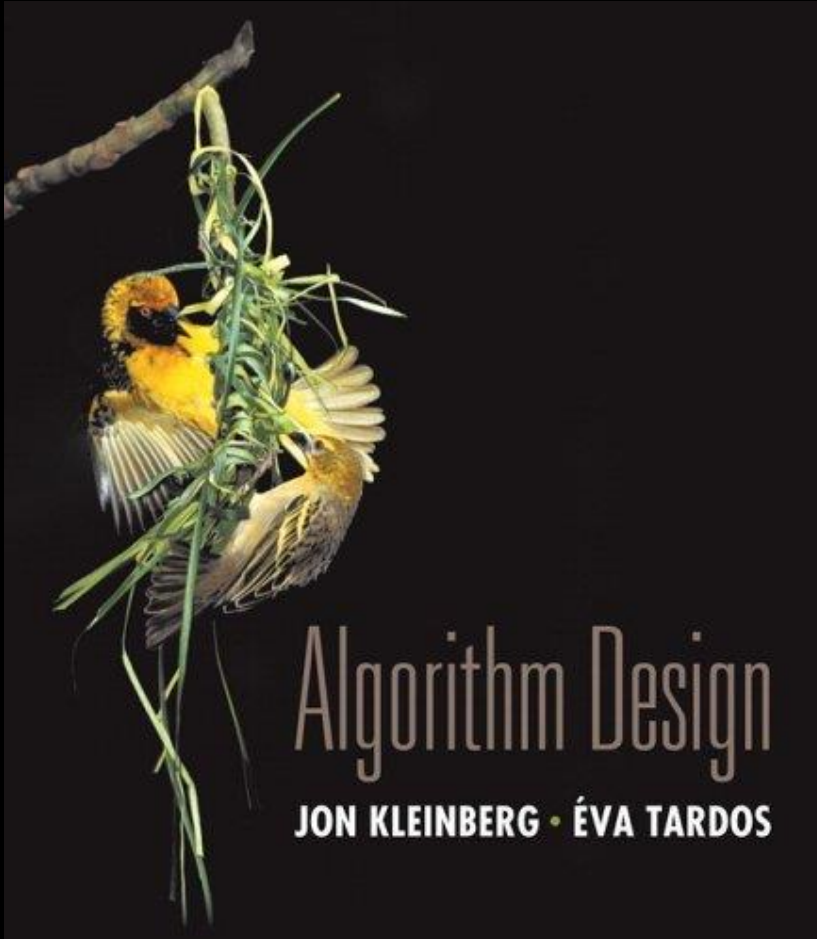


# Chapter 4

## Greedy Algorithms



Slides by Kevin Wayne.  
Copyright © 2005 Pearson-Addison Wesley.  
All rights reserved.

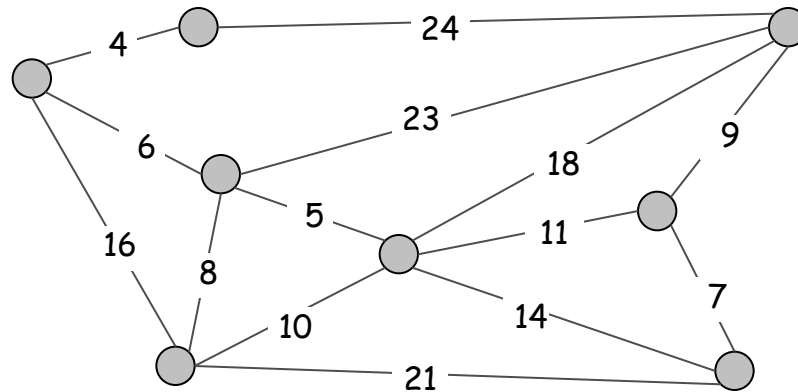
## 4.5 Minimum Spanning Tree

---

# Minimum Spanning Tree

**Minimum spanning tree.** Given a connected graph  $G = (V, E)$  with real-valued edge **positive** weights  $c_e$ , find a subset  $E' \subseteq E$  such that

