Network Layout

Maneesh Agrawala

CS 448B: Visualization
Winter 2020

1

Announcements

3

Final project

New visualization research or data analysis project

- **Research**: Pose problem, Implement creative solution
- **Data analysis**: Analyze dataset in depth & make a visual explainer

Deliverables

- **Research**: Implementation of solution
- **Data analysis/explainer**: Article with multiple interactive visualizations
- 6-8 page paper

Schedule

- Project proposal: Wed 2/19
- Design review and feedback: 3/9 and 3/11
- Final presentation: 3/16 (7-9pm) Location: TBD
- Final code and writeup: 3/18 11:59pm

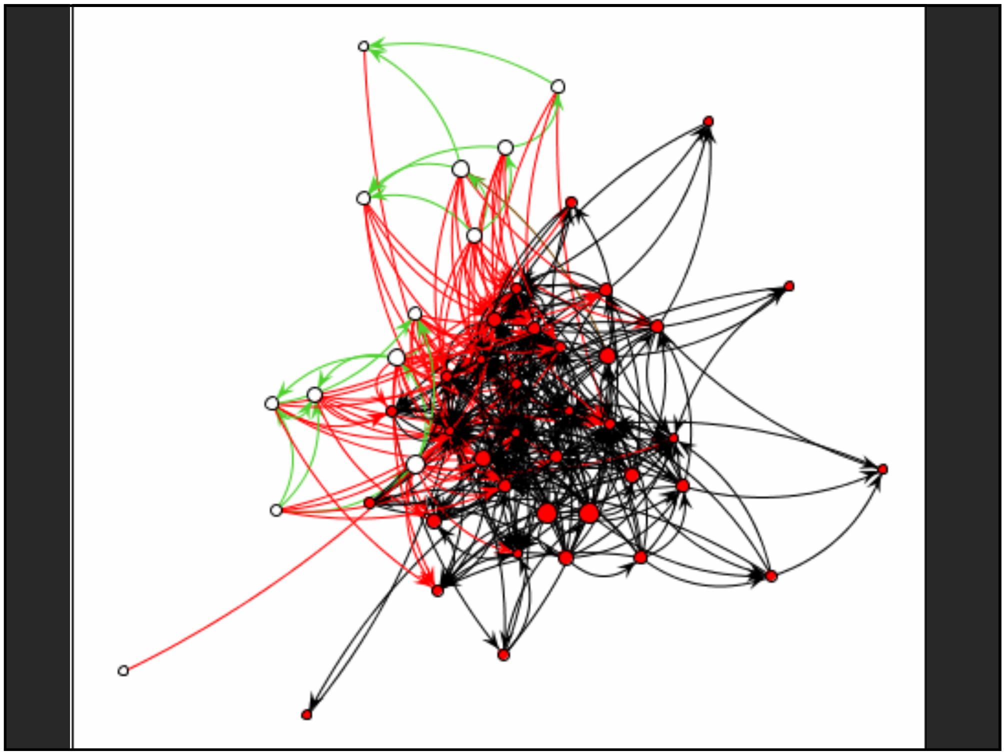
Grading

- Groups of up to 3 people, graded individually
- Clearly report responsibilities of each member

4

Network Layout

5



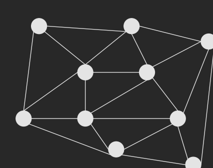
6

Graphs and Trees

Graphs

Model relations among data

Nodes and *edges*

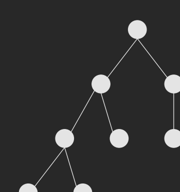


Trees

Graphs with hierarchical structure

Connected graph with N-1 edges

Nodes as *parents* and *children*



7

Spatial Layout

Primary concern – layout of nodes and edges

Often (but not always) goal is to depict structure

- Connectivity, path-following
- Network distance
- Clustering
- Ordering (e.g., hierarchy level)

8

Applications

Tournaments
Organization Charts
Genealogy
Diagramming (e.g., Visio)
Biological Interactions (Genes, Proteins)
Computer Networks
Social Networks
Simulation and Modeling
Integrated Circuit Design

9

Topics

Tree Layout
Network Layout
- Sugiyama-Style Layout
- Force-Directed Layout

Alternatives to Network Layout
- Matrix Diagrams
- Attribute-Drive Layout

10

Tree Layout

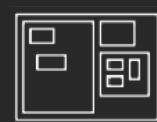
11

Tree Visualization

Indentation
- Linear list, indentation encodes depth

Node-Link diagrams
- Nodes connected by lines/curves

Enclosure diagrams
- Represent hierarchy by enclosure

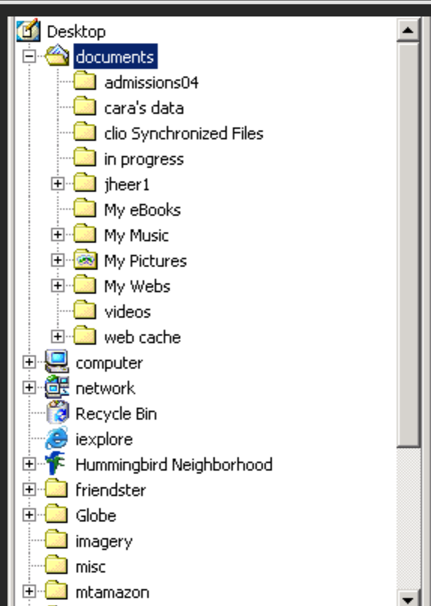


Layering
- Layering and alignment

Tree layout is fast: O(n) or O(n log n), enabling real-time layout for interaction

12

Indentation



Items along vertically spaced rows

Indentation shows parent/child relationships

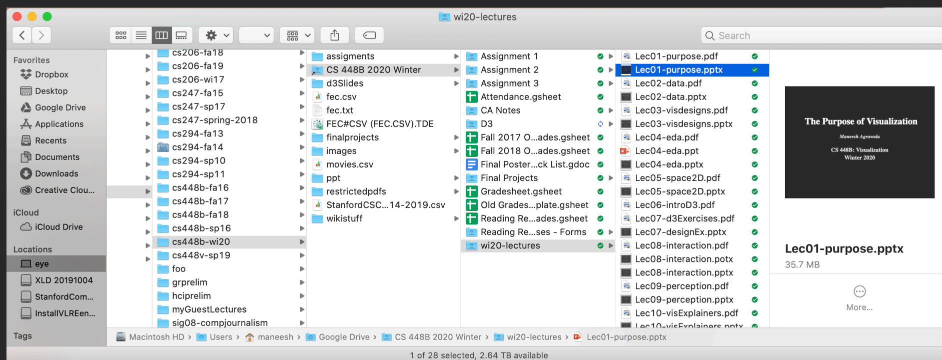
Often used in interfaces

Breadth/depth contend for space

Often requires scrolling

13

Single-Focus (Accordion) List



Separate breadth & depth in 2D
Focus on single path at a time

14

Node-Link Diagrams

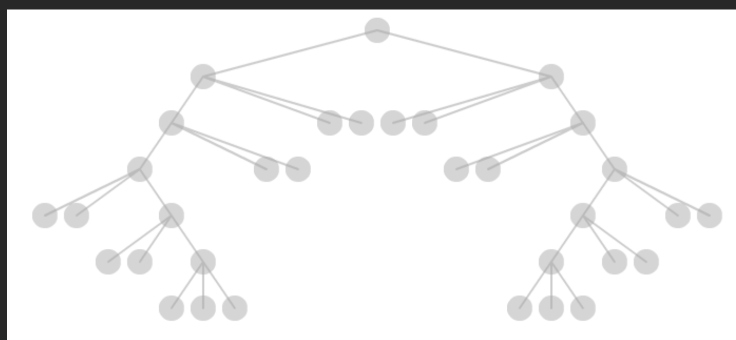
Nodes distributed in space, connected by lines
Use 2D space to break apart breadth and depth
Space used to communicate hierarchical orientation
- Typically towards authority or generality

15

Basic Recursive Approach

Repeatedly divide space for subtrees by leaf count

- Breadth of tree along one dimension
- Depth along the other dimension



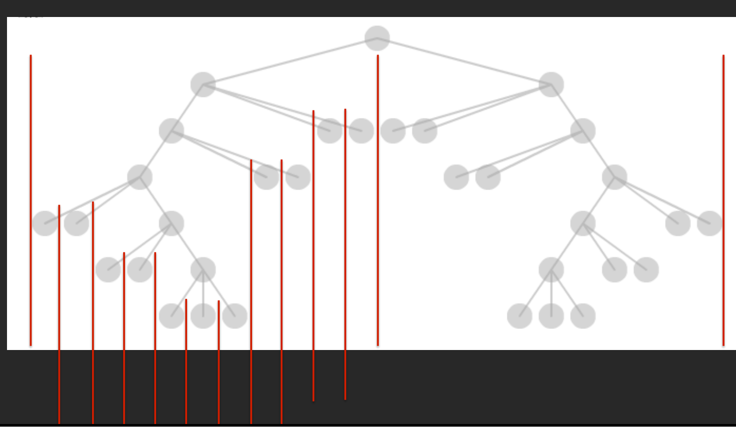
17

Basic Recursive Approach

Repeatedly divide space for subtrees by leaf count

- Breadth of tree along one dimension
- Depth along the other dimension

Problem:



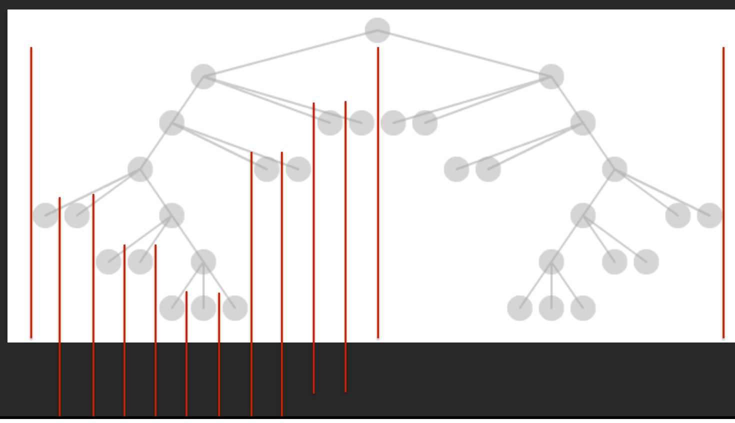
18

Basic Recursive Approach

Repeatedly divide space for subtrees by leaf count

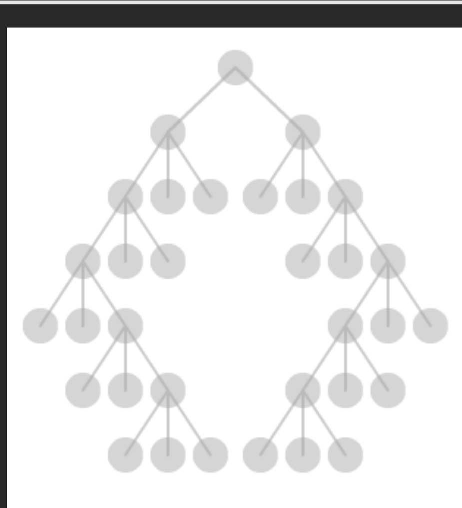
- Breadth of tree along one dimension
- Depth along the other dimension

Problem: Exponential growth of breadth



19

Reingold & Tilford's Tidier Layout



Goal: maximize density and symmetry.

Originally for binary trees, extended by Walker to cover general case.

This extension was corrected by Buchheim et al. to achieve a linear time algorithm

20

Reingold-Tilford Layout

Design concerns

- Clearly encode depth level
- No edge crossings
- Isomorphic subtrees drawn identically
- Ordering and symmetry preserved
- *Compact layout (don't waste space)*

21

Reingold-Tilford Algorithm

Linear algorithm – starts with bottom-up (postorder) pass
Set Y-coord by depth, arbitrary starting X-coord
Merge left and right subtrees

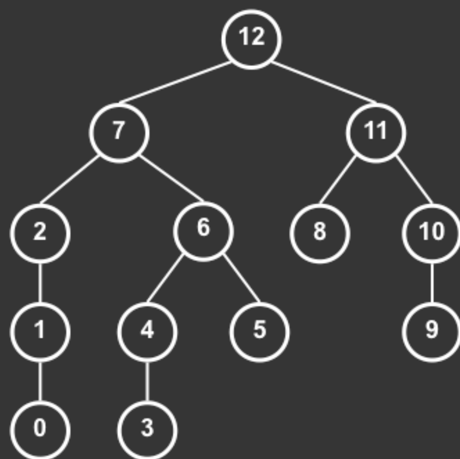
- Shift right as close as possible to left
  - Computed efficiently by maintaining subtree contours
- "Shifts" in position saved for each node as visited
- Parent nodes are centered above their children

