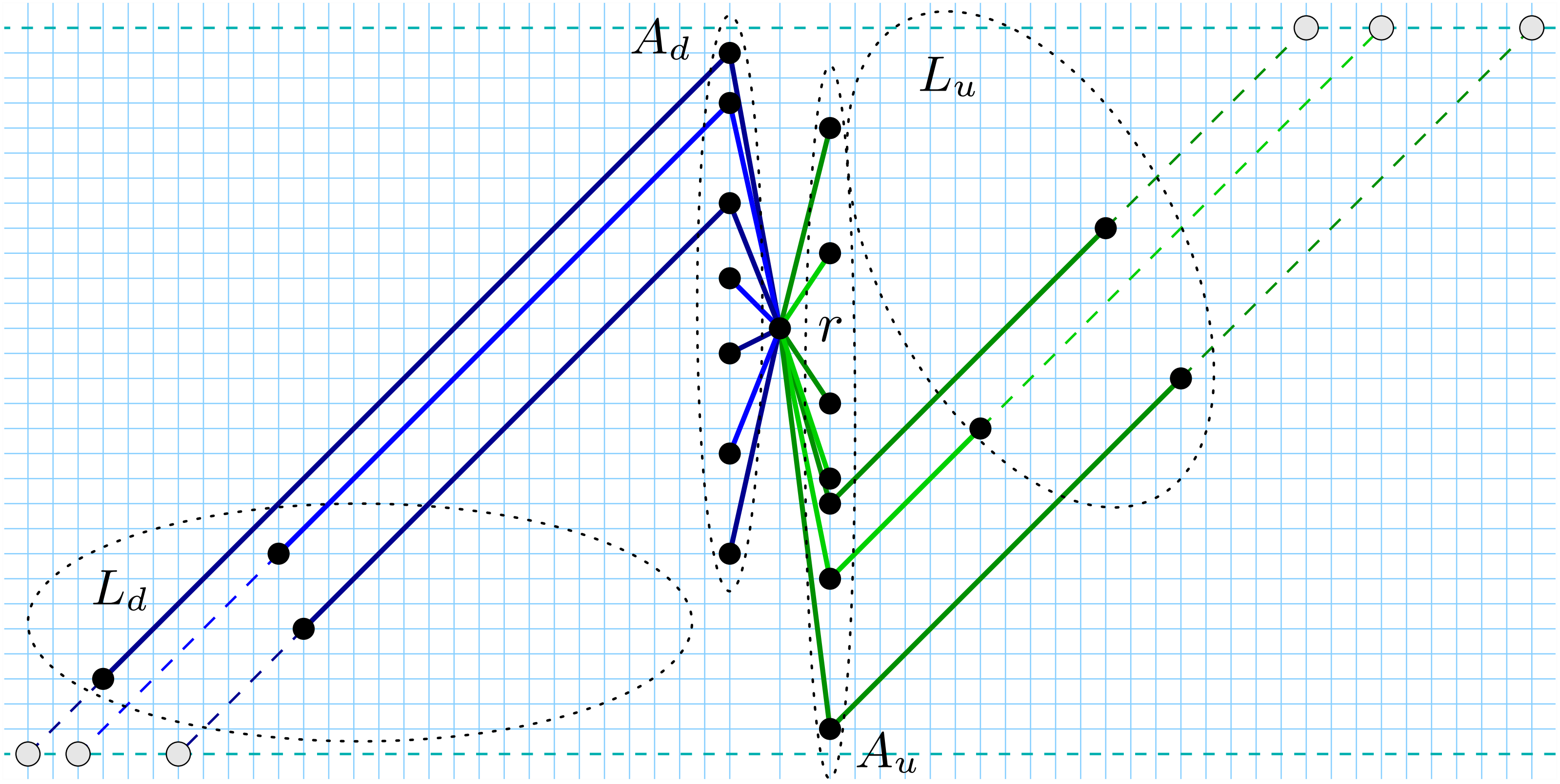
Lemma 4 *An n-vertex radius-2 star $T(V, E)$ can be n-level realized in $O(n)$ time on a $(2n+3) \times n$ grid for any vertex labeling $\phi : V \xrightarrow[\text{onto}]{1:1} \{1, 2, \ldots, n\}$.*



Radius-2 Stars – Linear Time Realization

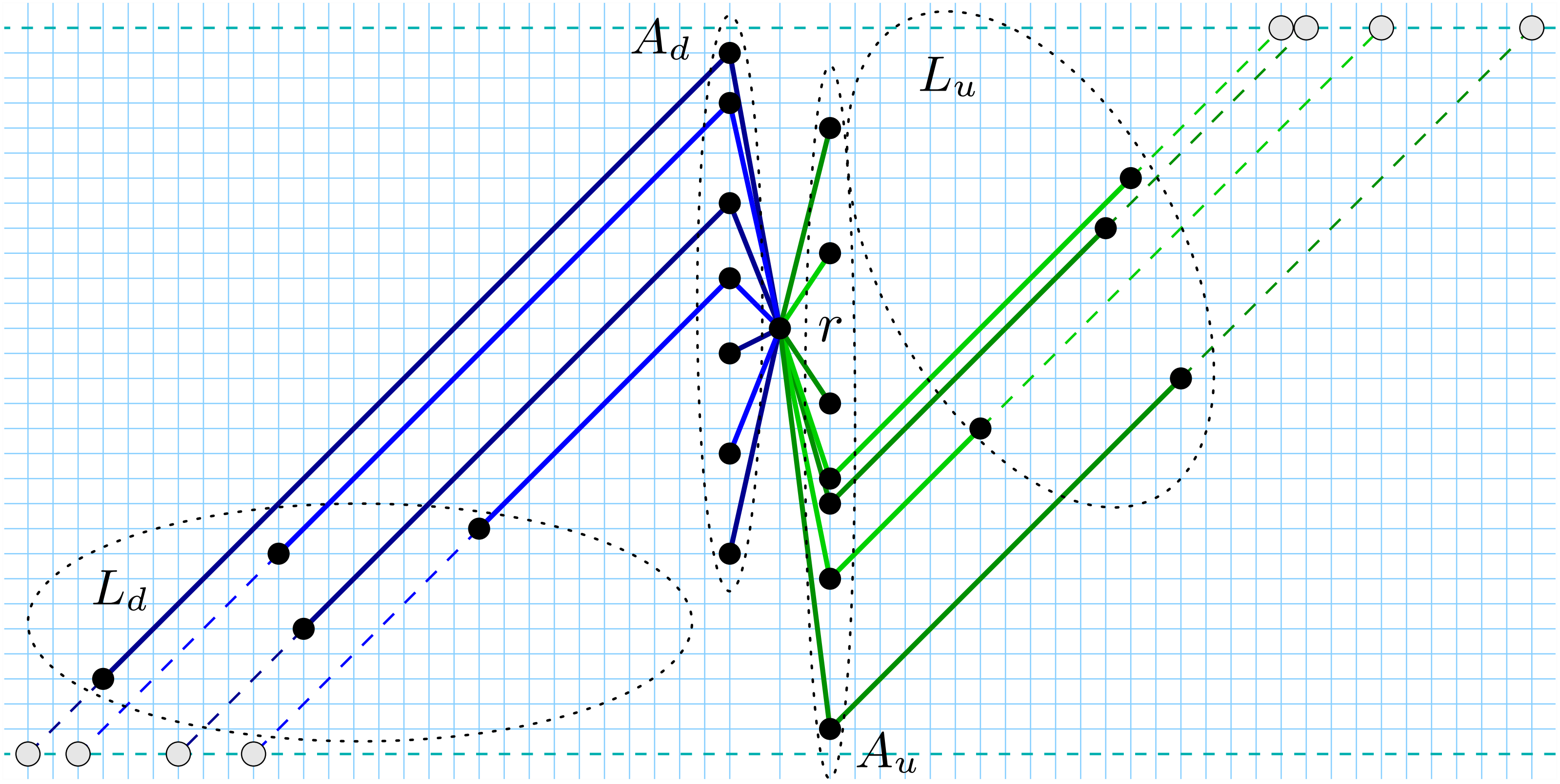
- An n-level realization of a radius-2 star in linear time yields the next lemma:

Lemma 4 *An n-vertex radius-2 star $T(V, E)$ can be n-level realized in $O(n)$ time on a $(2n+3) \times n$ grid for any vertex labeling $\phi : V \xrightarrow[\text{onto}]{1:1} \{1, 2, \ldots, n\}$.*



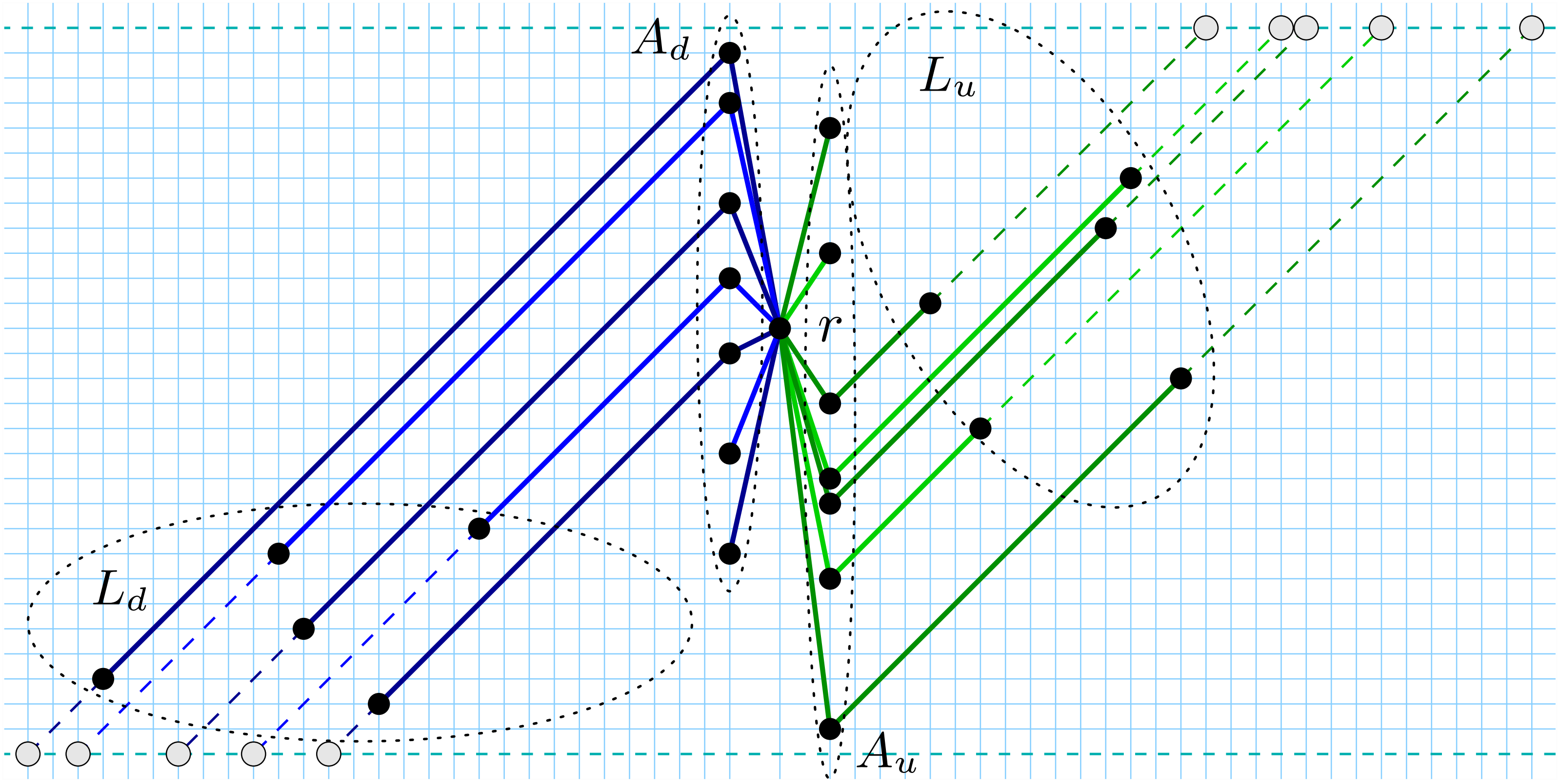
- An n-level realization of a radius-2 star in linear time yields the next lemma:

