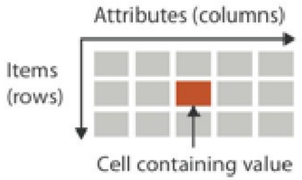
Paul Rosen
paul.rosen@utah.edu
@paulrosenphd
https://cspaul.com

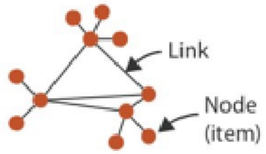
Visualization for Data Science
DS-4630 / CS-5630 / CS-6630

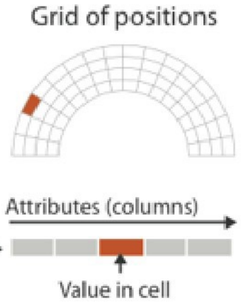
TREES & GRAPHS

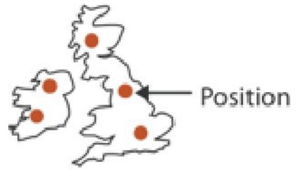
Dataset types

Tables	Networks & Trees	Fields	Geometry	Clusters, Sets, Lists
Items	Items (nodes)	Grids	Items	Items
Attributes	Links	Positions	Positions	
	Attributes	Attributes		

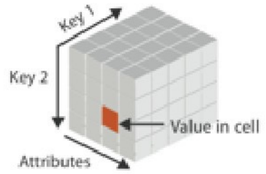




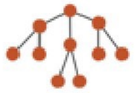


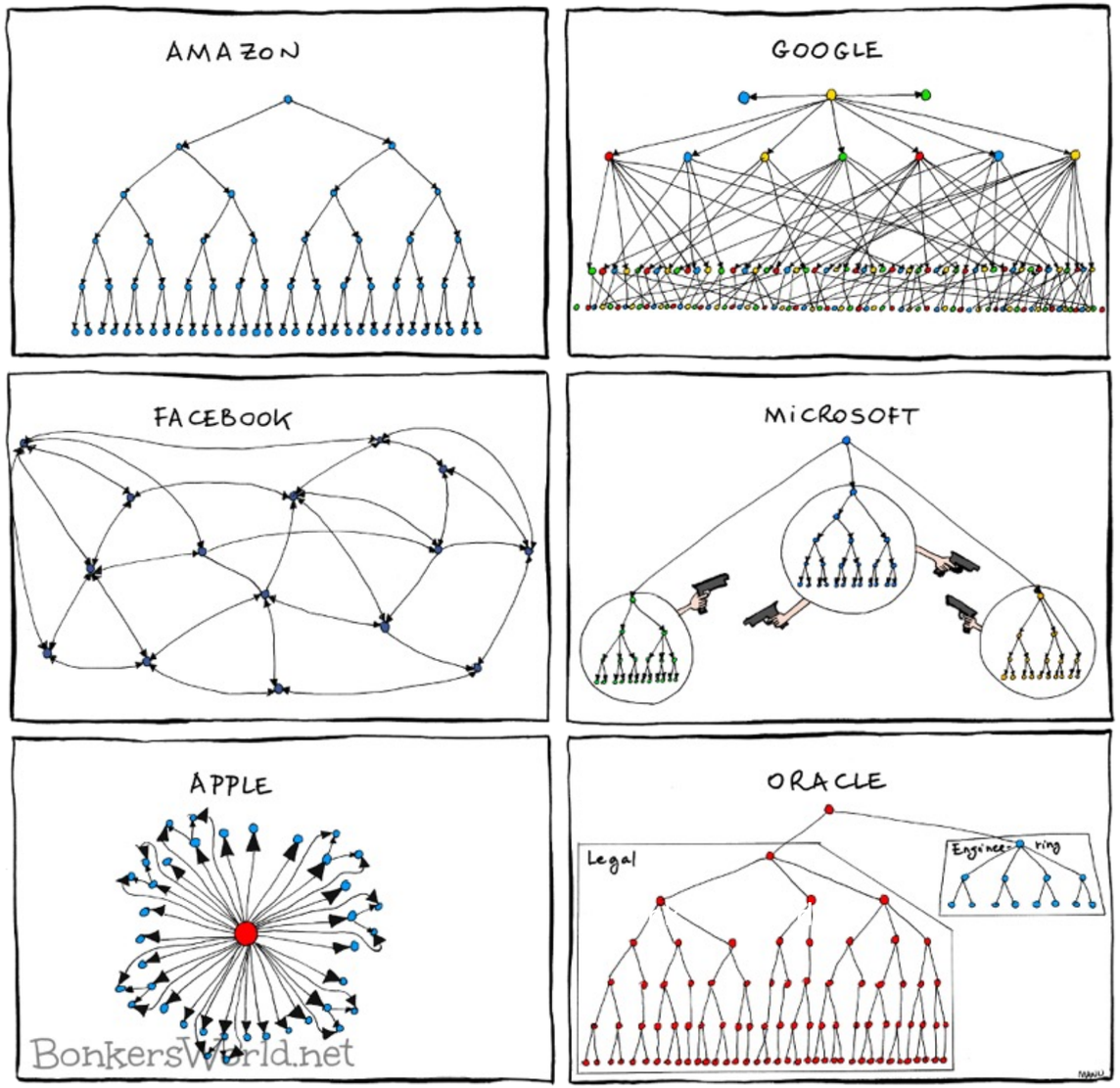


→ *Multidimensional Table*

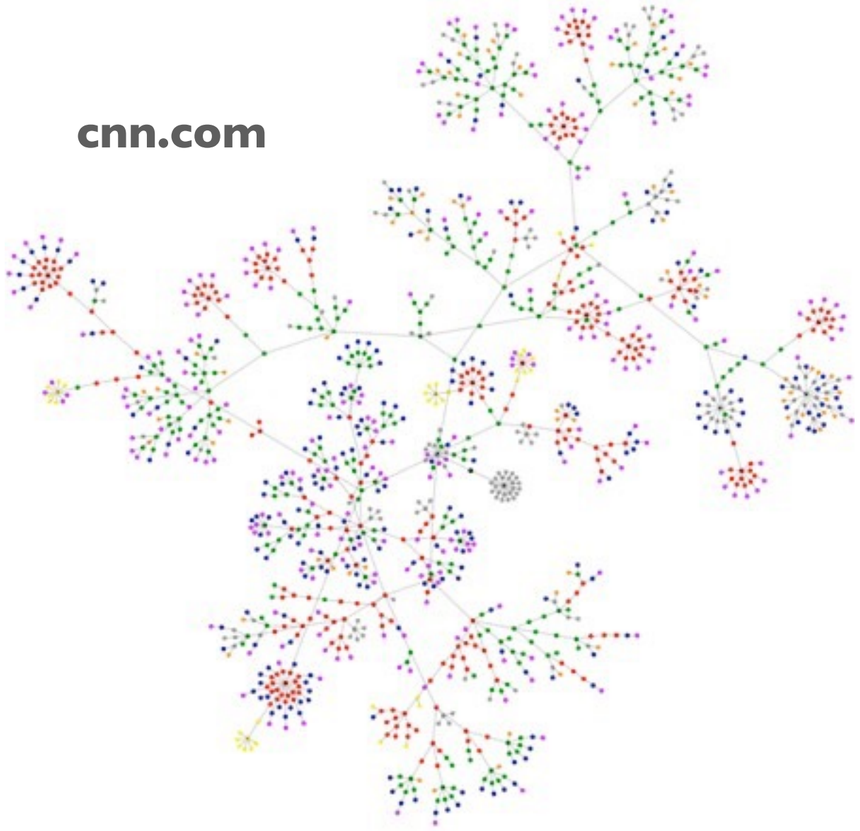


→ *Trees*

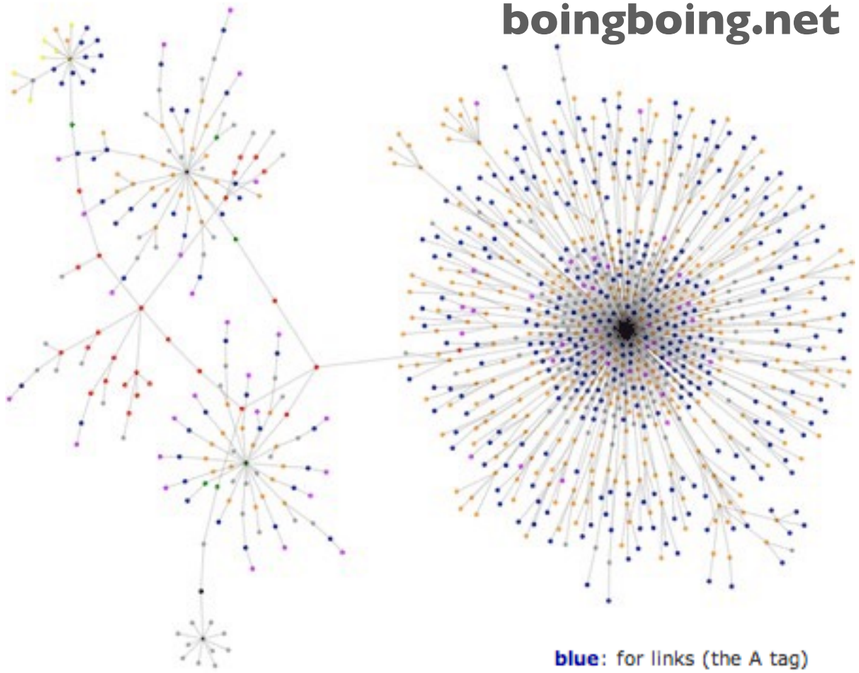



AMAZON
GOOGLE
FACEBOOK
MICROSOFT
APPLE
ORACLE
Legal
Engineering
BonkersWorld.net

cnn.com

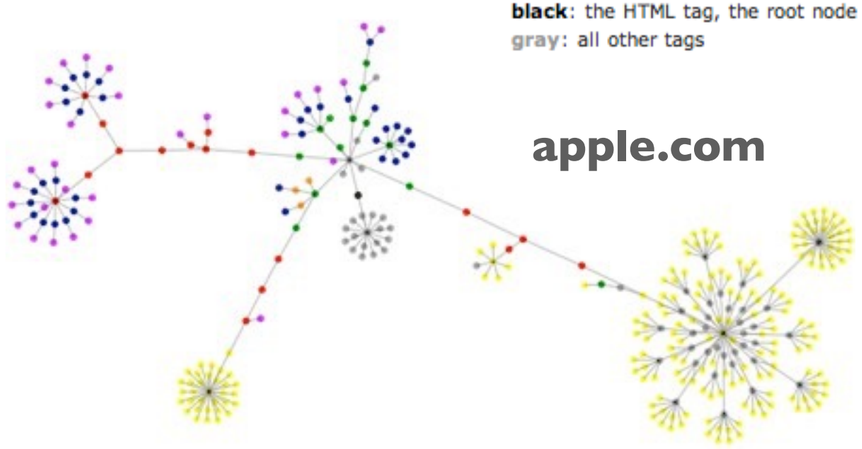


boingboing.net



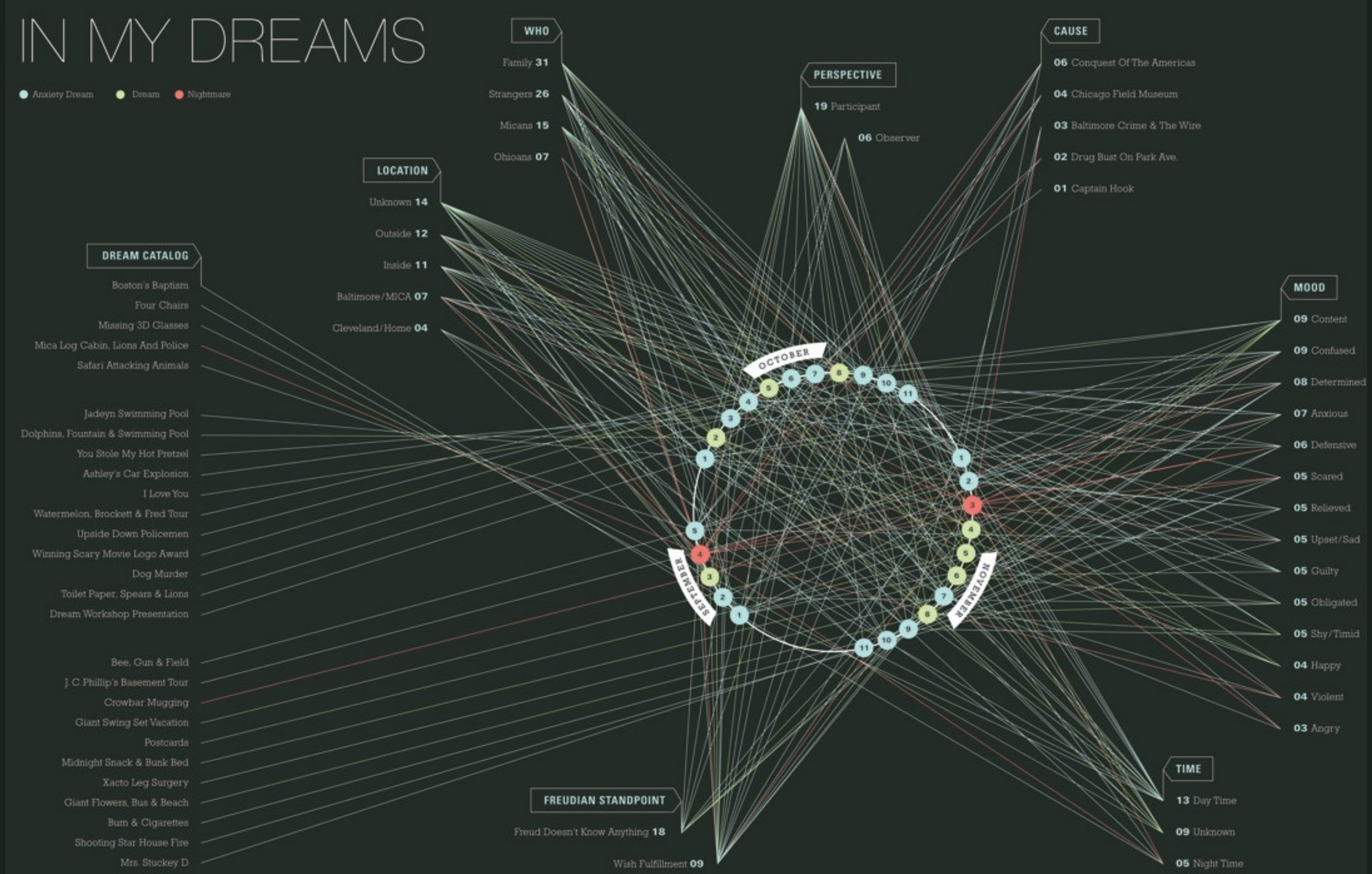
blue: for links (the A tag)
red: for tables (TABLE, TR and TD tags)
green: for the DIV tag
violet: for images (the IMG tag)
yellow: for forms (FORM, INPUT, TEXTAREA, SELECT and OPTION tags)
orange: for linebreaks and blockquotes (BR, P, and BLOCKQUOTE tags)
black: the HTML tag, the root node
gray: all other tags

wired.com



apple.com

Paul Butler



Kailie Parrish

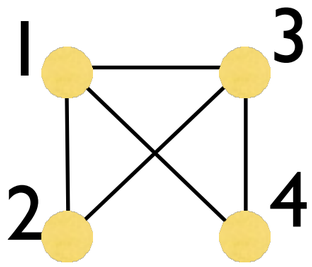
DEFINITIONS

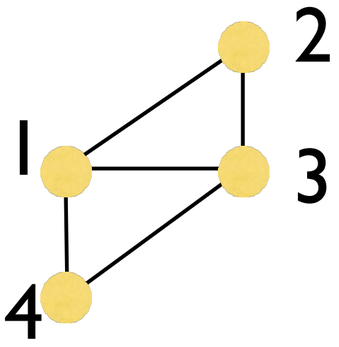
GRAPH

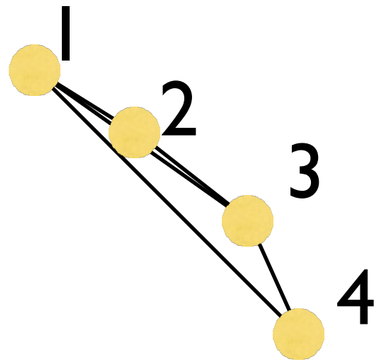
- A graph **G** consists of
  - **V** - a collection of vertices (or nodes)
  - **E** - a set of edges consisting of vertex pairs