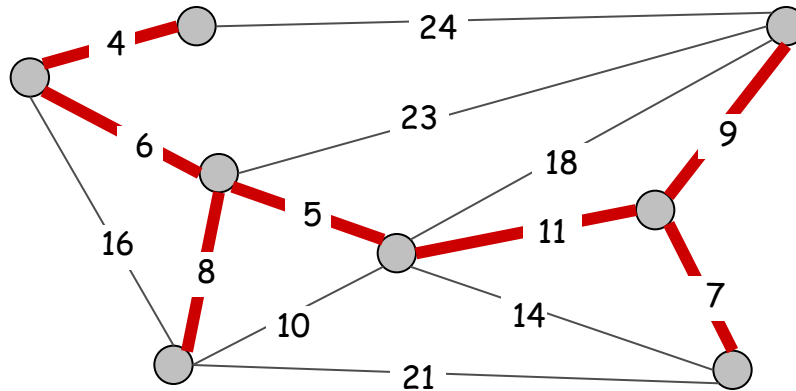
- (i) the graph  $G'=(V,E')$  is connected
- (ii) smallest possible cost



# Minimum Spanning Tree

**Minimum spanning tree.** Given a connected graph  $G = (V, E)$  with real-valued edge **positive** weights  $c_e$ , find a subset  $E' \subseteq E$  such that

- (i) the graph  $G'=(V,E')$  is connected
- (ii) smallest possible cost



**Key Observation.** The optimal solution **does not contain cycles**  $\Rightarrow$  it is a tree

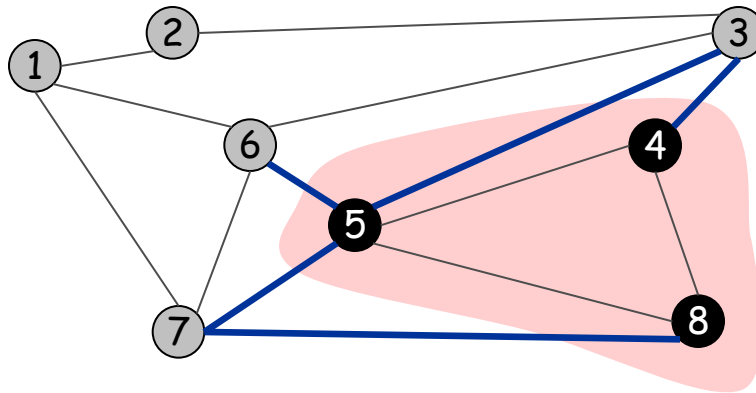
# Applications

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.**

# Cycles and Cuts

**Cut.** A cut for a graph  $G=(V,E)$  is a subset of nodes  $S$ .



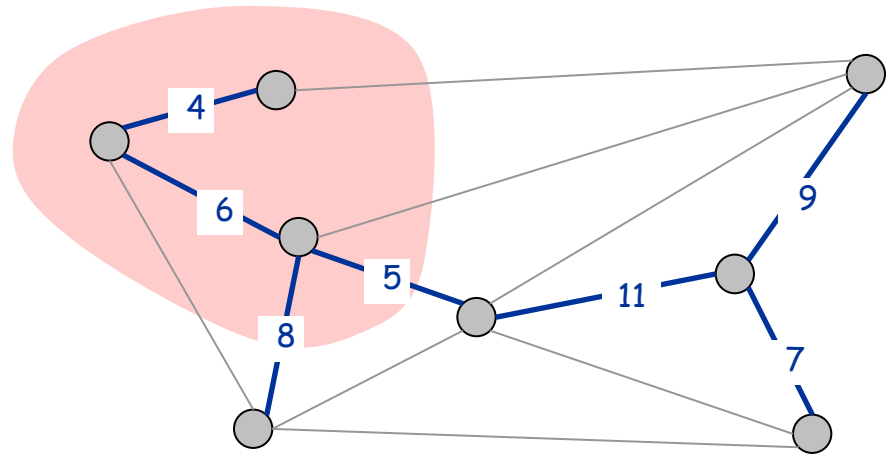
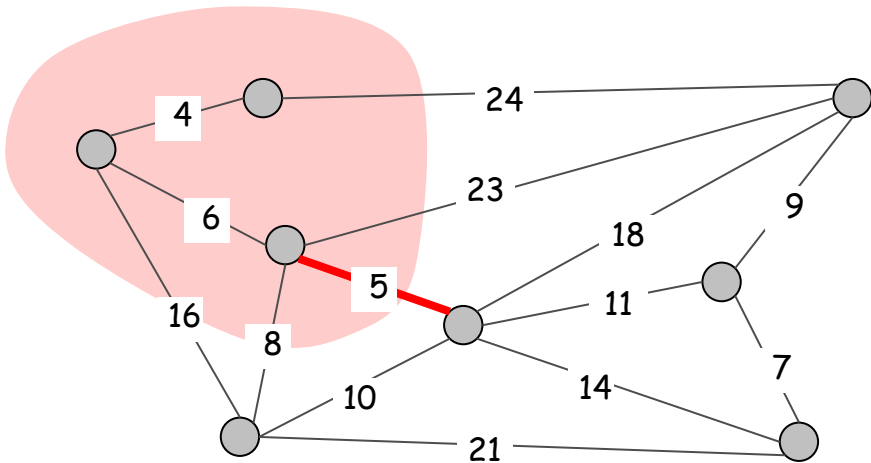
Cut  $S = \{4, 5, 8\}$   
Crossing edges = 5-6, 5-7, 3-4, 3-5, 7-8

