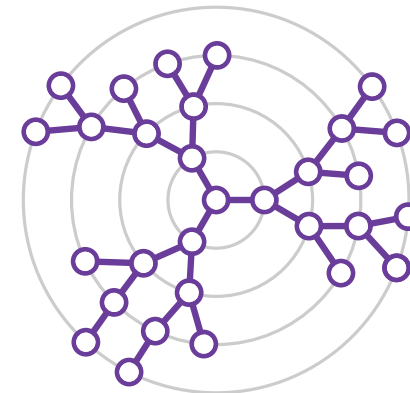
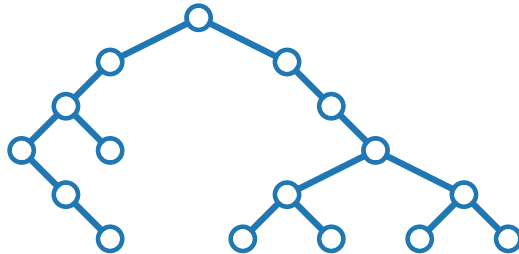
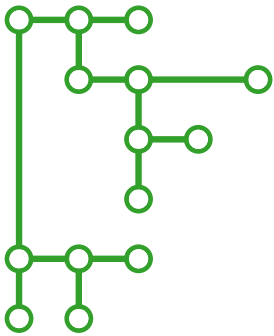


Visualization of Graphs

Lecture 1b: Drawing Trees

Part I: Layered Drawings

Johannes Zink



(Rooted) Trees

Leaf: vertex of degree 1

Rooted tree: tree with a designated **root**

Ancestor: vertex on the path to the root

Parent: neighbor on the path to the root

Successor: vertex on the path away from the root

Child: neighbor not on the path to the root

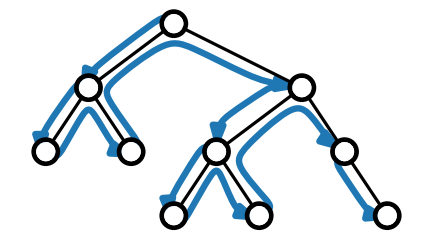
Depth: length of the path to the root

Height: maximum depth of a leaf

Binary Tree: at most two children per vertex (*left* and *right* child)

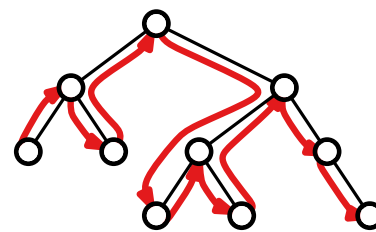
3 types of tree traversals:

preorder



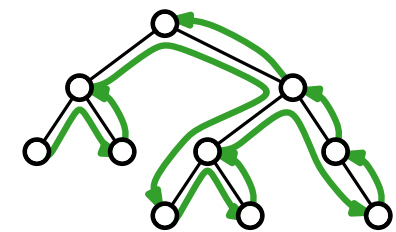
node – left – right

inorder



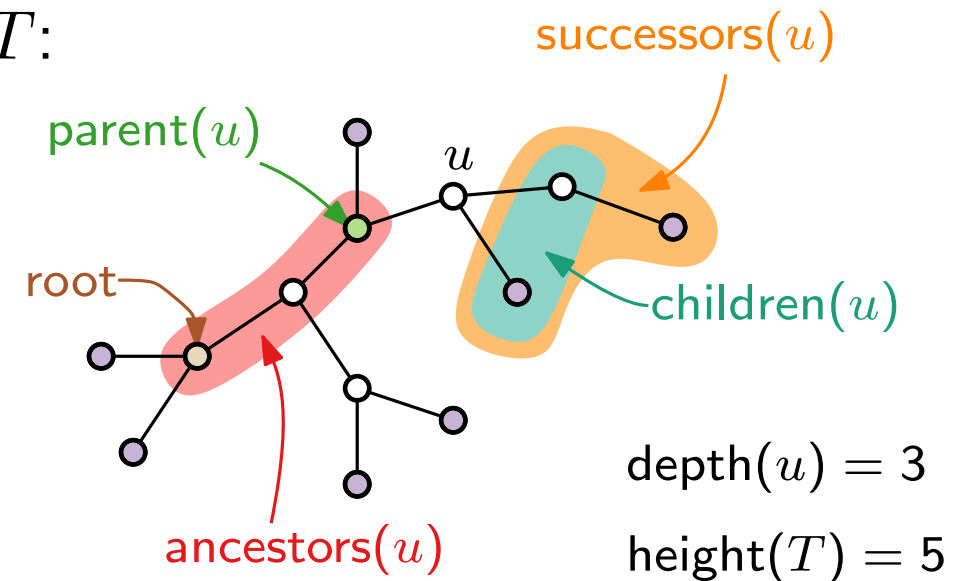
left – node – right

postorder



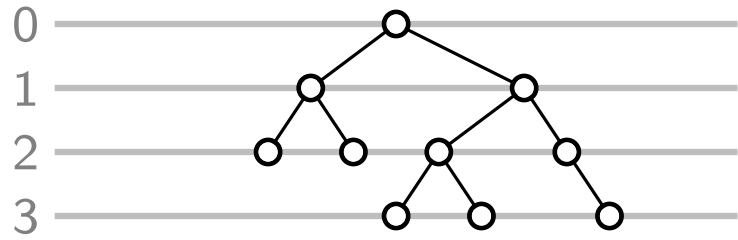
left – right – node

T :



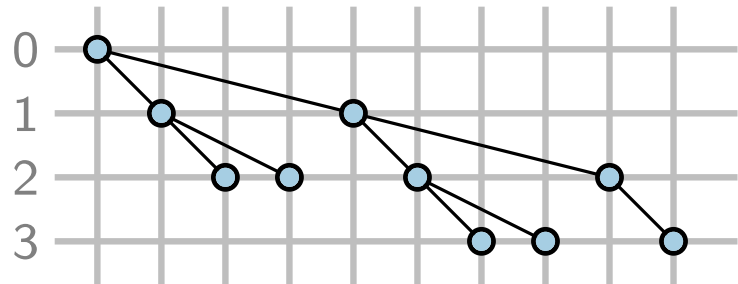
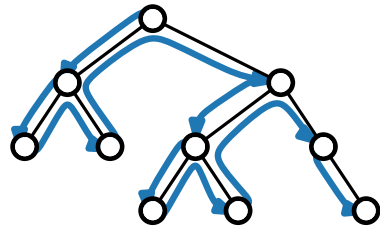
First Grid Layout of Binary Trees

1. Choose y -coordinates: $y(u) = \text{depth}(u)$

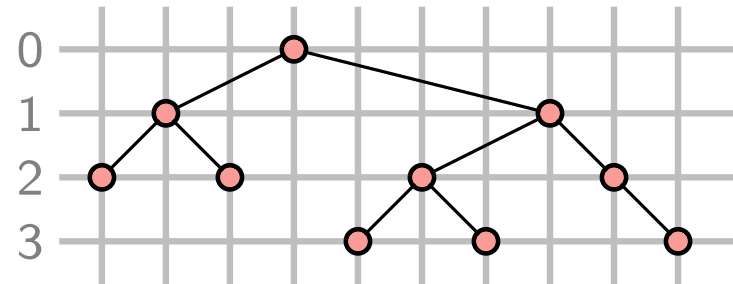
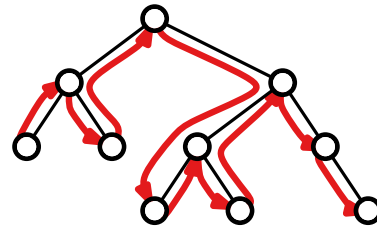


2. Choose x -coordinates:

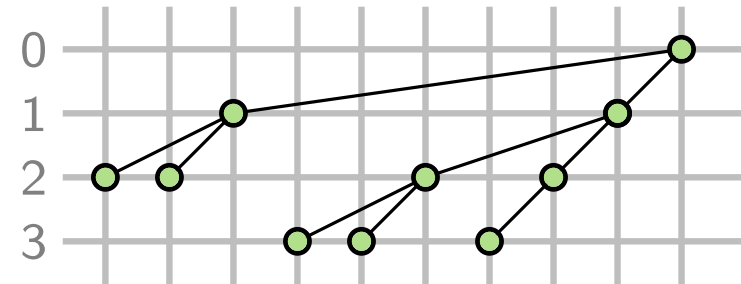
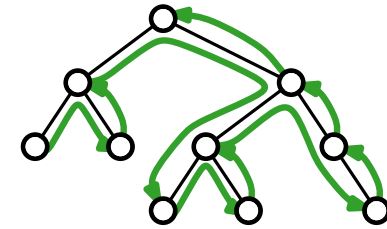
preorder



