

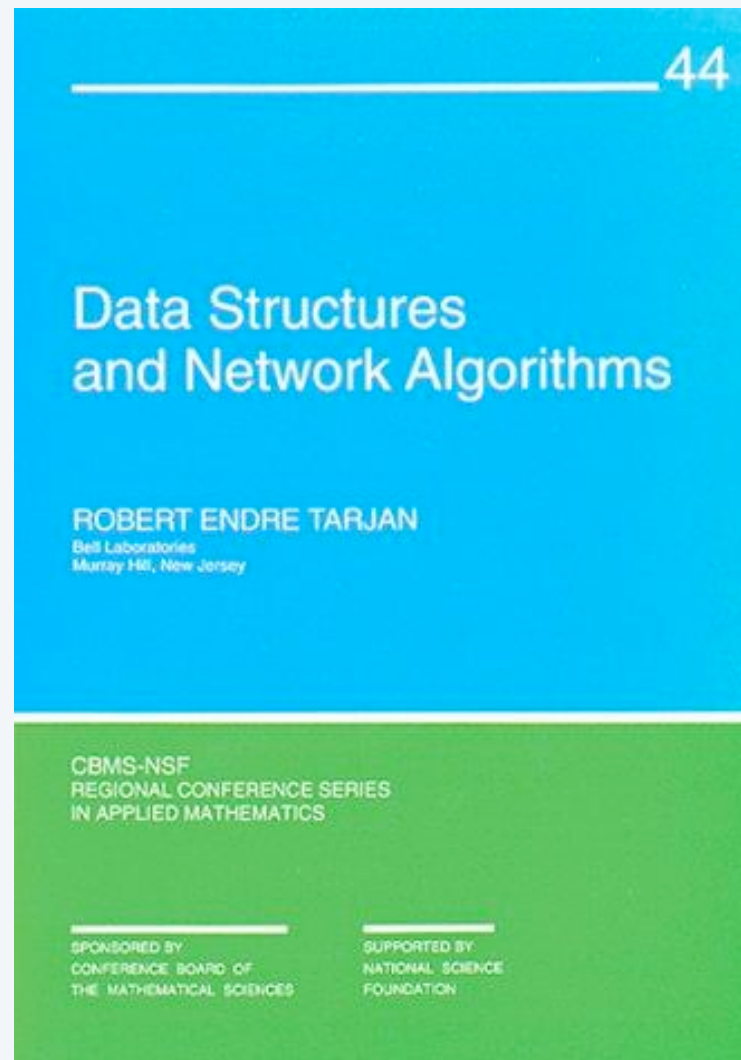
CÂY BAO TRÙM

- ▶ *cây bao trùm*
- ▶ *thuật toán Prim, Kruskal, Boruvka*
- ▶ *phân cụm liên kết đơn (single-link clustering)*

Lecture slides by Kevin Wayne

Copyright © 2005 Pearson–Addison Wesley

<http://www.cs.princeton.edu/~wayne/kleinberg-tardos>



SECTION 6.1

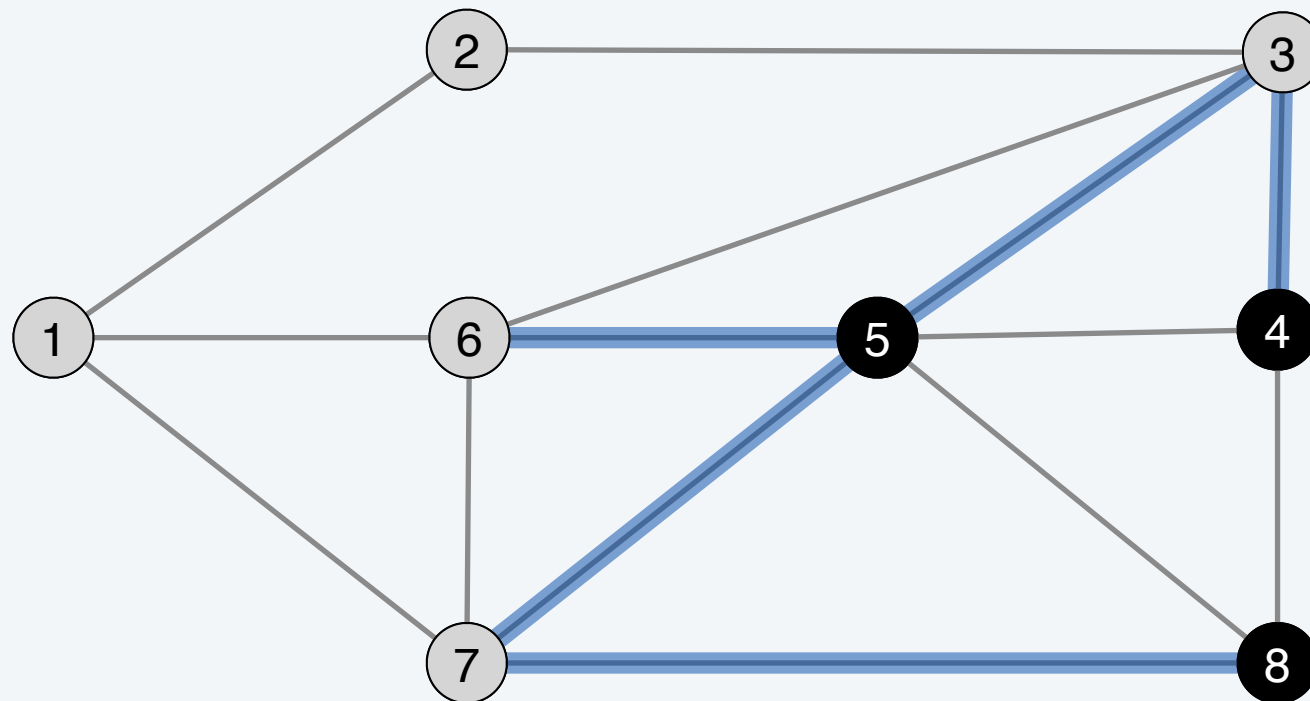
CÂY BAO TRÙM

- ▶ *cây bao trùm*
- ▶ *thuật toán Prim, Kruskal, Boruvka*
- ▶ *phân cụm liên kết đơn*

Lát cắt

Định nghĩa. Một **lát cắt (cut)** là một phân hoạch các điểm thành hai tập con khác rỗng S và $V - S$.

Định nghĩa. Tập **cạnh cắt (cutset)** của một lát cắt S là tập cạnh có đúng một đỉnh trong S .



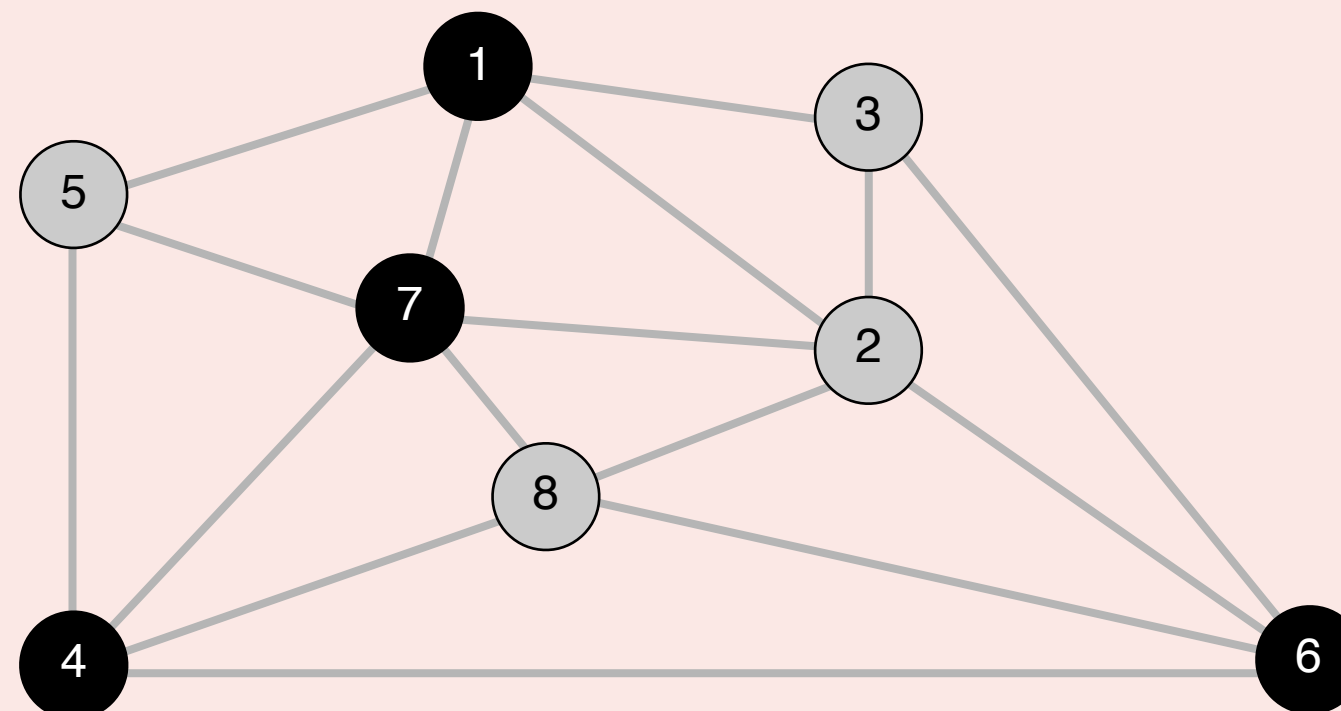
$$\text{cut } S = \{ 4, 5, 8 \}$$

$$\text{cutset } D = \{ (3, 4), (3, 5), (5, 6), (5, 7), (8, 7) \}$$



Consider the cut $S = \{ 1, 4, 6, 7 \}$. Which edge is in the cutset of S ?

- A. S is not a cut (not connected)
- B. 1–7
- C. 5–7
- D. 2–3



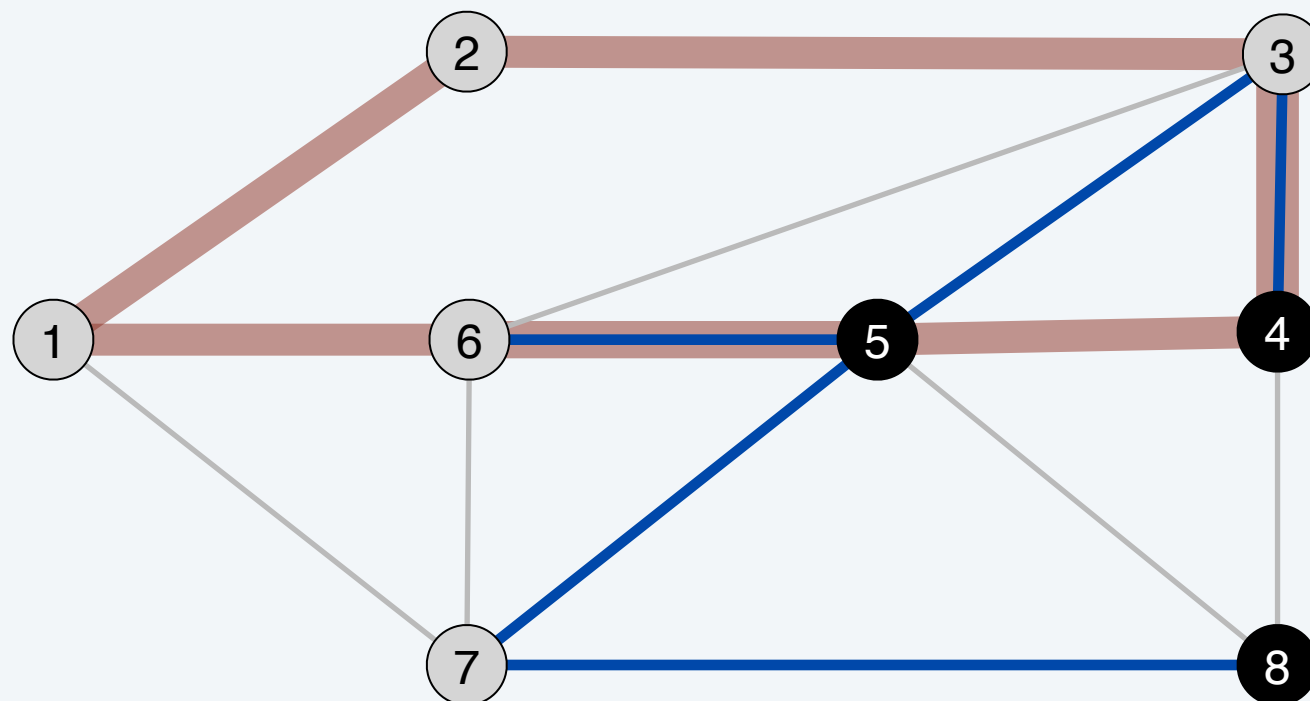


Let C be a cycle and let D be a cutset. How many edges do C and D have in common? Choose the best answer.

