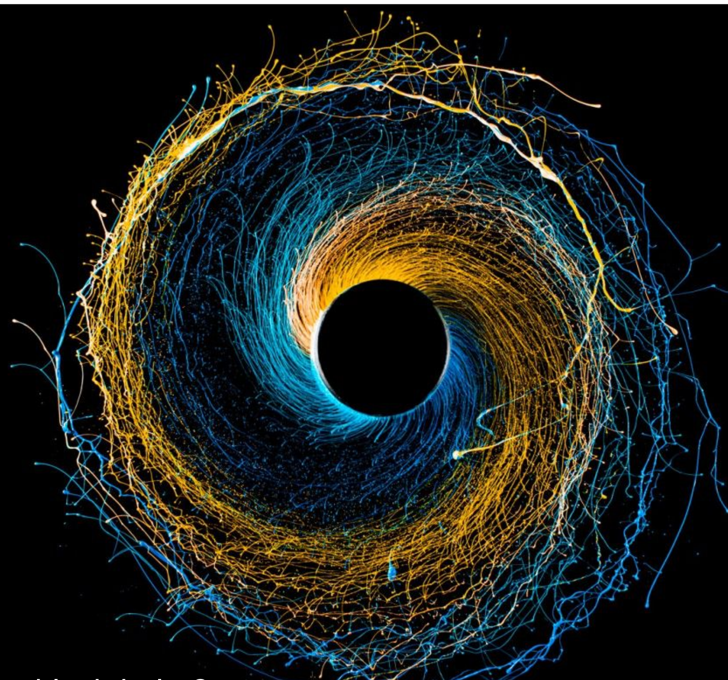
CSCI 4550/6550 Interactive Visualization

https://www.cs.rpi.edu/~cutler/classes/visualization/S24/

Lecture 3: Graph Visualization part 1

"Various shades of acrylic paint are dripped onto a metallic rod, which is connected to a drill. When switched on, the paint starts to move away from the rod."



http://fabianoefner.com/?portfolio=black-hole-2

Miscellaneous Announcements

- My office hours after lecture & Thursdays 1-3pm (Lally 302)
- Fauzan's office hours are… *TBA*
- Reading & worksheet grades are posted in Rainbow Grades
  - *Send me email when you complete your "Discussant" duty so I make sure to enter your grade!*
- Peer Grading for HW1 is in progress!
  - *How is it going?*
  - I will release the HW1 grades (both TA & peers) on Wednesday
- Rainbow Grades are available now and will update nightly
  - If you see something fishy, just ask…

Today

- Homework 2 Discussion/Questions
- Readings for Today
  - "Improved force-directed Layouts"
  - "A Technique for Drawing Directed Graphs"
- Graph Drawing Goals, Questions, & Challenges
- Some Related Terms/Algorithms (mentioned indirectly in the reading)
- Readings for Friday
- Computational Geometry: Closest pair of points

Homework 2: Time-Based Datasets

- Team of 2
- Obtain an interesting time-based dataset
  - Should be collectable* from online sources, and
  - Require a modest effort to prepare*

    ** = you'll submit your scripts/code to document*
- Use Microsoft Excel or Google Sheets or LibreOffice Calc
  - Create a variety (one of each?!) of the charts following the guidelines from "Eenie, Meenie, Minie, Moe: Selecting the Right Graph for Your Message"
  - Excellent labels and captions for each
- Upload your assignment to Submitty by Thursday @ 11:59pm
  And post two of the charts on the forum

Tools for Scraping Data from the Web

- copy-paste
- wget
- grep / sed / awk / sort / uniq
- Favorite programming language to parse/strip out unnecessary html formatting
- Save as .csv (comma separated value) files to upload to Excel / Google Sheets
- Python has lots of packages for parsing (e.g., json format)
- Selenium for automated browsing of websites

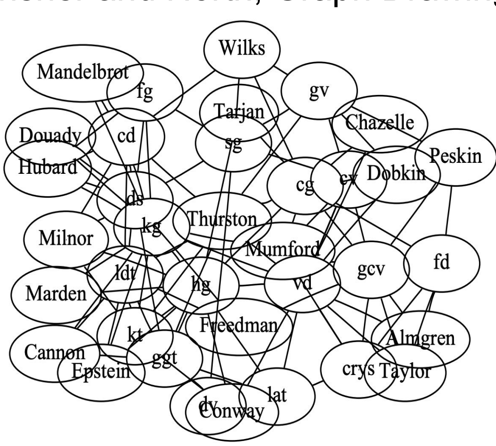
Homework Goal: Everyone learn something (or learn more) about one of these tools (or similar)

