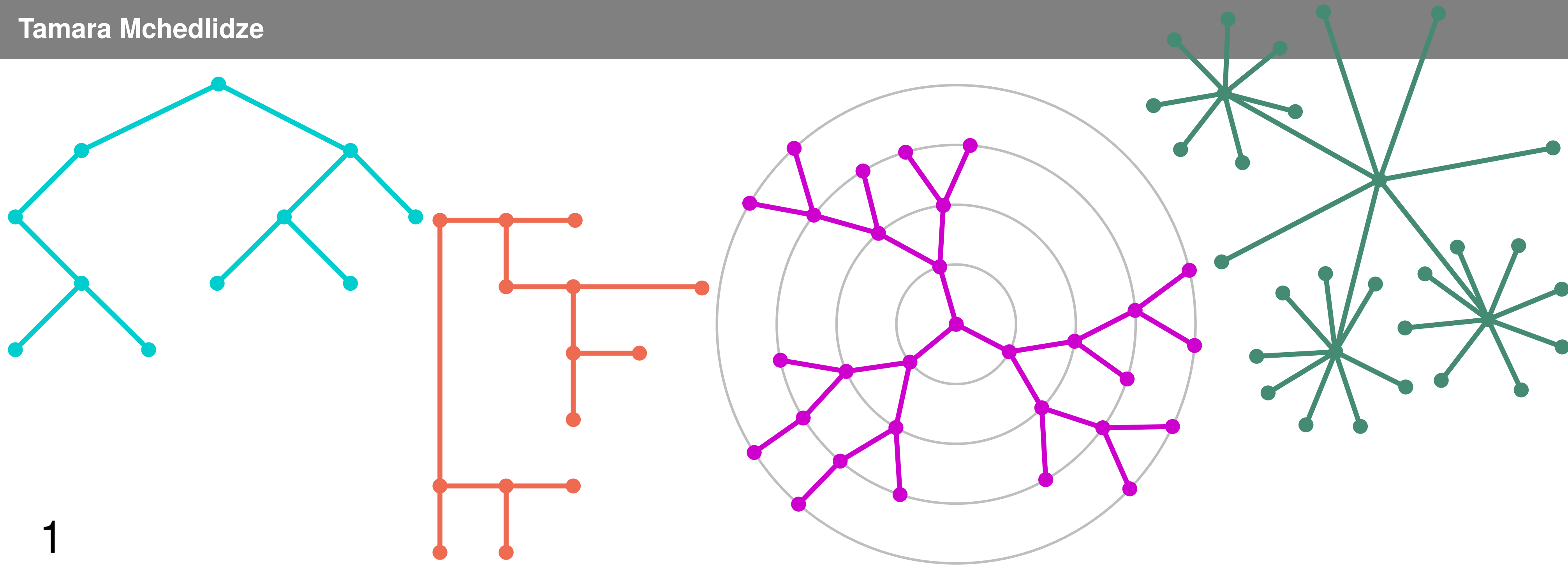
Karlsruhe Institute of Technology

Algorithms for graph visualization

Divide and Conquer - Tree Layouts - Part 2

WINTER SEMESTER 2017/2018

Tamara Mchedlidze



Feedback from the questionary

What have I learned in this lecture?

- Basic/simple/easy/cool algorithm for tree visualization in linear time
- Things that seem to be easy are very difficult
- Time complexity
- Separation into drawing conventions and aesthetics
- The algorithm for drawing a binary tree by **dynamic planning**?
- That a visual proof is legit
- Goals of the lecture (1!)

Feedback from the questionary

What was not clear?

- Time complexity
- Why boundary update is $O(1)$?
- Why called level based layout?
- Pseudocode?
- Running example
- Details in the data structure
- Boundaries?
- Example before the time complexity not clear

Feedback from the questionary

What was not clear?

- Time complexity
- Why boundary update is O(1)?
- Why called level based layout?
- Pseudocode?
- Running example
- Details in the data structure
- Boundaries?
- Example before the time complexity not clear

Can I work out the details myself?

- Yes
- Fundamentals missing

Feedback from the questionary

What would I like to learn about the tree layouts?

- Applications
- More layouts and more algorithms
- **Algorithm for non binary tree**
- Area/width optimization (K. J. Supowit, E. M. Reingold The complexity of drawing trees nicely [SR83])
- What kind of trees force the afterward optimization of the width?
- **Empirical findings**
- More efficient algorithm than $O(n)$?
- Is it easy to figure out which is the best parent node? (do you mean root node?) (best for what?)
- How the tree is constructed in the first place?

Overview

- H(horizontal) V(vertical) tree layout algorithm
- Radial tree layout algorithm
- Other visualization styles

Applications

Cons cell diagram in LISP.
Cons(constructs) are memory objects which hold two values or pointers to values.

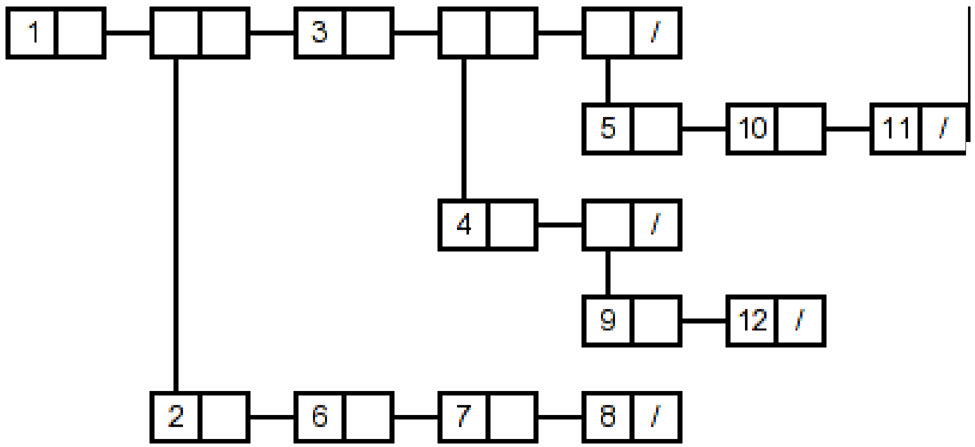


Figure 3: Diagram of cons cells of the simple tree. `http://gajon.org/`

Discuss with your neighbour(s) and then share

2+3 min

Applications

Cons cell diagram in LISP.
Cons(constructs) are memory objects which hold two values or pointers to values.

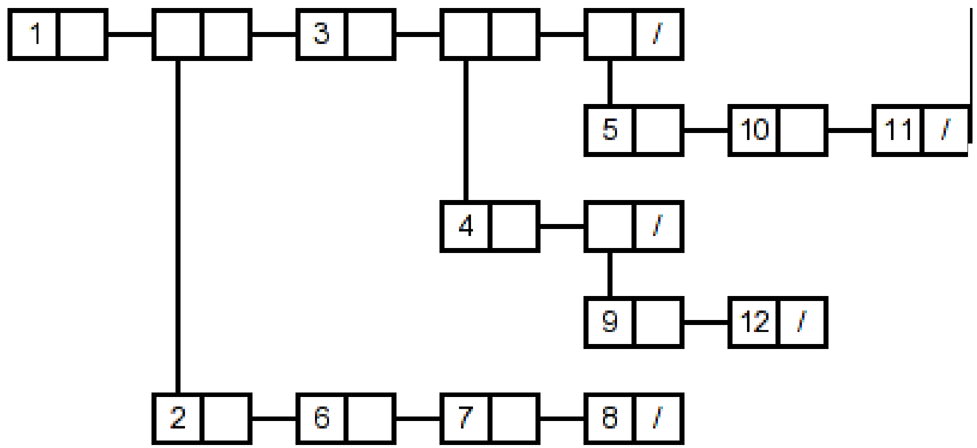


Figure 3: Diagram of cons cells of the simple tree. `http://gajon.org/`

4 - 2

Discussion with your neighbour(s) and then share

2+3 min

- What are the drawing conventions and aesthtetics?

HV-Layout

Drawing Conventions:

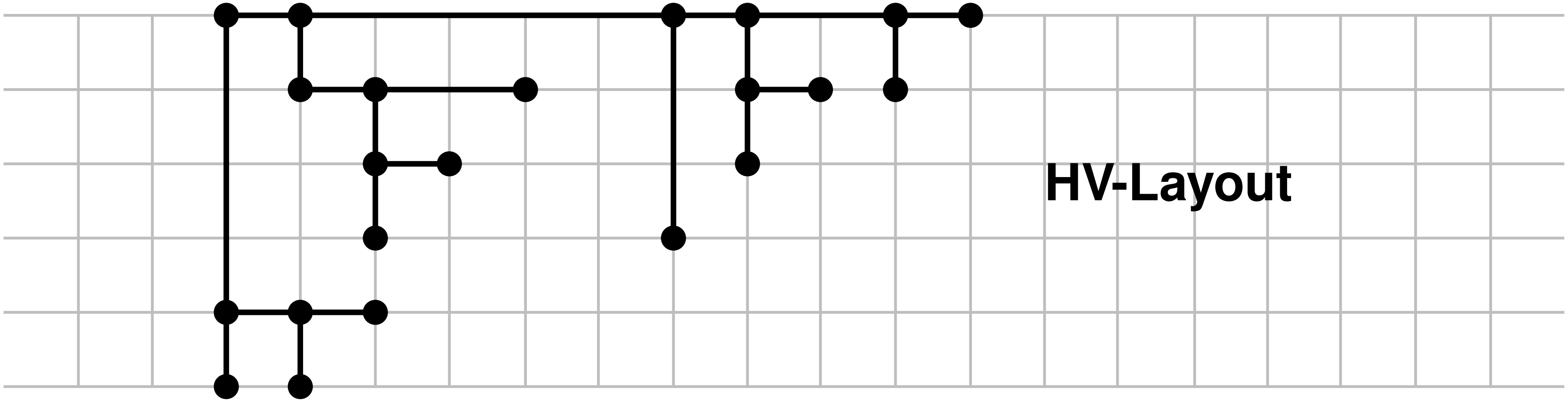
- Children are vertically and horizontally aligned with the root
- The bounding boxes of the children do not intersect

Drawing Aesthetics:

- Height, width, area

HV-Layout

Divide & Conquer Approach:

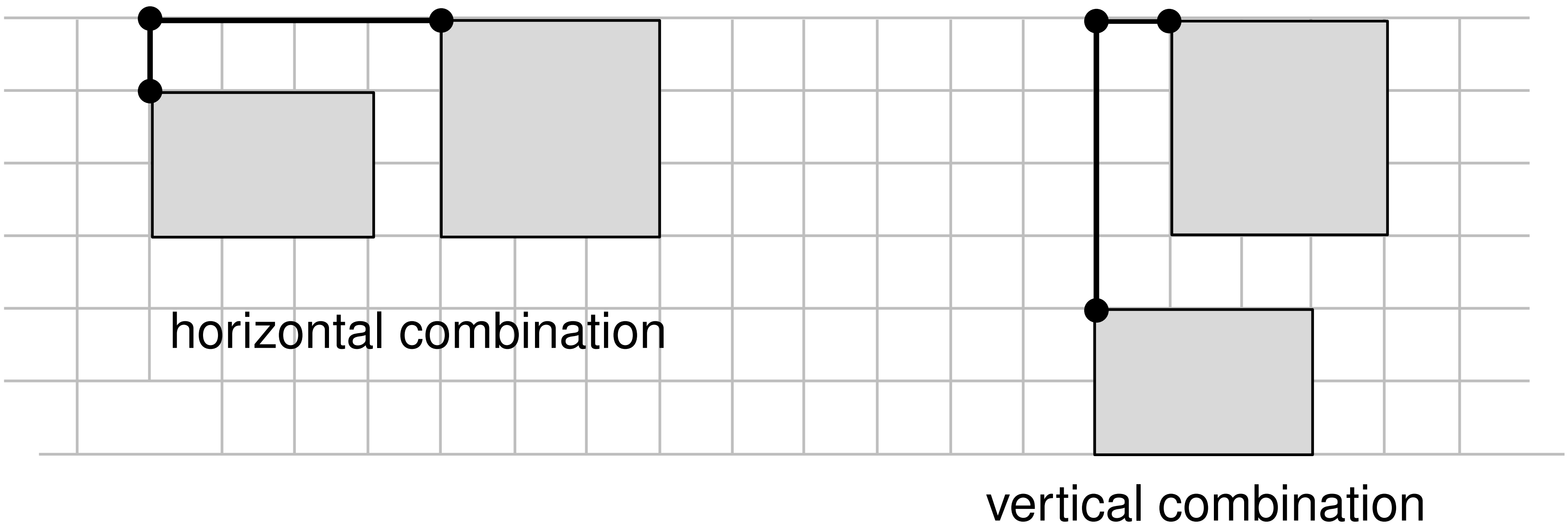


HV-Layout

HV-Layout

Induction base:

Induction step: combine layouts



Right-Heavy HV-Layout

Right-Heavy approach:

- At every induction step apply horizontal combination
- Place the larger subtree to the right

Right-Heavy HV-Layout

Right-Heavy approach:

- At every induction step apply horizontal combination
- Place the larger subtree to the right

Lemma

Let T be a binary tree. The hight of the drawing constructed by Right-Heavy approach is at most $\log n$.

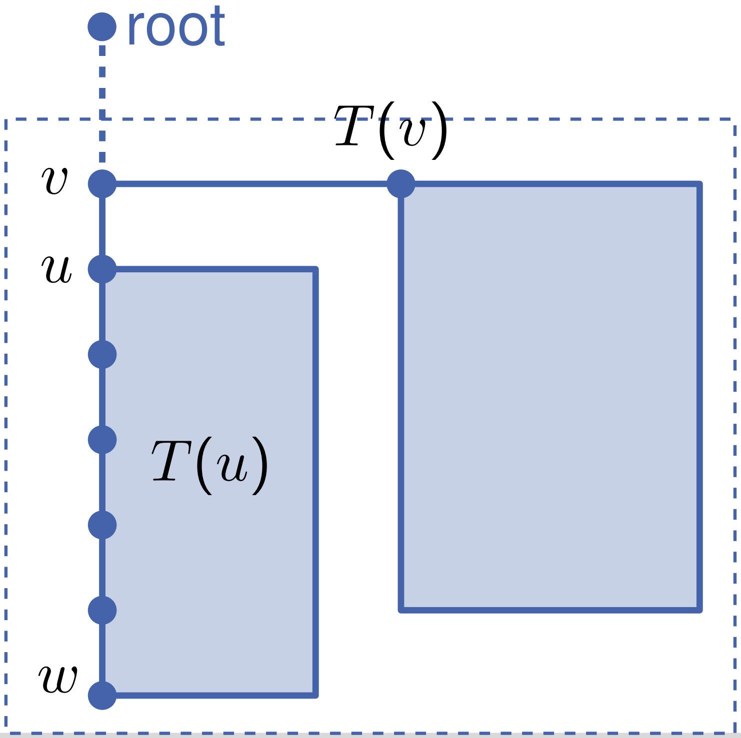
Right-Heavy HV-Layout

Right-Heavy approach:

- At every induction step apply horizontal combination
- Place the larger subtree to the right

Lemma

Let T be a binary tree. The hight of the drawing constructed by Right-Heavy approach is at most $\log n$.

