

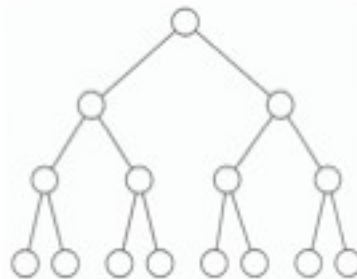
## CSE202 Greedy algorithms II

*An induced subgraph of the collaboration graph (with Erdos number at most 2).  
Made by Fan Chung Graham and Lencho La in 2002*

$G$  is a tree on  $n$  vertices.



$G$  is connected with  
no cycle.



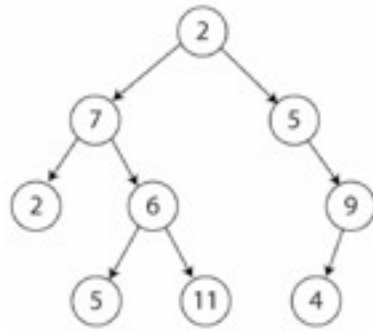
$G$  is connected with  $n-1$  edges.



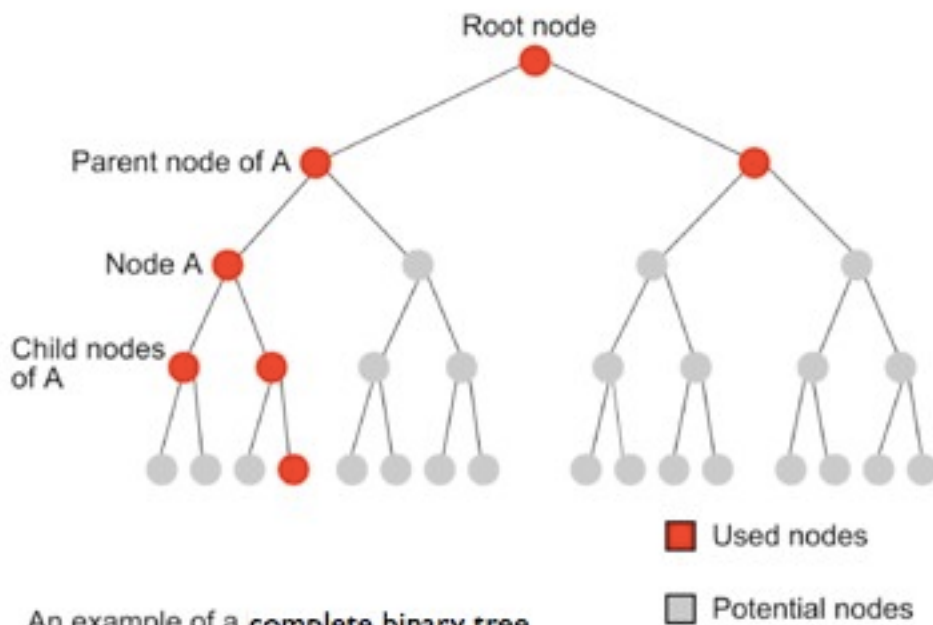
$G$  is formed by adding a leaf to a  
tree of  $n-1$  vertices.



There is a unique path between any  
two vertices.



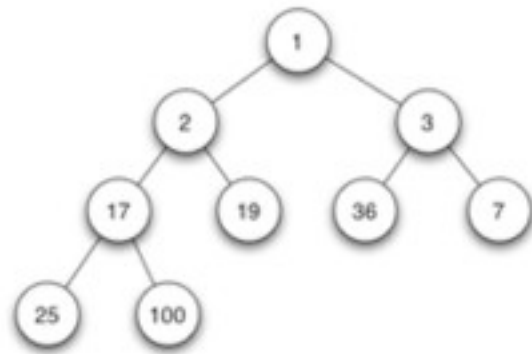
A binary tree



An example of a complete binary tree

A k-level complete binary tree has ?? vertices.

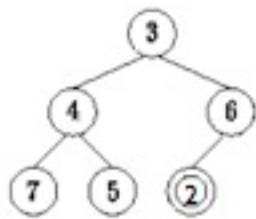
## Binary heap



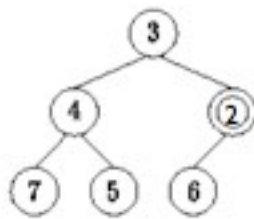
A binary heap is a useful data structure if you have a collection of objects to which you are adding more objects and need to remove the object with the lowest value.