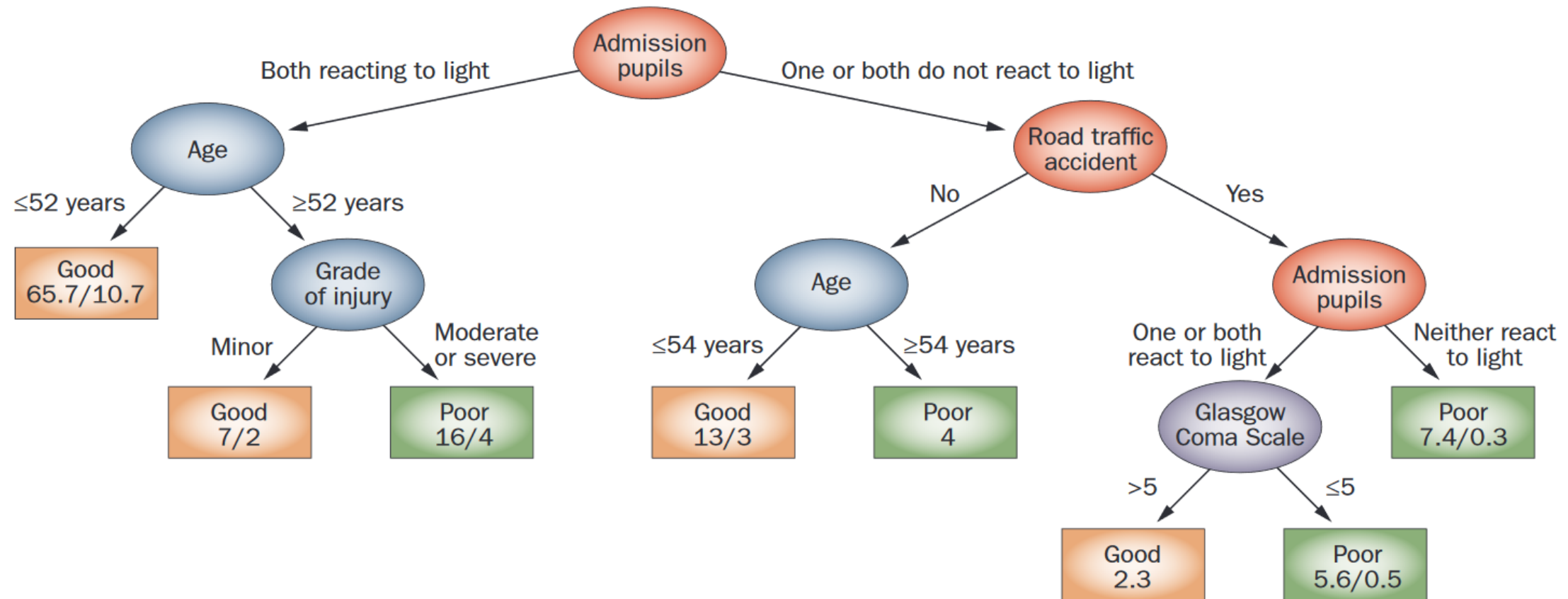
inorder



postorder



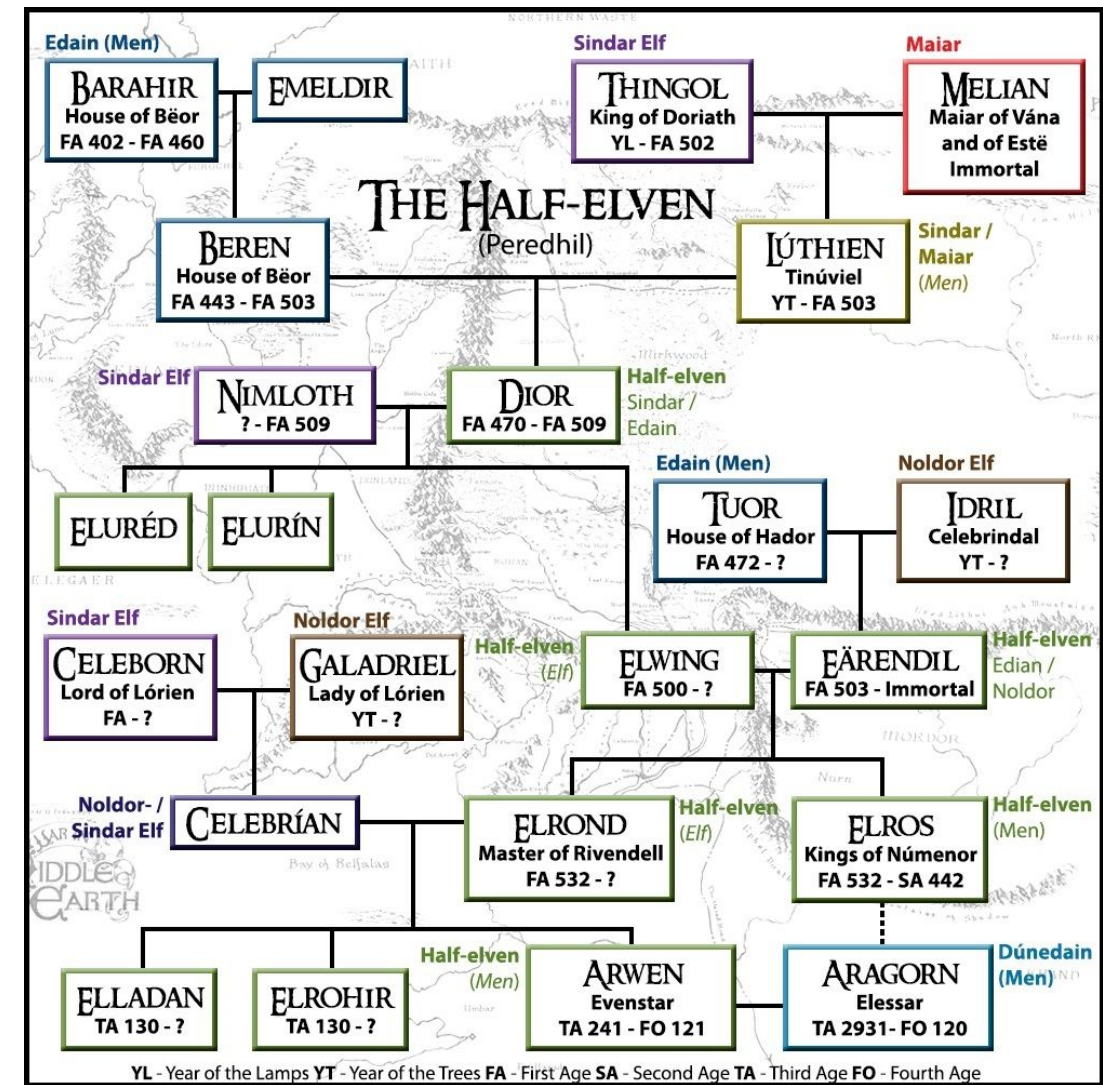
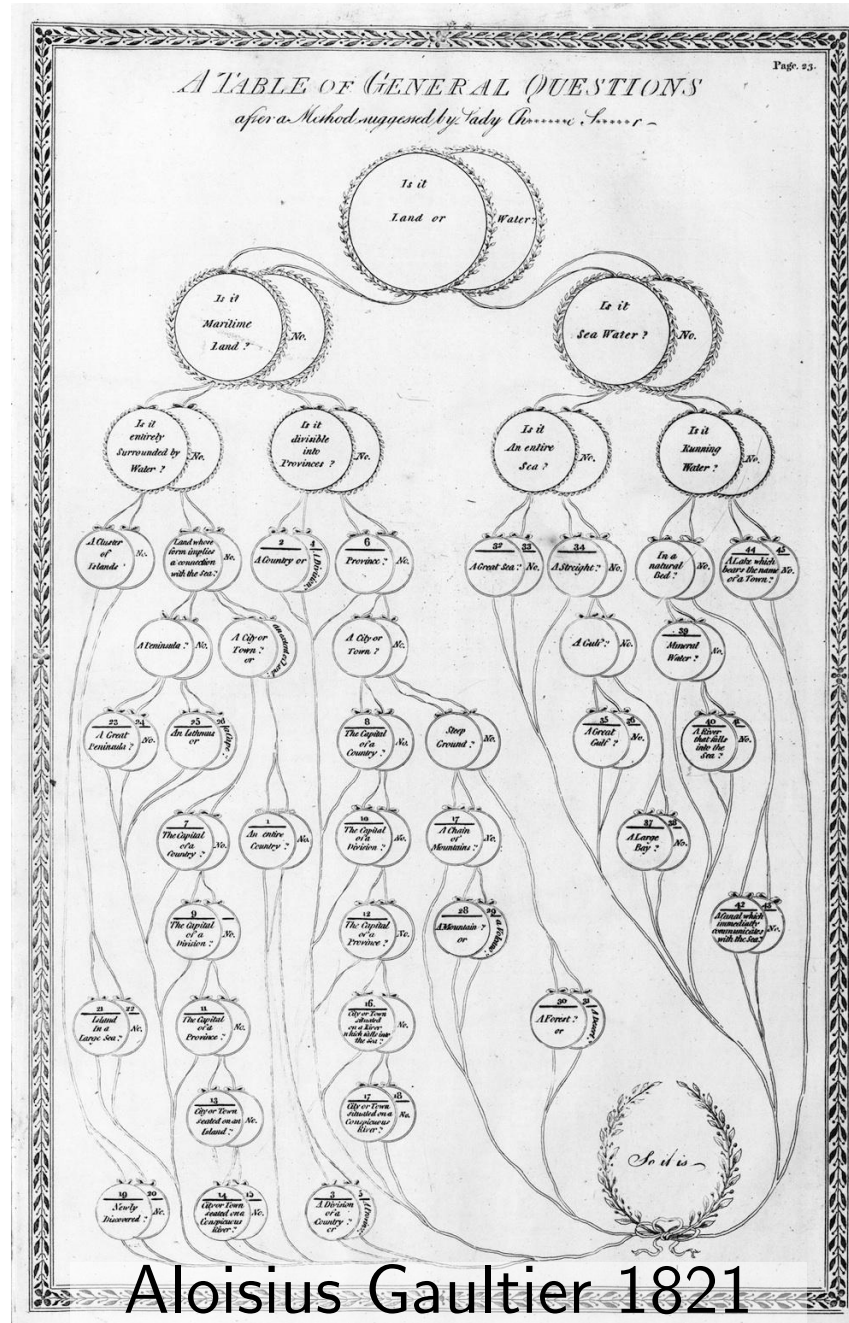
Layered Drawings – Applications



Decision tree for outcome prediction after traumatic brain injury

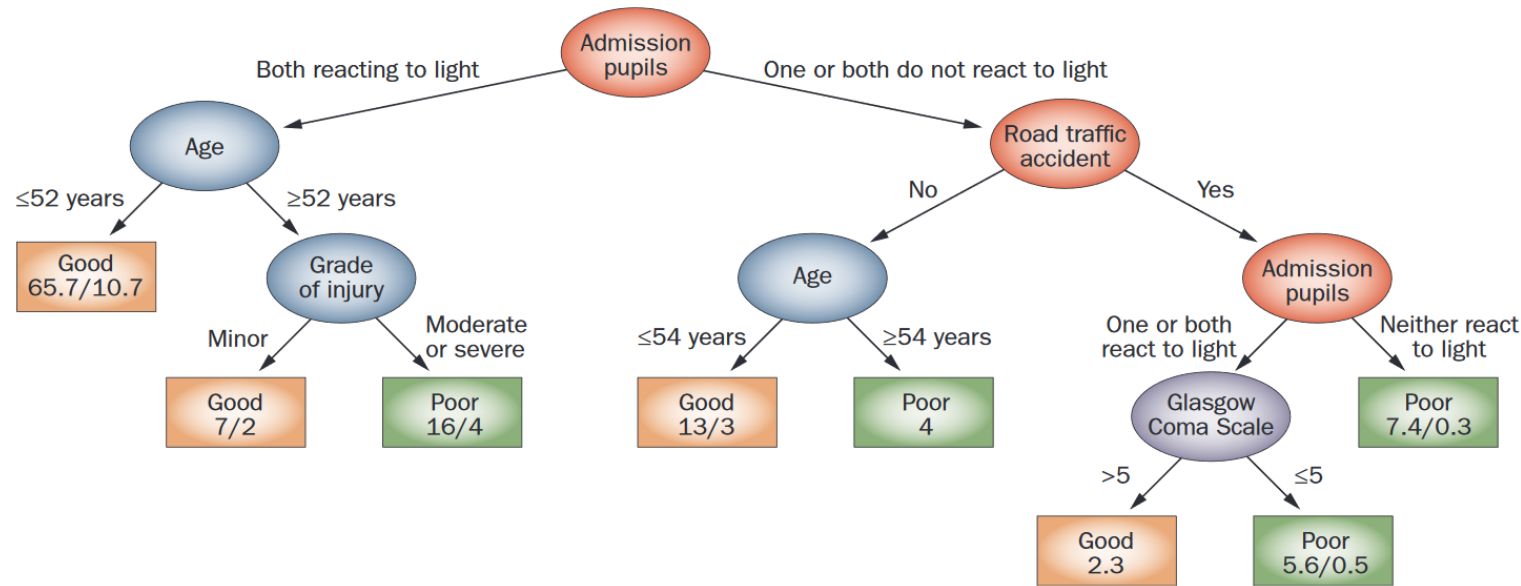
Source: Nature Reviews Neurology

Layered Drawings – Applications

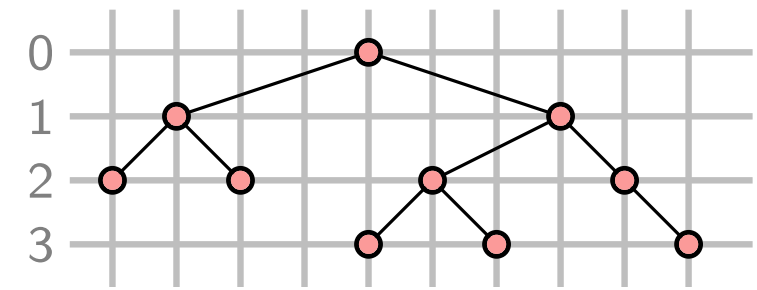
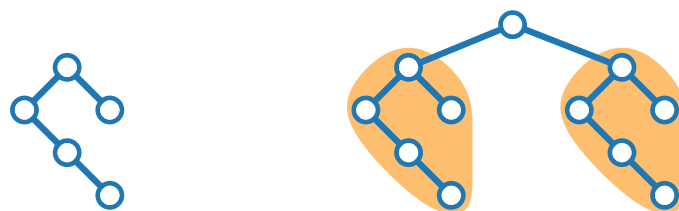


Family tree of LOTR elves and half-elves

Layered Drawings – Drawing Style



- What are properties of the layout?
- What are the drawing conventions?
- What are aesthetics to optimize?