Lemma 4 *An n-vertex radius-2 star $T(V, E)$ can be n-level realized in $O(n)$ time on a $(2n+3) \times n$ grid for any vertex labeling $\phi : V \xrightarrow[\text{onto}]{1:1} \{1, 2, \ldots, n\}$.*



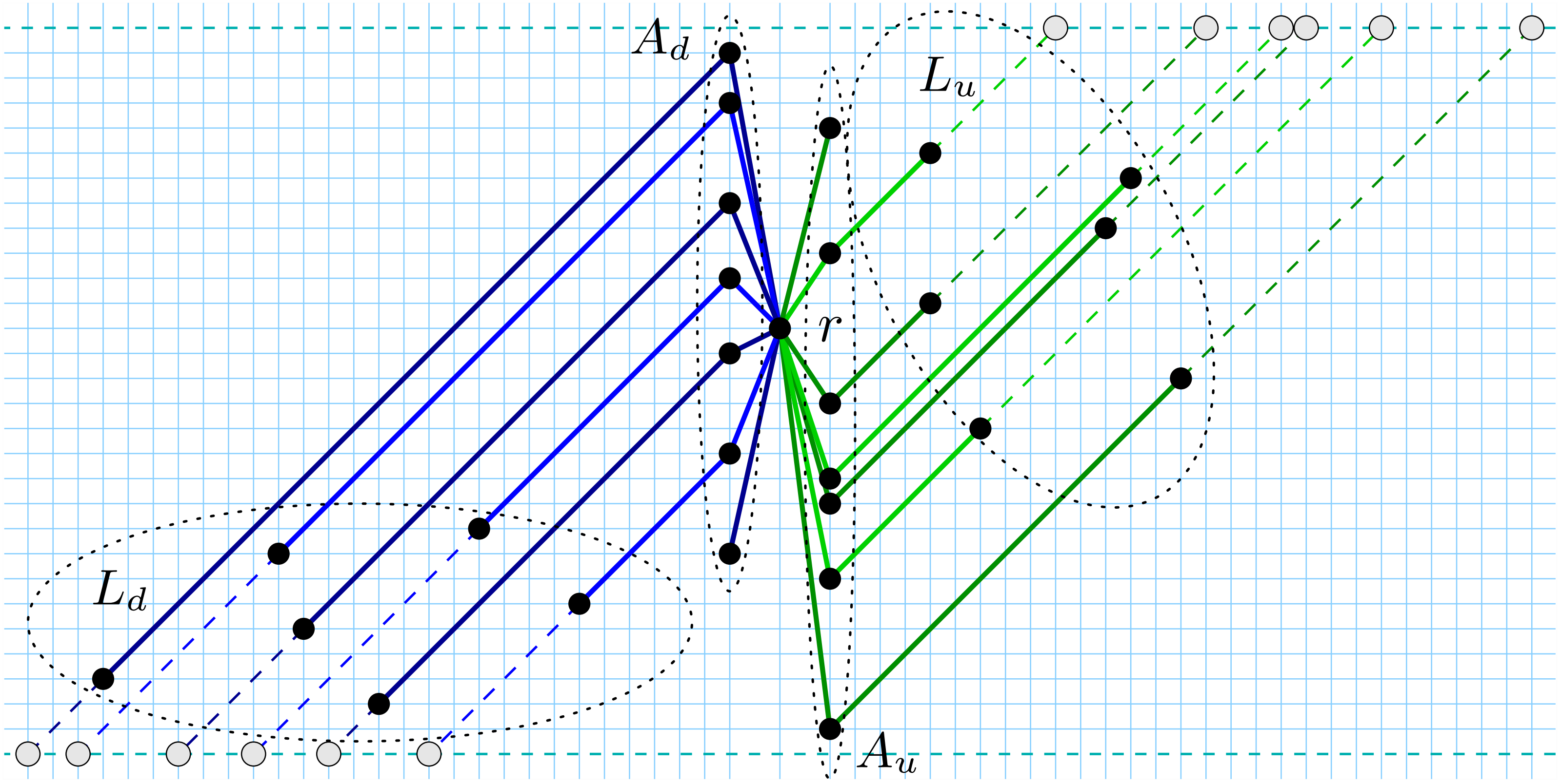
- An n-level realization of a radius-2 star in linear time yields the next lemma:

Lemma 4 *An n-vertex radius-2 star $T(V, E)$ can be n-level realized in $O(n)$ time on a $(2n + 3) \times n$ grid for any vertex labeling $\phi : V \xrightarrow[\text{onto}]{1:1} \{1, 2, \ldots, n\}$.*



- An n-level realization of a radius-2 star in linear time yields the next lemma:

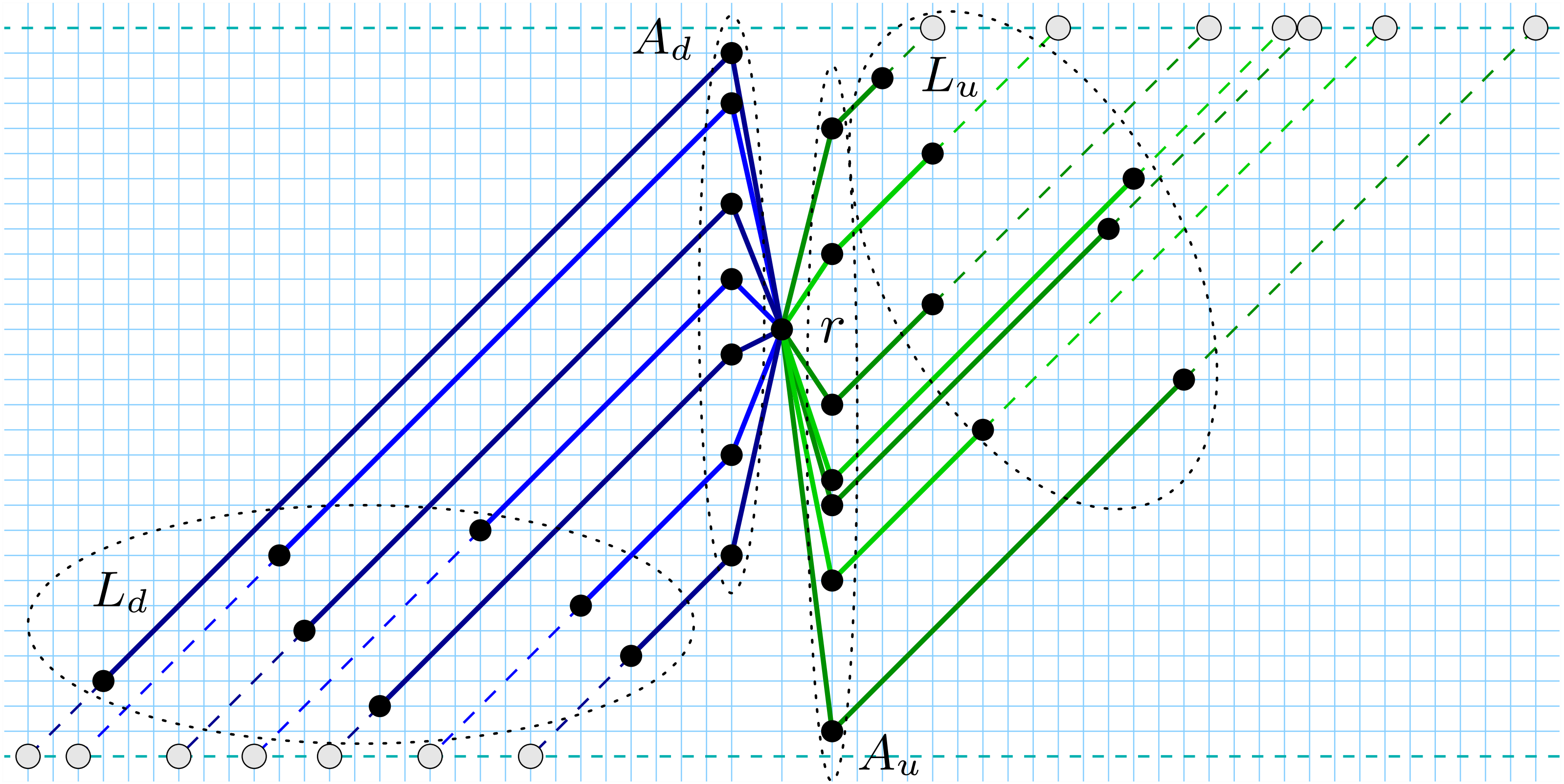
Lemma 4 *An n-vertex radius-2 star $T(V, E)$ can be n-level realized in $O(n)$ time on a $(2n+3) \times n$ grid for any vertex labeling $\phi : V \xrightarrow[\text{onto}]{1:1} \{1, 2, \ldots, n\}$.*



Radius-2 Stars – Linear Time Realization

- An n-level realization of a radius-2 star in linear time yields the next lemma:

Lemma 4 *An n-vertex radius-2 star $T(V, E)$ can be n-level realized in $O(n)$ time on a $(2n+3) \times n$ grid for any vertex labeling $\phi : V \xrightarrow[\text{onto}]{1:1} \{1, 2, \ldots, n\}$.*



Degree-3 Spiders – Linear Time Realizatio

- An n-level realization of a degree-3 spider in linear time gives the subsequent lemma:

Degree-3 Spiders – Linear Time Realizatio

- An n-level realization of a degree-3 spider in linear time gives the subsequent lemma:

Lemma 5 *An n-vertex degree-3 spider $T(V, E)$ can be n-level realized in $O(n)$ time for any vertex labeling $\phi : V \xrightarrow[\text{onto}]{1:1} \{1, 2, \ldots, n\}$.*

Degree-3 Spiders – Linear Time Realizatio

- An n-level realization of a degree-3 spider in linear time gives the subsequent lemma:

Lemma 5 *An n-vertex degree-3 spider $T(V, E)$ can be n-level realized in $O(n)$ time for any vertex labeling $\phi : V \xrightarrow[\text{onto}]{1:1} \{1, 2, \ldots, n\}$.*

- Proof idea:
  - First transform the degree-3 spider into a strictly expanding one

Degree-3 Spiders – Linear Time Realizatio

- An n-level realization of a degree-3 spider in linear time gives the subsequent lemma:

Lemma 5 *An n-vertex degree-3 spider $T(V, E)$ can be n-level realized in $O(n)$ time for any vertex labeling $\phi : V \xrightarrow[\text{onto}]{1:1} \{1, 2, \ldots, n\}$.*

- Proof idea:
  - First transform the degree-3 spider into a strictly expanding one
  - Then greedily draw the strictly expanding degree-3 spider starting from the root vertex and selecting the chain that has the least visibility until either

Degree-3 Spiders – Linear Time Realization

- An n-level realization of a degree-3 spider in linear time gives the subsequent lemma:

Lemma 5 *An n-vertex degree-3 spider $T(V, E)$ can be n-level realized in $O(n)$ time for any vertex labeling* $\phi : V \xrightarrow[\text{onto}]{1:1} \{1, 2, \ldots, n\}$.