**Insert? Delete?**

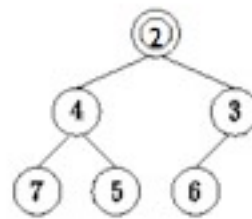
## inserting an item into a binary heap



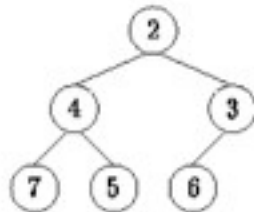
(a)



(b)



(c)





## Chapter 4

### Greedy Algorithms

PEARSON  
Addison  
Wesley

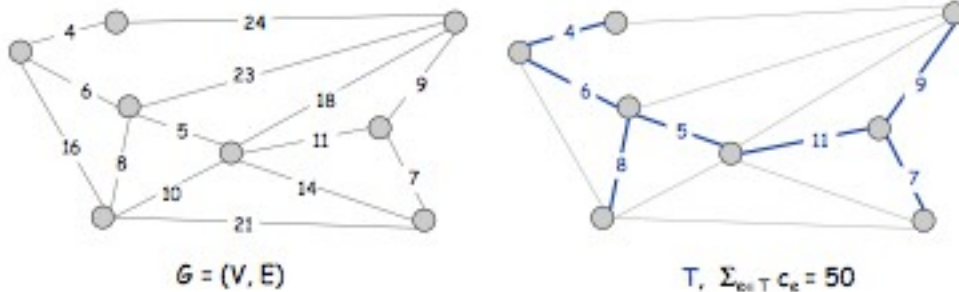
Slides by Kevin Worme,  
Copyright © 2003 Pearson-Addison Wesley.  
All rights reserved.

## 4.5 Minimum Spanning Tree

---

## Minimum Spanning Tree

**Minimum spanning tree.** Given a connected graph  $G = (V, E)$  with real-valued 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.



**Cayley's Theorem.** There are  $n^{n-2}$  spanning trees of  $K_n$ .

↑  
can't solve by brute force

3

## Applications

MST is fundamental problem with diverse applications.

- Network design.
  - telephone, electrical, hydraulic, TV cable, computer, road
- Approximation algorithms for NP-hard problems.
  - traveling salesperson problem, Steiner tree
- Indirect applications.
  - max bottleneck paths
  - LDPC codes for error correction
  - image registration with Renyi entropy
  - learning salient features for real-time face verification
  - reducing data storage in sequencing amino acids in a protein
  - model locality of particle interactions in turbulent fluid flows
  - autoconfig protocol for Ethernet bridging to avoid cycles in a network
- Cluster analysis.

4

## Greedy Algorithms

**Kruskal's algorithm.** Start with  $T = \emptyset$ . 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  $T$ .

**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  $T$ .

**Remark.** All three algorithms produce an MST.

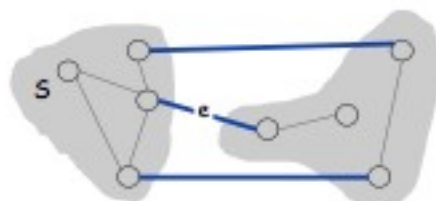
5

## Greedy Algorithms

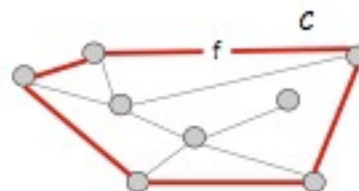
**Simplifying assumption.** All edge costs  $c_e$  are distinct.

**Cut property.** Let  $S$  be any subset of nodes, and let  $e$  be the min cost edge with exactly one endpoint in  $S$ . Then the MST contains  $e$ .

**Cycle property.** Let  $C$  be any cycle, and let  $f$  be the max cost edge belonging to  $C$ . Then the MST does not contain  $f$ .



$e$  is in the MST

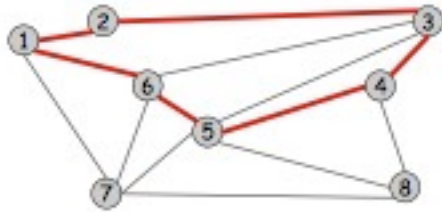


$f$  is not in the MST

6

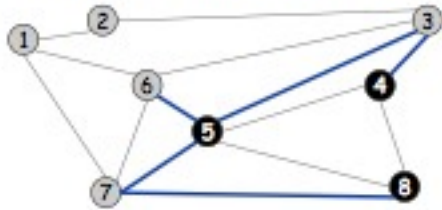
## Cycles and Cuts

**Cycle.** Set of edges the form  $a-b, b-c, c-d, \dots, y-z, z-a$ .



Cycle  $C = 1-2, 2-3, 3-4, 4-5, 5-6, 6-1$

**Cutset.** A cut is a subset of nodes  $S$ . The corresponding cutset  $D$  is the subset of edges with exactly one endpoint in  $S$ .