Top-down (preorder) pass for assignment of final positions

- Sum of initial layout and aggregated shifts

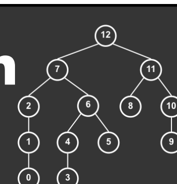
22

Reingold-Tilford Algorithm

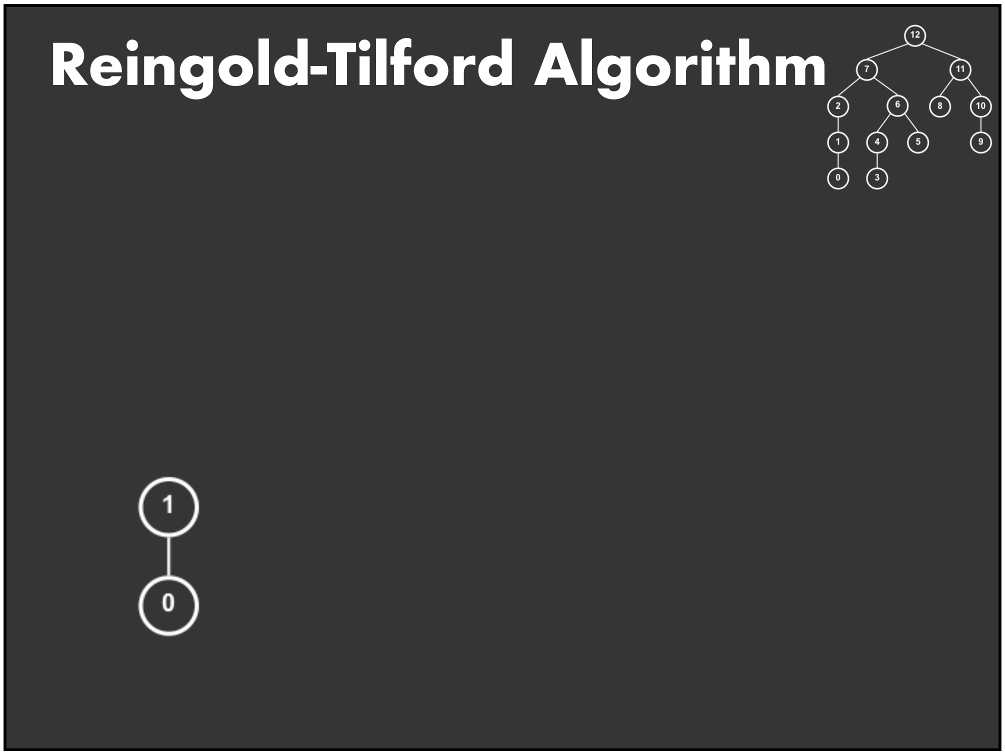


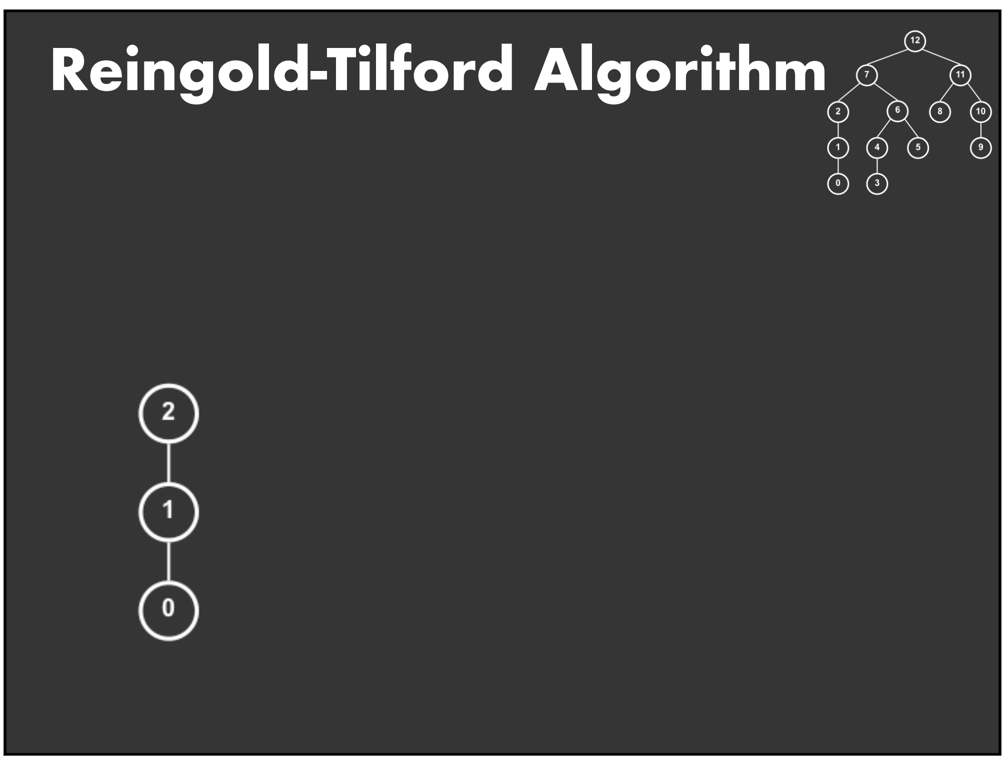
23

Reingold-Tilford Algorithm

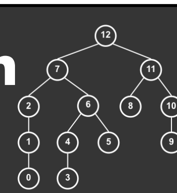


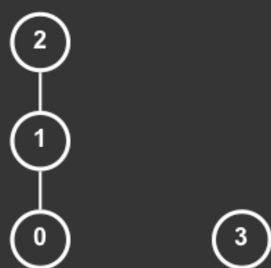
24


Reingold-Tilford Algorithm


Reingold-Tilford Algorithm

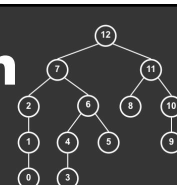
Reingold-Tilford Algorithm

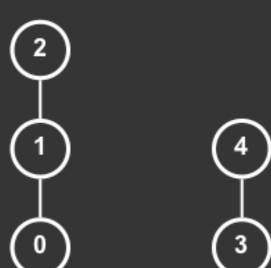




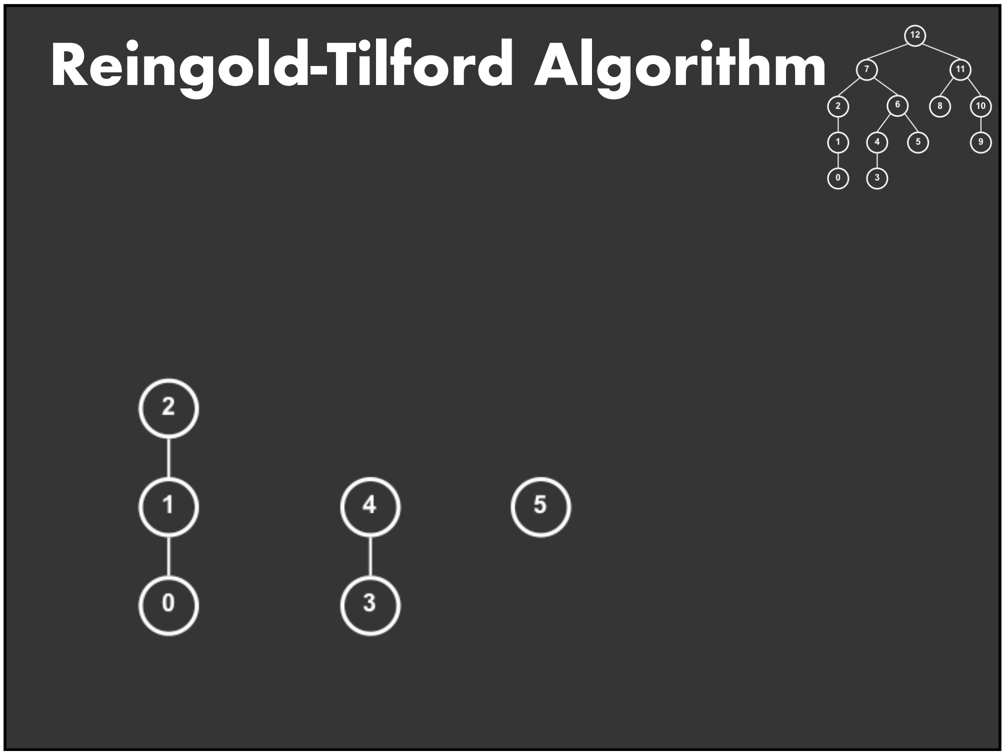
27

Reingold-Tilford Algorithm

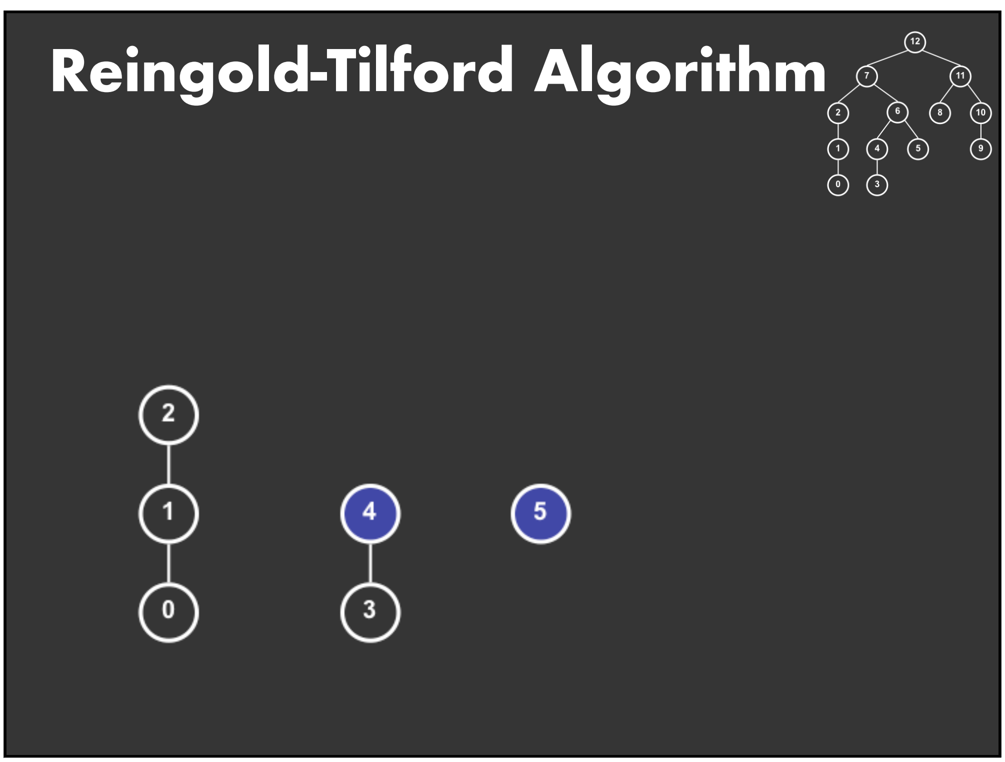




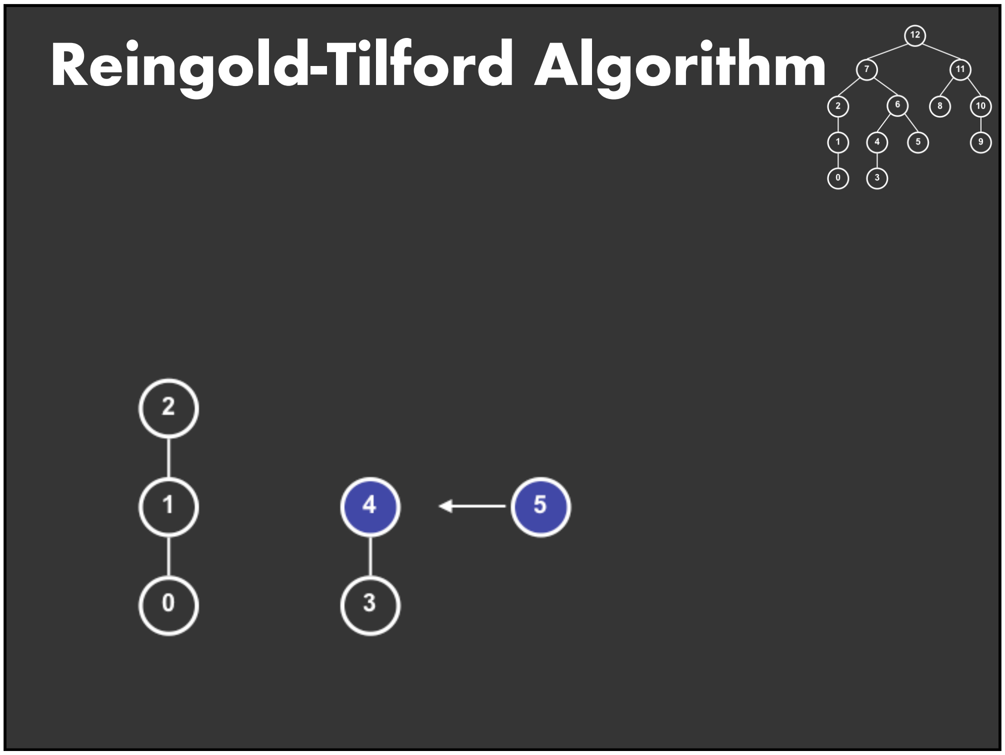
28


Reingold-Tilford Algorithm
2
1
0
4
3
5

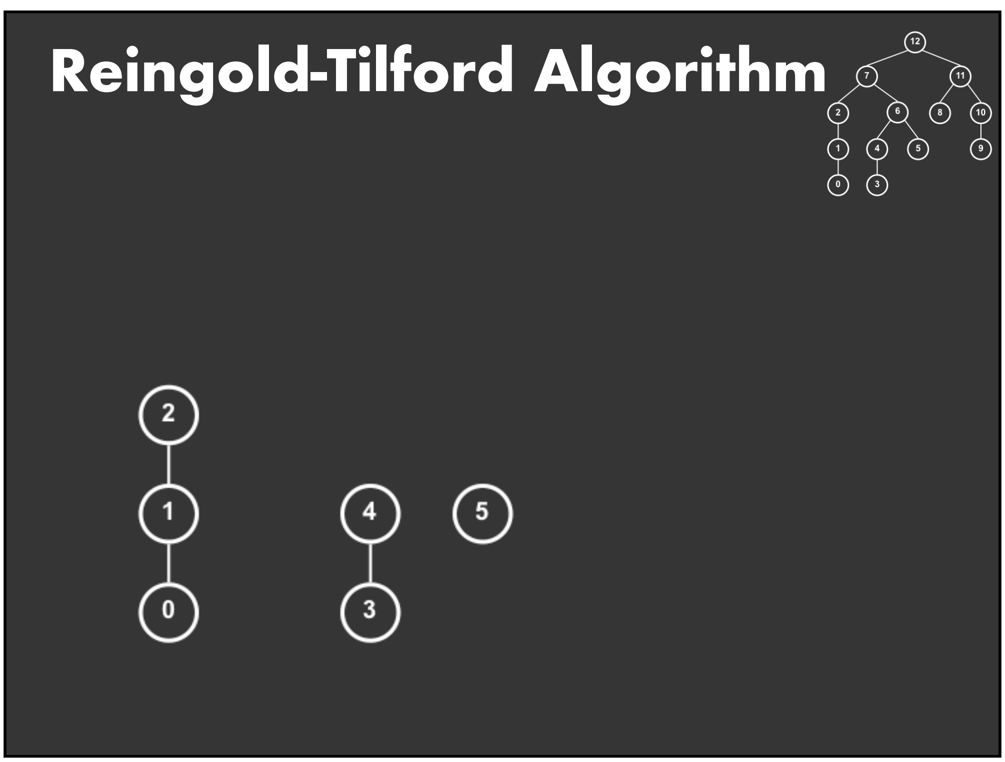
29


Reingold-Tilford Algorithm
2
1
0
4
3
5

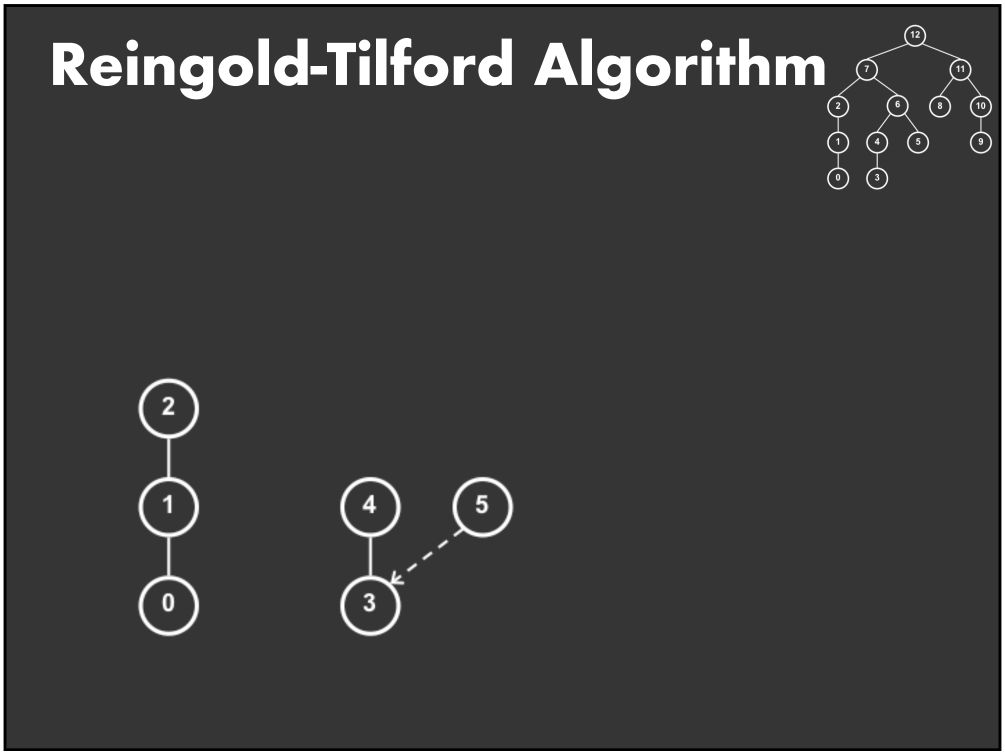
30


Reingold-Tilford Algorithm
2
1
0
4
3
5

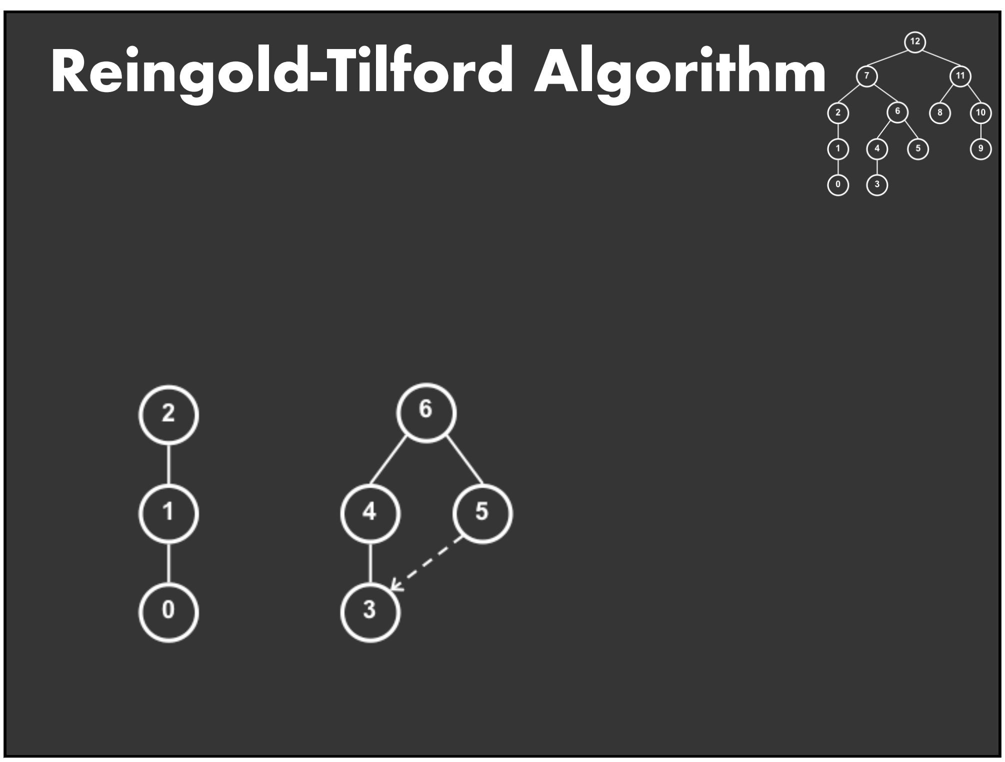
31


Reingold-Tilford Algorithm
2
1
0
4
3
5

32

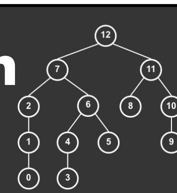

Reingold-Tilford Algorithm
12
7
11
2
6
8
10
1
4
5
9
0
3
2
1
0
4
5
3

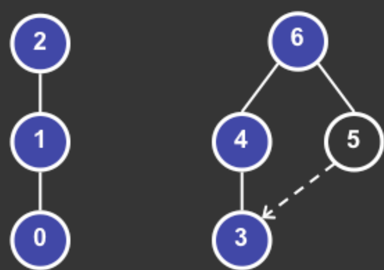
33


Reingold-Tilford Algorithm
12
7
11
2
6
8
10
1
4
5
9
0
3
2
1
0
6
4
5
3

34

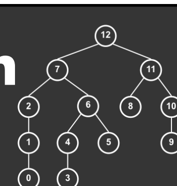