Drawing conventions

- Vertices lie on layers and have integer coordinates
- Parent centered above children (if there is more than one child)
- Edges are straight-line segments
- Isomorphic subtrees have identical drawings

Drawing aesthetics to optimize

- Area
- Symmetries

Layered Drawings – Algorithm

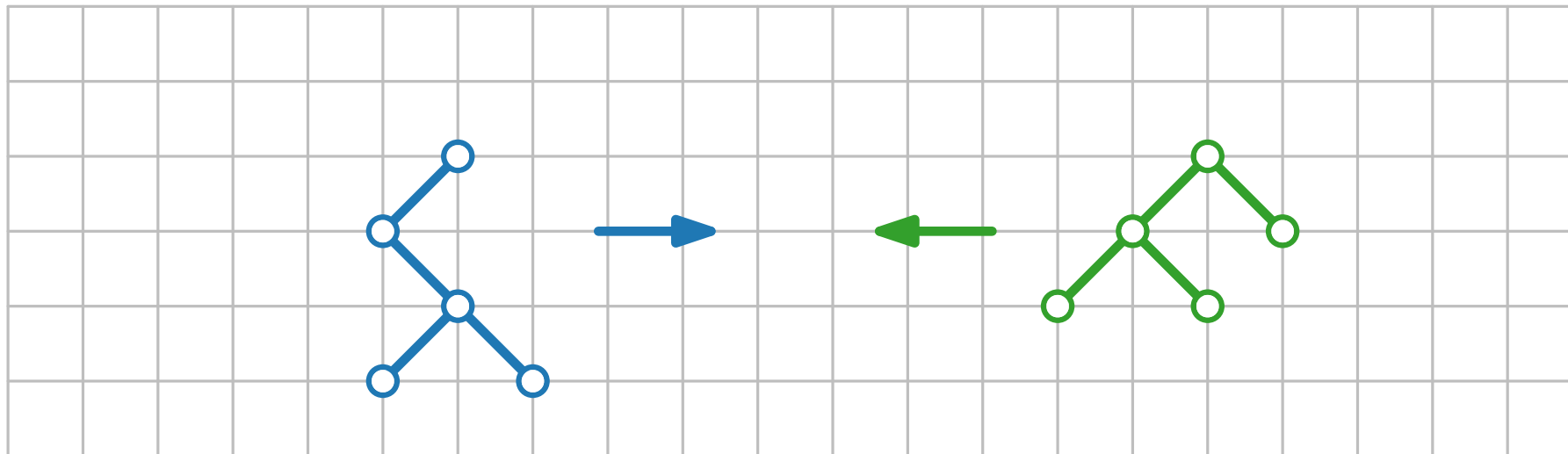
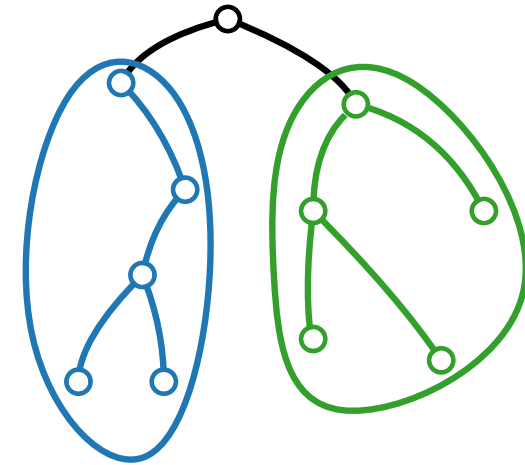
Input: A binary tree T

Output: A layered drawing of T

Base case: A single vertex 

Divide: Recursively apply the algorithm to draw the left and right subtrees

Conquer:



Layered Drawings – Algorithm

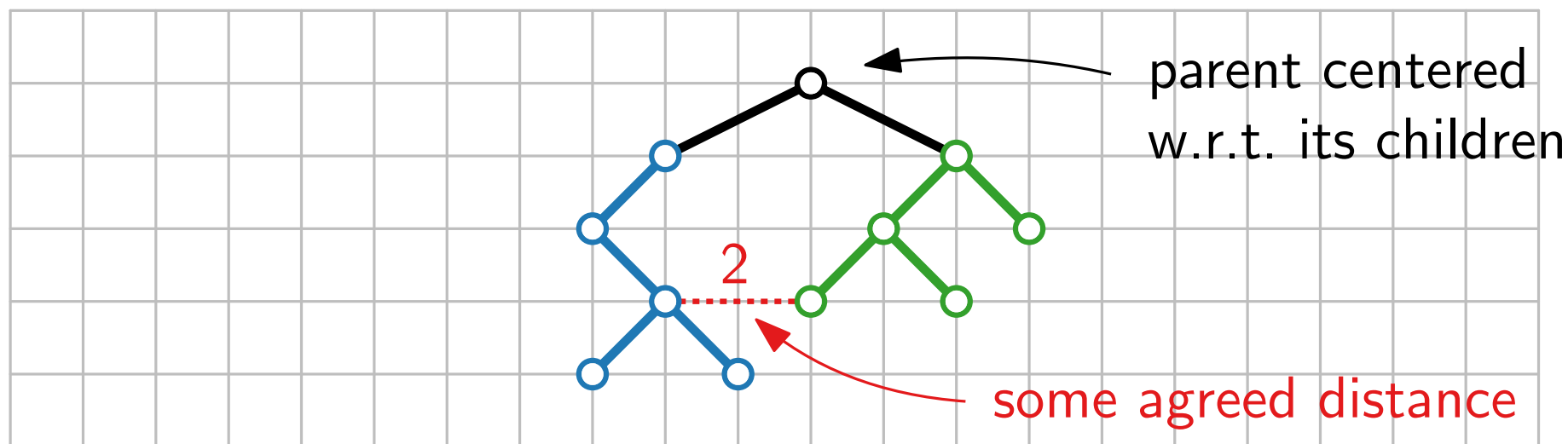
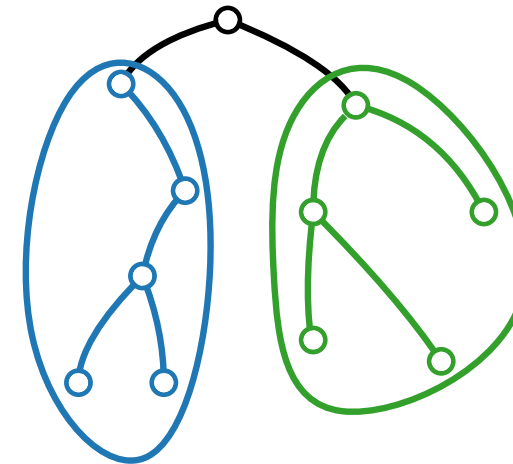
Input: A binary tree T

Output: A layered drawing of T

Base case: A single vertex 

Divide: Recursively apply the algorithm to draw the left and right subtrees

Conquer:



sometimes **3** apart for grid drawing!

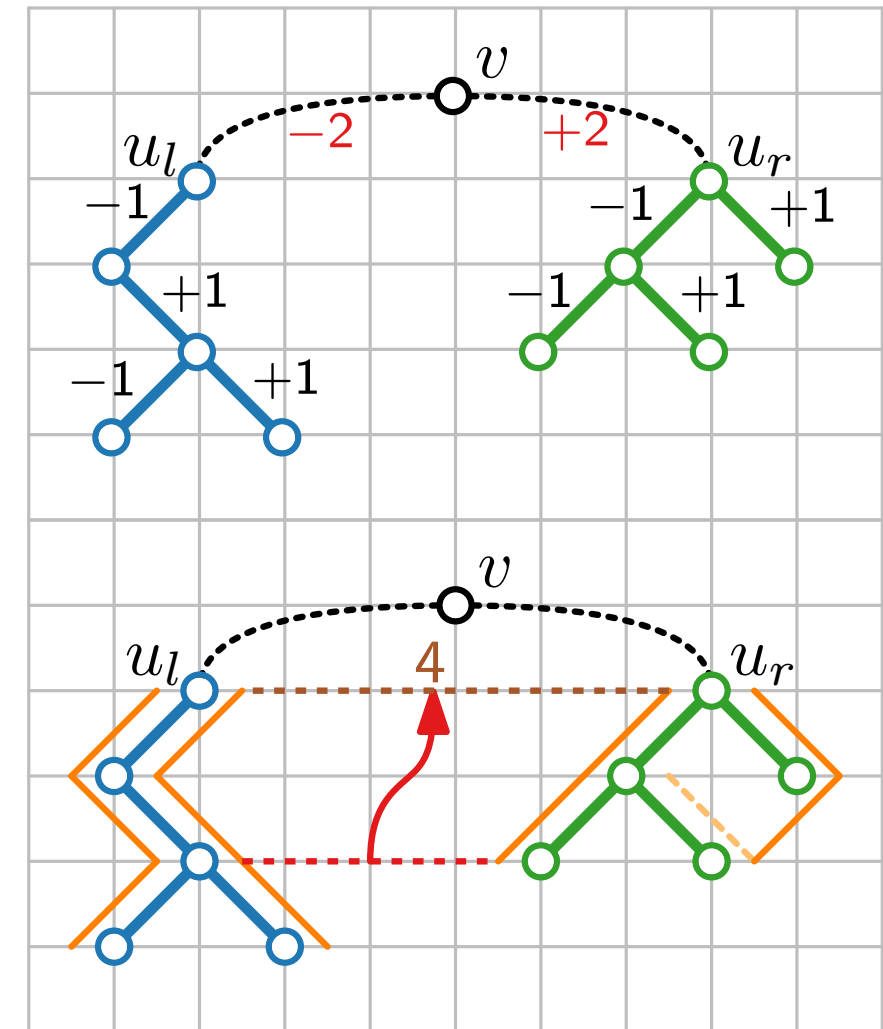
Layered Drawings – Algorithm Details

Phase 1 – postorder traversal:

- For each vertex compute horizontal displacement of left and right child
- $\text{x-offset}(u_l) = -\lceil \frac{d_v}{2} \rceil$, $\text{x-offset}(u_r) = \lceil \frac{d_v}{2} \rceil$
- At vertex u (below v) store left and right **contour** of subtree $T(u)$
- Contour is linked list of vertex coordinates/offsets
- Find $d_v = \text{min. horiz. distance between } v_l \text{ and } v_r$

Phase 2 – preorder traversal:

- Compute x- and y-coordinates



Layered Drawings – Algorithm Details

Phase 1 – postorder traversal:

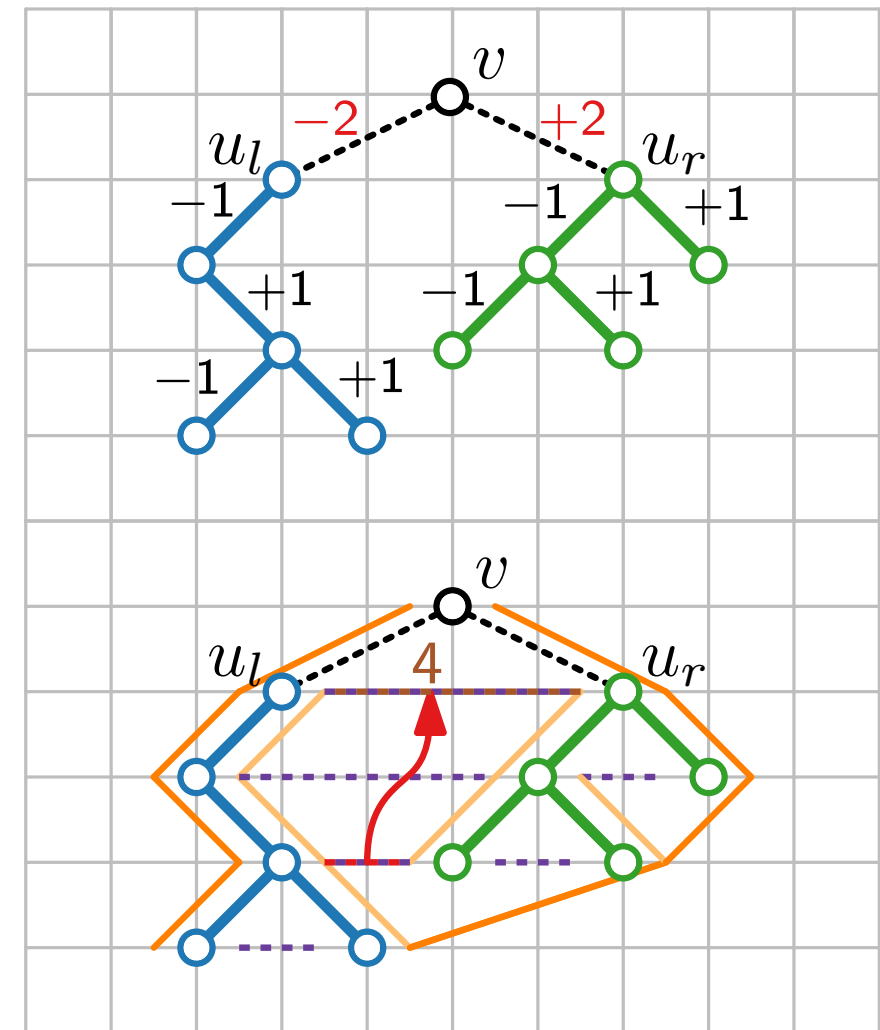
- For each vertex compute horizontal displacement of left and right child
- $\text{x-offset}(u_l) = -\lceil \frac{d_v}{2} \rceil$, $\text{x-offset}(u_r) = \lceil \frac{d_v}{2} \rceil$
- At vertex u (below v) store left and right **contour** of subtree $T(u)$
- Contour is linked list of vertex coordinates/offsets
- Find $d_v = \text{min. horiz. distance between } v_l \text{ and } v_r$

Phase 2 – preorder traversal:

- Compute x- and y-coordinates

Runtime?