- An edge e_{xy} = (x,y) connects two vertices x and y

- <u>Example</u>
  - V={1,2,3,4}
  - E={(1,2),(1,3),(2,3),(3,4),(4,1)}





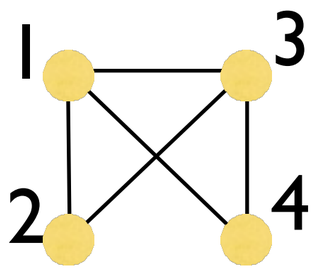


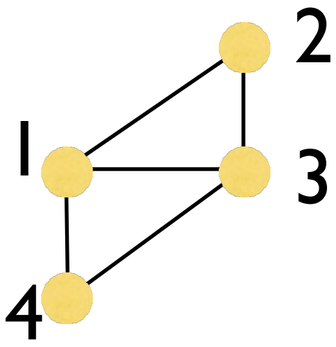
Nodes

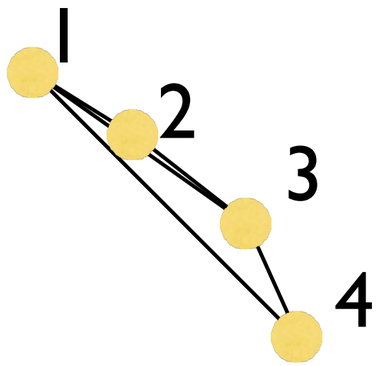
ID	Attribute 1	Attribute 2
1	3.4	Good
2	5.8	Bad
3	1.1	Ugly
4	-3.5	Really Ugly

Edges

Source	Target	Attribute 3
1	2	100
1	3	200
1	4	50
2	3	150
3	4	250





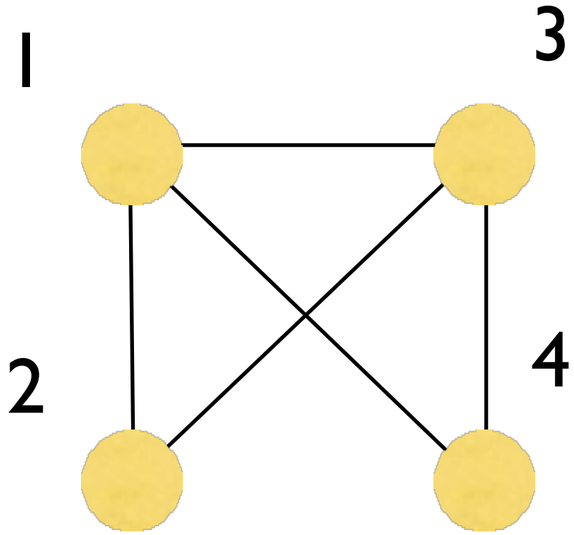


Adjacency Matrix

- A matrix that where each row/column represents a node and non-zero values represent edges

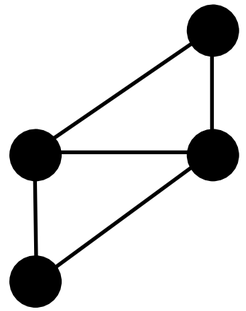
- Example

$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

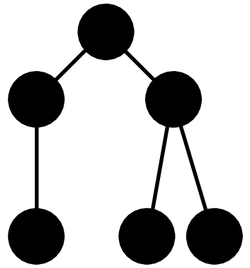


GRAPHS & TREES

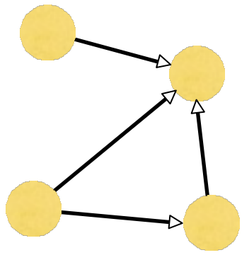
- graphs
  - model relationships about data
  - nodes and edges



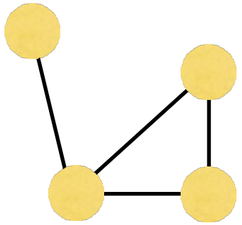
- trees
  - graphs with hierarchical structure
  - nodes as parents and children



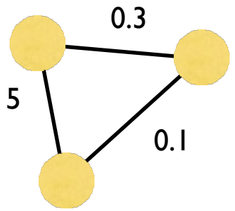
a bunch of definitions



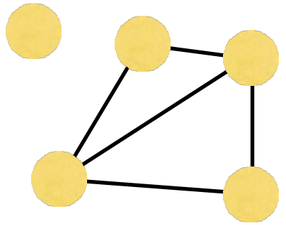
A directed graph



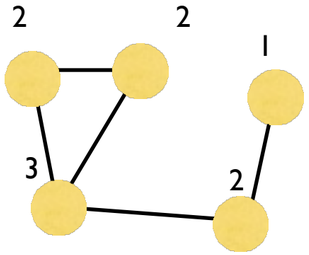
An undirected graph



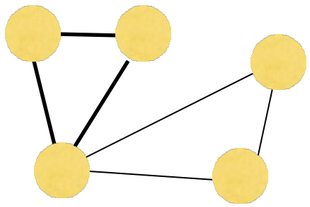
Weighted



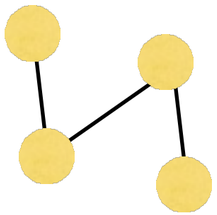
Unconnected



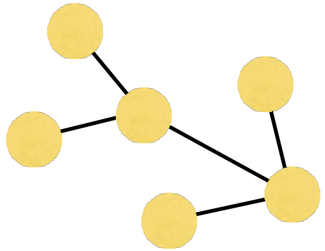
Node degrees



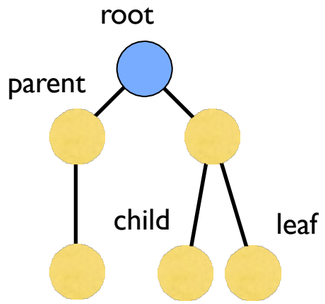
A cycle



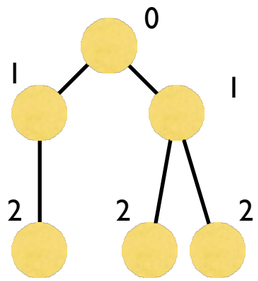
An acyclic graph



A connected acyclic graph,
a.k.a.a **tree**



A rooted tree
or hierarchy



Node depths

VISUALIZING TREES

How to cite this site?

Check out other surveys

treevis.net - A Visual Bibliography of Tree Visualization 2.0 by Hans-Jörg Schulz

v.28-AUG-2015

Dimensionality: All

Representation: All

Alignment: All

Fulltext Search

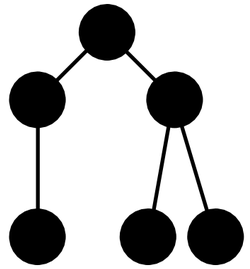
Techniques Shown: 286

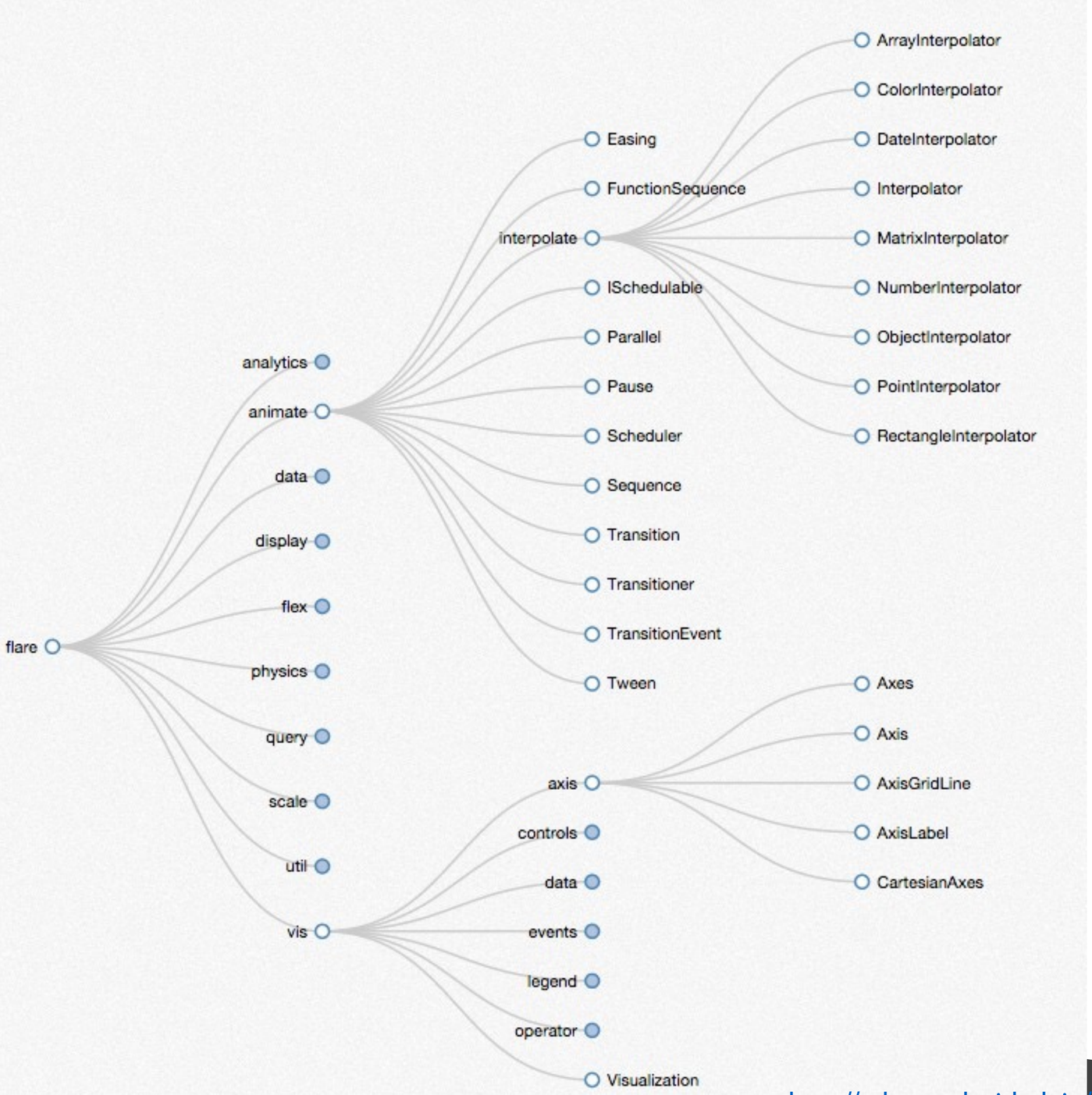
ROOTED TREES

- recursion makes it elegant and fast to draw trees
- approaches:
  - node link
  - layered
  - indentation
  - enclosure

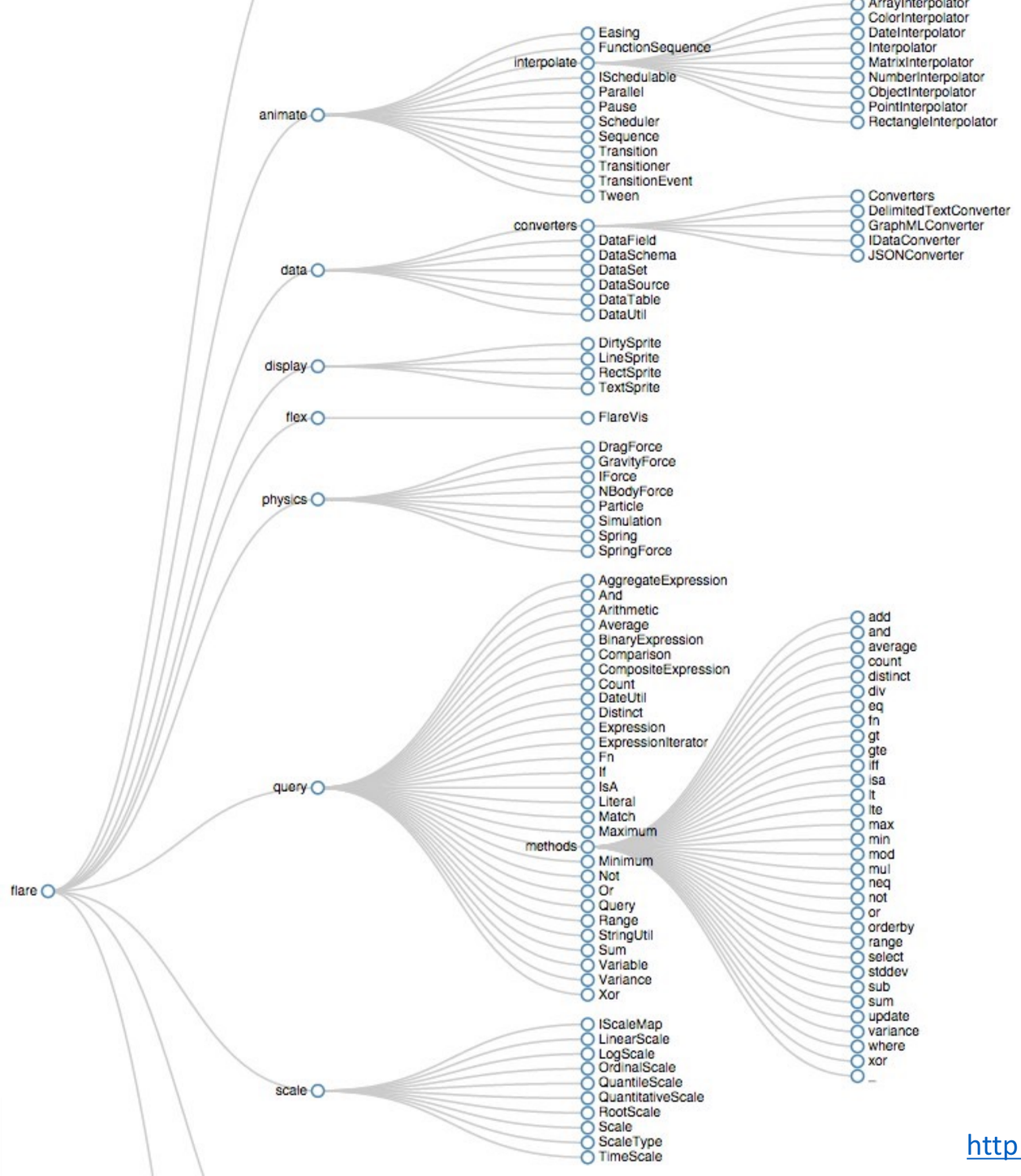
NODE-LINK DIAGRAMS

- nodes are distributed in space, connected by straight or curved lines
- typical approach is to use 2D space to break apart breadth and depth
- often space is used to communicate hierarchical orientation