Reingold-Tilford Algorithm

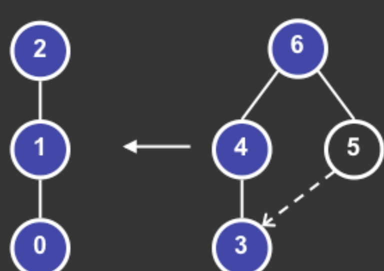




35

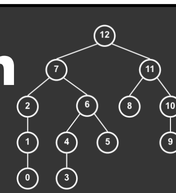
Reingold-Tilford Algorithm

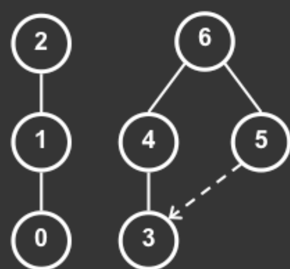




36

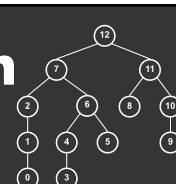
Reingold-Tilford Algorithm

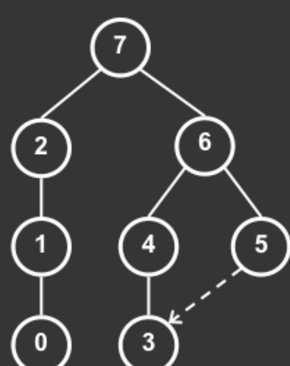




37

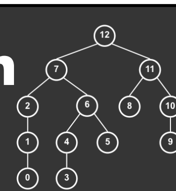
Reingold-Tilford Algorithm

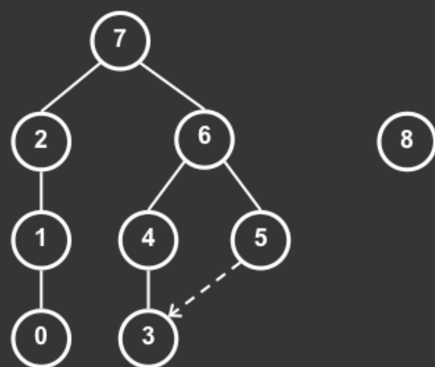




38

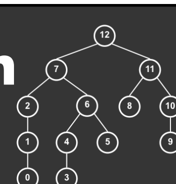
Reingold-Tilford Algorithm

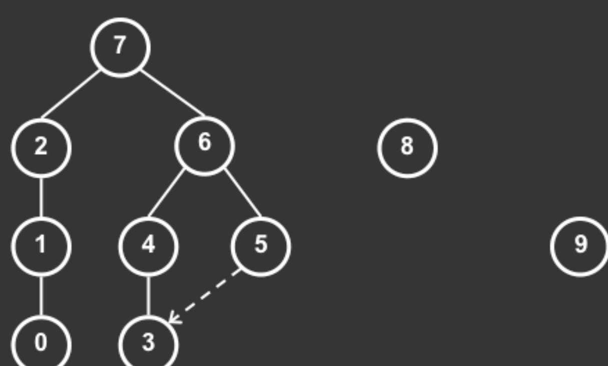




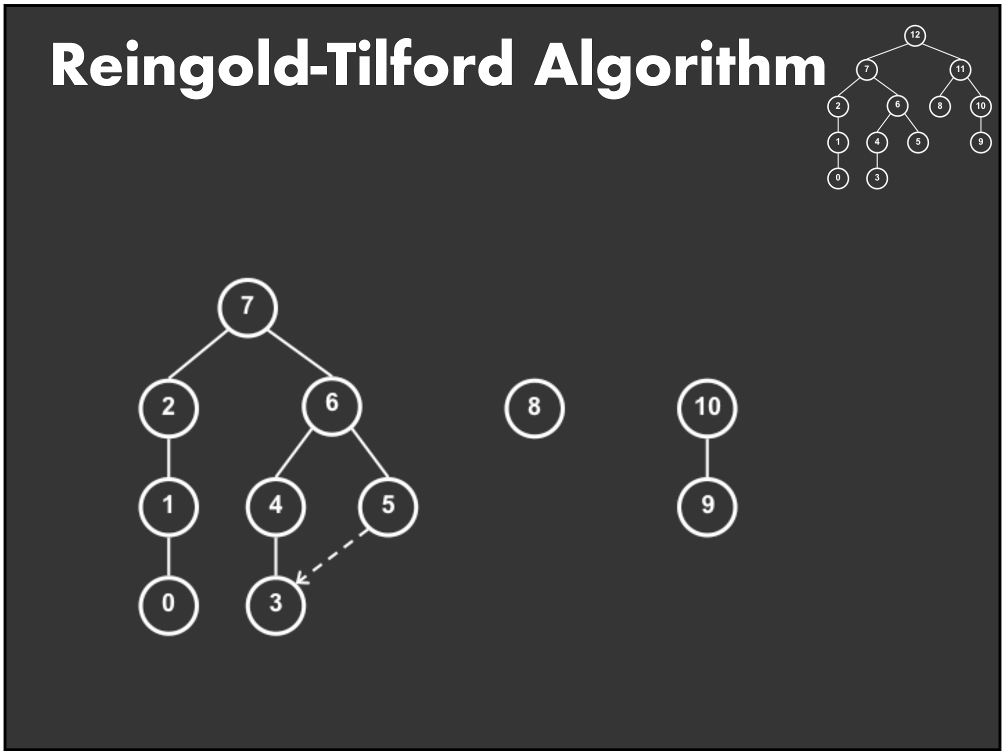
39

Reingold-Tilford Algorithm

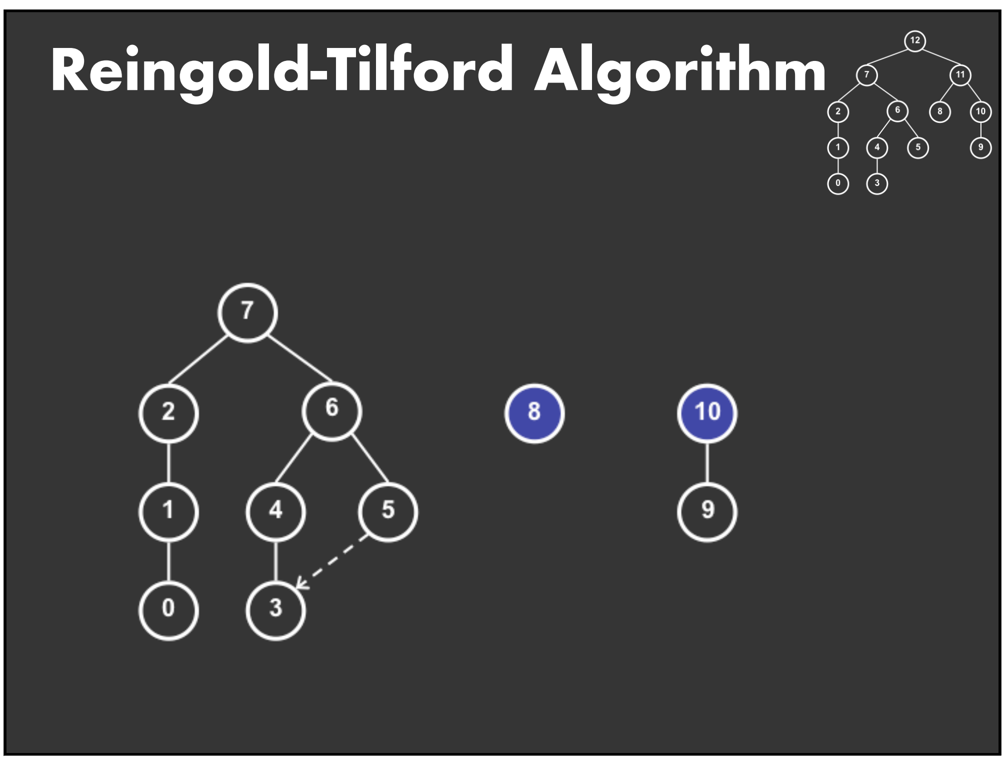




40

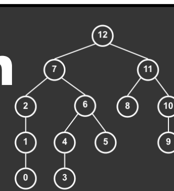

Reingold-Tilford Algorithm
12
7
11
2
6
8
10
1
4
5
9
0
3

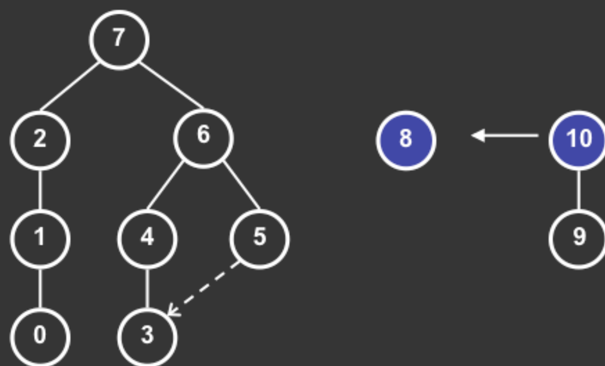
41


Reingold-Tilford Algorithm
12
7
11
2
6
8
10
1
4
5
9
0
3

42

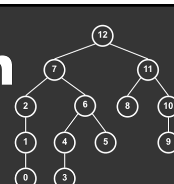
Reingold-Tilford Algorithm

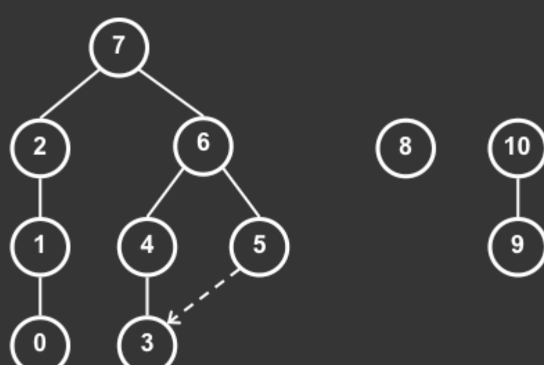




43

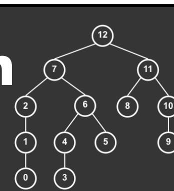
Reingold-Tilford Algorithm

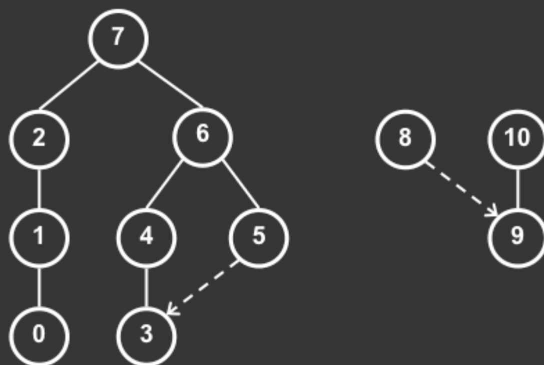




44

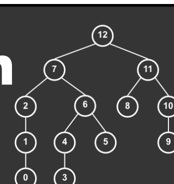
Reingold-Tilford Algorithm

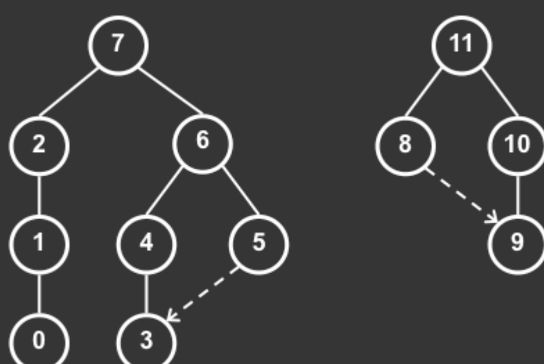




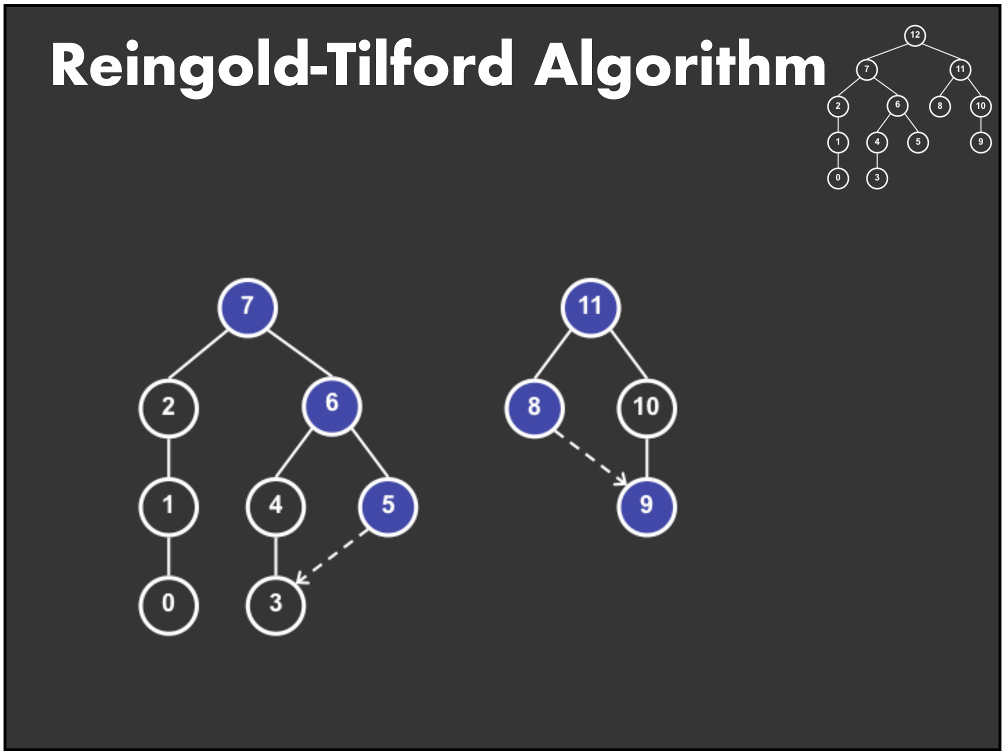
45

Reingold-Tilford Algorithm

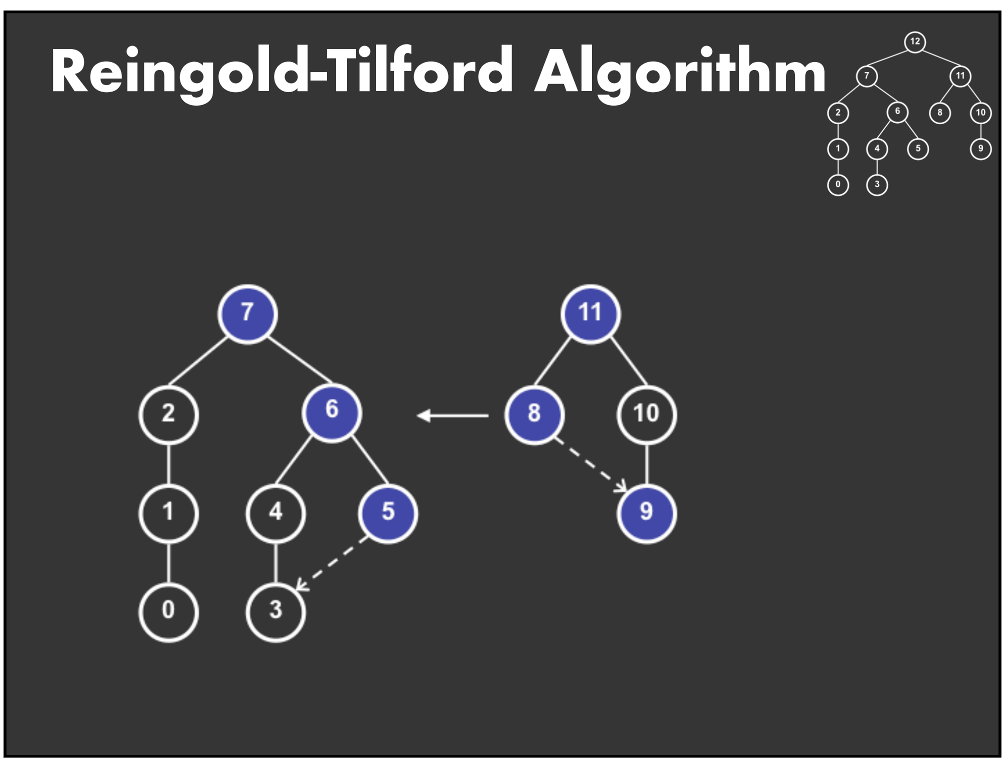