- An edge  $e$  **crosses** a cut  $S$  if  $e$  has an endpoint in  $S$  and the other one in  $V-S$

# Greedy Algorithms

To simplify, assume all edge weights are different  $\Rightarrow$  there is **unique MST**

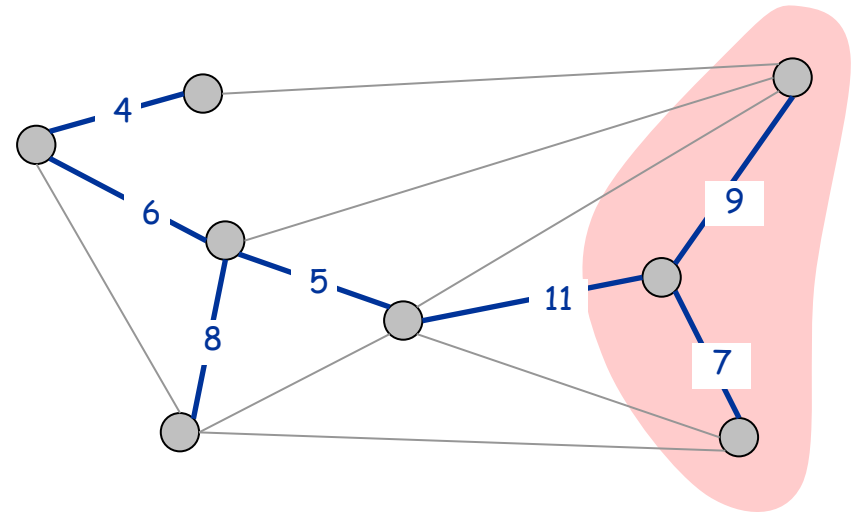
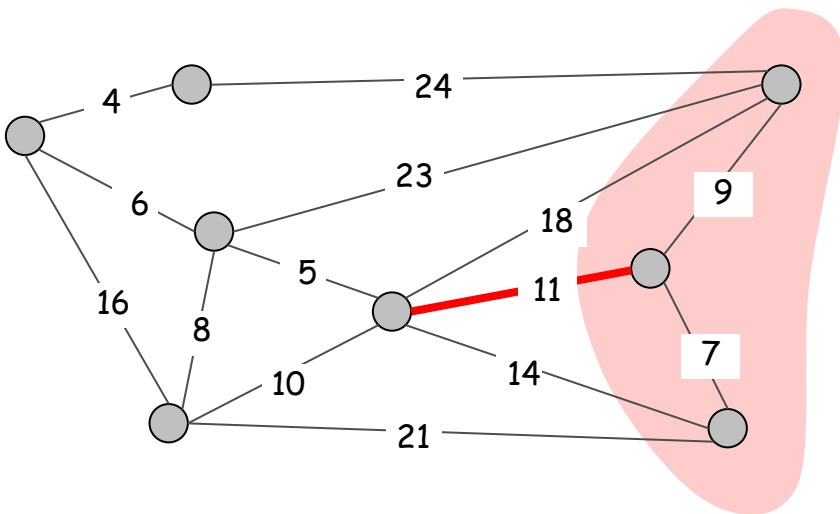
**Cut property** . Consider a graph  $G$ . Pick any cut  $S$ . If  $e$  is the lightest edge that crosses  $S$ , then  $e$  belongs to the MST