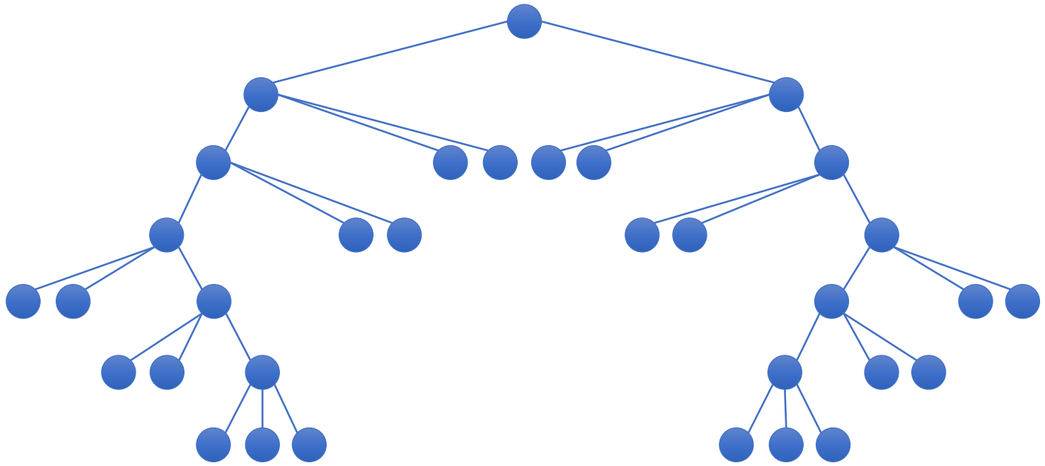
http://mbostock.github.io/d3/talk/20111018/tree.html


flare
animate
Easing
FunctionSequence
interpolate
ISchedulable
Parallel
Pause
Scheduler
Sequence
Transition
Transitioner
TransitionEvent
Tween
ArrayInterpolator
ColorInterpolator
DateInterpolator
Interpolator
MatrixInterpolator
NumberInterpolator
ObjectInterpolator
PointInterpolator
RectangleInterpolator
data
converters
Converters
DelimitedTextConverter
GraphMLConverter
IDataConverter
JSONConverter
DataField
DataSchema
DataSet
DataSource
DataTable
DataUtil
display
DirtySprite
LineSprite
RectSprite
TextSprite
flex
FlareVis
physics
DragForce
GravityForce
IForce
NBodyForce
Particle
Simulation
Spring
SpringForce
query
AggregateExpression
And
Arithmetic
Average
BinaryExpression
Comparison
CompositeExpression
Count
DateUtil
Distinct
Expression
ExpressionIterator
Fn
If
IsA
Literal
Match
Maximum
methods
Minimum
Not
Or
Query
Range
StringUtil
Sum
Variable
Variance
Xor
add
and
average
count
distinct
div
eq
fn
gt
gte
iff
isa
lt
lte
max
min
mod
mul
neq
not
or
orderby
range
select
stddev
sub
sum
update
variance
where
xor
_
scale
IScaleMap
LinearScale
LogScale
OrdinalScale
QuantileScale
QuantitativeScale
RootScale
Scale
ScaleType
TimeScale


flare
vis
data
Tree
TreeBuilder
ScaleBinding
render
NodeSprite
EdgeSprite
DataSprite
DataList
Data
events
TooltipEvent
SelectionEvent
DataEvent
legend
LegendRange
LegendItem
Legend
controls
TooltipControl
SelectionControl
PanZoomControl
IControl
HoverControl
ExpandControl
DragControl
ControlList
Control
ClickControl
AnchorControl
axis
CartesianAxes
AxisLabel
AxisGridLine
Axis
Axes
operator
distortion
encoder
filter
label
Visualization
IOperator
OperatorList
OperatorSequence
OperatorSwitch
SortOperator
layout
RadialTreeLayout
CircleLayout
NodeLinkTreeLayout
CirclePackingLayout
StackedAreaLayout
TreeMapLayout
RandomLayout
PieLayout
IndentedTreeLayout
IcicleTreeLayout
ForceDirectedLayout
DendrogramLayout
BundledEdgeRouter
AxisLayout
StackedAreaLabeler
RadialLabeler
GraphDistanceFilter
FisheyeTreeFilter
VisibilityFilter
SizeEncoder
ShapeEncoder
PropertyEncoder
ColorEncoder
FisheyeDistortion
BifocalDistortion
Distortion
ShapeRenderer
IRenderer
EdgeRenderer
ArrowType
util
Strings
Stats
Sort
Shapes
Property
palette
Orientation
Maths
math
IValueProxy
IPredicate
IEvaluable
heap
Geometry
Filter
Displays
Dates
Colors
Arrays
SizePalette
ShapePalette
Palette
ColorPalette
SparseMatrix
IMatrix
DenseMatrix
HeapNode
FibonacciHeap
scale
TimeScale
ScaleType
Scale
RootScale
QuantitativeScale
OrdinalScale
LogScale
LinearScale
IScaleMap
query
Xor
Variance
Variable
Update
Sum
StdDev
Range
Query
Or
Not
Minimum
Methods
Maximum
Match
Literal
IsA
If
Expression
ExpressionIterator
DateUtil
Distinct
Count
CompositeExpression
Comparison
BinaryExpression
Average
Arithmetic
And
AggregateExpression
vis
xor
variance
update
sum
stddev
sub
select
range
orderby
or
not
neq
mul
mod
min
max
lte
lt
gte
gt
eq
distinct
count
average
and
add
data
DataField
DataSchema
DataSet
DataSource
DataTable
DataUtil
display
DirtySprite
LineSprite
RectSprite
TextSprite
flex
FlareVis
physics
DragForce
GravityForce
IForce
NBodyForce
Particle
Simulation
Spring
SpringForce
analytics
cluster
AgglomerativeCluster
CommunityStructure
HierarchicalCluster
MergeEdge
graph
BetweennessCentrality
LinkDistance
MaxFlowMinCut
ShortestPaths
SpanningTree
optimization
AspectRatioBanker
animate
Interpolator
ArrayInterpolator
ColorInterpolator
DateInterpolator
MatrixInterpolator
NumberInterpolator
ObjectInterpolator
PointInterpolator
RectangleInterpolator
Easing
FunctionSequence
ISchedulable
Parallel
Pause
Scheduler
Sequence
Transition
Transitioner
TransitionEvent
Tween
converters
Converters
DelimitedTextConverter
GraphMLConverter
IDataConverter
JSONConverter

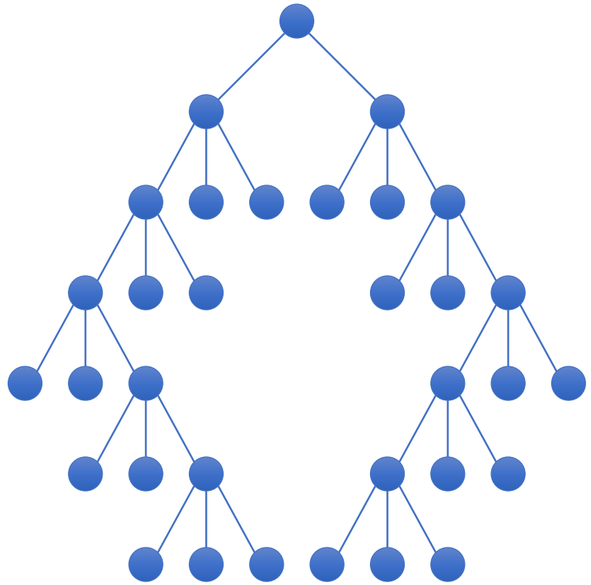
REINGOLD-TILFORD

- repeatedly divide space for subtrees by leaf count
  - breadth of tree along one dimension
  - depth along the other dimension



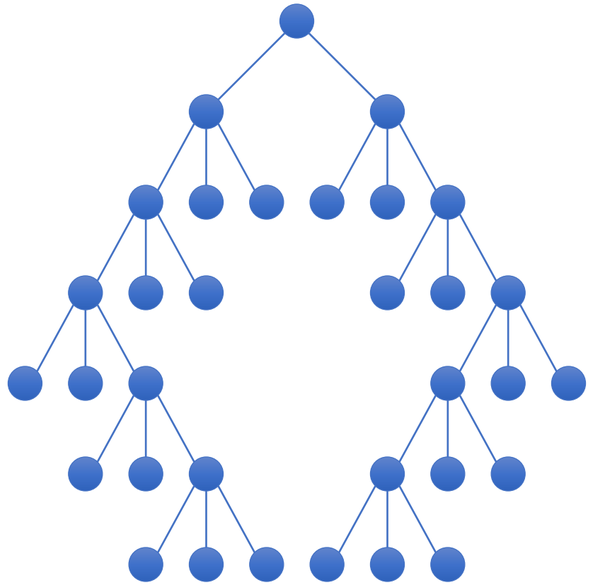
REINGOLD-TILFORD

- goal
  - make smarter use of space
  - maximize density and symmetry



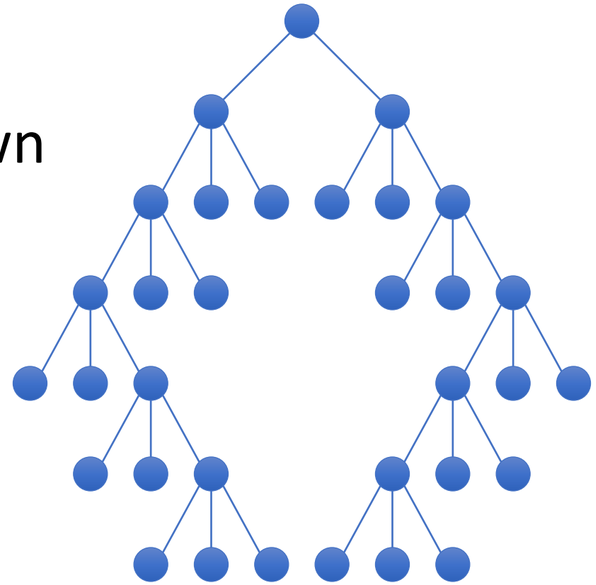
REINGOLD-TILFORD

- design concerns
  - clearly encode depth level
  - no edge crossings
  - isomorphic subtrees drawn identically
  - compact



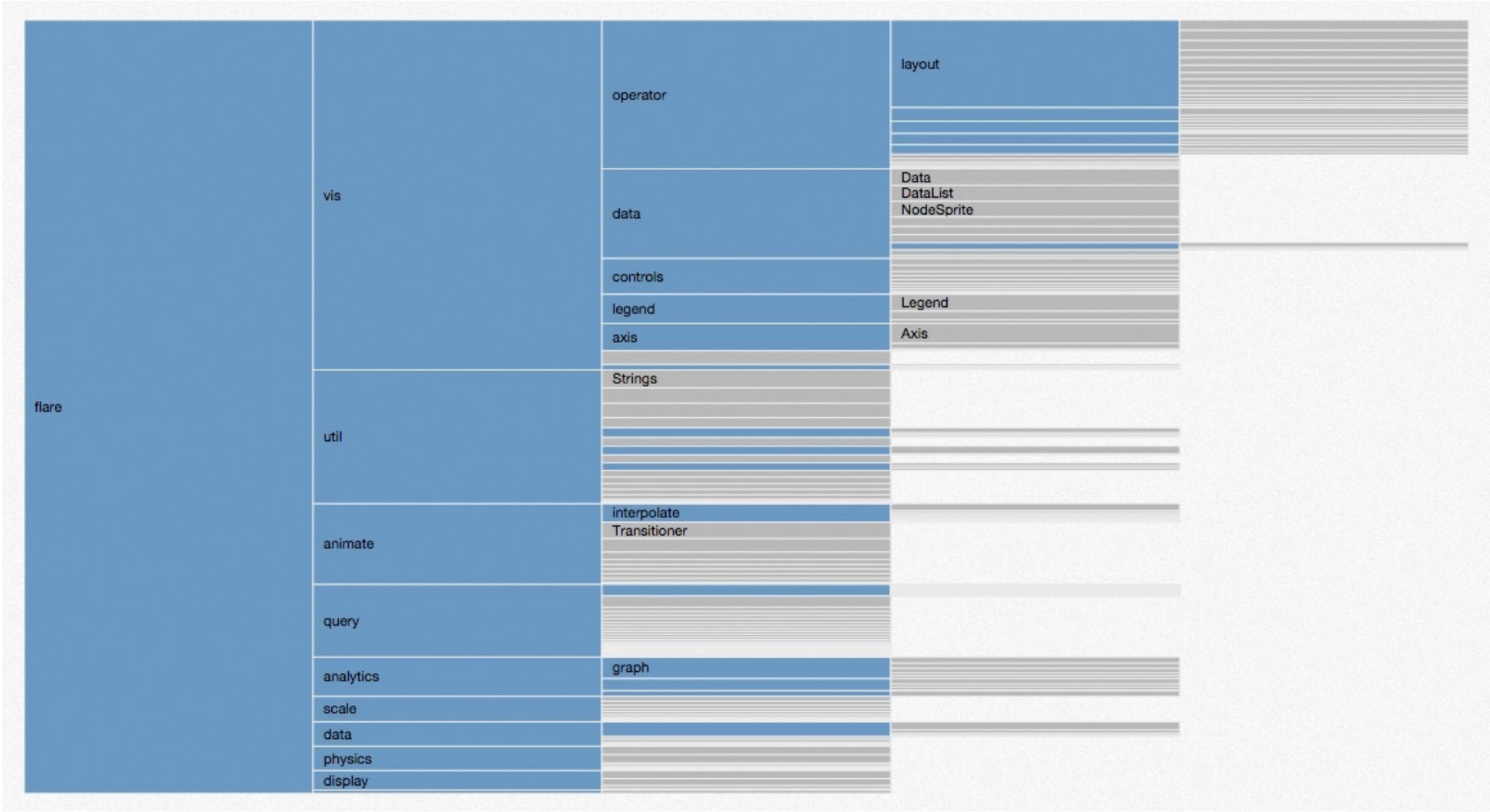
REINGOLD-TILFORD

- approach
  - bottom up recursive approach
  - for each parent make sure every subtree is drawn
  - pack subtrees as closely as possible
  - center parent over subtrees

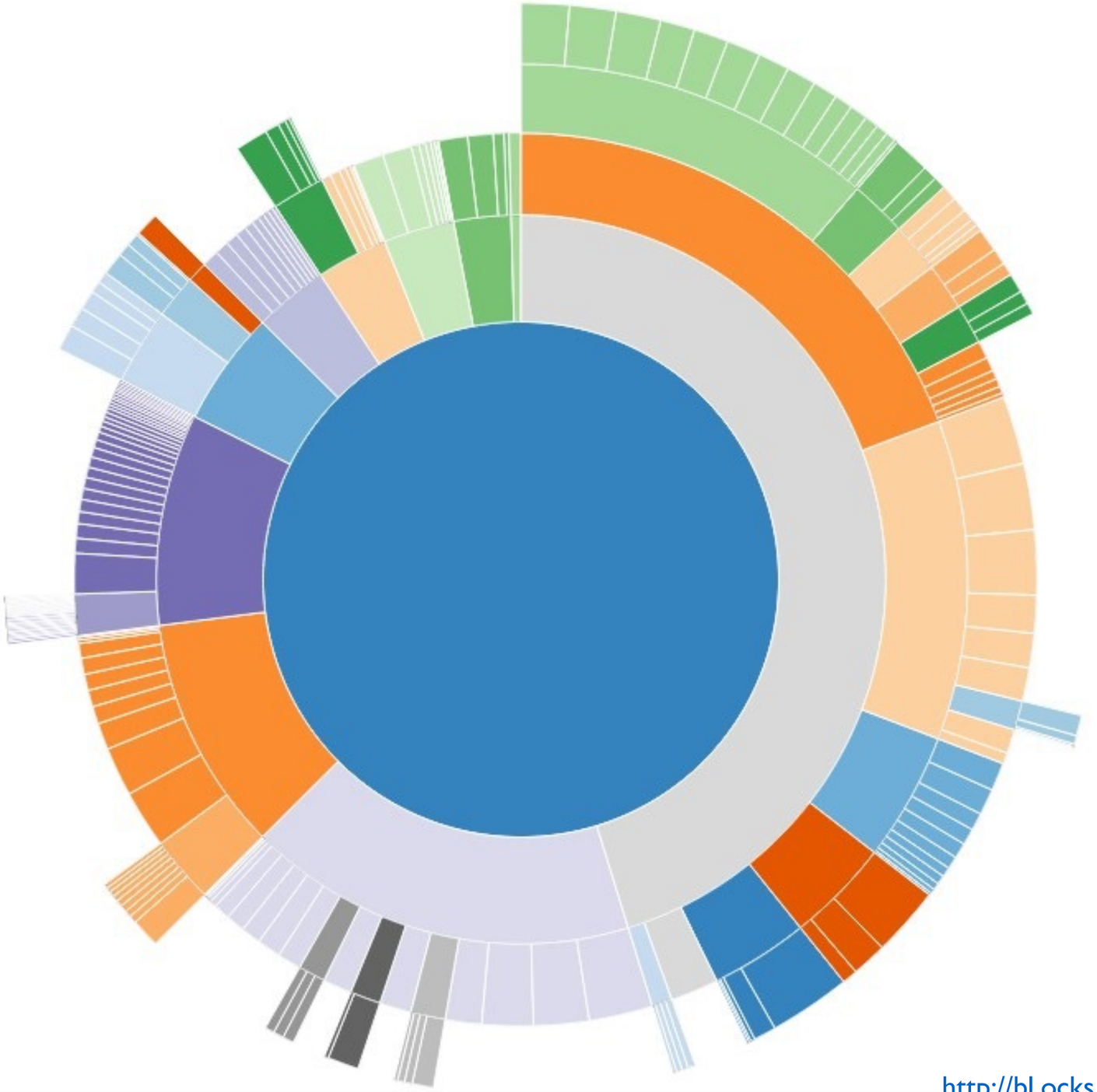


LAYERED DIAGRAMS

- recursive subdivision of space
- structure encoded using:
  - layering
  - adjacency
  - alignment