- How often do we have to **walk along a contour**?



– Less than $n = \# \text{ vertices times!}$

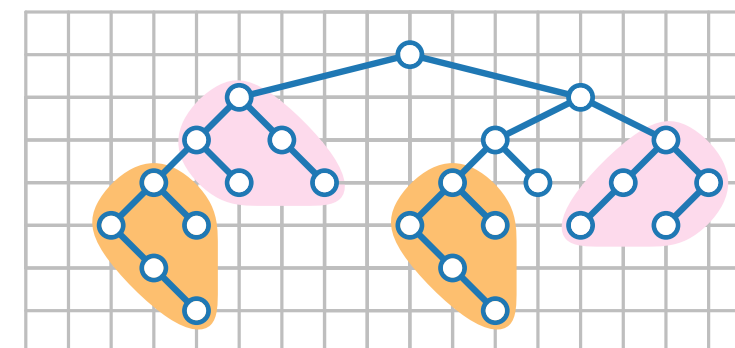
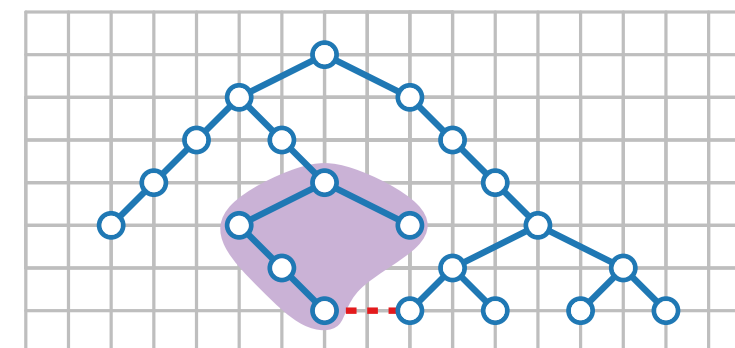
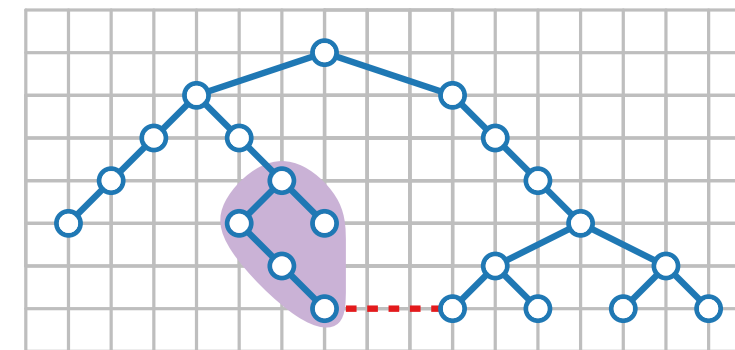
Layered Drawings – Result

Theorem.

[Reingold & Tilford '81]

Let T be a binary tree with n vertices. We can construct a drawing Γ of T in $\mathcal{O}(n)$ time such that:

- Γ is planar, straight-line and strictly downward
- Γ is layered: y-coordinate of vertex v is $-\text{depth}(v)$
- Horizontal and vertical distances are at least 1
- Each vertex with > 1 child is centered w.r.t. its children
- Area of Γ is in $\mathcal{O}(n^2)$ – but not optimal! ← NP-hard
- Simply isomorphic subtrees have congruent drawings, up to translation
- Axially isomorphic subtrees have congruent drawings, up to translation and reflection

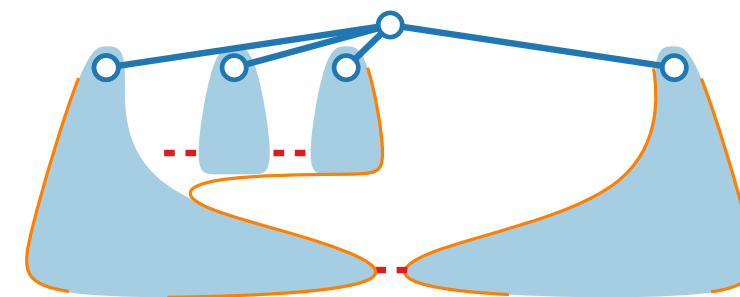
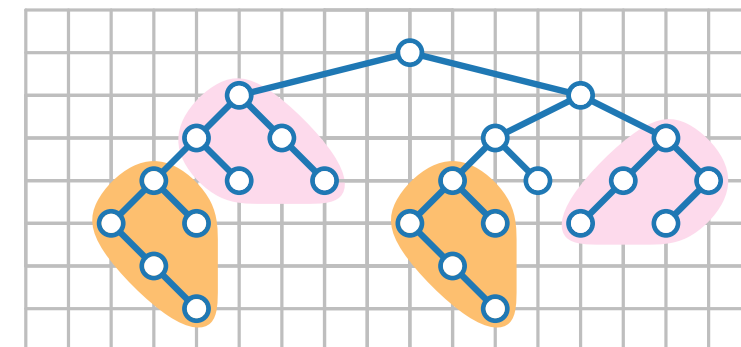
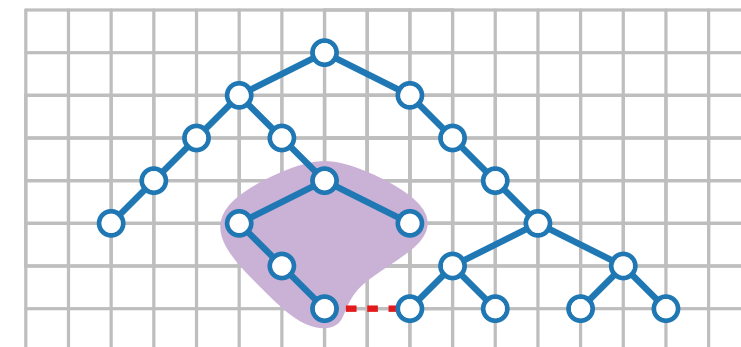
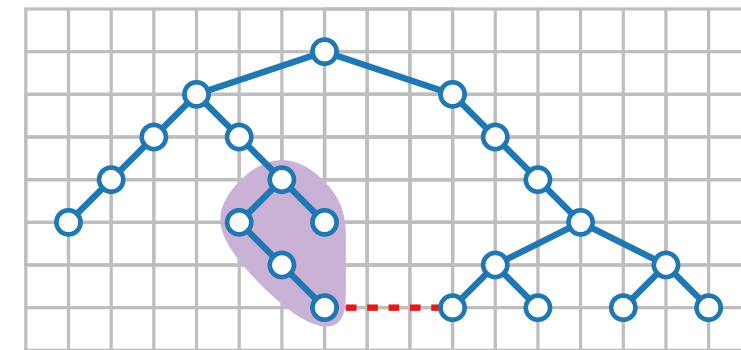


Layered Drawings – Result

Theorem. [Reingold & Tilford '81]

Let T be a ~~binary~~ ^{rooted} tree with n vertices. We can construct a drawing Γ of T in $\mathcal{O}(n)$ time such that:

- Γ is planar, straight-line and strictly downward
- Γ is layered: y-coordinate of vertex v is $-\text{depth}(v)$
- Horizontal and vertical distances are at least 1
- Each vertex with > 1 child is centered w.r.t. its children
- Area of Γ is in $\mathcal{O}(n^2)$ – but not optimal! ← NP-hard
- Simply isomorphic subtrees have congruent drawings, up to translation
- ~~■ Axially isomorphic subtrees have congruent drawings, up to translation and reflection~~

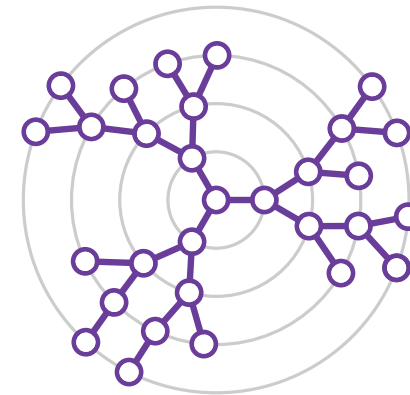
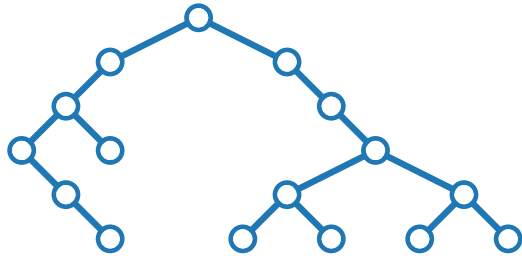
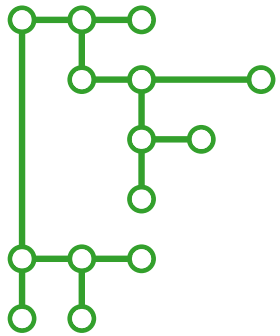


extension to non-binary rooted trees

Visualization of Graphs

Lecture 1b: Drawing Trees

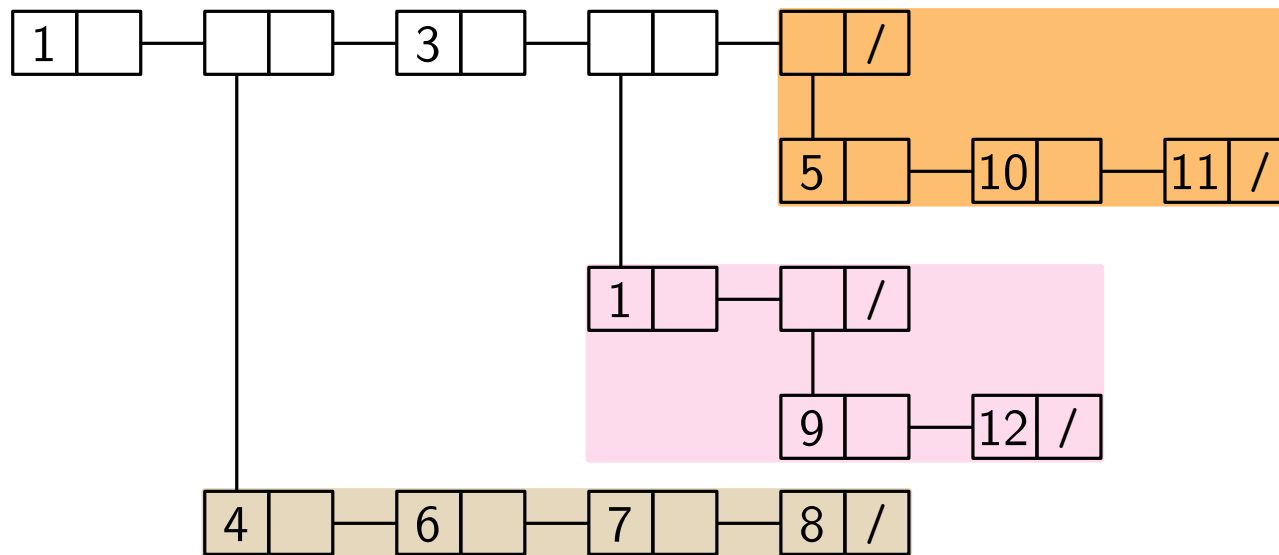
Part II: HV-Drawings



HV-Drawings – Drawing Style

Applications

- Cons cell diagram in LISP
- *Cons* (constructs) are memory objects that hold two values or pointers to values