# Greedy Algorithms

To simplify, assume all edge weights are different  $\Rightarrow$  there is **unique MST**

**Cut property.** Consider a graph  $G$ . Pick any cut  $S$ . If  $e$  is the lightest edge that crosses  $S$ , then  $e$  belongs to the MST





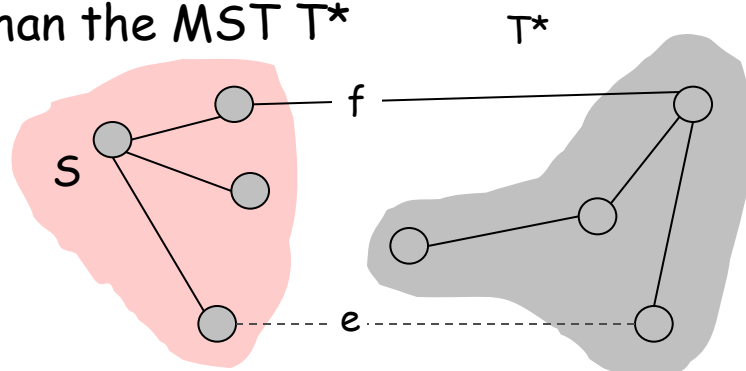
# Greedy Algorithms

To simplify, assume all edge weights are different  $\Rightarrow$  there is **unique MST**

**Cut property.** Consider a graph  $G$ . Pick any cut  $S$ . If  $e$  is the lightest edge that crosses  $S$ , then  $e$  belongs to the MST

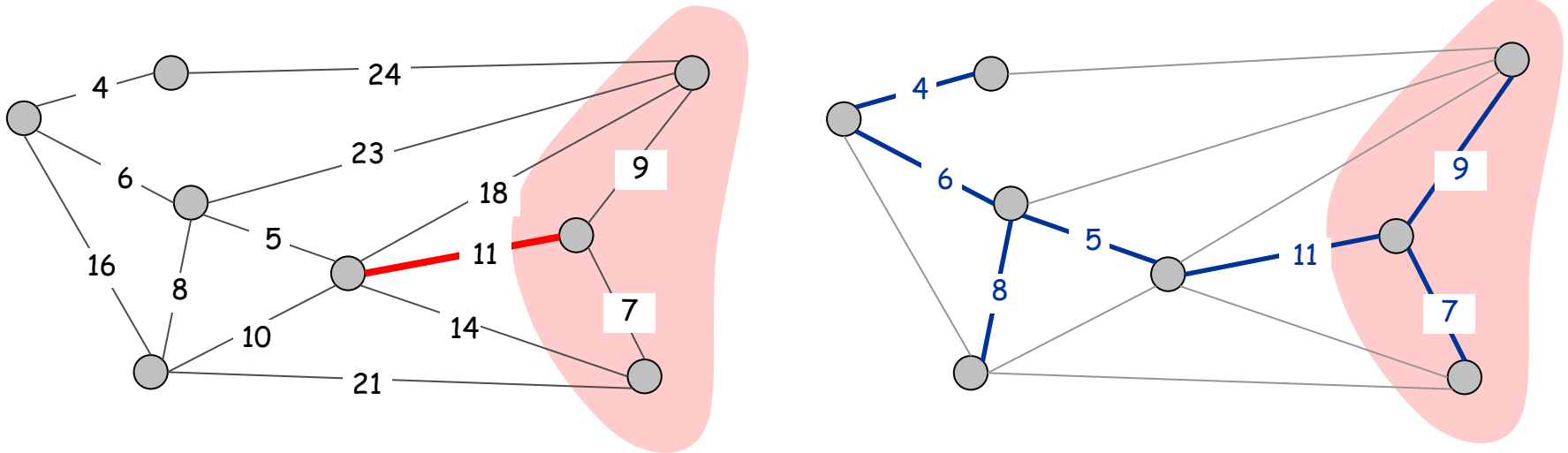
Pf. (exchange argument)