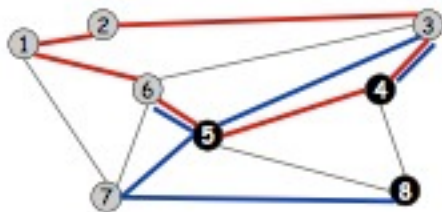


Cut  $S = \{4, 5, 8\}$   
Cutset  $D = 5-6, 5-7, 3-4, 3-5, 7-8$

7

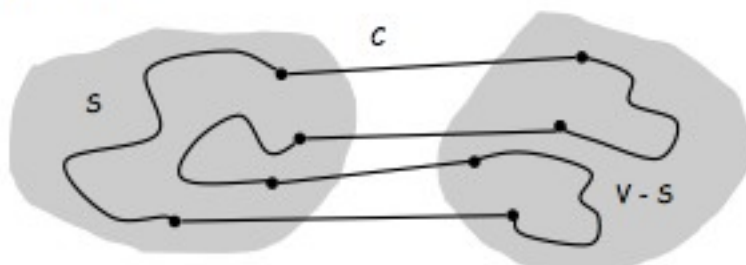
## Cycle-Cut Intersection

**Claim.** A cycle and a cutset intersect in an even number of edges.



Cycle  $C = 1-2, 2-3, 3-4, 4-5, 5-6, 6-1$   
Cutset  $D = 3-4, 3-5, 5-6, 5-7, 7-8$   
Intersection =  $3-4, 5-6$

**Pf.** (by picture)



8

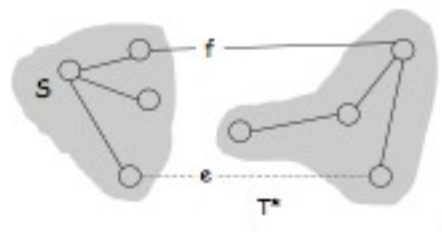
## Greedy Algorithms

**Simplifying assumption.** All edge costs  $c_e$  are distinct.

**Cut property.** Let  $S$  be any subset of nodes, and let  $e$  be the min cost edge with exactly one endpoint in  $S$ . Then the MST  $T^*$  contains  $e$ .

**Pf.** (exchange argument)

- Suppose  $e$  does not belong to  $T^*$ , and let's see what happens.
- Adding  $e$  to  $T^*$  creates a cycle  $C$  in  $T^*$ .
- Edge  $e$  is both in the cycle  $C$  and in the cutset  $D$  corresponding to  $S$   
 $\Rightarrow$  there exists another edge, say  $f$ , that is in both  $C$  and  $D$ .
- $T' = T^* \cup \{e\} - \{f\}$  is also a spanning tree.
- Since  $c_e < c_f$ ,  $\text{cost}(T') < \text{cost}(T^*)$ .
- This is a contradiction. \*



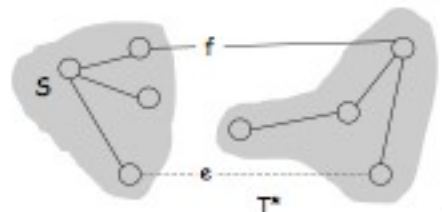
## Greedy Algorithms

**Simplifying assumption.** All edge costs  $c_e$  are distinct.

**Cycle property.** Let  $C$  be any cycle in  $G$ , and let  $f$  be the max cost edge belonging to  $C$ . Then the MST  $T^*$  does not contain  $f$ .