- A. 0
- B. 2
- C. not 1
- D. an even number

Giao của chu trình và lát cắt

Mệnh đề. Một chu trình và một tập cạnh cắt giao với nhau bởi một **số chẵn** các cạnh.



cycle $C = \{ (1, 2), (2, 3), (3, 4), (4, 5), (5, 6), (6, 1) \}$

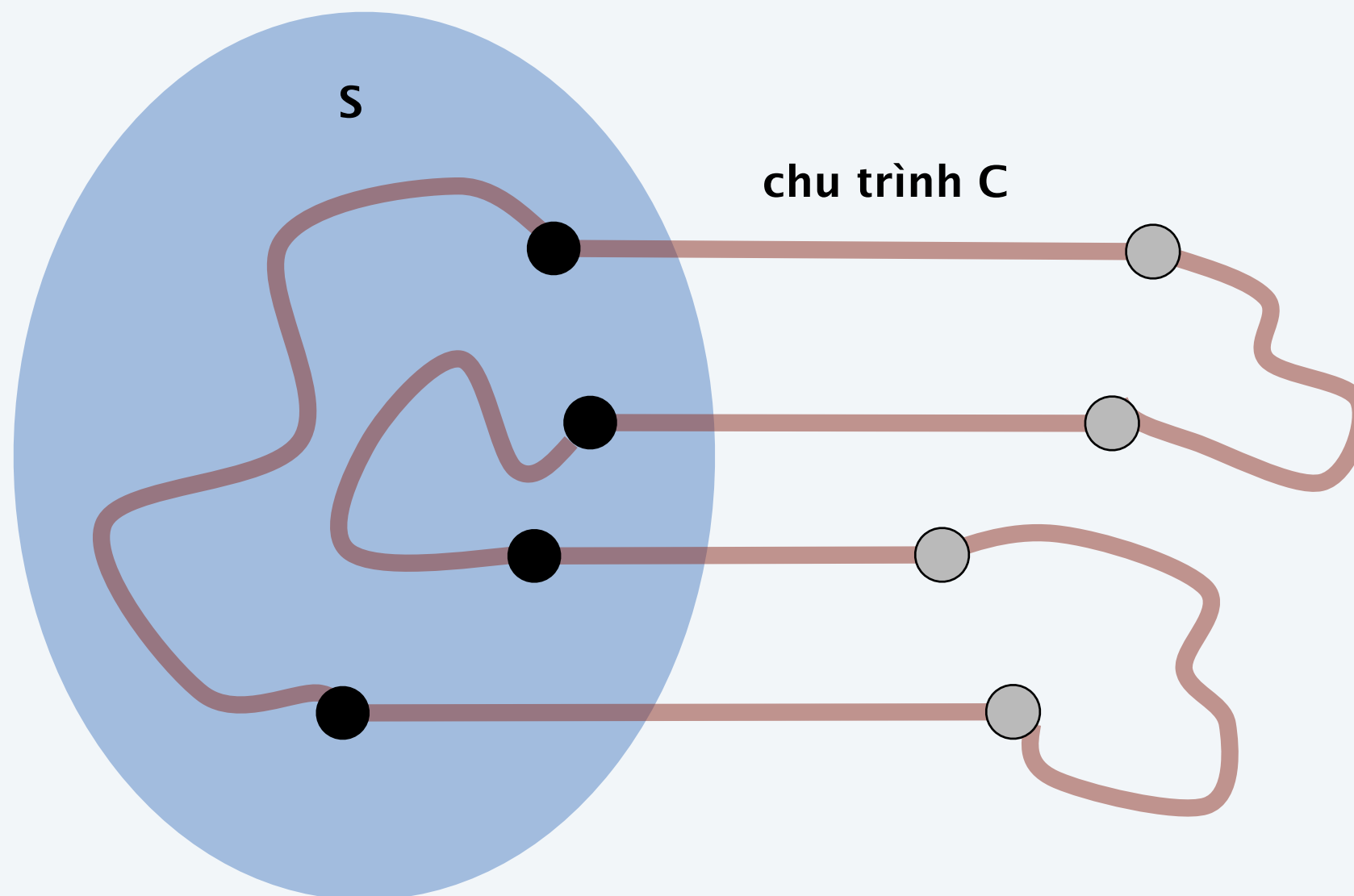
cutset $D = \{ (3, 4), (3, 5), (5, 6), (5, 7), (8, 7) \}$

intersection $C \cap D = \{ (3, 4), (5, 6) \}$

Giao của chu trình và lát cắt

Mệnh đề. Một chu trình và một tập cạnh cắt giao với nhau bởi một **số chẵn** các cạnh.

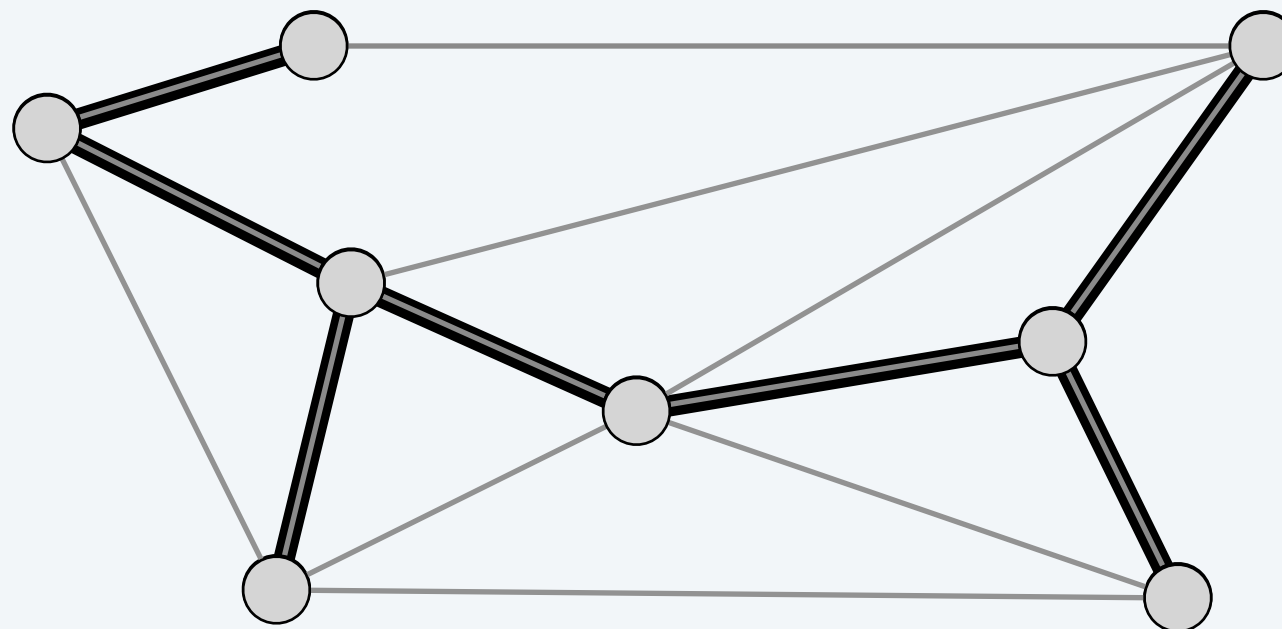
Chứng minh. [xem hình vẽ]



Cây bao trùm

Định nghĩa. Cho $H = (V, T)$ là một đồ thị con của một đồ thị vô hướng $G = (V, E)$. H được gọi là **cây bao trùm (spanning tree)** của G nếu H phi chu trình (acyclic) và liên thông (connected).

Lưu ý. $H = (V, T)$ nghĩa là H chứa mọi đỉnh của G .



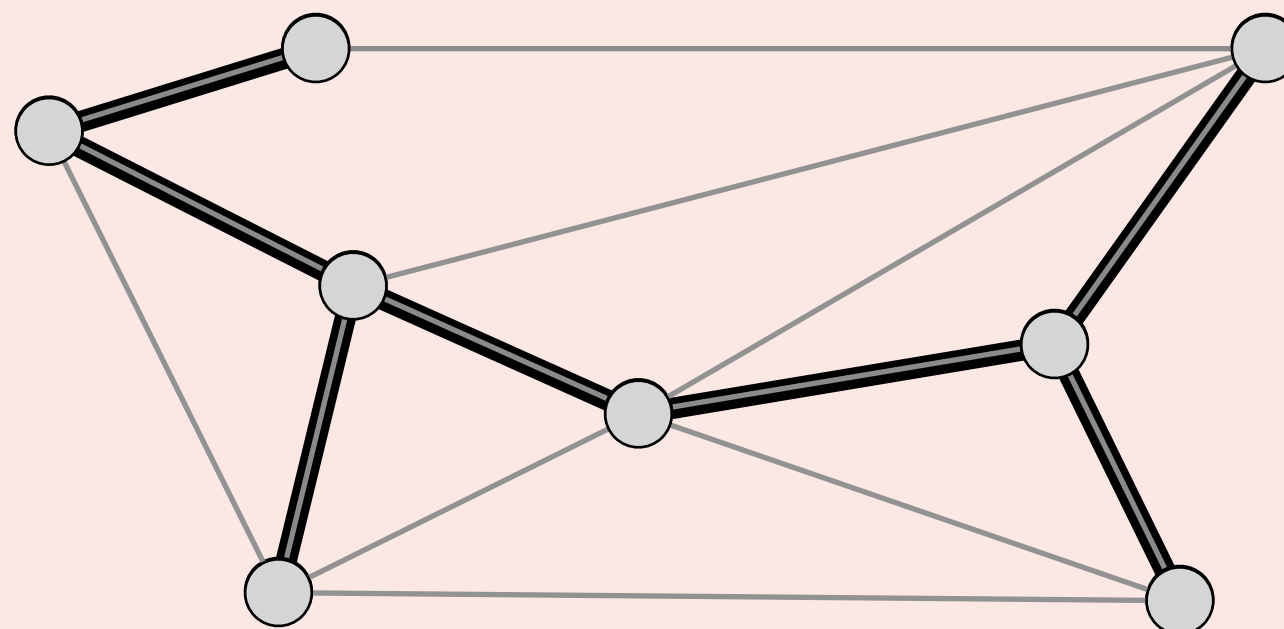
đồ thị $G = (V, E)$

cây bao trùm $H = (V, T)$



Which of the following properties are true for all spanning trees H ?

- A.** Contains exactly $|V| - 1$ edges.
- B.** The removal of any edge disconnects it.
- C.** The addition of any edge creates a cycle.
- D.** All of the above.



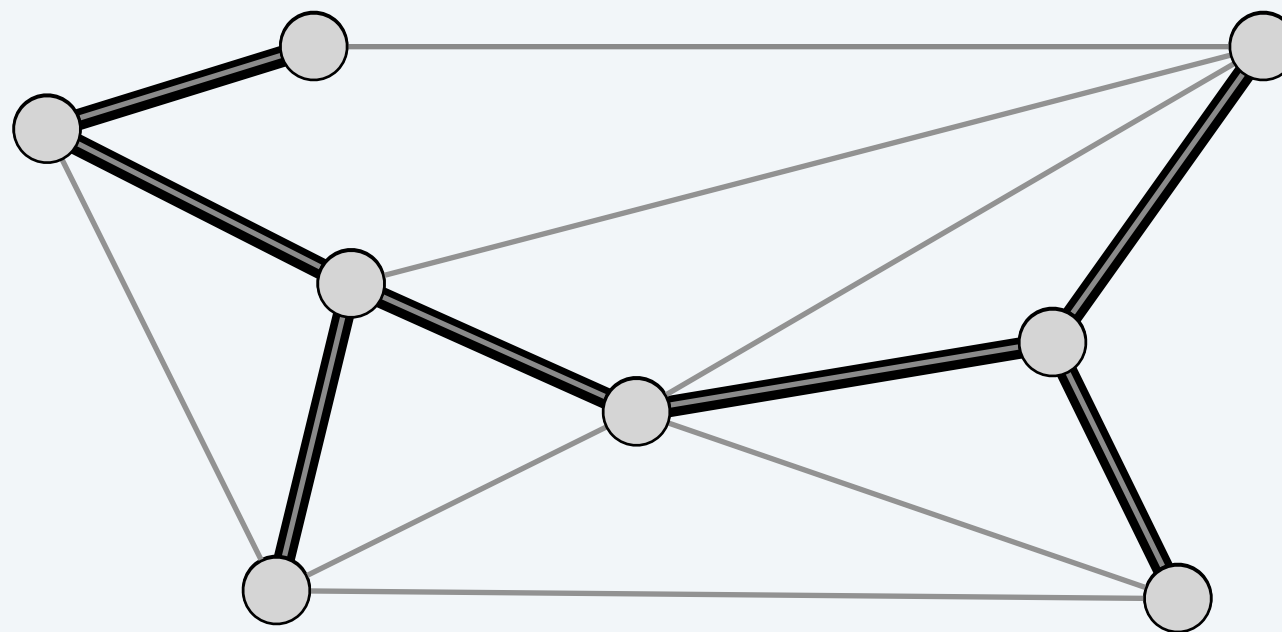
graph $G = (V, E)$

spanning tree $H = (V, T)$

Tính chất của cây bao trùm

Tính chất. Cho $H = (V, T)$ là một đồ thị con của đồ thị vô hướng $G = (V, E)$. Các khẳng định dưới đây là tương đương

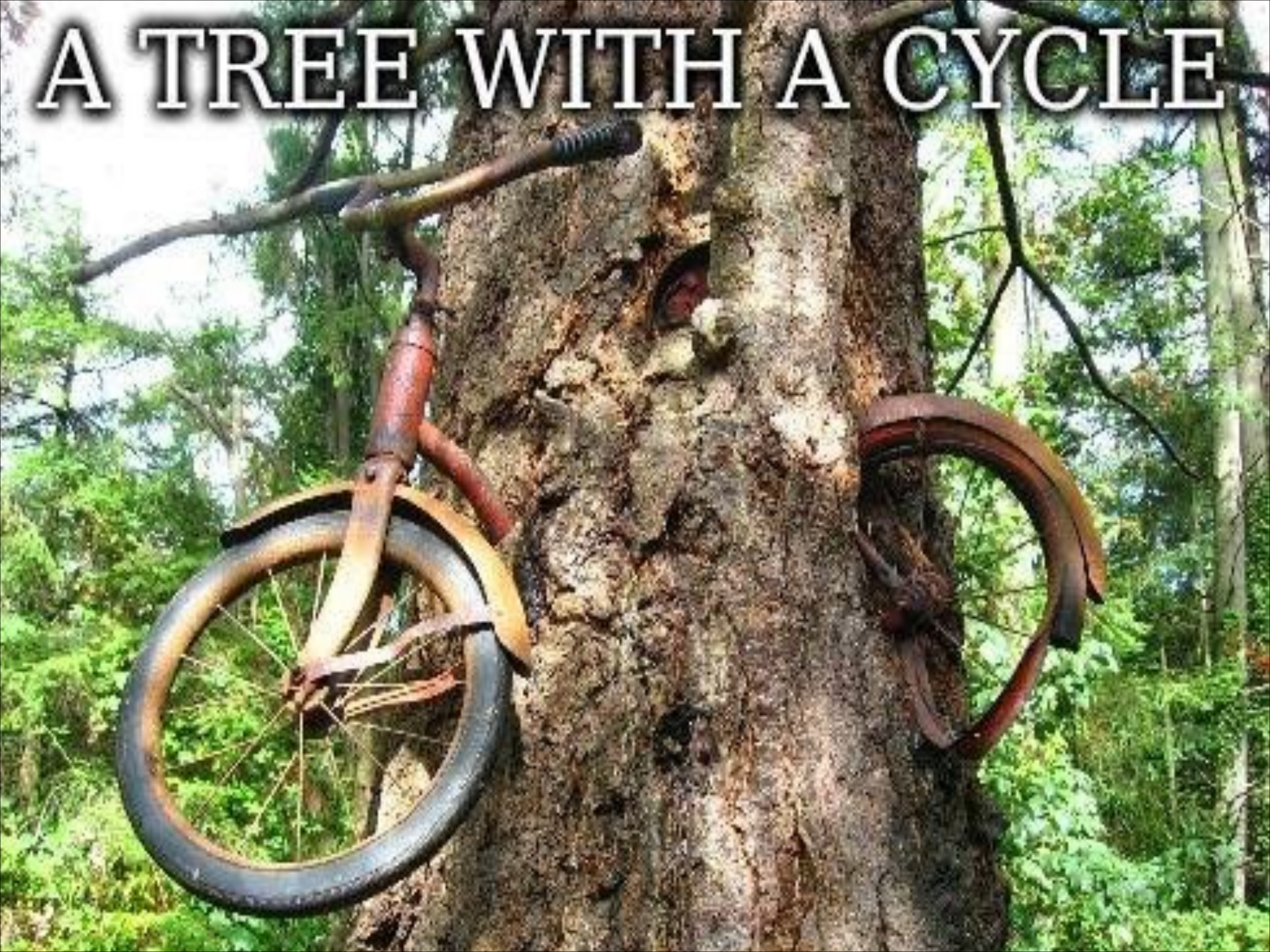
- H là **cây bao trùm** của G .
- H là phi chu trình và liên thông.
- H là liên thông và có $|V| - 1$ cạnh.
- H là phi chu trình và có $|V| - 1$ cạnh.
- H là liên thông tối thiểu (minimally connected), tức là cứ xóa bất kì cạnh nào sẽ khiến H trở thành không liên thông.
- H là phi chu trình tối đa (maximally acyclic), tức là thêm cạnh bất kì sẽ tạo ra chu trình.