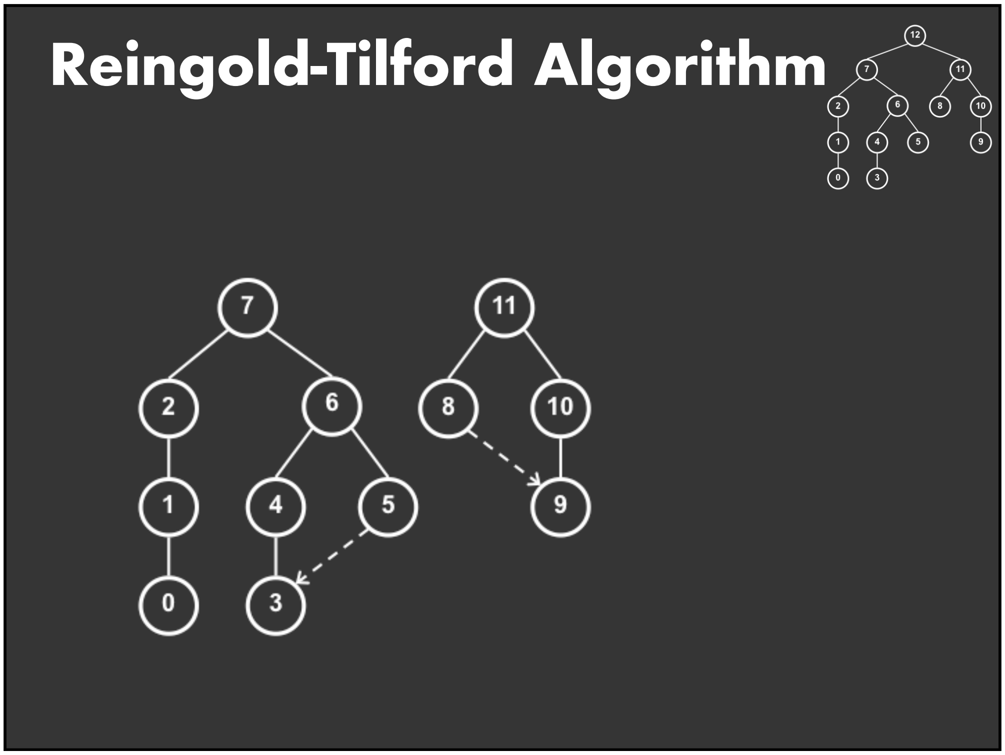
46


Reingold-Tilford Algorithm
12
7
11
2
6
8
10
1
4
5
9
0
3

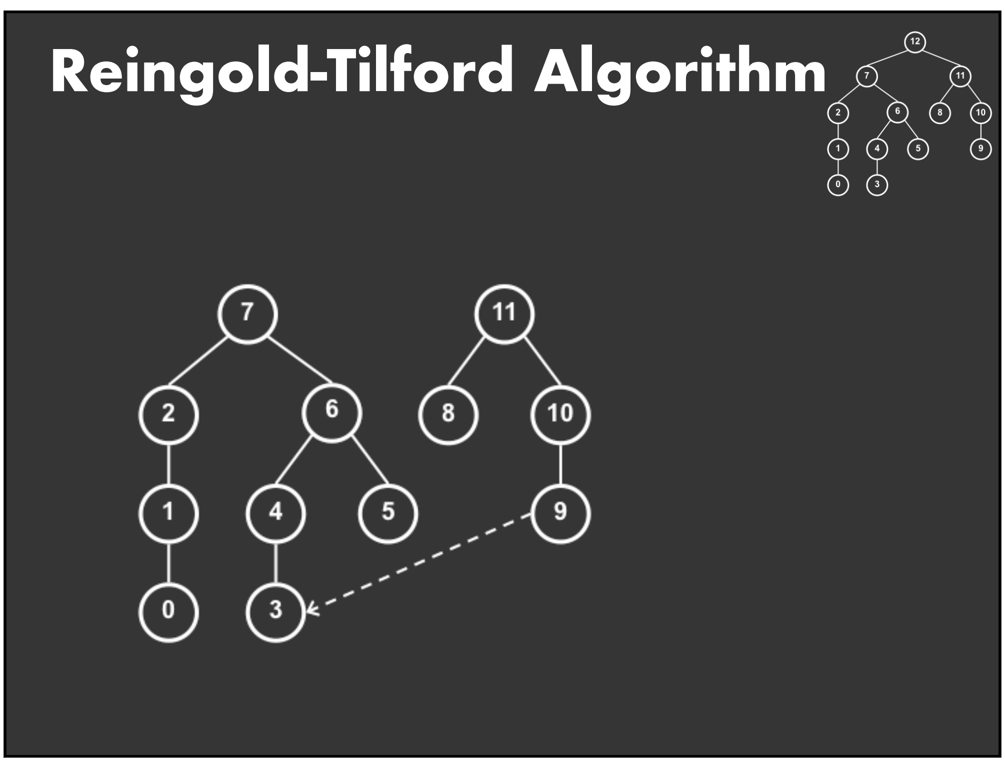
47


Reingold-Tilford Algorithm
12
7
11
2
6
8
10
1
4
5
9
0
3

48

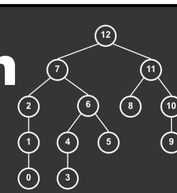

Reingold-Tilford Algorithm
12
7
11
2
6
8
10
1
4
5
9
0
3

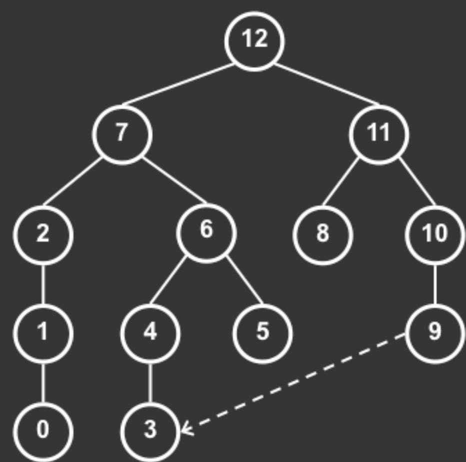
49


Reingold-Tilford Algorithm
12
7
11
2
6
8
10
1
4
5
9
0
3

50

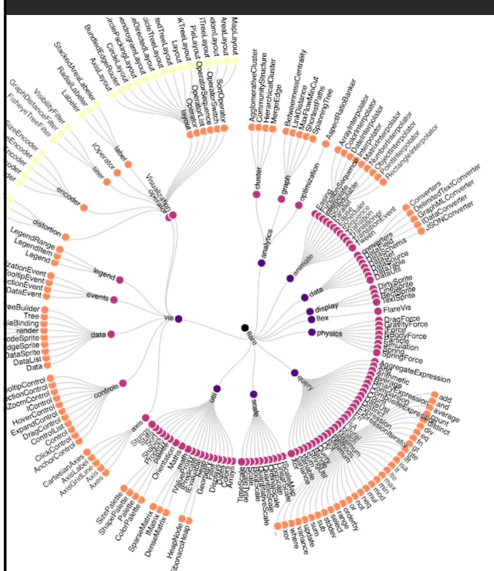
Reingold-Tilford Algorithm





51

Radial Layout



- Node-link diagram in polar coords
- Radius encodes depth root at center
- Angular sectors assigned to subtrees (recursive approach)
- Reingold-Tilford approach can also be applied here

54

Problems with Node-Link Diagrams

Scale

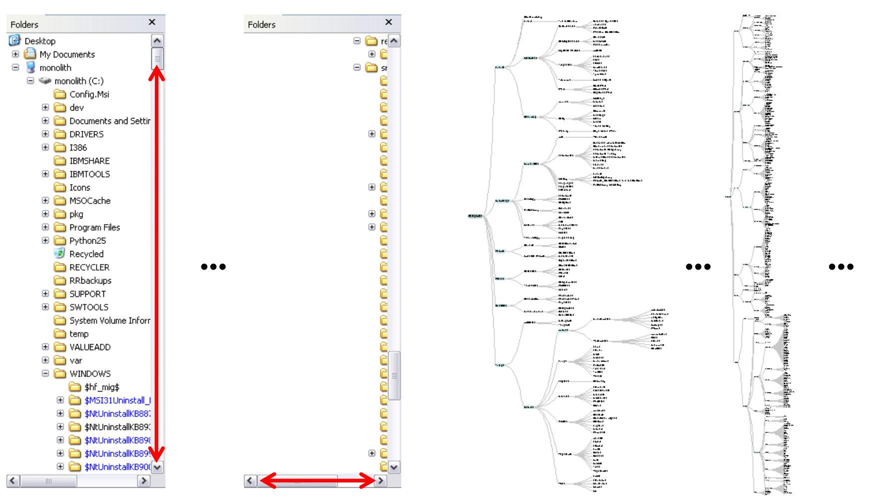
Tree breadth often grows exponentially
Even with tidier layout, quickly run out of space

Possible solutions

Filtering
Focus+Context
Scrolling or Panning
Zooming
Aggregation

57

Visualizing Large Hierarchies



Indented Layout

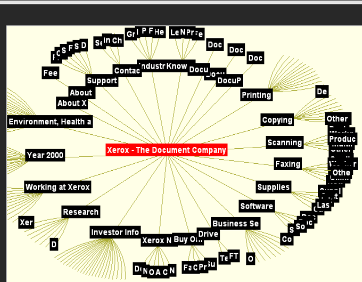
Reingold-Tilford Layout

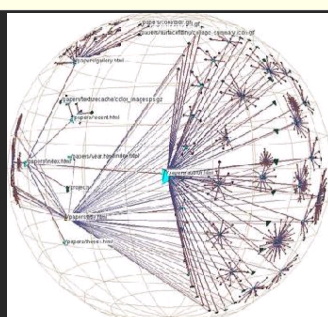
58

MC Escher, *Circle Limit IV*

59

Hyperbolic Layout





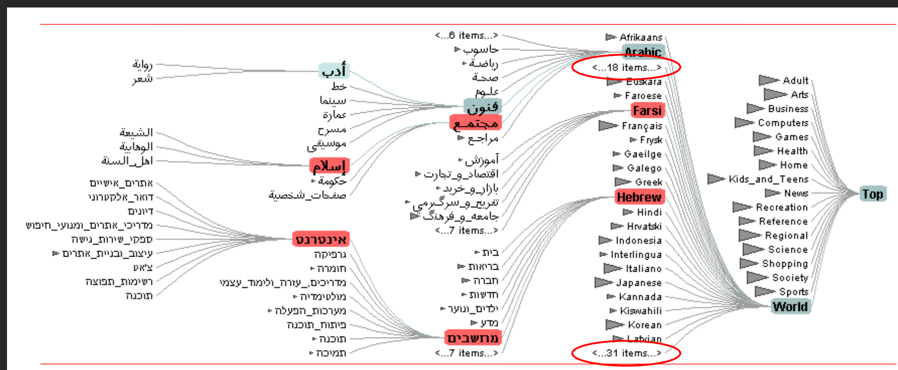
Layout in hyperbolic space, then project on to Euclidean plane