**Pf.** (exchange argument)

- Suppose  $f$  belongs to  $T^*$ , and let's see what happens.
- Deleting  $f$  from  $T^*$  creates a cut  $S$  in  $T^*$ .
- Edge  $f$  is both in the cycle  $C$  and in the cutset  $D$  corresponding to  $S$   
 $\Rightarrow$  there exists another edge, say  $e$ , that is in both  $C$  and  $D$ .
- $T' = T^* \cup \{e\} - \{f\}$  is also a spanning tree.
- Since  $c_e < c_f$ ,  $\text{cost}(T') < \text{cost}(T^*)$ .
- This is a contradiction. \*

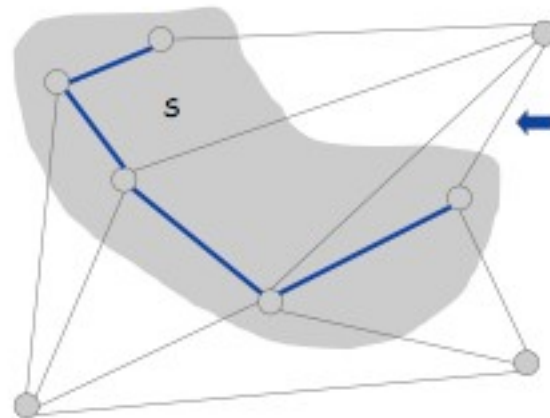




## Prim's Algorithm: Proof of Correctness

Prim's algorithm. [Jarník 1930, Dijkstra 1957, Prim 1959]

- Initialize  $S$  = any node.
- Apply cut property to  $S$ .
- Add min cost edge in cutset corresponding to  $S$  to  $T$ , and add one new explored node  $u$  to  $S$ .



11

## Implementation: Prim's Algorithm

Implementation. Use a priority queue ala Dijkstra.

- Maintain set of explored nodes  $S$ .
- For each unexplored node  $v$ , maintain attachment cost  $a[v]$  = cost of cheapest edge  $v$  to a node in  $S$ .
- $O(n^2)$  with an array;  $O(m \log n)$  with a binary heap.

```
Prim(G, c) {  
  foreach (v ∈ V) a[v] ← ∞  
  Initialize an empty priority queue Q  
  foreach (v ∈ V) insert v onto Q  
  Initialize set of explored nodes S ← ∅  
  
  while (Q is not empty) {  
    u ← delete min element from Q  
    S ← S ∪ { u }  
    foreach (edge e = (u, v) incident to u)  
      if ((v ∉ S) and (ce < a[v]))  
        decrease priority a[v] to ce  
  }  
}
```

12

## Implementing Prim's algorithm

Prim's algorithm finds a minimum spanning tree.

Using a priority queue

Maintaining minimum

## Prim's algorithm

Prim's algorithm finds a minimum spanning tree.

Input:  $G, c$

Initially,  $S = \{s\} = \{\text{vertices in the current tree}\}$

For  $V - S \neq \emptyset$ , select  $u \notin S$  but adjacent to some vertex in  $S$ .

Compute  $f(u) = \min_{\substack{e=(w,u) \\ w \in S}} c(e)$

Pick  $v$  so that  $f(v) = \min_{u \in S} \{f(u)\}$

Set  $S \leftarrow S \cup \{v\}$  and add edge  $\{w, v\}$ .

Endfor

## Implementing Prim's algorithm

Prim's algorithm finds a minimum spanning tree.

Input:  $G, c$

Initially,  $S = \{s\} = \{\text{vertices in the current tree}\}$  **heap**

For  $S' = V - S \neq \emptyset$ , ~~select  $u \notin S$  but adjacent to some~~  
**Priority heap** ~~vertex in  $S$ .~~

Compute  ~~$f(u) = \min_{\substack{e=(w,u) \\ w \in S}} c(e)$~~

Pick  $v$  so that  ~~$f(v) = \min_{u \in S'} \{f(u)\}$~~  **by extractmin**

Set  $S \leftarrow S \cup \{v\}$  and add edge  $\{w, v\}$ .

**Update  $S'$  for neighbors  $u$  of  $v$**

Endfor  $f(u) \leftarrow \min\{c(\{u, v\}), \text{old } f(u)\}$

## Implementing Prim's algorithm

Prim's algorithm finds a minimum spanning tree.

Running time: at most  $n$  extractmin

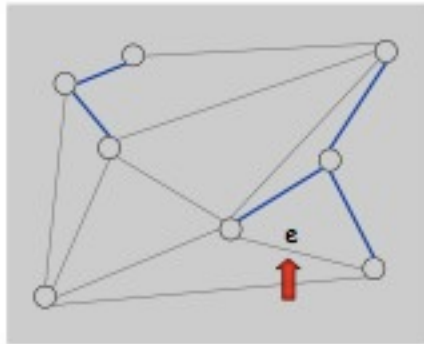
and at most  $m$  changekey

Total:  $O(m \log n)$ .