đồ thị $G = (V, E)$

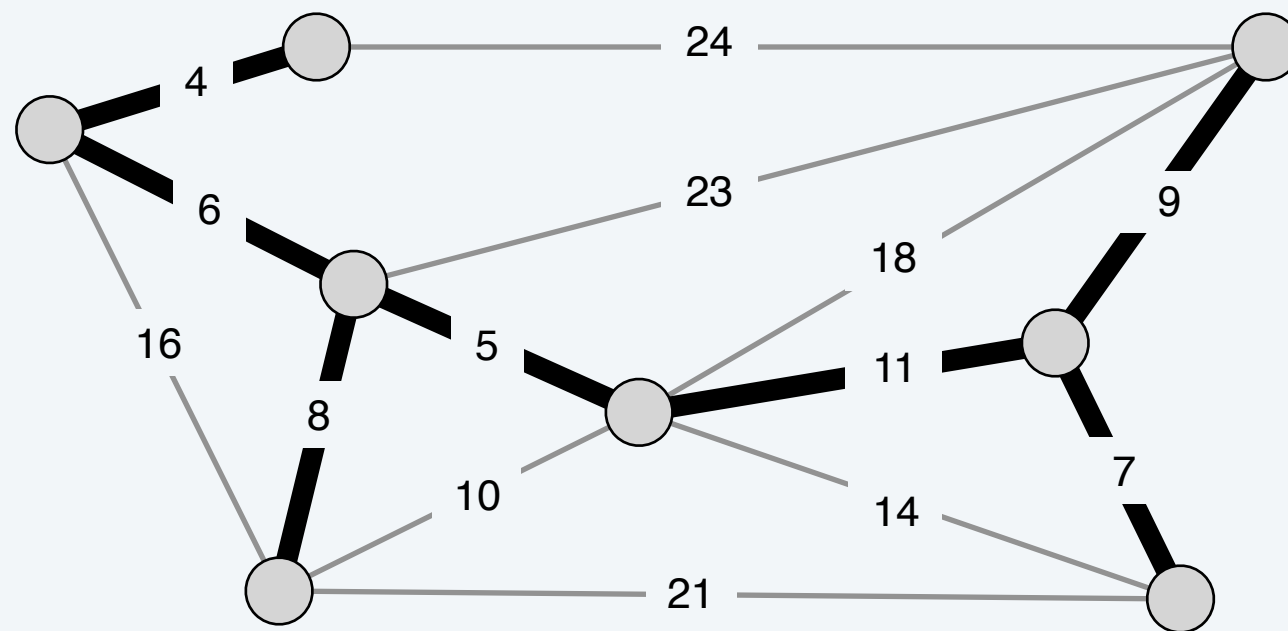
cây bao trùm $H = (V, T)$

A TREE WITH A CYCLE



Cây bao trùm nhỏ nhất (MST)

Định nghĩa. Cho một đồ thị vô hướng liên thông $G = (V, E)$ với trọng số cạnh c_e , **cây bao trùm nhỏ nhất (minimum spanning tree)** (V, T) là một cây bao trùm của G sao cho tổng trọng số cạnh trong T là nhỏ nhất.



$$\text{Trọng số MST} = 50 = 4 + 6 + 8 + 5 + 11 + 9 + 7$$

Định lý Cayley. Đồ thị đầy đủ trên n nodes has n^{n-2} spanning trees.



không thể giải bằng cách vét cạn

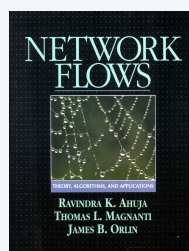


Suppose that you change the cost of every edge in G as follows.
For which is every MST in G an MST in G' (and vice versa)?
Assume $c(e) > 0$ for each e .

- A. $c'(e) = c(e) + 17$.
- B. $c'(e) = 17 \times c(e)$.
- C. $c'(e) = \log_{17} c(e)$.
- D. All of the above.

MST is fundamental problem with diverse applications.

- Dithering.
- Cluster analysis.
- Max bottleneck paths.
- Real-time face verification.
- LDPC codes for error correction.
- Image registration with Renyi entropy.
- Find road networks in satellite and aerial imagery.
- Model locality of particle interactions in turbulent fluid flows.
- Reducing data storage in sequencing amino acids in a protein.
- Autoconfig protocol for Ethernet bridging to avoid cycles in a network.
- Approximation algorithms for NP-hard problems (e.g., TSP, Steiner tree).
- Network design (communication, electrical, hydraulic, computer, road).

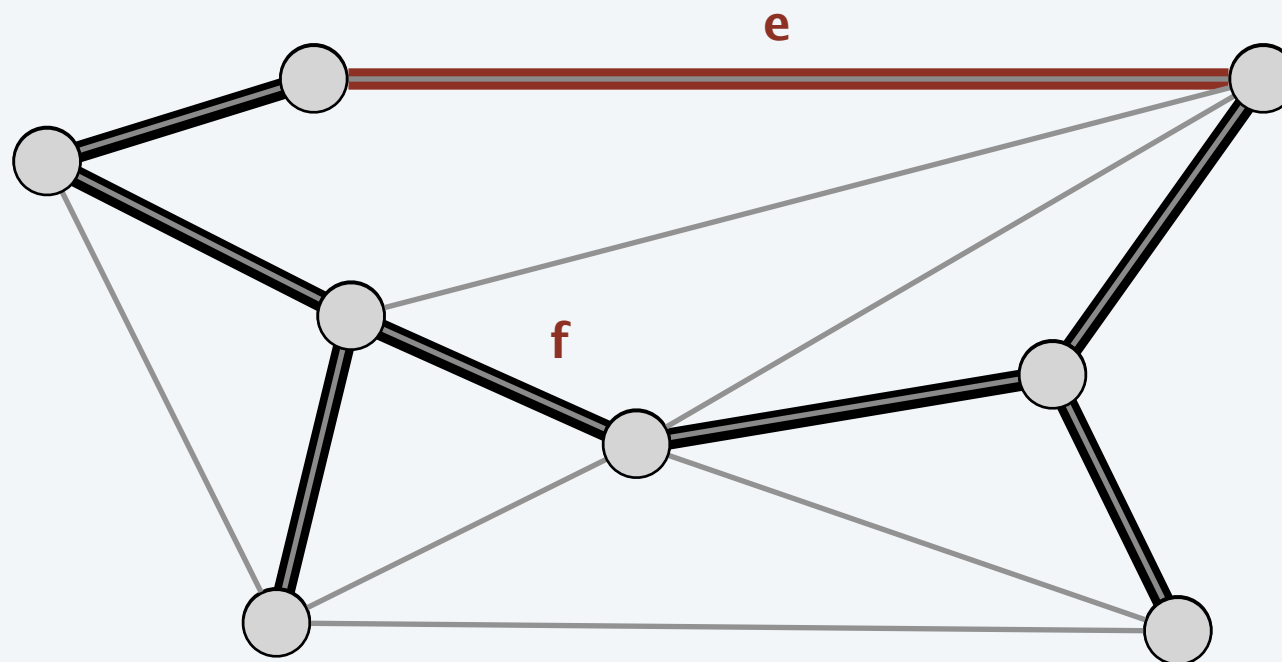


Network Flows: Theory, Algorithms, and Applications,
by Ahuja, Magnanti, and Orlin, Prentice Hall, 1993.

Chu trình cơ bản (fundamental cycle)

Chu trình cơ bản. Cho $H = (V, T)$ là một cây bao trùm của $G = (V, E)$.

- Với mọi cạnh $e \in E$ không thuộc cây, $T \cup \{e\}$ chứa một chu trình duy nhất, gọi là C .
- Với mỗi cạnh $f \in C$, $T \cup \{e\} - \{f\}$ là một cây bao trùm.



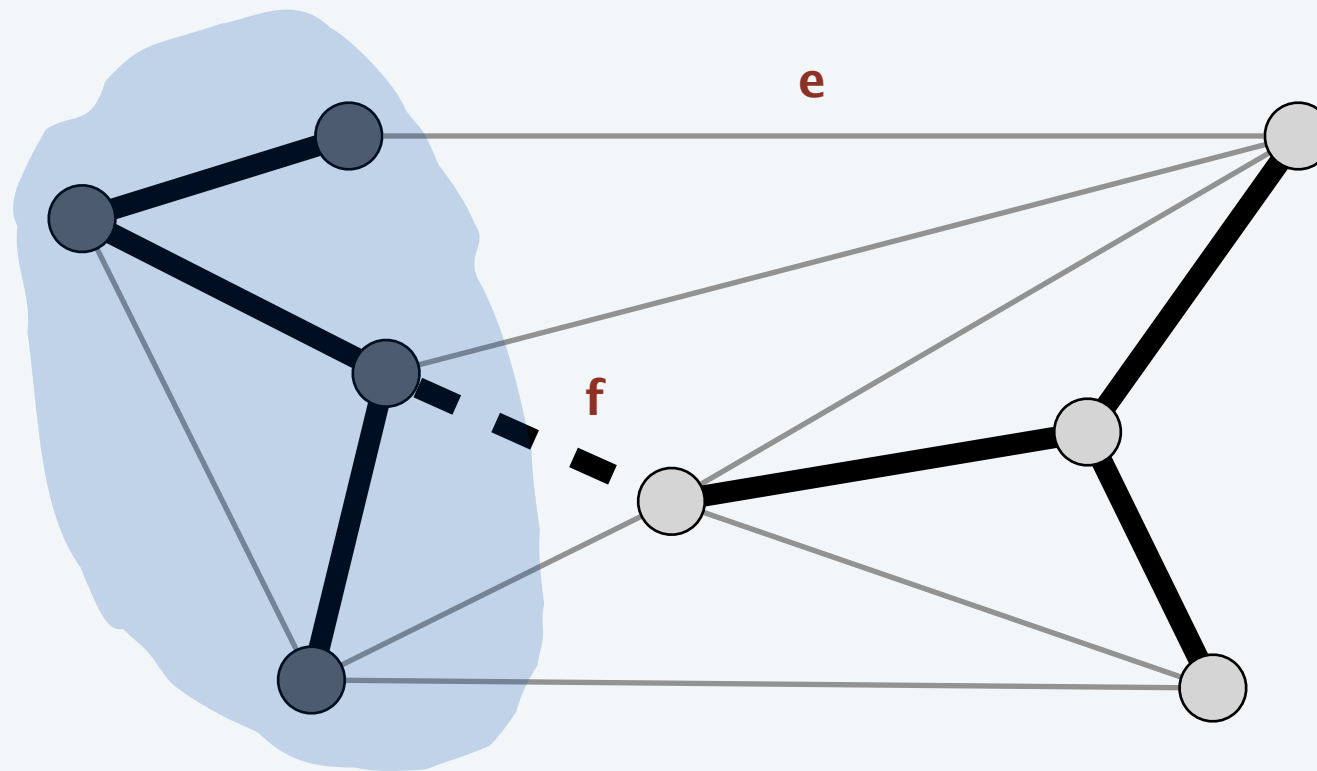
đồ thị $G = (V, E)$

cây bao trùm $H = (V, T)$

Tập cạnh cắt cơ bản (fundamental cutset)

Tập cạnh cắt cơ bản. Cho $H = (V, T)$ là một cây bao trùm của $G = (V, E)$.

- Với mỗi cạnh trên cây $f \in T$, $T - \{f\}$ có hai thành phần liên thông. Gọi D là tập cạnh cắt tương ứng.
- Với mỗi cạnh $e \in D$, $T - \{f\} \cup \{e\}$ là một cây bao trùm.



đồ thị $G = (V, E)$

cây bao trùm $H = (V, T)$

Nhận xét. Nếu $c_e < c_f$ thì (V, T) không phải là cây bao trùm nhỏ nhất.

Thuật toán tham lam (greedy algorithm)

Quy tắc đỏ.



- Cho C là một chu trình không có cạnh màu đỏ.
- Chọn một cạnh chưa tô màu của C có trọng số lớn nhất và tô màu đỏ.

Quy tắc xanh.

- Cho D là một tập cạnh cắt không có cạnh nào tô màu xanh.
- Chọn một cạnh trong D chưa tô màu có trọng số nhỏ nhất và tô màu xanh.

Thuật toán tham lam.

- Áp dụng quy tắc đỏ và xanh (một cách không tất định (nondeterministically)!) cho đến khi tất cả các cạnh đều được tô màu. Các cạnh tô màu xanh tạo thành một cây bao trùm nhỏ nhất.
- Chú ý: Ta có thể dừng chừng nào có $n - 1$ cạnh được tô màu xanh.