Source: after gajon.org/trees-linked-lists-common-lisp/

Drawing conventions

- Children are vertically or horizontally aligned with their parent
- The bounding boxes of the subtrees of the children are disjoint
- Edges are strictly down- or rightwards

Drawing aesthetics to optimize

- Height, width, area

HV-Drawings – Algorithm

Input: A binary tree T

Output: An HV-drawing of T

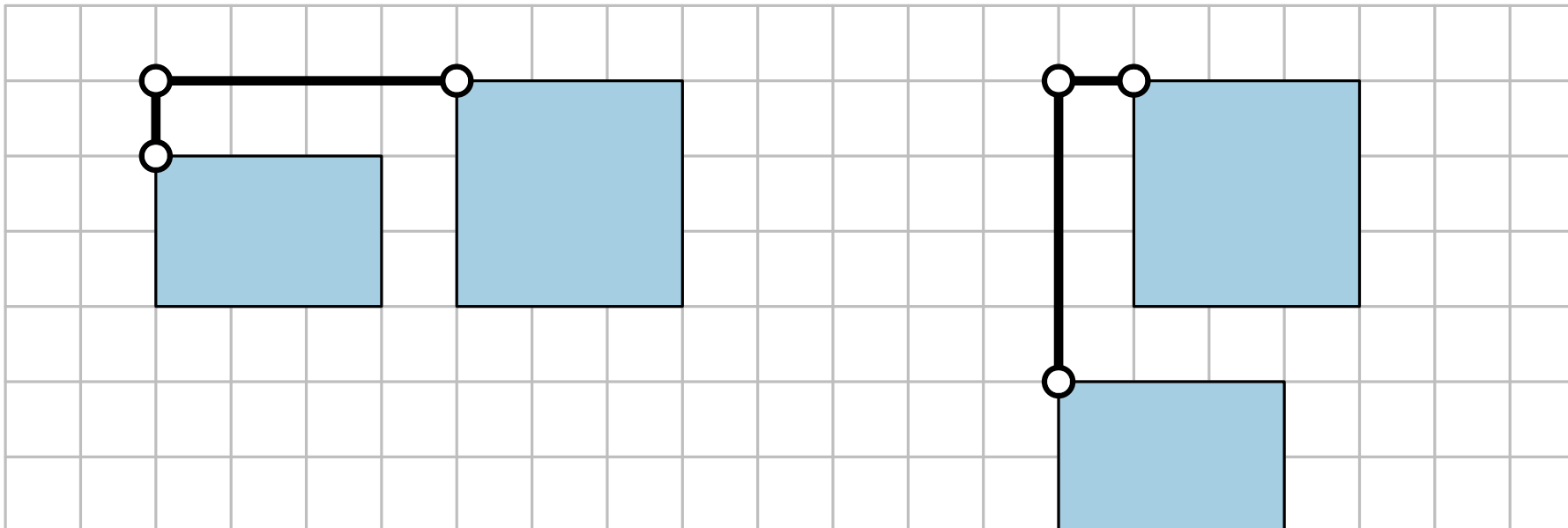
Base case: 

Divide: Recursively apply the algorithm to draw the left and right subtrees

Conquer:

horizontal combination

vertical combination



HV-Drawings – Right-Heavy HV-Layout

Right-heavy approach

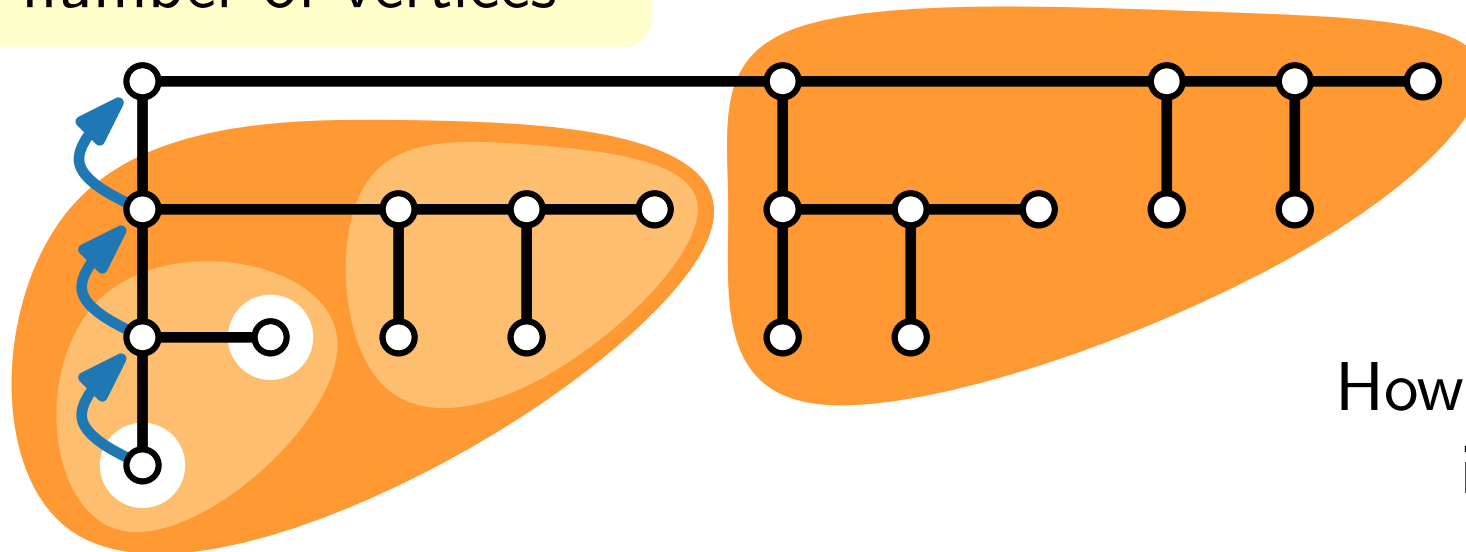
- Always apply horizontal combination
- Place the larger subtree to the right
Size of subtree := number of vertices

← *This can change the embedding!*

at least $\cdot 2$

at least $\cdot 2$

at least $\cdot 2$



How to implement this
in **linear time**?


Lemma. Let T be a binary tree. The drawing constructed by the right-heavy approach has

- width at most $n - 1$ and
- height at most $\log n$.

HV-Drawings – Result

Theorem.

Let T be a binary tree with n vertices. The right-heavy algorithm constructs in $O(n)$ time a drawing Γ of T s.t.:

- Γ is an HV-drawing
(planar, orthogonal, strictly right-/downward)
- Width is at most $n - 1$
- Height is at most $\log n$
- Area is in $\mathcal{O}(n \log n)$  worst-case optimal [exercise]
- Simply and axially isomorphic subtrees have congruent drawings up to translation

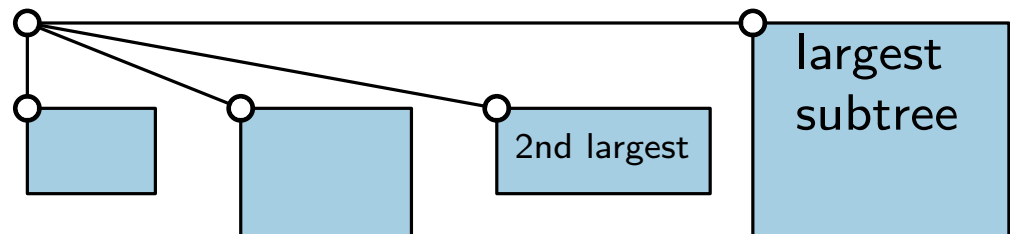
HV-Drawings – Result

Theorem. ~~binary~~ ^{rooted}

Let T be a ~~binary~~ tree with n vertices. The right-heavy algorithm constructs in $O(n)$ time a drawing Γ of T s.t.:

- Γ is an HV-drawing
(planar, ~~orthogonal~~, strictly right-/downward)
- Width is at most $n - 1$
- Height is at most $\log n$
- Area is in $\mathcal{O}(n \log n)$ ← worst-case optimal [exercise]
- Simply and axially isomorphic subtrees have congruent drawings up to translation

General rooted tree



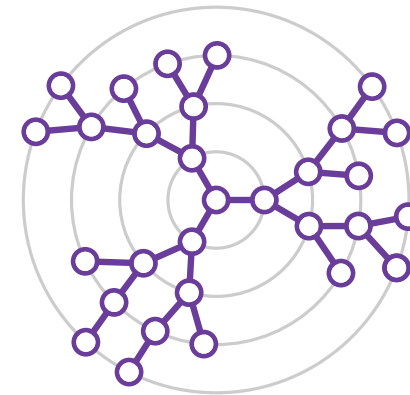
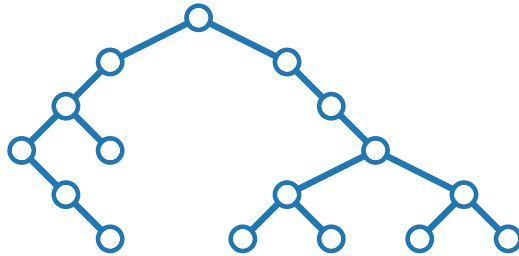
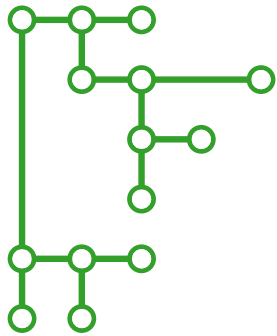
Optimal area?

Not with divide & conquer approach, but can be computed with Dynamic Programming.