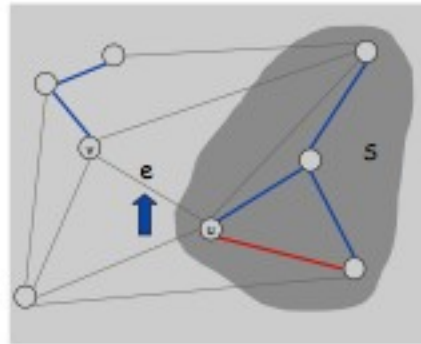
## Kruskal's Algorithm: Proof of Correctness

Kruskal's algorithm. [Kruskal, 1956]

- Consider edges in ascending order of weight.
- Case 1: If adding  $e$  to  $T$  creates a cycle, discard  $e$  according to cycle property.
- Case 2: Otherwise, insert  $e = (u, v)$  into  $T$  according to cut property where  $S =$  set of nodes in  $u$ 's connected component.



Case 1



Case 2

13

## Implementation: Kruskal's Algorithm

Implementation. Use the **union-find** data structure.

- Build set  $T$  of edges in the MST.
- Maintain set for each connected component.
- $O(m \log n)$  for sorting and  $O(m \alpha(m, n))$  for union-find.

$m \leq n^2 \Rightarrow \log m$  is  $O(\log n)$       essentially a constant

```

Kruskal(G, c) {
  Sort edges weights so that  $c_1 \leq c_2 \leq \dots \leq c_m$ .
   $T \leftarrow \emptyset$ 

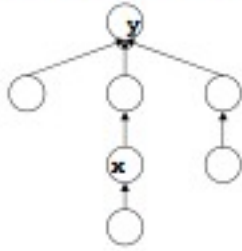
  foreach ( $u \in V$ ) make a set containing singleton  $u$ 

  for  $i = 1$  to  $m$    are  $u$  and  $v$  in different connected components?
    ( $u, v$ ) =  $e_i$ 
    if ( $u$  and  $v$  are in different sets) {
       $T \leftarrow T \cup \{e_i\}$ 
      merge the sets containing  $u$  and  $v$ 
    }
  }
  return  $T$ 
}
    
```

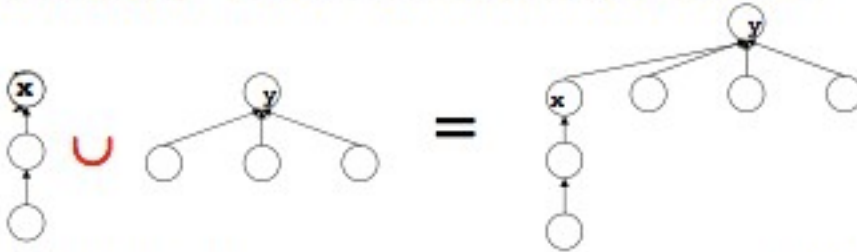
14

# Union-Find Implementation

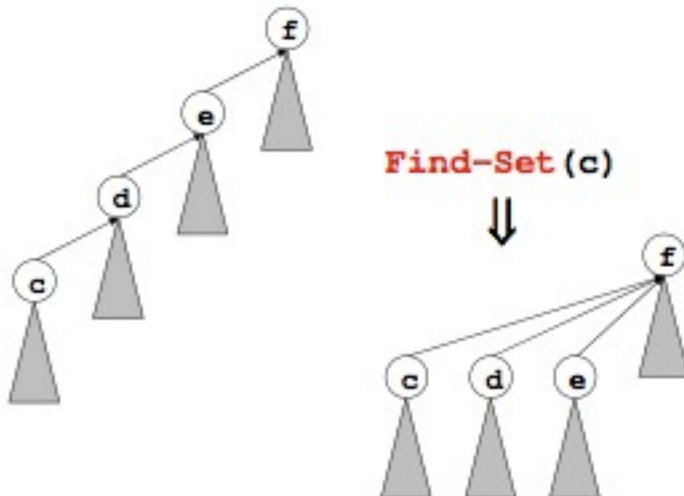
**Find-Set (x)** -  
follow pointers  
from **x** up to root



**Union (x, y)** - make **x** a child of **y** and return **y**



# Path Compression



## The function $\lg^*n$

$\lg^*n$  = the number of times we have to take the  $\log_2$  of  $n$  repeatedly to reach 1