Thuật toán tham lam: tính đúng đắn

Mệnh đề (tính chất bất biến về màu). Tồn tại một cây bao trùm (V, T^*) chứa mọi cạnh xanh và không chứa cạnh đỏ nào.

Chứng minh. [quy nạp theo số bước của thuật toán]

Bước cơ sở. Không có cạnh nào được tô màu \Rightarrow thỏa mãn tính chất bất biến.

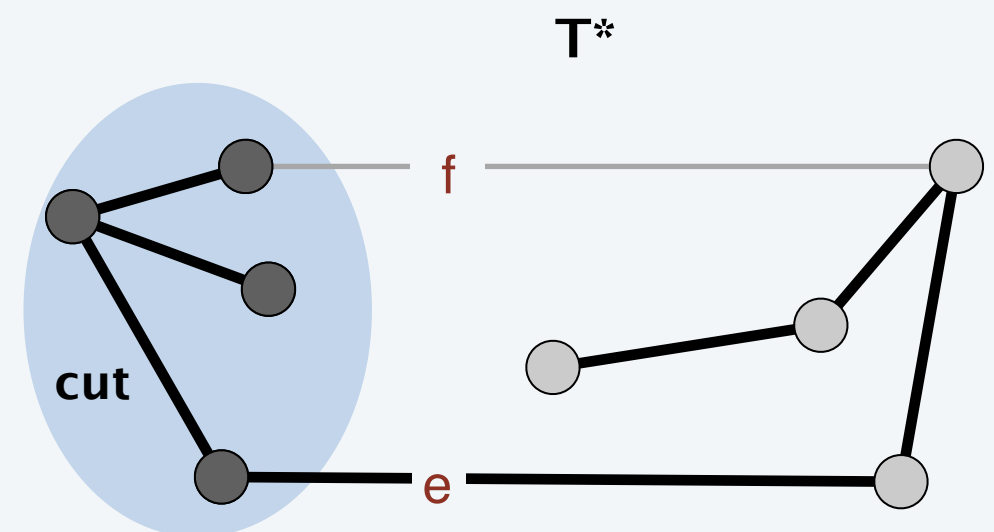
Thuật toán tham lam: tính đúng đắn

Mệnh đề (tính chất bất biến về màu). Tồn tại một cây bao trùm (V, T^*) chứa mọi cạnh xanh và không chứa cạnh đỏ nào.

Chứng minh. [quy nạp theo số bước của thuật toán]

Bước quy nạp (quy tắc màu xanh). Giả sử tính chất bất biến được thỏa mãn trước khi áp dụng quy tắc **xanh**.

- Gọi D là tập cạnh cắt được chọn và f là cạnh sẽ được tô màu xanh.
- Nếu $f \in T^*$, thì T^* vẫn thỏa mãn tính chất bất biến.
- Ngược lại, xét chu trình cơ bản C được tạo ra khi thêm f vào T^* .
- Xét $e \in C$ là một cạnh khác thuộc D (tồn tại do 1 chu trình và 1 tập cạnh cắt chung một số chẵn cạnh).
- e không được tô màu và $c_e \geq c_f$ vì
 - $e \in T^* \Rightarrow e$ không có màu đỏ
 - quy tắc màu xanh $\Rightarrow e$ không có màu xanh và $c_e \geq c_f$
- Do đó $T^* \cup \{f\} - \{e\}$ thỏa mãn t/c bất biến và là một cây bao trùm nhỏ nhất.



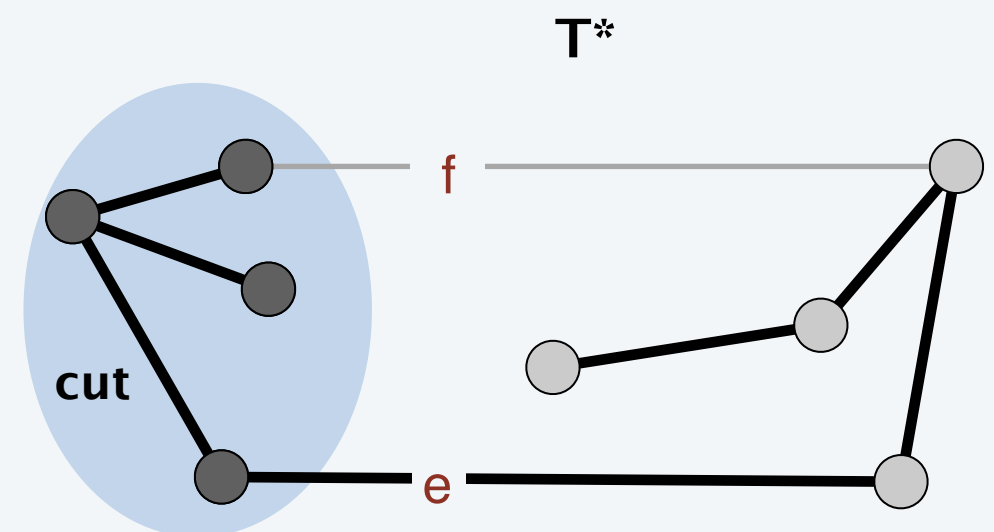
Thuật toán tham lam: tính đúng đắn

Mệnh đề (tính chất bất biến về màu). Tồn tại một cây bao trùm nhỏ nhất (V, T^*) chứa mọi cạnh xanh và không chứa cạnh đỏ nào.

Chứng minh. [quy nạp theo số bước của thuật toán]

Bước quy nạp (quy tắc đỏ). Giả sử tính chất bất biến được thỏa mãn trước khi áp dụng quy tắc **đỏ**.

- Gọi C là chu trình được chọn và e là cạnh sẽ được tô màu đỏ.
- Nếu $e \notin T^*$, thì T^* vẫn thỏa mãn tính chất bất biến.
- Ngược lại, xét tập cạnh cắt cơ bản D nhận được từ việc xóa e khỏi T^* .
- Xét $f \in D$ là một cạnh khác trong C (tồn tại do 1 chu trình và 1 tập cạnh cắt chung một số chẵn cạnh).
- f không được tô màu và $c_e \geq c_f$ vì
 - $f \notin T^* \Rightarrow f$ không có màu xanh
 - quy tắc đỏ $\Rightarrow f$ không có màu đỏ và $c_e \geq c_f$.
- Do đó, $T^* \cup \{f\} - \{e\}$ thỏa mãn t/c bất biến và là một cây bao trùm nhỏ nhất. ■

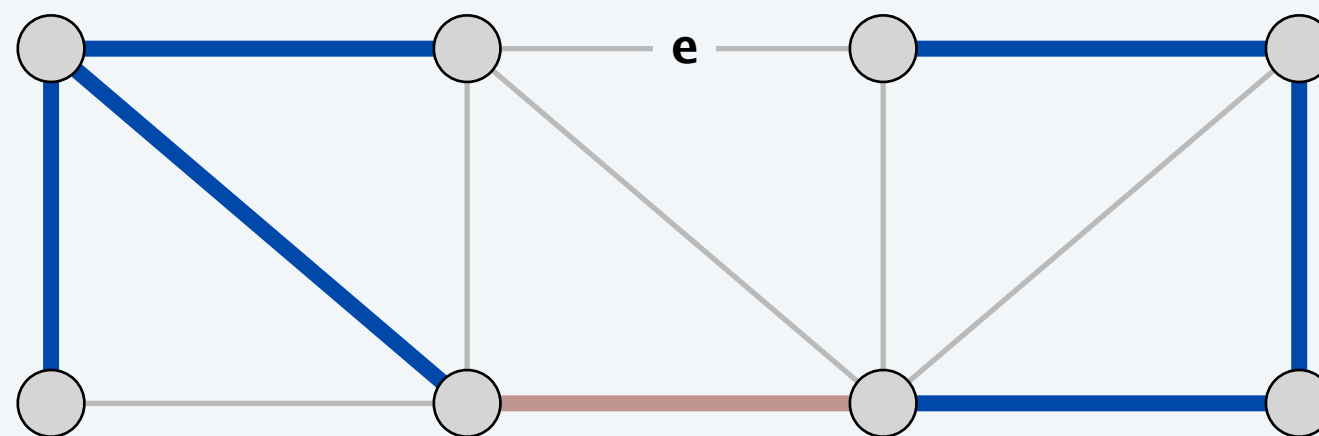


Thuật toán tham lam: tính đúng đắn

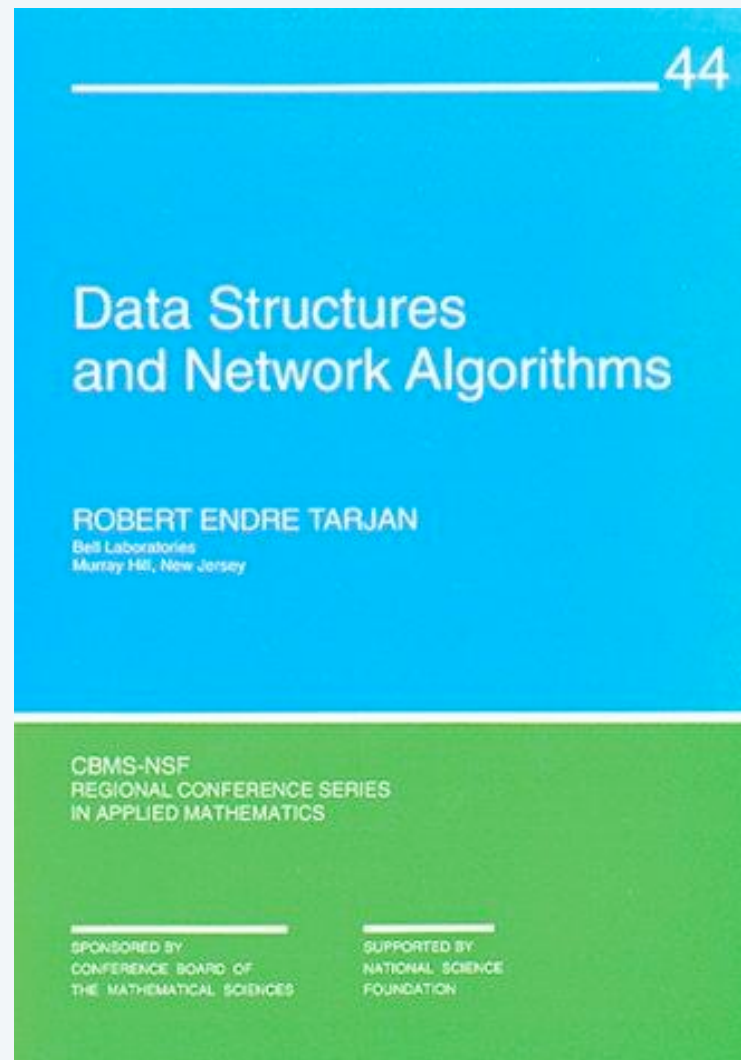
Định lý. Thuật toán tham lam dừng. Tập cạnh xanh tạo ra 1 cây bao trùm nhỏ nhất.

Chứng minh. Chứng minh rằng có thể áp dụng quy tắc xanh hoặc đỏ (hoặc cả hai).

- Giả sử cạnh e không được tô màu.
- Các cạnh màu xanh tạo ra một rừng.
- Trường hợp 1: cả hai đỉnh của e thuộc cùng một cây màu xanh.
⇒ áp dụng quy tắc đỏ cho chu trình được tạo ra khi thêm e vào cây màu xanh.
- Trường hợp 2: đỉnh của e thuộc hai cây màu xanh khác nhau.
⇒ áp dụng quy tắc màu xanh cho tập cạnh cắt được tạo ra bởi một trong hai cây màu xanh trên. Tập cạnh cắt này có ít nhất một cạnh không tô màu (cạnh e) ■



Trường hợp 2



SECTION 6.2

CÂY BAO TRÙM

- ▶ *cây bao trùm*
- ▶ *thuật toán Prim, Kruskal, Boruvka*
- ▶ *phân cụm liên kết đơn*

Thuật toán Prim

Khởi tạo $S =$ đỉnh bất kì, $T = \emptyset$.

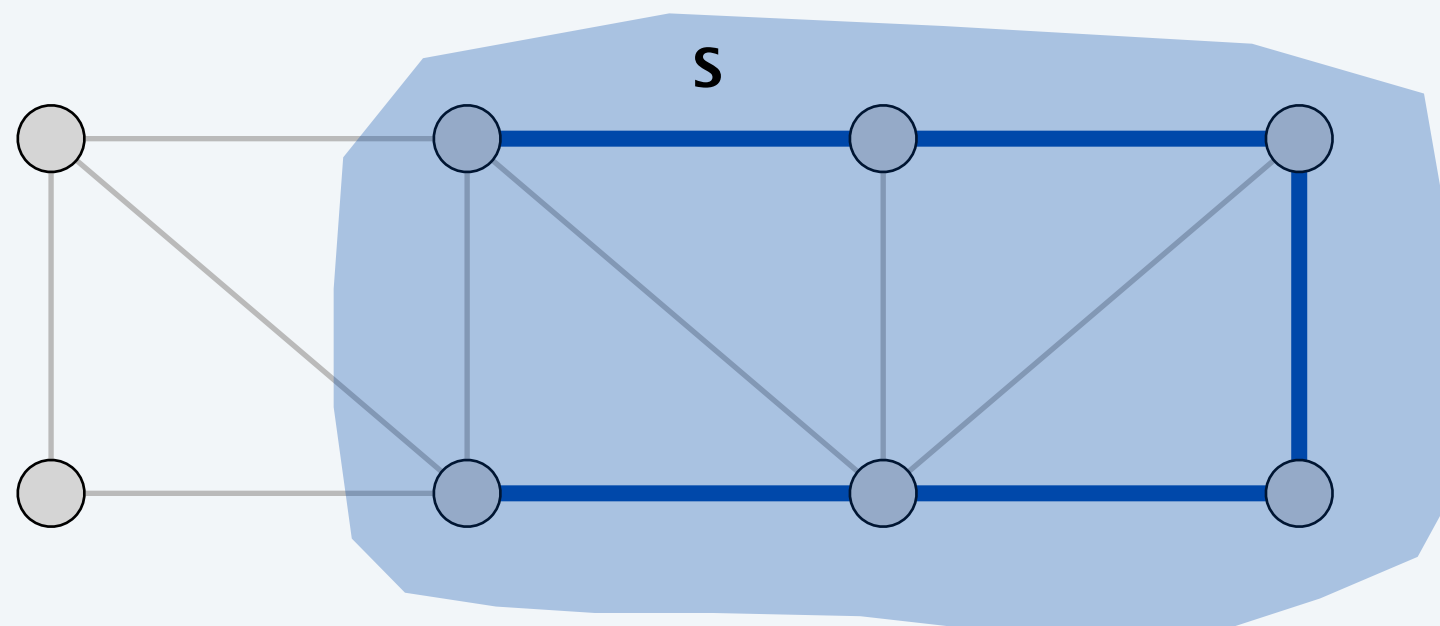
Lặp $n - 1$ lần:

- Thêm vào T cạnh có trọng số nhỏ nhất với một đỉnh thuộc S .
- Thêm đỉnh còn lại vào S .