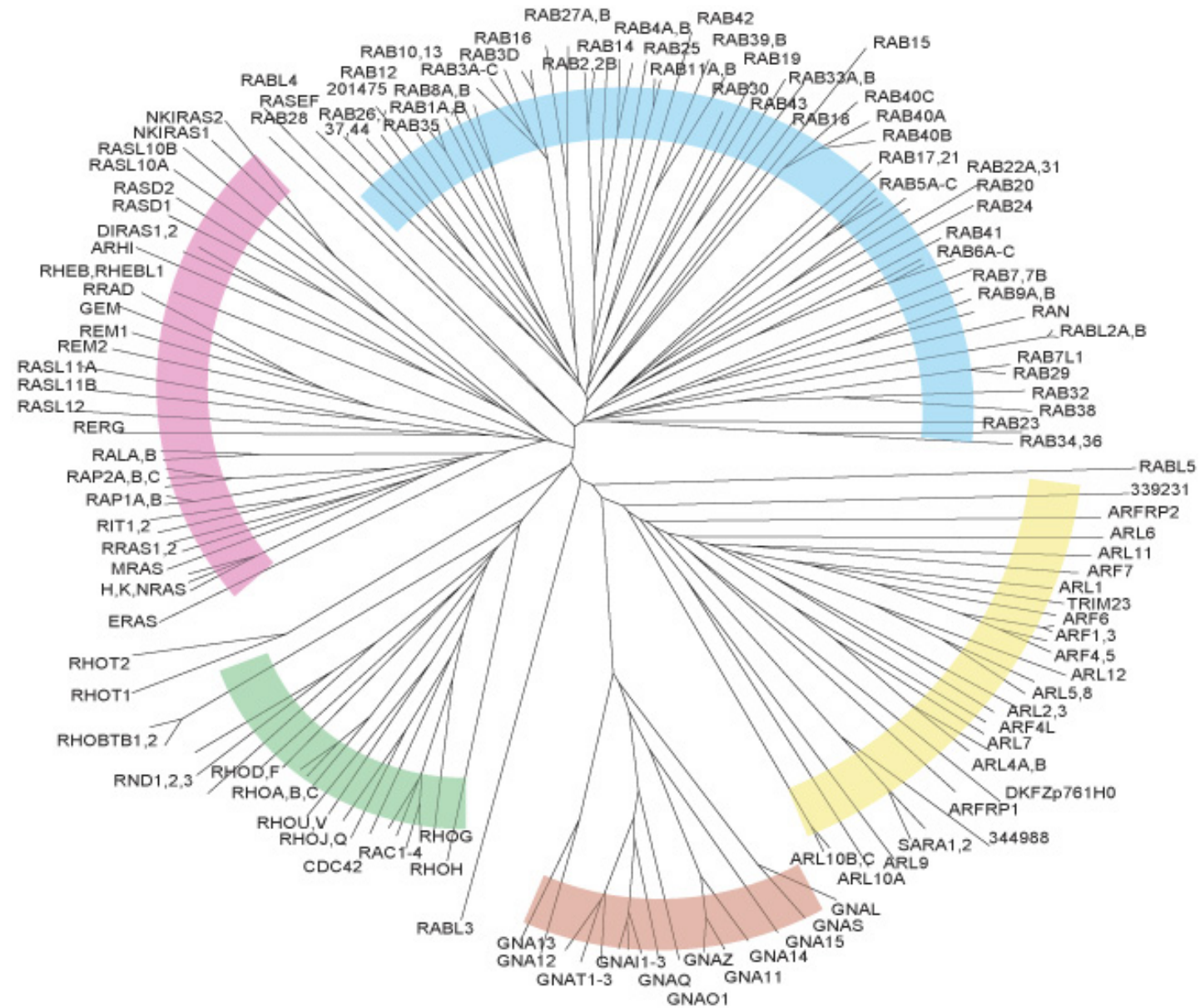
Visualization of Graphs

Lecture 1b: Drawing Trees

Part III: Radial Layouts

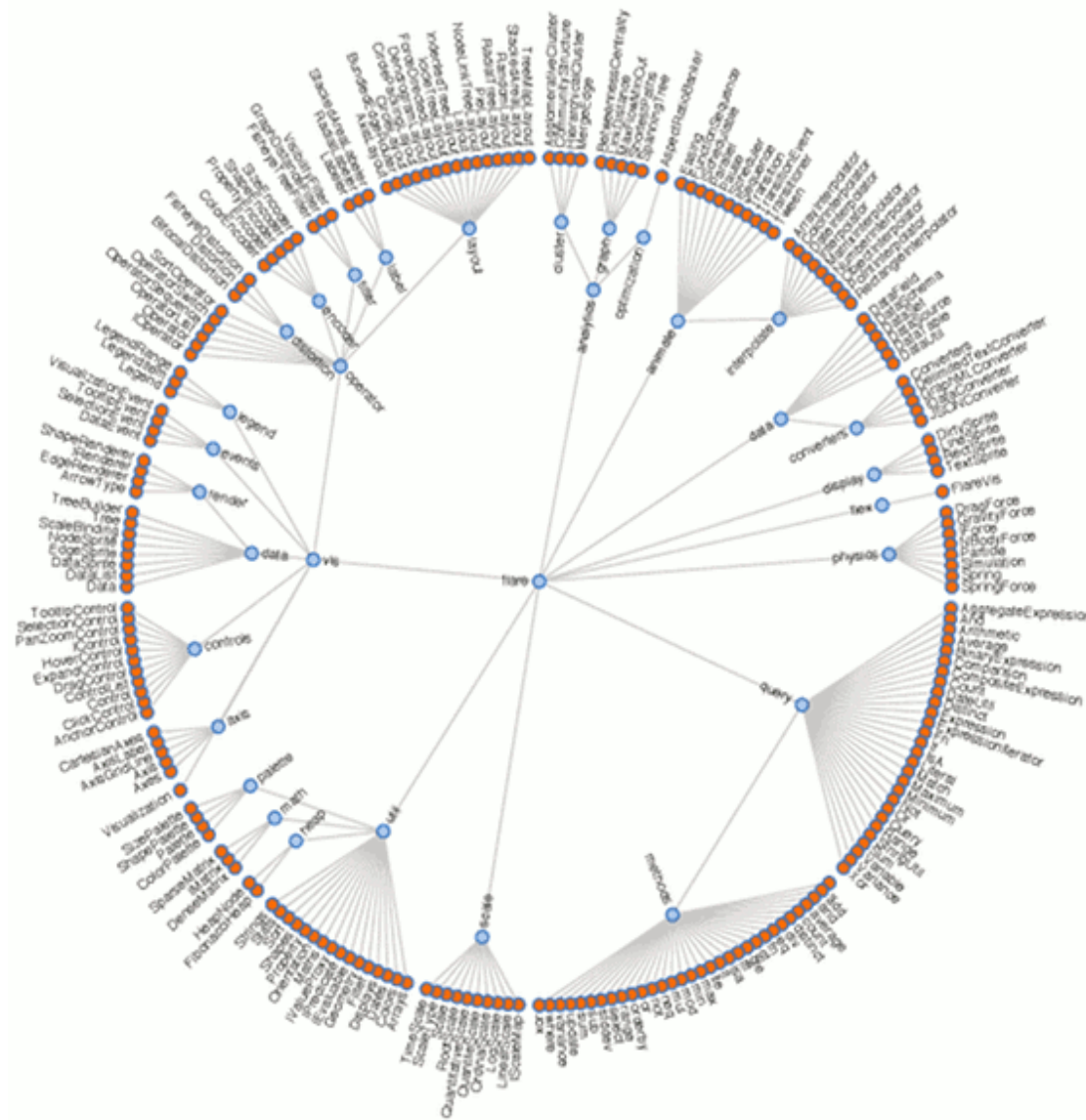


Radial Layouts – Applications

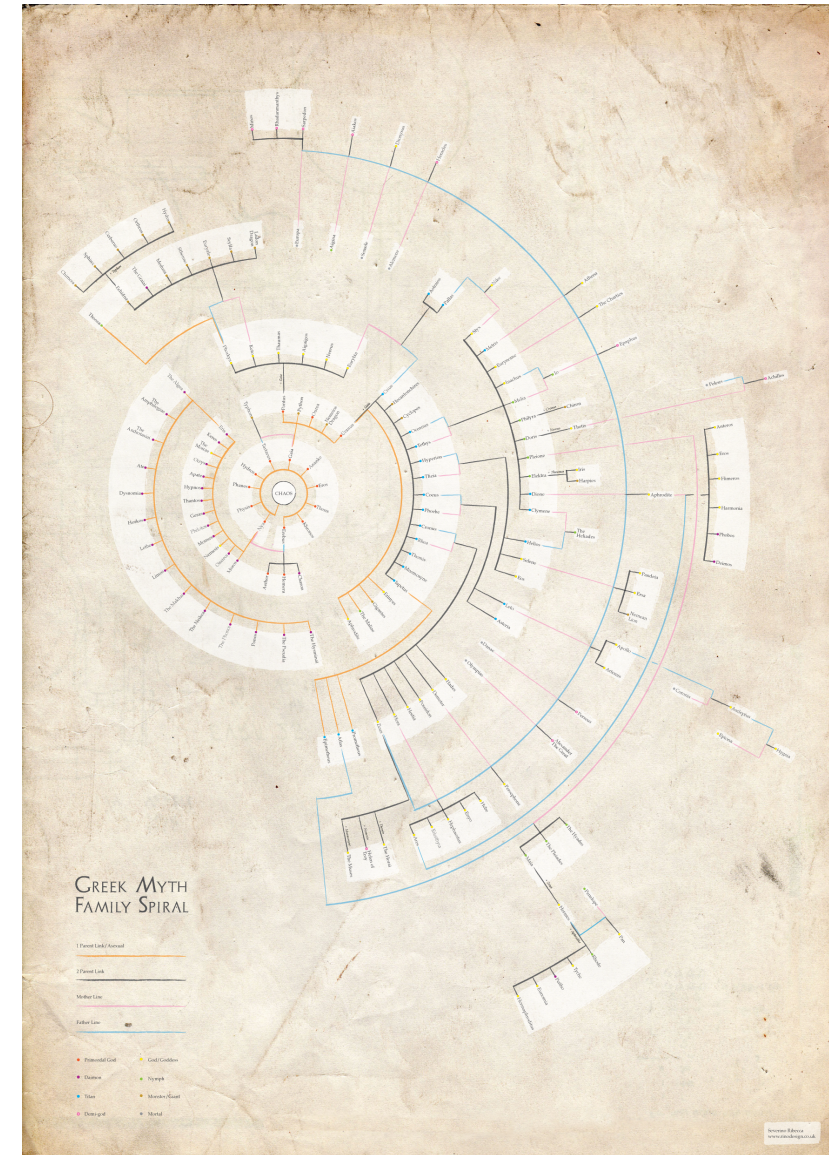


Phylogenetic tree
by Colicelli, ScienceSignaling, 2004

Radial Layouts – Applications

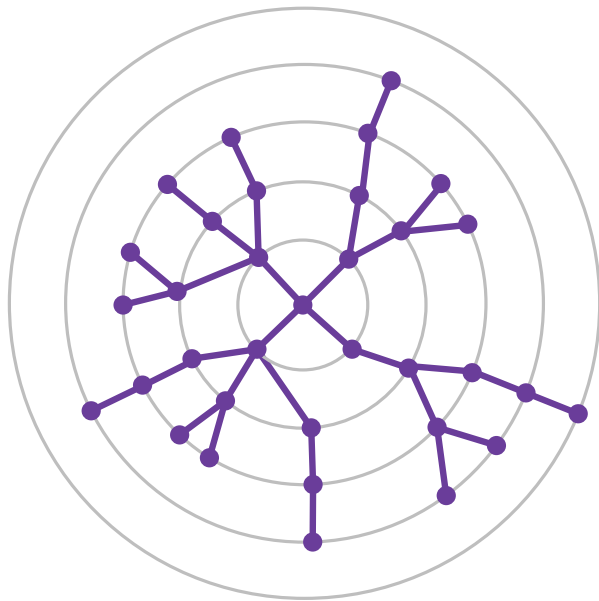


Flare Visualization Toolkit code structure by Heer, Bostock and Ogievetsky, 2010



Greek Myth Family
by Ribecca, 2011

Radial Layouts – Drawing Style



Drawing conventions

- Vertices lie on circular layers according to their depth
- Drawing is planar

Drawing aesthetics to optimize

- Balanced distribution of the vertices

How can an algorithm optimize the distribution of the vertices?

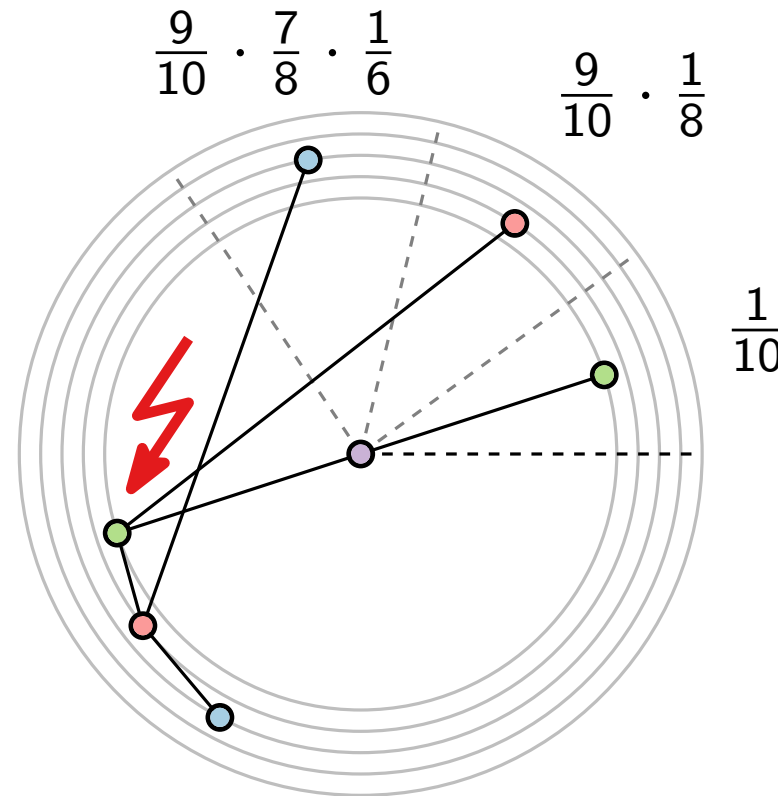
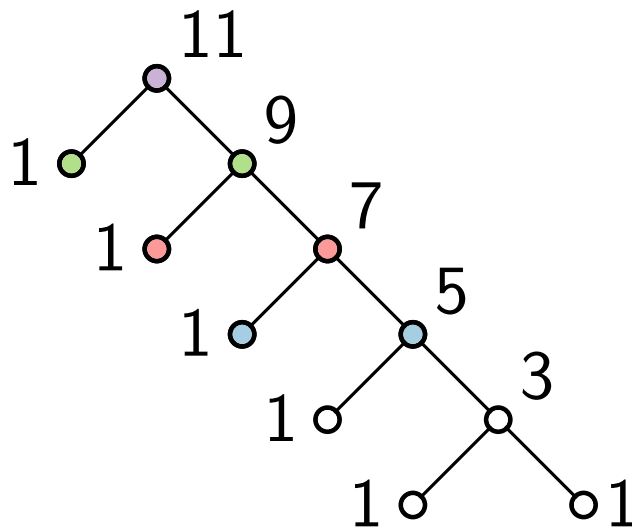
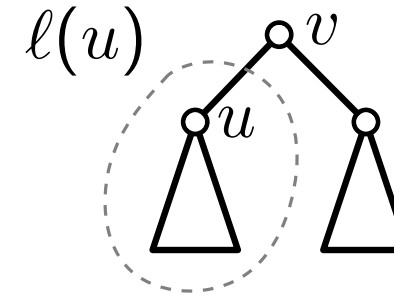
Radial Layouts – Algorithm Attempt

