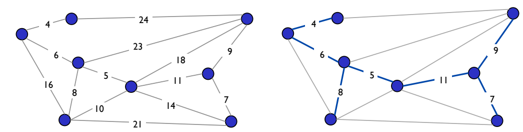
Objectives

- Clustering

Review: Minimum Spanning Tree

- Spanning tree: spans all nodes in graph
- Given a connected graph G = (V, E) with positive edge weights c_e, an MST is a subset of the edges $T \subseteq E$ such that T is a *spanning tree* whose sum of edge weights is *minimized*

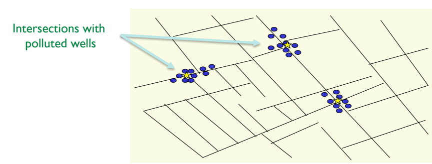


G = (V, E) T, $\Sigma_{e \in T}\, c_e = 50$

Review: Greedy Algorithms

All three algorithms produce a MST

- Prim's algorithm. Start with some root node *s* and greedily grow a tree *T* from *s* outward. At each step, add the cheapest edge *e* to *T* that has exactly one endpoint in *T*.
  - Similar to Dijkstra's (but simpler)
- Kruskal's algorithm. Start with *T* = ϕ. Consider edges in ascending order of cost. Insert edge *e* in *T* unless doing so would create a cycle.
- Reverse-Delete algorithm. Start with T = E. Consider edges in descending order of cost. Delete edge *e* from *T* unless doing so would disconnect *T*.



Outbreak of cholera deaths in London in 1850s.
Reference: Nina Mishra, HP Labs

CLUSTERING

Clustering

- Given a set *U* of *n* objects (or points) labeled $p_1, \ldots, p_n$, classify into coherent groups
  - Example objects: photos, documents, micro-organisms
- Distance function. Numeric value specifying "closeness" of two objects

Clustering Problem

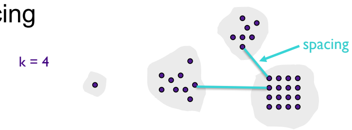
- Divide objects into clusters so that points in different clusters are far apart
- Applications
  - Routing in mobile ad hoc networks
  - Identify patterns in gene expression
  - Identifying patterns in web application use cases
    - Sets of URLs
  - Similarity searching in medical image databases
  - Skycat: cluster 109 sky objects into stars, quasars, galaxies

Clustering

- k-clustering. Divide objects into *k* non-empty groups
- Distance function. Assume it satisfies several natural properties
  - $d(p_i, p_j) = 0$ iff $p_i = p_j$ (identity of indiscernibles)
  - $d(p_i, p_j) \geq 0$ (nonnegativity)
  - $d(p_i, p_j) = d(p_j, p_i)$ (symmetry)

Clustering of Maximum Spacing

- k-clustering. Divide objects into *k* non-empty groups
- Spacing. Min distance between any pair of points in different clusters
- Clustering of maximum spacing. Given an integer *k*, find a *k*-clustering of maximum spacing



Ideas about Solving?

- Greedy algorithm?
- How relates to the minimum spanning tree?

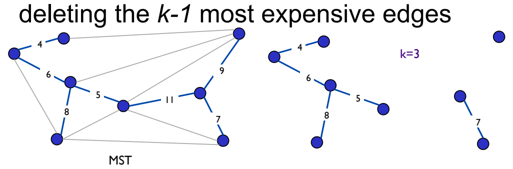
Greedy Clustering Algorithm

- Single-link *k*-clustering algorithm
  - Form a graph on the vertex set *U*, corresponding to *n* clusters
  - Find the closest pair of objects such that *each object is in a different cluster*, and add an edge between them
  - Repeat *n-k* times until there are exactly *k* clusters

How is this related to the MST?

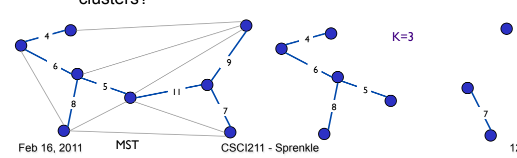
Greedy Clustering Algorithm

- Key observation. Same as Kruskal's algorithm
  - Except we stop when there are *k* connected components
- Remark. Equivalent to finding an MST and deleting the *k-1* most expensive edges



Greedy Clustering Algorithm: Analysis

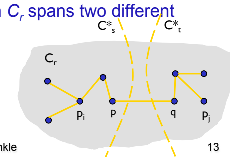
- Theorem. Let C denote the clustering $C_1, \ldots, C_k$ formed by deleting the *k-1* most expensive edges of a MST. C is a *k*-clustering of *max spacing*.
- Pf Intuition:
  - What can we say about C's spacing?
    - Within clusters and between clusters
  - What if C isn't optimal?
    - What does that mean about C's clusters vs (optimal) C*'s clusters?



Greedy Clustering Algorithm: Analysis

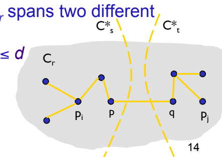
- Theorem. Let C denote the clustering $C_1, \ldots, C_k$ formed by deleting the *k-1* most expensive edges of a MST. C is a *k*-clustering of ***maximum spacing***.
- Pf Sketch. Let C* denote some other clustering $C^*_1, \ldots, C^*_k$. C* and C must be different; otherwise we're done.
  - The spacing of *C* is length *d* of (k-1)[st] most expensive edge
  - Let p_i, p_j be in the same cluster in *C* (say C_r) but different clusters in C*, say C^*_s and C^*_t
  - Some edge (*p*, *q*) on p_i-p_j path in C_r spans two different clusters in *C**

What do we know about (*p*, *q*)?