- Consider the MST  $T^*$
- Suppose lightest edge  $e$  does not belong to  $T^*$   $\Rightarrow$  when we add  $e$  to  $T^*$  we create a **cycle**
- This cycle has **another edge  $f$**  crossing the cut  $S$
- $T' = T^* \cup \{e\} - \{f\}$  is also a spanning tree
- Since  $c_e < c_f$ , the new tree  $T'$  is **cheaper** than the MST  $T^*$   
 $\Rightarrow$  contradiction



# Greedy Algorithms

This property help us to start building the MST: just look at any cut, add the lightest edge to the solution



We can continue to find other edge to add

# Greedy Algorithms

## Cut property.

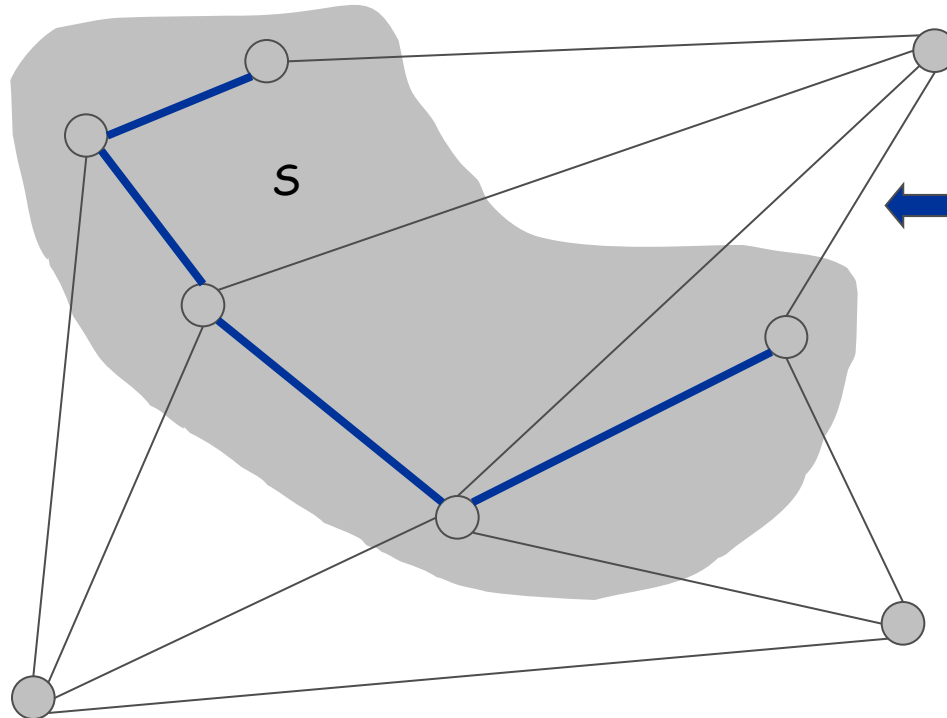
- Different applications of Cut property lead to different algorithms for constructing Minimal Spanning Trees.
- Prim and Kruskal algorithm construct a MST applying the Cut property  $n-1$  times.

# Prim's Algorithm

Idea: keep growing the **same cut**

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

- Initialize  $S$  = any node, tree  $T$  = empty
- Apply cut property to  $S$ .
- Add min cost edge that crosses  $S$  to  $T$ , and add one new explored node  $u$  to  $S$ .