- Proof idea:
  - First transform the degree-3 spider into a strictly expanding one
  - Then greedily draw the strictly expanding degree-3 spider starting from the root vertex and selecting the chain that has the least visibility until either
    - A new minimum or maximum vertex is obtained OR

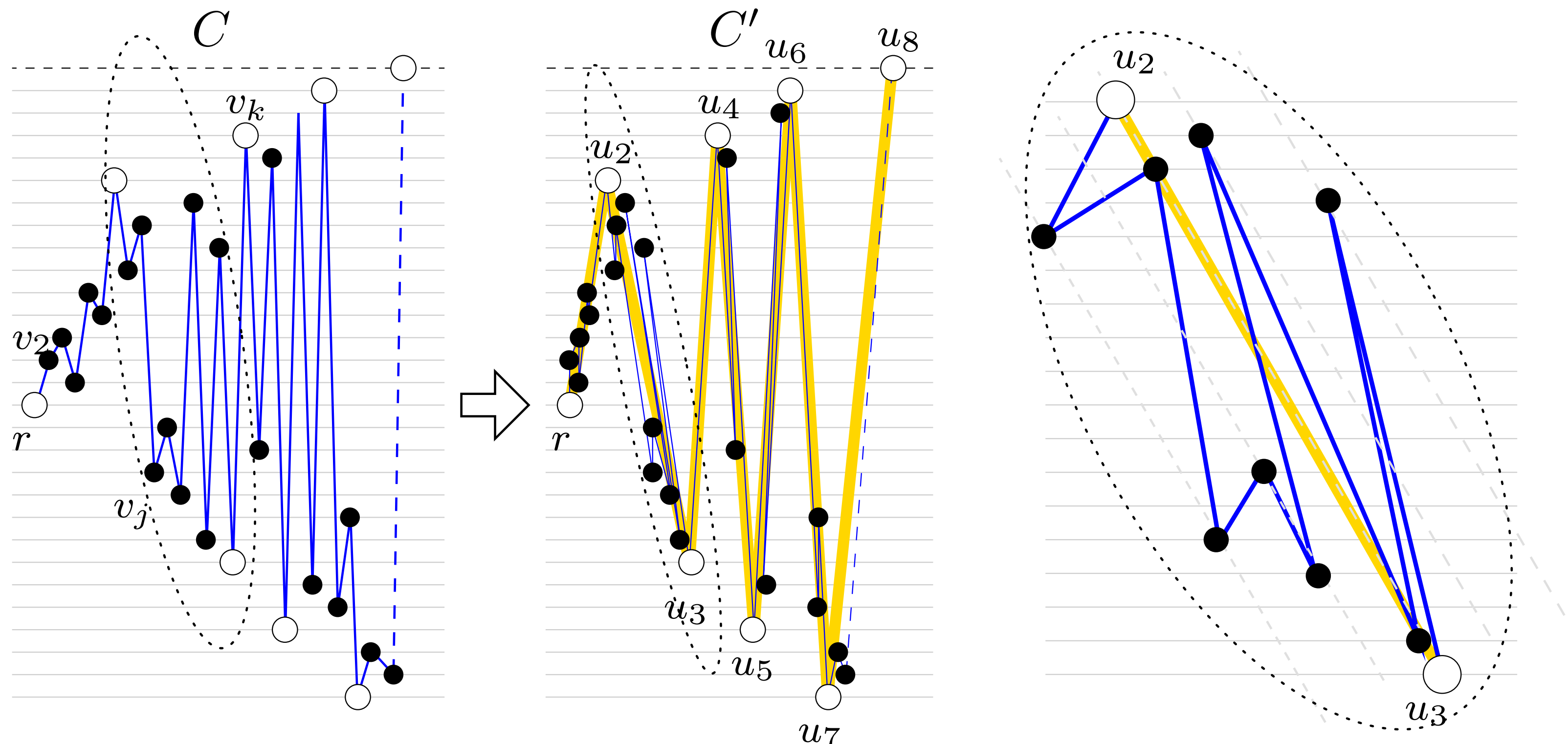
- An n-level realization of a degree-3 spider in linear time gives the subsequent lemma:

Lemma 5 *An n-vertex degree-3 spider $T(V, E)$ can be n-level realized in $O(n)$ time for any vertex labeling $\phi : V \xrightarrow[\text{onto}]{1:1} \{1, 2, \ldots, n\}$.*

- Proof idea:
  - First transform the degree-3 spider into a strictly expanding one
  - Then greedily draw the strictly expanding degree-3 spider starting from the root vertex and selecting the chain that has the least visibility until either
    - A new minimum or maximum vertex is obtained OR
    - The end of the chain is reached

- An n-level realization of a degree-3 spider in linear time gives the subsequent lemma:

Lemma 5 *An n-vertex degree-3 spider $T(V, E)$ can be n-level realized in $O(n)$ time for any vertex labeling* $\phi : V \xrightarrow[\text{onto}]{1:1} \{1, 2, \ldots, n\}$.



(a) (b) (c)

Degree-3 Spiders – Linear Time Realizatio

- An n-level realization of a degree-3 spider in linear time gives the subsequent lemma:

Lemma 5 *An n-vertex degree-3 spider $T(V, E)$ can be n-level realized in $O(n)$ time for any vertex labeling $\phi : V \xrightarrow[\text{onto}]{1:1} \{1, 2, \ldots, n\}$.*

Degree-3 Spiders – Linear Time Realization

- An n-level realization of a degree-3 spider in linear time gives the subsequent lemma:

Lemma 5 *An n-vertex degree-3 spider $T(V, E)$ can be n-level realized in $O(n)$ time for any vertex labeling* $\phi : V \xrightarrow[\text{onto}]{1:1} \{1, 2, \ldots, n\}$.

Degree-3 Spiders – Linear Time Realization

- An n-level realization of a degree-3 spider in linear time gives the subsequent lemma:

Lemma 5 *An n-vertex degree-3 spider $T(V, E)$ can be n-level realized in $O(n)$ time for any vertex labeling $\phi : V \xrightarrow[\text{onto}]{1:1} \{1, 2, \ldots, n\}$.*

Degree-3 Spiders – Linear Time Realizatio

- An n-level realization of a degree-3 spider in linear time gives the subsequent lemma:

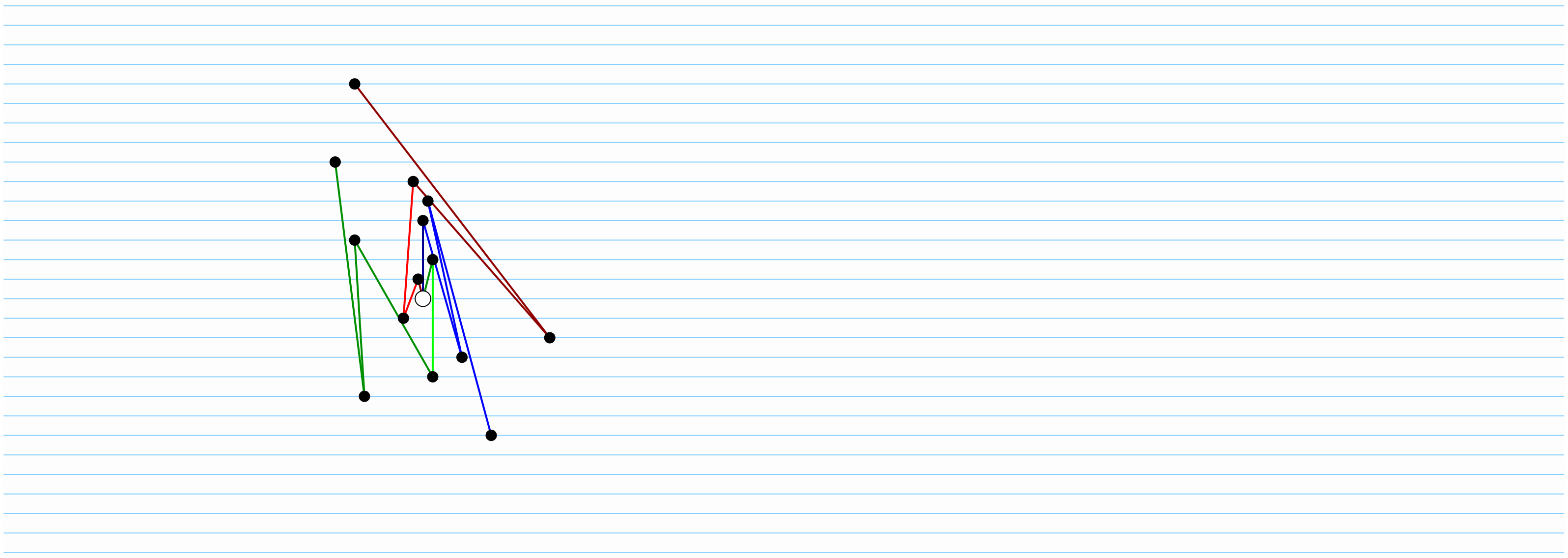
Lemma 5 *An n-vertex degree-3 spider $T(V, E)$ can be n-level realized in $O(n)$ time for any vertex labeling $\phi : V \xrightarrow[\text{onto}]{1:1} \{1, 2, \ldots, n\}$.*

- An n-level realization of a degree-3 spider in linear time gives the subsequent lemma:

Lemma 5 *An n-vertex degree-3 spider $T(V, E)$ can be n-level realized in $O(n)$ time for any vertex labeling* $\phi : V \xrightarrow[\text{onto}]{1:1} \{1, 2, \ldots, n\}$.

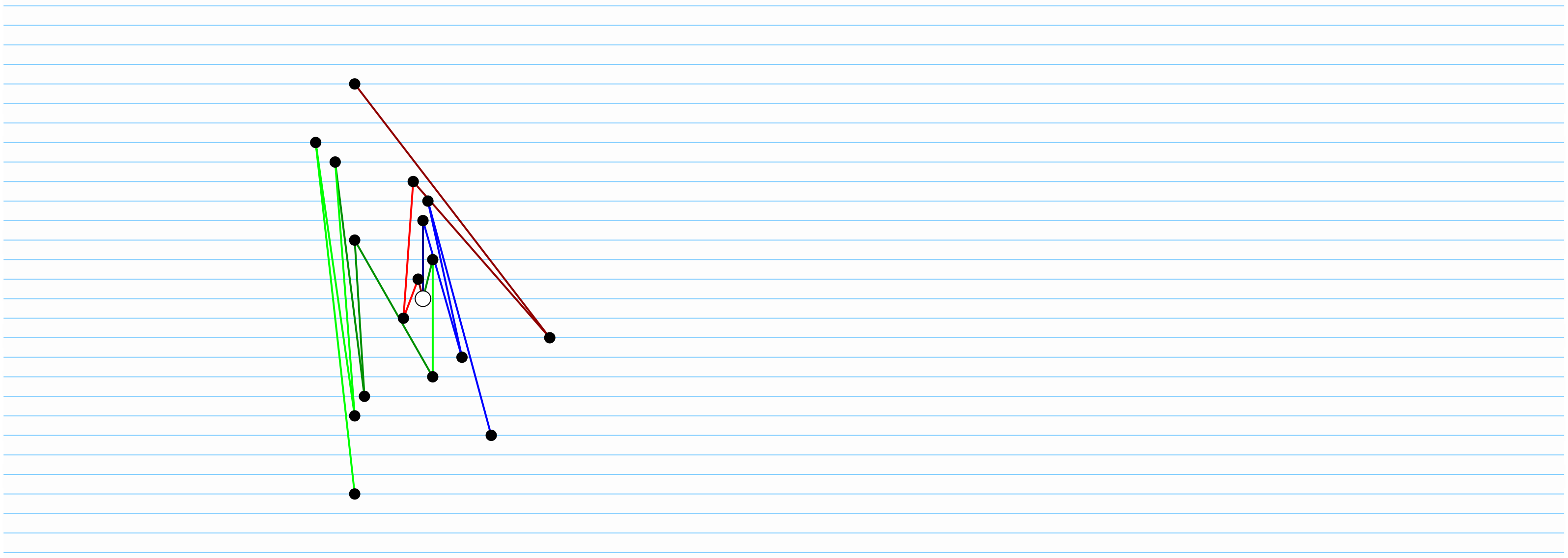
- An n-level realization of a degree-3 spider in linear time gives the subsequent lemma:

Lemma 5 *An n-vertex degree-3 spider $T(V, E)$ can be n-level realized in $O(n)$ time for any vertex labeling $\phi : V \xrightarrow[\text{onto}]{1:1} \{1, 2, \ldots, n\}$.*



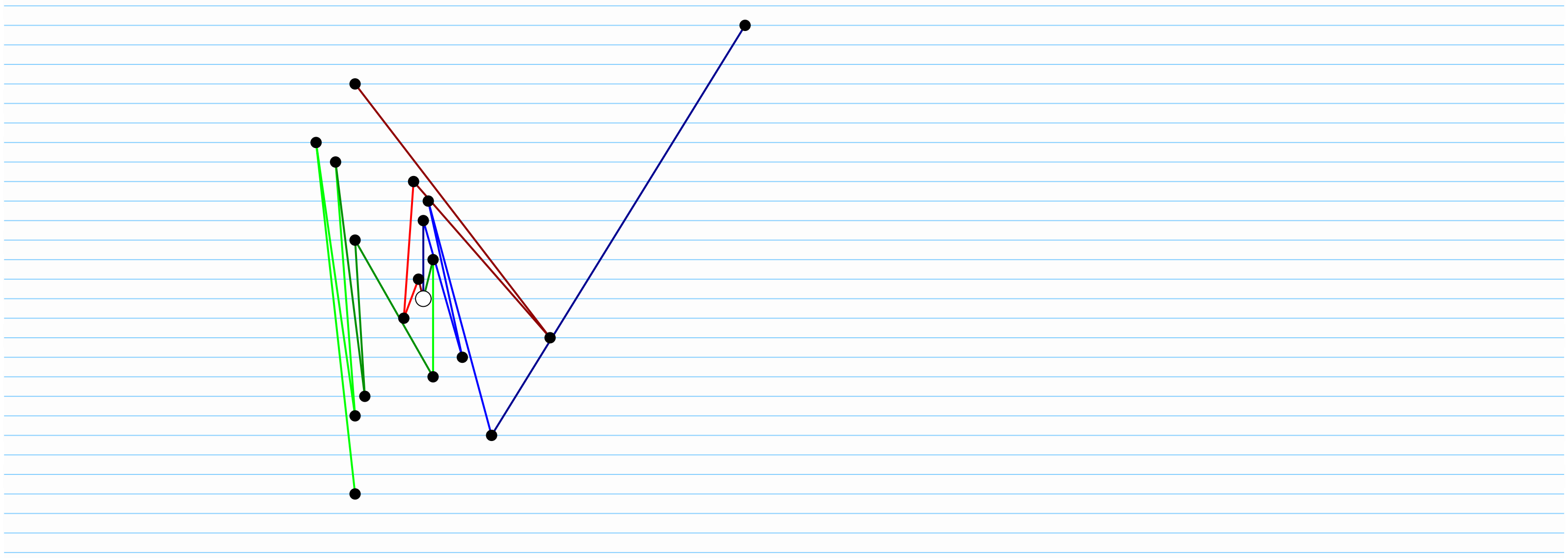
- An n-level realization of a degree-3 spider in linear time gives the subsequent lemma:

Lemma 5 *An n-vertex degree-3 spider $T(V, E)$ can be n-level realized in $O(n)$ time for any vertex labeling $\phi : V \xrightarrow[\text{onto}]{1:1} \{1, 2, \ldots, n\}$.*



- An n-level realization of a degree-3 spider in linear time gives the subsequent lemma:

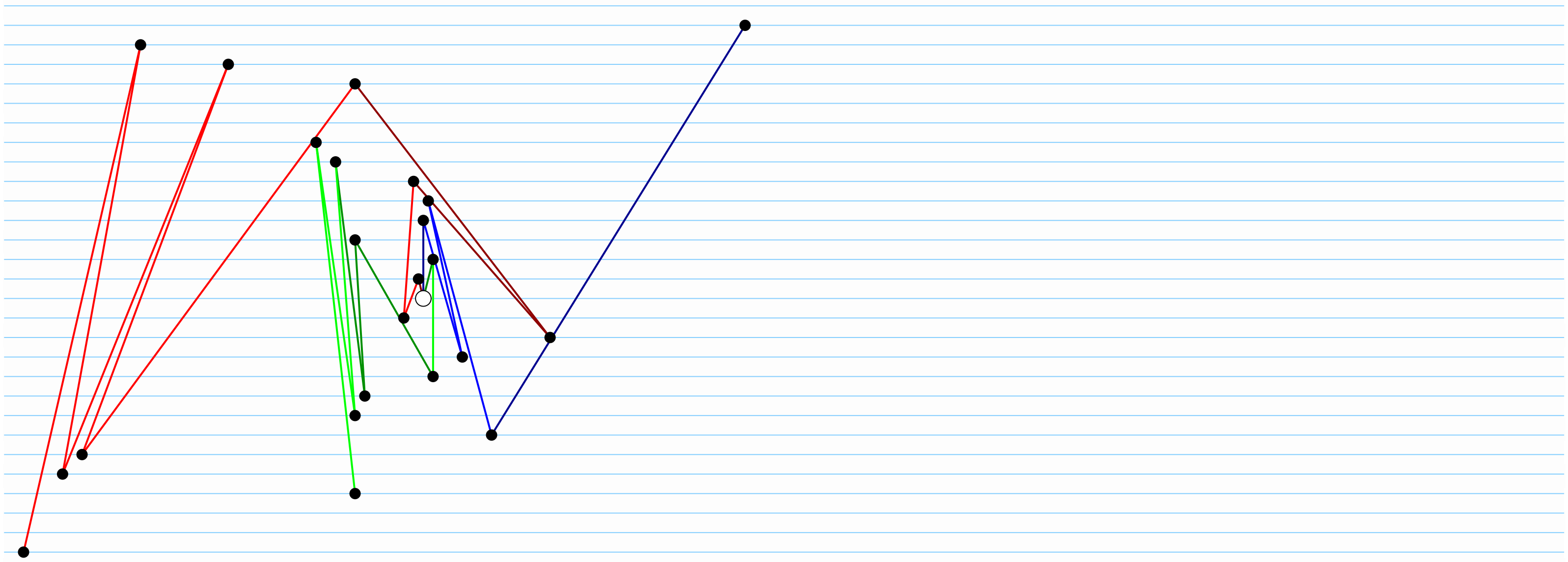
Lemma 5 *An n-vertex degree-3 spider $T(V, E)$ can be n-level realized in $O(n)$ time for any vertex labeling* $\phi : V \xrightarrow[\text{onto}]{1:1} \{1, 2, \ldots, n\}$.



Degree-3 Spiders – Linear Time Realization

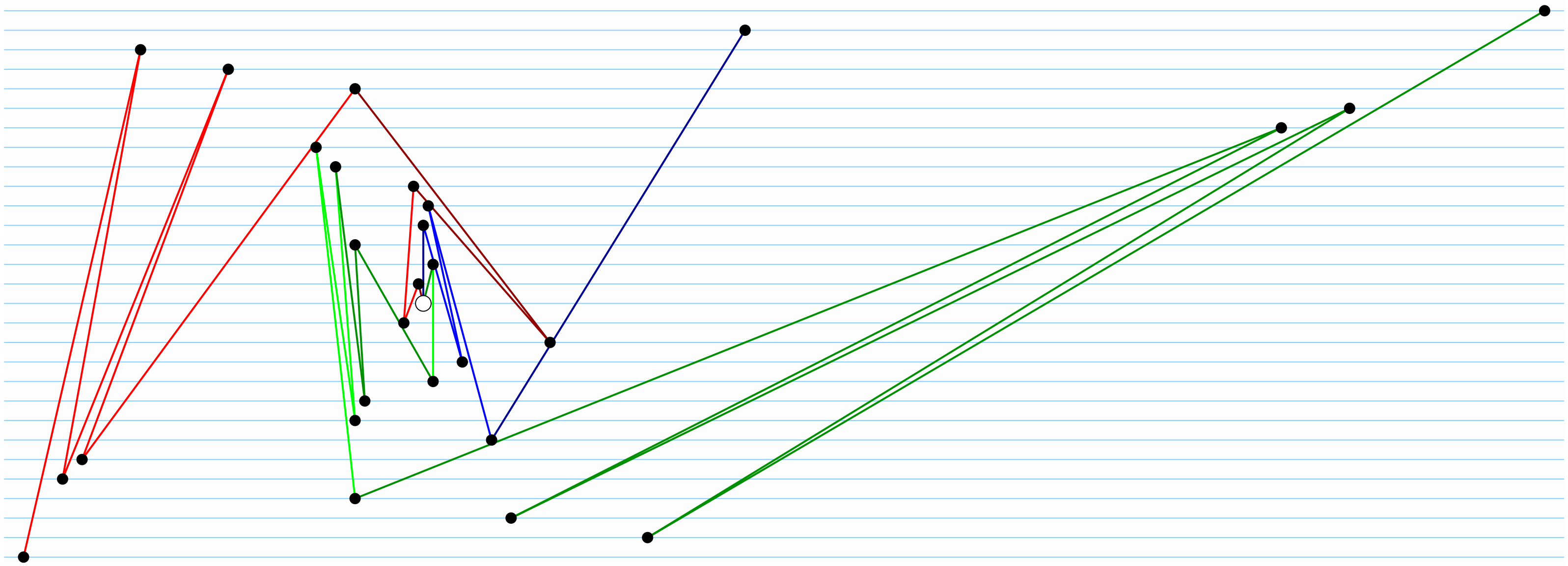
- An n-level realization of a degree-3 spider in linear time gives the subsequent lemma:

Lemma 5 *An n-vertex degree-3 spider $T(V, E)$ can be n-level realized in $O(n)$ time for any vertex labeling $\phi : V \xrightarrow[\text{onto}]{1:1} \{1, 2, \ldots, n\}$.*



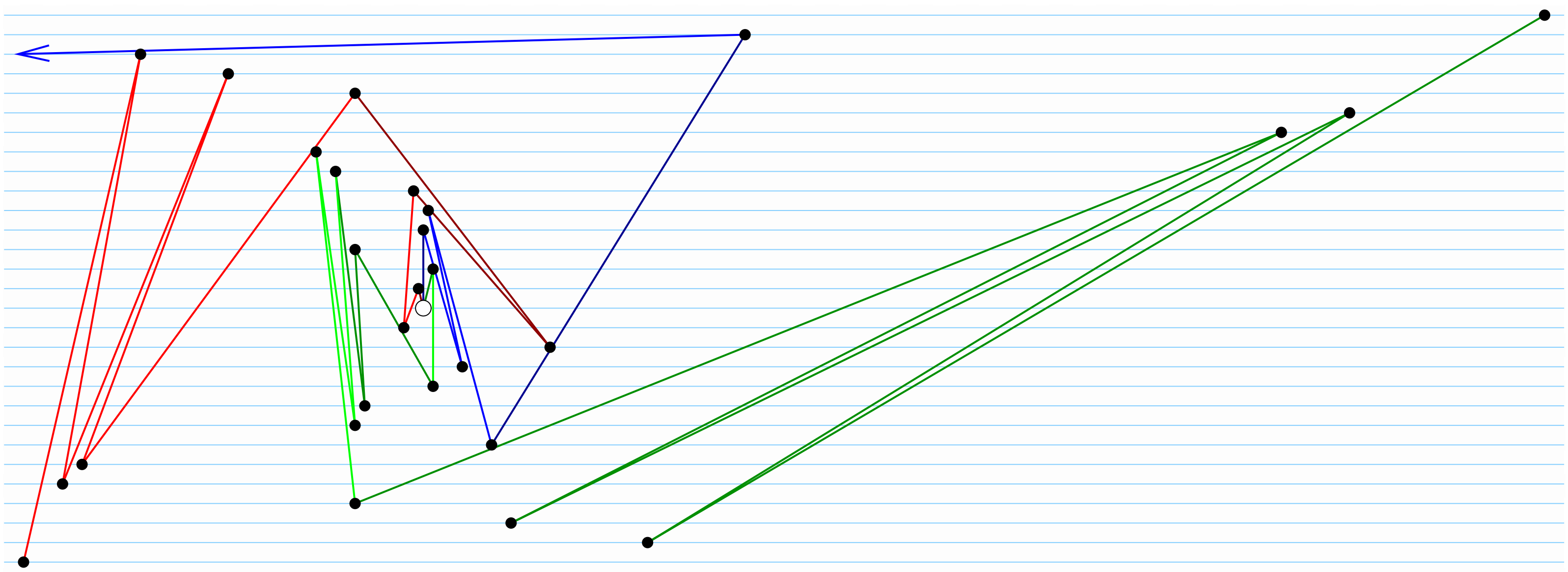
- An n-level realization of a degree-3 spider in linear time gives the subsequent lemma:

Lemma 5 *An n-vertex degree-3 spider $T(V, E)$ can be n-level realized in $O(n)$ time for any vertex labeling* $\phi : V \xrightarrow[\text{onto}]{1:1} \{1, 2, \ldots, n\}$.