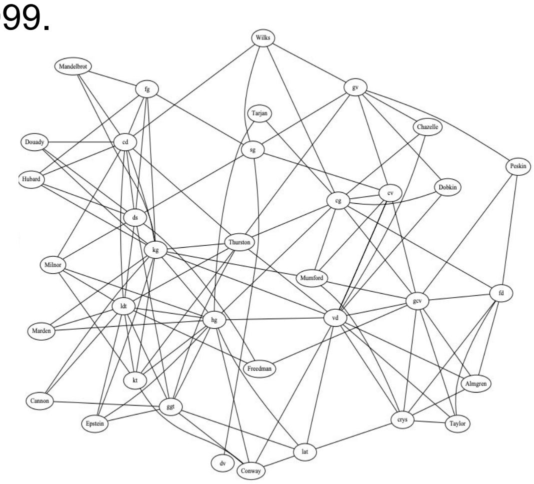
Today

- Homework 2 Discussion/Questions
- Readings for Today
  - “Improved force-directed Layouts”
  - “A Technique for Drawing Directed Graphs”
- Graph Drawing Goals, Questions, & Challenges
- Some Related Terms/Algorithms
  (mentioned indirectly in the reading)
- Readings for Friday
- Computational Geometry: Closest pair of points

Reading for Tuesday *(pick one)*

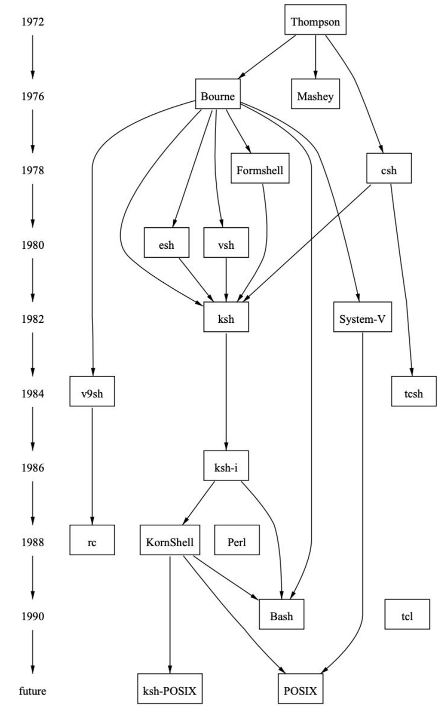
“Improved force-directed layouts”,
Gansner and North, Graph Drawing, 1999.





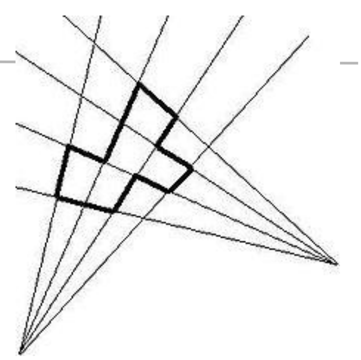
Reading for Tuesday *(pick one)*

"A Technique for Drawing Directed Graphs"
Gansner, Koutsofios, North, & Vo,
IEEE Trans. on Software Engineering, 1993.



Miscellaneous Notes

https://en.wikipedia.org/wiki/Isothetic_polygon#/media/File:Isothet.jpg



- Undirected graphs have too much freedom"
- "Isothetic rectangle"
- Writing
  - "Will be reported in [ ]": are forward references to unpublished work ok?
  - Casual mentions of names of computer programs (written by authors)
  - Related work as 2nd section or as last section: which do you prefer?
  - Jumped straight into pseudocode without much overview / intuition (hard to read)
  - Don't use contractions in formal writing

Today

- Homework 2 Discussion/Questions
- Readings for Today
  - "Improved force-directed Layouts"
  - "A Technique for Drawing Directed Graphs"
- Graph Drawing Goals, Questions, & Challenges
- Some Related Terms/Algorithms
  (mentioned indirectly in the reading)
- Readings for Friday
- Computational Geometry: Closest pair of points

Graph Drawing Goals

- Automated!
- Can read all of the labels (not overlapping, font not too small)
- Can follow the line and see exactly which 2 vertices it connects
- Aesthetically pleasing
- Layout should display as much symmetry as possible
- Crossing free or minimal-crossing layout
- Consistent direction for directed edges
- All edge lengths are approximately equal
- Even vertex distribution
- Distance between nodes in final layout should be as close as possible to "graph distance" (# of edges on shortest path between those nodes)

Graph Drawing Questions

- What is the metric of success for each of our goals?
- Can we guarantee to find a solution? The optimal or best solution?
- Can we use randomness? Does it help?
- How expensive/slow are the different algorithms to draw graphs?
- How does it scale with more nodes/edges?
  - Does it lose effectiveness in meeting our goals?
  - How is the running time affected?
- How do we label the nodes/edges with color/words/images?
- *Is there still use for graph drawing tools for data with 40-100 nodes? Or should we focus exclusively on modern, "big data" datasets?*

Graph Drawing Challenges

- What if the graph is non planar?
- What if the graph has many nodes & edges?
  - ~40-100 works well for simple force-based methods
  - Is # of springs = # of edges?
    Or is # of springs ≫ # of edges?
  - Computation & convergence & getting stuck in a local minimum
- Does 3D (or 4D or …) or layout on the surface of a sphere or torus or … non Euclidean space help?
- *Does adding interaction help? Are high quality static layout tools necessary for building a high quality interactive graph visualization?*