# Bad Implementation: Prim's Algorithm

## Implementation (Naïve)

- Maintain set of explored nodes  $S$ .
- Find the lightest edge that crosses  $S$  in  $O(m)$  time
- Total complexity  $O(m.n)$

# Good Implementation: Prim's Algorithm

**Implementation.** Use a priority queue.

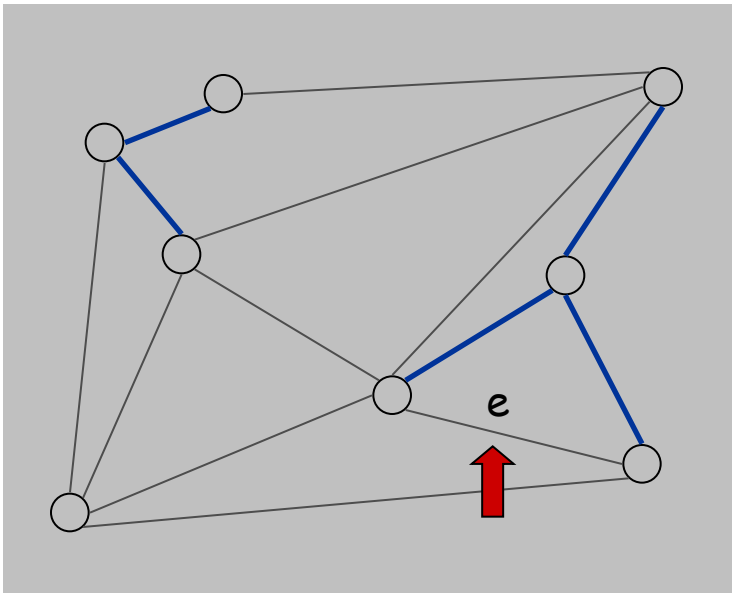
- Maintain set of explored nodes  $S$ .
- For each unexplored node  $v$ , maintain attachment cost  $a[v] = \text{cost of cheapest edge } v \text{ to a node in } S$ .
- $O(n^2)$  with an array;  $O(m \log n)$  with a binary heap.

```
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  
    }  
}
```

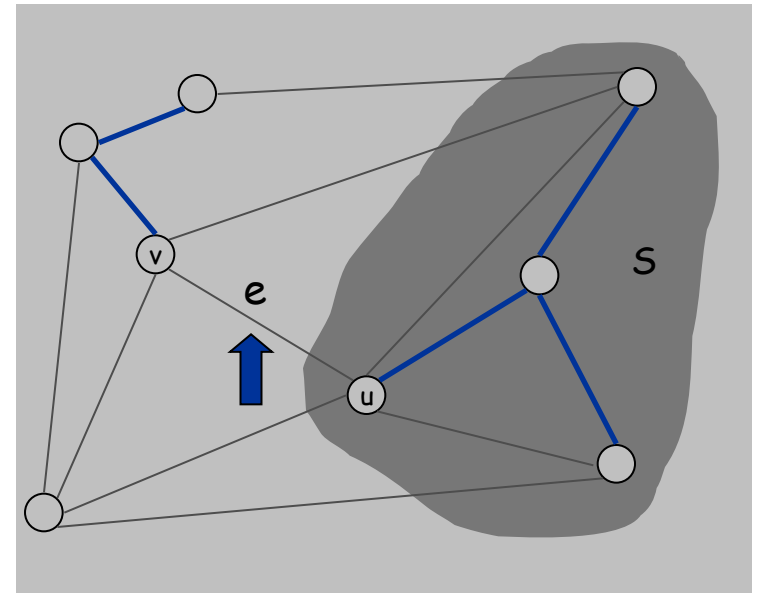
# Kruskal's Algorithm

Kruskal's algorithm. [Kruskal, 1956]

- Consider edges in ascending order of weight.
- Case 1: If adding  $e$  to  $T$  creates a cycle, discard  $e$
- Case 2: Otherwise, insert  $e = (u, v)$  into  $T$   
(set  $S$  to be the connected component containing  $u$ )



