Clustering and greedy algorithms — Part 2

Prof. Noah Snavely
CS1114
http://cs1114.cs.cornell.edu

Cornell University
Computer Science

Administrivia

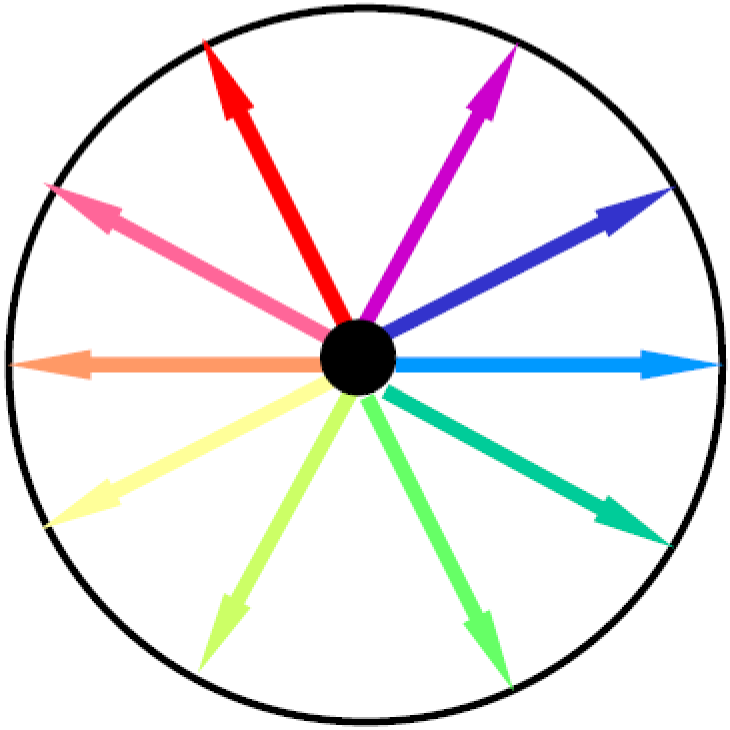
- Prelim 3 on Thursday
  - Will be comprehensive, but focused on Markov chains and clustering
  - Review session Wednesday at 7pm, Upson 315
  - Steve will proctor the exam

Administrivia

- Final projects
  - Due May 8th – we'll have sign ups for demo sessions (tentative times: 2-6pm)
  - There will be prizes for the best demos!

- Next year:
  - We hope to offer CS1114 again in SP10
  - Possibly will start a fall version as well
  - We want you for the course staff!

Life after CS1114?



Computer Science major

- AI
- Network Science
- Theory
- CSE
- Graphics
- Systems
- Security
- Data-Intensive
- Programming Languages
- Human Language Tech.

Reversing a Markov chain?

	P	Q	L
P	0.1	0.2	0.7
Q	0.3	0.1	0.6
L	0.2	0.3	0.5

- What is the probability that a lecture will be followed by a prelim?
- What is the probability that a prelim was *preceded* by a lecture?
  - Why isn't the answer 0.2?
  - Why isn't the answer 1/3?
  - The real answer is ~0.556 or 55.6%

Reversing a Markov chain?

- PQLLQLLPLQQLQLLQLQQPLLQLLLLLLQPQL PLLLQLPQLLLLPPLQLLLQPLLQLPLLQPPLLQ QPLPLPLPLPLLLPLLLPQPLPQQLQLQLLQQP

- The reason why a lecture is more likely to precede a prelim is that lectures are overall more frequent than prelims or quizzes
- You'll learn more in a course on probability / statistics (c.f. Bayes' Rule)

Clustering

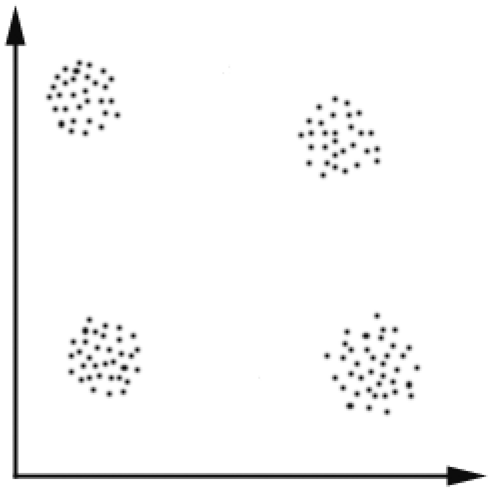
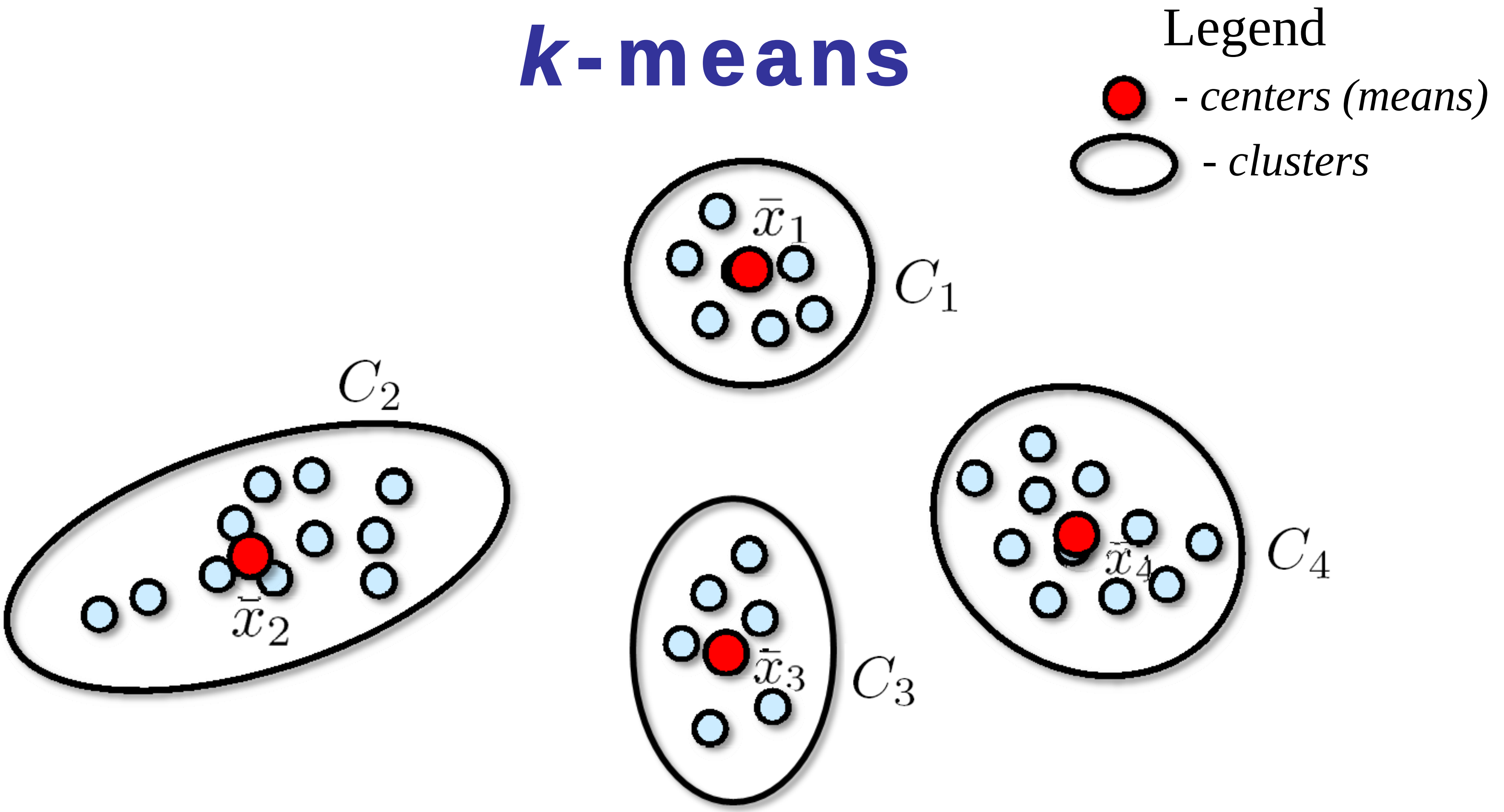