- An n-level realization of a degree-3 spider in linear time gives the subsequent lemma:

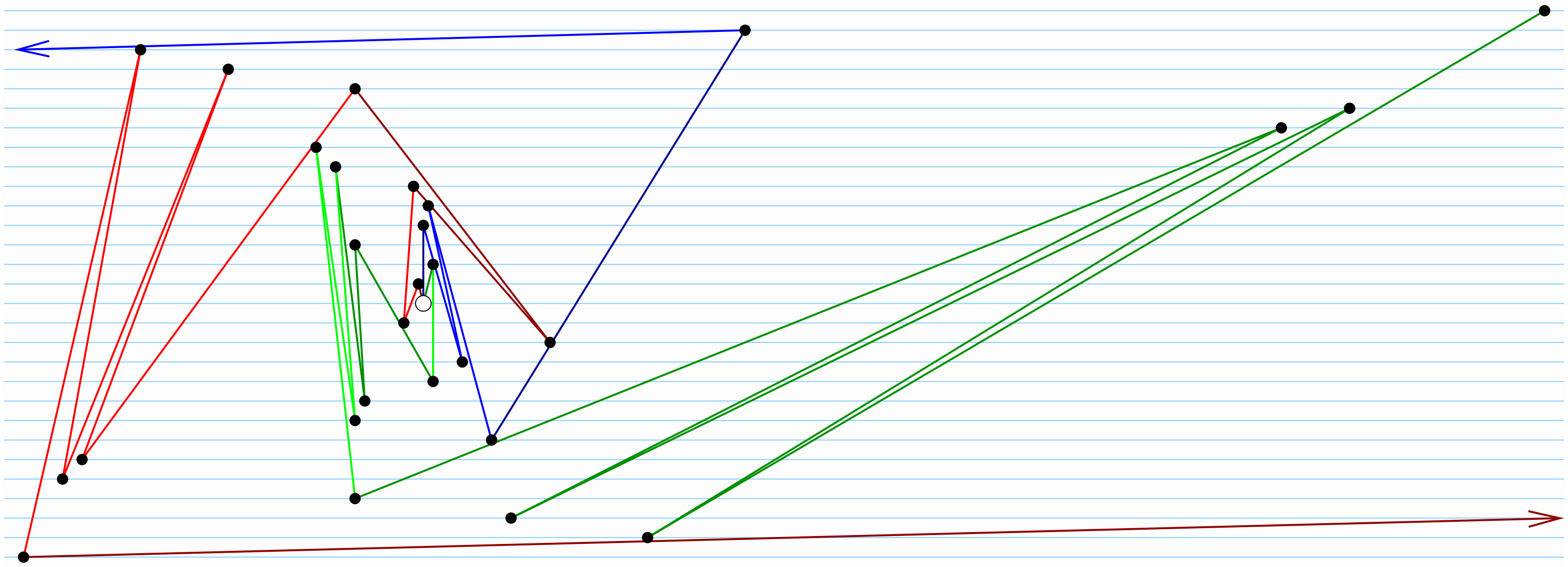
Lemma 5 *An n-vertex degree-3 spider $T(V, E)$ can be n-level realized in $O(n)$ time for any vertex labeling* $\phi : V \xrightarrow[\text{onto}]{1:1} \{1, 2, \ldots, n\}$.



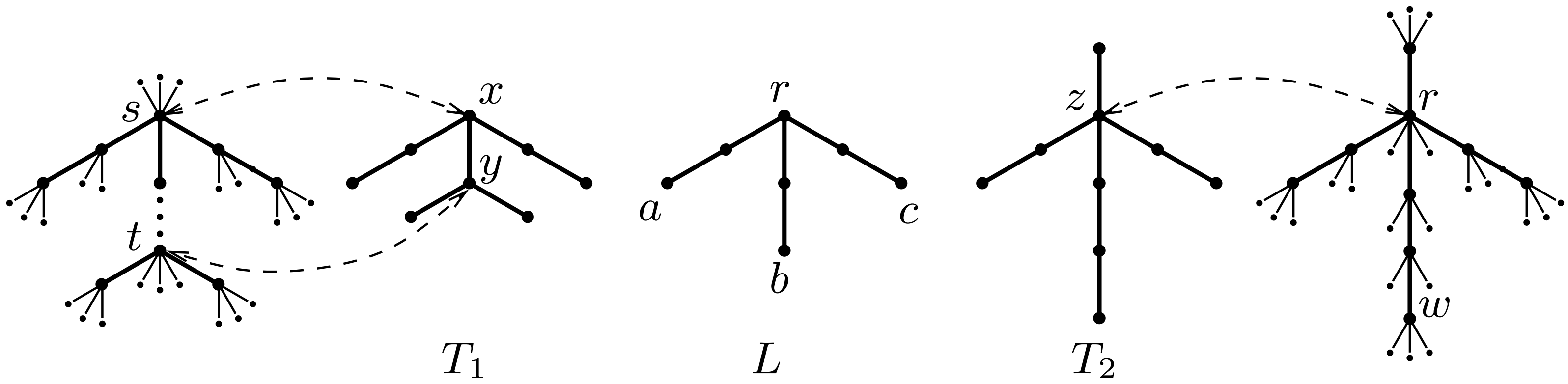
Degree-3 Spiders – Linear Time Realization

- An n-level realization of a degree-3 spider in linear time gives the subsequent lemma:

Lemma 5 *An n-vertex degree-3 spider $T(V, E)$ can be n-level realized in $O(n)$ time for any vertex labeling* $\phi : V \xrightarrow[\text{onto}]{1:1} \{1, 2, \ldots, n\}$.



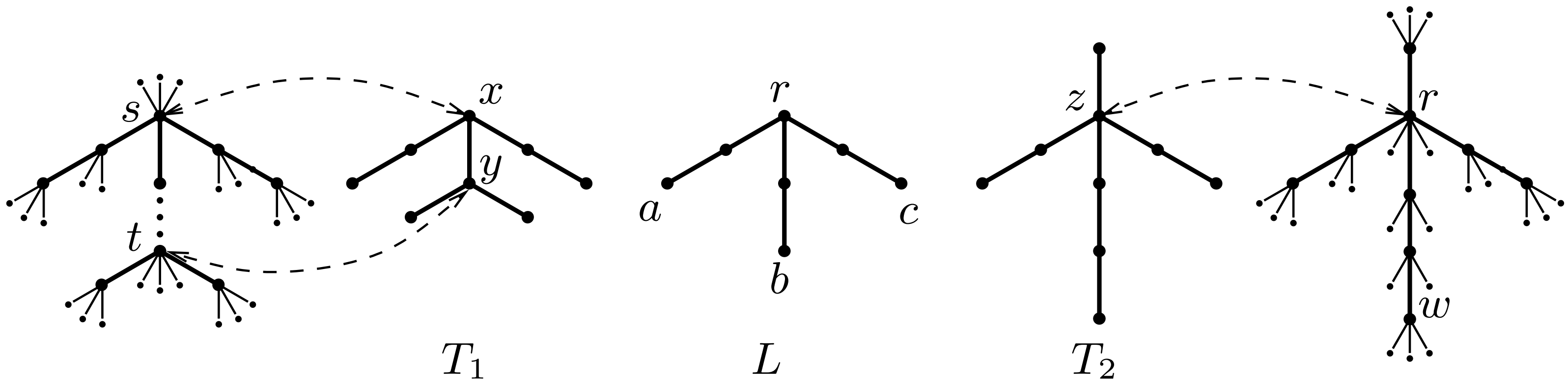
Forbidden Trees – Minimality



- Brute force consideration of the removal of edges from T_1 and T_2 gives the following lemma:

Lemma 6 *Removing any edge from T_1 or T_2 yields a forest of ULP trees.*

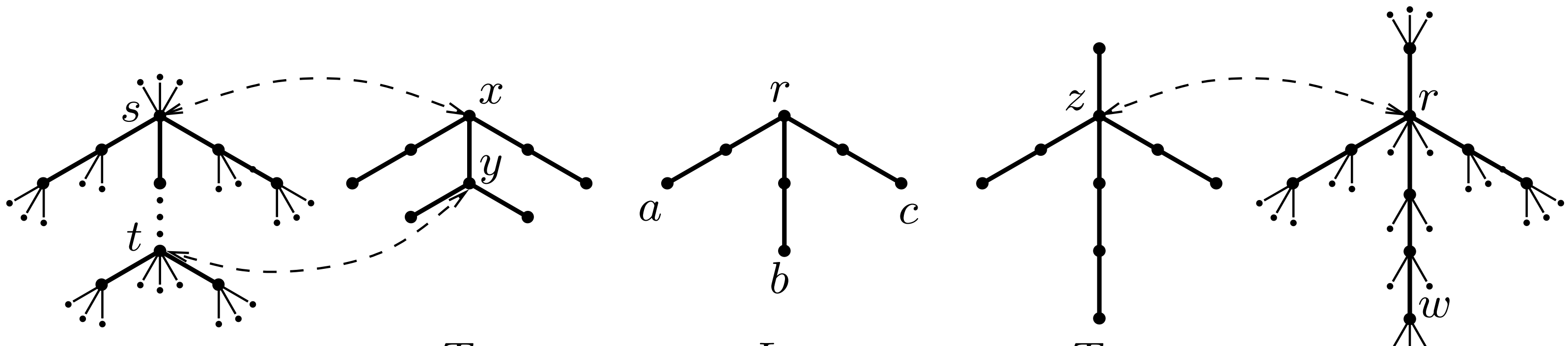
Forbidden Trees – Characterization



- A minimal lobster argument gives the next theorem:

Theorem 7 *Every tree either contains a subdivision of T_1 or T_2 in which case it is not ULP, or it is a caterpillar, a radius-2 star, or a 3 spider in which case it is ULP.*

Forbidden Trees – Characterization



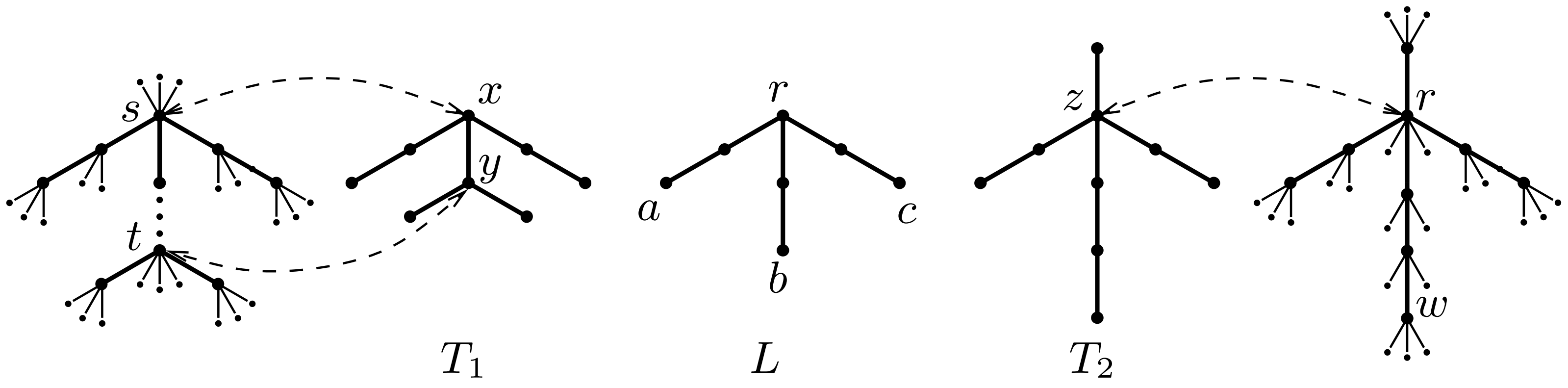
T_1 L T_2

- A minimal lobster argument gives the next theorem:

  Theorem 7 *Every tree either contains a subdivision of T_1 or T_2 in which case it is not ULP, or it is a caterpillar, a radius-2 star, or a 3 spider in which case it is ULP.*

- Proof idea:

Forbidden Trees – Characterization

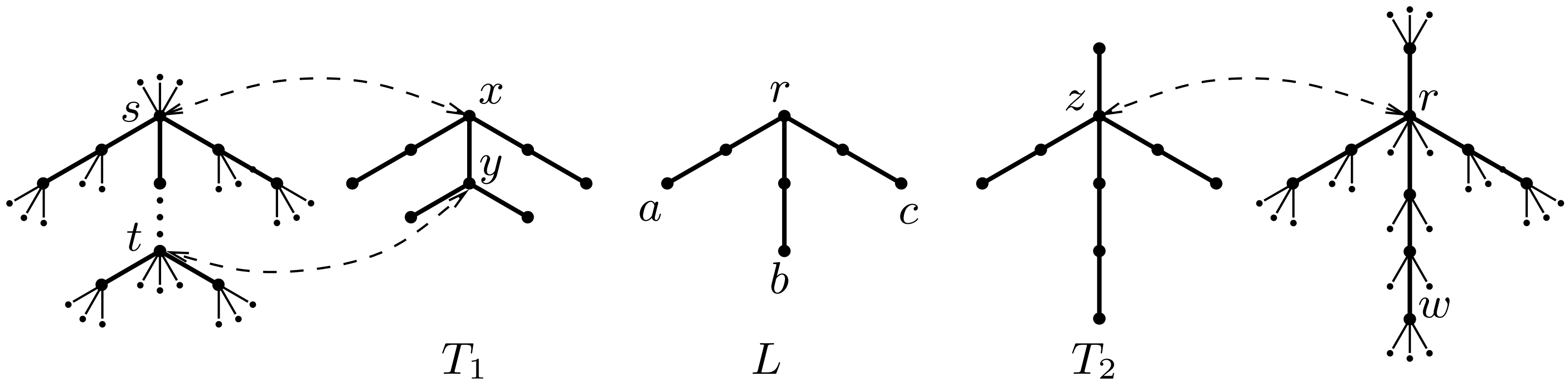


- A minimal lobster argument gives the next theorem:

  Theorem 7 *Every tree either contains a subdivision of T_1 or T_2 in which case it is not ULP, or it is a caterpillar, a radius-2 star, or a 3 spider in which case it is ULP.*

- Proof idea:
  - T_1 and T_2 are not caterpillars, radius-2 stars, or degree-3 spiders

Forbidden Trees – Characterization

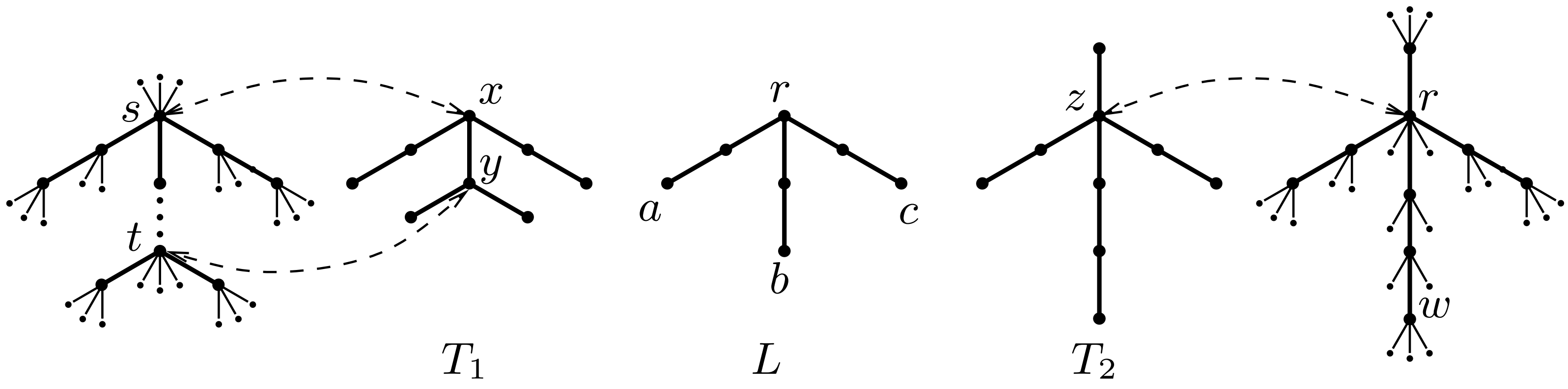


- A minimal lobster argument gives the next theorem:

  Theorem 7 *Every tree either contains a subdivision of* T_1 *or* T_2 *in which case it is not ULP, or it is a caterpillar, a radius-*2 *star, or a 3 spider in which case it is ULP.*

- Proof idea:
  - T_1 and T_2 are not caterpillars, radius-2 stars, or degree-3 spiders
  - A graph that is not a caterpillar has a minimal lobster L.

Forbidden Trees – Characterization

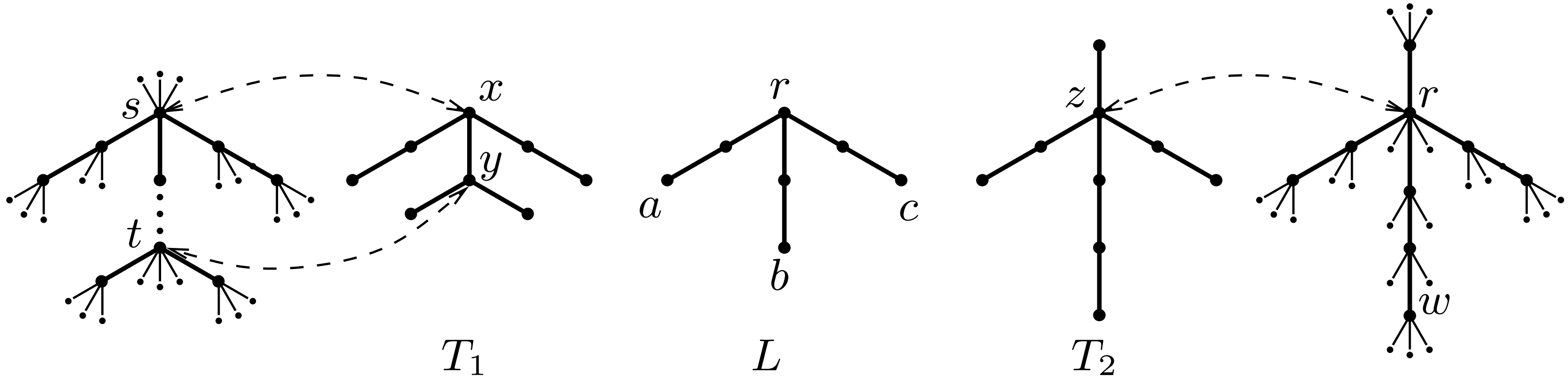


- A minimal lobster argument gives the next theorem:

  Theorem 7 *Every tree either contains a subdivision of T_1 or T_2 in which case it is not ULP, or it is a caterpillar, a radius-2 star, or a 3 spider in which case it is ULP.*

- Proof idea:
  - T_1 and T_2 are not caterpillars, radius-2 stars, or degree-3 spiders
  - A graph that is not a caterpillar has a minimal lobster L.
  - A graph that is not a degree-3 spider has two cases

Forbidden Trees – Characterization

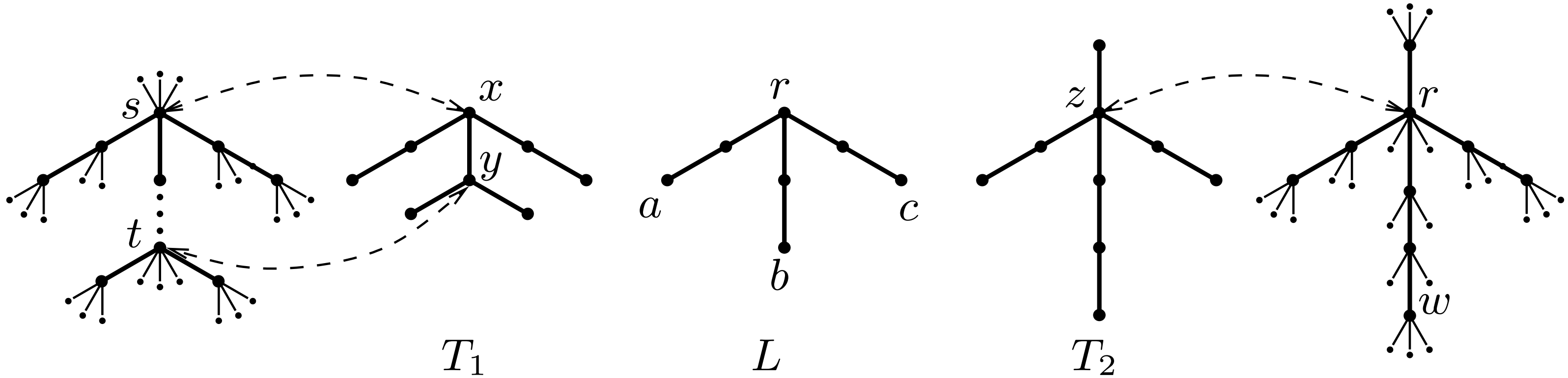


- A minimal lobster argument gives the next theorem:

  Theorem 7 *Every tree either contains a subdivision of T_1 or T_2 in which case it is not ULP, or it is a caterpillar, a radius-2 star, or a 3 spider in which case it is ULP.*

- Proof idea:
  - T_1 and T_2 are not caterpillars, radius-2 stars, or degree-3 spiders
  - A graph that is not a caterpillar has a minimal lobster L.
  - A graph that is not a degree-3 spider has two cases
    - Has at least two vertices of degree-3–contains T_1

Forbidden Trees – Characterization



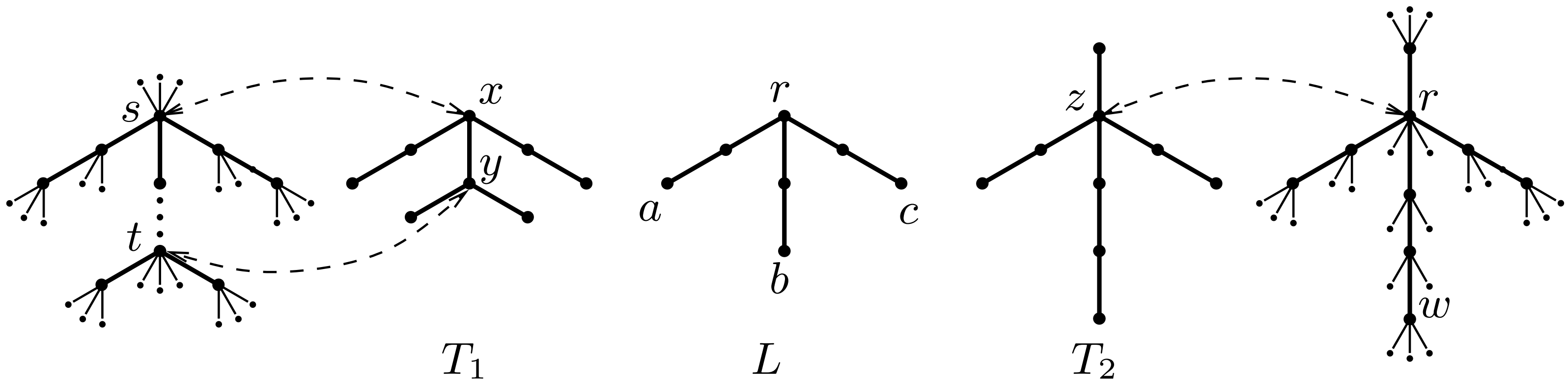
T_1 L T_2

- A minimal lobster argument gives the next theorem:

  Theorem 7 *Every tree either contains a subdivision of T_1 or T_2 in which case it is not ULP, or it is a caterpillar, a radius-2 star, or a 3 spider in which case it is ULP.*

- Proof idea:
  - T_1 and T_2 are not caterpillars, radius-2 stars, or degree-3 spiders
  - A graph that is not a caterpillar has a minimal lobster L.
  - A graph that is not a degree-3 spider has two cases
    - Has at least two vertices of degree-3–contains T_1
    - Has at least one vertex of degree-4–contains T_2

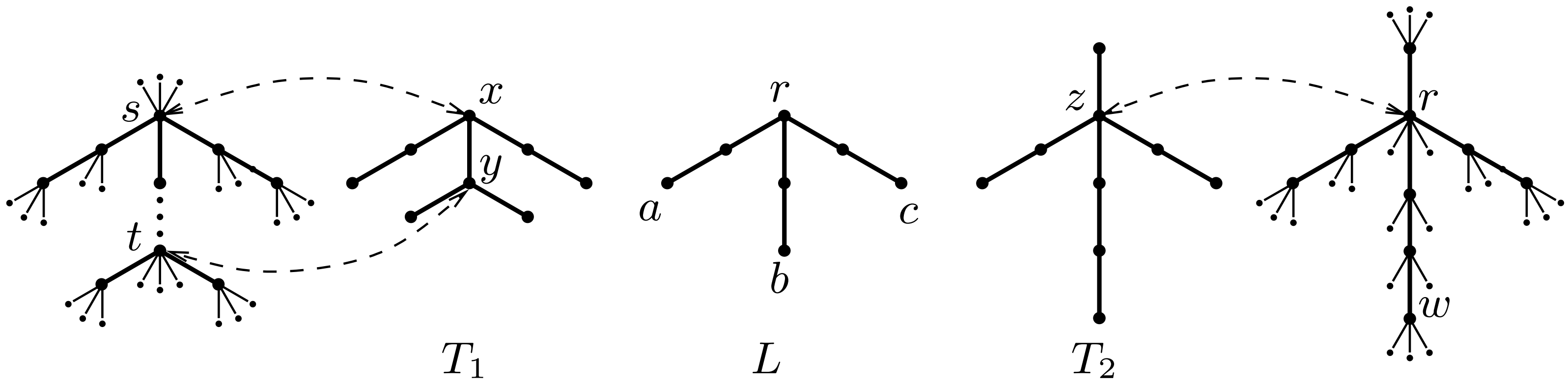
ULP Trees – Recognition



- The final corollary is a consequence of Theorem 7 and properties of degree sequences of caterpillars, radius-2 stars, and degree-3 spiders (Lemmas 8, 9, 10, resp.):