Định lý. Thuật toán Prim tìm được một cây bao trùm nhỏ nhất.

Chứng minh. Thuật toán là một trường hợp đặc biệt của thuật toán tham lam trong đó ta liên tục áp dụng quy tắc màu xanh cho S . ■



Thuật toán Prim: thực thi (implementation)

Định lí. Thuật toán Prim (có thể được thực thi để) có thời gian $O(m \log n)$.

Chứng minh.

PRIM (V, E, c)

$S \leftarrow \emptyset, T \leftarrow \emptyset.$

$s \leftarrow$ any node in V .

FOREACH $v \neq s : \pi[v] \leftarrow \infty, pred[v] \leftarrow null; \pi[s] \leftarrow 0.$

Create an empty priority queue pq .

FOREACH $v \in V : \text{INSERT}(pq, v, \pi[v]).$

WHILE (**IS-NOT-EMPTY**(pq))

$u \leftarrow \text{DEL-MIN}(pq).$


$S \leftarrow S \cup \{u\}, T \leftarrow T \cup \{pred[u]\}.$

FOREACH edge $e = (u, v) \in E$ with $v \notin S :$

IF ($c_e < \pi[v]$)

DECREASE-KEY(pq, v, c_e).

$\pi[v] \leftarrow c_e; pred[v] \leftarrow e.$



$\pi[v]$ = weight of cheapest known edge between v and S

Thuật toán Kruskal

Xét các cạnh theo thứ tự trọng số tăng dần:

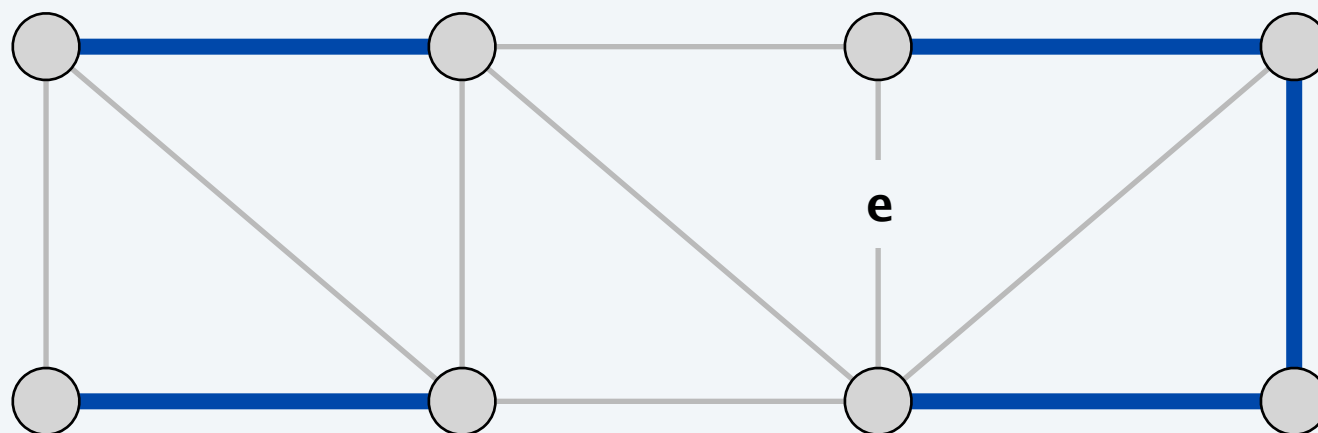
- Thêm cạnh vào cây trừ khi nó tạo ra chu trình.



Định lí. Thuật toán Kruskal tìm ra cây bao trùm nhỏ nhất.

Chứng minh. Là trường hợp đặc biệt của thuật toán tham lam.

- Trường hợp 1: cả hai đỉnh của e thuộc cùng 1 cây màu xanh.
 \Rightarrow tô cạnh e màu đỏ bằng cách áp dụng quy tắc đó cho chu trình duy nhất.
all other edges in cycle are blue
- Trường hợp 2: hai đỉnh của e thuộc các cây màu xanh khác nhau.
 \Rightarrow tô cạnh e màu xanh bằng cách áp dụng quy tắc màu xanh cho tập cạnh cắt được tạo bởi một trong hai cây. ■
no edge in cutset has smaller weight (since Kruskal chose it first)



Thuật toán Kruskal: thực thi

Định lí. Thuật toán Kruskal có thể được thực thi để chạy trong thời gian $O(m \log m)$.

- Sắp xếp cạnh theo trọng số.
- Use **union–find** data structure to dynamically maintain connected components.

KRUSKAL (V, E, c)

sort m edges by weight and renumber so that $c(e_1) \leq c(e_2) \leq \dots \leq c(e_m)$.

$T \leftarrow \emptyset$.

foreach $v \in V$: **MAKE-SET**(v).

for $i = 1$ **to** m

$(u, v) \leftarrow e_i$.

if (**FIND-SET**(u) \neq **FIND-SET**(v)) \leftarrow are u and v in same component?

$T \leftarrow T \cup \{e_i\}$.

union(u, v). \leftarrow make u and v in same component

return T .

Reverse-delete algorithm

Start with all edges in T and consider them in descending order of weight:

- Delete edge from T unless it would disconnect T .

Theorem. The reverse-delete algorithm computes an MST.

Pf. Special case of greedy algorithm.

- Case 1. [deleting edge e does not disconnect T]
⇒ apply red rule to cycle C formed by adding e to another path
in T between its two endpoints

no edge in C has larger weight
(it would have already been considered and deleted)

- Case 2. [deleting edge e disconnects T]
⇒ apply blue rule to cutset D induced by either component ■

e is the only remaining edge in the cutset
(all other edges in D must have been colored red / deleted)

Fact. [Thorup 2000] Can be implemented to run in $O(m \log n (\log \log n)^3)$ time.

Review: the greedy MST algorithm

Red rule.

- Let C be a cycle with no red edges.
- Select an uncolored edge of C of max weight and color it red.

Blue rule.

- Let D be a cutset with no blue edges.
- Select an uncolored edge in D of min weight and color it blue.

Greedy algorithm.

- Apply the red and blue rules (nondeterministically!) until all edges are colored. The blue edges form an MST.
- Note: can stop once $n - 1$ edges colored blue.

Theorem. The greedy algorithm is correct.

Special cases. Prim, Kruskal, reverse-delete, ...

Borůvka's algorithm

Repeat until only one tree.

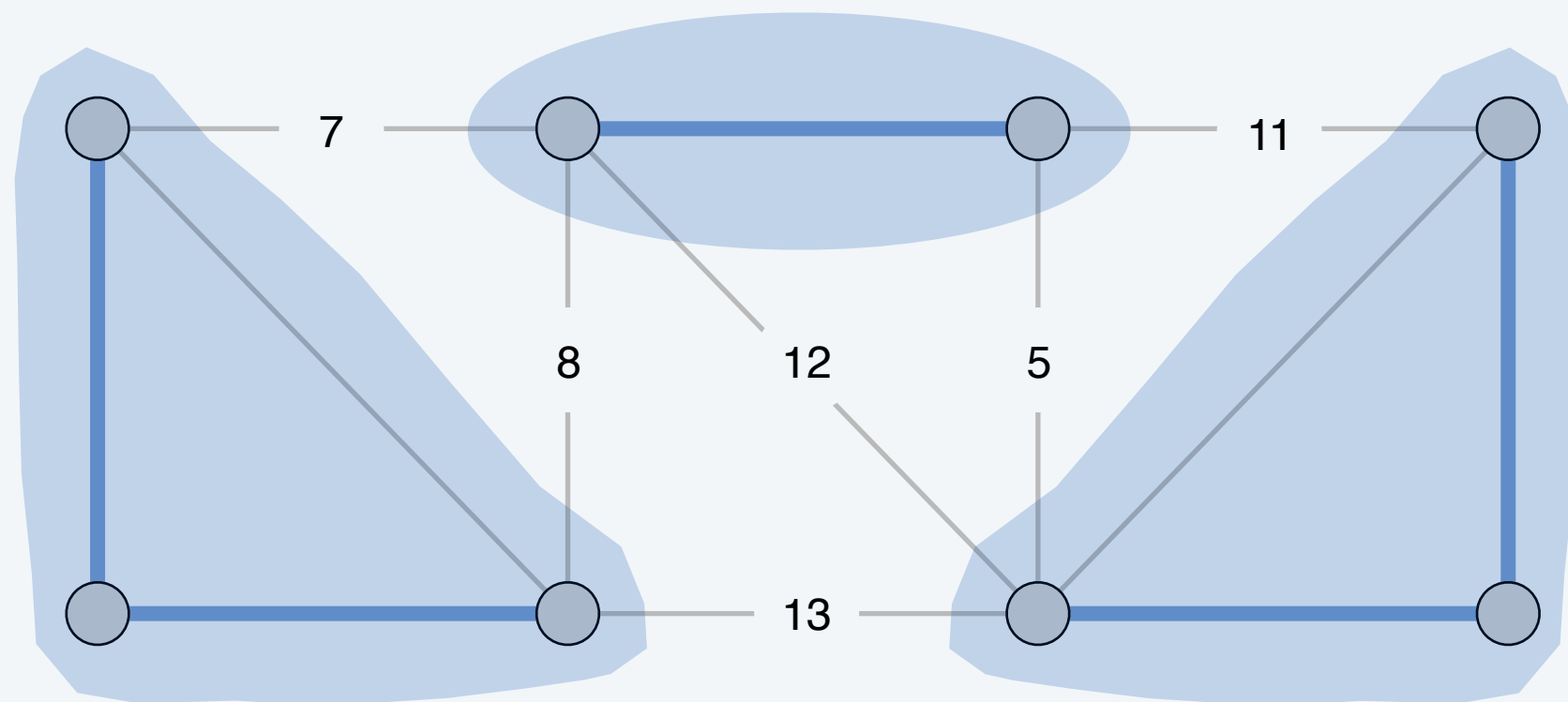
- Apply blue rule to cutset corresponding to **each** blue tree.
- Color **all** selected edges blue.



Theorem. Borůvka's algorithm computes the MST.

← assume edge costs are distinct

Pf. Special case of greedy algorithm (repeatedly apply blue rule). ■

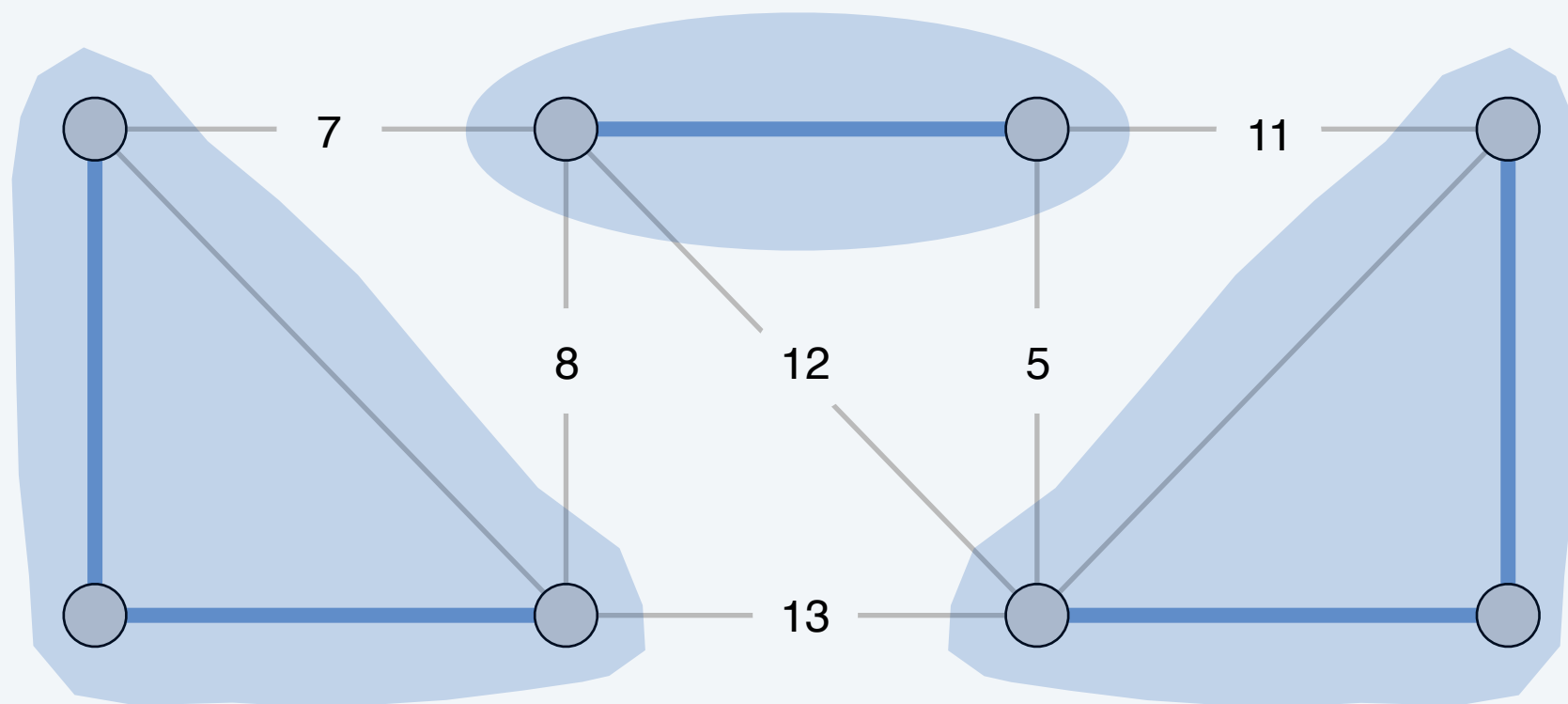


Borůvka's algorithm: implementation

Theorem. Borůvka's algorithm can be implemented to run in $O(m \log n)$ time.