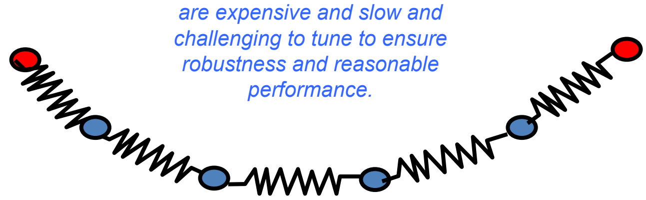
Today

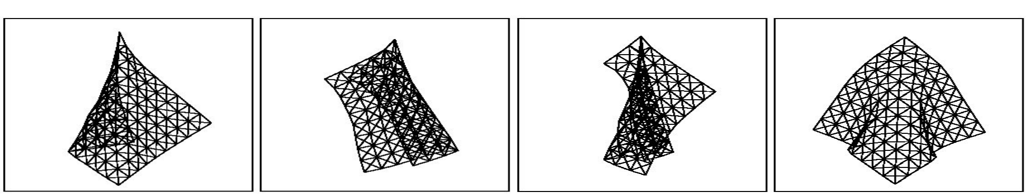
- Homework 2 Discussion/Questions
- Readings for Today
  - “Improved force-directed Layouts”
  - “A Technique for Drawing Directed Graphs”
- Graph Drawing Goals, Questions, & Challenges
- Some Related Terms/Algorithms (mentioned indirectly in the reading)
- Readings for Friday
- Computational Geometry: Closest pair of points

String/Hair/Cloth Simulation

- Springs link the particles
- Springs try to keep their rest lengths and preserve the length of the string

Note: mass-spring simulations are expensive and slow and challenging to tune to ensure robustness and reasonable performance.





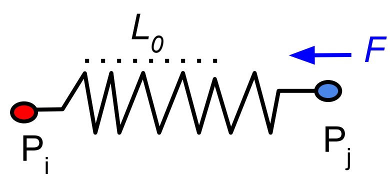
Interactive Animation of Structured Deformable Objects Desbrun, Schröder, & Barr 1999

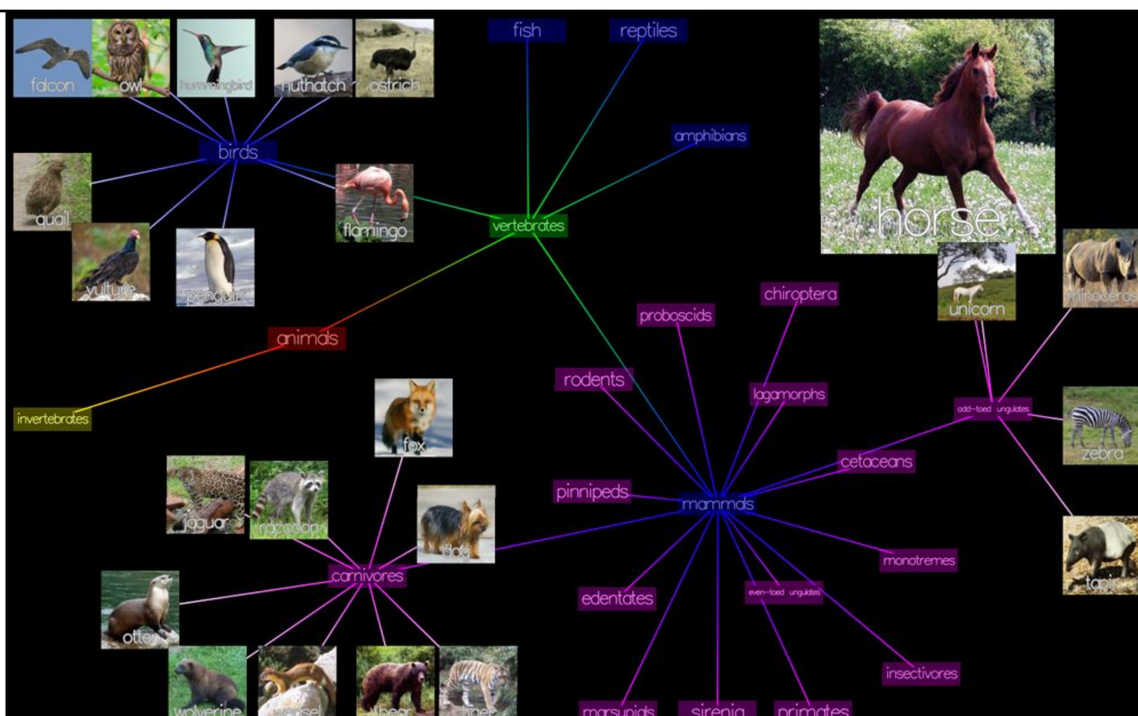
Spring Forces

- Force in the direction of the spring and proportional to difference with rest length L_0

$$F(P_i, P_j) = K(L_0 - ||\vec{P_iP_j}||)\frac{\vec{P_iP_j}}{||\vec{P_iP_j}||}$$

- K is the stiffness of the spring
  - When K gets bigger, the spring really wants to keep its rest length