Case 1

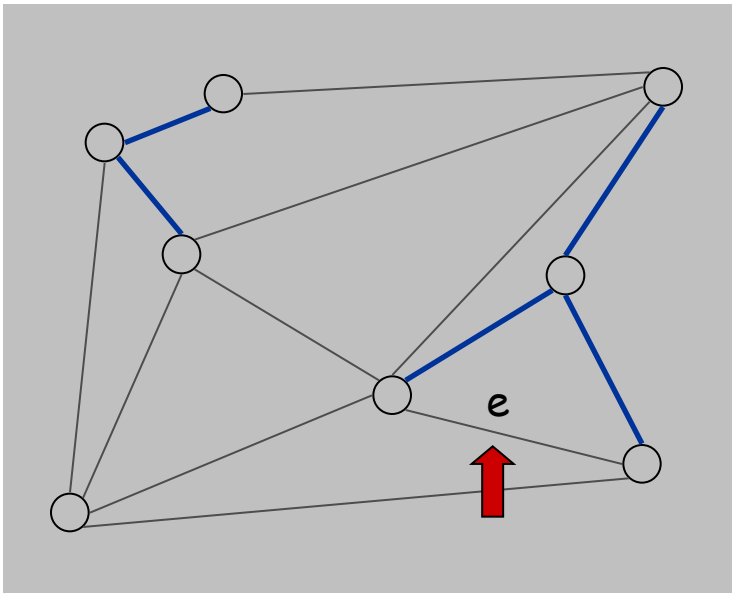


Case 2

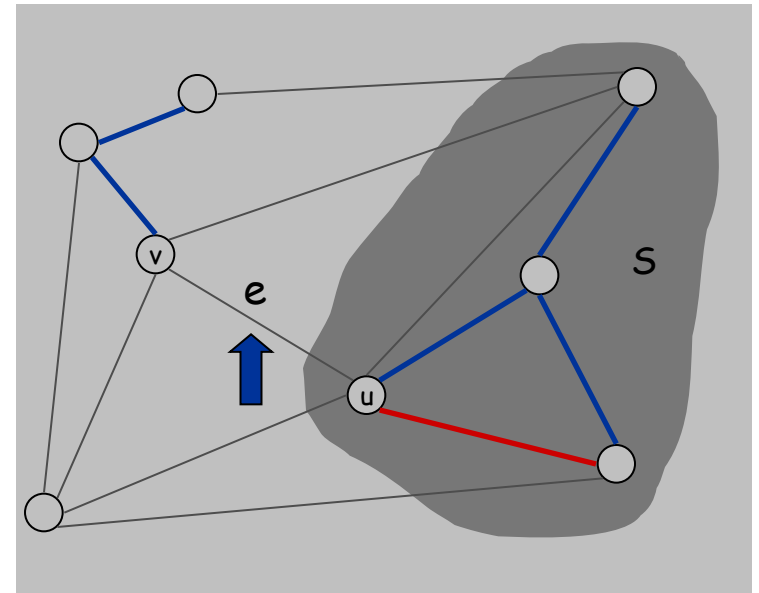
# Kruskal's Algorithm: Proof of correctness

Kruskal's algorithm. [Kruskal, 1956]

- Case 1: If adding  $e$  to  $T$  creates a cycle, discard  $e$ 
  - Optimal solution does not have a cycle
- Case 2: Otherwise, insert  $e = (u, v)$  into  $T$ 
  - Pick the cut  $S$  as the nodes that are reachable from  $u$  in  $T$



Case 1



Case 2



# Kruskal's Algorithm: Bad Implementation

Kruskal's algorithm. [Kruskal, 1956]

- Sorting the edges  $O(m \log m)$
- Testing the existence of a cycle while considering edge  $e$ :  $O(n)$  via a DFS( BFS). Note that a tree has at most  $n$  edges.
- For all edges  $O(m.n)$
- Total complexity  $O(m \log m) + O(m n) = O(n.m)$

# 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  $O(\log n)$        $\underbrace{\hspace{2em}}$  essentially a constant