http://mbostock.github.io/d3/talk/20111018/partition.html



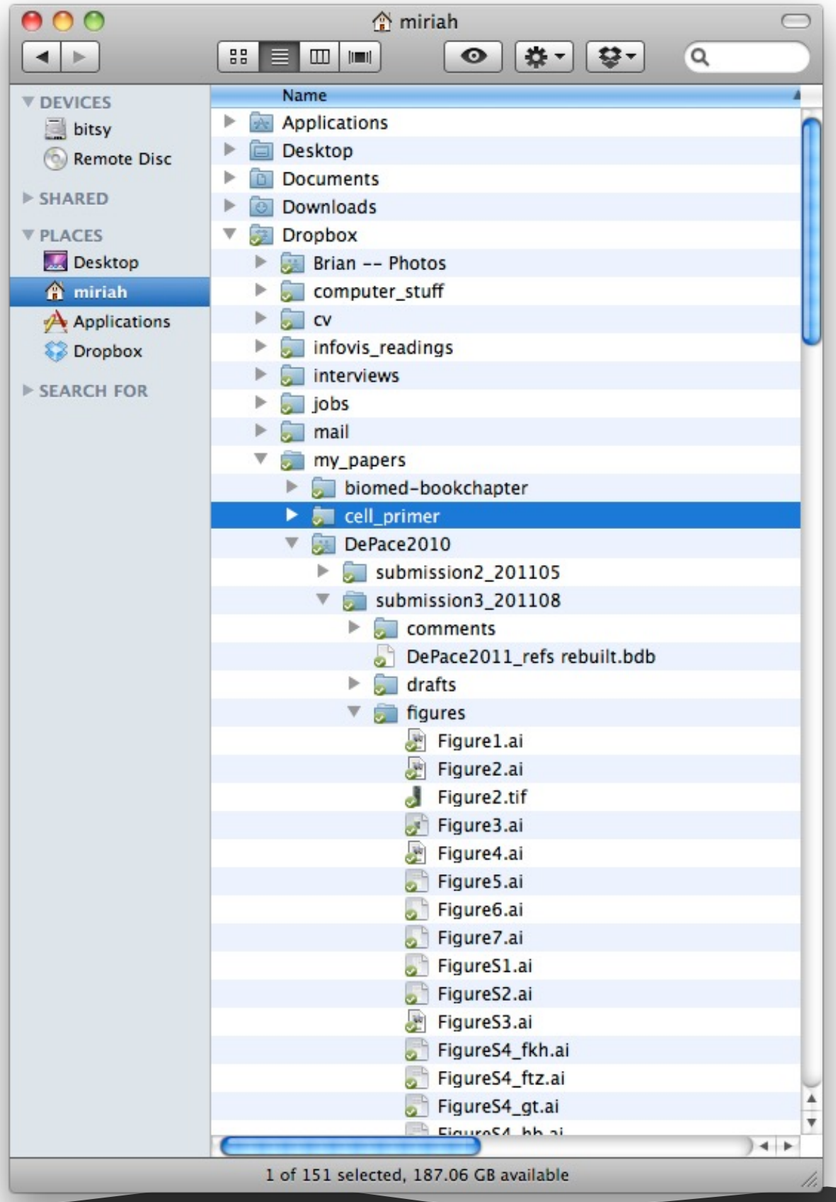
http://bl.ocks.org/mbostock/raw/4348373/

SCALE PROBLEM

- tree breadth often grows exponentially—quickly run out of space!
- solutions
  - scrolling or panning
  - filtering or zooming
  - hyperbolic layout

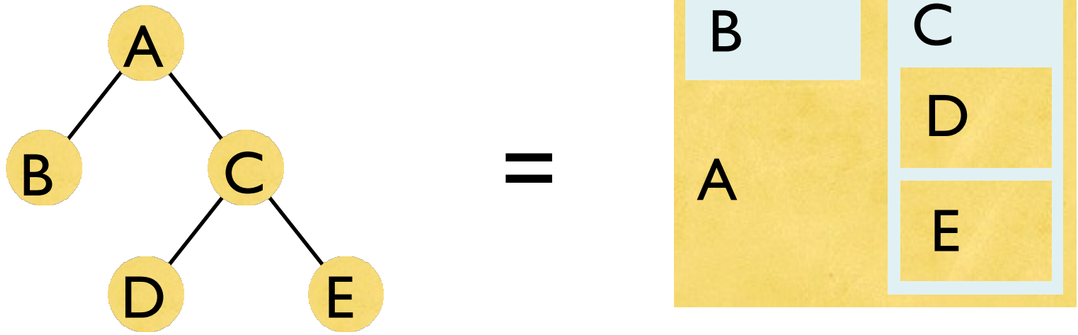
INDENTATION

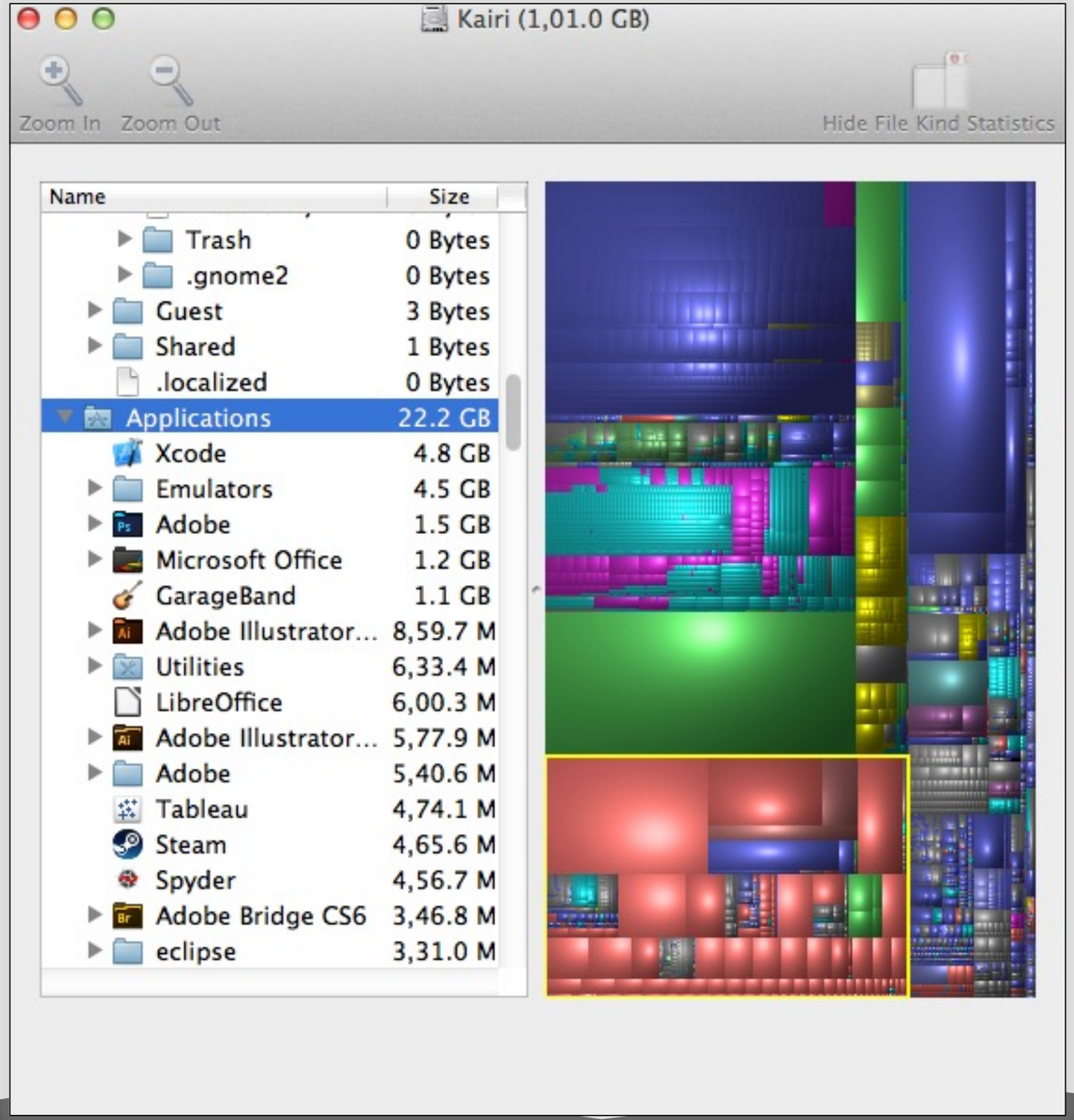
- indentation used to show parent/child relationships
- breadth and depth contend for space
- problem: often requires a great deal of scrolling



ENCLOSURE DIAGRAMS

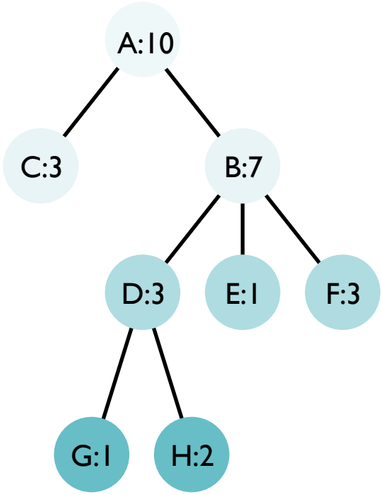
- encode structure using spatial enclosure
  - often referred to as treemaps
- benefits
  - provides single view of entire tree
  - easier to spot small / large nodes
- problems
  - difficult to accurately read depth

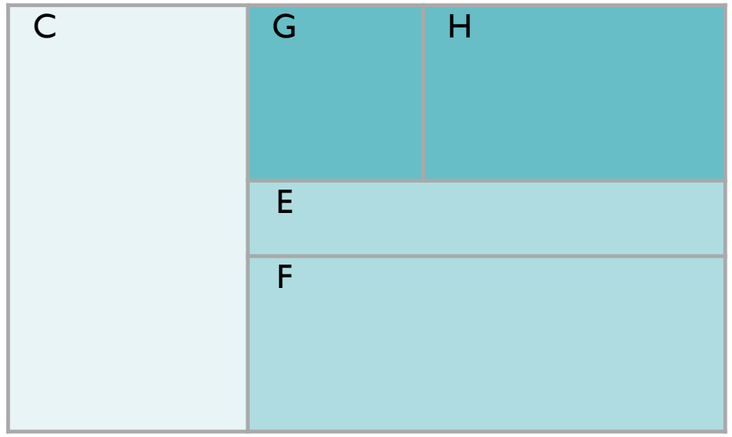


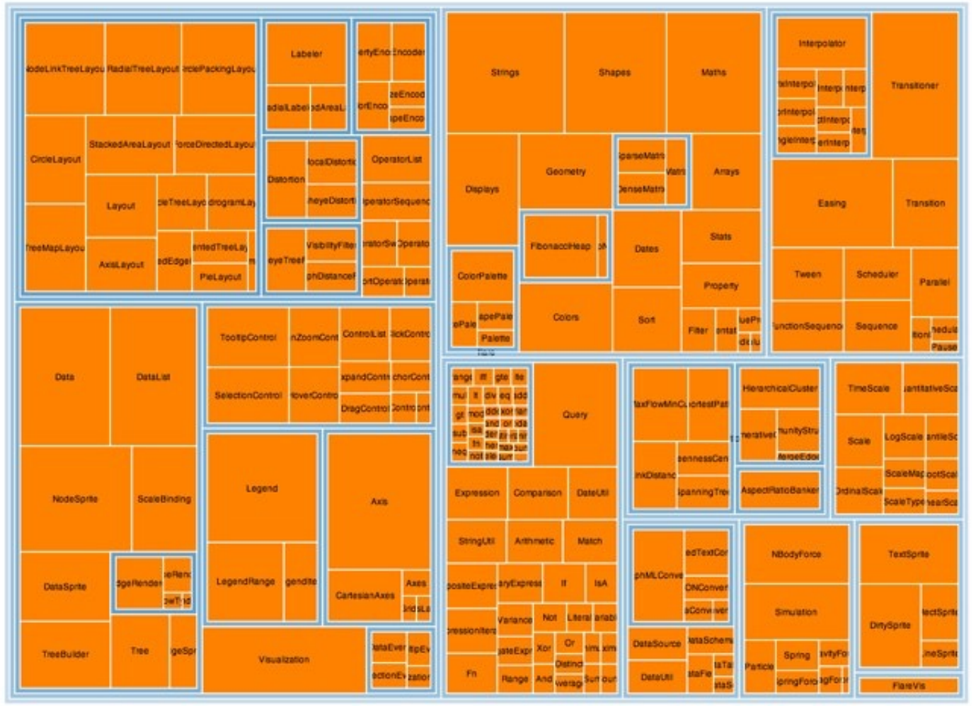

Kairi (1,01.0 GB)
Zoom In
Zoom Out
Hide File Kind Statistics
Name
Size
Trash 0 Bytes
.gnome2 0 Bytes
Guest 3 Bytes
Shared 1 Bytes
.localized 0 Bytes
Applications 22.2 GB
Xcode 4.8 GB
Emulators 4.5 GB
Adobe 1.5 GB
Microsoft Office 1.2 GB
GarageBand 1.1 GB
Adobe Illustrator... 8,59.7 M
Utilities 6,33.4 M
LibreOffice 6,00.3 M
Adobe Illustrator... 5,77.9 M
Adobe 5,40.6 M
Tableau 4,74.1 M
Steam 4,65.6 M
Spyder 4,56.7 M
Adobe Bridge CS6 3,46.8 M
eclipse 3,31.0 M

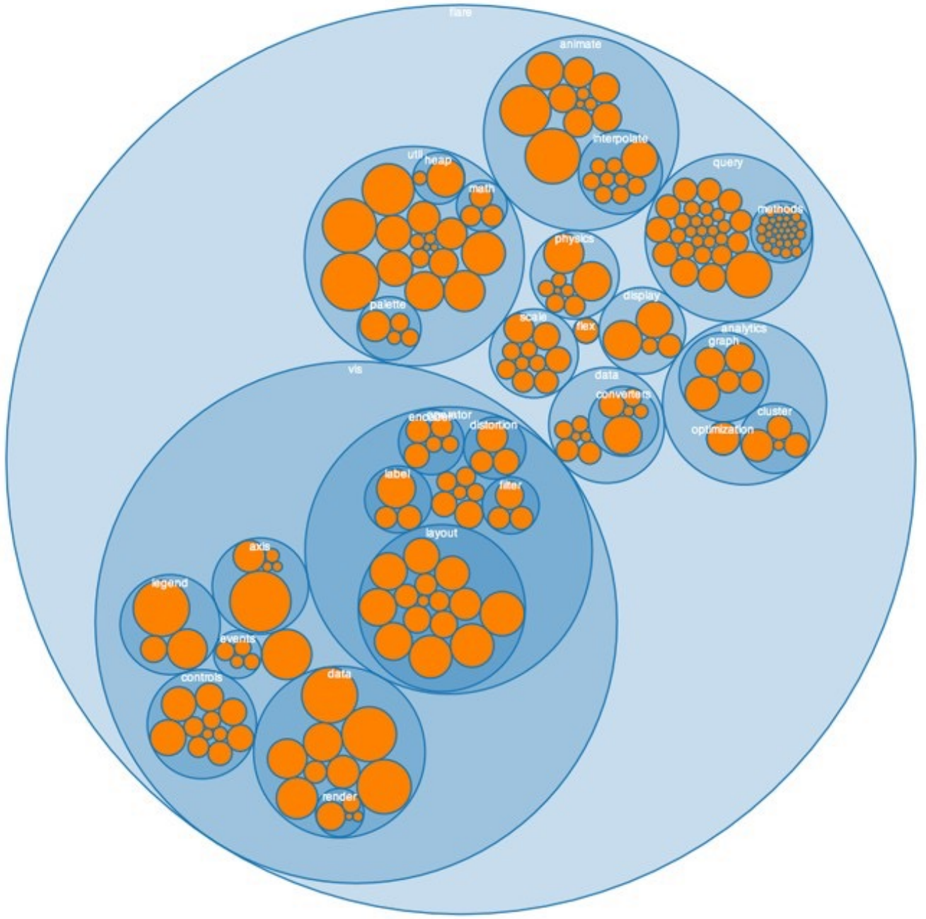
TREEMAPS

- recursively fill space based on a size metric for nodes
- enclosure indicates hierarchy
- additional measures can control aspect ratio of cells
- most often use rectangles, but other shapes possible
  - square, circle, voronoi tessellation









http://hci.stanford.edu/jheer/files/zoo/

‘What Do You Think Is the Most Important Problem Facing This Country Today?’