Using Springs for Graph Drawing

- Value for spring rest length?
  - Rest length = 0 - *springs only attract*, or
  - Springs both attract & repel (non-zero edge length), or
  - Rest length = infinity - *springs only repel*
- What is the correct spring constant?
  - Too high/stiff → *system explodes (does not converge)*
  - Too low → *takes too long to converge*

exerting attractive and repulsive forces from one another." The attractive and repulsive forces are redefined to

$$f_a(d) = d^2/k, \qquad f_r(d) = -k^2/d,$$

in terms of the distance d between two vertices and the optimal distance between vertices k defined as

$$k = C\sqrt{\frac{area}{number\ of\ vertices}}.$$

Writing Note: Algorithms with arbitrary constants

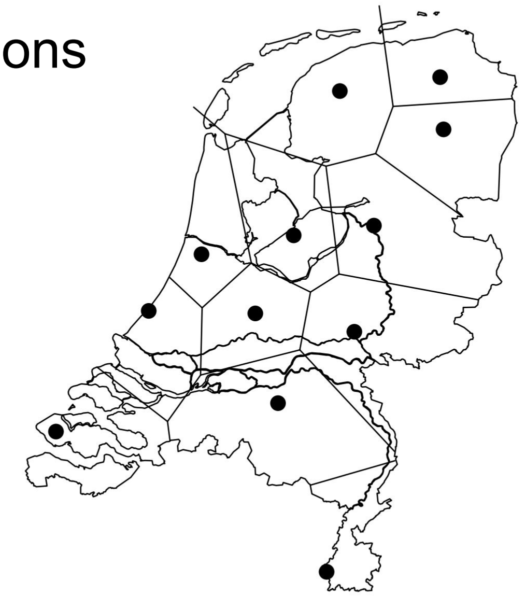
- *A red flag?* Algorithm might not be sufficiently general or robust.
- But honest and complete documentation is necessary for the work to be reproducible*, and improved in future.*

```
algorithm SPRING(G:graph);
place vertices of G in random locations;
repeat M times
    calculate the force on each vertex;
    move the vertex c4 * (force on vertex)
draw graph on CRT or plotter.
```

The values $c_1 = 2$, $c_2 = 1$, $c_3 = 1$, $c_4 = 0.1$, are appropriate for most graphs. Almost all graphs achieve a minimal energy state after the simulation step is run 100 times, that is, $M = 100$.

Voronoi Diagram/Cells/Regions

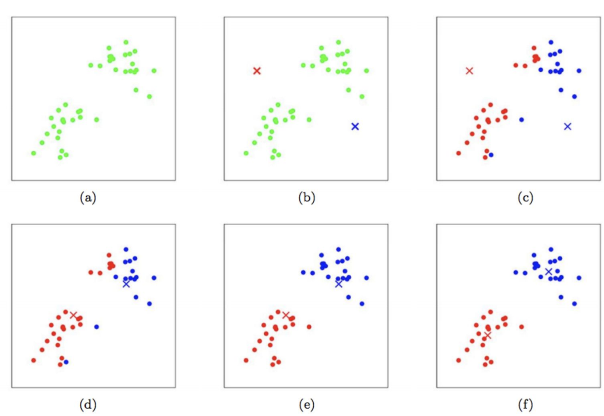
- How to re-district the Netherlands into provinces so that everyone reports to the closest capital
- Cell edges are the perpendicular bisectors of nearby points
- 2D or 3D
- Supports efficient *Nearest Neighbor* queries



Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 7

K-Means Clustering *Same/Similar to: Lloyd's Algorithm*

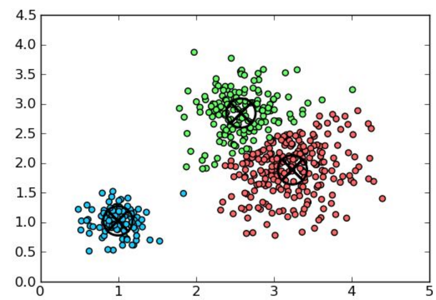
- For a set of 2D/3D/*n*D points:
- Choose *k*, # of clusters (maybe an "oracle" tells us...)
- Select *k* points from your data at random as initial team representatives
- Every other point determines which team representative it is closest to and joins that team
- The team averages the positions of all members, this is the team's new representative
- Repeat *x* times or until change < threshold



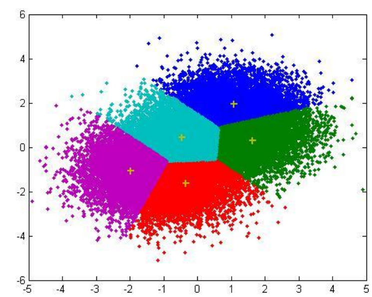
Wei Zhang
https://wei2624.github.io/MachineLearning/usv_kmeans/

K-Means Clustering

- Works quite well, when the data can be meaningfully classified (and we know how many clusters to use).
- With dense data, output is visually similar to Voronoi diagram (k-Means chooses the data points that define the cells)