- $\lg^* 2 = 1$
- $\lg^* 3 = \lg^* 4 = \lg^* 2^2 = 2$
- $\lg^* 16 = \lg^* 2^{2^2} = 3$
- $\lg^* 65536 = \lg^* 2^{2^{2^{2^2}}} = 4$

$\Rightarrow \lg^*n \leq 5$  for all practical values of  $n$

Theorem(Tarjan): If

$S$  = a sequence of  $O(n)$  Unions and Find-Sets

The worst-case time for  $S$  with

- Weighted Unions, and
- Path Compressions

is  $O(n\lg^*n)$

$\Rightarrow$  The average time is  $O(\lg^*n)$  per operation

# Ackerman's function

Define the function  $A_i(x)$  inductively by

$$A_0(x) = x+1$$

$$A_{i+1}(x) = A_i(A_i(A_i \dots (x))), \text{ where } A_i \text{ is applied } x+1 \text{ times.}$$

$$A_1(x) = 2x+1$$

$$A_2(x) = A_1(A_1(A_1 \dots (x))) > 2^{x+1}$$

$$A_3(x) = A_2(A_2(A_2 \dots (x))) > \underbrace{2^{2^{2^{\dots (x)}}}}_{x+1}$$

The **Inverse Ackerman** function is

$$\alpha(n) = \min\{k: A_k(1) > n\}$$

Theorem(Tarjan): Let

$S$  = sequence of  $\Omega(n)$  **Unions** and **Find-Sets**

The worst-case time for  $S$  with

- Weighted Unions, and
- Path Compressions

is  $O(n\alpha(n))$

$\Rightarrow$  The average time is  $O(\alpha(n))$  steps per operation

## Implementation: Kruskal's Algorithm

**Implementation.** Use the **union-find** data structure.

- Build set  $T$  of edges in the MST.
- Maintain set for each connected component.
- $O(m \log n)$  for sorting and  $O(m \alpha(m, n))$  for union-find.

$m \leq n^2 \Rightarrow \log m$  is  $O(\log n)$       essentially a constant

```
Kruskal(G, c) {
  Sort edges weights so that  $c_1 \leq c_2 \leq \dots \leq c_m$ .
   $T \leftarrow \emptyset$ 

  foreach ( $u \in V$ ) make a set containing singleton  $u$ 

  for  $i = 1$  to  $m$     are  $u$  and  $v$  in different connected components?
    ( $u, v$ ) =  $e_i$     ←
    if ( $u$  and  $v$  are in different sets) {
       $T \leftarrow T \cup \{e_i\}$ 
      merge the sets containing  $u$  and  $v$ 
    }                    ← merge two components
  return  $T$ 
}
```

14

## Lexicographic Tiebreaking

To remove the assumption that all edge costs are distinct: perturb all edge costs by tiny amounts to break any ties.

**Impact.** Kruskal and Prim only interact with costs via pairwise comparisons. If perturbations are sufficiently small, MST with perturbed costs is MST with original costs.

↑  
e.g. if all edge costs are integers,  
perturbing cost of edge  $e$ , by  $i / n^2$

**Implementation.** Can handle arbitrarily small perturbations implicitly by breaking ties lexicographically, according to index.

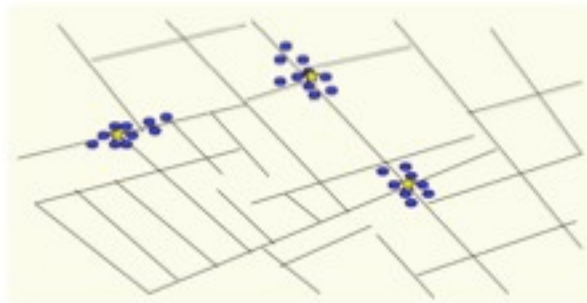
```
boolean less(i, j) {
  if (cost( $e_i$ ) < cost( $e_j$ )) return true
  else if (cost( $e_i$ ) > cost( $e_j$ )) return false
  else if (i < j) return true
  else return false
}
```

15



## 4.7 Clustering

---



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

### Clustering

**Clustering.** Given a set  $U$  of  $n$  objects labeled  $p_1, \dots, p_n$ , classify into coherent groups.

↑  
photos, documents, micro-organisms

**Distance function.** Numeric value specifying "closeness" of two objects.

↑  
number of corresponding pixels whose intensities differ by some threshold

**Fundamental problem.** Divide into clusters so that points in different clusters are far apart.

- Routing in mobile ad hoc networks.
- Identify patterns in gene expression.
- Document categorization for web search.
- Similarity searching in medical image databases
- Skycat: cluster  $10^9$  sky objects into stars, quasars, galaxies.