Pf.

- To implement a phase in $O(m)$ time:
 - compute connected components of blue edges
 - for each edge $(u, v) \in E$, check if u and v are in different components; if so, update each component's best edge in cutset
- $\leq \log_2 n$ phases since each phase (at least) halves total # components. ■

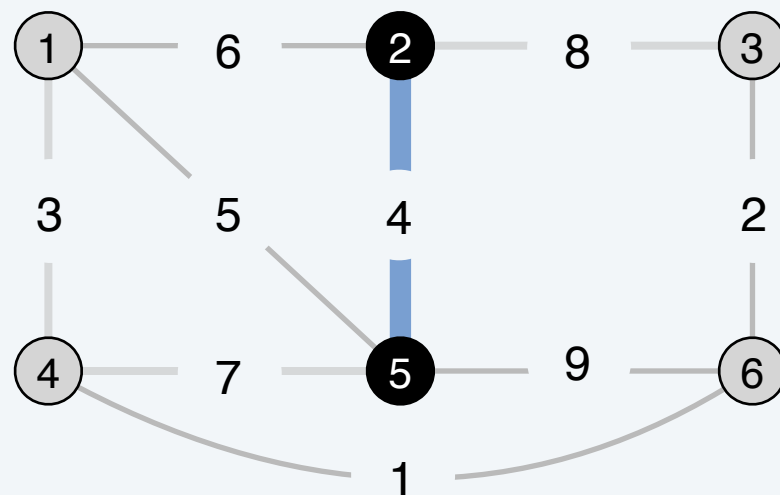


Borůvka's algorithm: implementation

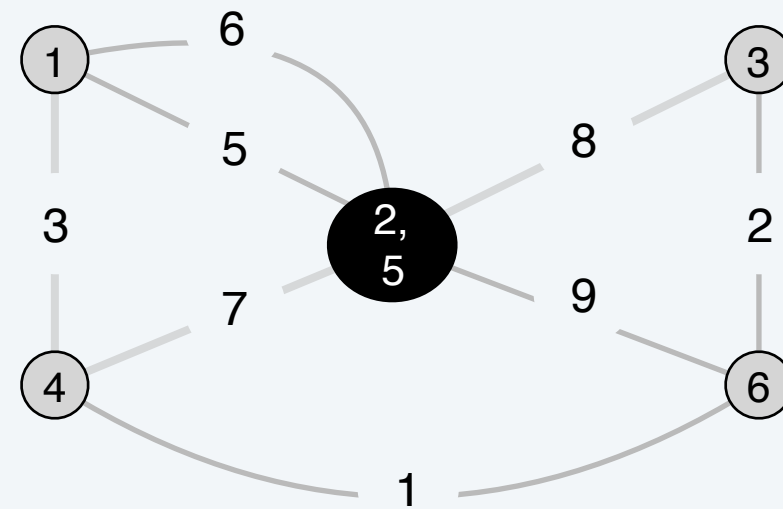
Contraction version.

- After each phase, **contract** each blue tree to a single supernode.
- Delete self-loops and parallel edges (keeping only cheapest one).
- Borůvka phase becomes: take cheapest edge incident to each node.

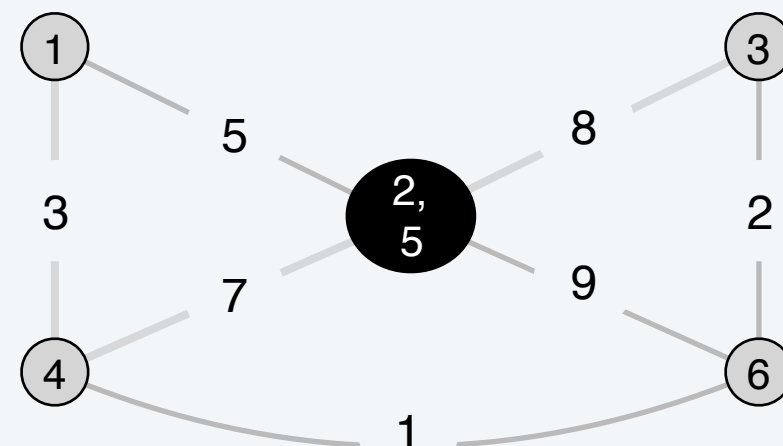
graph G



contract edge 2-5



delete self-loops and parallel edges



Q. How to contract a set of edges?

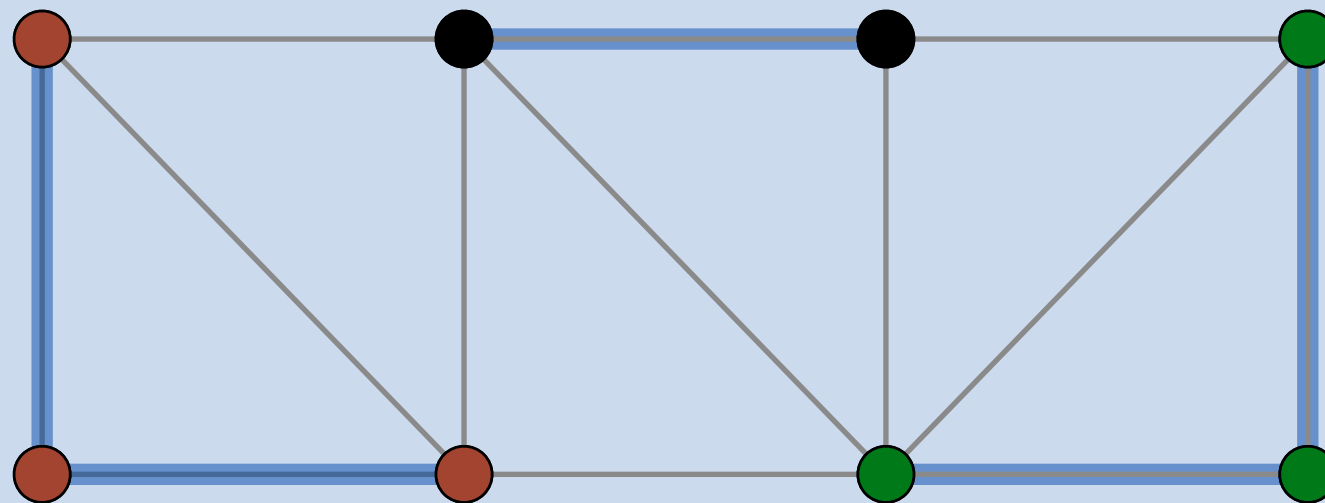
CONTRACT A SET OF EDGES



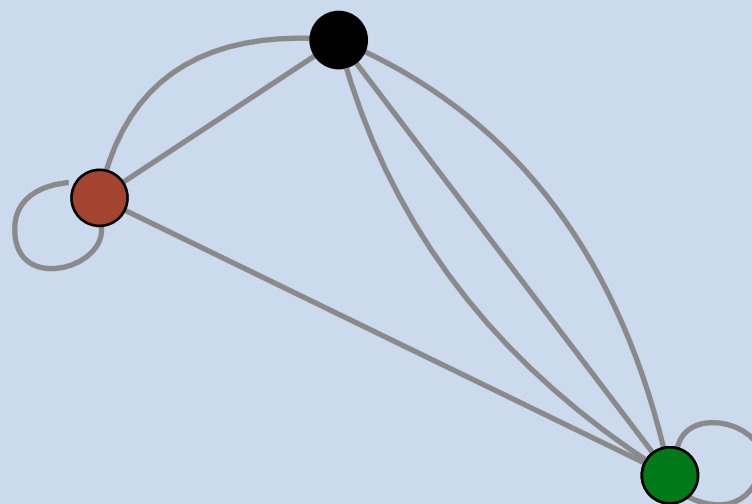
Problem. Given a graph $G = (V, E)$ and a set of edges F , contract all edges in F , removing any self-loops or parallel edges.

Goal. $O(m + n)$ time.

graph G



contracted graph G'



CONTRACT A SET OF EDGES



Problem. Given a graph $G = (V, E)$ and a set of edges F , contract all edges in F , removing any self-loops or parallel edges.

Solution.

- Compute the n' connected components in (V, F) .
- Suppose $id[u] = i$ means node u is in connected component i .
- The contracted graph G' has n' nodes.
- For each edge $u-v \in E$, add an edge $i-j$ to G' , where $i = id[u]$ and $j = id[v]$.

Removing self loops. Easy.

Removing parallel edges.

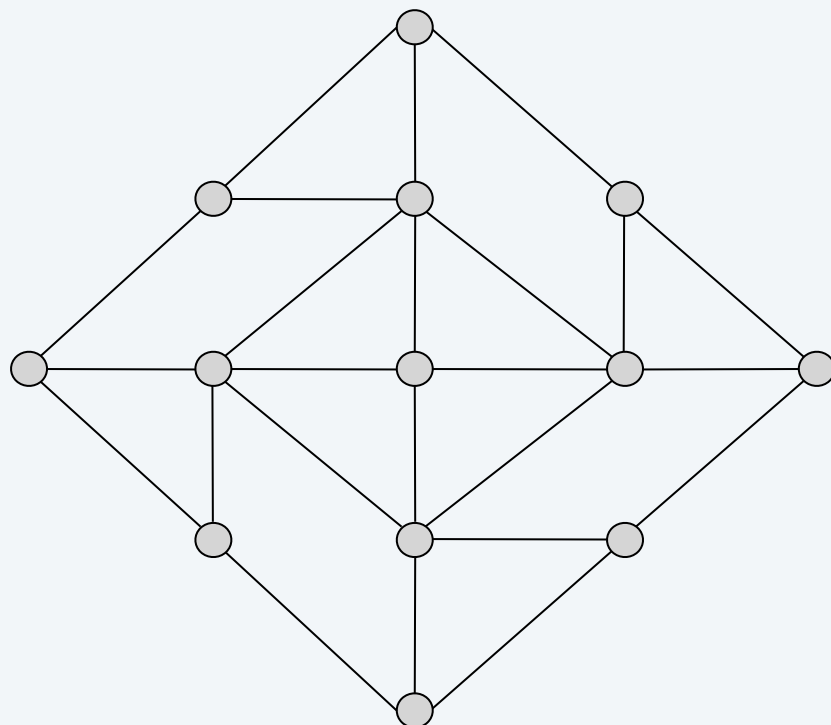
- Create a list of edges $i-j$ with the convention that $i < j$.
- Sort the edges lexicographically via LSD radix sort.
- Add the edges to the graph G' , removing parallel edges.

Borůvka's algorithm on planar graphs

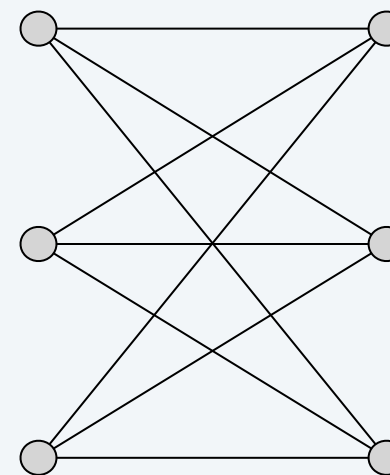
Theorem. Borůvka's algorithm (contraction version) can be implemented to run in $O(n)$ time on planar graphs.

Pf.

- Each Borůvka phase takes $O(n)$ time:
 - Fact 1: $m \leq 3n$ for simple planar graphs.
 - Fact 2: planar graphs remains planar after edge contractions/deletions.
- Number of nodes (at least) halves in each phase.
- Thus, overall running time $\leq cn + cn/2 + cn/4 + cn/8 + \dots = O(n)$. ■



planar



$K_{3,3}$ not planar

A hybrid algorithm

Borůvka–Prim algorithm.

- Run Borůvka (contraction version) for $\log_2 \log_2 n$ phases.
- Run Prim on resulting, contracted graph.


Theorem. Borůvka–Prim computes an MST.

Pf. Special case of the greedy algorithm.

Theorem. Borůvka–Prim can be implemented to run in $O(m \log \log n)$ time.

Pf.