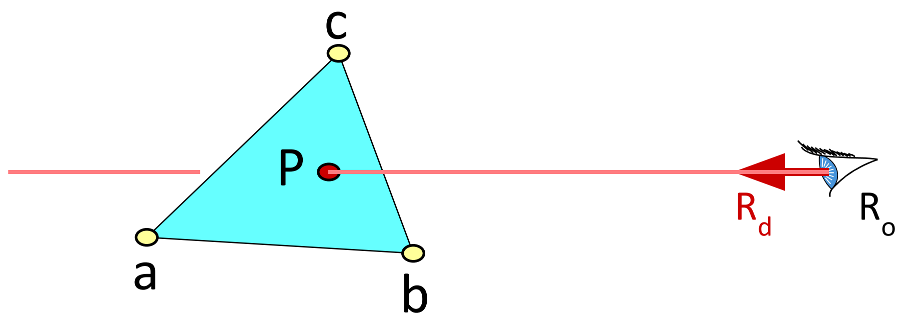
http://blog.mpacula.com/2011/04/27/k-means-clustering-example-python/



"Efficient K-Means Clustering using JIT" Yi Cao

Barycentric Coordinates

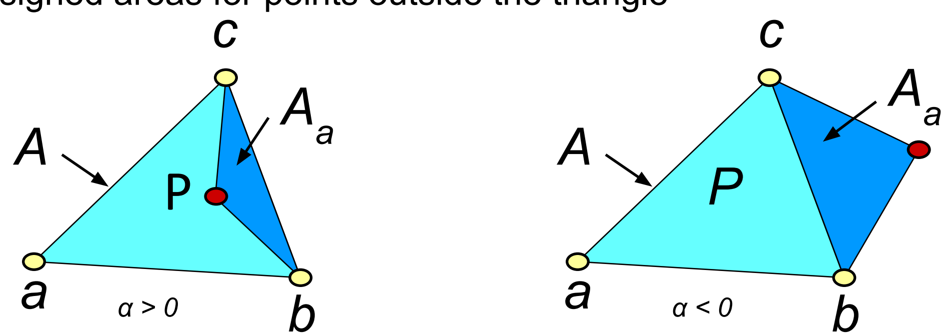
- $P(\alpha, \beta, \gamma) = \alpha a + \beta b + \gamma c$
  with $\alpha + \beta + \gamma = 1$
- If $0 < \alpha < 1$ & $0 < \beta < 1$ & $0 < \gamma < 1$
  then the point is inside the triangle!



How Do We Compute α, β, γ ?

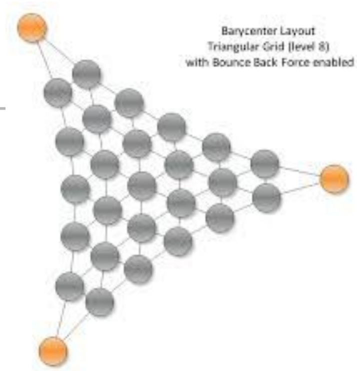
- Ratio of opposite sub-triangle area to total area

  $\alpha = A_a/A \qquad \beta = A_b/A \qquad \gamma = A_c/A$
- Use signed areas for points outside the triangle

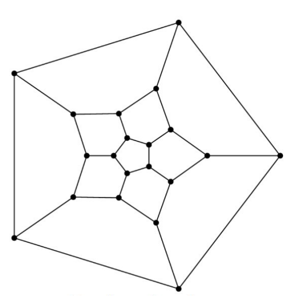


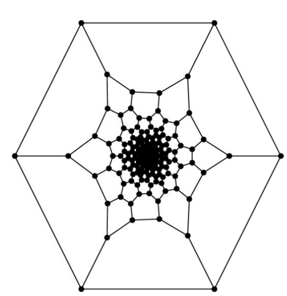
But how do I know if the point is outside the triangle?
That's what I was trying to determine!

Barycentric Graph Layout



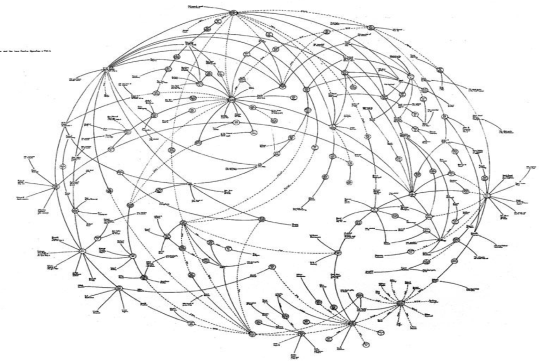
- Planar graphs (only?)
- Start with some fixed positions
- Each vertex is placed at the average of its neighbors
- Can be solved as a system of equations
- Results can have poor resolution & poor distribution of vertices