  Corollary 11 *The class of ULP trees can be recognized in linear time.*

ULP Trees – Recognition

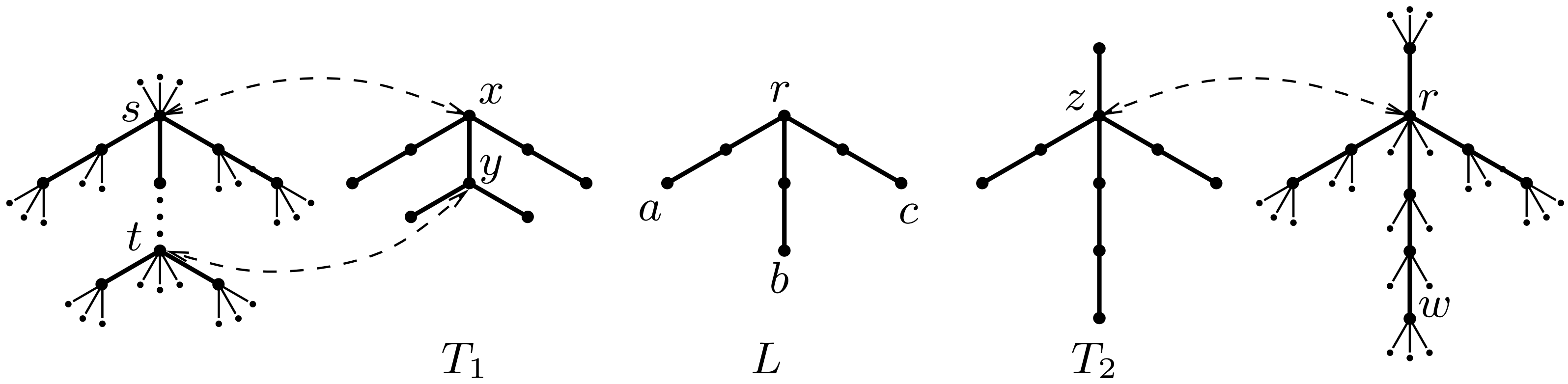


- The final corollary is a consequence of Theorem 7 and properties of degree sequences of caterpillars, radius-2 stars, and degree-3 spiders (Lemmas 8, 9, 10, resp.):

  Corollary 11 *The class of ULP trees can be recognized in linear time.*

- Proof idea:

ULP Trees – Recognition

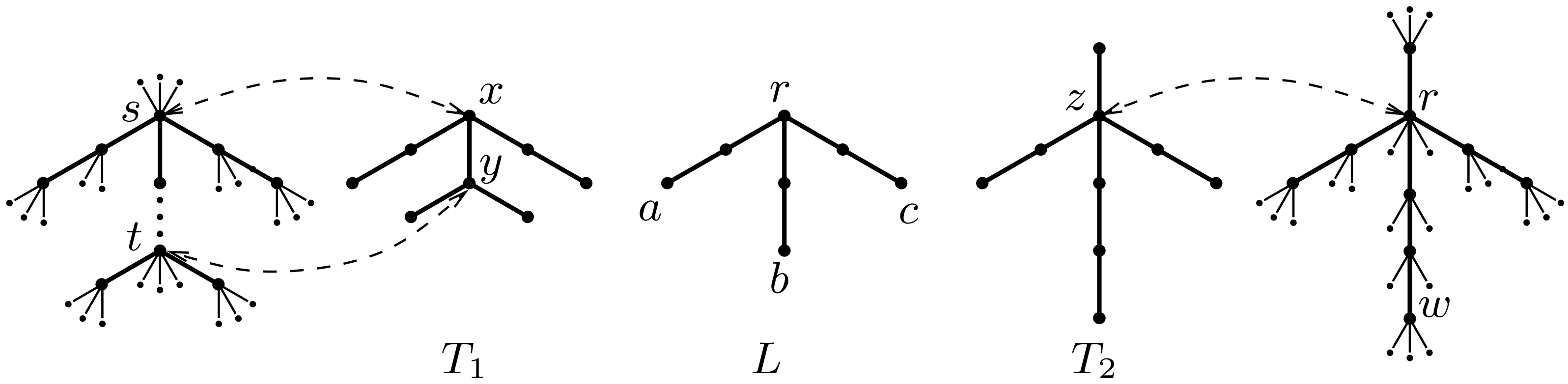


- The final corollary is a consequence of Theorem 7 and properties of degree sequences of caterpillars, radius-2 stars, and degree-3 spiders (Lemmas 8, 9, 10, resp.):

  Corollary 11 *The class of ULP trees can be recognized in linear time.*

- Proof idea:
  - First check the degree sequence to see if the graph is a degree-3 spider

ULP Trees – Recognition

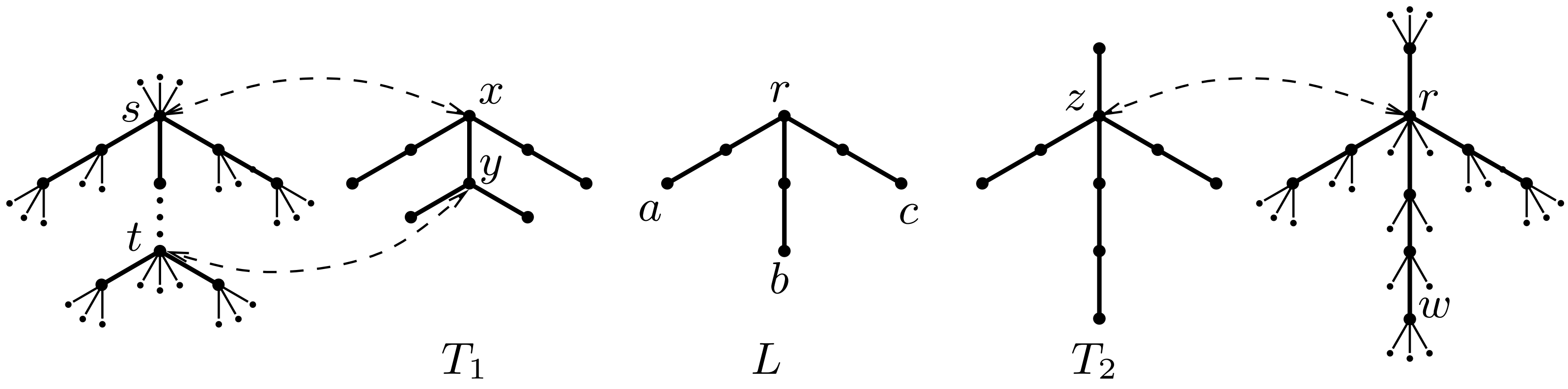


- The final corollary is a consequence of Theorem 7 and properties of degree sequences of caterpillars, radius-2 stars, and degree-3 spiders (Lemmas 8, 9, 10, resp.):

  Corollary 11 *The class of ULP trees can be recognized in linear time.*

- Proof idea:
  - First check the degree sequence to see if the graph is a degree-3 spider
  - Else strip off degree-1 vertices to see if the remaining graph is a

ULP Trees – Recognition

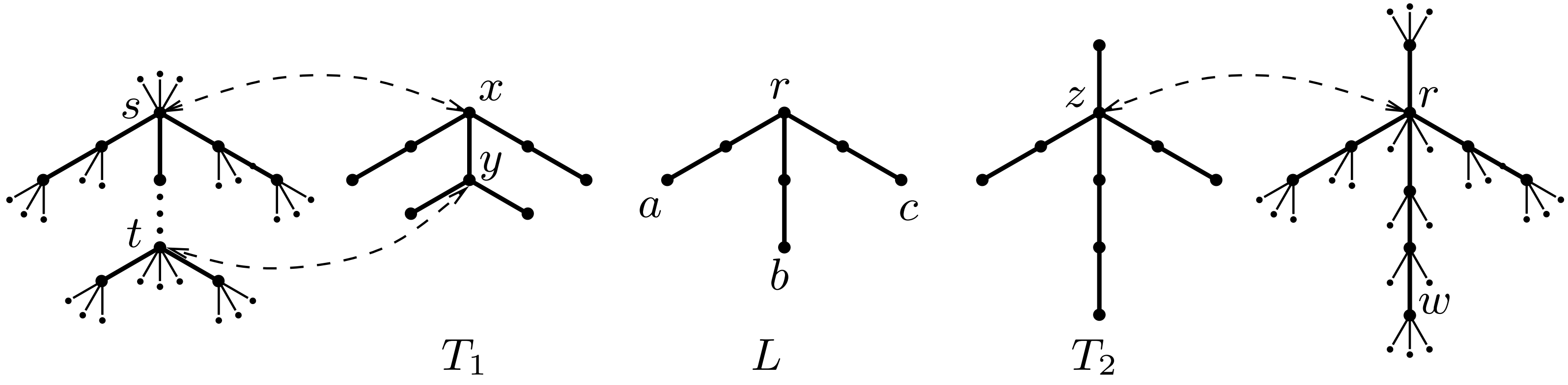


- The final corollary is a consequence of Theorem 7 and properties of degree sequences of caterpillars, radius-2 stars, and degree-3 spiders (Lemmas 8, 9, 10, resp.):

  Corollary 11 *The class of ULP trees can be recognized in linear time.*

- Proof idea:
  - First check the degree sequence to see if the graph is a degree-3 spider
  - Else strip off degree-1 vertices to see if the remaining graph is a
    - **Path** in which case it is a caterpillar

ULP Trees – Recognition



- The final corollary is a consequence of Theorem 7 and properties of degree sequences of caterpillars, radius-2 stars, and degree-3 spiders (Lemmas 8, 9, 10, resp.):

  Corollary 11 *The class of ULP trees can be recognized in linear time.*

- Proof idea:
  - First check the degree sequence to see if the graph is a degree-3 spider
  - Else strip off degree-1 vertices to see if the remaining graph is a
    - **Path** in which case it is a caterpillar
    - **Star** in which case it is a radius-2 star

Future Work

- Provide certificate of unlabeled level non-planarity
  - I.e., find copy of T_1 or T_2

Future Work

- Provide certificate of unlabeled level non-planarity
  - I.e., find copy of T_1 or T_2
- Provide similar characterization of all graphs

Future Work

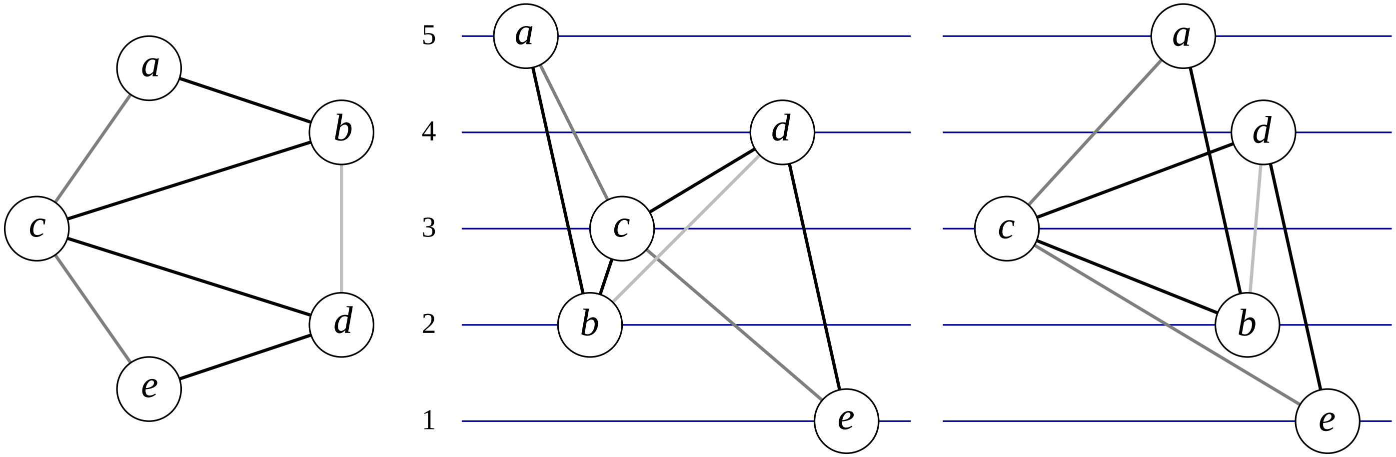
- Provide certificate of unlabeled level non-planarity
  - I.e., find copy of T_1 or T_2
- Provide similar characterization of all graphs
- Also provide recognition algorithm for all graphs