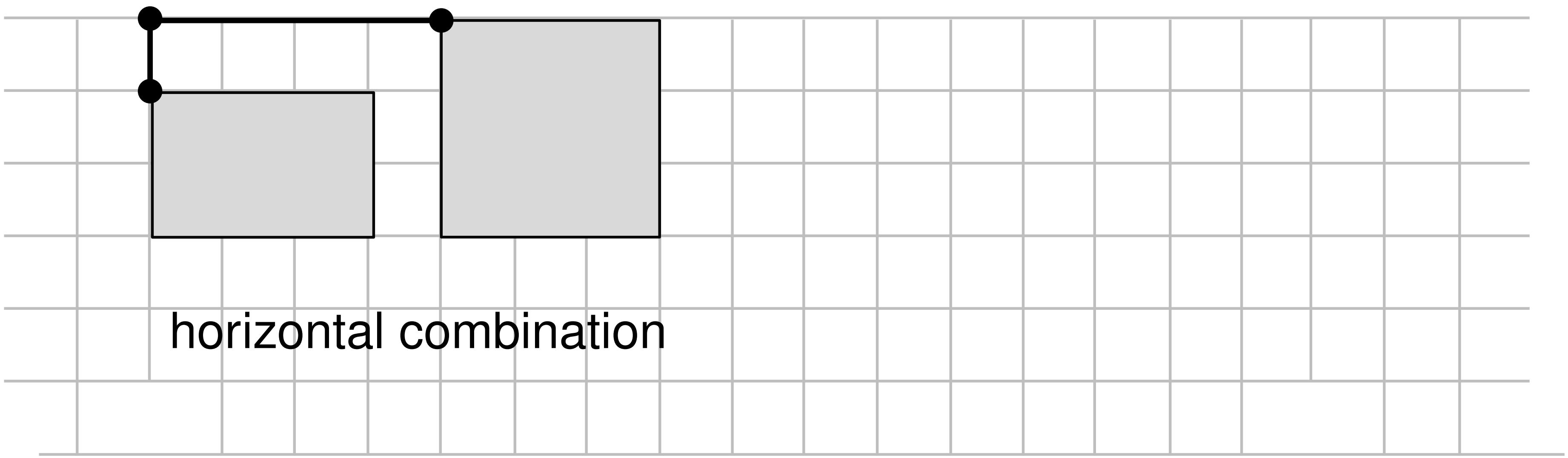


Proof:

- Each vertical edge has length 1
- Let w be the lowest node in the drawing
- Let P be a path from w to the root of T
- For every edge (u, v) in P: $|T(v)| > 2|T(u)|$
- $\Rightarrow P$ contains at most $\log n$ edges

Right-Heavy HV-Layout

- At every induction step apply horizontal combination
- Place the larger subtree to the right



Discuss with your neighbour(s) and then share **10 min**

- What are the implementational details of the algorithm? How to compute the coordinates? Can we do it in $O(n)$ time?

Right-Heavy HV-Layout

Theorem

Let T be a binary tree with n vertices. The Right-Heavy algorithm constructs in $O(n)$ time a drawing Γ of T such that:

Right-Heavy HV-Layout

Theorem

Let T be a binary tree with n vertices. The Right-Heavy algorithm constructs in $O(n)$ time a drawing Γ of T such that:

- Γ is HV-drawing (planar, orthogonal)

Right-Heavy HV-Layout

Theorem

Let T be a binary tree with n vertices. The Right-Heavy algorithm constructs in $O(n)$ time a drawing Γ of T such that:

- Γ is HV-drawing (planar, orthogonal)
- The width of Γ is at most

Take a minute to think about the width of the layout

1 min

Right-Heavy HV-Layout

Theorem

Let T be a binary tree with n vertices. The Right-Heavy algorithm constructs in $O(n)$ time a drawing Γ of T such that:

- Γ is HV-drawing (planar, orthogonal)
- The width of Γ is at most n-1

Right-Heavy HV-Layout

Theorem

Let T be a binary tree with n vertices. The Right-Heavy algorithm constructs in $O(n)$ time a drawing Γ of T such that:

- Γ is HV-drawing (planar, orthogonal)
- The width of Γ is at most n-1
- The height is at most $\log n$

Right-Heavy HV-Layout

Theorem

Let T be a binary tree with n vertices. The Right-Heavy algorithm constructs in $O(n)$ time a drawing Γ of T such that:

- Γ is HV-drawing (planar, orthogonal)
- The width of Γ is at most n-1
- The height is at most $\log n$
- The area is $O(n \log n)$

Right-Heavy HV-Layout

Karlsruhe Institute of Technology

Theorem

Let T be a binary tree with n vertices. The Right-Heavy algorithm constructs in $O(n)$ time a drawing Γ of T such that:

- Γ is HV-drawing (planar, orthogonal)
- The width of Γ is at most n-1
- The height is at most $\log n$
- The area is $O(n \log n)$
- Simply and axially isomorphic subtrees have congruent drawings, up to translation

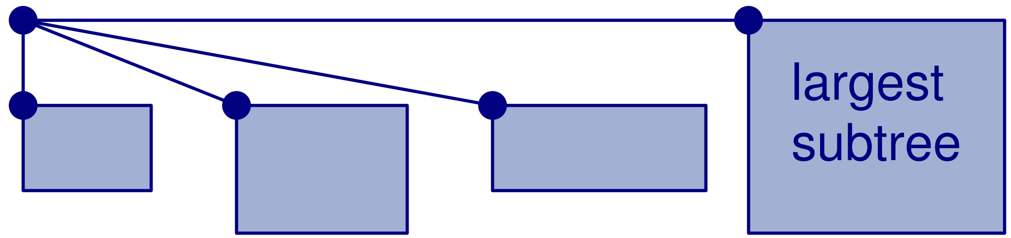
Right-Heavy HV-Layout

Theorem

Let T be a binary tree with n vertices. The Right-Heavy algorithm constructs in $O(n)$ time a drawing Γ of T such that:

- Γ is HV-drawing (planar, orthogonal)
- The width of Γ is at most n-1
- The height is at most $\log n$
- The area is $O(n \log n)$
- Simply and axially isomorphic subtrees have congruent drawings, up to translation

General rooted tree:



HV-Layout

Bad news We can not minimize the area by using divide & conquer approach

HV-Layout

Bad news We can not minimize the area by using divide & conquer approach

Good news We can compute minimum area using Dynamic Programming

HV-Layout

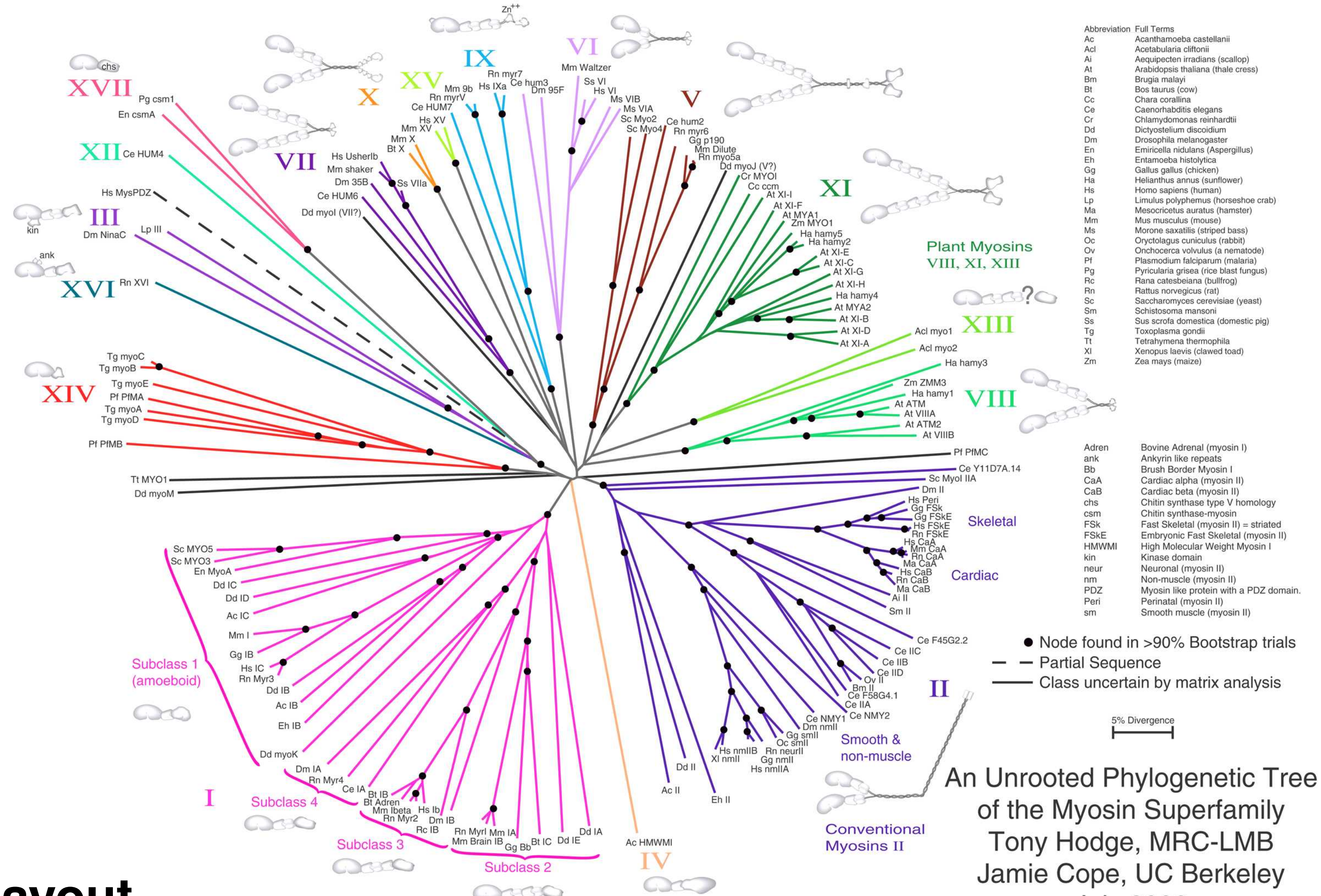
Bad news We can not minimize the area by using divide & conquer approach

Good news We can compute minimum area using Dynamic Programming

HV-Layout for Trees

- Book Di Battista et al: Chapter 3.1.4
- Skript: page 86

Applications

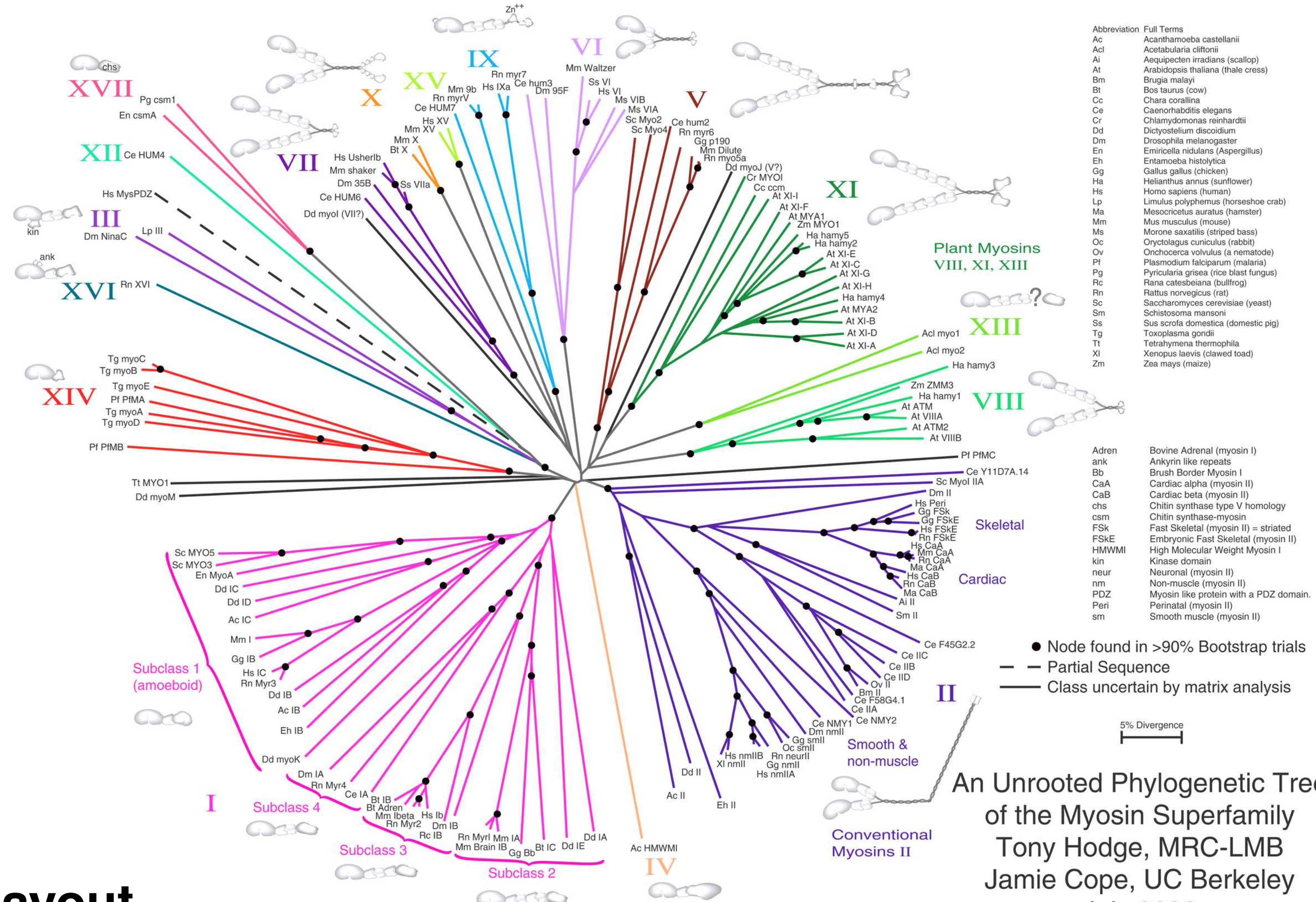


Radial layout

An unrooted phylogenetic tree for myosin, a superfamily of proteins.
"A myosin family tree" *Journal of Cell Science*