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

  foreach (u ∈ V) make a set containing singleton u

  for i = 1 to m    are u and v in different connected components?
    (u, v) = ei
    if (u and v are in different sets) {
      T ← T ∪ {ei}
      merge the sets containing u and v
    }
  return T
}
```

merge two components

# 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. $O(m)$ .

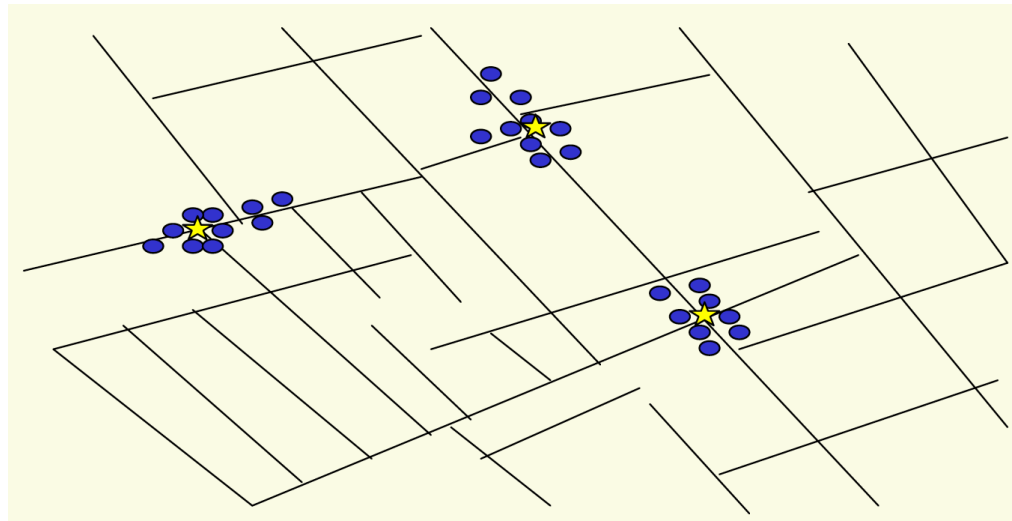
## Notable.

- $O(m)$  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

## 4.7 Clustering



Outbreak of cholera deaths in London in 1850s.  
Reference: Nina 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.

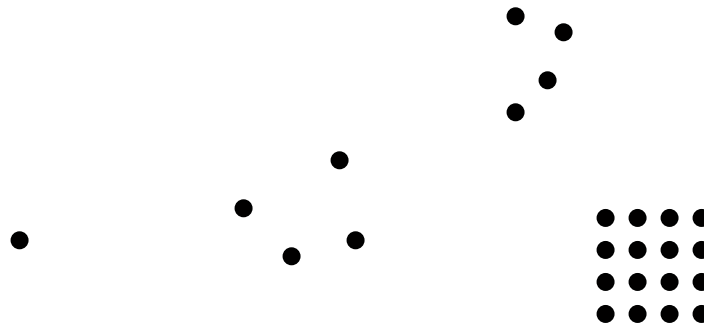
# Clustering

**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

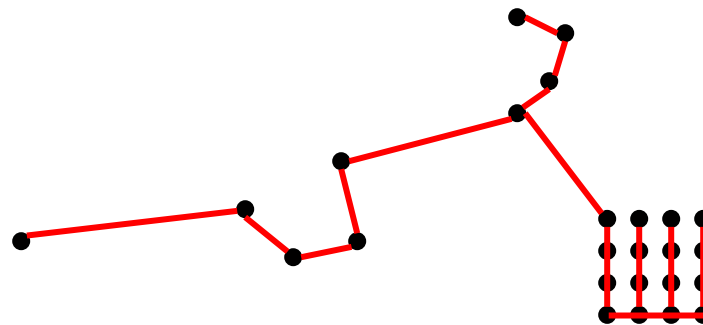


# Clustering of Maximum Spacing

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

**Q:** Can we use an MST to perform  $k$ -clustering?

**A:** Start with MST, keep removing heaviest edge until we get  $k$  connected components



# Clustering of Maximum Spacing

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

**Q:** Can we use an MST to perform  $k$ -clustering?

**A:** Start with MST, keep removing heaviest edge until we get  $k$  connected components

**Guarantees:**

1. This algorithm **gives cheapest way of forming  $k$  connected components** (generalizes MST, which gives cheapest 1 conn. comp)
2. Maximizes **spacing**: minimum space between different classes