By GREGOR AISCH and ALICIA PARLAPIANO FEB. 27, 2017

Since the presidency of Franklin D. Roosevelt, the Gallup polling organization has asked Americans an open-ended question: “What do you think is the most important problem facing this country today?”

As Donald J. Trump prepares for his first major address to the nation on Tuesday, he has a unique set of issues to tackle. But there is not one singular issue that is dominating the American consciousness.

January 2015

The biggest problems cited by Americans this month:

https://www.nytimes.com/interactive/2017/02/27/us/politics/most-important-problem-gallup-polling-question.html?smid=pl-share

VISUALIZING GRAPHS

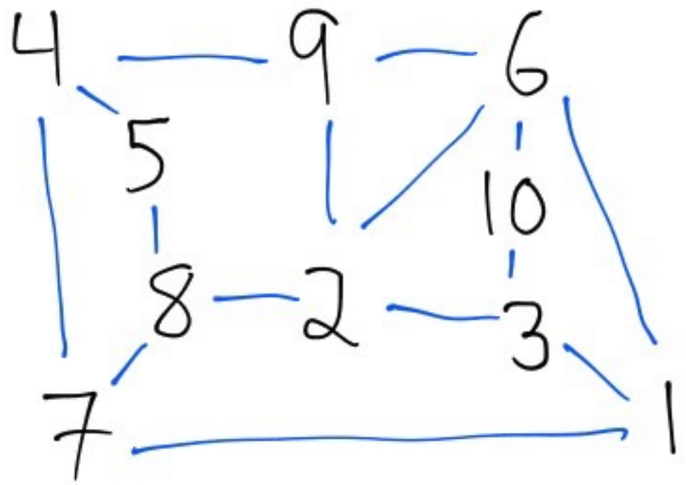
Graph Drawing Exercise

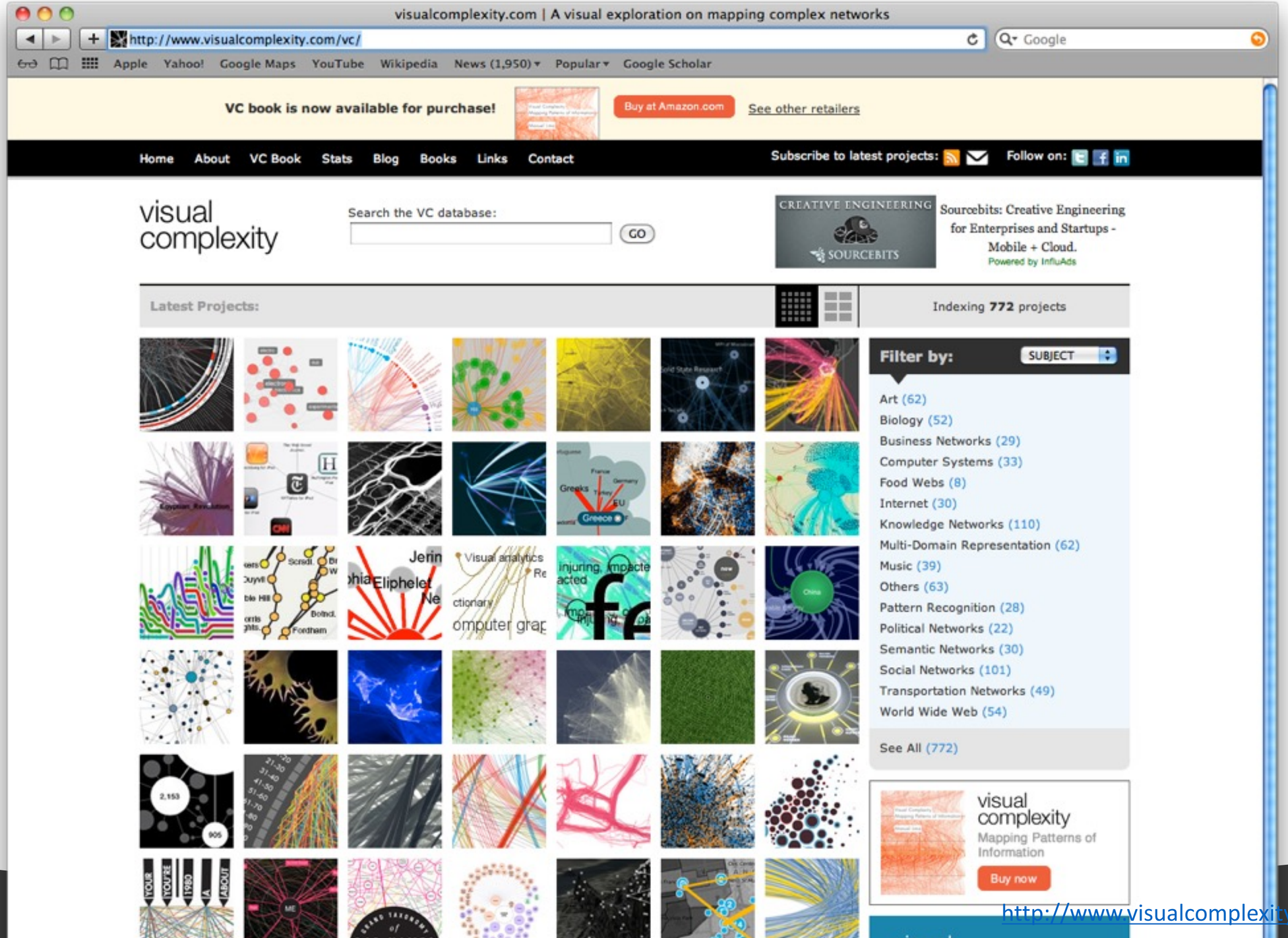
- create an aesthetically pleasing **node-link diagram** representation

	1	2	3	4	5	6	7	8	9	10
1	0	0	1	0	0	1	1	0	0	0
2	0	0	1	0	0	1	0	1	1	0
3	1	1	0	0	0	0	0	0	0	1
4	0	0	0	0	1	0	1	0	1	0
5	0	0	0	1	0	0	0	1	0	0
6	1	1	0	0	0	0	0	0	1	1
7	1	0	0	1	0	0	0	1	0	0
8	0	1	0	0	1	0	1	0	0	0
9	0	1	0	1	0	1	0	0	0	0
10	0	0	1	0	0	1	0	0	0	0

GRAPH DRAWING EXERCISE

- create an aesthetically pleasing **node-link diagram** representation





http://www.visualcomplexity.com/vc/

VISUALIZING GRAPHS

- node link layouts
  - Reingold-Tilford (discussed previously)
  - Sugiyama (directed acyclic graphs)
  - Force directed
  - Attribute-based
- adjacency matrices
- aggregate views
  - Motif Glyphs
  - PivotGraph

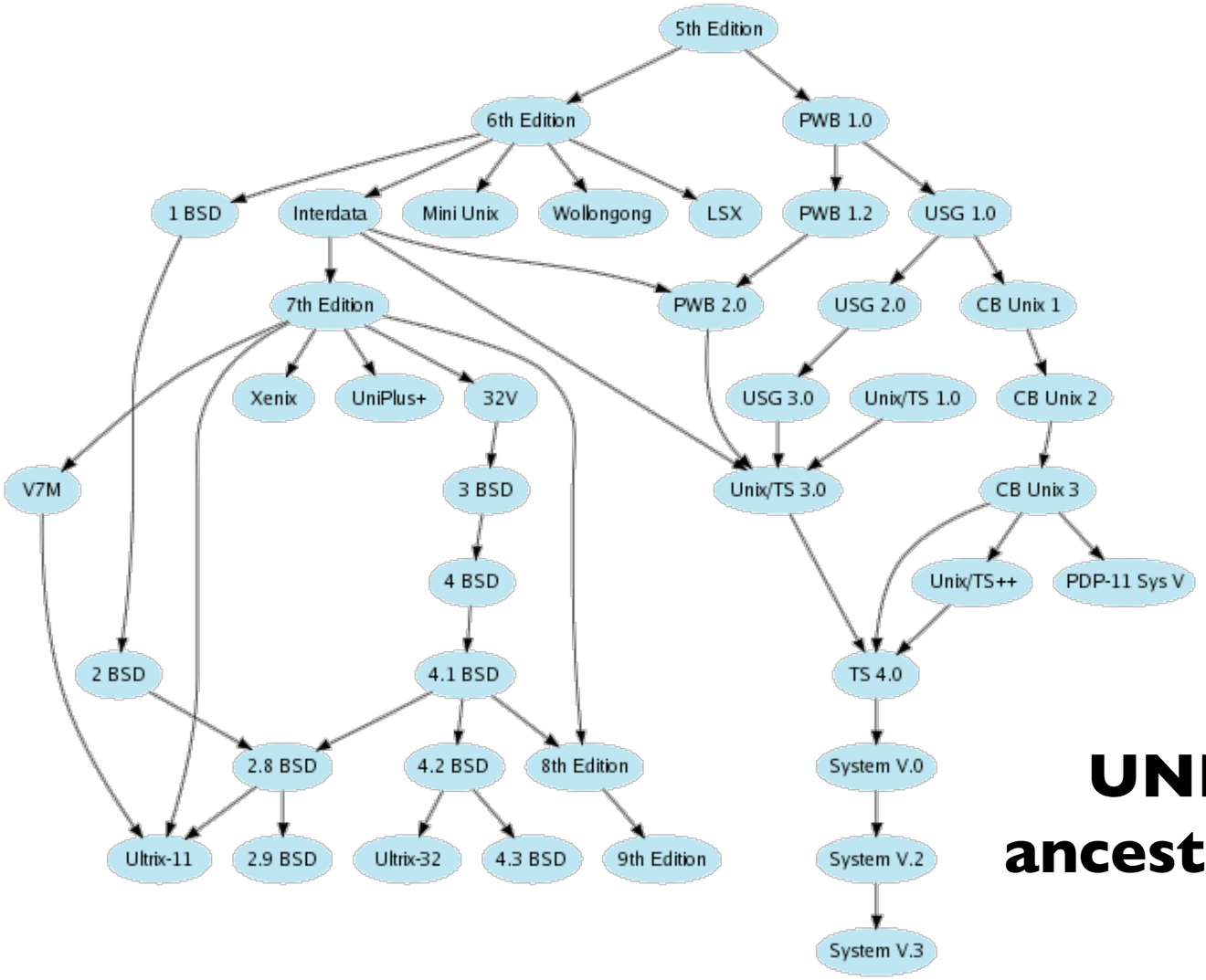
SPATIAL LAYOUT

- primary concern of graph drawing is the spatial layout of nodes and edges
- often (but not always) the goal is to effectively depict the graph structure
  - connectivity, path-following
  - network distance
  - clustering
  - ordering (e.g., hierarchy level)

SUGIYAMA

- great for graphs that have an intrinsic ordering
- depth not strictly encoded

- What is the depth of V7M?



UNIX ancestry

SUGIYAMA

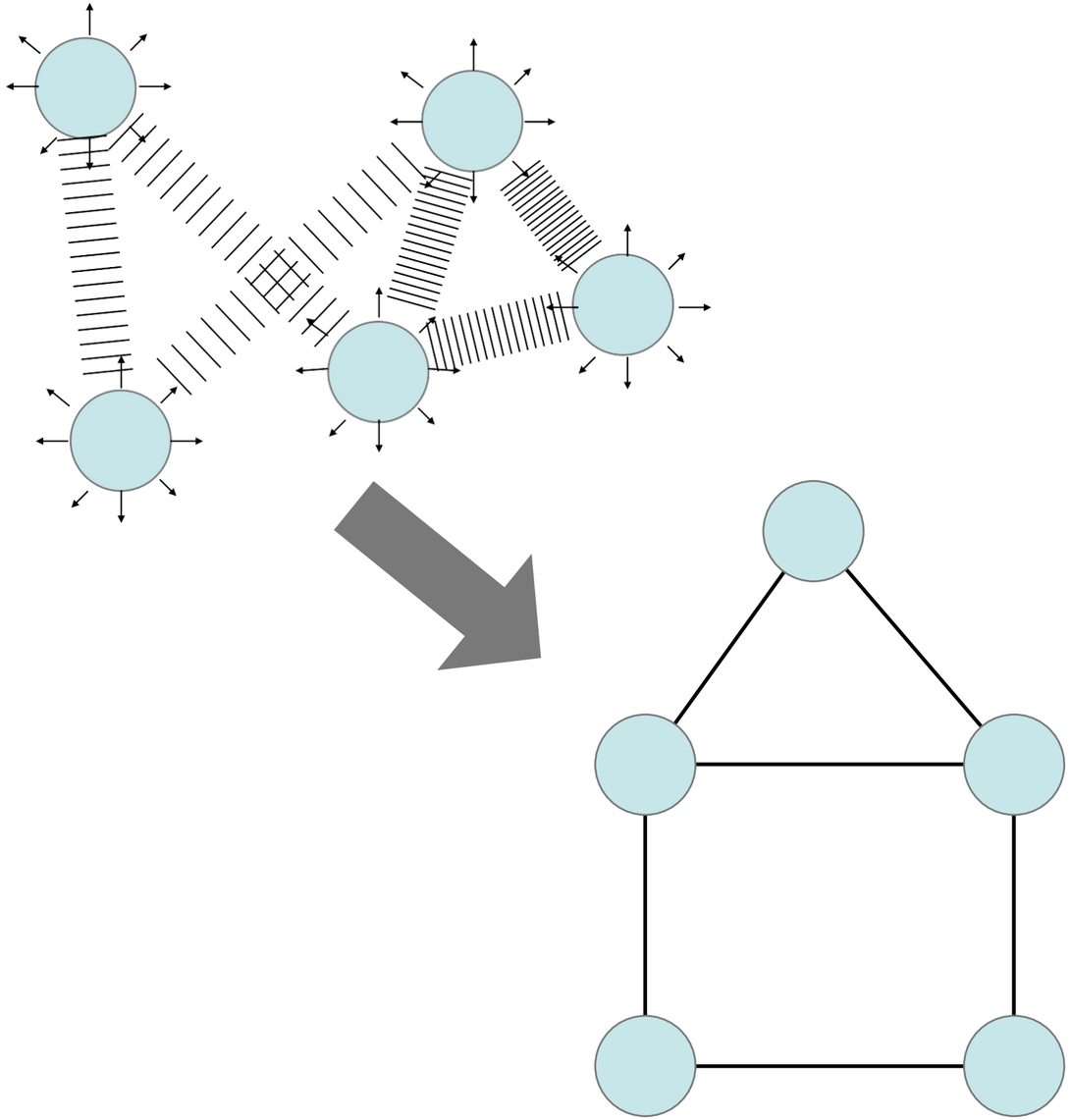
- + nice, readable top down flow
- + relatively fast (depending on heuristic used for crossing minimization)