Why? Like tree breadth, the hyperbolic plane expands exponentially

Also computable in 3D, projected into a sphere

60

Degree-of-Interest Trees [AVI 04]

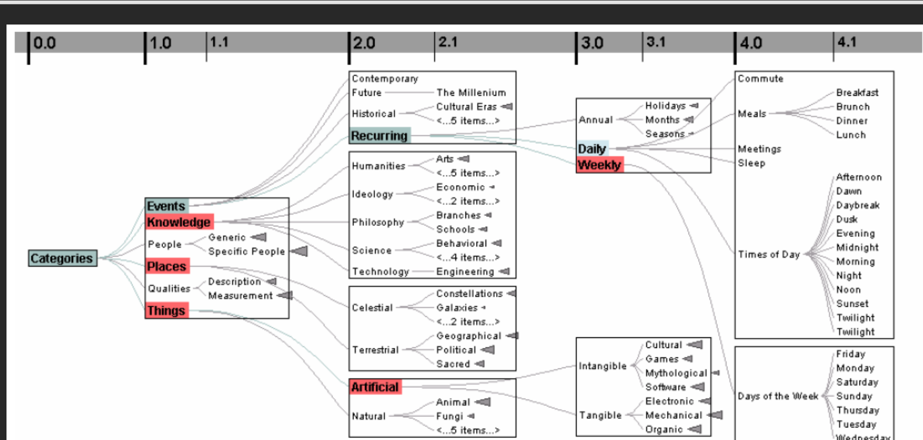


Space-constrained, multi-focal tree layout

https://www.youtube.com/watch?v=RTQ0N4QY0yc
https://observablehq.com/@d3/collapsible-tree

61

Degree-of-Interest Trees



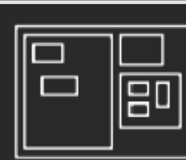
Cull "un-interesting" nodes on a per block basis until all blocks on a level fit within bounds

Center child blocks under parents

https://www.youtube.com/watch?v=RTQ0N4QY0yc
https://observablehq.com/@d3/collapsible-tree

62

Enclosure Diagrams



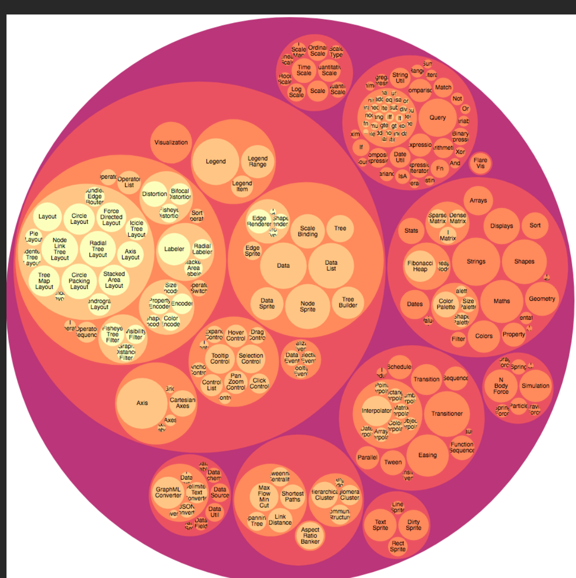
Encode structure using spatial enclosure
Popularly known as TreeMaps

Benefits
- Provides a single view of an entire tree
- Easier to spot large/small nodes

Problems
- Difficult to accurately read depth

63

Circle Packing Layout



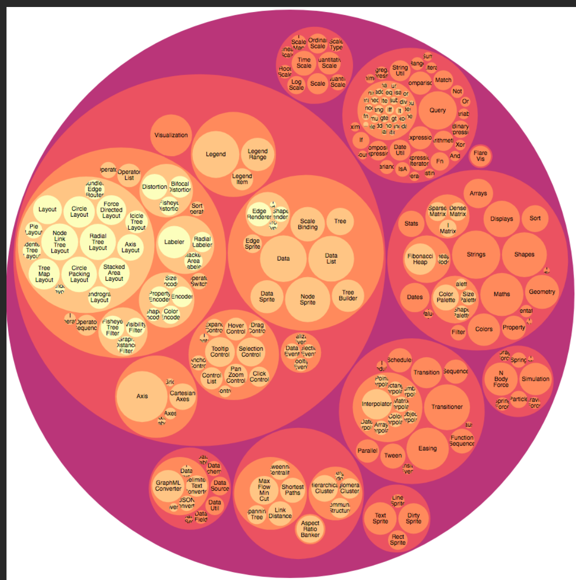
Nodes represented as sized circles

Nesting to show parent-child relationships

Problems:

64

Circle Packing Layout



Nodes represented as sized circles

Nesting to show parent-child relationships

Problems:

- Inefficient use of space
- Parent size misleading

65

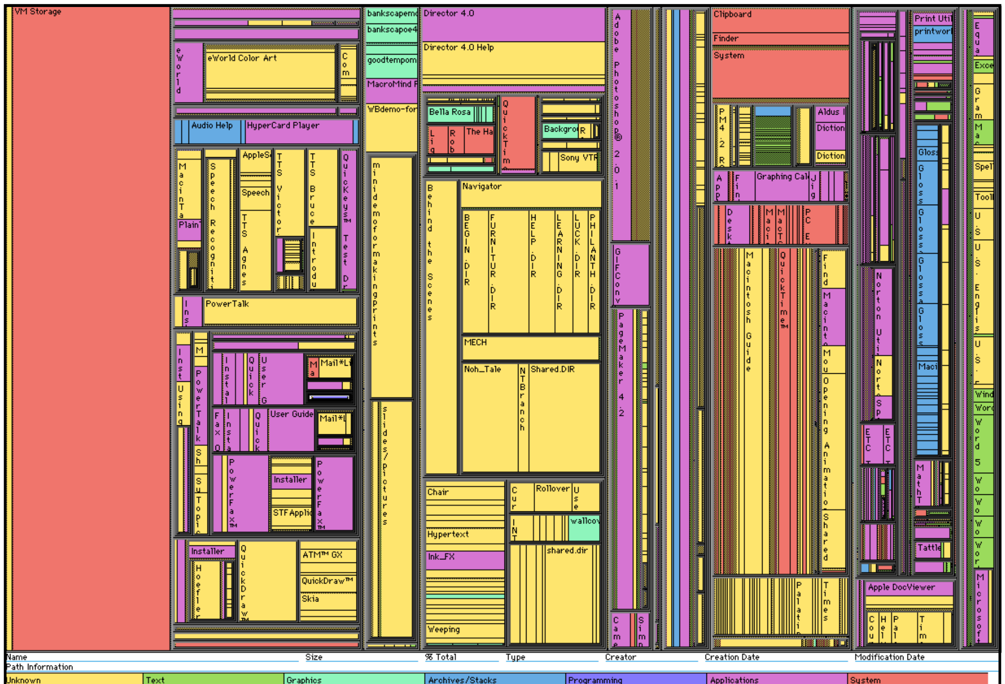
Treemaps

Hierarchy visualization that emphasizes values of nodes via area encoding

Partition 2D space such that leaf nodes have sizes proportional to data values

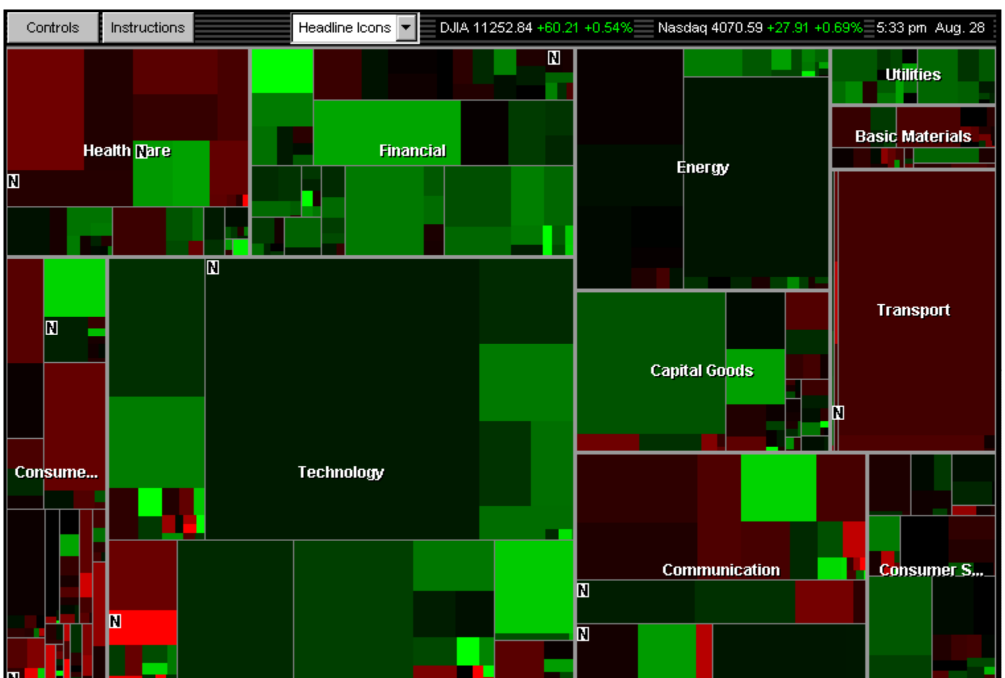
First layout algorithms proposed by Shneiderman et al. in 1990, with focus on showing file sizes on a hard drive

67



Slice & Dice **layout: Alternate horizontal / vertical partitions.**

68

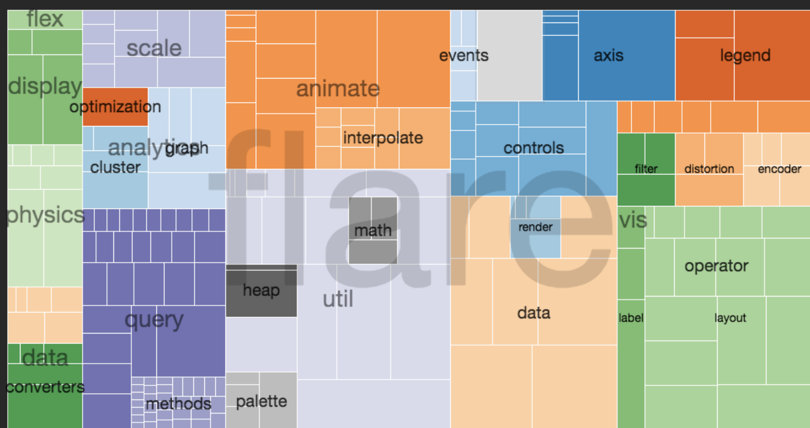


Squarifed **layout: Try to produce square (1:1) aspect ratios**

69

Squarified Treemaps [Bruls 00]

Greedy optimization for objective of square rectangles
Slice/dice within siblings; alternate whenever ratio worsens



https://vega.github.io/vega/examples/treemap/

70

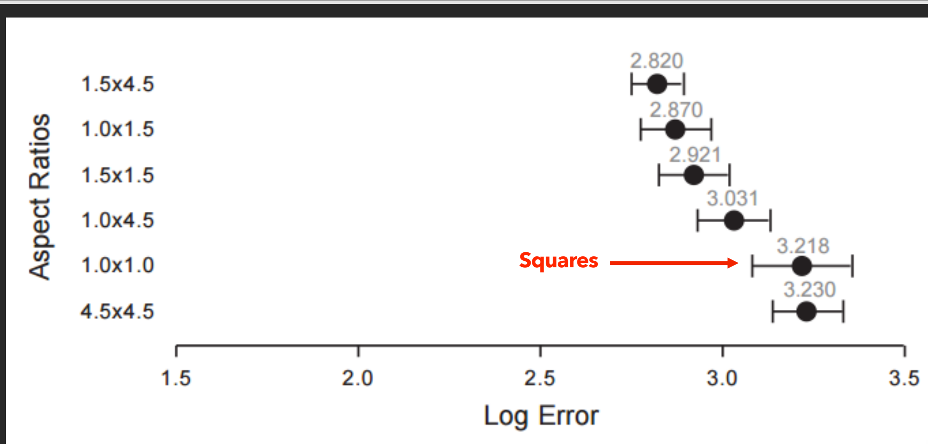
Why Squares

Posited Benefits of 1:1 Aspect Ratios

1. Minimize perimeter, reducing border ink.
2. Easier to select with a mouse cursor.
   Validated by empirical research & Fitt's Law!
3. Similar aspect ratios are easier to compare.
   Seems intuitive, but is this true?

71

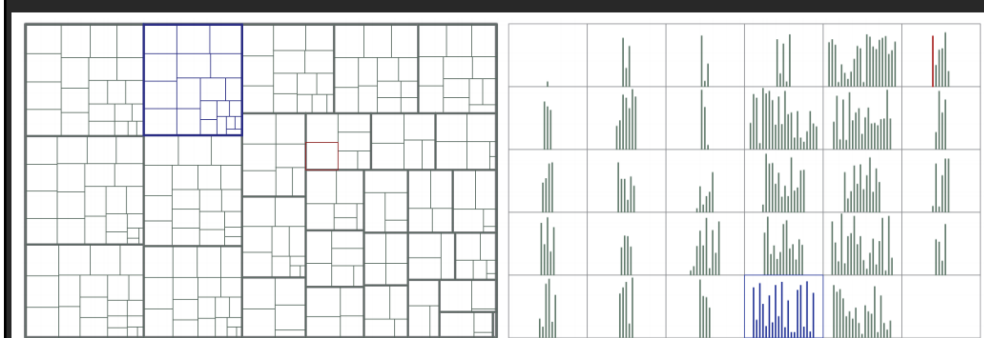
Error vs. Aspect Ratio [Kong 10]



1. Comparison of squares has higher error!
2. Squarify works because it fails to meet its objective?

72

Treemaps vs. Bar Charts [Kong 10]

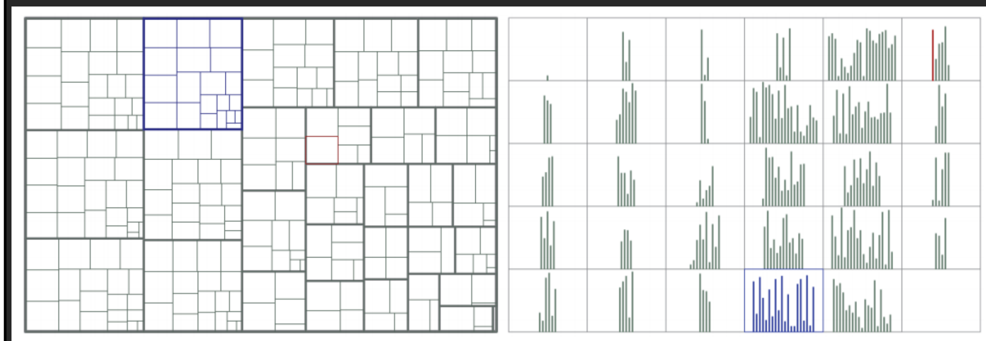


Height more perceptually effective than area

What if element count is high?
What about comparing groups of elements such as leaf values to internal node values?