12

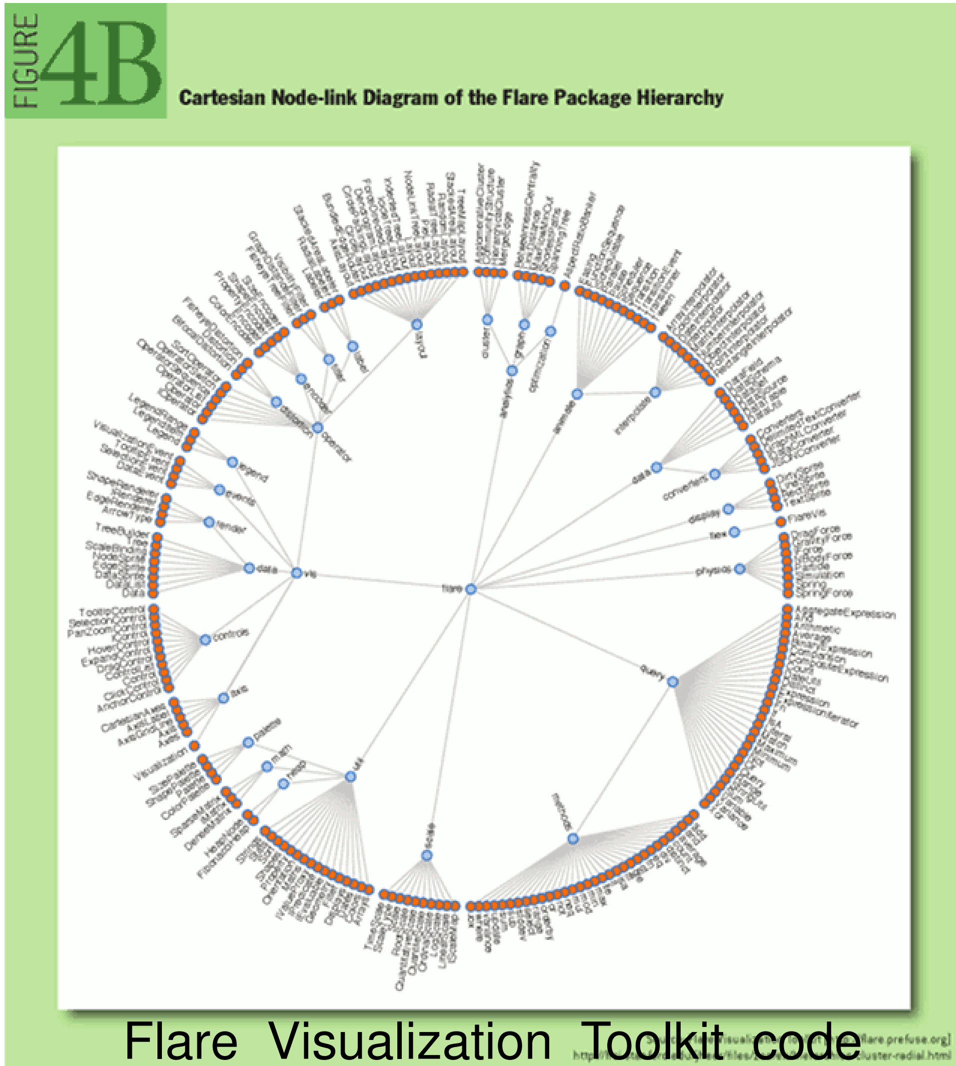
Applications



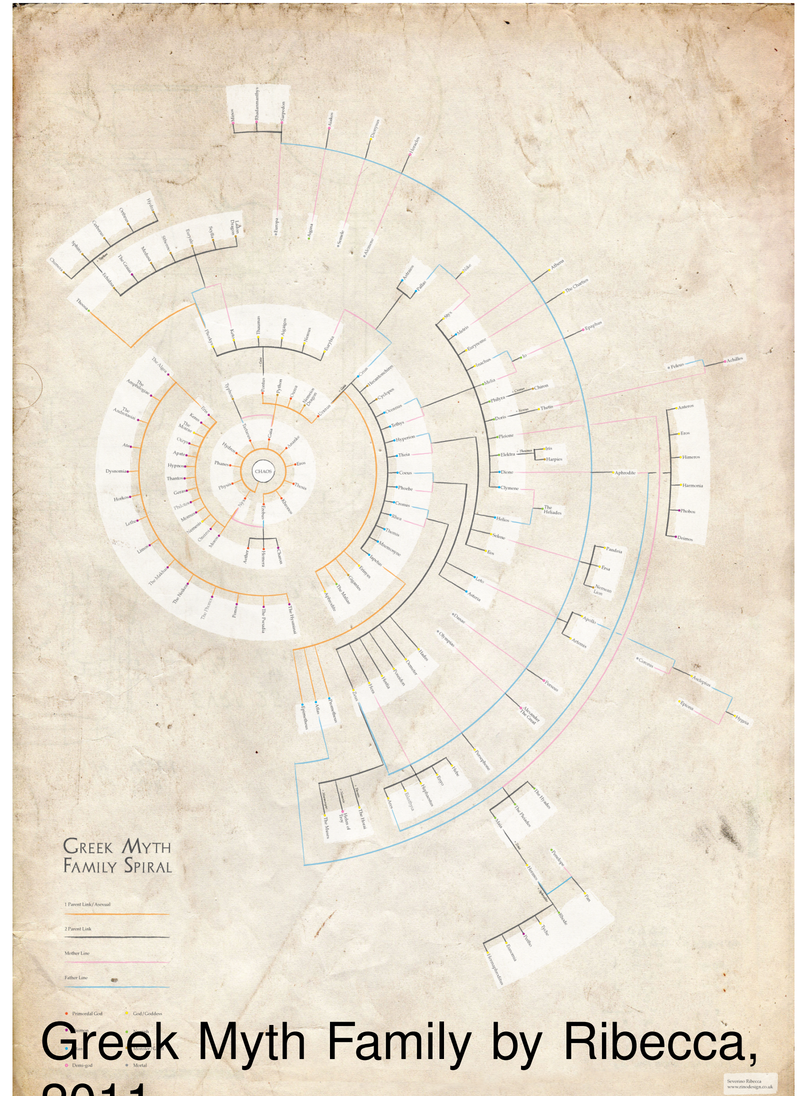
Radial layout

An unrooted phylogenetic tree for myosin, a superfamily of proteins.
"A myosin family tree" *Journal of Cell Science*

Applications

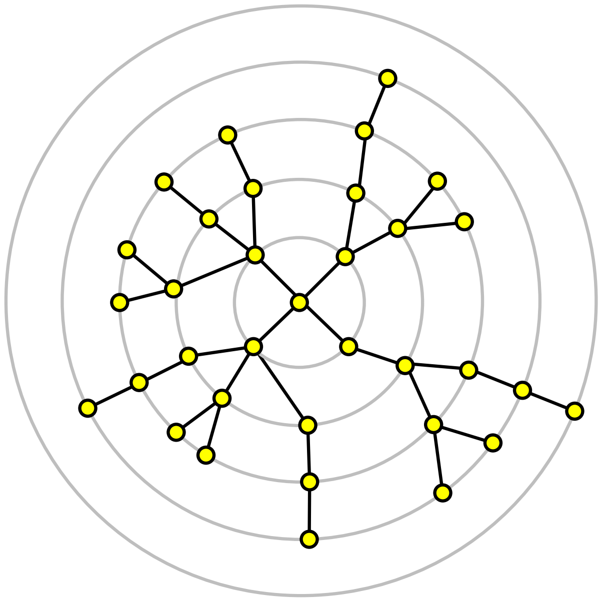


Flare Visualization Toolkit code structure by Heer, Bostock and Ogievetsky, 2010



Greek Myth Family by Ribecca, 2011

Radial Layout



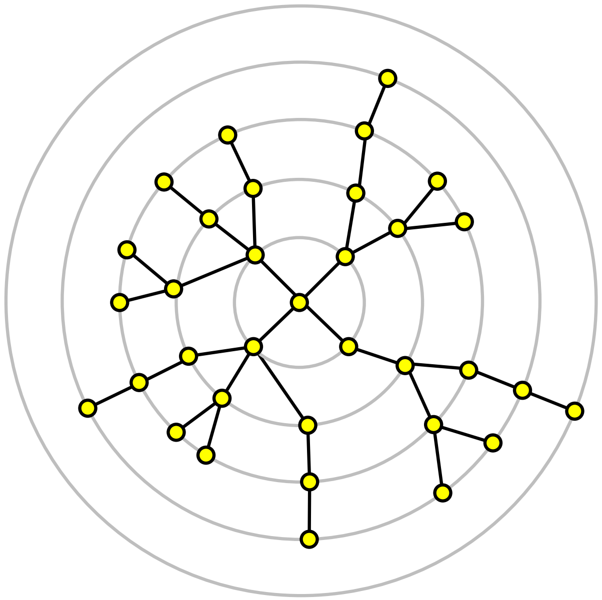
Drawing Conventions:

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

Drawing Aesthetics:

- Distribution of the vertices

Radial Layout



Drawing Conventions:

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

Drawing Aesthetics:

- Distribution of the vertices

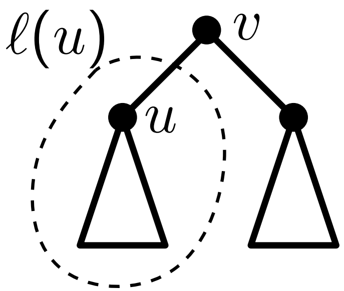
Take a minute to think about a possible algorithm to optimize the distribution of the vertices

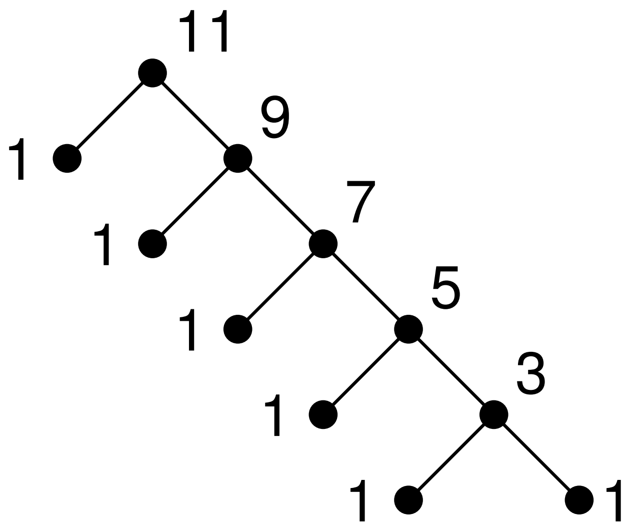
1 min

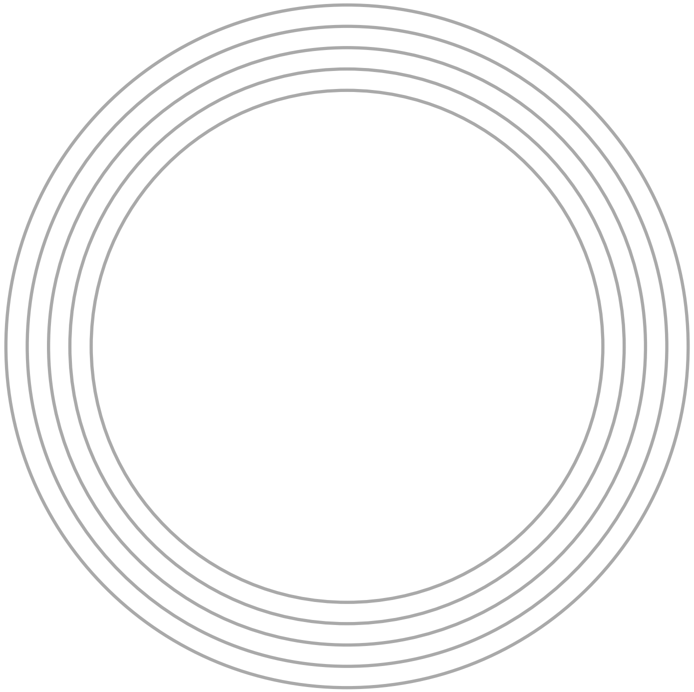
Radial Layout

Example:

- Angle corresponding to the subtree rooted at u: $\tau_u = \frac{\ell(u)}{\ell(v)-1}$



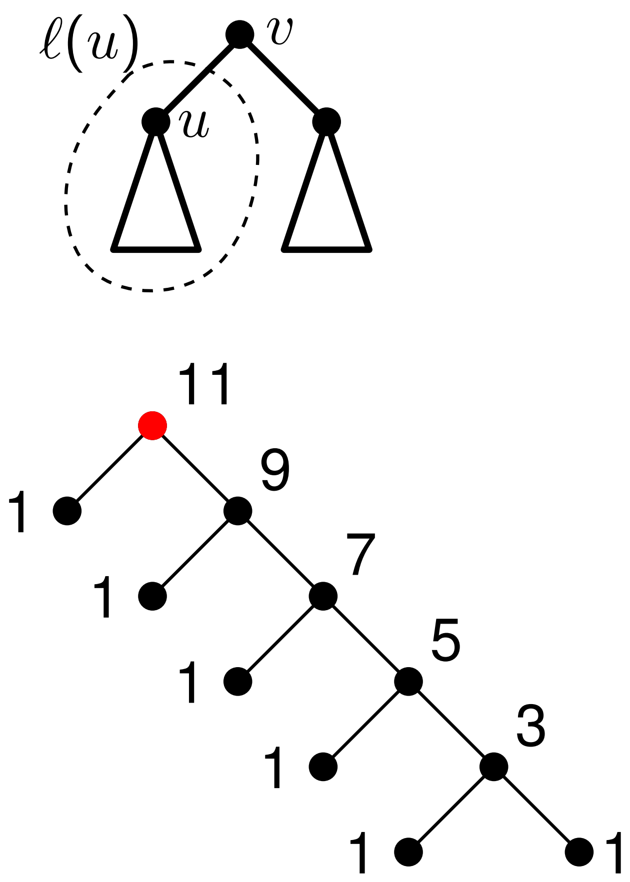


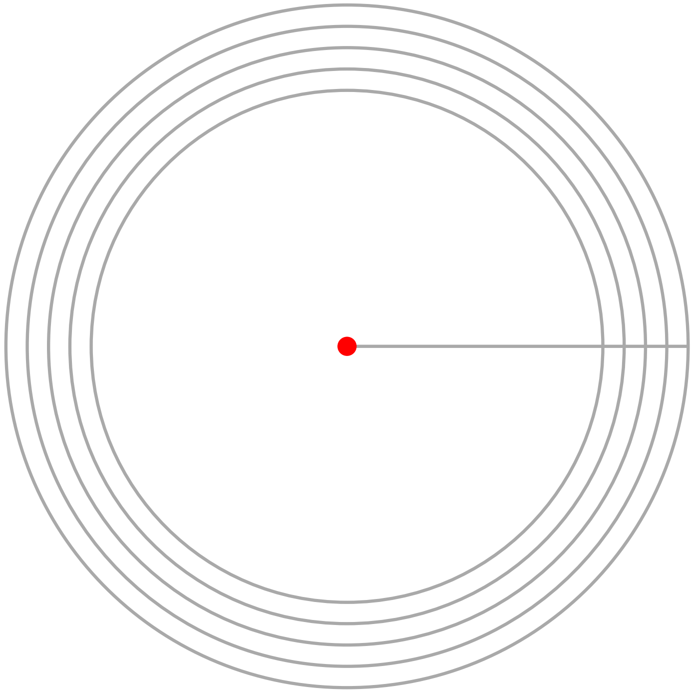


Radial Layout

Example:

- Angle corresponding to the subtree rooted at u: $\tau_u = \frac{\ell(u)}{\ell(v)-1}$

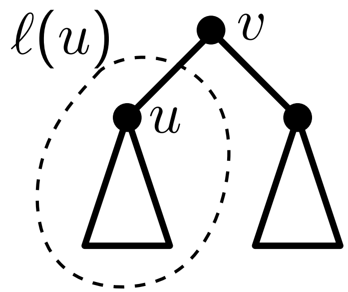


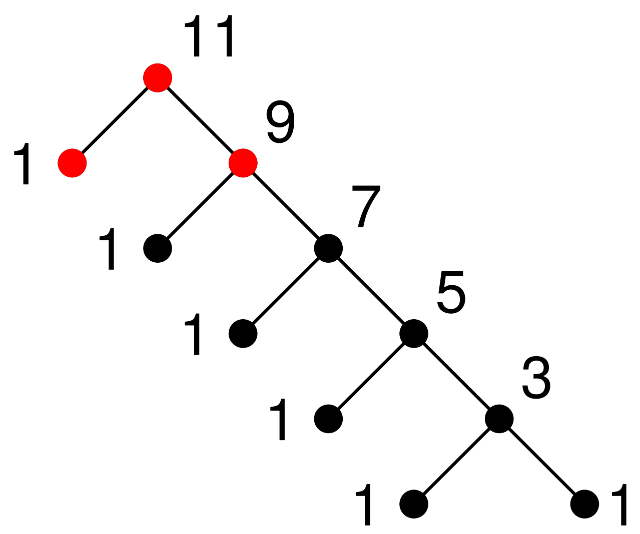


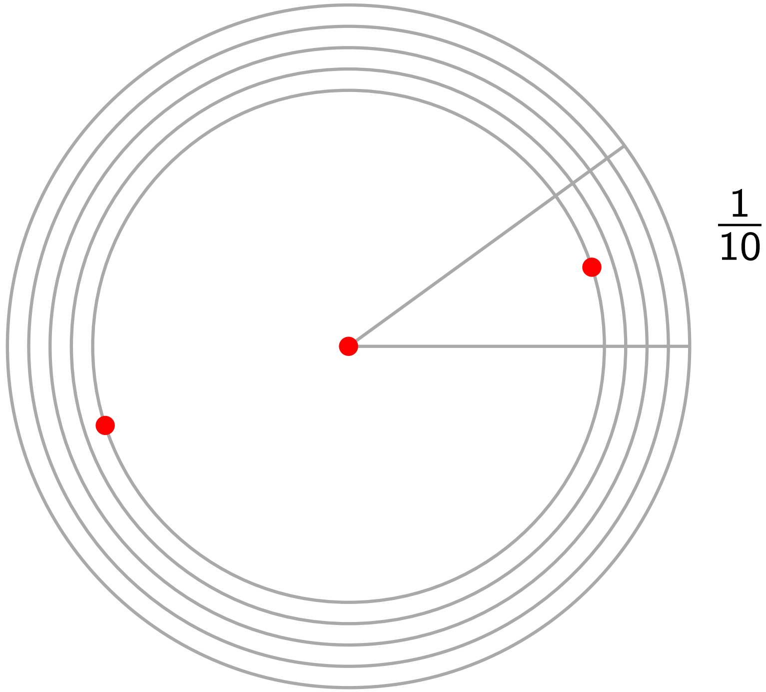
Radial Layout

Example:

- Angle corresponding to the subtree rooted at u: $\tau_u = \frac{\ell(u)}{\ell(v)-1}$





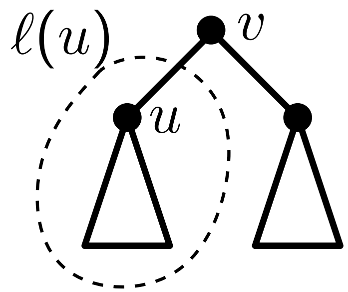


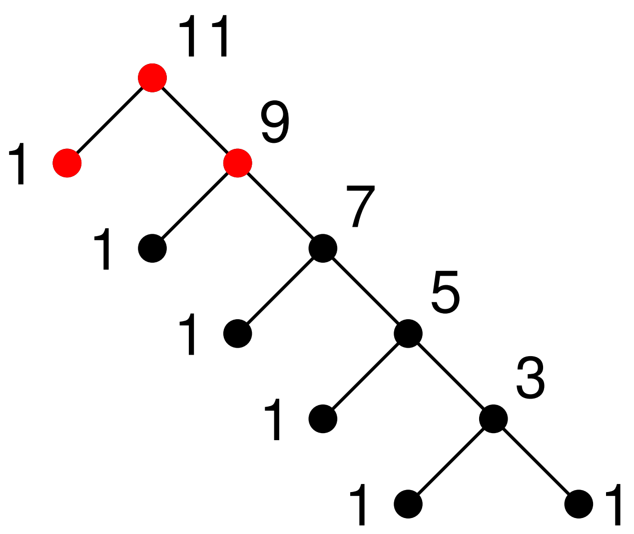
16 - 3

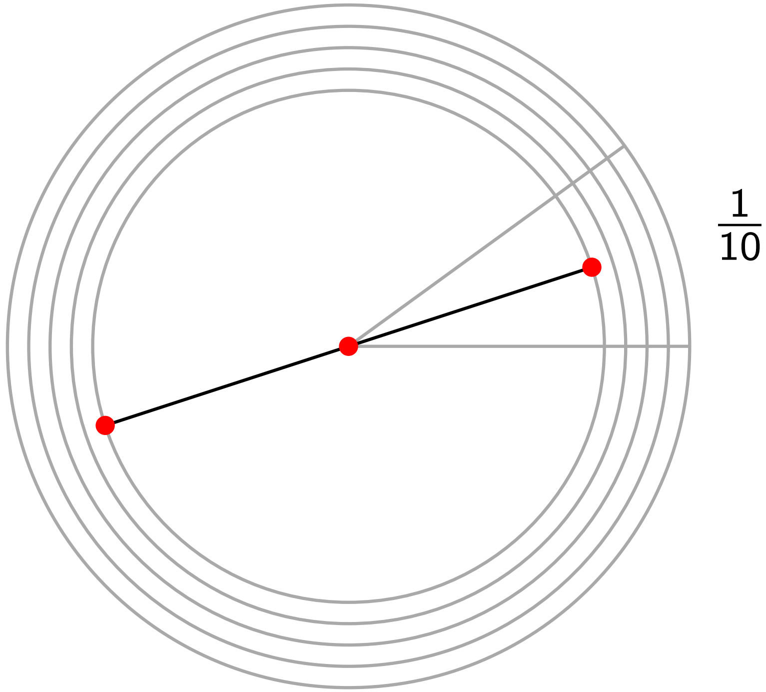
Radial Layout

Example:

- Angle corresponding to the subtree rooted at u: $\tau_u = \frac{\ell(u)}{\ell(v)-1}$



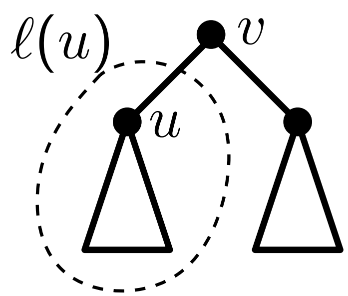


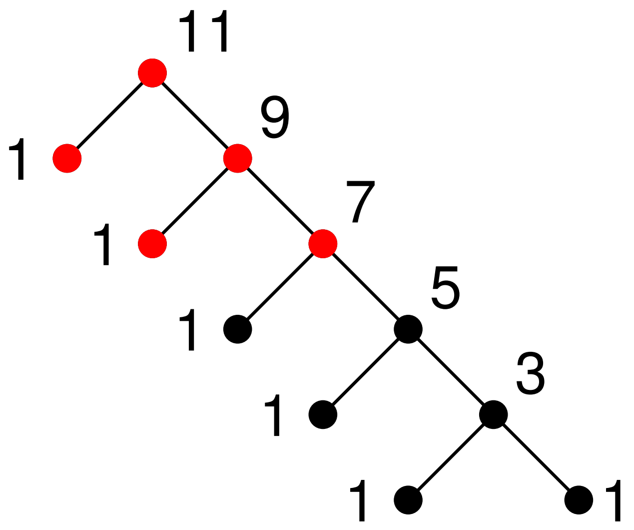


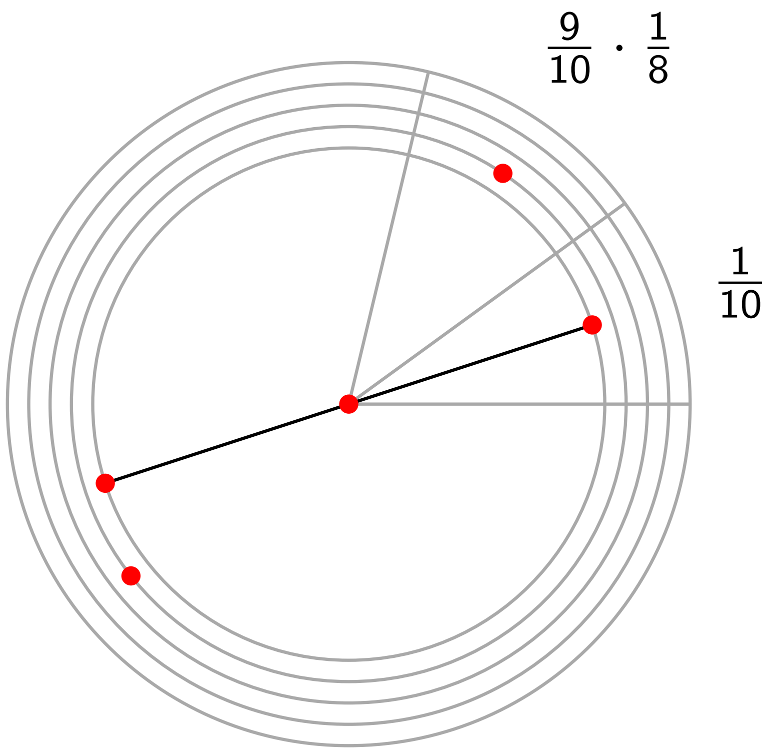
16 - 4

Example:

- Angle corresponding to the subtree rooted at u: $\tau_u = \frac{\ell(u)}{\ell(v)-1}$



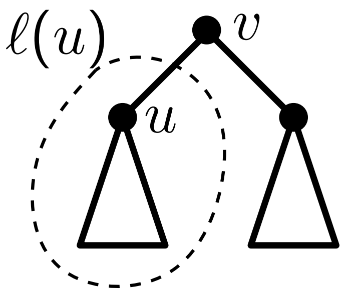


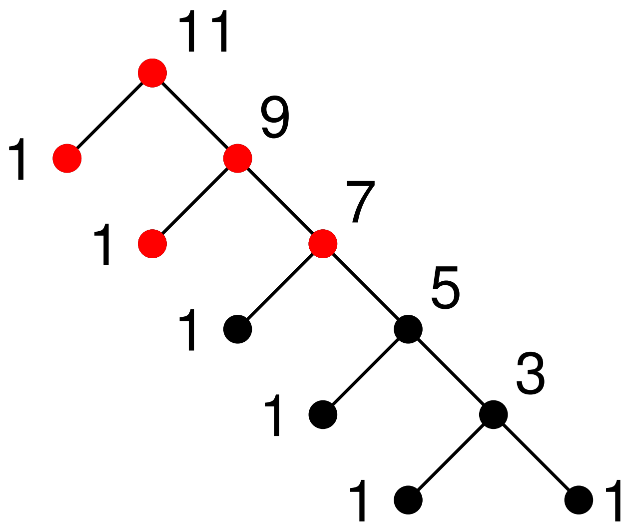


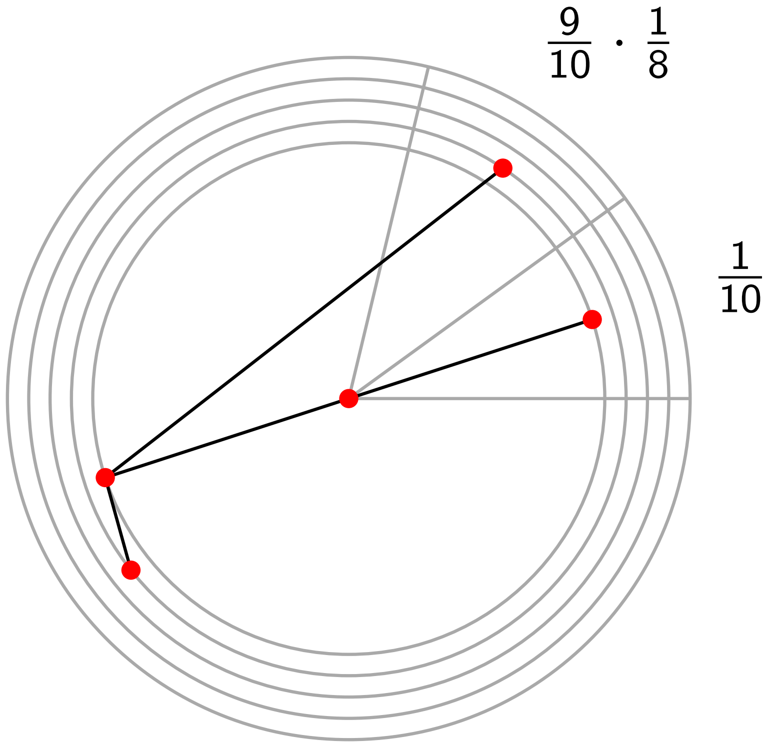
Radial Layout

Example:

- Angle corresponding to the subtree rooted at u: $\tau_u = \frac{\ell(u)}{\ell(v)-1}$