Figure from Johan Everts

One approach: *k*-means

- Suppose we are given *n* points, and want to find *k* clusters
- We will find *k* cluster centers (or means), and assign each of the *n* points to the nearest cluster center $\bar{x}_j$
  - A *cluster* is a subset of the *n* points, called C_j
  - We'll call each cluster center a *mean*

k-means



- How do we define the best *k* means and clusters?

Optimizing *k*-means

Given input points $x_1, x_2, x_3, \ldots, x_n$, find the clusters $C_1, C_2, \ldots C_k$ and the cluster centers $\bar{x}_1, \bar{x}_2, \bar{x}_3, \ldots, \bar{x}_k$ that minimize

$$\sum_{j=1}^{k} \sum_{x_i \in C_j} |x_i - \bar{x}_j|^2$$

- The bad news: this is not a convex optimization
- The worse news: it is practically impossible to find the global minimum of this objective function
  - no one has ever come up with an algorithm that is faster than exponential time (and probably never will)
- There are many problems like this (called *NP-hard*)

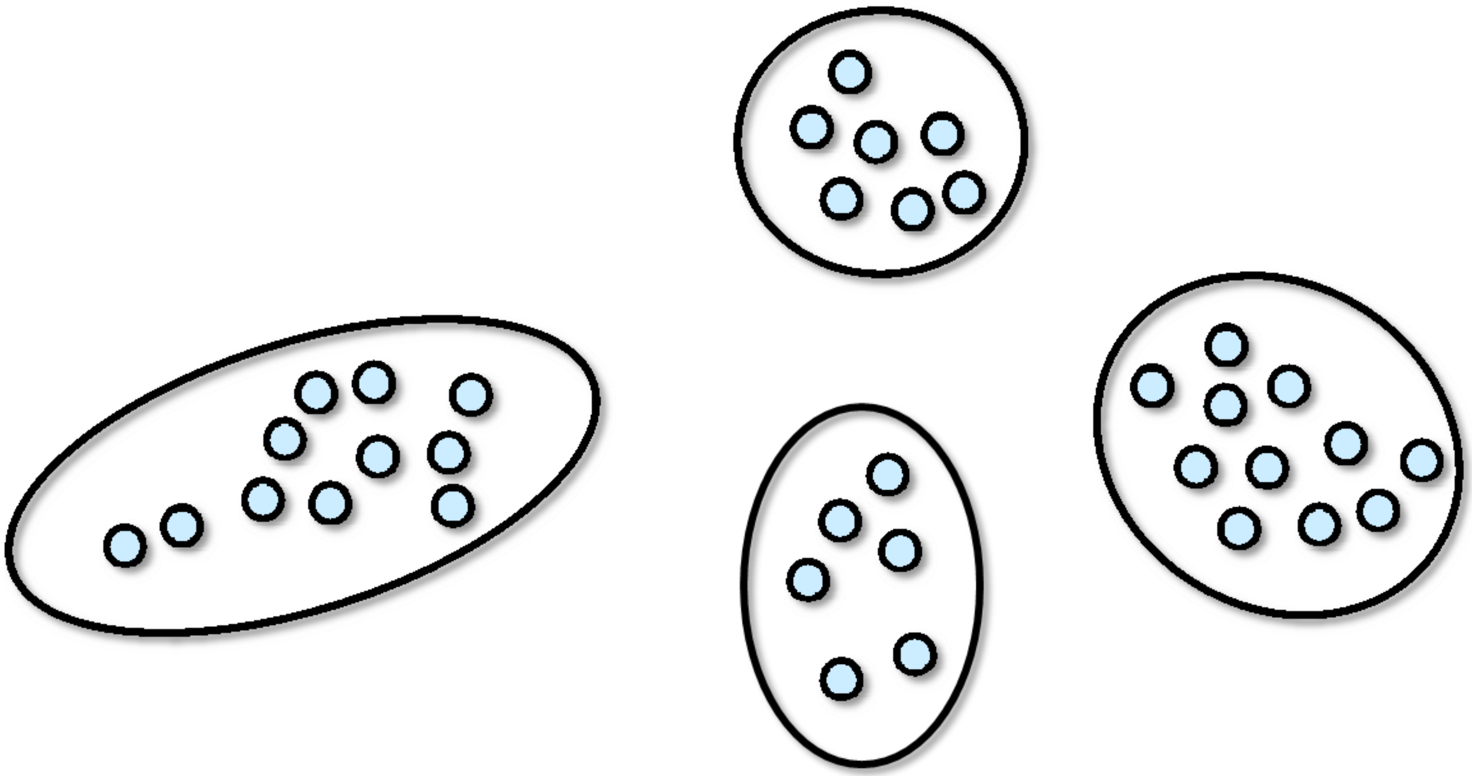
Greedy algorithms

- Many CS problems can be solved by repeatedly doing whatever seems best at the moment
  - I.e., without needing a long-term plan
- These are called greedy algorithms
- Example: gradient descent for convex function minimization
- Example: sorting by swapping out-of-order pairs (e.g., bubble sort)
- Example: making change (with US currency)