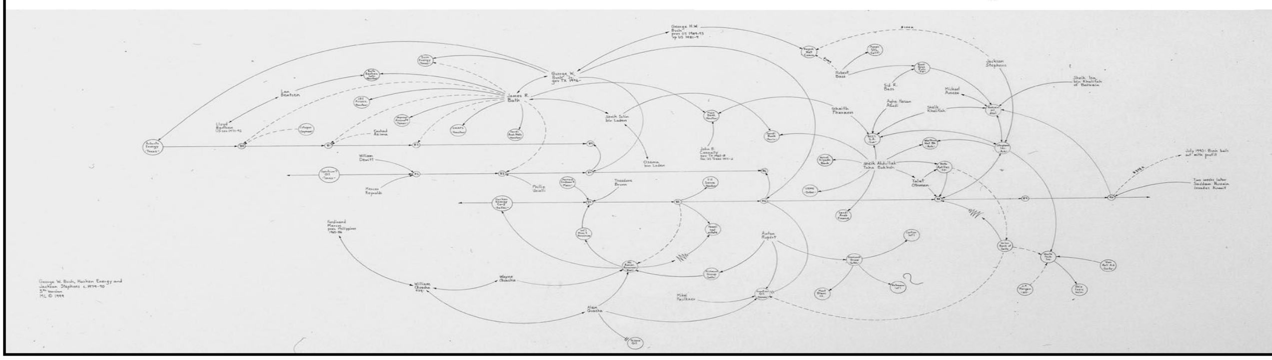




Using Non-Straight Edges

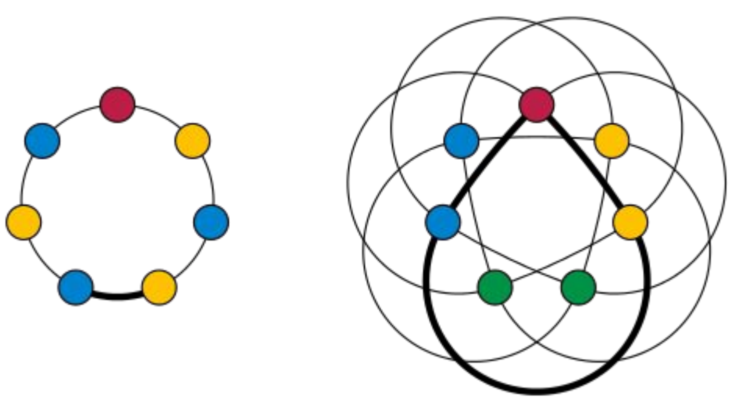
- Increase/maximize angular resolution at vertices
  - Can use Bezier Curves
- Lombardi or “near-Lombardi”





Effectiveness of Curved Edges

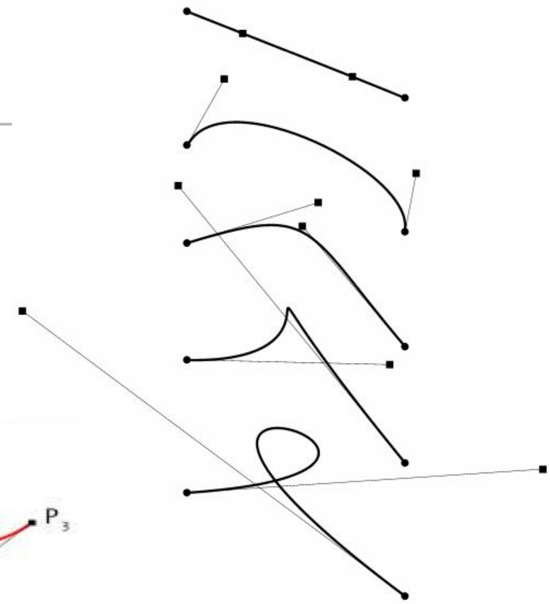
- What are the advantages of straight vs curved edges for the graph on the right?



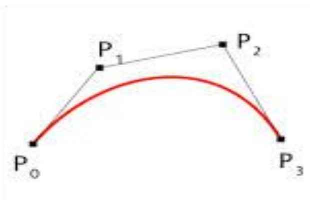
A seven-vertex cycle and its complement, showing in each case an optimal coloring and a maximum clique (shown with heavy edges). Since neither graph uses a number of colors equal to its clique size, neither is a “Perfect Graph”.

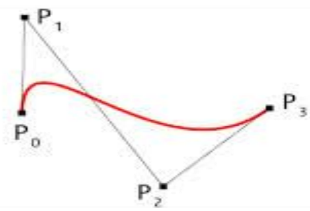
https://en.wikipedia.org/wiki/Perfect_graph#/media/File:7-hole_and_antihole.svg

Cubic Bézier Curve



- 4 control points
- Curve passes through first & last control point
- Curve is tangent at $\mathbf{P_0}$ to ($\mathbf{P_1}$- $\mathbf{P_0}$) and at $\mathbf{P_3}$ to ($\mathbf{P_3}$- $\mathbf{P_2}$)





http://www.e-cartouche.ch/content_reg/cartouche/graphics/en/html/Curves_learningObject2.html
http://www.webreference.com/dlab/9902/bezier.html

Non Euclidean

- Hyperbolic layout has better equal distance layout for leaves of a "complete tree"
- Related to
  - Fisheye view
  - Focus + context
  - Assumption that center is more important