- - not really suitable for graphs that don’t have an intrinsic top down structure
- - hard to implement

- use free graphviz lib: http://www.graphviz.org

FORCE-DIRECTED

- no intrinsic layering, now what?
- physically-based model

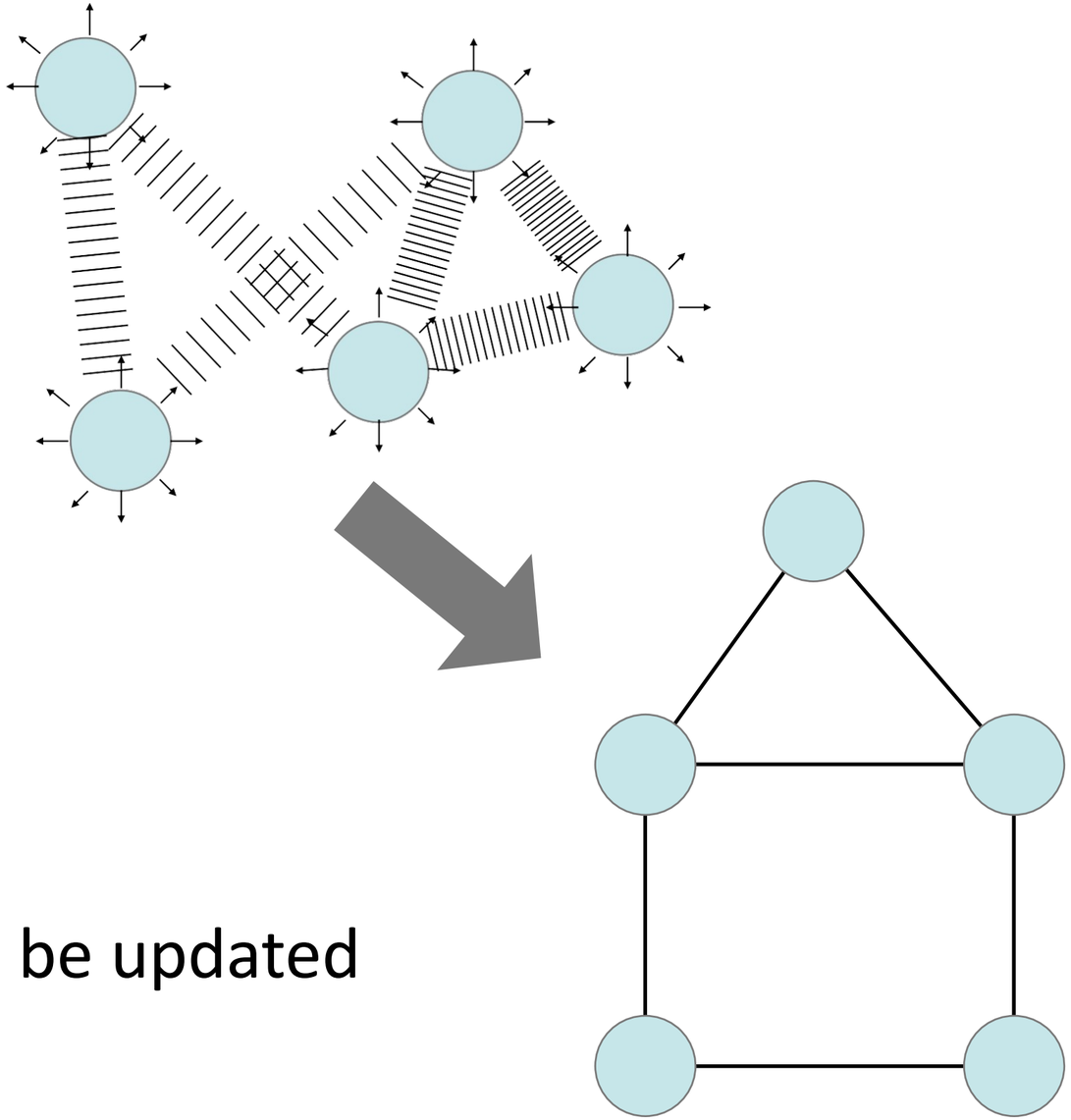


FORCE-DIRECTED

- many variations, but usually physical analogy of repulsion and attraction

- Generally…
  - edges = springs
  - nodes = repulsive particles

- Requires an iterative calculation, should be updated each time the draw loop is called



Physics Review

- $F = ma$
  - (Force = mass * acceleration)
- $\Delta v = a\ \Delta t$
  - (change in velocity = acceleration * time step)
- $p' = p + v\ \Delta t + \frac{1}{2} a \Delta t^2$
  - (new position = old position + velocity * time step)

Physics Review

- $a' = \frac{F}{m}$
  - (acceleration = Force / mass)
- $v' = v + a'\,\Delta t$
  - (new velocity = old velocity + acceleration * time step)
- $p' = p + v'\Delta t + \frac{1}{2}a'\Delta t^2$
  - (new position = old position + new velocity * time step)

For a Given Node i

- $\overrightarrow{a_i'} = \frac{\overrightarrow{F_i}}{m_i}$
  - (acceleration = Force / mass)

- $\overrightarrow{v_i'} = \overrightarrow{v_i} + \overrightarrow{a_i'}\Delta t$
  - (new velocity = old velocity + acceleration * time step)

- $\dot{p_i'} = \dot{p_i} + \overrightarrow{v_i'}\Delta t + \frac{1}{2}\overrightarrow{a_i'}\Delta t^2$
  - (new position = old position + new velocity * time step + ½ acceleration* time step2)

FORCE MODEL

- many variations, but usually physical analogy of repulsion and attraction

- Every node feels repulsion (or attraction) to every other node

FORCE MODEL: Repulsive Forces

- $f_R(d) = \frac{C_R\, m_1\, m_2}{d^2}$
  - C_R is a strength constant
  - $m_1,\ m_2$ are node masses
  - d is a distance between nodes

FORCE MODEL

- Repulsive force:
  - $F_R(P) = \Sigma_{\text{all neighbors(Q)}} f_R(||Q-P||) * (Q-P)$

- Attractive force:
  - $F_A(P) = \Sigma_{\text{connected neighbors (Q)}} f_A(||P-Q||) * (P-Q)$

FORCE MODEL

- Every node feels repulsion to every other node
- Only **connected** nodes feel attracted

FORCE MODEL: Attractive Forces

- $f_A(d) = C_A * (d - L)$
  - C_A is a strength constant
  - d is a distance between nodes
  - L is the rest length of the spring (i.e. Hooke's Law)

FORCE MODEL: Attractive Forces

- $f_A(d) = \mathrm{C_A} \cdot max(0, \mathrm{d} - \mathrm{L})$
  - C_A is a strength constant
  - d is a distance between nodes
  - L is the rest length of the spring (i.e. Hooke's Law)

FORCE MODEL

- Use force to update acceleration
- Use acceleration to update velocity
- Use velocity and acceleration to update position

ALGORITHM

- start from random layout
- (global) loop:
  - for every node pair compute repulsive force
  - for every edge compute attractive force
  - accumulate forces per node and update velocity
  - update each node position in direction of velocity
- stop when layout is ‘good enough’

What values do we set for…

- time step Δt ?
- initial position p_i ?
- initial velocity v_i ?
- mass m_i ?
- Force F_i ?

What values do we set for…

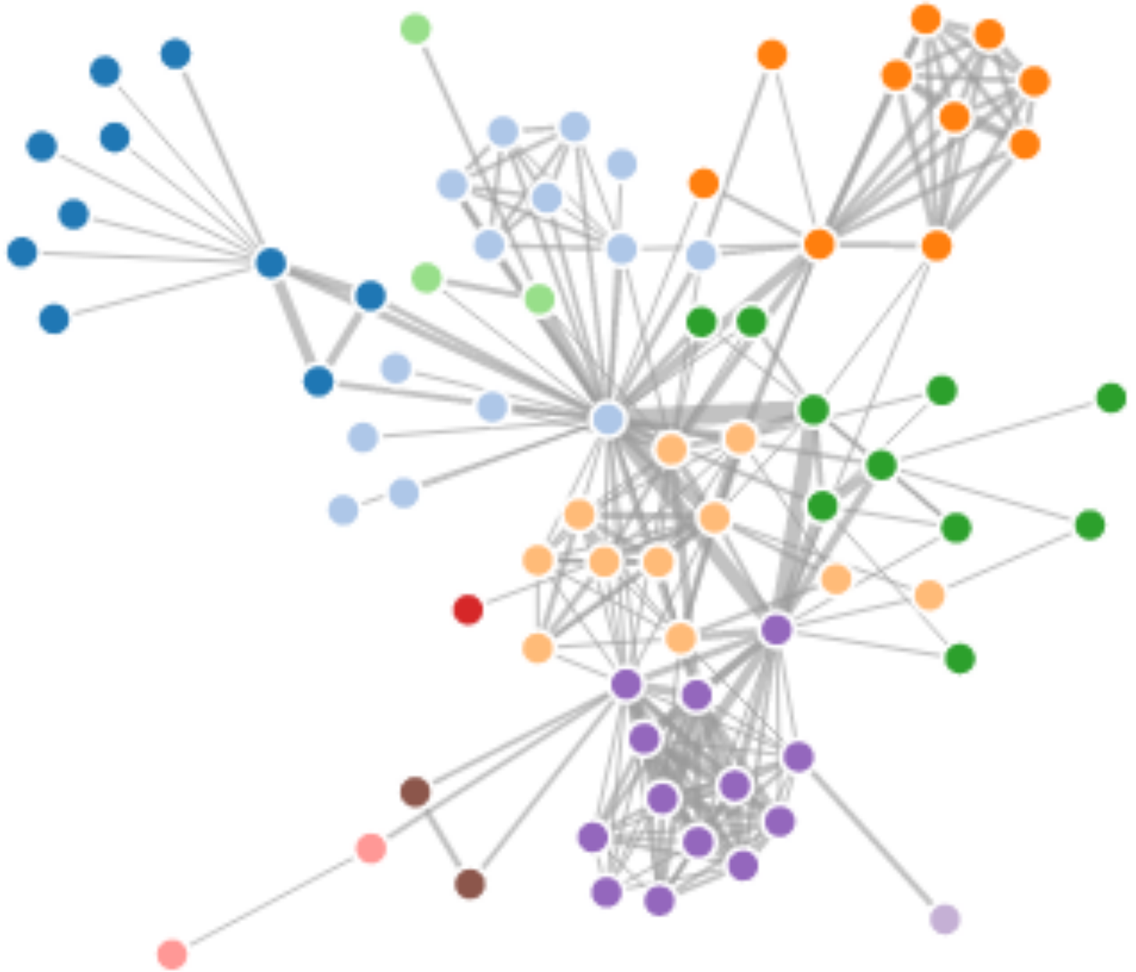
- time step Δt ?
  - Fixed timestep, time since last frame was drawn, …
- initial position p_i ?
  - Start with a random position
- initial velocity v_i ?
  - Zero
- mass m_i ?
  - Depends, however, the heavier it is, the slower it moves
- Force F_i ?
  - That is what we still need to calculate…

What values for…

- Repulsive constant C_R ?
- Attractive constant C_A ?
- Rest length of the spring L ?

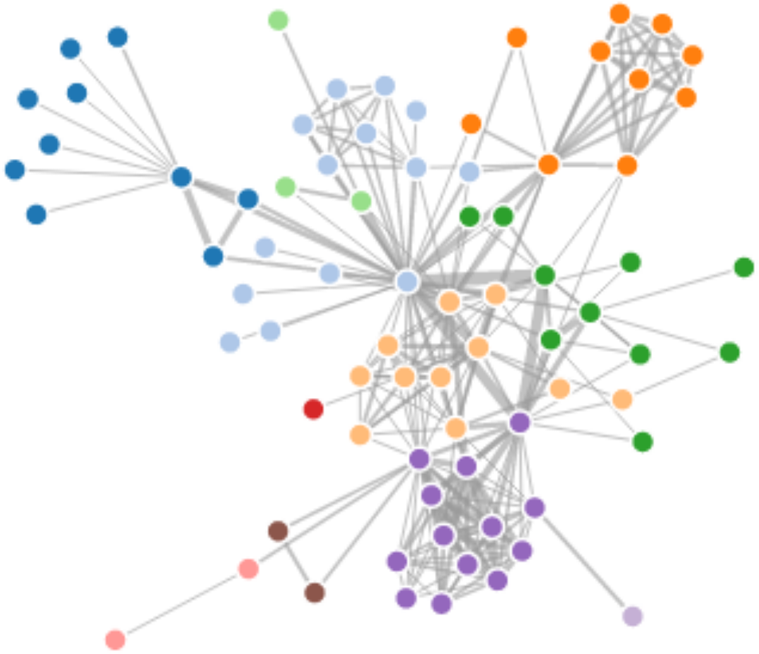
What values for…

- Repulsive constant C_R ?
  - Start with something small (weaker force)
- Attractive constant C_A ?
  - Start with something small (weaker force)
- Rest length of the spring L ?
  - Closest you would *like* 2 nodes to be together (they will be closer) – 10-20 pixels is a good start



FORCE DIRECTED

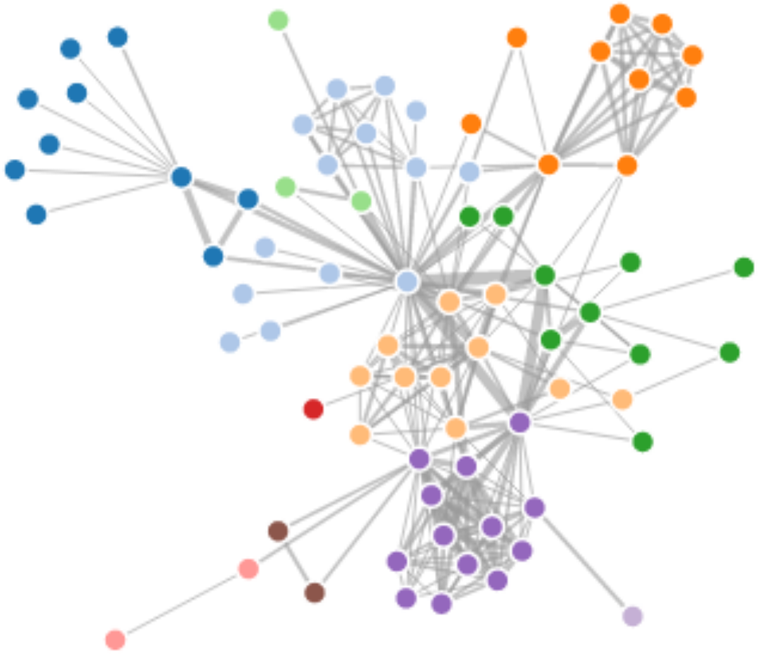
- + very flexible, aesthetic layouts on many types o
- + can add custom forces
- + relatively easy to implement



http://bl.ocks.org/mbostock/4062045

FORCE DIRECTED

- - repulsion loop is O(n2) per iteration
  - can speed up to O(n log n) using quadtree or k-d tree
- - prone to local minima
  - can use simulated annealing
- - doesn’t work well on highly connected (low diameter) graphs