A greedy method for *k*-means

- Pick a random point to start with, this is your first cluster mean
- Find the farthest point from the cluster center, this is a new cluster mean
- Find the farthest point from any cluster mean and add it
- Repeat until we have *k* means
- Assign each point to its closest mean

A greedy method for *k*-means



A greedy method for *k*-means

- Unfortunately, this doesn’t work that well in general
- The answer we get could be **much** worse than the optimum

The *k*-centers problem

- Let's look at a related problem: *k*-centers
- Find *k* cluster centers that minimize the *maximum* distance between any point and its nearest center
  - We want the worst point in the worst cluster to still be good (i.e., close to its center)
  - Concrete example: place *k* hospitals in a city so as to minimize the maximum distance from a hospital to a house

k-centers

- What objective function does this correspond to?

Given input points $x_1, x_2, x_3, \ldots, x_n$, find the clusters $C_1, C_2, \ldots C_k$ and the cluster centers $\bar{x}_1, \bar{x}_2, \bar{x}_3, \ldots, \bar{x}_k$ that minimize

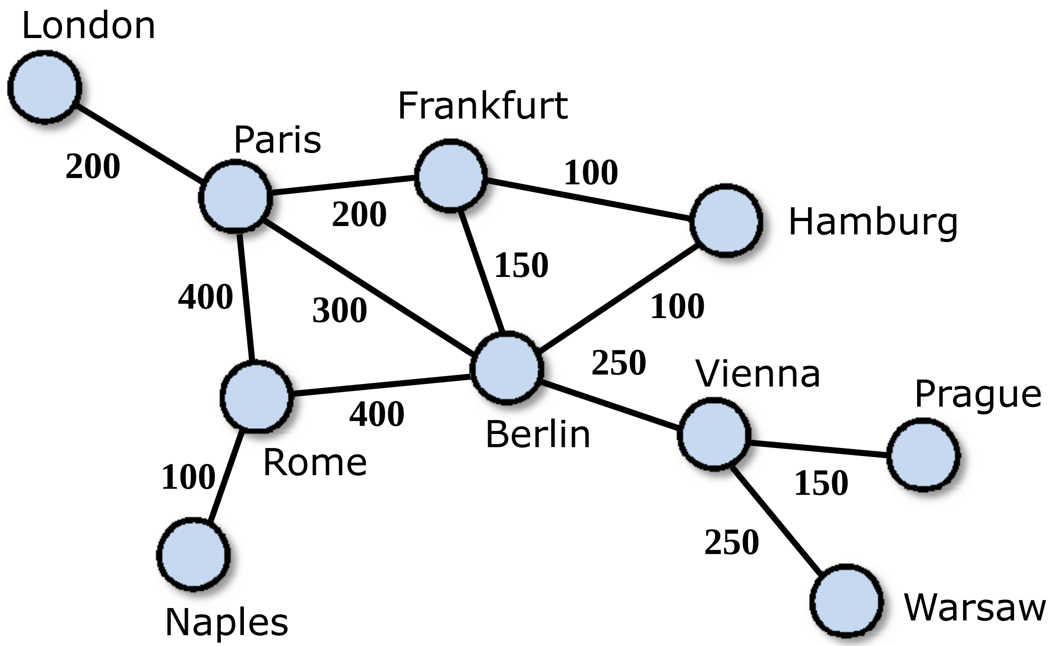
$$\max_{j=1}^{k} \max_{x_i \in C_j} |x_i - \bar{x}_j|^2$$

- We can use the same greedy algorithm

An amazing property

- This algorithm gives you a solution that is no worse than twice the optimum
- (*k*-centers is still NP-hard, just like *k*-means)
- Such results are sometimes difficult to achieve, and the subject of much research
  - Mostly in CS6810, a bit in CS4820
  - You can't find the optimum, yet you can prove something about it!
- Sometimes related problems (e.g. *k*-means vs. *k*-centers) have very different guarantees