## Clustering of Maximum Spacing

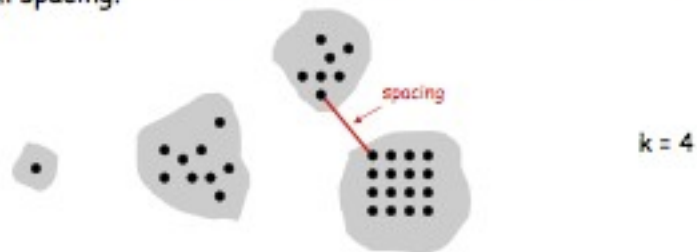
**k-clustering.** Divide objects into  $k$  non-empty groups.

**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)

**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.



18

## 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.

**Key observation.** This procedure is precisely 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.

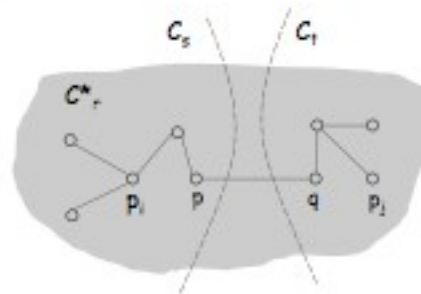
19

## Greedy Clustering Algorithm: Analysis

**Theorem.** Let  $C^*$  denote the clustering  $C_1^*, \dots, C_k^*$  formed by deleting the  $k-1$  most expensive edges of a MST.  $C^*$  is a  $k$ -clustering of max spacing.

**Pf.** Let  $C$  denote some other clustering  $C_1, \dots, C_x$ .

- The spacing of  $C^*$  is the length  $d^*$  of the  $(k-1)^{st}$  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  $\leq d^*$  since  $p$  and  $q$  are in different clusters. •



## Extra Slides

---

## MST Algorithms: Theory

### Deterministic comparison based algorithms.

- $O(m \log n)$  [Jarník, Prim, Dijkstra, Kruskal, Boruvka]
- $O(m \log \log n)$ . [Cheriton-Tarjan 1976, Yao 1975]
- $O(m \beta(m, n))$ . [Fredman-Tarjan 1987]
- $O(m \log \beta(m, n))$ . [Gabow-Galil-Spencer-Tarjan 1986]
- $O(m \alpha(m, n))$ . [Chazelle 2000]

Holy grail.  $O(m)$ .

### Notable.

- $O(m)$  randomized. [Karger-Klein-Tarjan 1995]
- $O(m)$  verification. [Dixon-Rauch-Tarjan 1992]

### Euclidean.

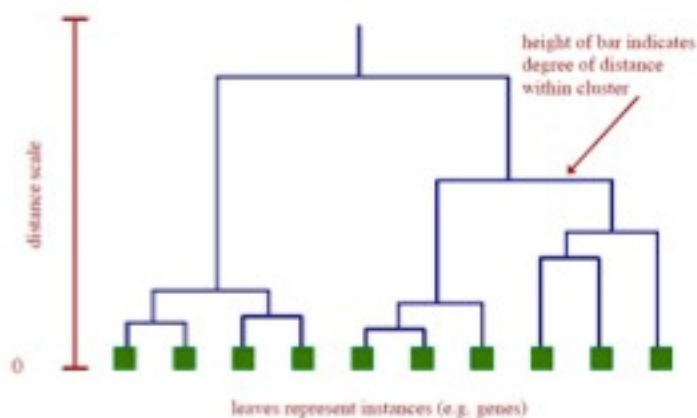
- 2-d:  $O(n \log n)$ . compute MST of edges in Delaunay
- k-d:  $O(k n^2)$ . dense Prim

22

## Dendrogram

**Dendrogram.** Scientific visualization of hypothetical sequence of evolutionary events.

- Leaves = genes.
- Internal nodes = hypothetical ancestors.



Reference: <http://www.biostat.wisc.edu/bmi576/fall-2003/lecture13.pdf>

23

# Dendrogram of Cancers in Human

Tumors in similar tissues cluster together.