- The $\log_2 \log_2 n$ phases of Borůvka's algorithm take $O(m \log \log n)$ time; resulting graph has $\leq n / \log_2 n$ nodes and $\leq m$ edges.
- Prim's algorithm (using Fibonacci heaps) takes $O(m + n)$ time on a graph with $n / \log_2 n$ nodes and m edges. ■


$$O\left(m + \frac{n}{\log n} \log\left(\frac{n}{\log n}\right)\right)$$

Does a linear-time compare-based MST algorithm exist?

year	worst case	discovered by
1975	$O(m \log \log n)$	Yao
1976	$O(m \log \log n)$	Cheriton–Tarjan
1984	$O(m \log^* n), O(m + n \log n)$	Fredman–Tarjan
1986	$O(m \log (\log^* n))$	Gabow–Galil–Spencer–Tarjan
1997	$O(m \alpha(n) \log \alpha(n))$	Chazelle
2000	$O(m \alpha(n))$	Chazelle
2002	<i>asymptotically optimal</i>	Pettie–Ramachandran
20xx	$O(m)$???

iterated logarithm function

$$\lg^* n = \begin{cases} 0 & \text{if } n \leq 1 \\ 1 + \lg^*(\lg n) & \text{if } n > 1 \end{cases}$$

n	$\lg^* n$
$(-\infty, 1]$	0
$(1, 2]$	1
$(2, 4]$	2
$(4, 16]$	3
$(16, 2^{16}]$	4
$(2^{16}, 2^{65536}]$	5

deterministic compare-based MST algorithms

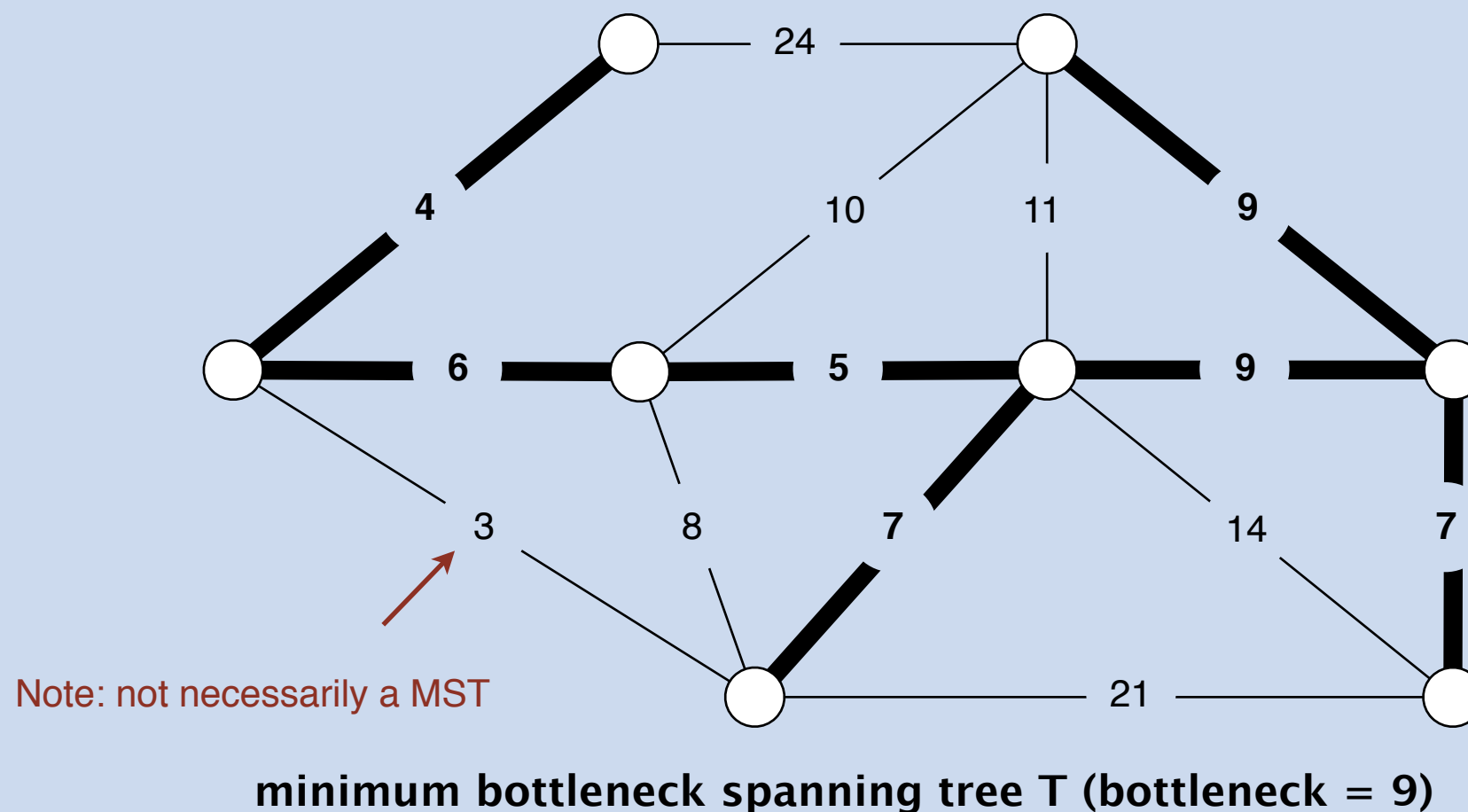
- Theorem. [Fredman–Willard 1990] $O(m)$ in word RAM model.
- Theorem. [Dixon–Rauch–Tarjan 1992] $O(m)$ MST verification algorithm.
- Theorem. [Karger–Klein–Tarjan 1995] $O(m)$ randomized MST algorithm.

MINIMUM BOTTLENECK SPANNING TREE



Problem. Given a connected edge-weighted graph G , find a spanning tree that **minimizes the maximum weight** of its edges.

Goal. $O(m \log m)$ time or better.



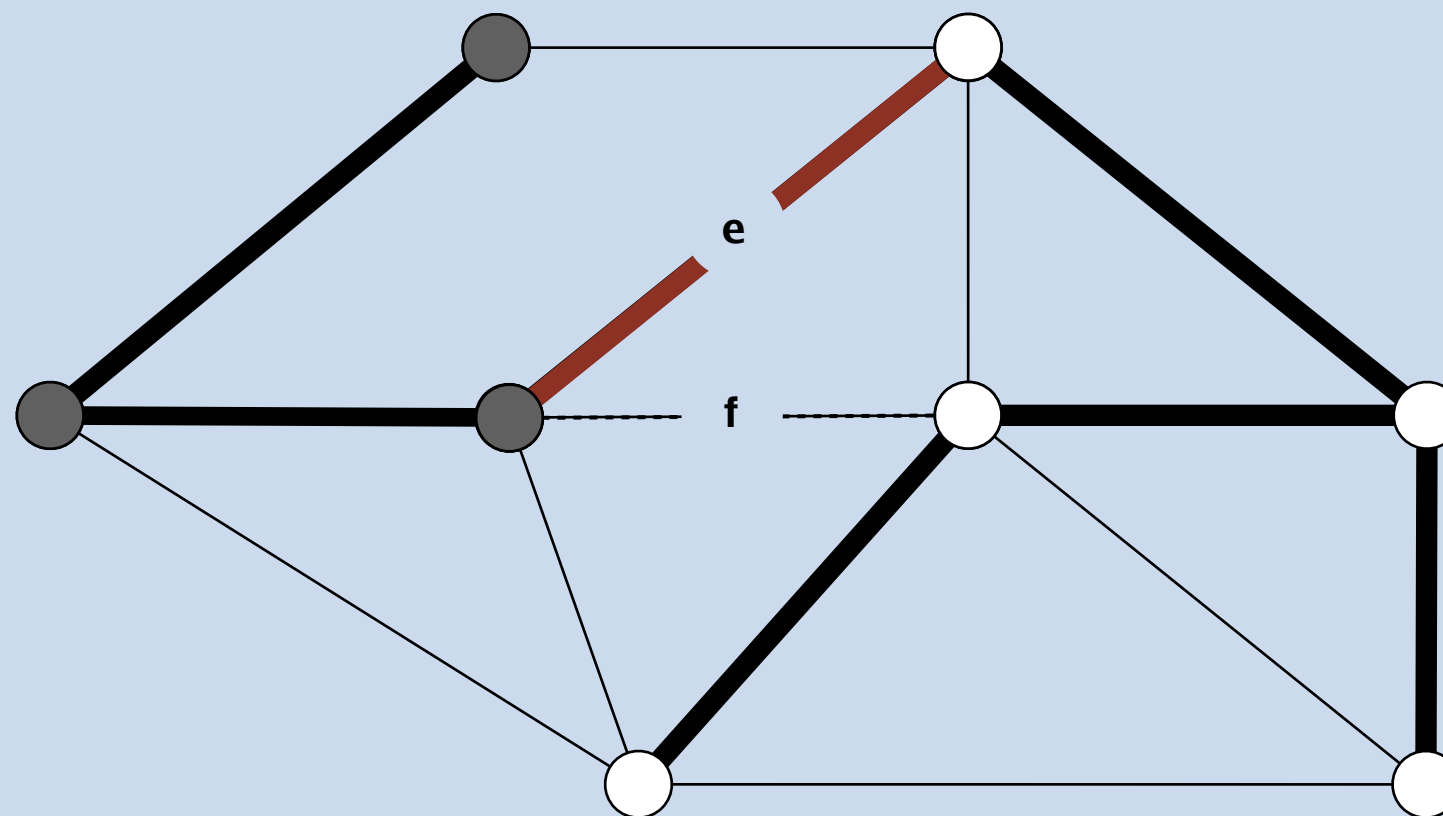
MINIMUM BOTTLENECK SPANNING TREE



Solution 1. Compute a MST T^* . It is also a MBST.

Pf. Suppose for sake of contradiction that T^* is not a MBST.

- Let $e \in T^*$ have weight strictly larger than min bottleneck weight.
- Consider cut formed by deleting e from T^* .
- MBST contains at least one edge f in cutset.
- $T^* \cup \{f\} - \{e\}$ yields better MST. ✖



minimum spanning tree T^*

MINIMUM BOTTLENECK SPANNING TREE



Solution 2.

- Sort the edges in increasing order of weight $e_1 \leq e_2 \leq \dots \leq e_m$.
- Define $E_x = \{ e \in E : c_e \leq x \}$ and $G_x = (V, E_x)$.
- Use binary search to find smallest value of x for which G_x is connected.

Note. This algorithm can be improved to $O(m)$ time using linear-time median finding and edge contractions.

Volume 7, number 1

INFORMATION PROCESSING LETTERS

January 1978

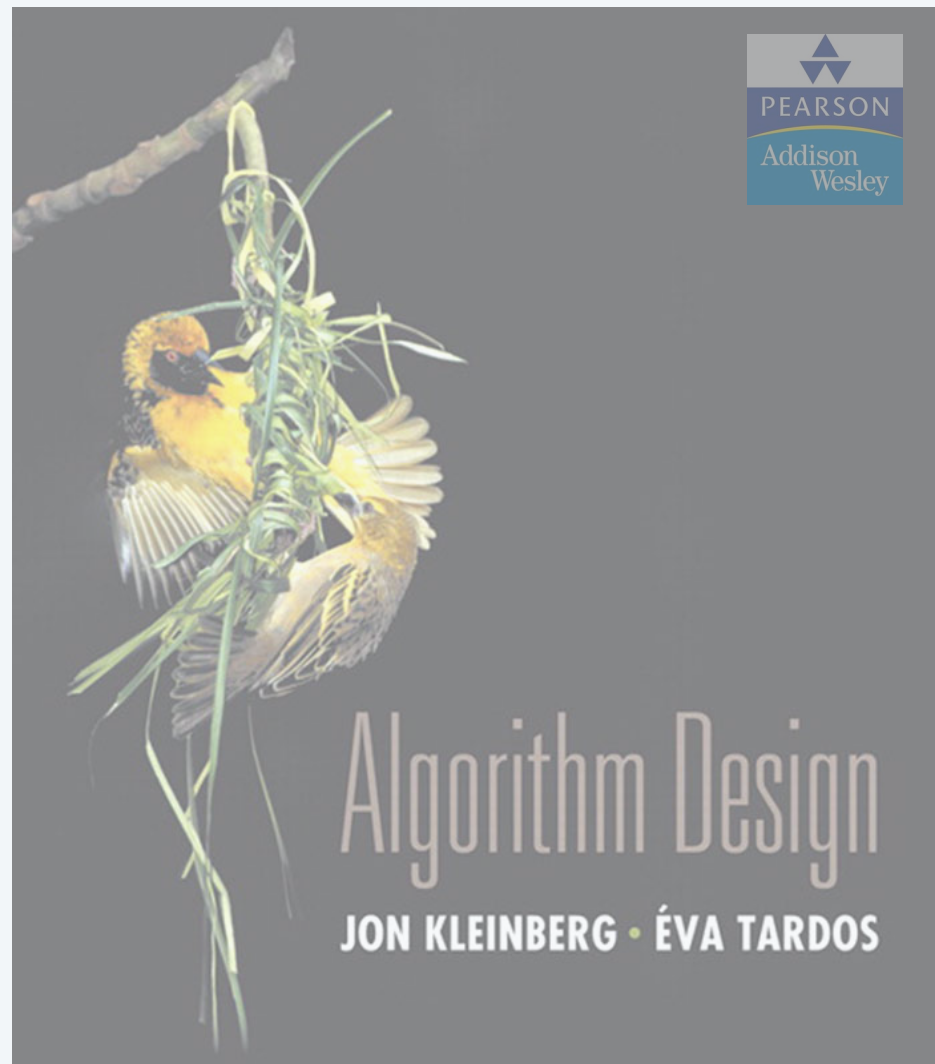
THE MIN-MAX SPANNING TREE PROBLEM AND SOME EXTENSIONS

P.M. CAMERINI

*Istituto di Elettrotecnica ed Elettronica, Politecnico di Milano,
Piazza L. da Vinci 32, 20133 Milano, Italy*

Received 29 July 1977; revised version received 15 September 1977

Spanning trees, min-max problems, computational complexity, matroids



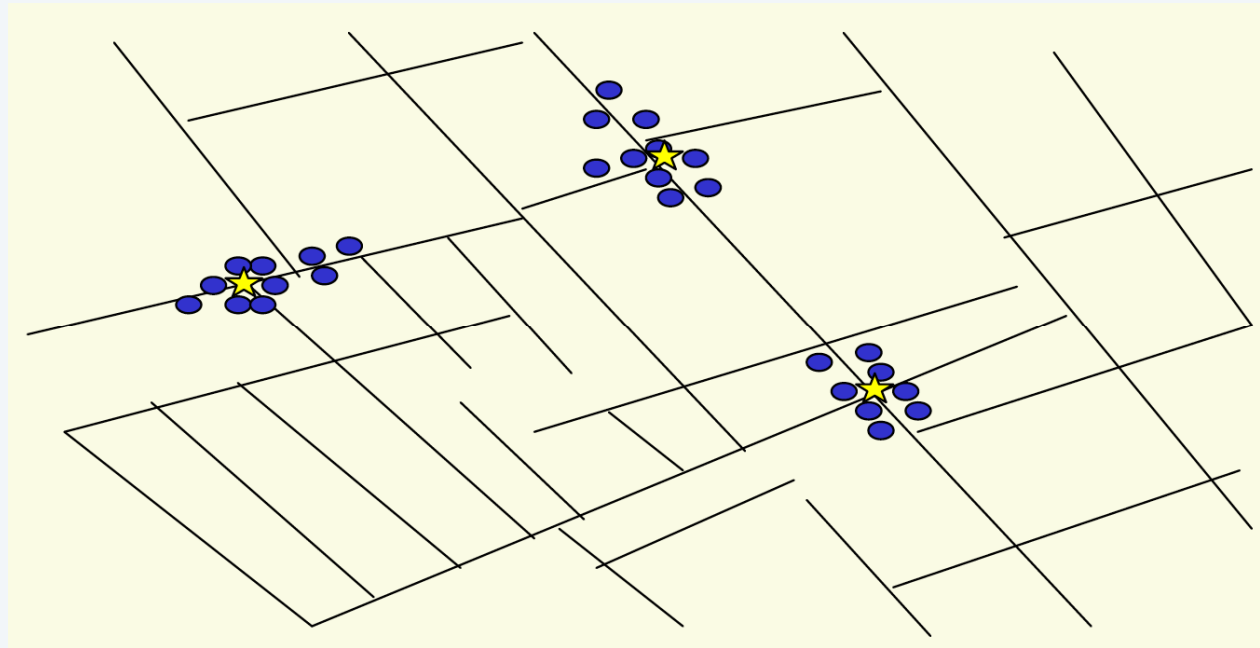
SECTION 4.7

4. GREEDY ALGORITHMS II

- ▶ *minimum spanning trees*
- ▶ *Prim, Kruskal, Boruvka*
- ▶ ***single-link clustering***

Clustering

Goal. Given a set U of n objects labeled p_1, \dots, p_n , partition into clusters so that objects in different clusters are far apart.



outbreak of cholera deaths in London in 1850s (Nina Mishra)

Applications.