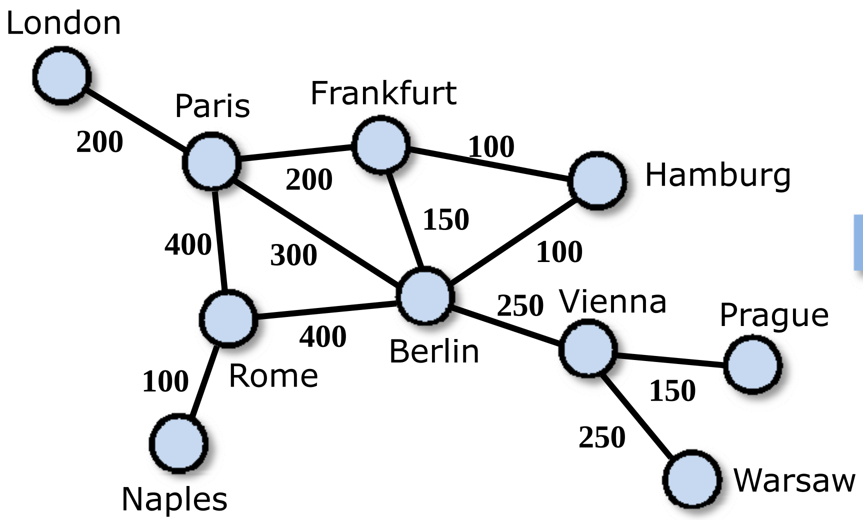
Detour into graphs

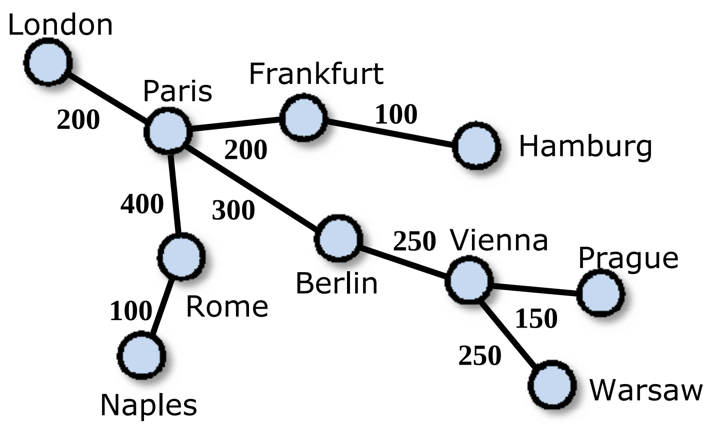


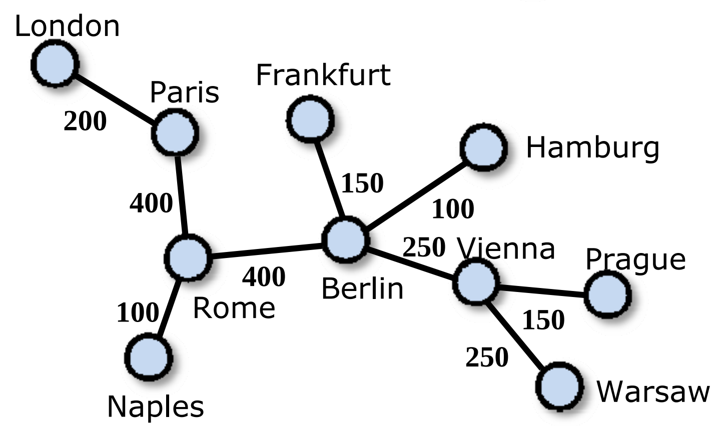
- We can also associate a *weight* with each edge (e.g., the distance between cities)

Spanning trees

- A spanning tree of a graph is a subgraph that (a) connects all the vertices and (b) is a tree



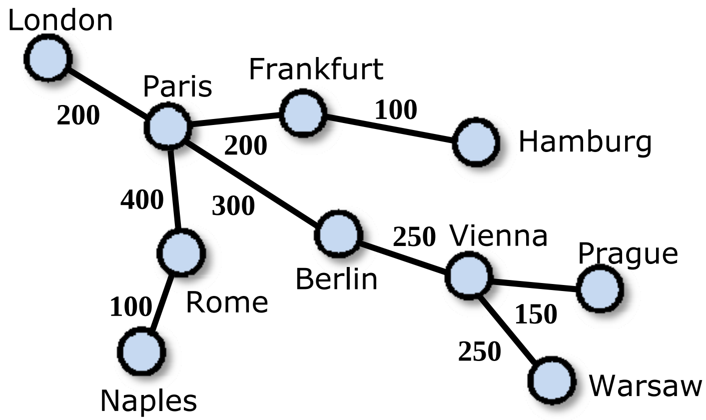


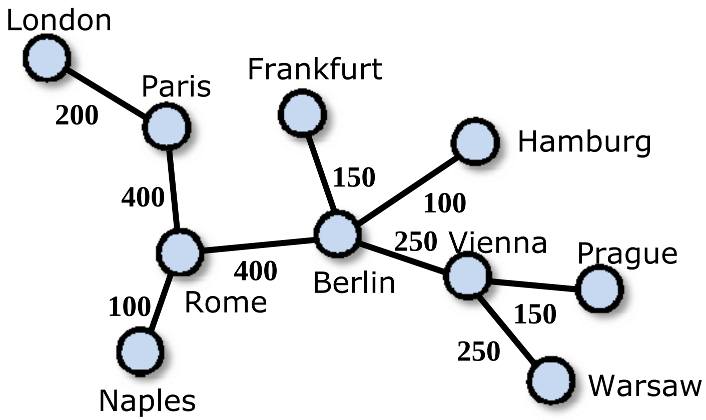


Q: How many edges are there in a spanning tree on n vertices?

Graph costs

- We'll say the *cost* of a graph is the sum of its edge weights



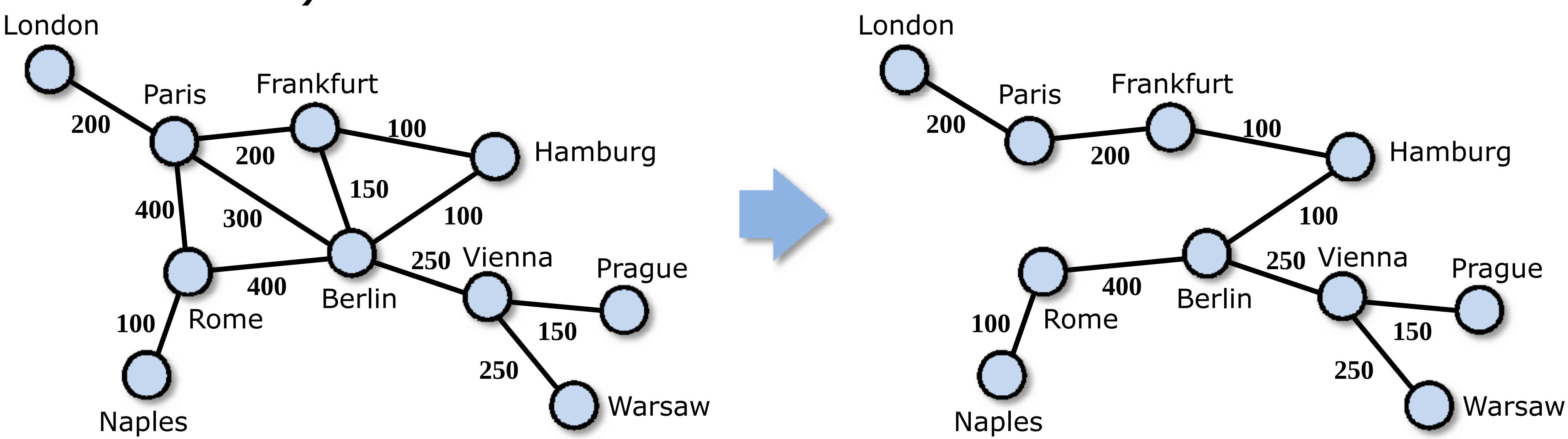


$$Cost = 200 + 200 + 100 + 400 + 300 + 100 + 250 + 150 + 250 = \mathbf{1950}$$

$$Cost = 200 + 400 + 100 + 400 + 150 + 250 + 100 + 150 + 250 = \mathbf{2000}$$

Minimum spanning trees

- We define the *minimum spanning tree* (MST) of a graph as the spanning tree with minimum cost
- (Suppose we want to build the minimum length of track possible while still connecting all the cities.)