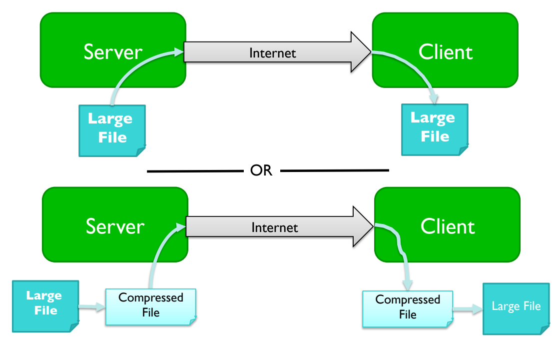
Greedy Clustering Algorithm: Analysis

- Theorem. Let C denote the clustering $C_1, \ldots, C_k$ formed by deleting the *k-1* most expensive edges of a MST. C is a *k*-clustering of ***maximum spacing***.
- Pf. Let C* denote some other clustering $C^*_1, \ldots, C^*_k$. C* and C must be different; otherwise we're done.
  - The spacing of *C* is length *d* of (k-1)[st] most expensive edge
  - Let p_i, p_j be in the same cluster in *C* (say C_r) but different clusters in C*, say C^*_s and C^*_t
  - Some edge (*p*, *q*) on p_i-p_j path in C_r spans two different clusters in *C**
  - All edges on p_i-p_j path have length $\leq d$ since Kruskal chose them
  - Spacing of C* is at most $\leq$ d since *p* and *q* are in different clusters



MOTIVATING FRIDAY'S PROBLEM

Which Is Better?

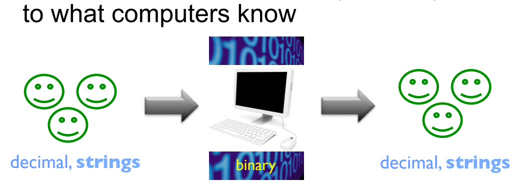


Discussion: Which Is Better?

- Depends on your metrics, compression time/amount
- Case 1 requires
  - More network resources
  - Less CPU time (server: compress; client: uncompress)
- Case 2 requires
  - Less network resources
  - More CPU time (client and server)
- Overall best
  - Depends on file size, network speed, compression time/ amount
  - Bigger files → Case 2

Problem: Encoding

- Computers use bits: 0s and 1s
- Need to represent what we (humans) know to what computers know



- Map **symbol** → unique sequence of 0s and 1s
- Process is called *encoding*

Problem: Encoding

- Let's say we want to encode characters using 0s and 1s
  - Lower case letters (26)
  - Space
  - Punctuation (, . ? ! ')

What is the **least** number of bits we would we need to encode these characters?

Feb 16, 2011 CSCI211 - Sprenkle 19

Problem: Encoding Symbols

- 32 characters to encode
  - $\log_2(32)$ = 5 bits
  - Can't use fewer bits
- Examples:
  - a → 00000
  - b → 00001
- Actual mapping from character to encoding doesn't matter
  - Easier if have a way to compare …

Feb 16, 2011 CSCI211 - Sprenkle 20

For Long Strings of Characters…

- Do we need an average of 5 bits/character always?
- What if we could use *shorter encodings* for *frequently* used characters, like a, e, s, t?

Goal: Optimal encoding that takes advantage of *nonuniformity* of letter frequencies

- A fundamental problem for **data compression**
  - Represent data *as compactly as possible*

Feb 16, 2011 CSCI211 - Sprenkle 21

Example: Morse Code

- Used for encoding messages over telegraph
- Example of *variable-length encoding*

How are letters encoded?
How are letters differentiated?

Feb 16, 2011 CSCI211 - Sprenkle 22

Example: Morse Code

- Used for encoding messages over telegraph
- Example of *variable-length encoding*
- How are letters encoded?
  - Dots, dashes
  - Most frequent letters use shorter sequences
    - e → dot; t → dash; a → dot-dash
- How are letters differentiated?
  - Spaces in between letters
    - Otherwise, ambiguous

Feb 16, 2011 CSCI211 - Sprenkle 23

Ambiguity in Morse Code

- Encoding:
  - e → dot; t → dash; a → dot-dash
- Example: dot-dash-dot-dash could correspond to

Feb 16, 2011 CSCI211 - Sprenkle 24

Ambiguity in Morse Code

- Encoding:
  - e → dot; t → dash; a → dot-dash
- Example: dot-dash-dot-dash could correspond to
  - etet
  - aa
  - eta
  - aet

What's the problem?

Problem

- Ambiguity caused by encoding of one character is a *prefix* of encoding of another

Prefix Codes

- Problem: Encoding of one character is a *prefix* of encoding of another
- Solution: **Prefix Codes**: map letters to bit strings such that *no encoding is a prefix of any other*
  - Won't need artificial devices like spaces to separate characters
- Example encodings:
  - Verify that no encoding is a prefix of another
  - What is `0010000011101`?

```
a: 11     d: 10
b: 01     e: 000
c: 001
```


Problem Set 3

- Binary tree proof
- Finding a cycle
- Communication network distance
- Analyze algorithm's efficiency
- Test cases for your algorithms

Assignments

- PS 4 due Friday
- Continue reading chapter 4
  - 4.5-4.8