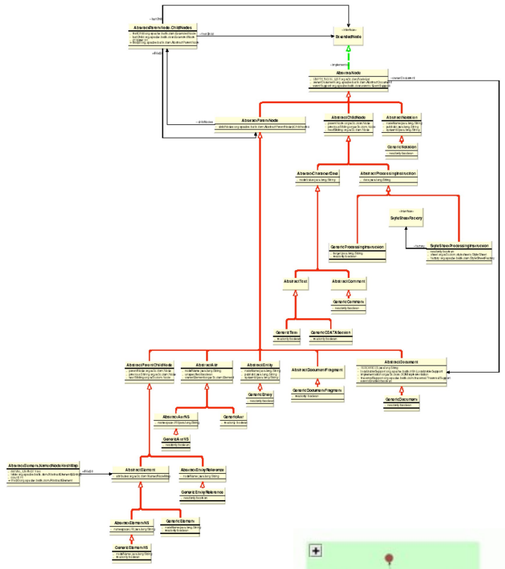
http://bl.ocks.org/mbostock/4062045

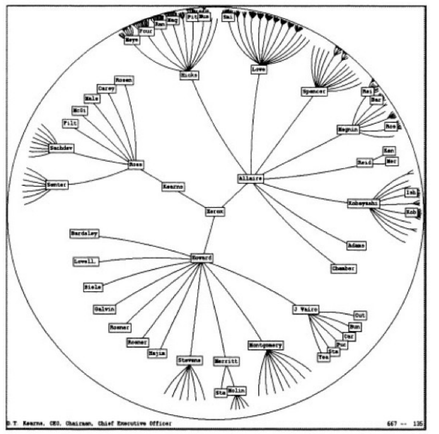
Ideas to make it better

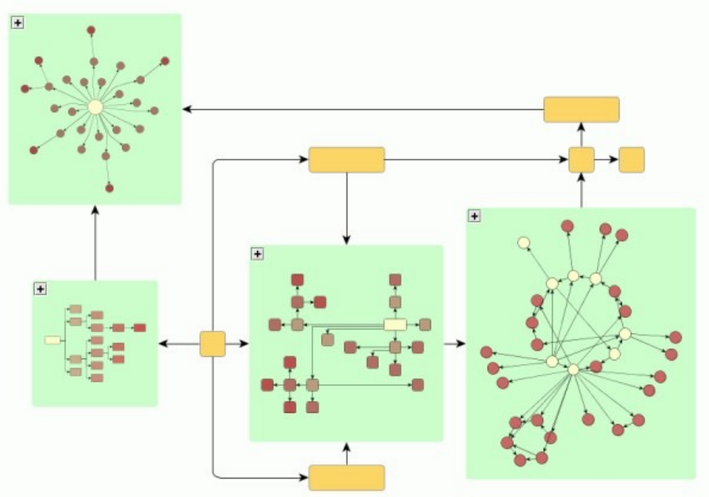
- Add extra forces, such as repulsion from the boundary or attraction to the center of the screen.
- Allow overriding node positions using the mouse (dragging vertices)
- Allow fixing the position of certain nodes

OTHER LAYOUTS

- orthogonal
  - great for UML diagrams
  - algorithmically complex
- circular layouts
  - emphasizes ring topologies
  - used in social network diagrams
- nested layouts
  - recursively apply layout algorithms
  - great for graphs with hierarchical structure
- Attribute driven layouts
  - Use “extra data” to help inform layout
  - (more next lecture)







Gephi, an open source graph visualization and manipulation software

http://gephi.org/ — hookes law

Apple Yahoo! Google Maps YouTube Wikipedia News (1,949) Popular Google Scholar

Gephi
makes graphs handy

Download Blog Store Wiki Forum Support Bug tracker

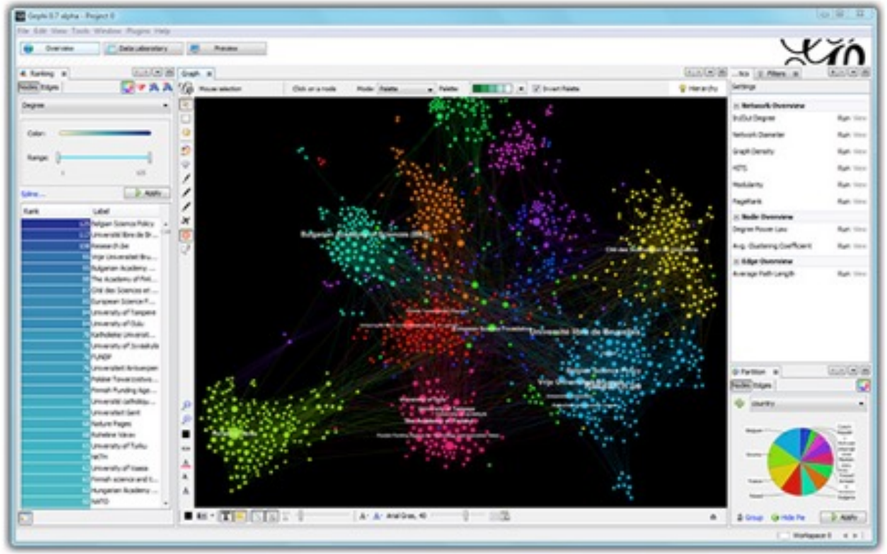
Home Features Learn Develop Plugins Consortium

The Open Graph Viz Platform

Gephi is an interactive visualization and exploration platform for all kinds of networks and complex systems, dynamic and hierarchical graphs.

Runs on Windows, Linux and Mac OS X. Gephi is open-source and free.

Learn More on Gephi Platform »



Download FREE
Gephi 0.8 beta

Release Notes | System Requirements

- Features
- Quick start
- Screenshots
- Videos

Gephi 0.8 beta has been released! Discover a new Preview and dynamic features, start building commercial applications with the new open source license.

Learn More »

APPLICATIONS

- **Exploratory Data Analysis**: intuition-oriented analysis by networks manipulations in real time.
- **Link Analysis**: revealing the underlying structures of associations between objects, in particular in scale-free networks.
- **Social Network Analysis**: easy creation of social data connectors to map community organizations and small-world networks.
- **Biological Network analysis**: representing patterns of biological data.
- **Poster creation**: scientific work promotion with hi-quality printable maps.

Learn More »

“Like Photoshop™ for graphs.

— the Community

LATEST NEWS

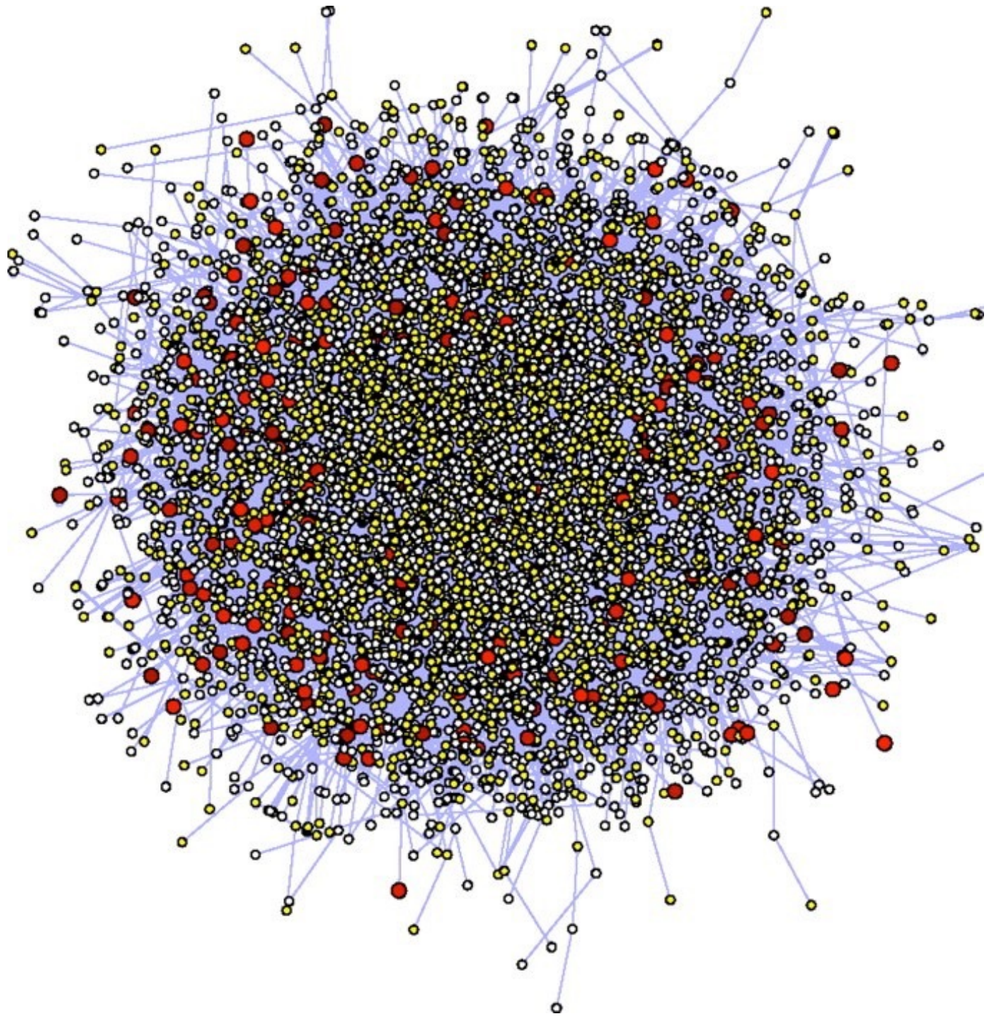
- Weekly news
  February 27, 2012
- Annual report 2011
  February 25, 2012
- Gephi-Neo4j presentation at FOSDEM
  February 20, 2012
- Gephi meet-up #4 in Berlin
  February 2, 2012
- Introducing the Gephi Plugins Bootcamp
  January 12, 2012

PAPERS

NODE LINK

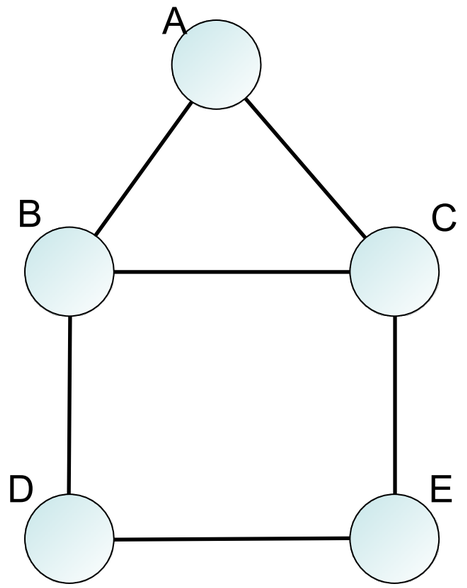
- + understandable visual mapping
- + can show overall structure, clusters, paths
- + flexible, many variations

- - all but the most trivial algorithms are > $O(n^2)$
- - not good for dense graphs
  - hairball problem!



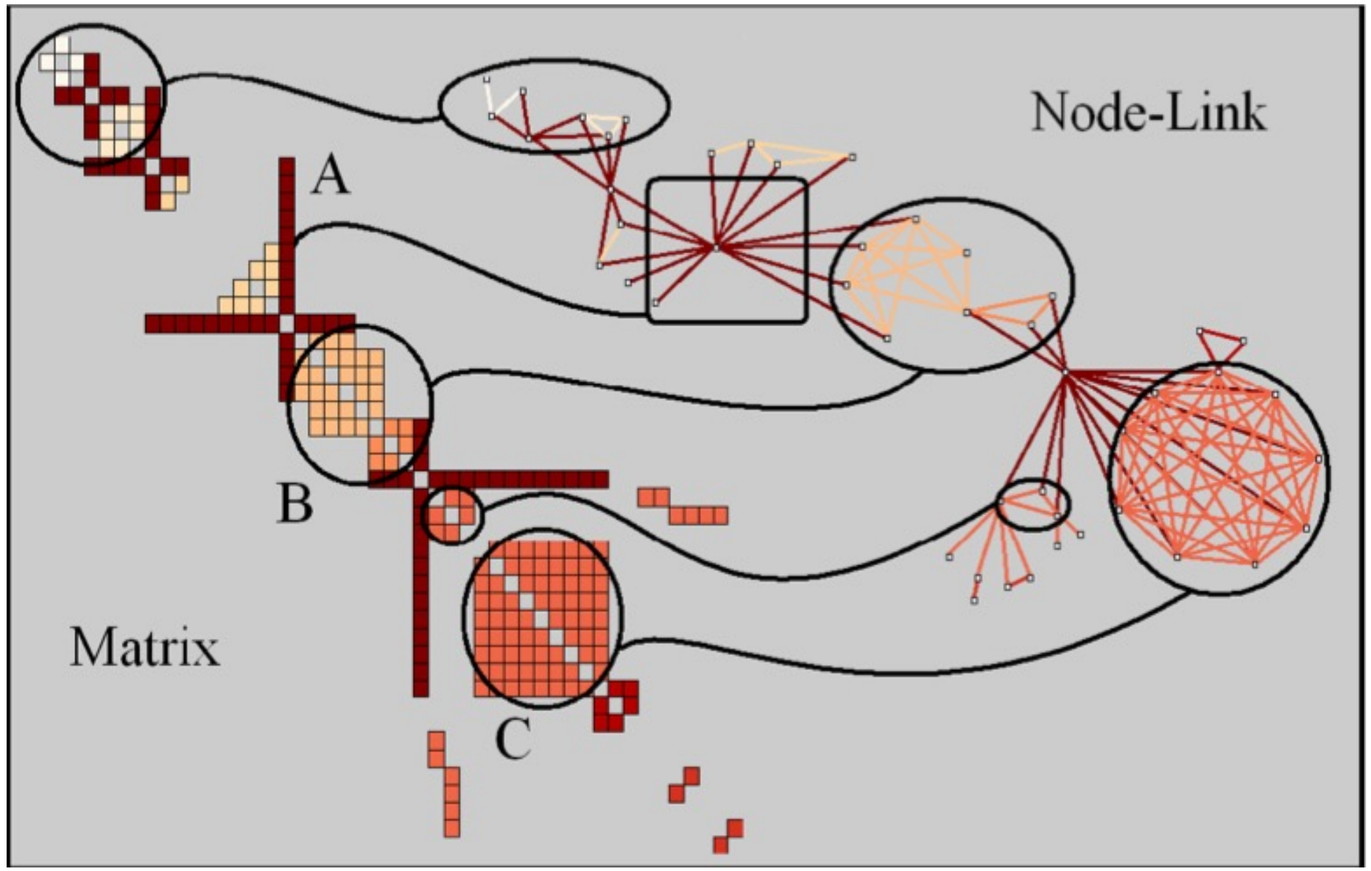
ALTERNATIVE: ADJACENCY MATRIX

- instead of node link diagram, use adjacency matrix representation



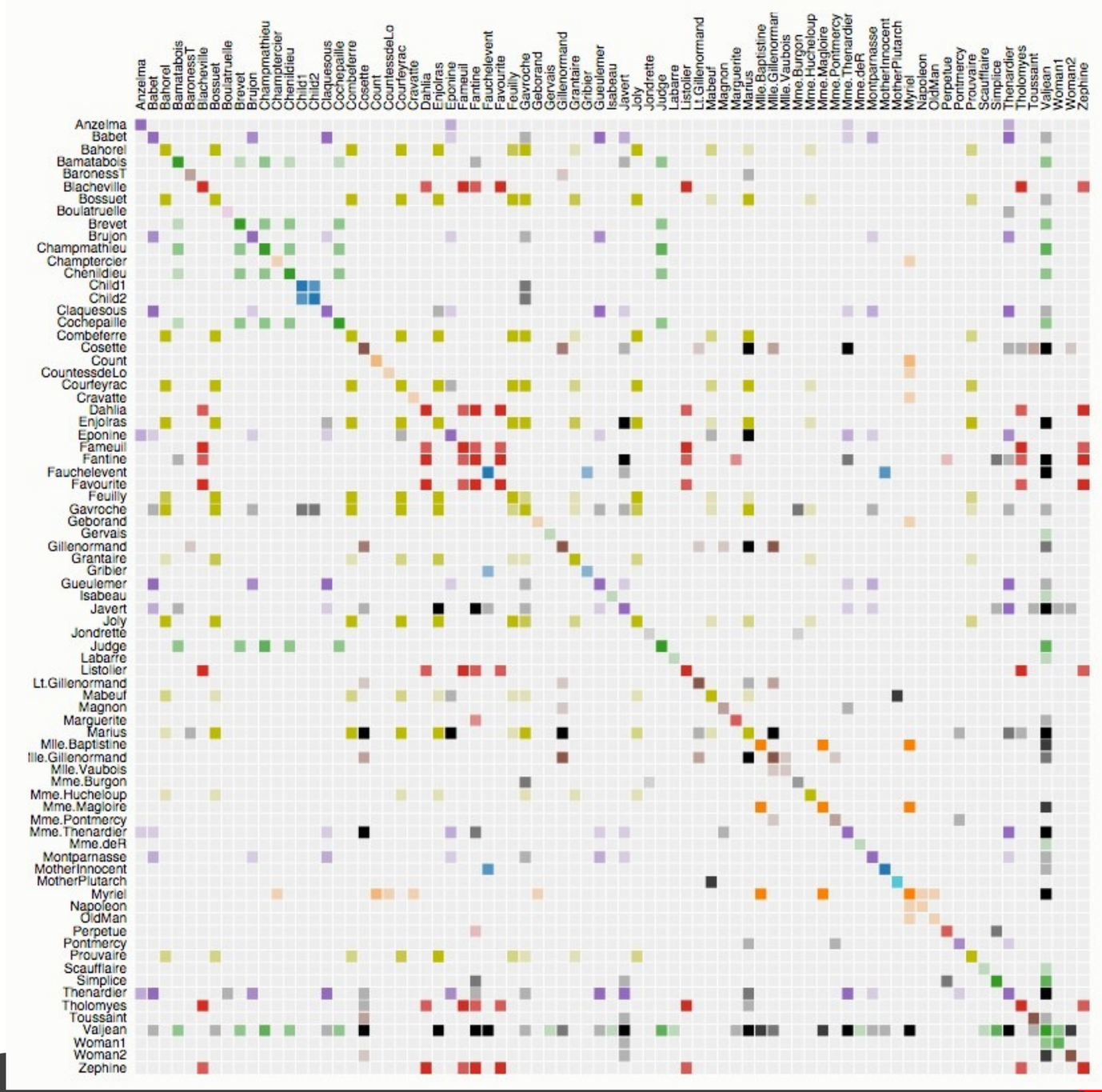
	A	B	C	D	E
A		■	■		
B	■		■	■	
C	■	■			■
D		■			■
E			■	■	