74

Treemaps vs. Bar Charts [Kong 10]



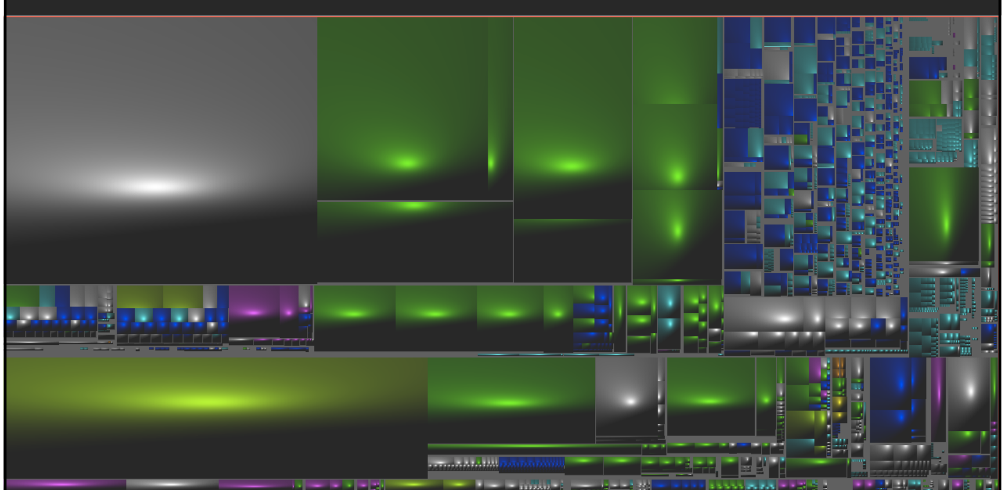
At low densities (< 4k elements), bar charts more accurate than treemaps for leaf-node comparisons.

At higher density, treemaps led to faster judgments.

Treemaps better for group-level comparisons.

75

Cushion Treemaps [van Wijk 99]



Use shading to emphasize hierarchical structure

76

Cascaded Treemaps [Lü 08]

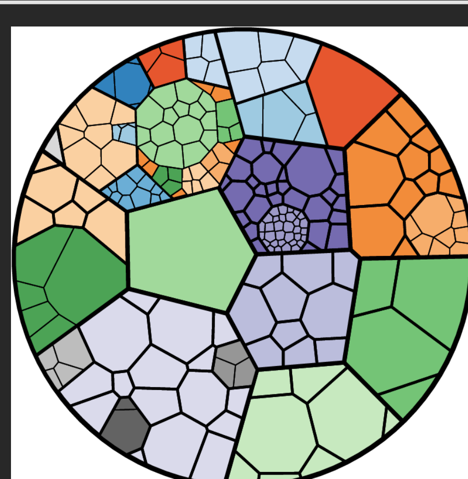
Use 2.5D effect emphasize hierarchical structure

77

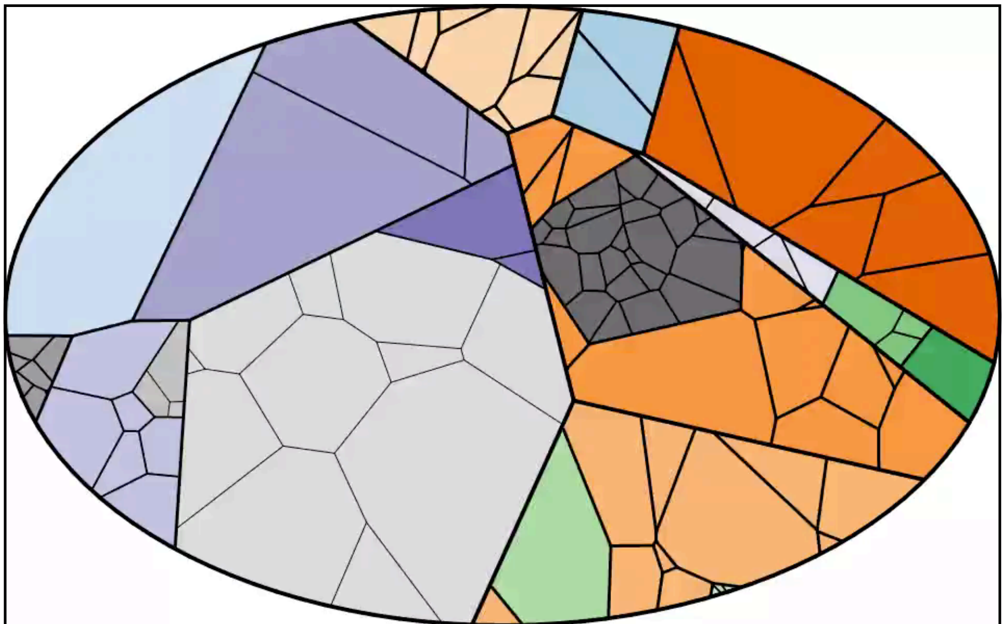
Voronoi Treemaps [Balzer 05]

Treemaps with arbitrary polygonal shape and boundary

Uses iterative, eighted Voronoi tessellations to achieve cells with value-proportional areas



78



Iterative Voronoi Tesselations [Jason Davies]

79

Layered Diagrams

Signify tree structure using

Layering

Adjacency

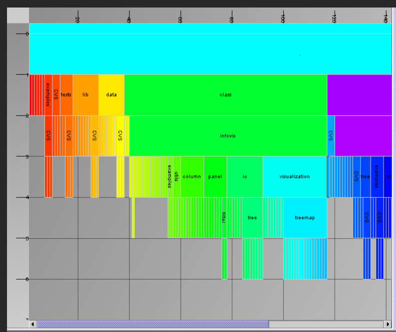
Alignment

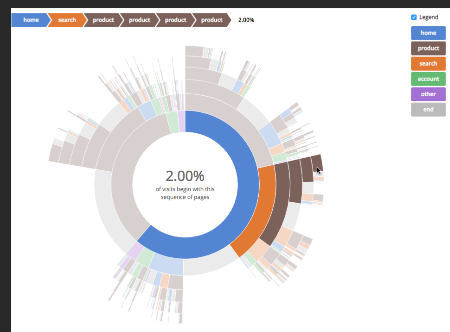
Involves recursive sub-division of space

Can apply the same set of approaches as in node-link layout

80

Icicle and Sunburst Trees



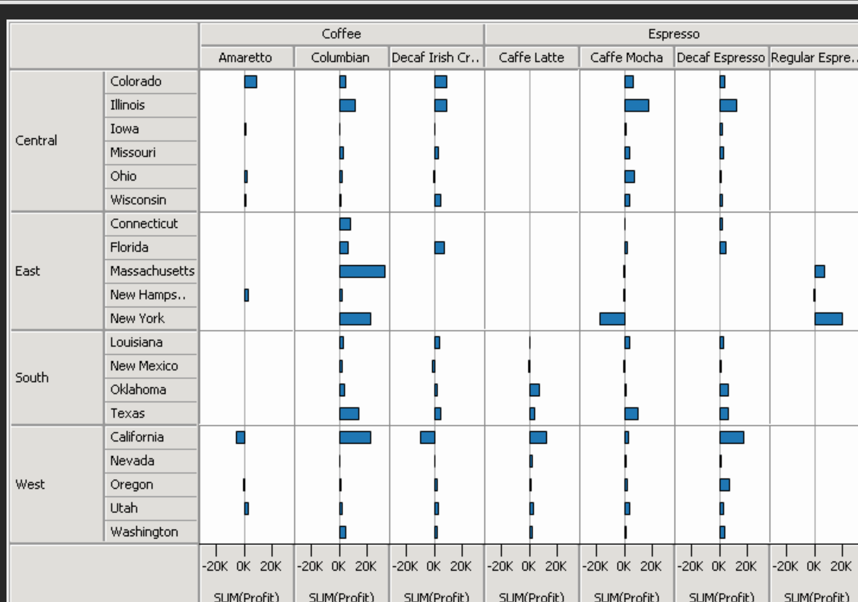


Higher-level nodes get a larger layer area, whether that is horizontal or angular extent

Child levels are layered, constrained to parent’s extent

81

Layered Tree Drawing



82

Node-Link Graph Layout

84

Spanning Tree Layout

Many graphs are tree-like or have useful spanning trees

- Websites, Social Networks

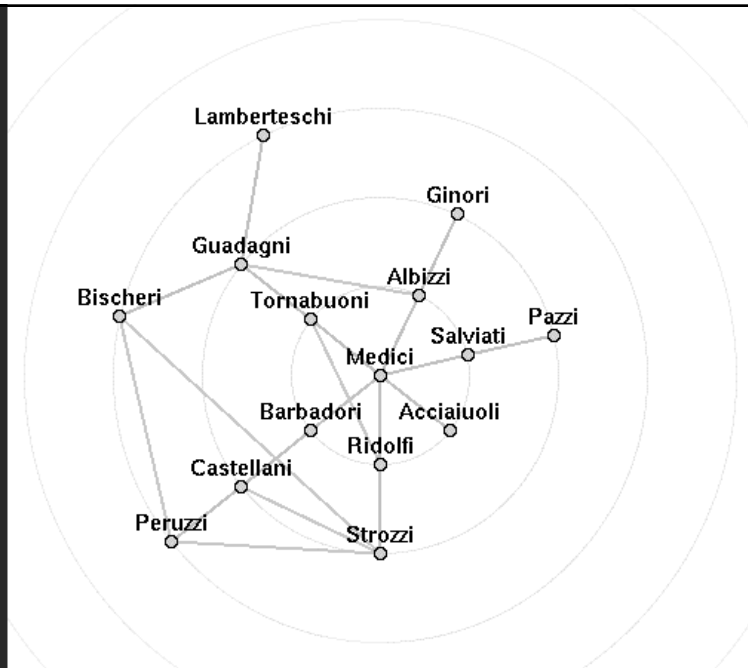
Use tree layout on spanning tree of graph

- Trees created by BFS / DFS
- Min/max spanning trees

Fast tree layouts allow graph layouts to be recalculated at interactive rates

Heuristics may further improve layout

86



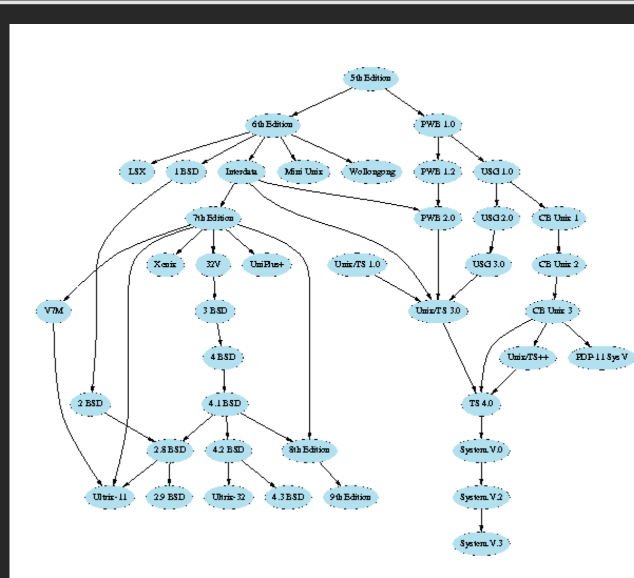
Spanning tree layout may result in arbitrary parent node

87

Sugiyama-style graph layout

Evolution of the UNIX operating system

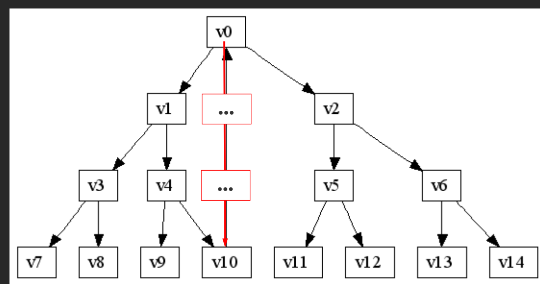
Hierarchical layering based on descent



88

Sugiyama-style graph layout

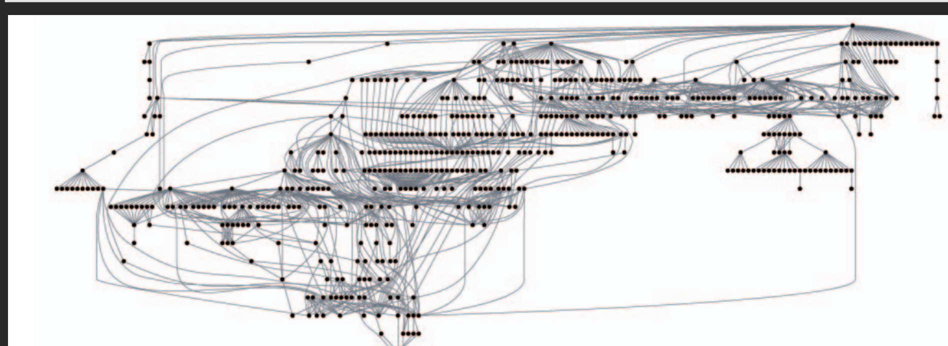
Layer 1
Layer 2
Layer 3
Layer 4



Reverse some edges to remove cycles
Assign nodes to hierarchy layers → Longest path layering
    Create dummy nodes to "fill in" missing layers
Arrange nodes within layer, minimize edge crossings
    Route edges – layout splines if needed

89

Produces hierarchical layout



Sugiyama-style layout emphasizes hierarchy

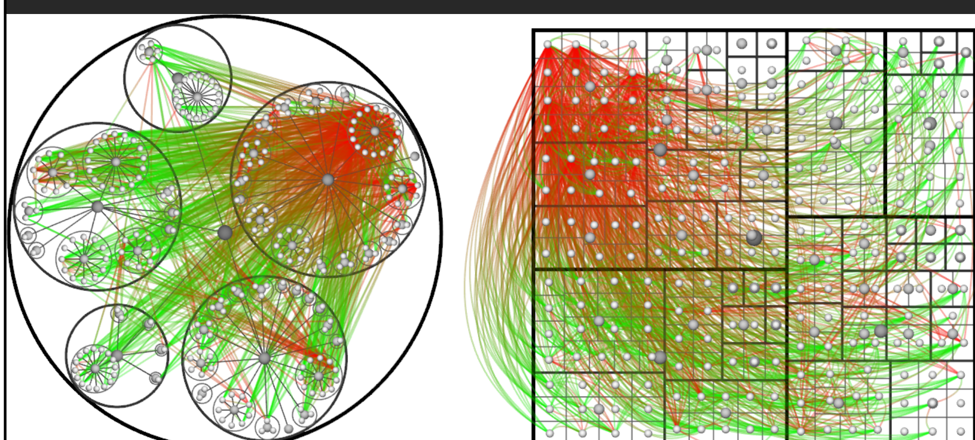
However, cycles in the graph may mislead.
Long edges can impede perception of proximity.