Preview of Future Work

- There are five forbidden ULP subdivisions with cycles

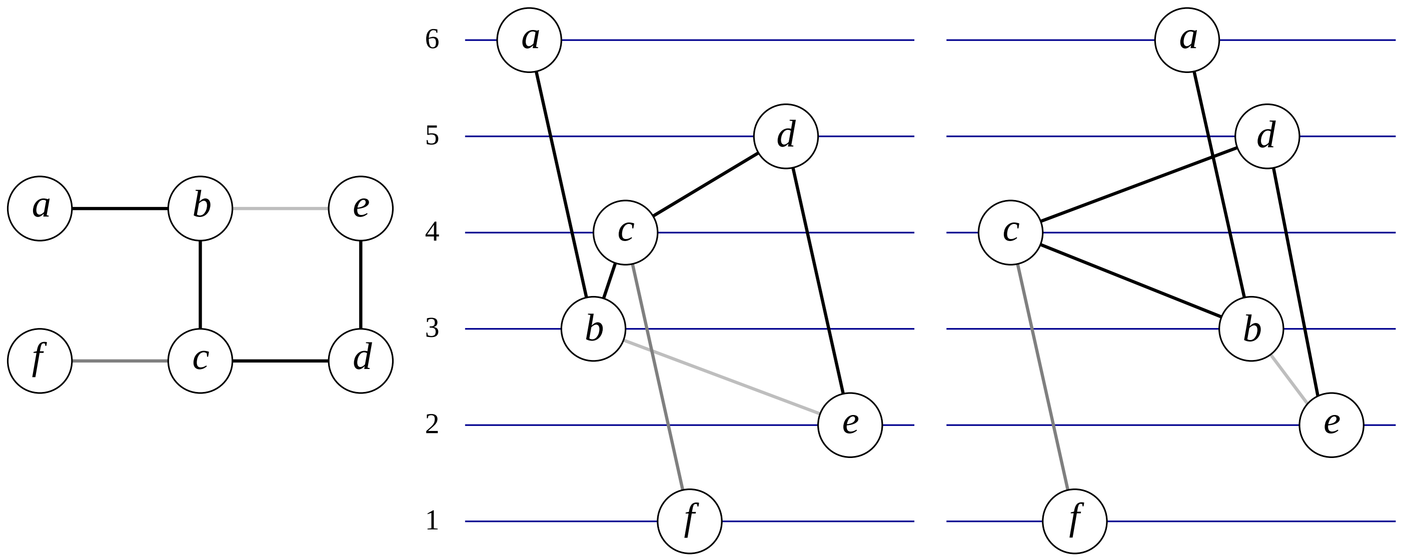
Preview of Future Work

- There are five forbidden ULP subdivisions with cycles
  - First has 5 vertices and two degree 4 vertices



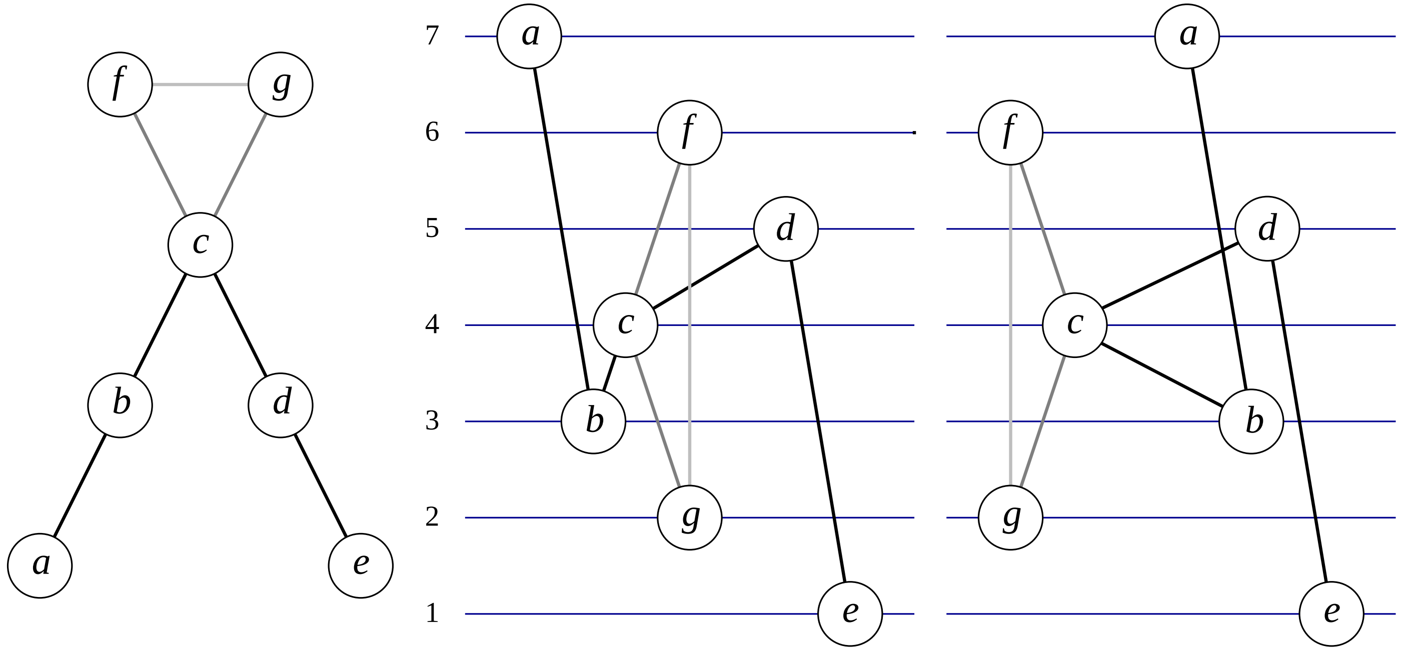
Preview of Future Work

- There are five forbidden ULP subdivisions with cycles
  - Second has 6 vertices and one 4-cycle



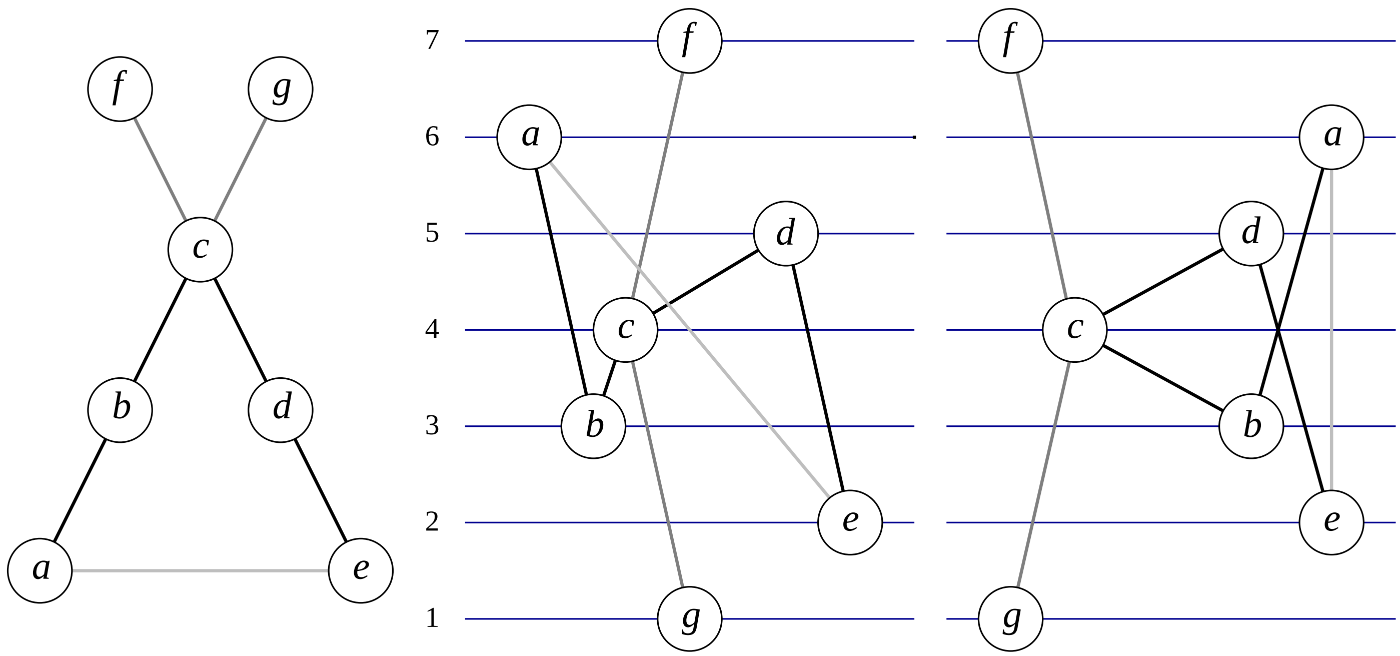
Preview of Future Work

- There are five forbidden ULP subdivisions with cycles
  - Third has 7 vertices and one 3-cycle and one degree 4 vertex



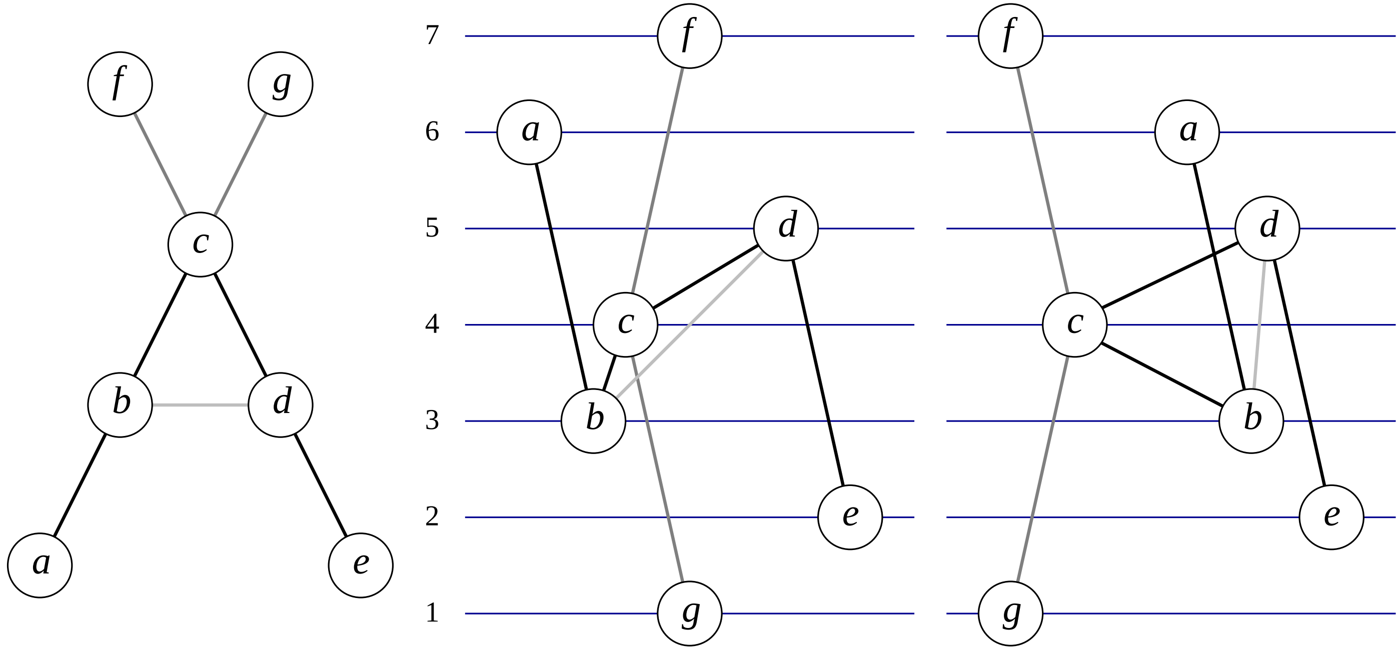
Preview of Future Work

- There are five forbidden ULP subdivisions with cycles
  - Fourth has 7 vertices and one 5-cycle and one degree 4 vertex



Preview of Future Work

- There are five forbidden ULP subdivisions with cycles
  - Third has 7 vertices and one 3-cycle and one degree 4 vertex and two degree 3 vertices



Thank You!