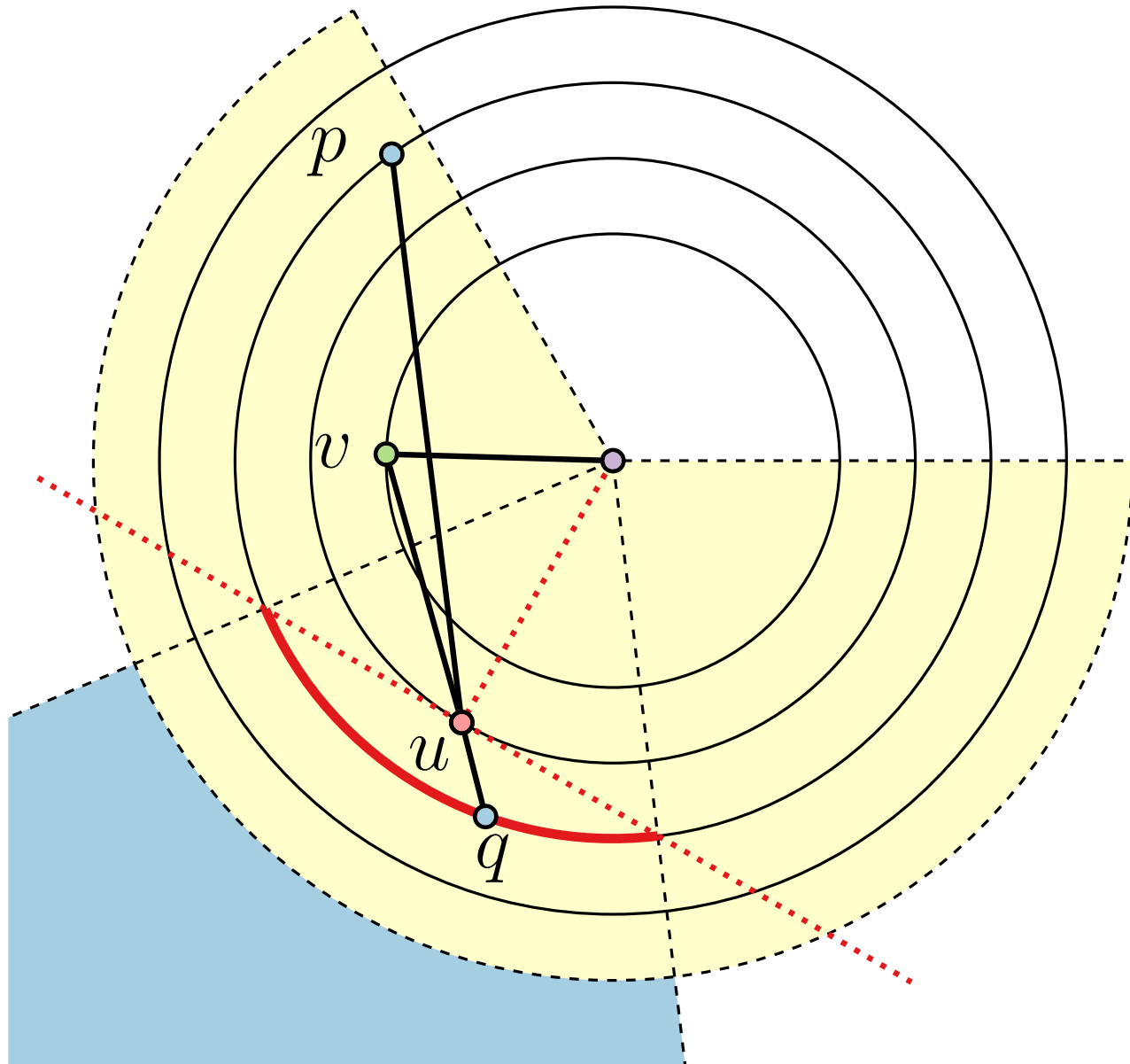
Idea

- Reserve area corresponding to size $\ell(u)$ of $T(u)$:

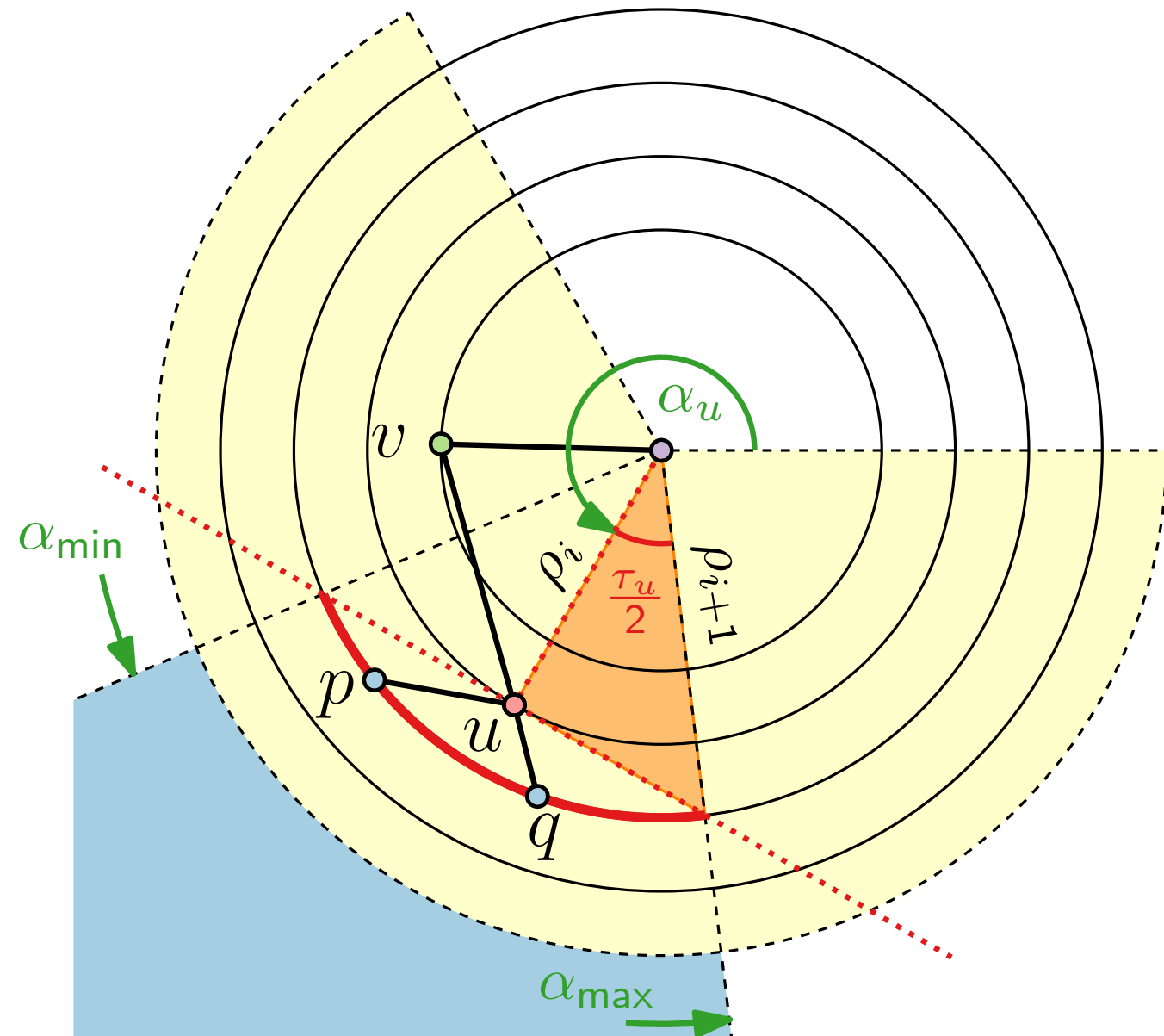
$$\tau_u = \frac{\ell(u)}{\ell(v) - 1}$$

- Place u in the middle of its area





Radial Layouts – How To Avoid Crossings



- τ_u – angle of the wedge corresponding to vertex u
- $\ell(u)$ – number of nodes in the subtree rooted at u
- ρ_i – radius of layer i
- $\cos \frac{\tau_u}{2} = \frac{\rho_i}{\rho_{i+1}}$
- $\tau_u = \min \left\{ \frac{\ell(u)}{\ell(v)-1} \cdot \tau_v, 2 \arccos \frac{\rho_i}{\rho_{i+1}} \right\}$
- Alternative:
 - $\alpha_{\min} = \alpha_u - \arccos \frac{\rho_i}{\rho_{i+1}}$
 - $\alpha_{\max} = \alpha_u + \arccos \frac{\rho_i}{\rho_{i+1}}$

Radial Layouts – Pseudocode

RadialTreeLayout(tree T , root $r \in T$, radii $\rho_1 < \dots < \rho_k$)

begin

 postorder(r)

 preorder(r , 0, 0, 2π)

return $(d_v, \alpha_v)_{v \in V(T)}$

 // vertex positions in polar coordinates

postorder(vertex v)

$\ell(v) \leftarrow 1$

foreach child w of v **do**

 postorder(w)

$\ell(v) \leftarrow \ell(v) + \ell(w)$

preorder(vertex v , t , α_{\min} , α_{\max})

$d_v \leftarrow \rho_t$ // output

$\alpha_v \leftarrow (\alpha_{\min} + \alpha_{\max})/2$

if $t > 0$ **then**

$\alpha_{\min} \leftarrow \max\{\alpha_{\min}, \alpha_v - \arccos \frac{\rho_t}{\rho_{t+1}}\}$

$\alpha_{\max} \leftarrow \min\{\alpha_{\max}, \alpha_v + \arccos \frac{\rho_t}{\rho_{t+1}}\}$

$left \leftarrow \alpha_{\min}$

foreach child w of v **do**

$right \leftarrow left + \frac{\ell(w)}{\ell(v)-1} \cdot (\alpha_{\max} - \alpha_{\min})$

 preorder(w , $t + 1$, $left$, $right$)

$left \leftarrow right$

Runtime? $\mathcal{O}(n)$

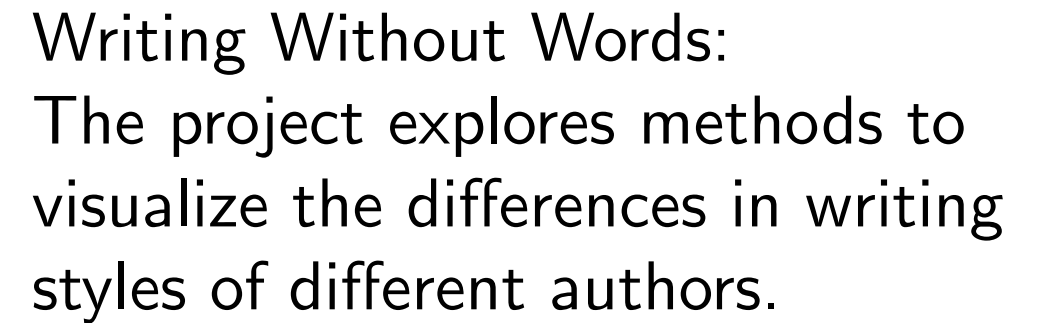
Correctness? ✓

Radial Layouts – Result

Theorem.