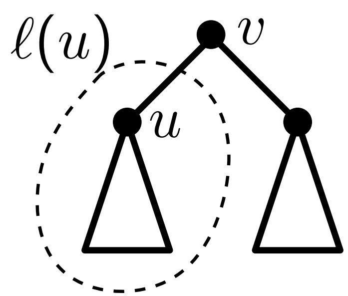


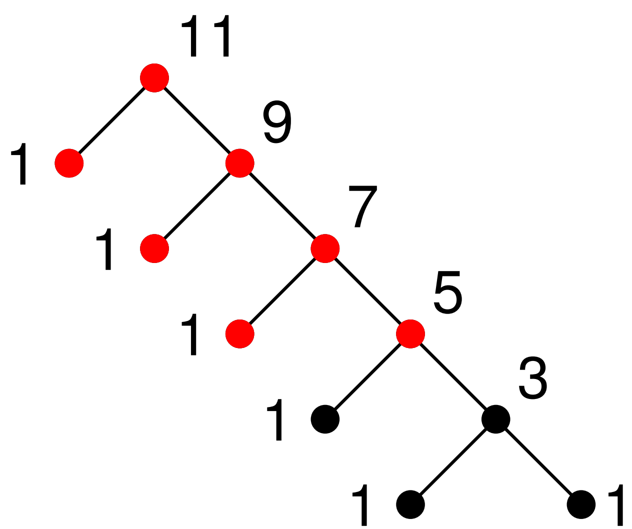


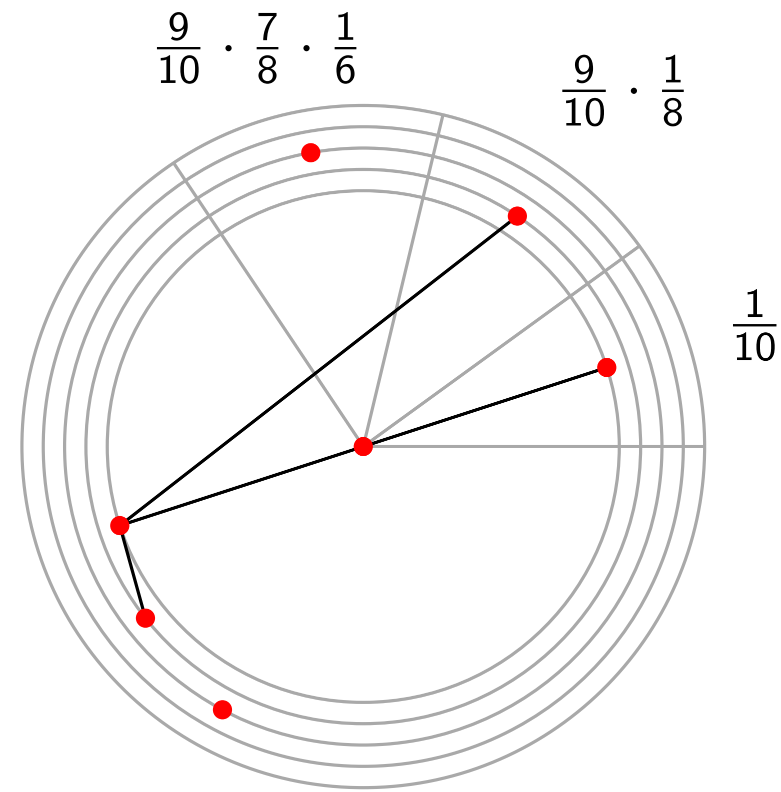


Example:

- Angle corresponding to the subtree rooted at u: $\tau_u = \frac{\ell(u)}{\ell(v)-1}$





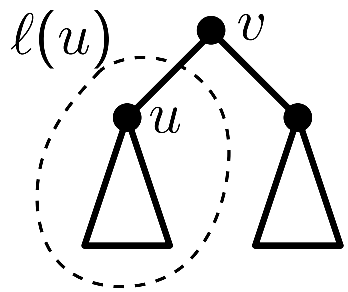


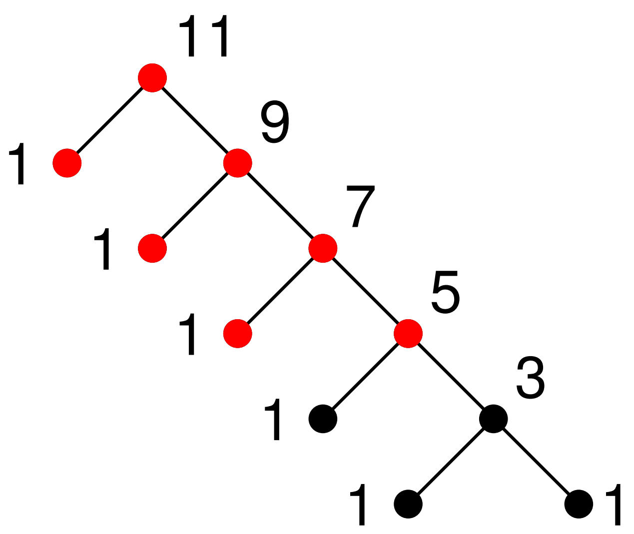
16 - 7

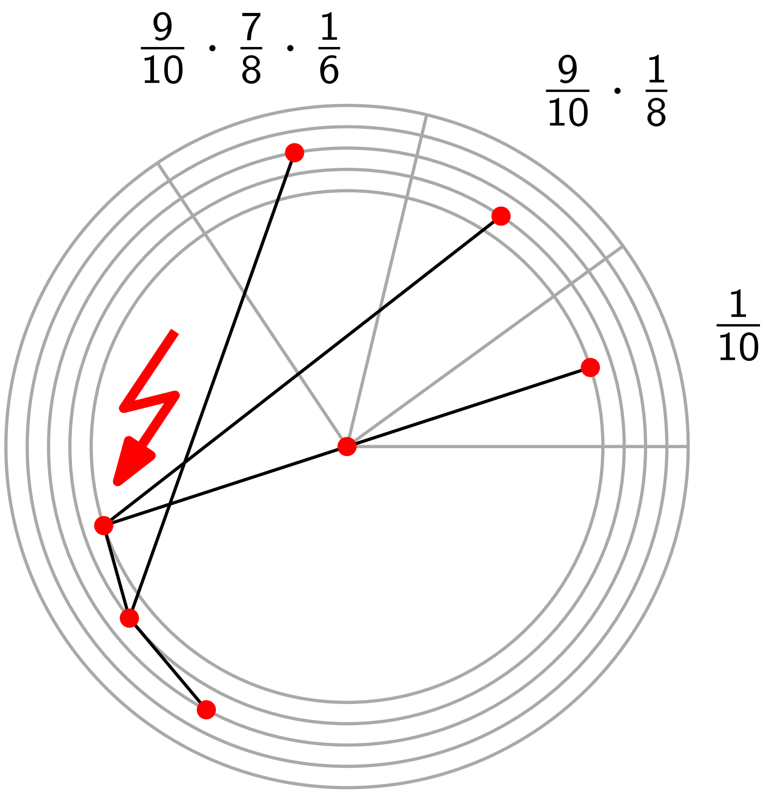
Radial Layout

Example:

- Angle corresponding to the subtree rooted at u: $\tau_u = \frac{\ell(u)}{\ell(v)-1}$

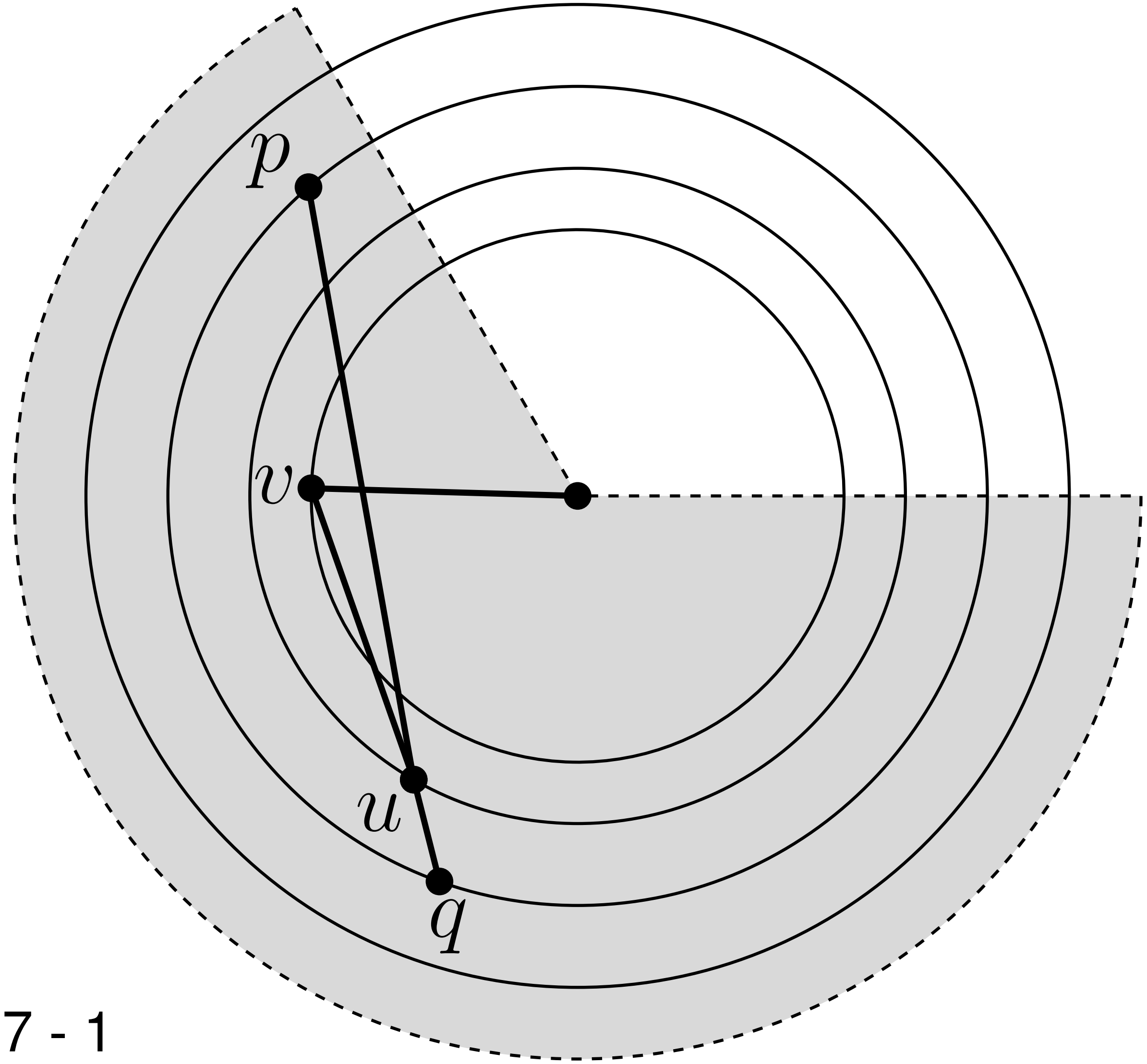






Radial Layout

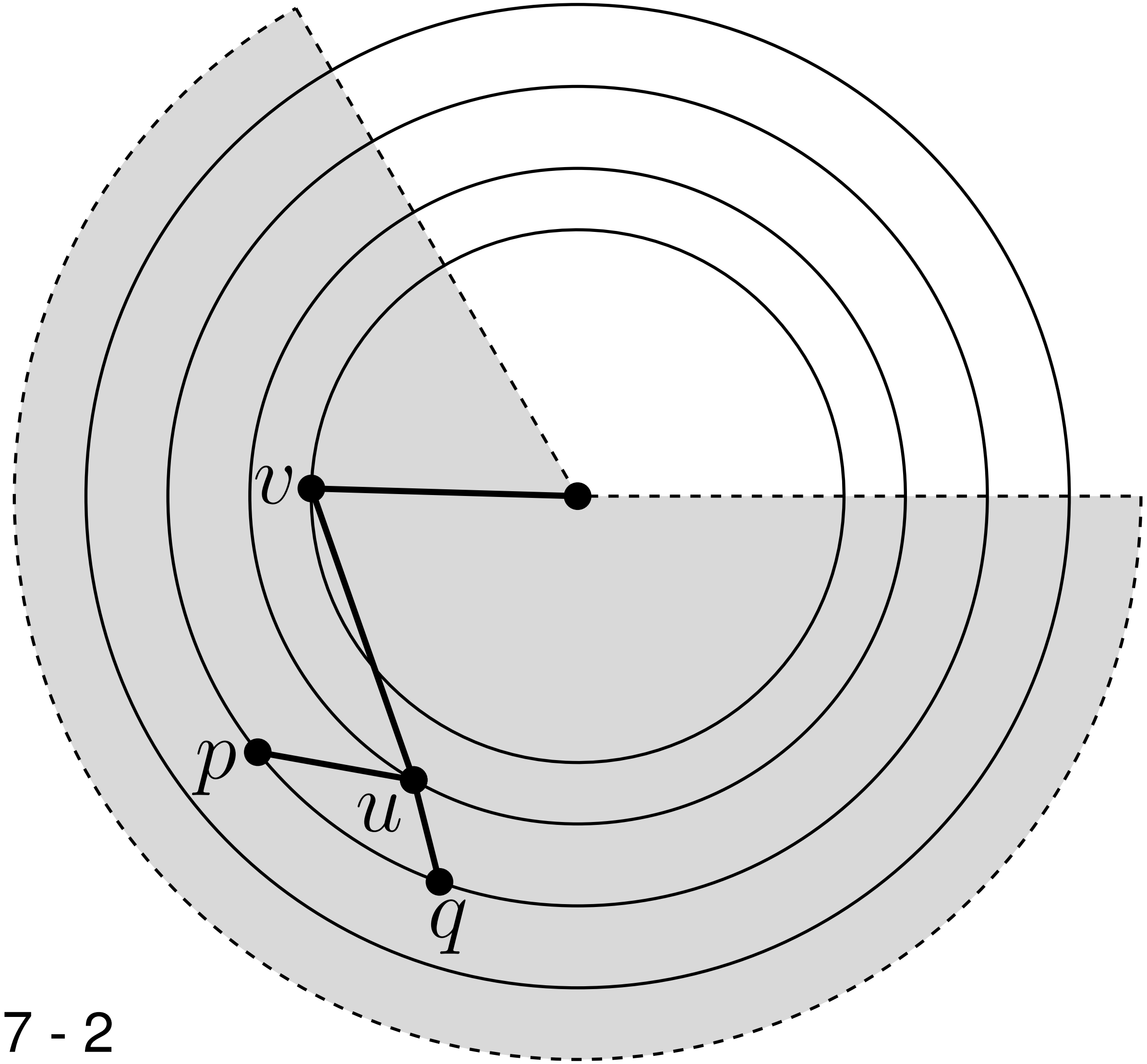
How to avoid crossings:



Institut für Theoretische Informatik
Lehrstuhl Algorithmik I

Radial Layout

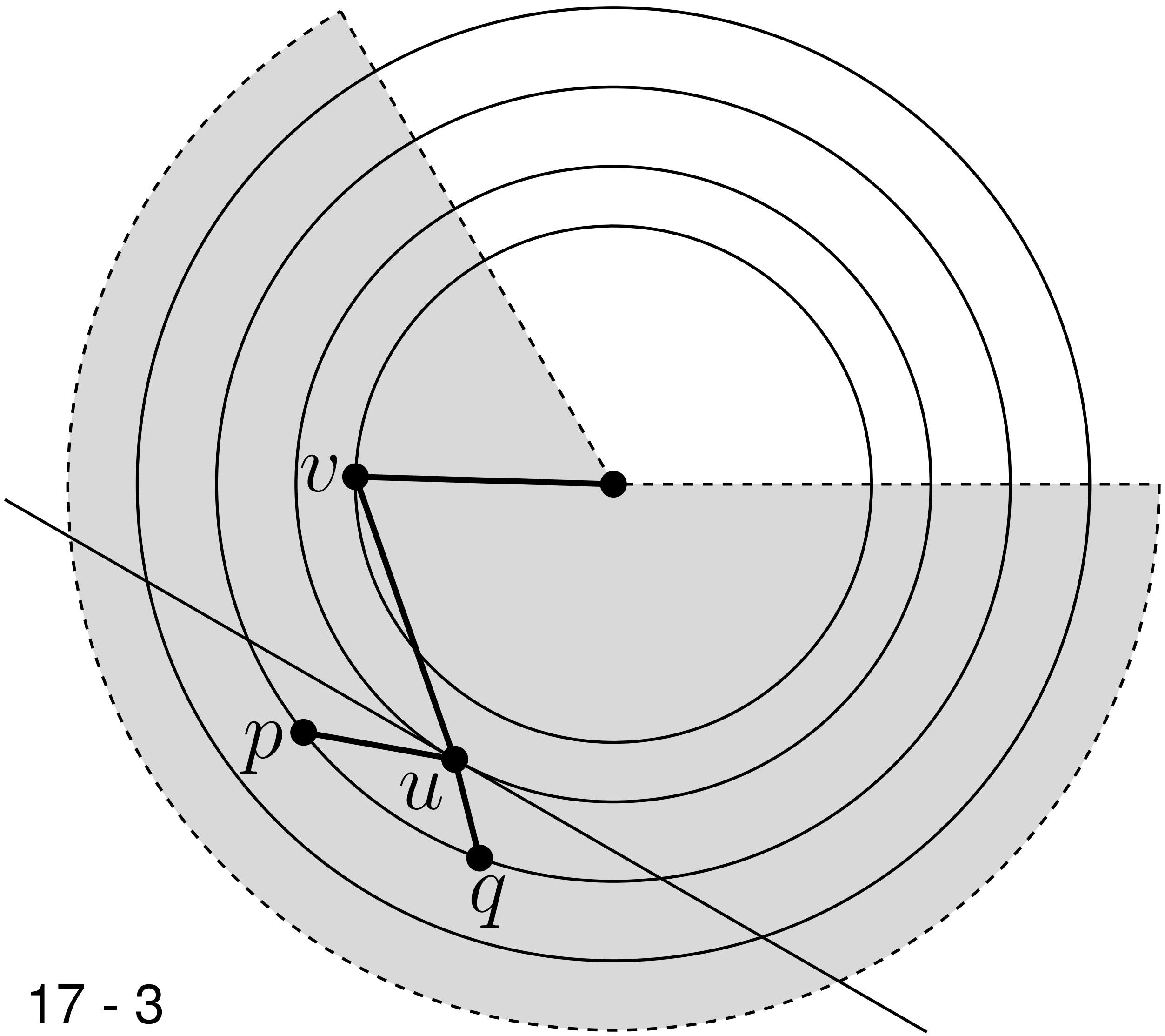
How to avoid crossings:



Institut für Theoretische Informatik
Lehrstuhl Algorithmik I

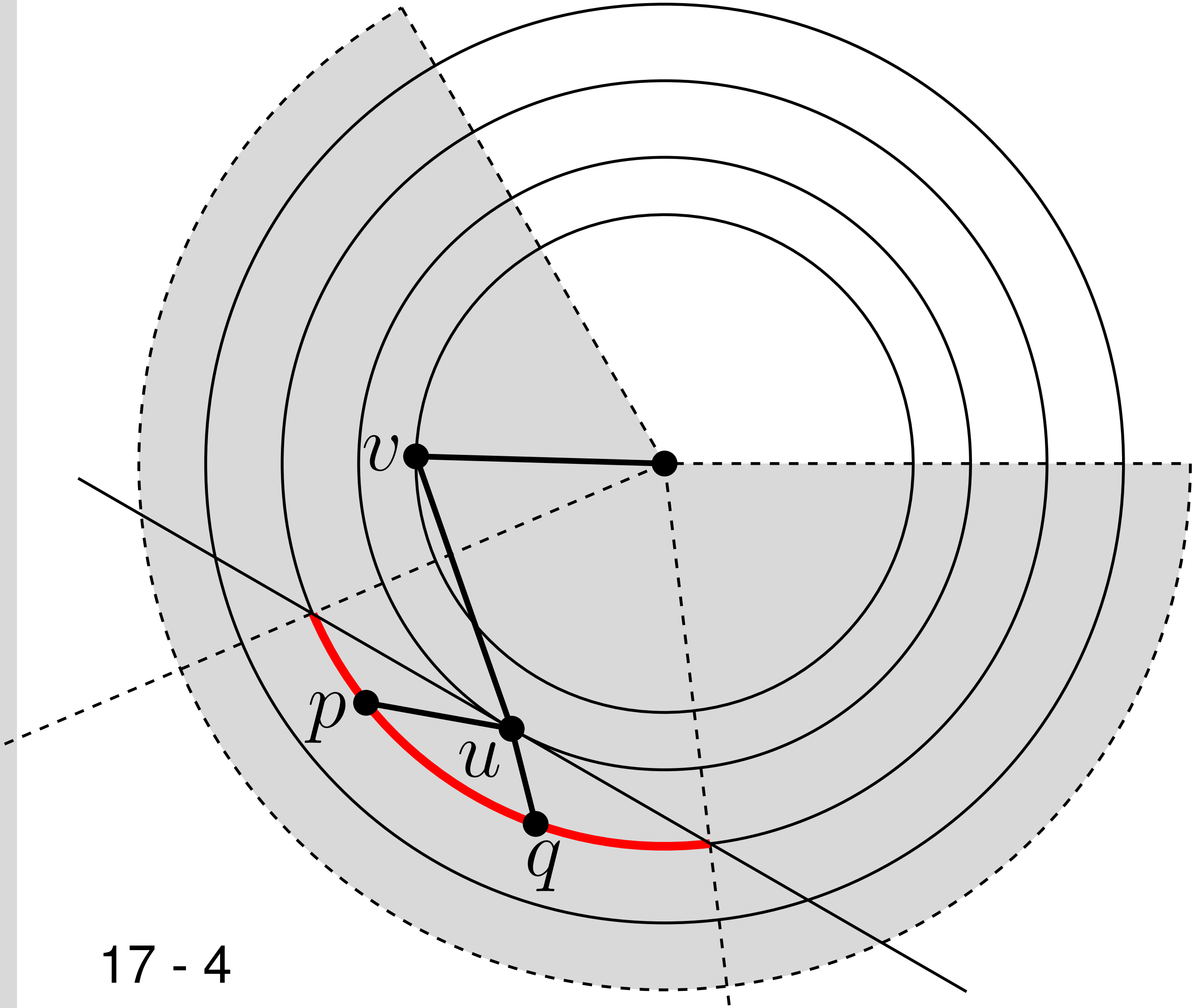
Radial Layout

How to avoid crossings:



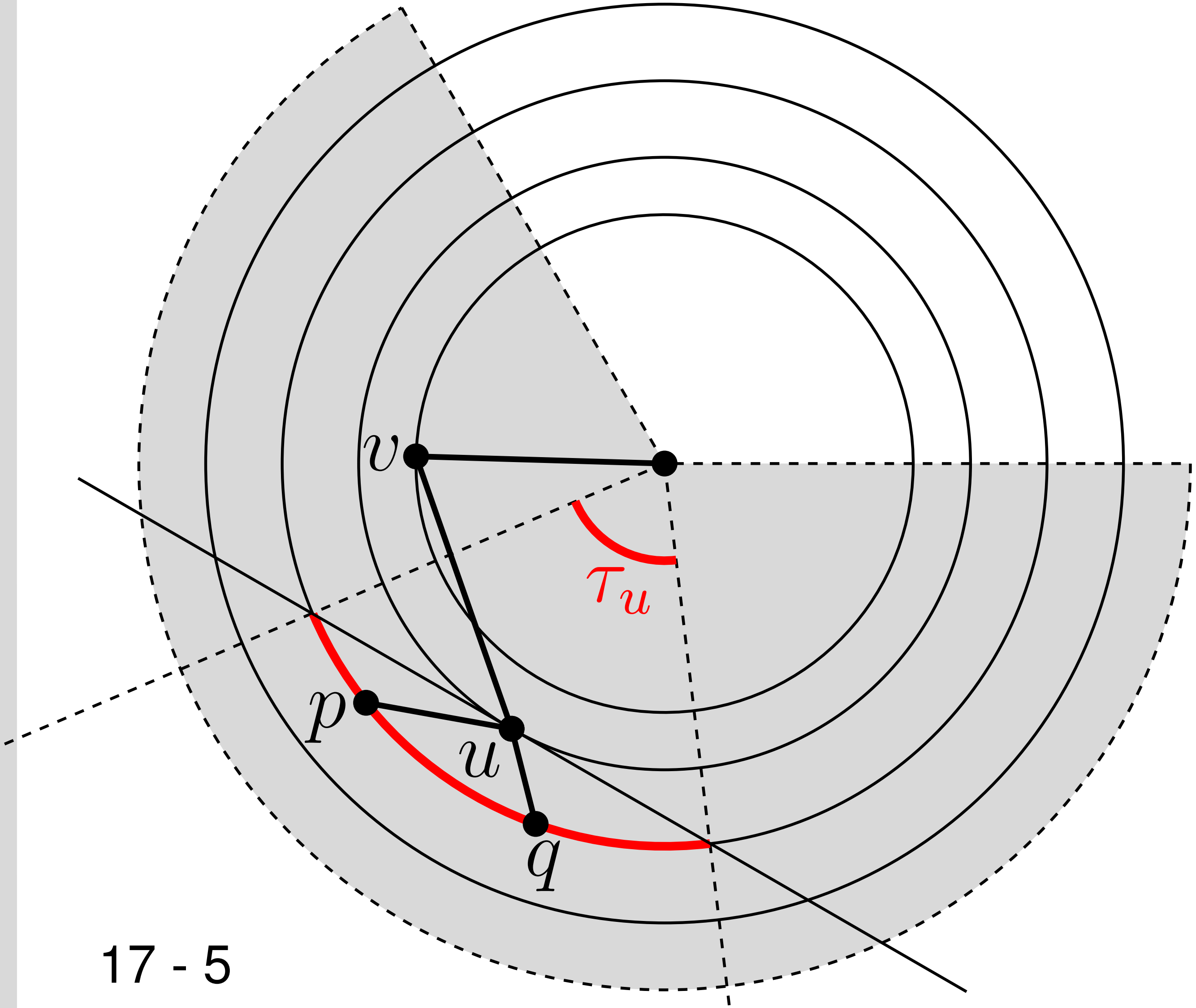
Radial Layout

How to avoid crossings:



Radial Layout

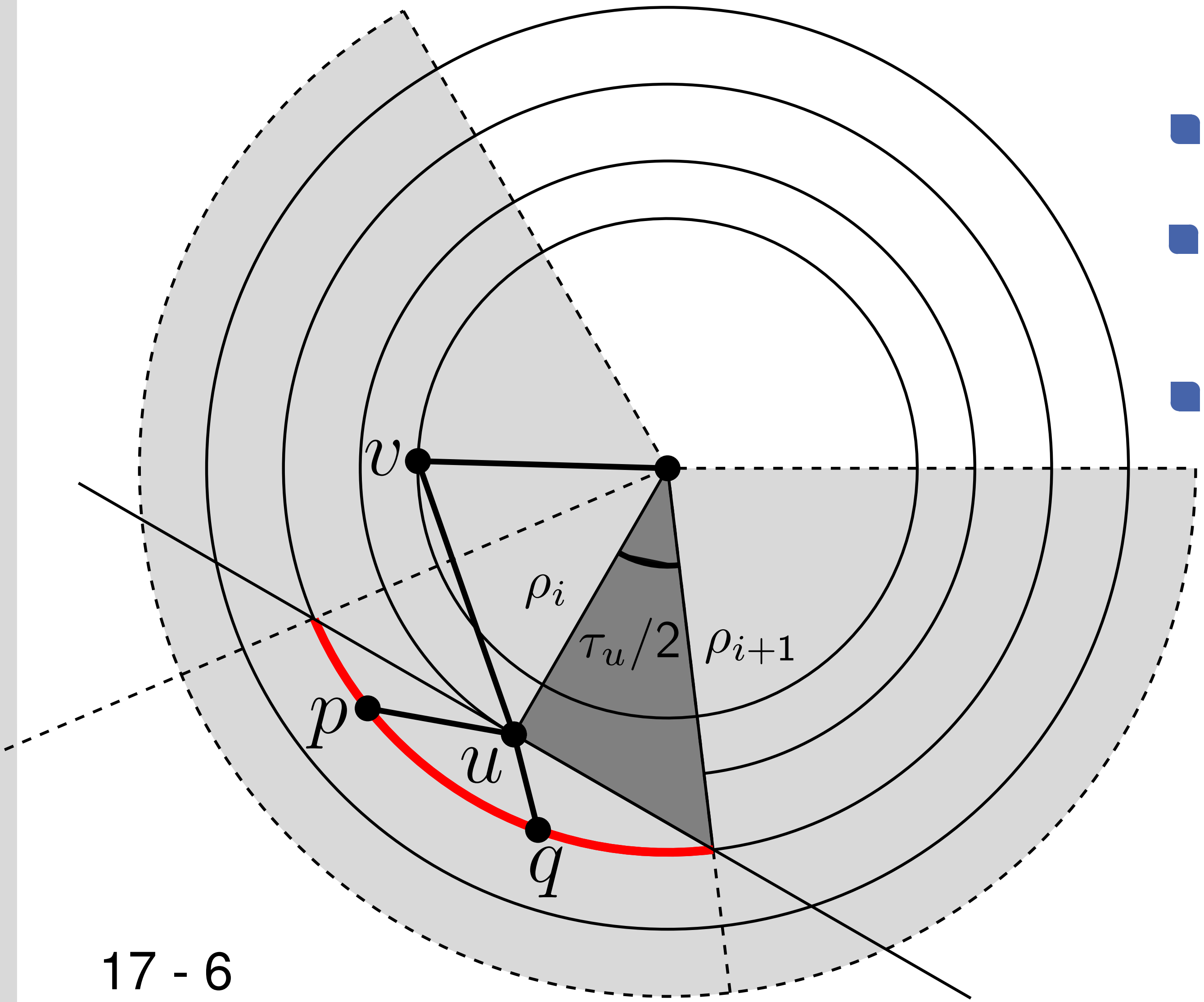
How to avoid crossings:



17 - 5

Radial Layout

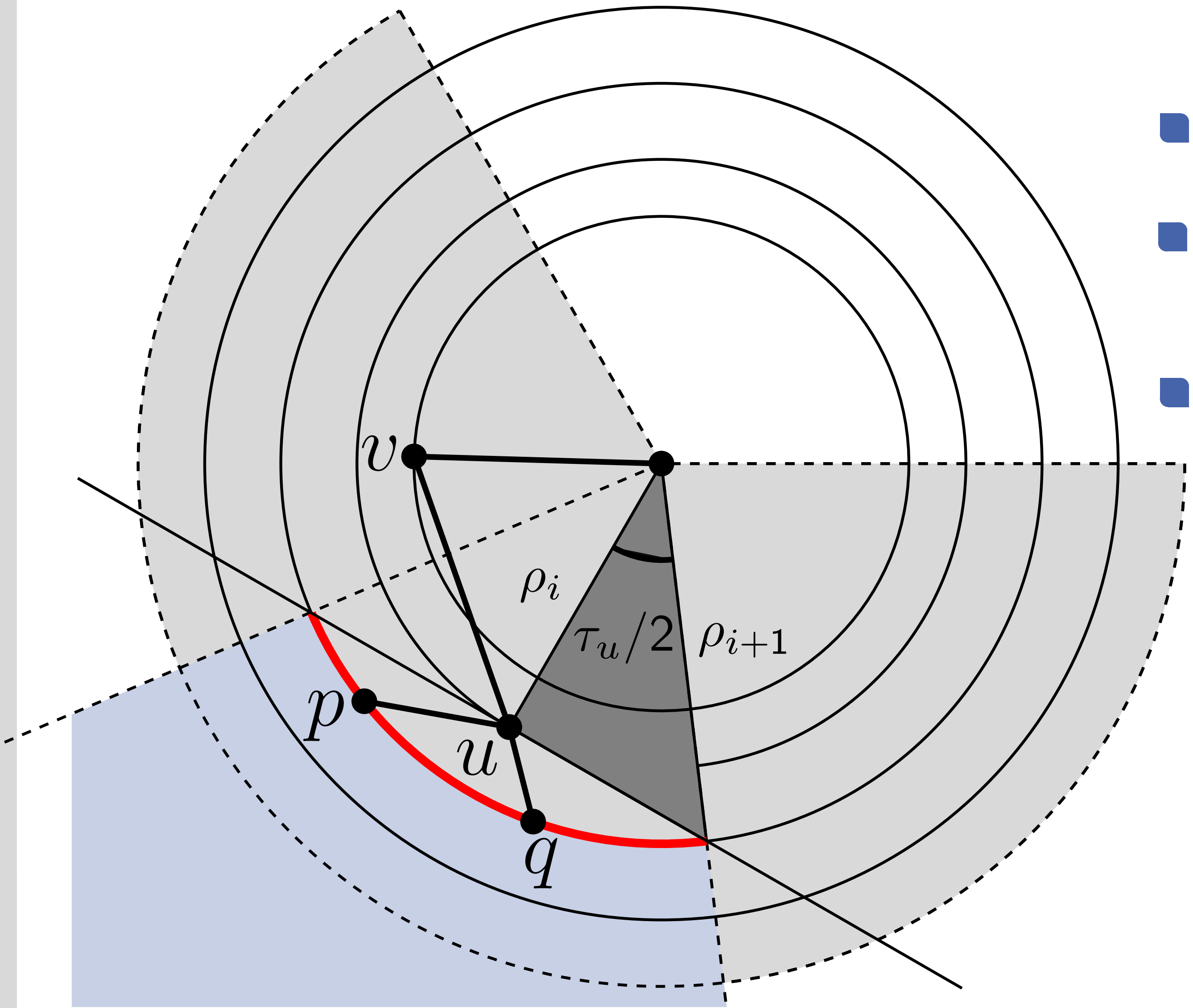
How to avoid crossings:



- τ_u - angle of the wedge corresponding to vertex u
- ρ_i - raduis of layer i
- $\ell(v)$-number of nodes in the subtree rooted at v
- $\cos \frac{\tau_u}{2} = \frac{\rho_i}{\rho_{i+1}}$

Radial Layout

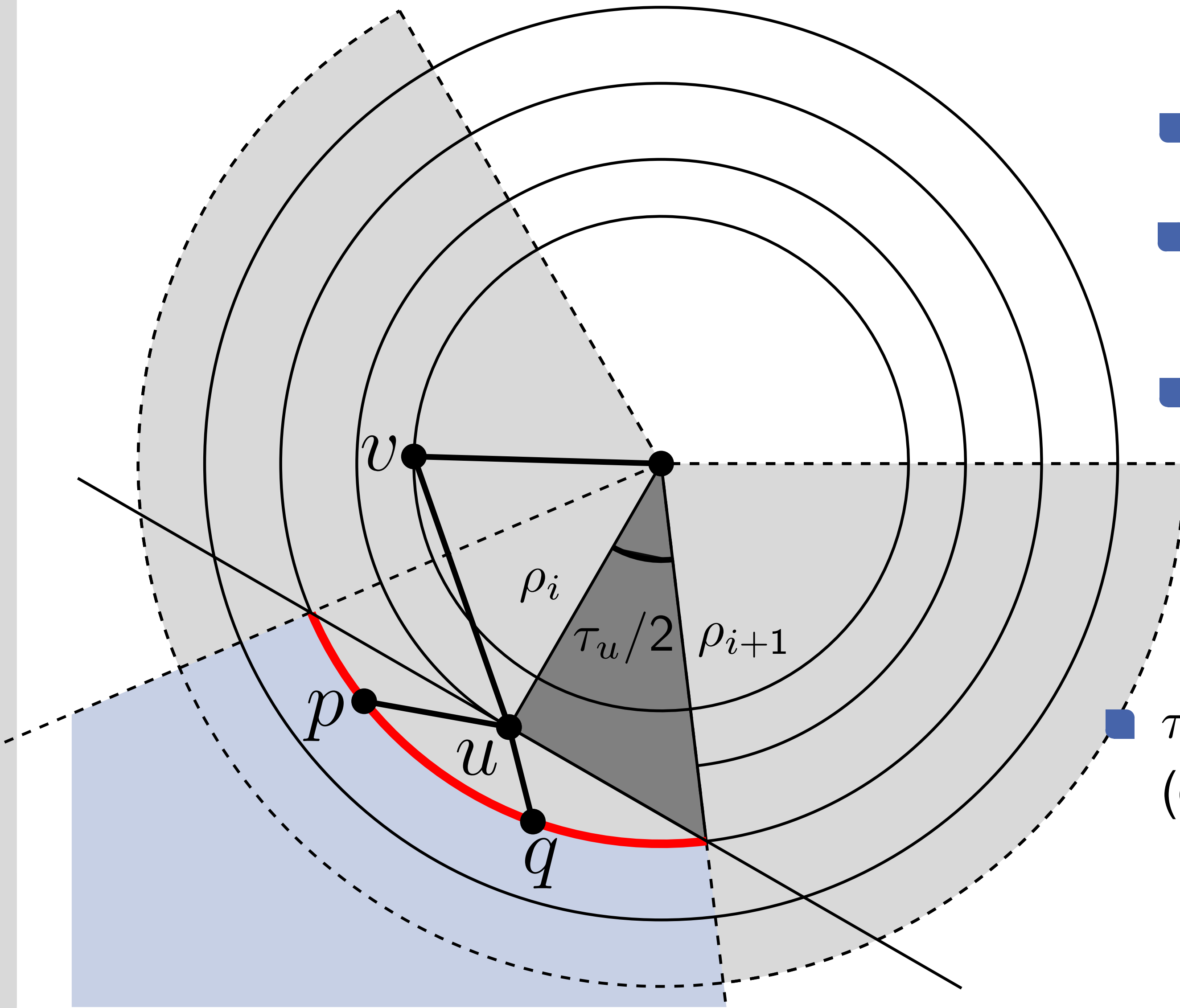
How to avoid crossings:



- τ_u - angle of the wedge corresponding to vertex u
- ρ_i - raduis of layer i
- $\ell(v)$-number of nodes in the subtree rooted at v
- $\cos \frac{\tau_u}{2} = \frac{\rho_i}{\rho_{i+1}}$

Karlsruhe Institute of Technology

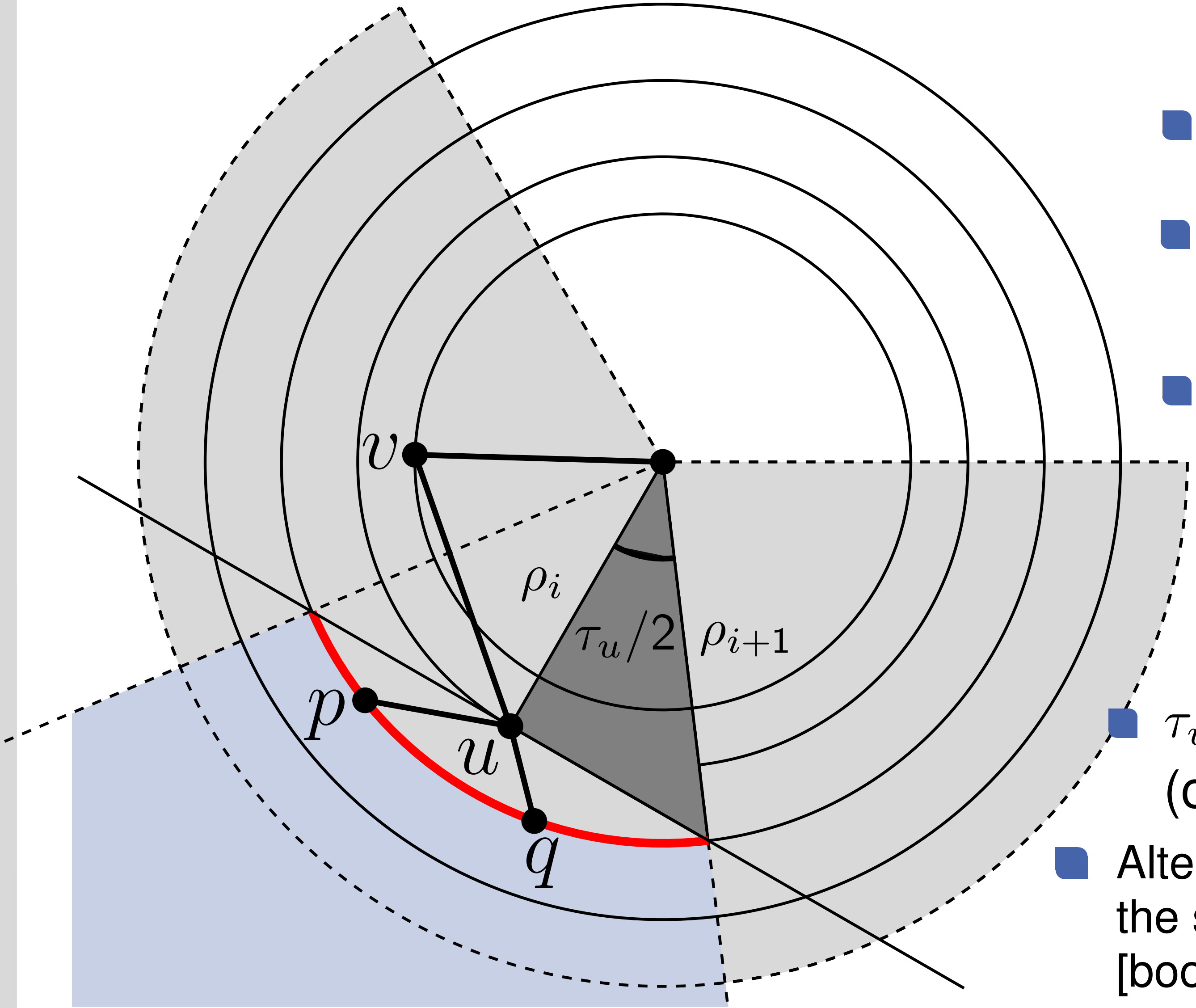
How to avoid crossings:



- τ_u - angle of the wedge corresponding to vertex u
- ρ_i - raduis of layer i
- $\ell(v)$-number of nodes in the subtree rooted at v
- $\cos\frac{\tau_u}{2} = \frac{\rho_i}{\rho_{i+1}}$
- $\tau_u = \min\{\frac{\ell(u)}{\ell(v)-1}, 2\arccos\frac{\rho_i}{\rho_{i+1}}\}$ (correction)

Radial Layout

How to avoid crossings:



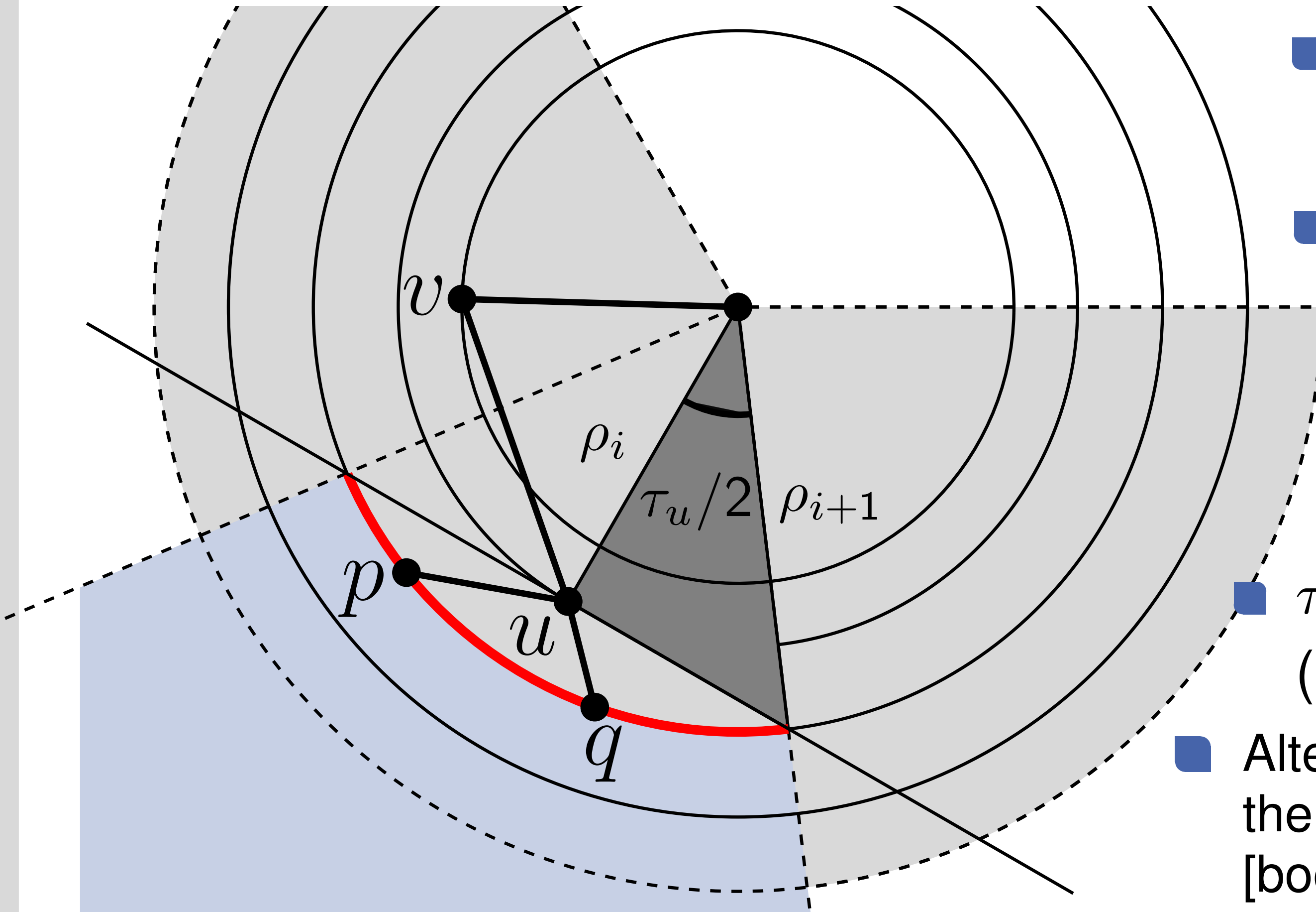
- τ_u - angle of the wedge corresponding to vertex u
- ρ_i - raduis of layer i
- $\ell(v)$-number of nodes in the subtree rooted at v
- $\cos\frac{\tau_u}{2} = \frac{\rho_i}{\rho_{i+1}}$
- $\tau_u = \min\{\frac{\ell(u)}{\ell(v)-1}, 2\arccos\frac{\rho_i}{\rho_{i+1}}\}$ (correction)
- Alternatively use number of leaves in the subtree to subdivide the angles [book Di Battista et al.]