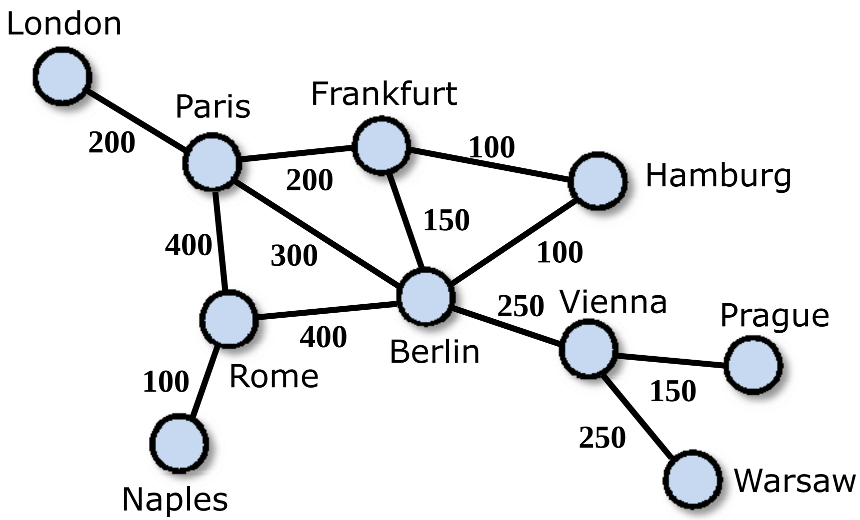


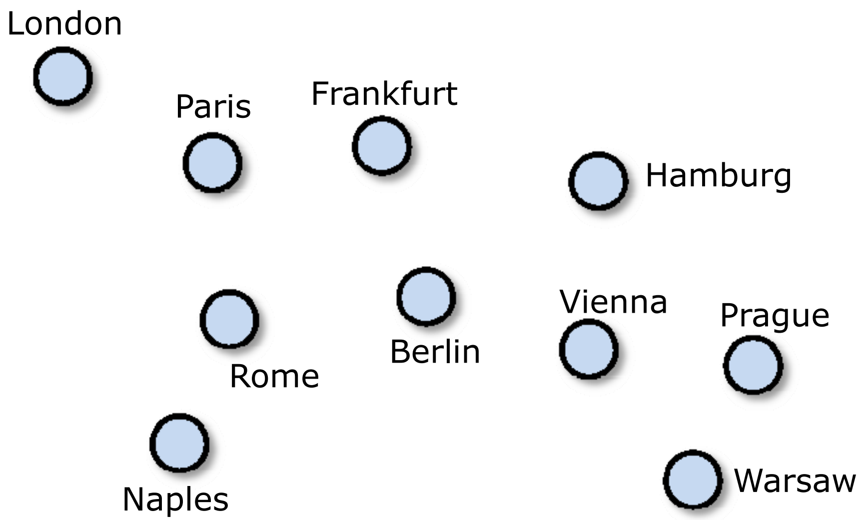
MST: Cost = 1750

(Eurorail needs to build 1750 mi of track at minimum)

Minimum spanning trees

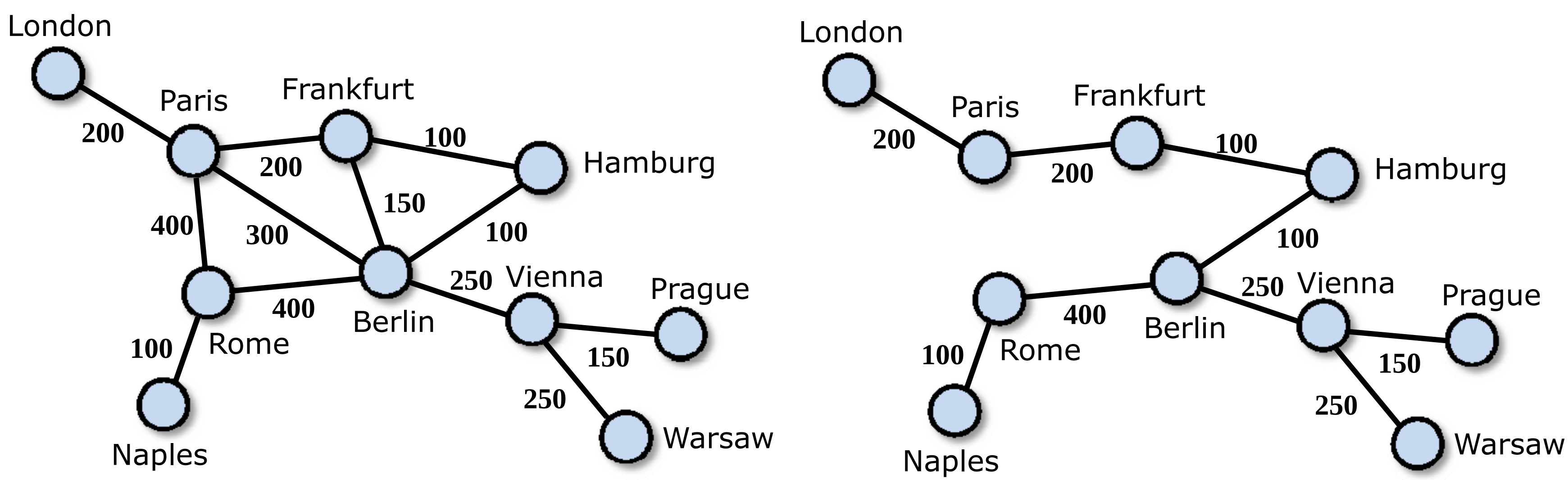
- How do we find the minimum spanning tree?
- Can you think of a greedy algorithm to do this?