90

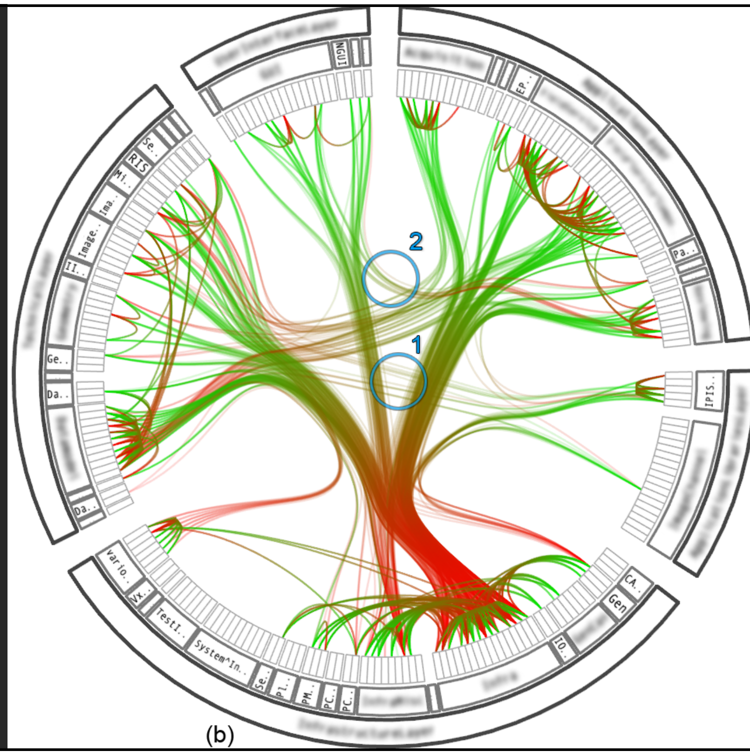
Hierarchical Edge Bundles

92

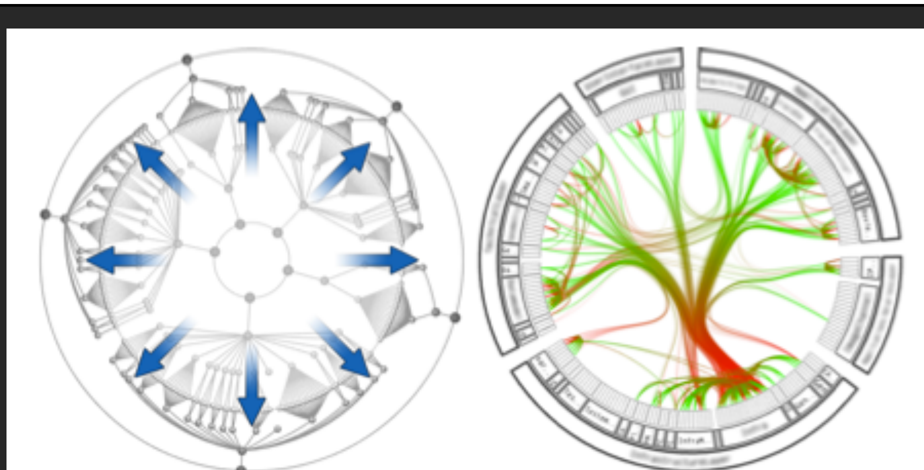
Trees with Adjacency Relations


(a) (b)

93



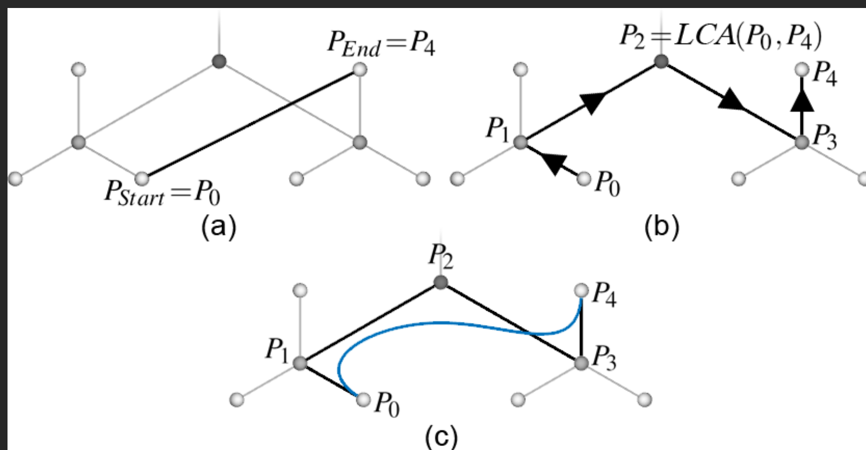
94



Use radial tree layout for inner circle
Mirror to outside
Replace inner tree with hierarchical edge bundles

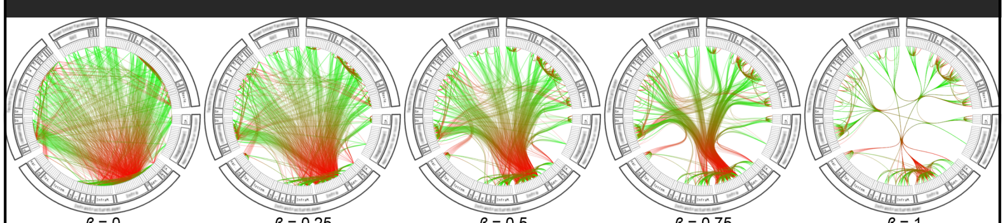
95

Bundle Edges along Hierarchy



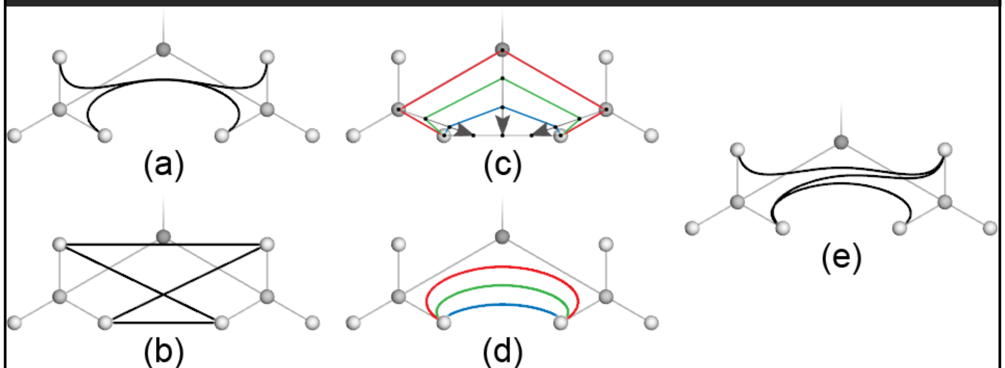
96

Increasing Edge Tension



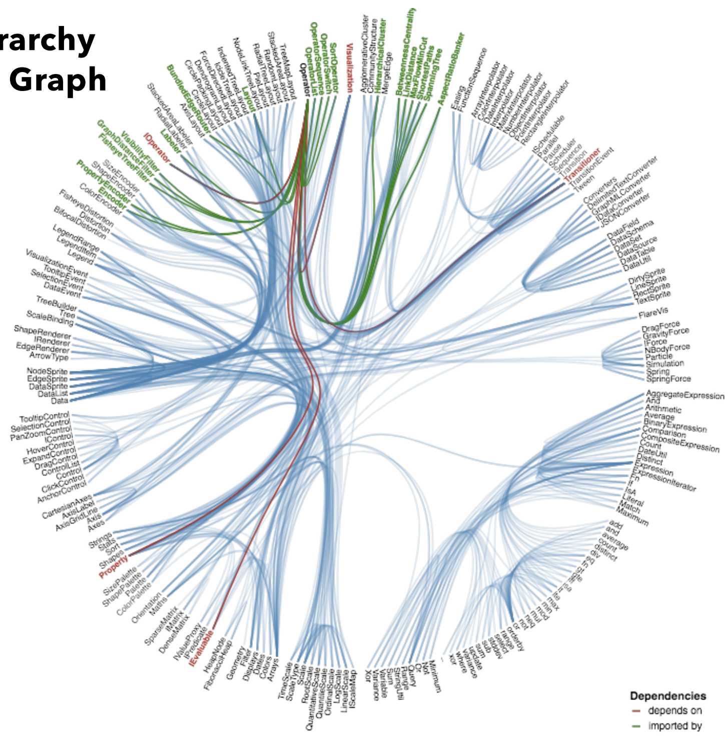
97

Configuring Edge Tension



98

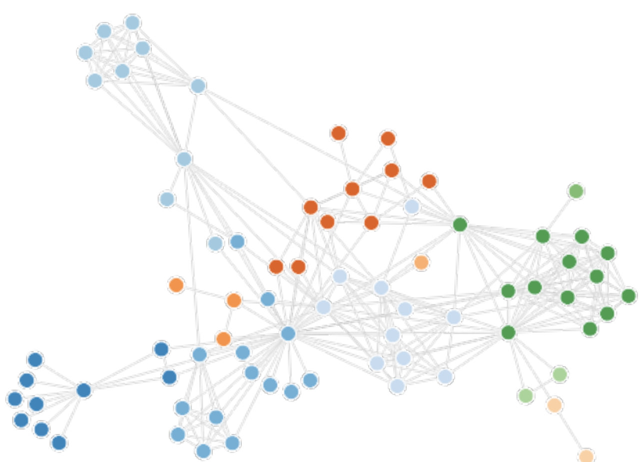
Flare Class Hierarchy & Dependency Graph



99

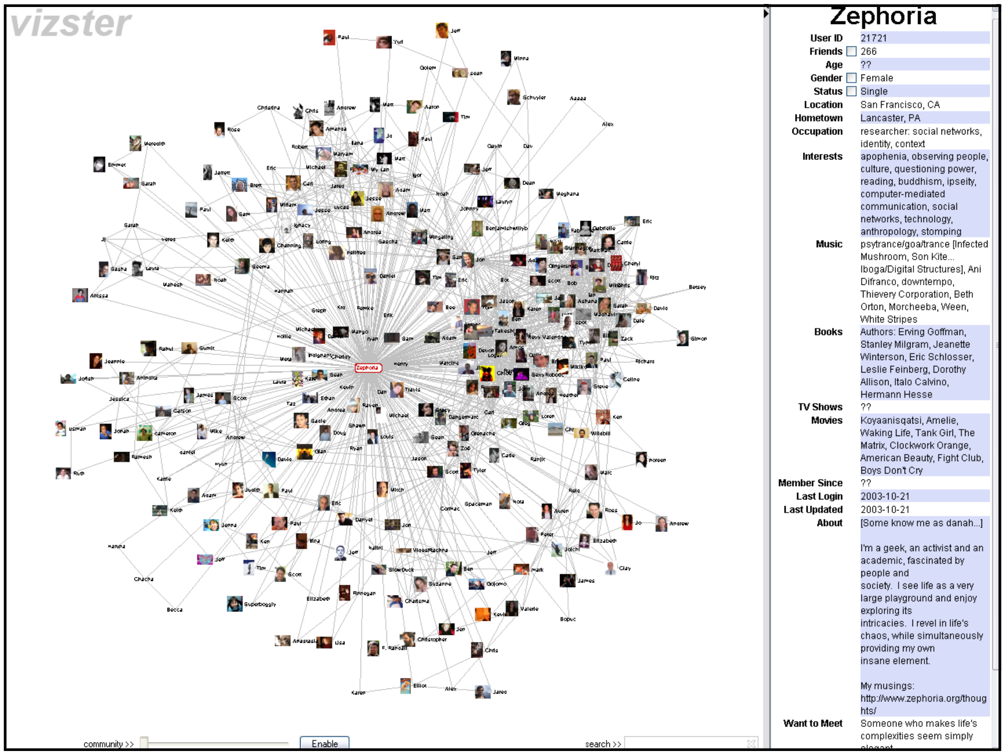
Force-Directed Layout

100



Interactive Example: Configurable Force Layout

101

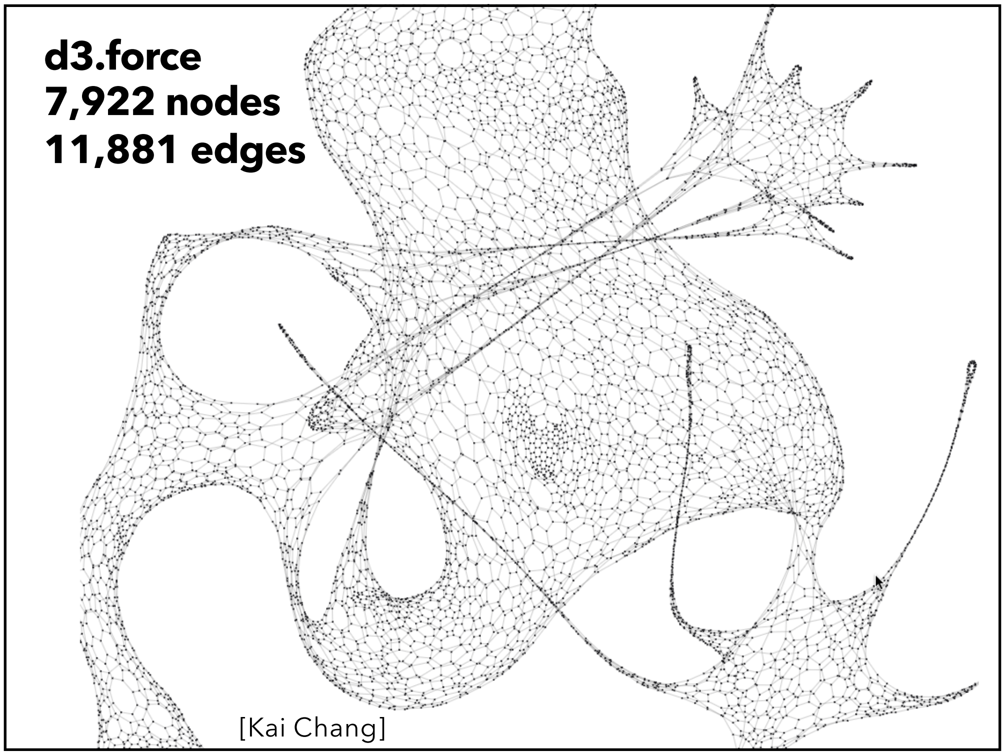


102

Use the Force!

http://mbostock.github.io/d3/talk/20110921/

103



104

Force-Directed Layout

Nodes = charged particles with air resistance — $F = q_i * q_j / d_{ij}^2$, $F = -b * v_i$

Edges = springs — $F = k * (L - d_{ij})$

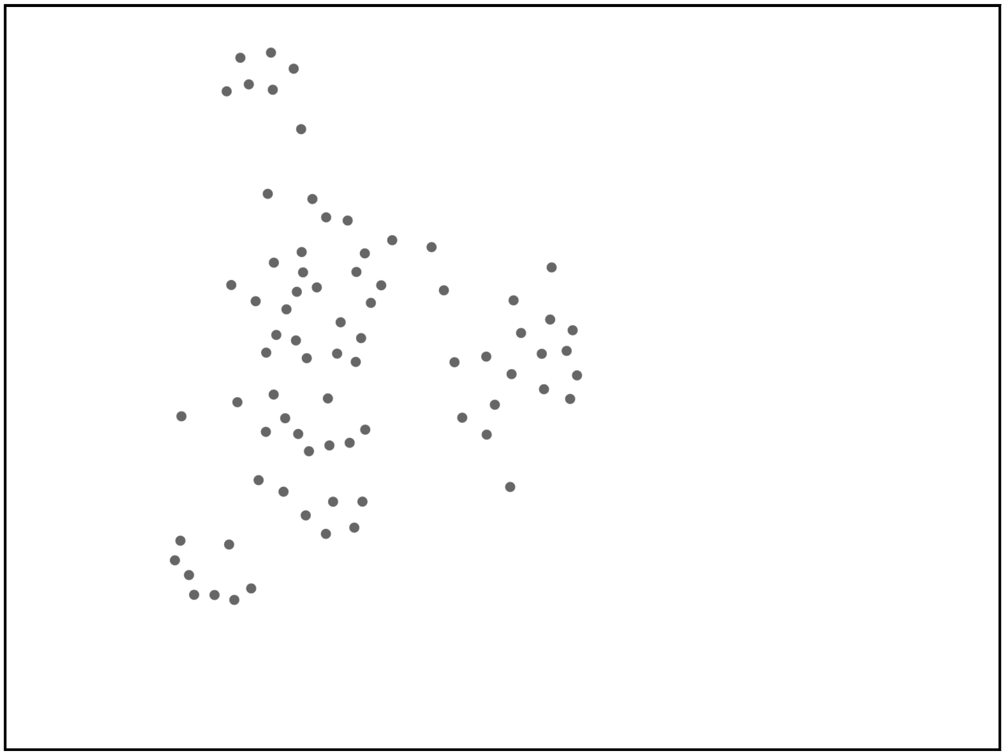
D3's force layout uses velocity Verlet integration