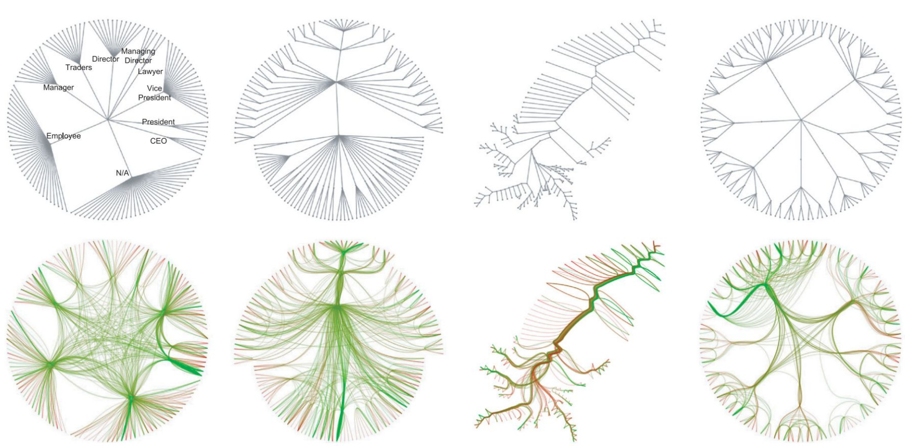
Daina Taimina
http://www.math.cornell.edu/~dtaimina/Artexhibits.htm

Today

- Homework 2 Discussion/Questions
- Readings for Today
  - “Improved force-directed Layouts”
  - “A Technique for Drawing Directed Graphs”
- Graph Drawing Goals, Questions, & Challenges
- Some Related Terms/Algorithms
  (mentioned indirectly in the reading)
- Readings for Friday
- Computational Geometry: Closest pair of points

Readings for Friday *(pick one)*

“Social Network Clustering and Visualization using Hierarchical Edge Bundles”, Jia, Garland, & Hart, Computer Graphics Forum, 2011.



Manual (ideal) | Strength clustering [ACJM03] | BC communities [GN02] | Our method

Readings for Friday *(pick one)*

"Force-directed Lombardi-style graph drawing", Chernobelskiy et al., Graph Drawing 2011.

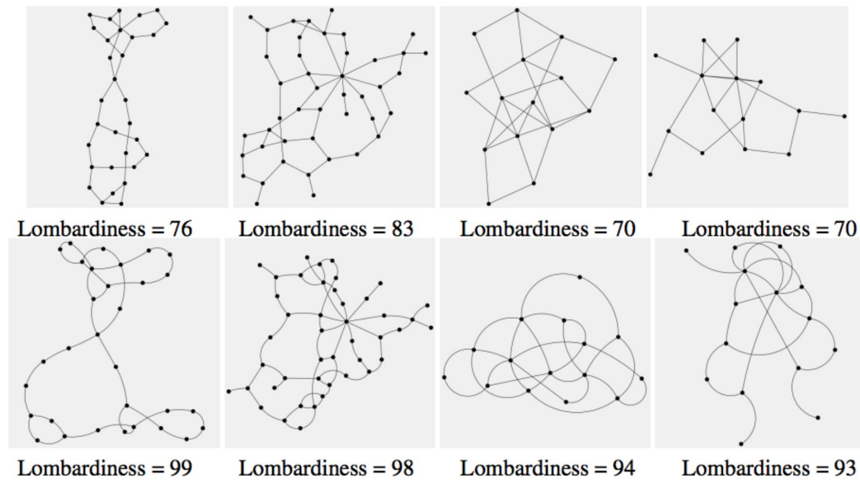


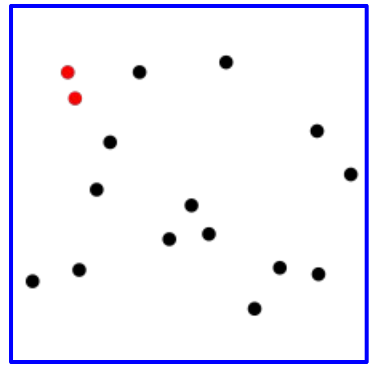
Fig. 5. Standard force-directed drawings (above) and near-Lombardi drawings (below).

Today

- Homework 2 Discussion/Questions
- Readings for Today
  - "Improved force-directed Layouts"
  - "A Technique for Drawing Directed Graphs"
- Graph Drawing Goals, Questions, & Challenges
- Some Related Terms/Algorithms (mentioned indirectly in the reading)
- Readings for Friday
- Computational Geometry: Closest pair of points