Radial Layout

Discussion with your neighbour(s) and then share

10 min

- Why the produced drawing is planar?



- $\ell(v)$-number of nodes in the subtree rooted at v
- $\cos\frac{\tau_u}{2} = \frac{\rho_i}{\rho_{i+1}}$
- $\tau_u = \min\{\frac{\ell(u)}{\ell(v)-1}, 2\arccos\frac{\rho_i}{\rho_{i+1}}\}$ (correction)
- Alternatively use number of leaves in the subtree to subdivide the angles [book Di Battista et al.]

Radial Layout

Theorem

Let T be a rooted tree with n vertices. The radial algorithm constructs in $O(n)$ time a drawing Γ of T such that:

- Γ is planar
- Each vertex lies on the radial layer equal to its height
- The area of the drawing is at most $O(h^2 d_M^2)$, h-height, d_M-max number of children

Assuming that the radii of consecutive layers differ by the same number and the distance between the vertices on the layer is at least one

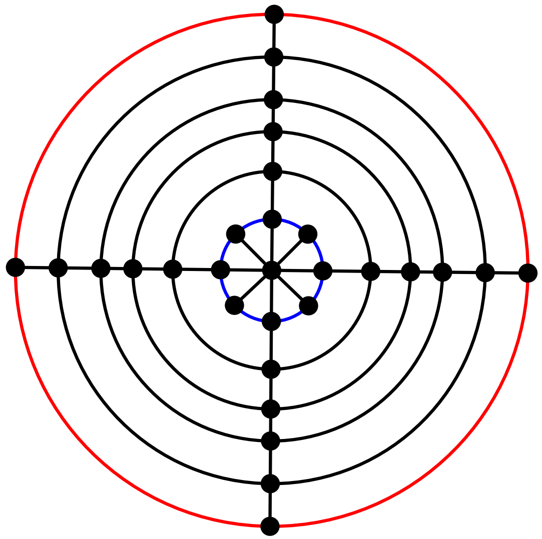
Radial Layout

Theorem

Let T be a rooted tree with n vertices. The radial algorithm constructs in $O(n)$ time a drawing Γ of T such that:

- Γ is planar
- Each vertex lies on the radial layer equal to its height
- The area of the drawing is at most $O(h^2 d_M^2)$, h-height, d_M-max number of children

Assuming that the radii of consecutive layers differ by the same number and the distance between the vertices on the layer is at least one



Radial Layout

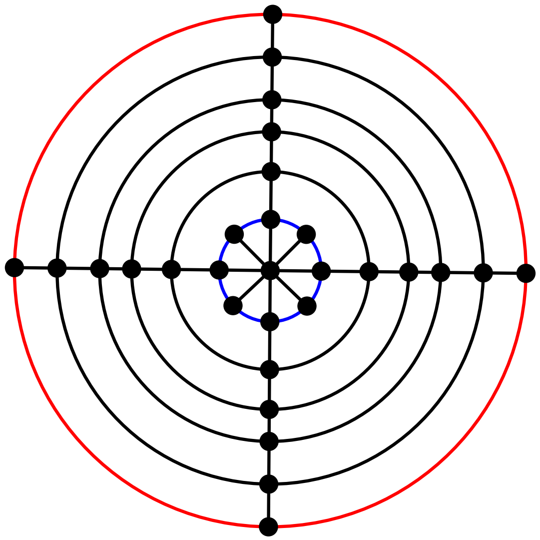
Theorem

Let T be a rooted tree with n vertices. The radial algorithm constructs in $O(n)$ time a drawing Γ of T such that:

- Γ is planar
- Each vertex lies on the radial layer equal to its height
- The area of the drawing is at most $O(h^2 d_M^2)$, h-height, d_M-max number of children

Assuming that the radii of consecutive layers differ by the same number and the distance between the vertices on the layer is at least one

radius is at least d_M
radius is at least hd_M



Reading

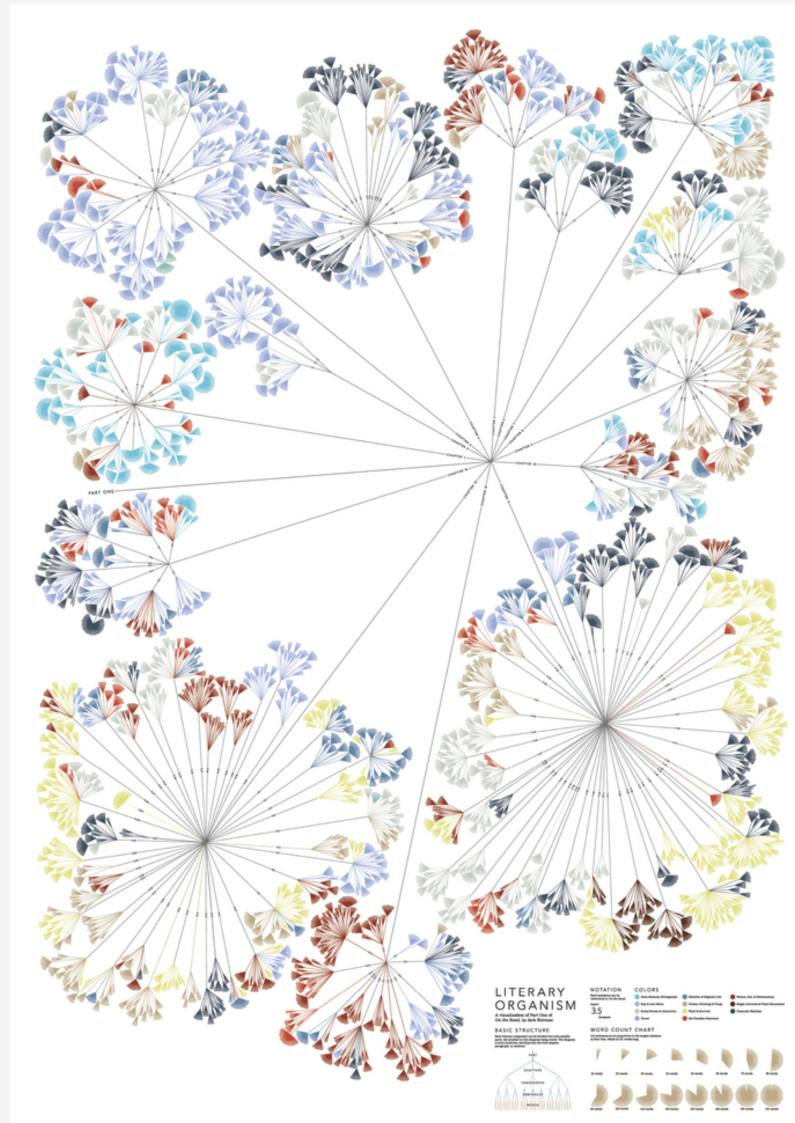
Radial Layout for Trees

- Book Di Battista et al: Chapter 3.1.3
- Skript: Chapter 6.1.2

Other Visualization Styles

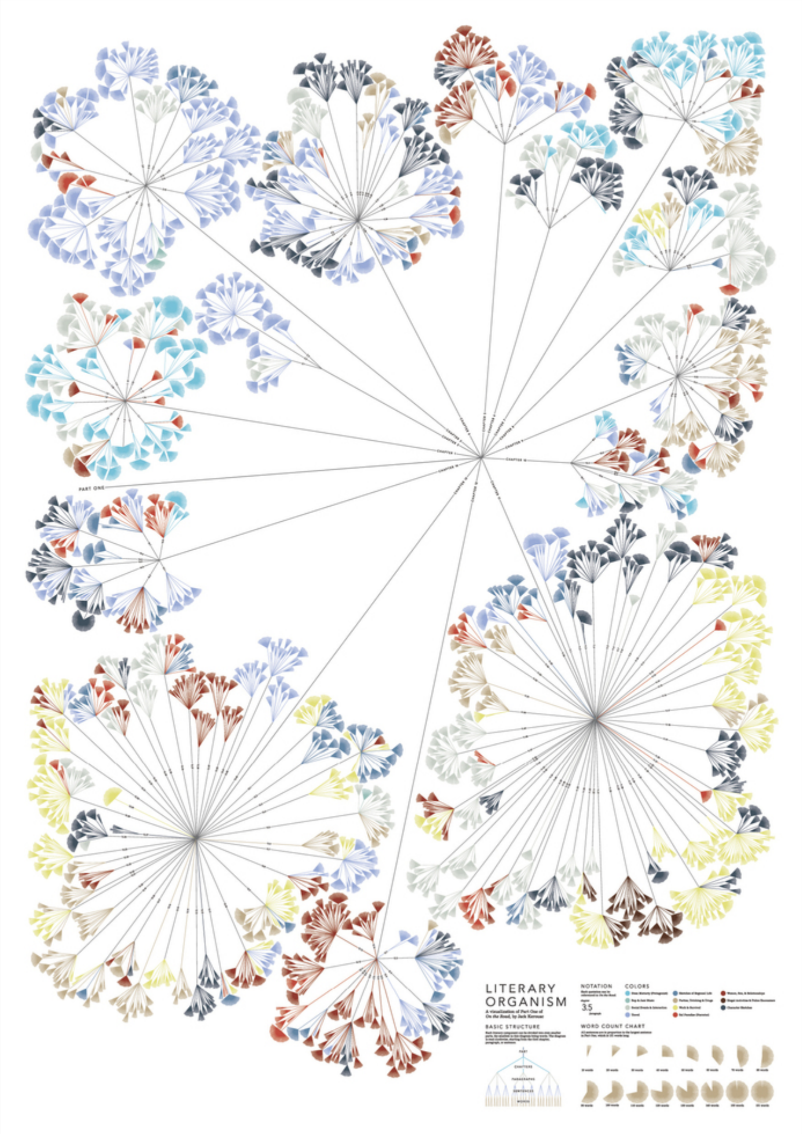
Karlsruhe Institute of Technology

Writing Without Words: the project explores methods of visually-representing text and visualises the differences in writing styles when comparing different authors.



Other Visualization Styles

Writing Without Words: the project explores methods of visually-representing text and visualises the differences in writing styles when comparing different authors.

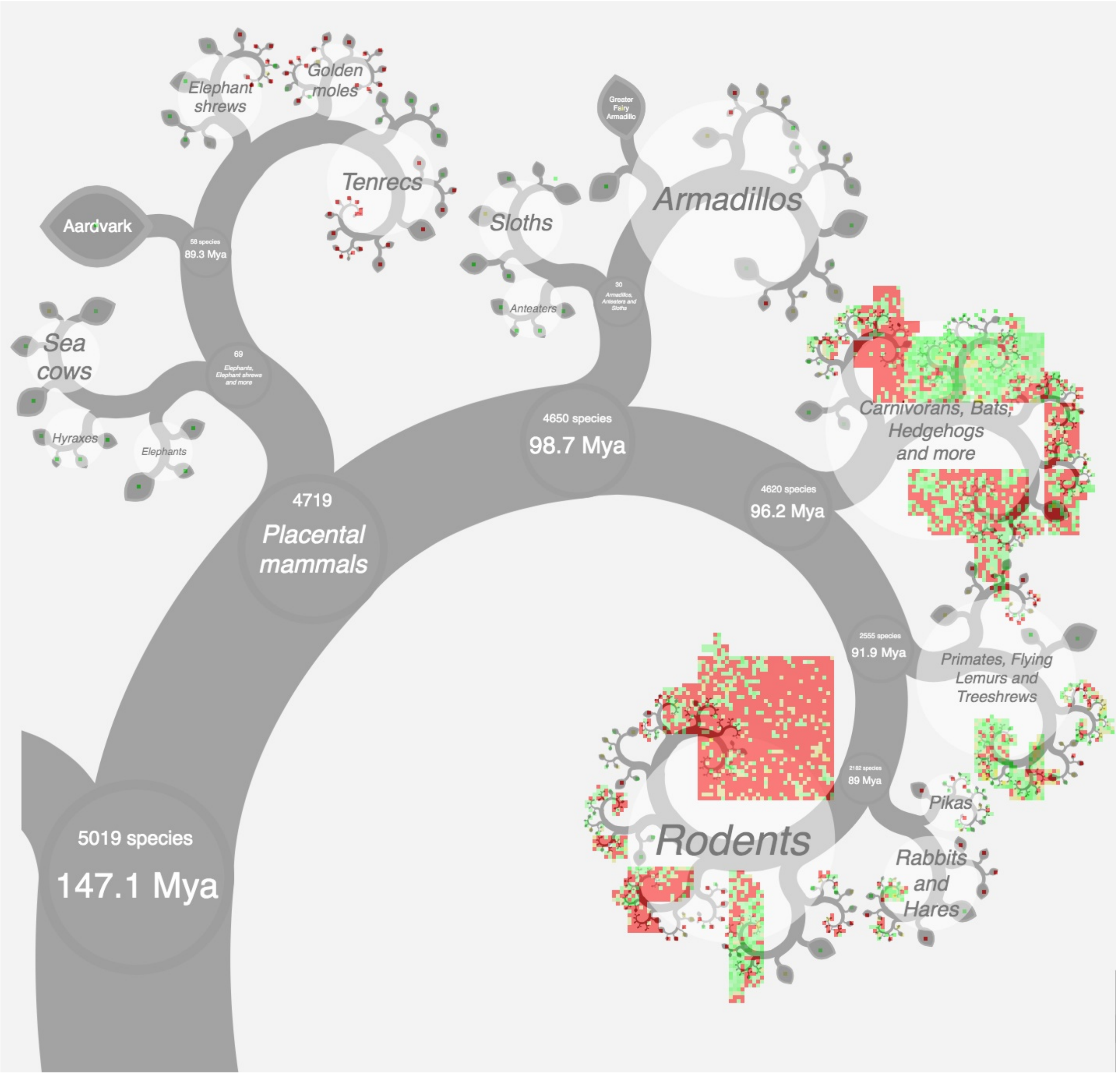


similar to Ballon layout

20 - 2

Other Visualization Styles

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



Fractal tree layout

for more applications and layouts...