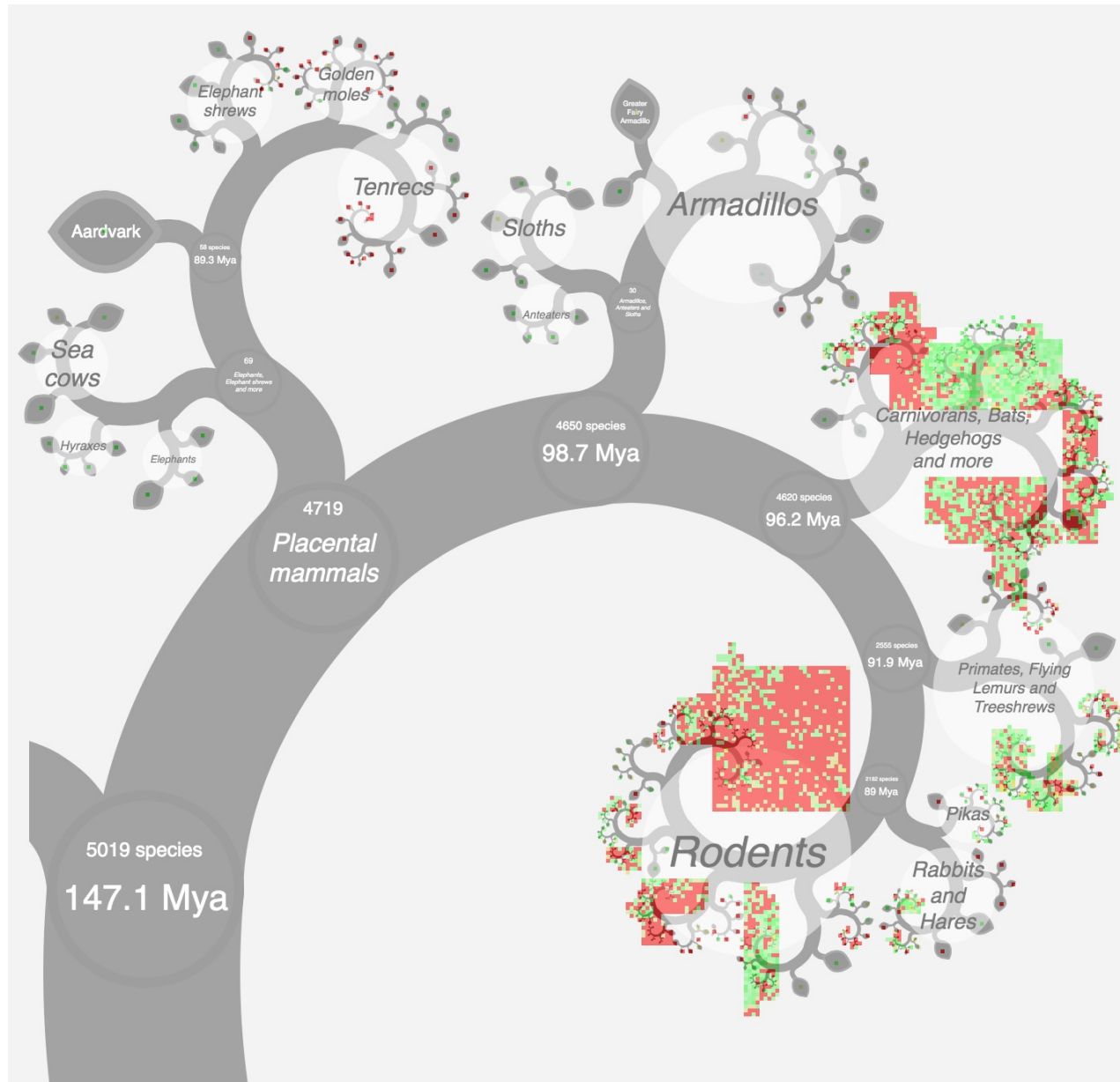
Let T be a tree with n vertices. The RadialTreeLayout algorithm constructs in $O(n)$ time a drawing Γ of T s.t.:

- Γ is radial drawing
- Vertices lie on circle according to their depth
- Area quadratic in $\max\text{-degree}(T) \times \text{height}(T)$
(see [GD Ch. 3.1.3] if interested)



Similar to balloon layout

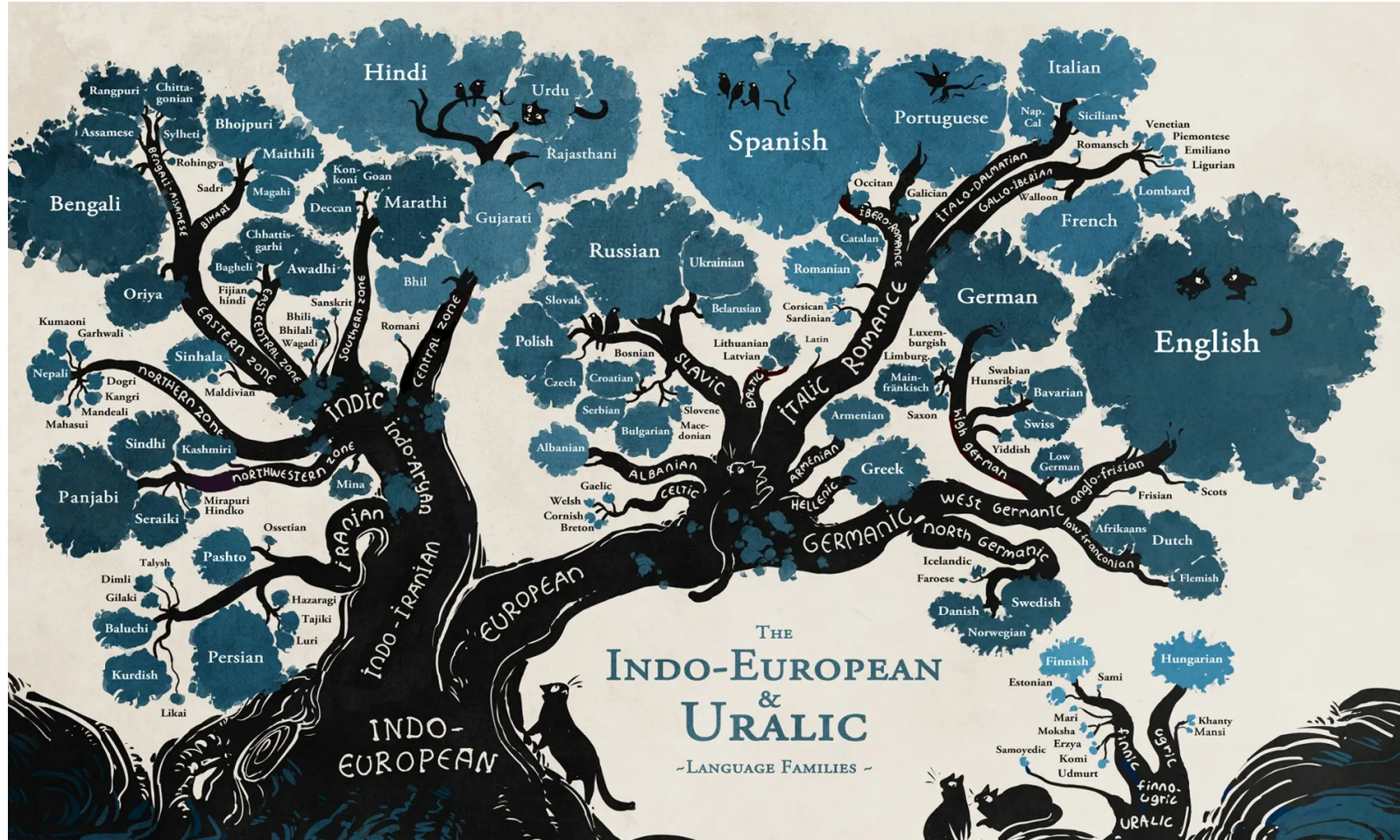
Other Tree Visualization Styles



A phylogenetically organized display of data for all placental mammal species.

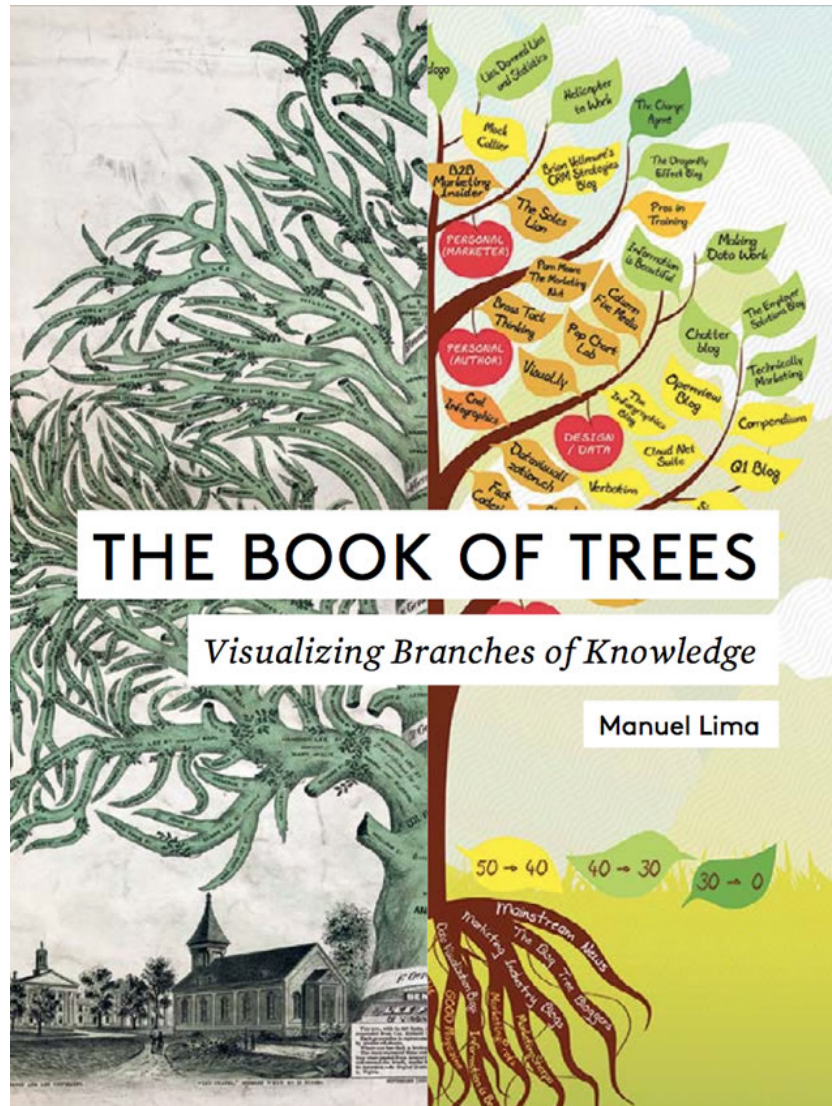
Fractal layout

Other Tree Visualization Styles



A language family tree – in pictures

Other Tree Visualization Styles



treevis.net

Literature

- [GD, Chapter 3] for divide and conquer methods for rooted trees and series-parallel graphs
- [Reingold, Tilford '81] “Tidier Drawings of Trees”
original paper for level-based layout algo
- [Reingold, Supowit '83] “The complexity of drawing trees nicely”
linear program and NP-hardness proof for area minimization
- `treevis.net` – compendium of drawing methods for trees