- Assume uniform mass *m* and timestep *Δt*:
- $F = ma \rightarrow F = a \rightarrow F = \Delta v / \Delta t \rightarrow F = \Delta v$
- *Forces simplify to velocity offsets!*

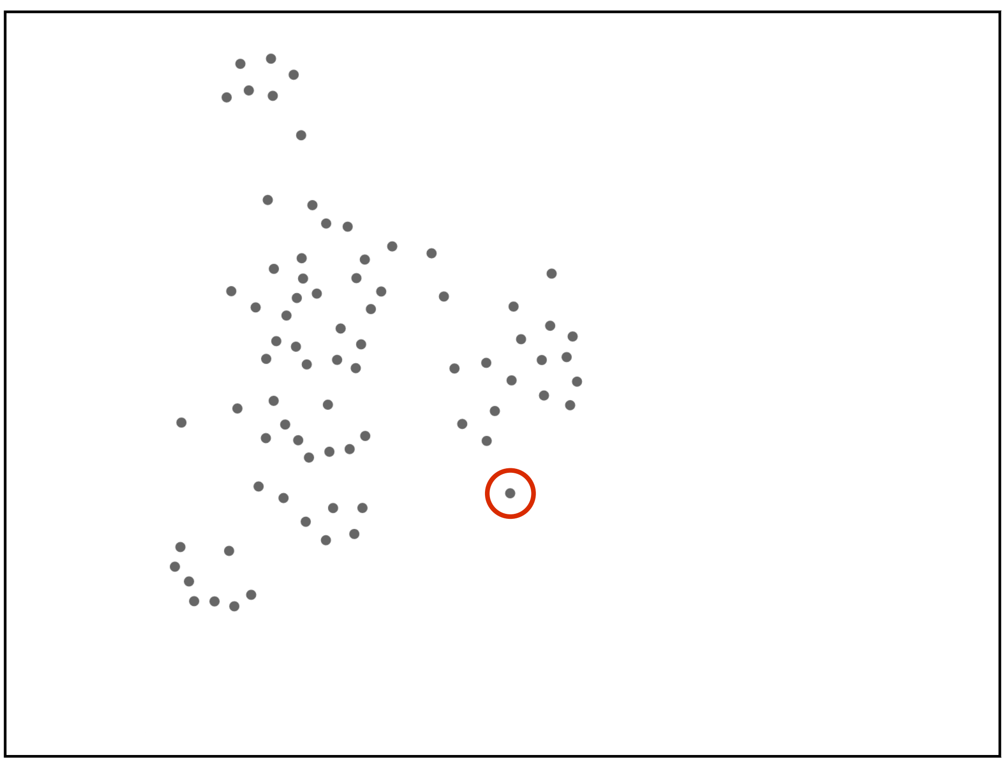
Repeatedly calculate forces, update node positions

- Naïve approach $O(N^2)$
- Speed up to $O(N \log N)$ using quadtree or k-d tree
- Numerical integration of forces at each time step

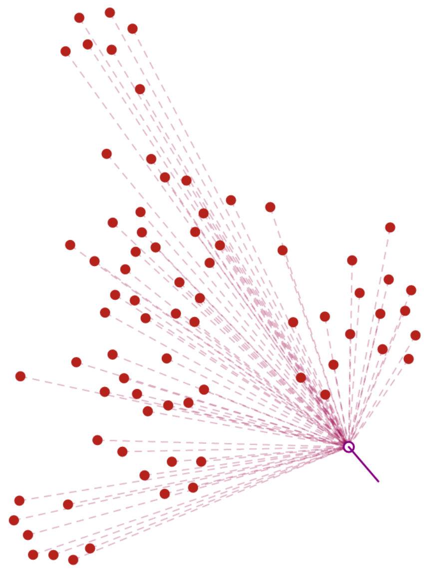
105



106

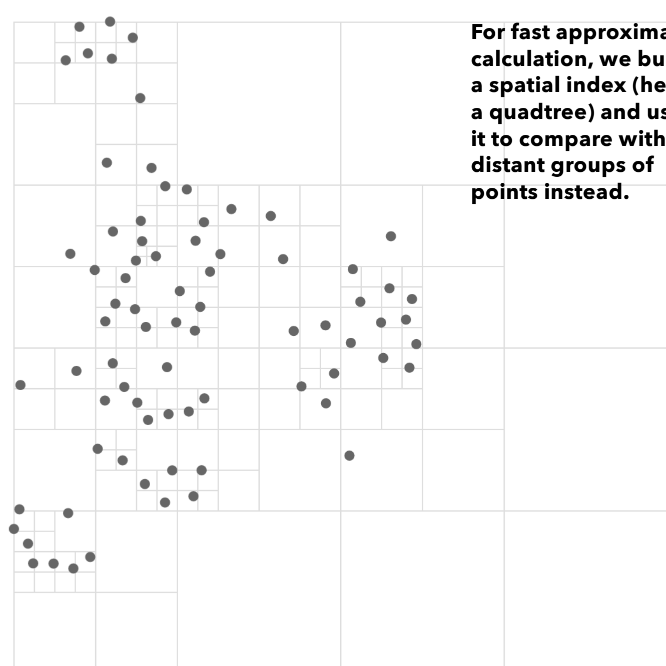


107



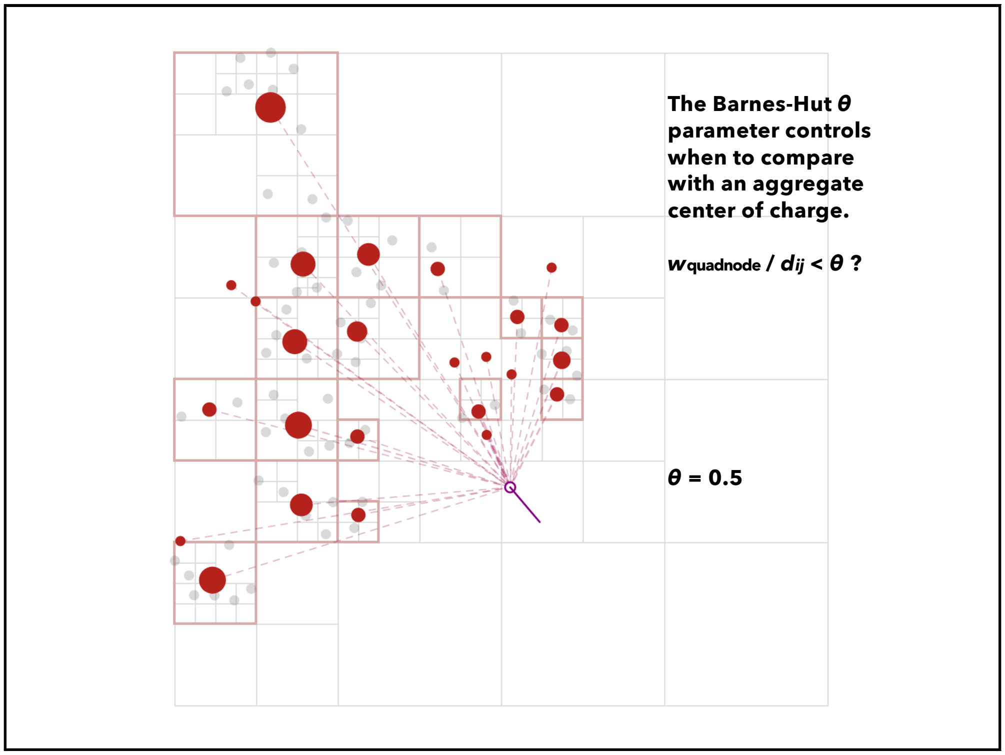
Naive calculation of forces at a point uses sum of forces from all other n-1 points.

108

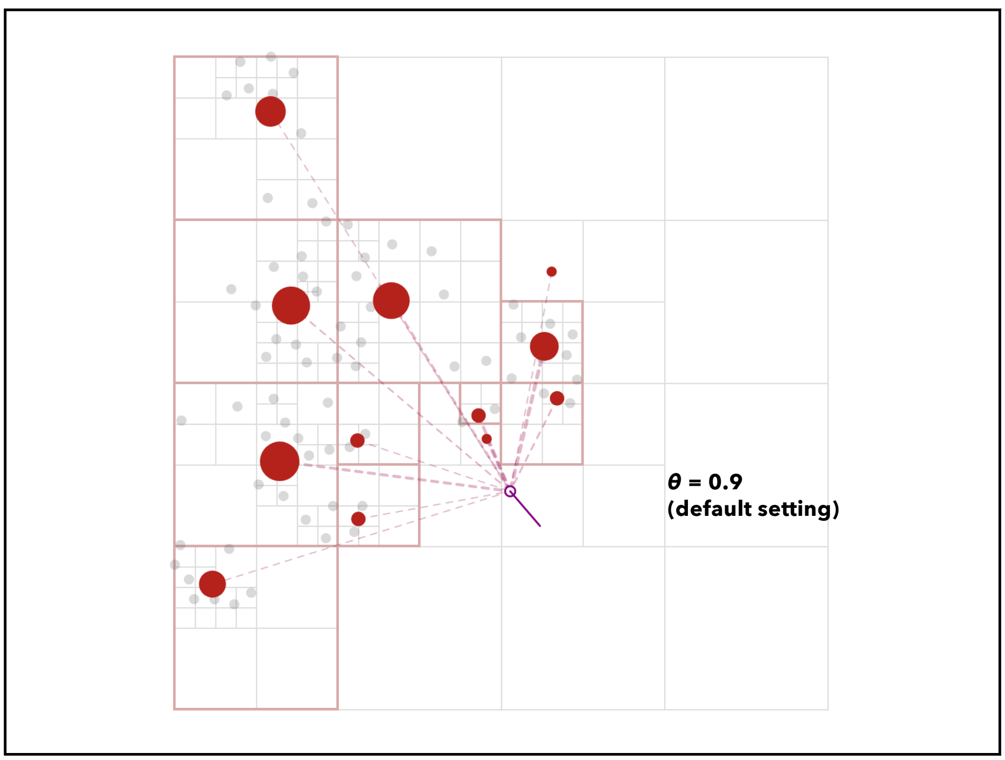


For fast approximate calculation, we build a spatial index (here, a quadtree) and use it to compare with distant groups of points instead.

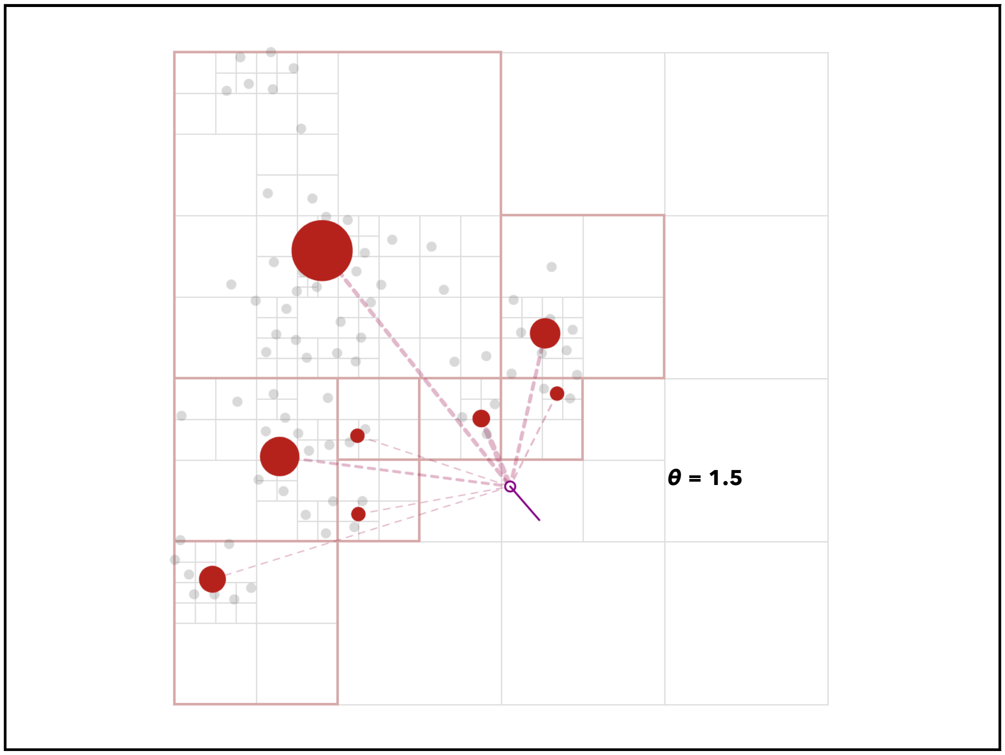
109


The Barnes-Hut θ parameter controls when to compare with an aggregate center of charge.
$w_{quadnode} / d_{ij} < \theta$?
θ = 0.5

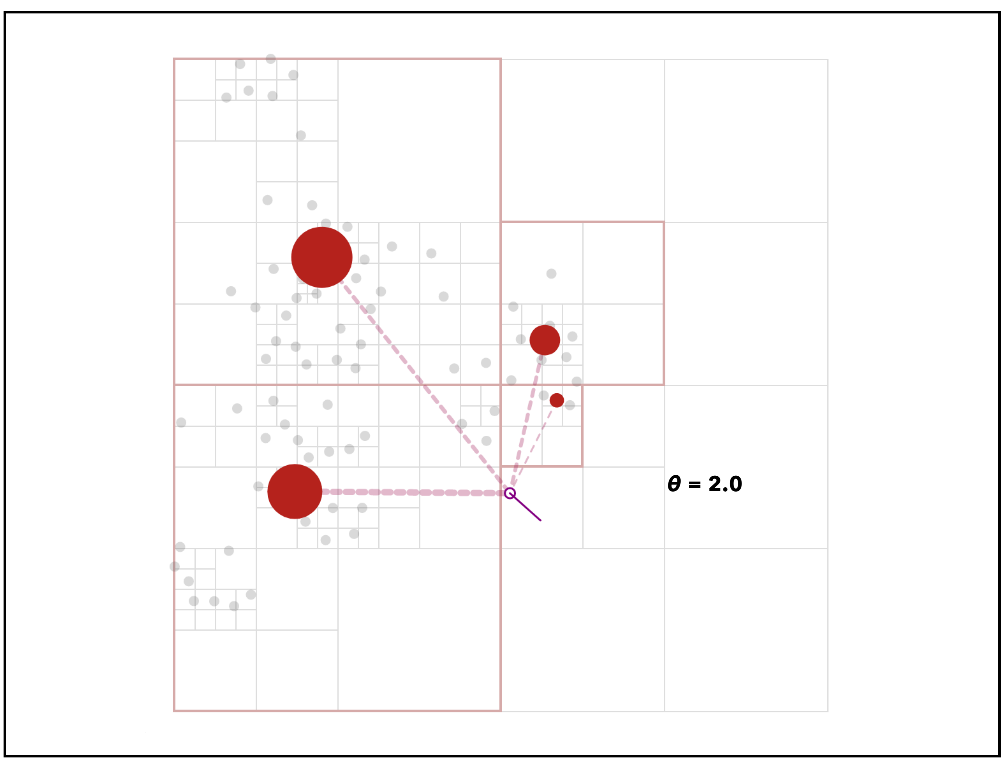
110


θ = 0.9
(default setting)

111



112



113