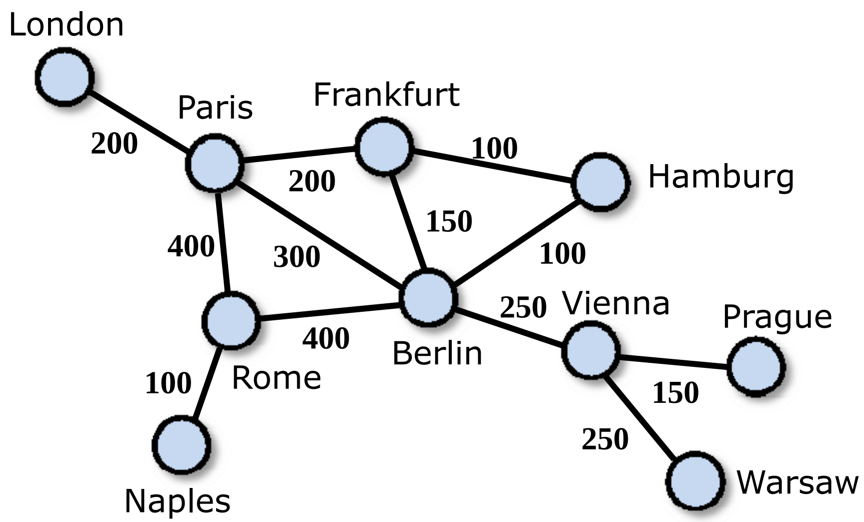
Minimum spanning tree

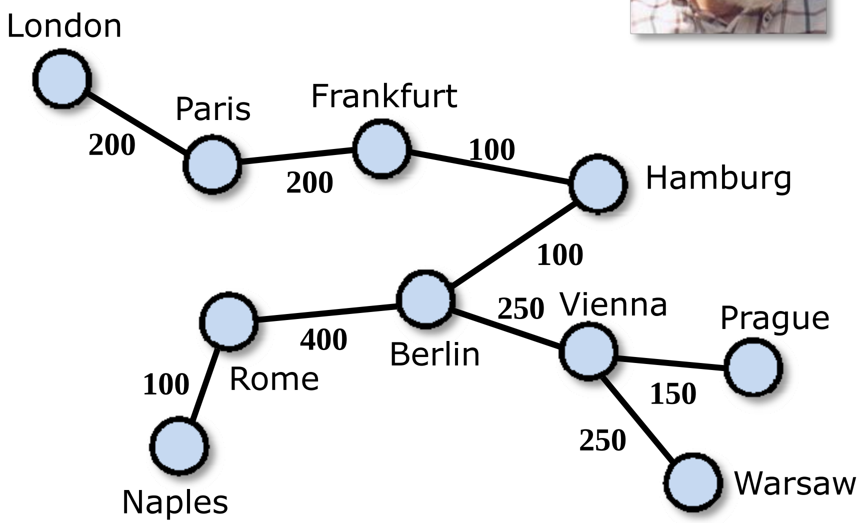
- Greedy algorithm:



Minimum spanning tree

- This greedy algorithm is called **Kruskal's algorithm**

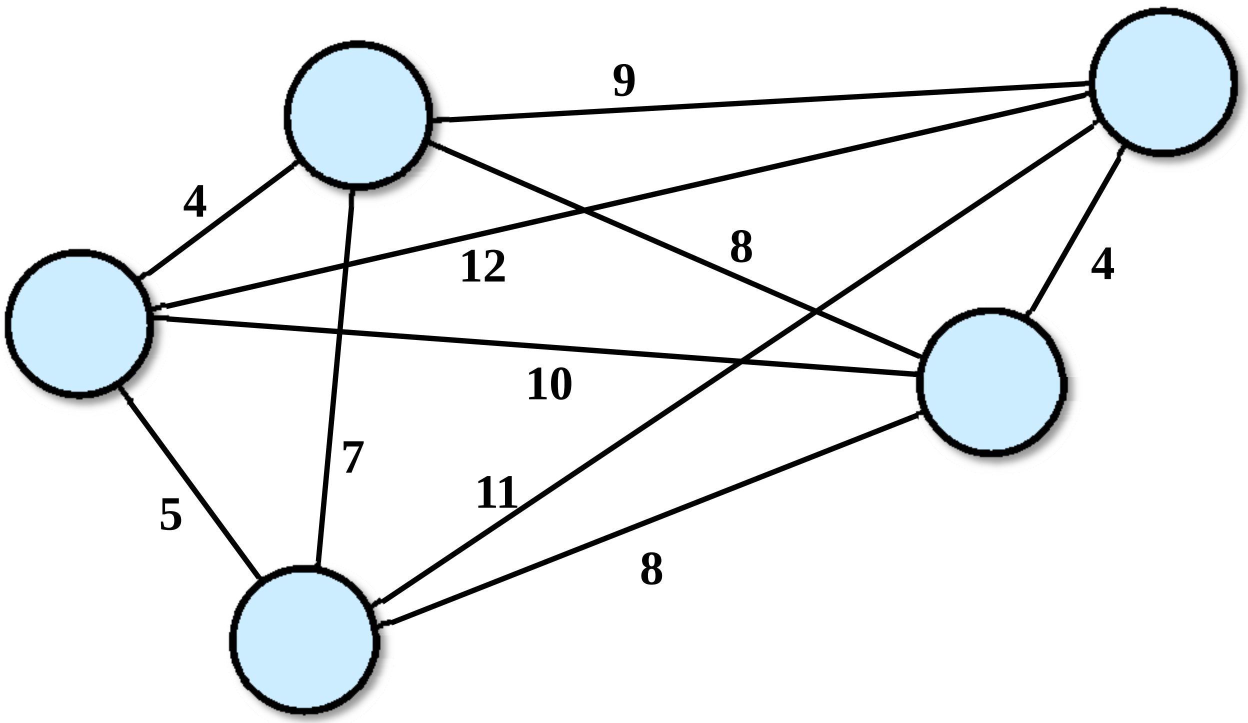




- Not that simple to prove that it gives the MST
- How many connected components are there after adding the k^{th} edge?