SPOTTING PATTERNS IN MATRICES



Les Misérables

- character co-occurrence

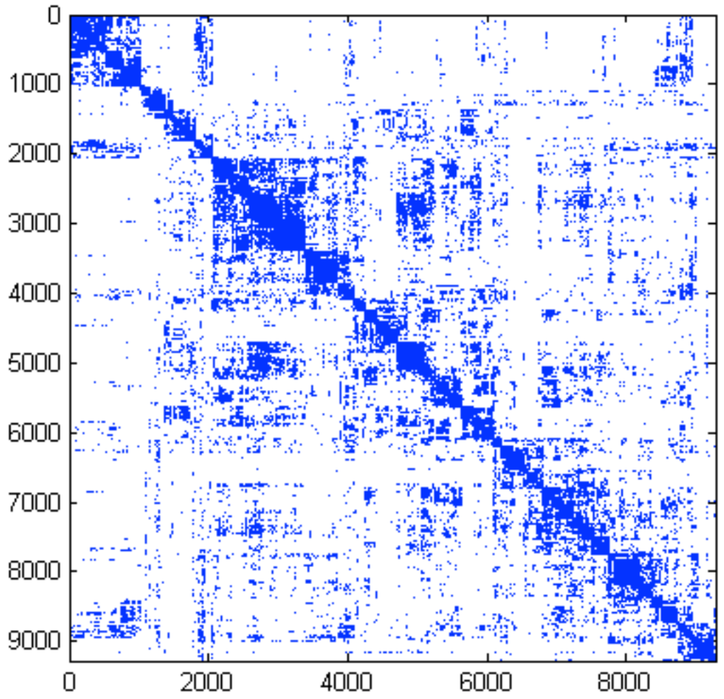


http://bost.ocks.org/mike/miserables/

Adjacency Diagram

- + great for dense graphs
- + visually scalable
- + can spot clusters

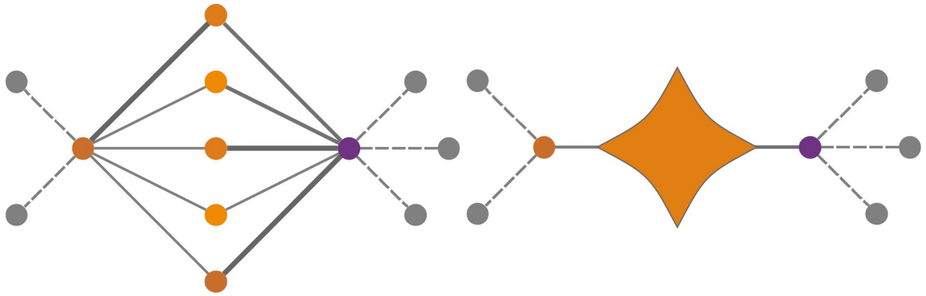
- - row order affects what you can see
- - abstract visualization
- - hard to follow paths



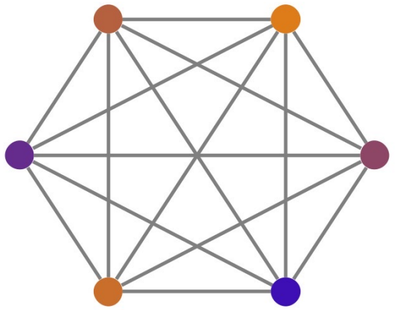
AGGREGATE VIEWS

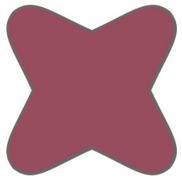
MOTIF GLYPHS

Connector

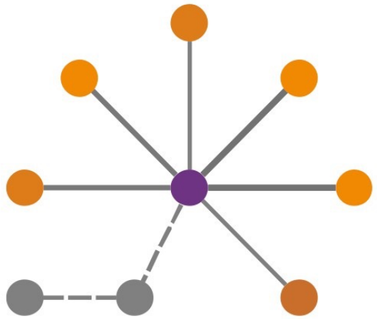


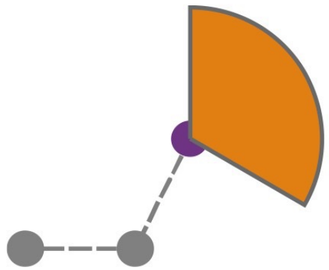
Clique



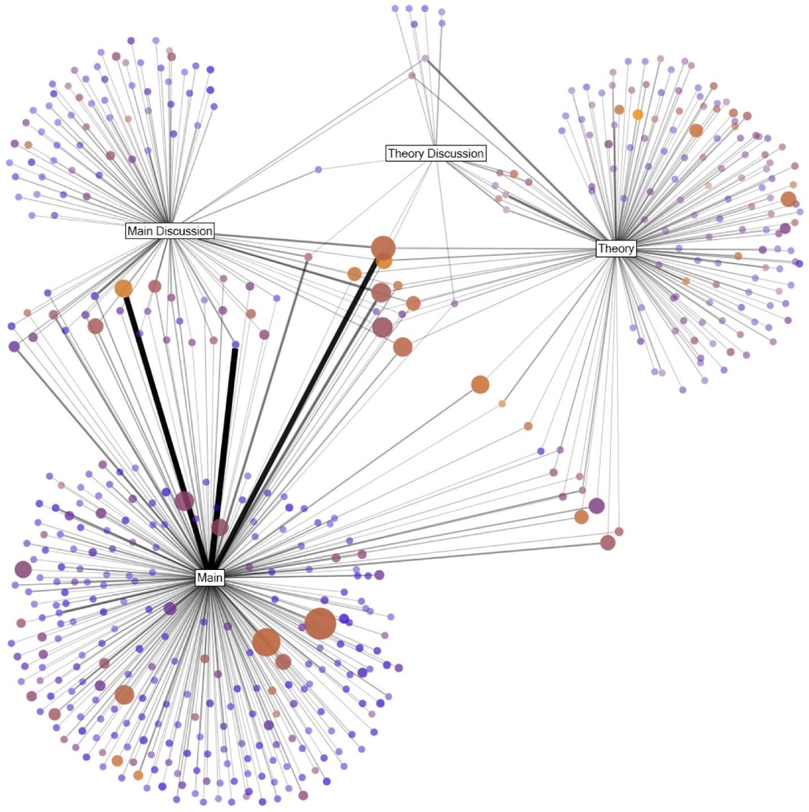


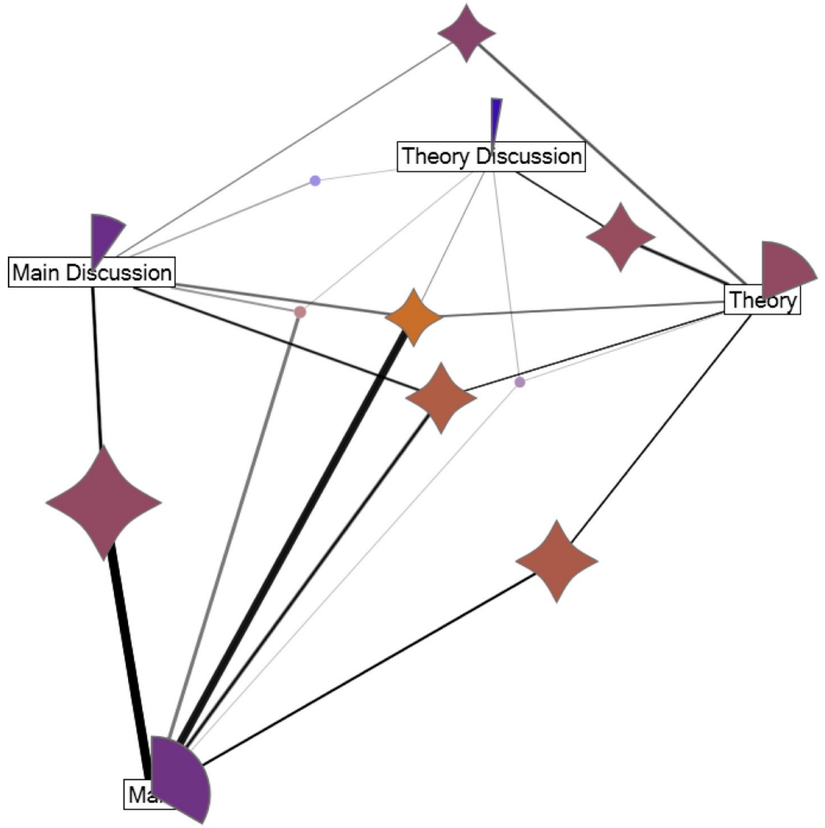
Fan



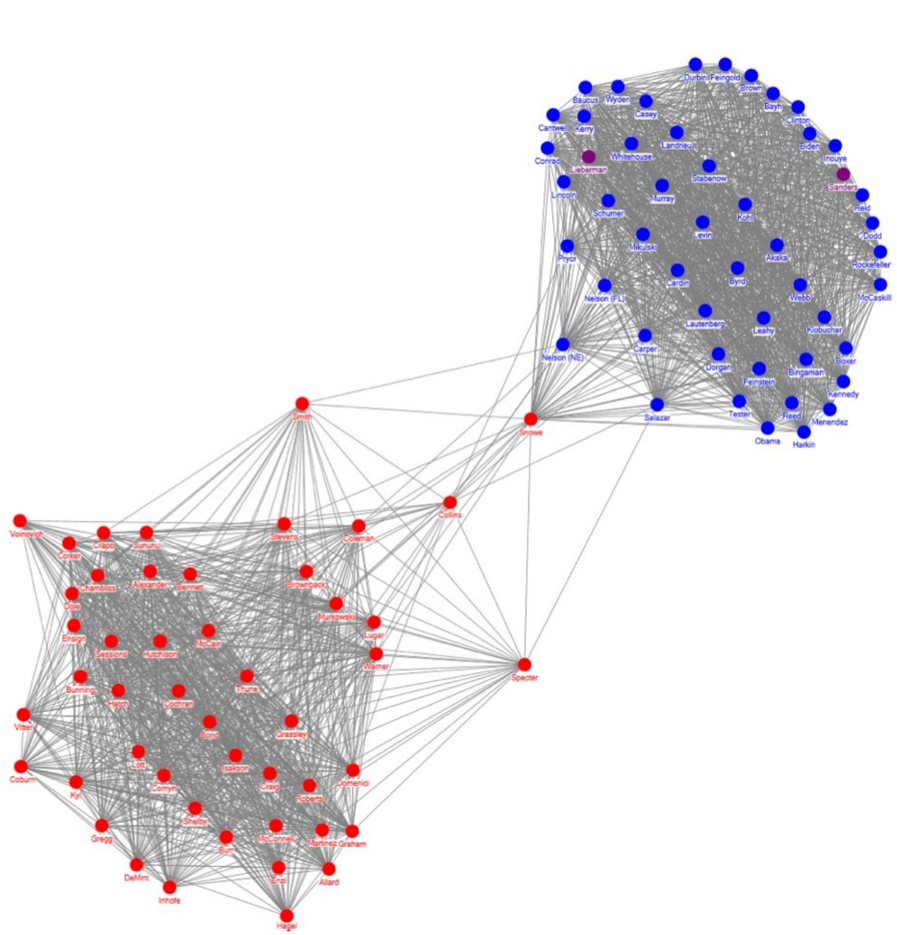


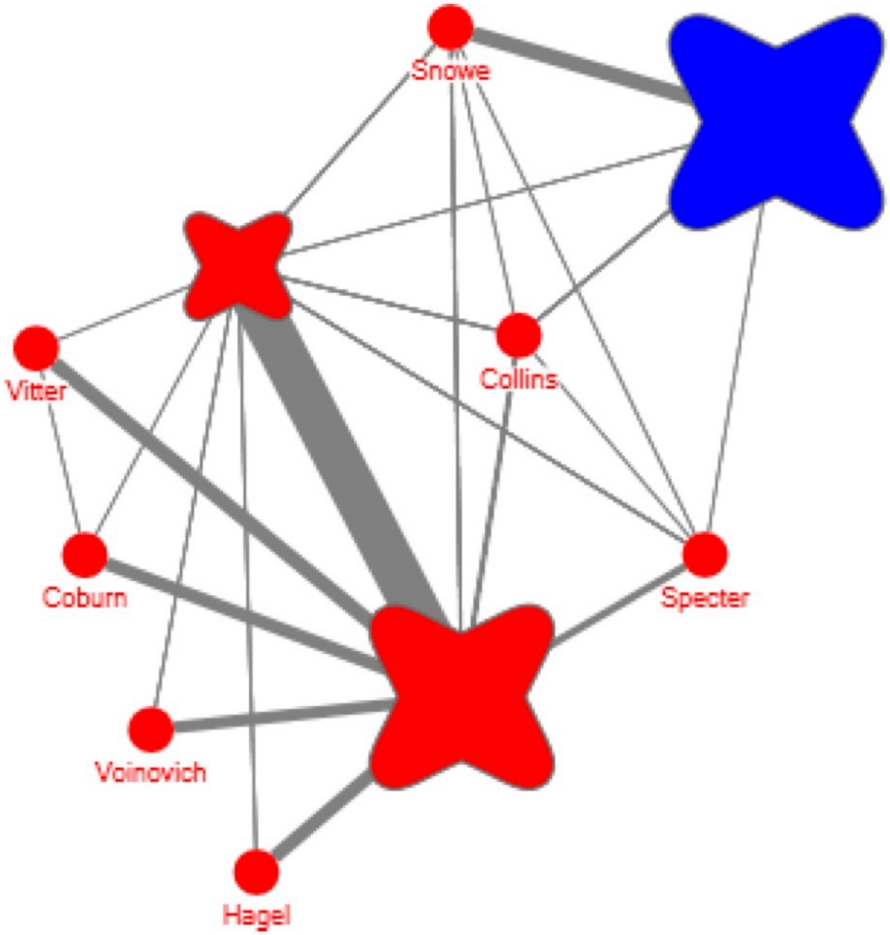
MOTIF GLYPHS





MOTIF GLYPHS





RECAP

TREES

- indentation
  - simple, effective for small trees
- node link and layered
  - looks good but needs exponential space
- enclosure (treemaps)
  - great for size related tasks but suffer in structure related tasks

GRAPHS

- node link
  - familiar, but problematic for large or dense graphs
- adjacency matrices
  - abstract, hard to follow paths
- aggregation can help
  - not always possible, not always appropriate
- extracting structure can help
  - unclear how crosscutting it will be

TAKE HOME MESSAGE
No best solution!