- Routing in mobile ad-hoc networks.
- Document categorization for web search.
- Similarity searching in medical image databases
- Cluster celestial objects into stars, quasars, galaxies.
- ...

Clustering of maximum spacing

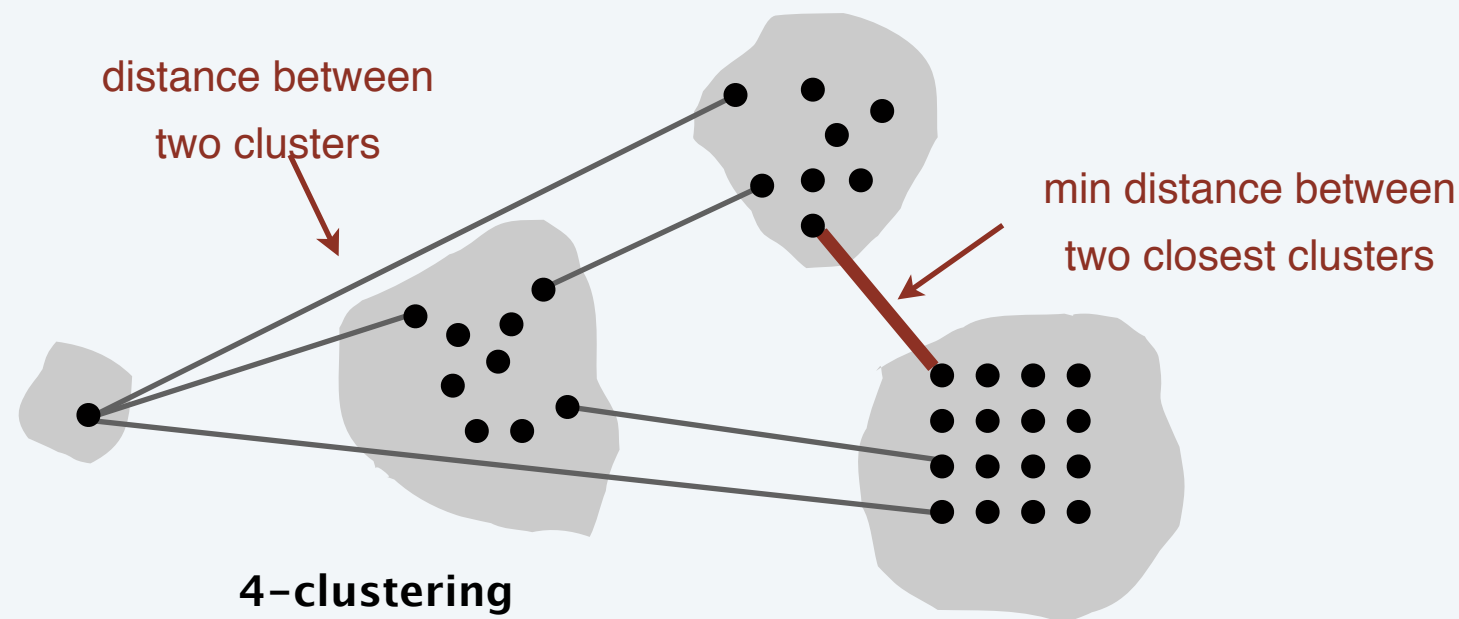
k-clustering. Divide objects into k non-empty groups.

Distance function. Numeric value specifying “closeness” of two objects.

- $d(p_i, p_j) = 0$ iff $p_i = p_j$ [identity of indiscernibles]
- $d(p_i, p_j) \geq 0$ [non-negativity]
- $d(p_i, p_j) = d(p_j, p_i)$ [symmetry]

Spacing. Min distance between any pair of points in different clusters.

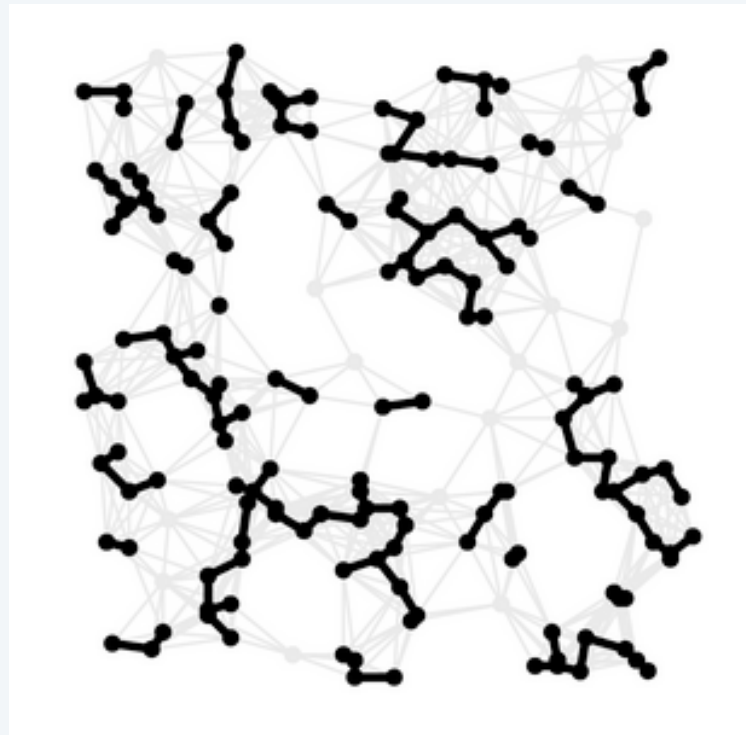
Goal. Given an integer k , find a k -clustering of maximum spacing.



Greedy clustering algorithm

“Well-known” algorithm in science literature for single-linkage k -clustering:

- Form a graph on the node 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).

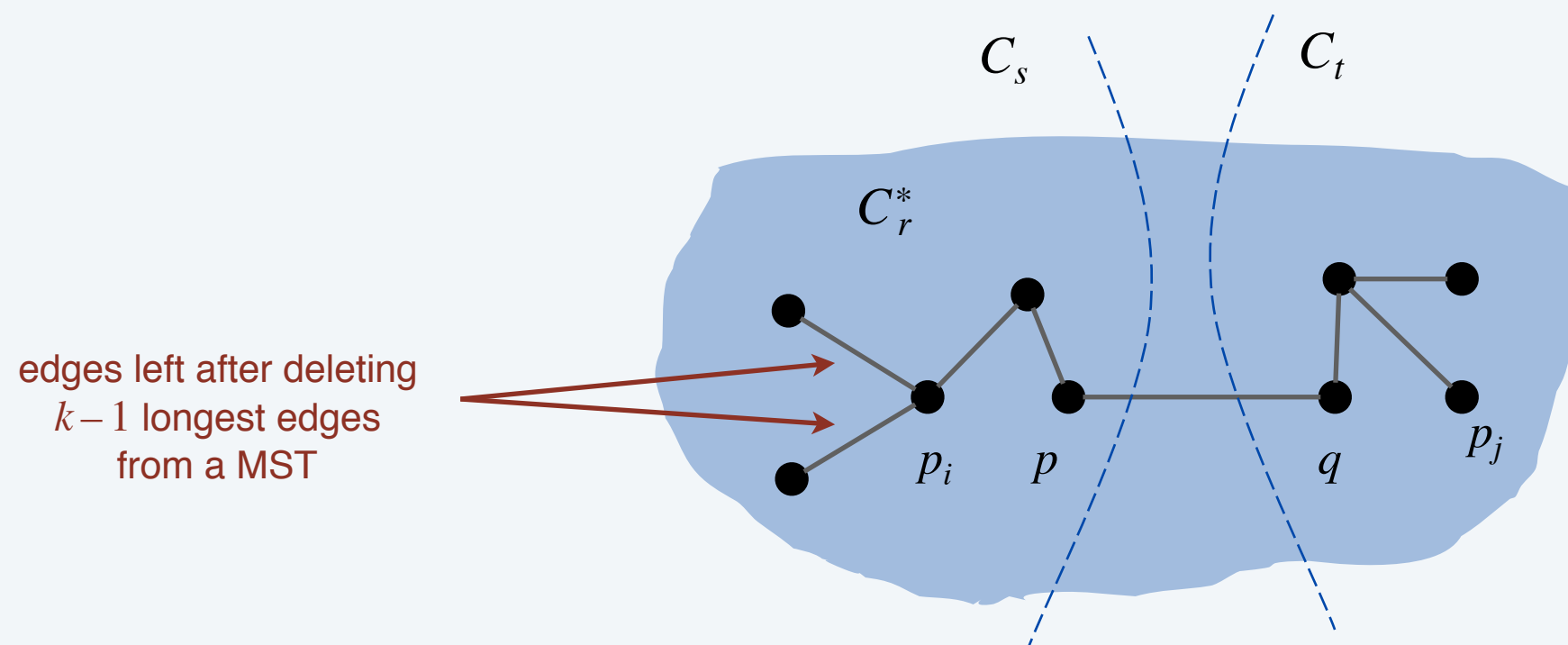
Alternative. Find an MST and delete the $k - 1$ longest edges.

Greedy clustering algorithm: analysis

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

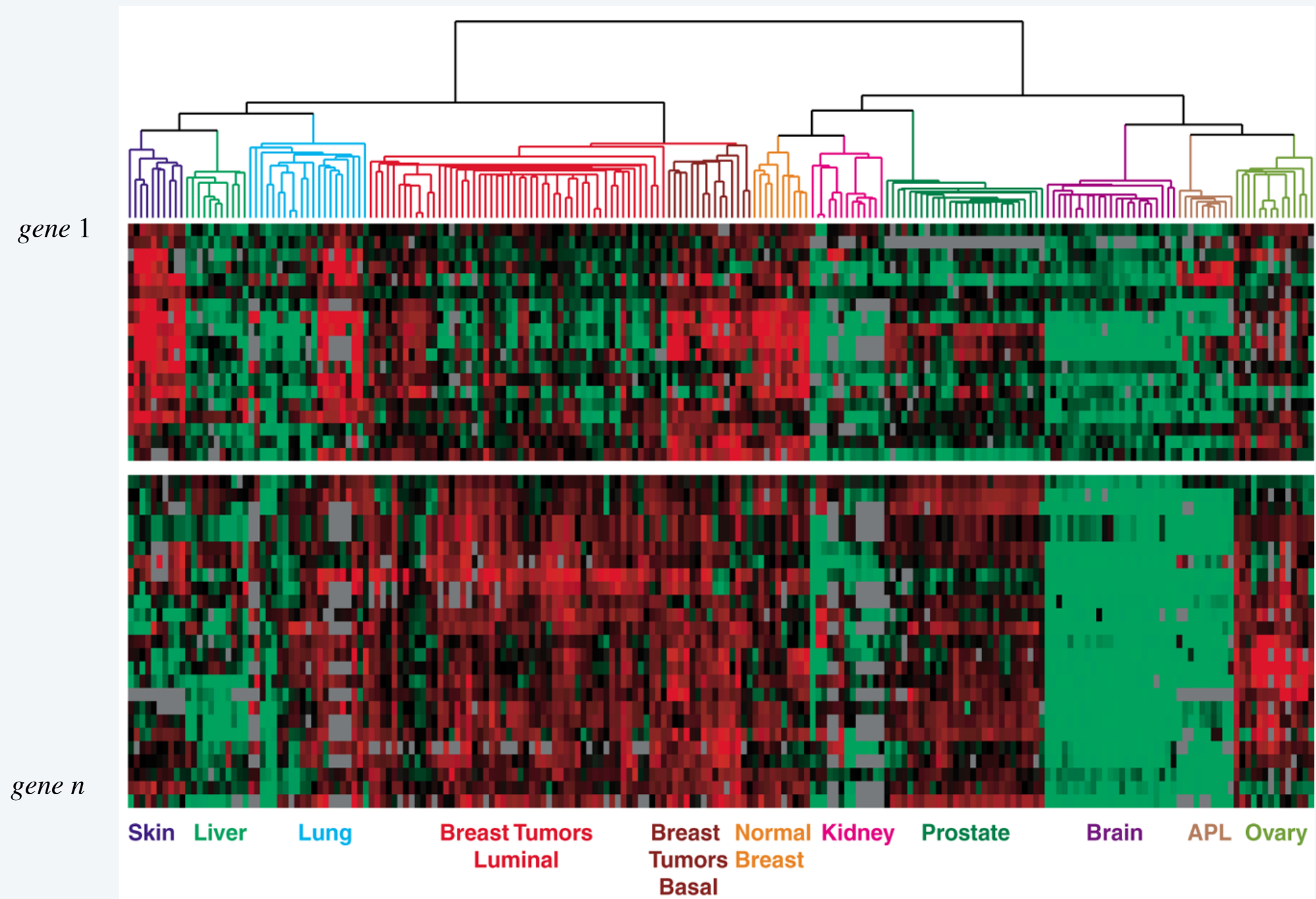
Pf.

- Let C denote any other clustering C_1, \dots, C_k .
- Let p_i and 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 .
- The spacing of C^* is the length d^* of the $(k - 1)^{\text{st}}$ longest edge in MST.
- Edge (p, q) has length $\leq d^*$ since it wasn't deleted.
- Spacing of C is $\leq d^*$ since p and q are in different clusters. ■



Dendrogram of cancers in human

Tumors in similar tissues cluster together.



Reference: Botstein & Brown group

gene expressed
gene not expressed



Which MST algorithm should you use for single-link clustering?

↑
number of objects n
can be very large

- A. Kruskal (stop when there are k components).
- B. Prim (delete $k - 1$ longest edges).
- C. Either A or B.
- D. Neither A nor B.