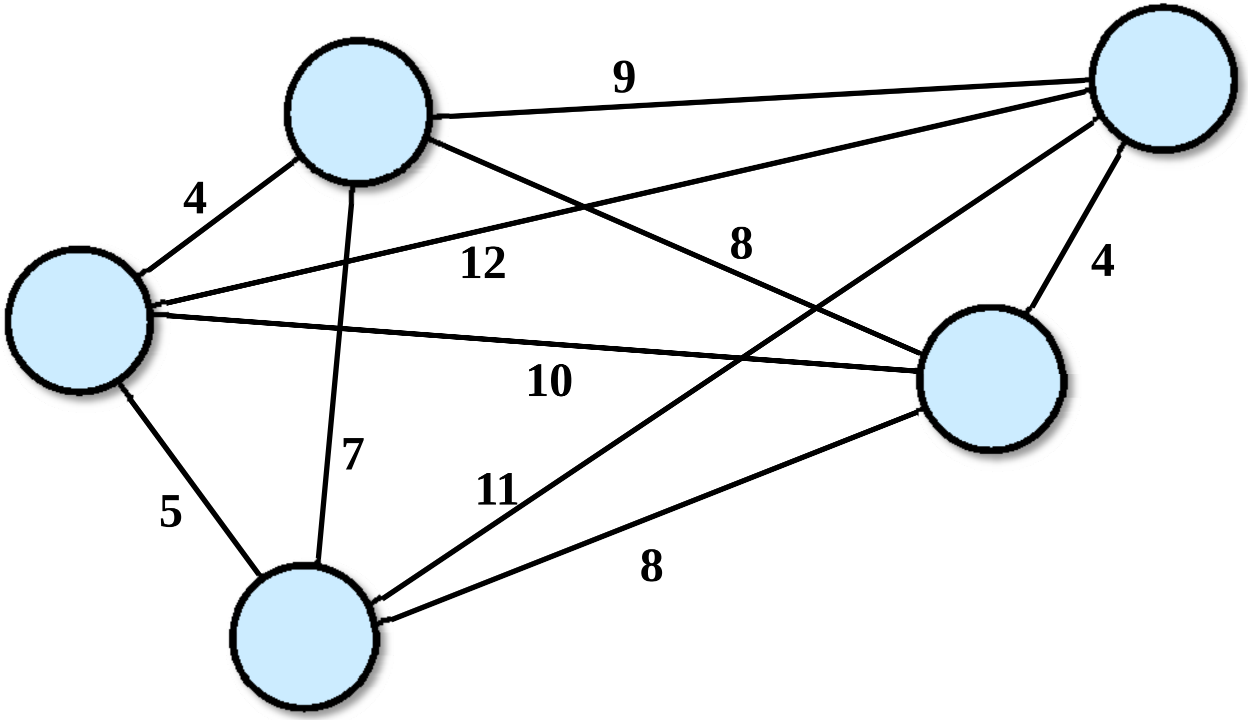
Back to clustering

- We can define the clustering problem on graphs



Clustering using graphs

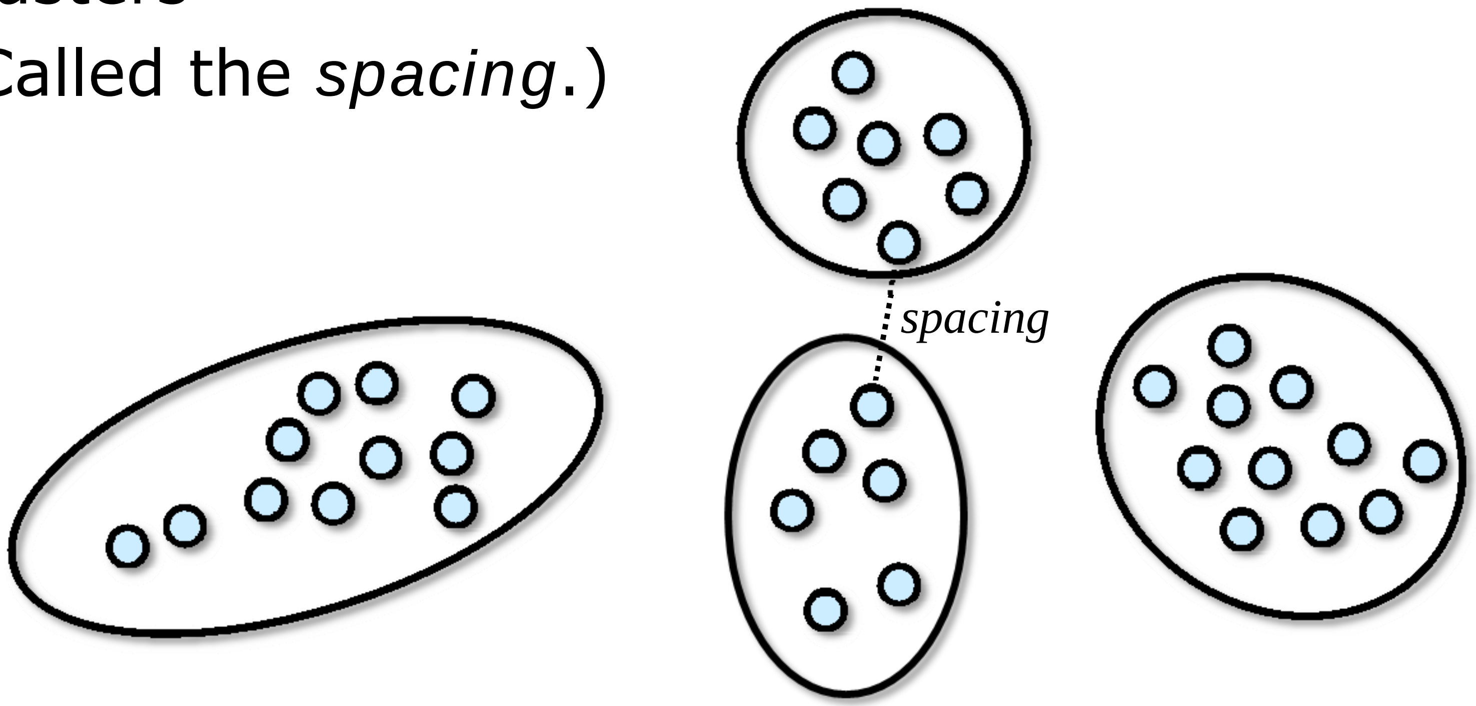
- Clustering → breaking apart the graph by cutting long edges



- Which edges do we break?