Closest Pair of Points Problem

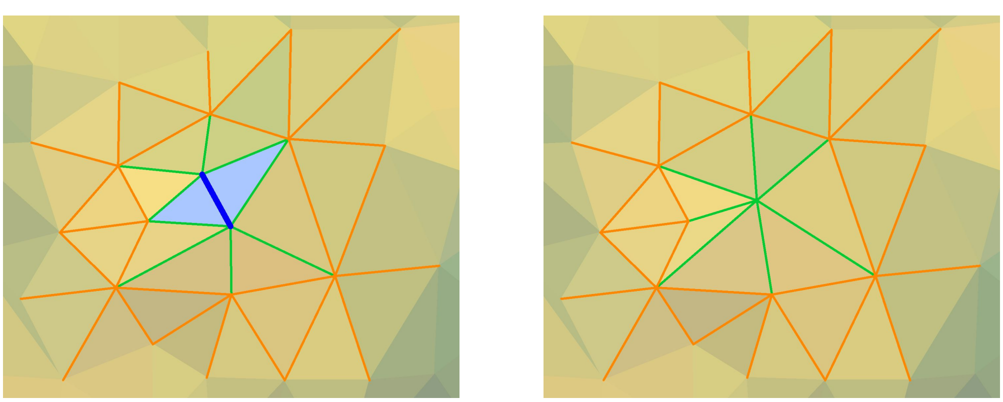
- Given *n* points,
  find the two points that
  have the smallest distance
  between each other.
- Applications?
  - Preventing graph node overlap
  - Collision detection simulation
    (air traffic control, games, etc)
  - Merging similar data points
    (data size reduction)



https://en.wikipedia.org/wiki/Closest_pair_of_points_problem

Edge Contraction / Edge Collapse

- Goal: Reduce number of vertices/edges while minimize shape/color/attribute loss
- Possible algorithm for 2D/3D meshes: Always collapse shortest edge



Brute Force Algorithm

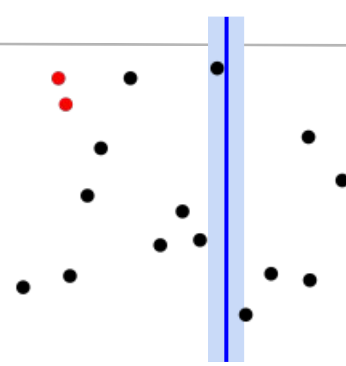
- Analysis?
  For *n* points?

  $O(n^2)$

```
minDist = infinity
for i = 1 to length(P) - 1
 for j = i + 1 to length(P)
  let p = P[i], q = P[j]
  if dist(p, q) < minDist:
   minDist = dist(p, q)
   closestPair = (p, q)
return closestPair
```

https://en.wikipedia.org/wiki/Closest_pair_of_points_problem

Divide & Conquer Algorithm



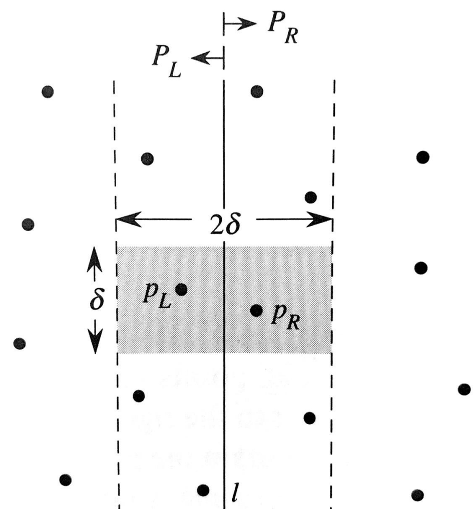
- Sort points by one of the axes
  - Find middle point,
  - Split points into two equal sized groups
  - & Recurse…
- Combine results: Overall closest pair must be:
  - Closest pair in left half (distance = δ_l), or
  - Closest pair in right half (distance = δ_r), or
  - A pair that spans the halves w/ distance < min(δ_l , δ_r)

Divide & Conquer Algorithm

- How many pairs do we need to consider at the boundary?

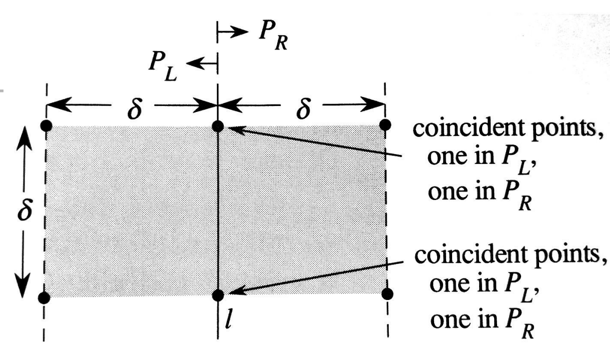
- In the worst case,
  all points are within
  δ of the split point!
  Where $\delta = \min(\delta_l, \delta_r)$

- Isn't this $O(n^2)$??



Introduction to Algorithms, Cormen, Leiserson, & Rivest

Divide & Conquer



- Let's also sort the points by the y-axis

- Walk from top to bottom and compare each point to all points within δ vertical distance (grey box).

- Worst case, how many other points are in this rectangle?

- No more than 7 other points!

Introduction to Algorithms, Cormen, Leiserson, & Rivest

Divide & Conquer

- Analysis:
    - Store the points twice, sorted by x & y axes
        - Sort once at the start, not in each recursion
    - Per recursion
        - Max of *O(7n)* pairwise comparisons
    - Overall: *O (n log n)*
- Does it work in 3D? Or higher dimensions?
- Can we do better?
- What about dynamic data? What applications?