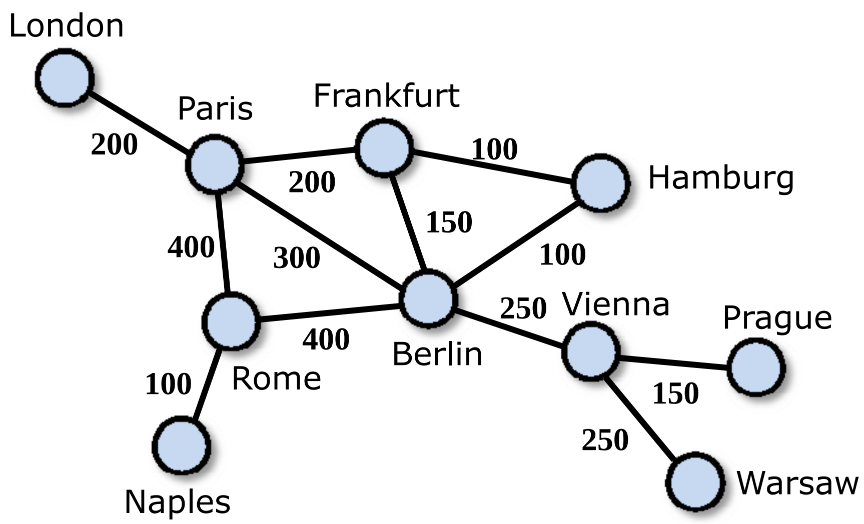
Spacing as a clustering metric

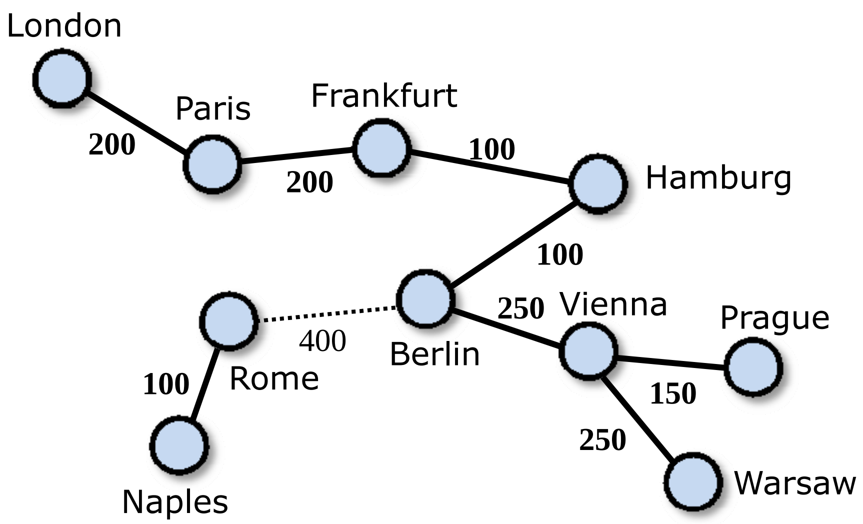
- Another objective function for clustering:
  - Maximize the minimum distance between clusters
  - (Called the *spacing*.)



Cool fact

- We compute the clusters with the maximum spacing during MST!
- To compute the best *k* clusters, just stop MST construction *k*-1 edges early





2 clusters with max spacing (=400)

Proof of cool fact

- Suppose this wasn't true – then someone could give us a different clustering with a bigger spacing
- Let C^* be our MST clustering, and let C be the purportedly better one
- There must be two nodes u and v in different clusters in C but in the same cluster in C^*
- There's a path between u and v in C^*, and at some point this path crosses a cluster boundary in C

Proof of cool fact

- Let this boundary-crossing edge be called e

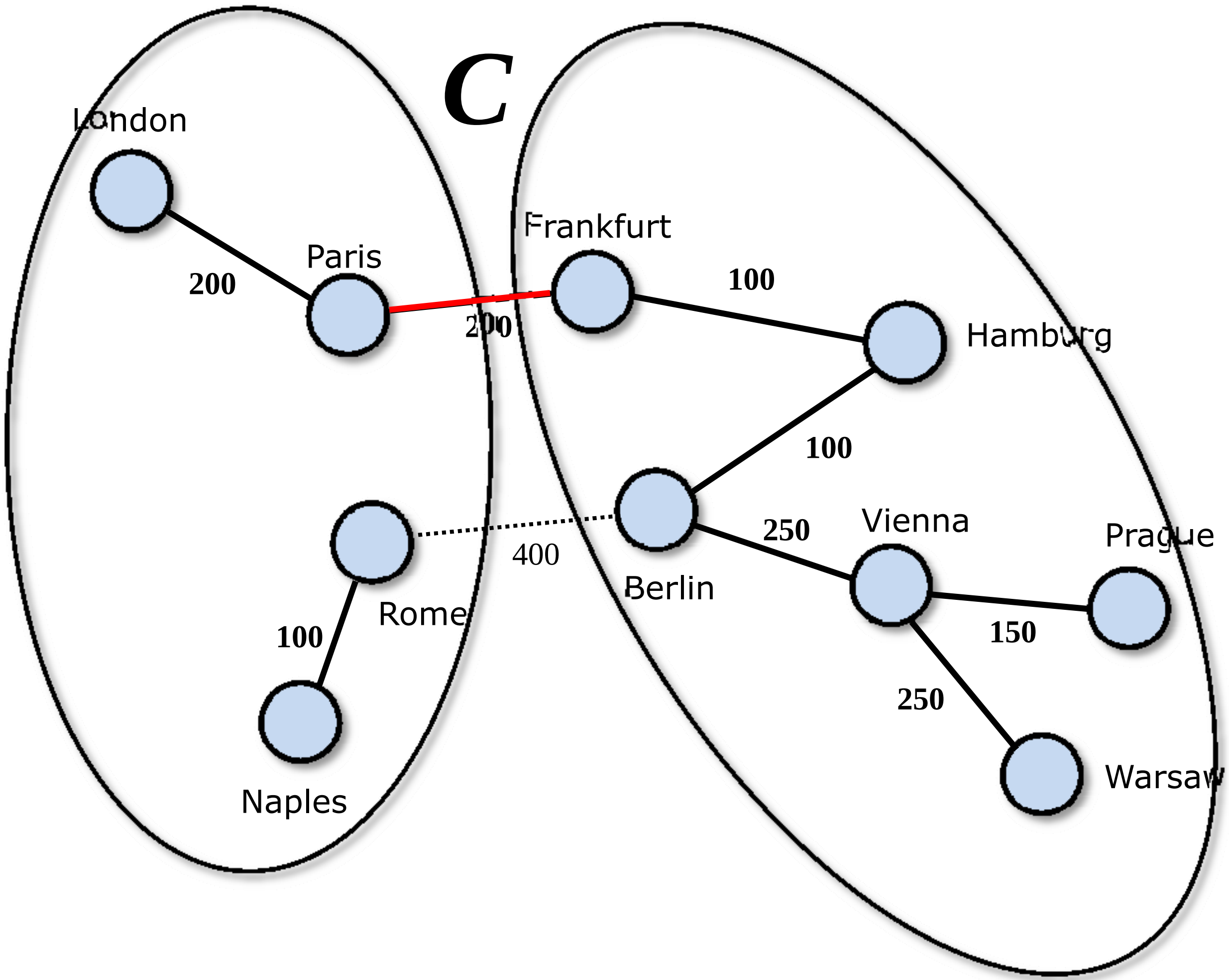
- We know that weight(e) ≤ the next edge we would add to the MST (why?)

- → weight(e) ≤ spacing of C^*

- → spacing of C ≤ spacing of C^*

- So C wasn't really better after all…

Pictorial proof



Conclusions

- Greedy algorithms work sometimes (e.g., with MST)

- Some clustering objective functions are easier to optimize than others:
  - *k*-means → very hard
  - *k*-centers → very hard, but we can use a greedy algorithm to get within a factor of two of the best answer
  - maximum spacing → very easy! Just do MST and stop early

Next time

- Make sure to fill in course evals online (2 pts)
  http://www.engineering.cornell.edu/CourseEval/
- Review session tomorrow, 7pm, Upson 315
- Prelim on Thursday