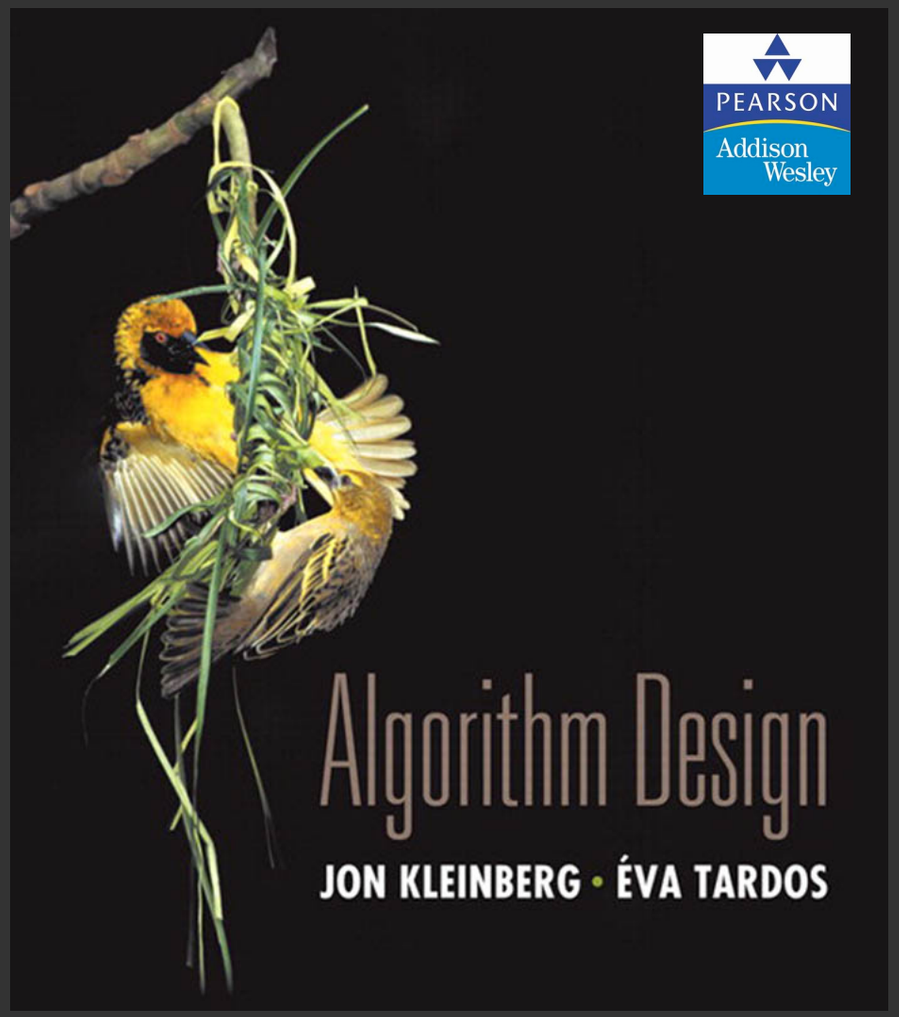


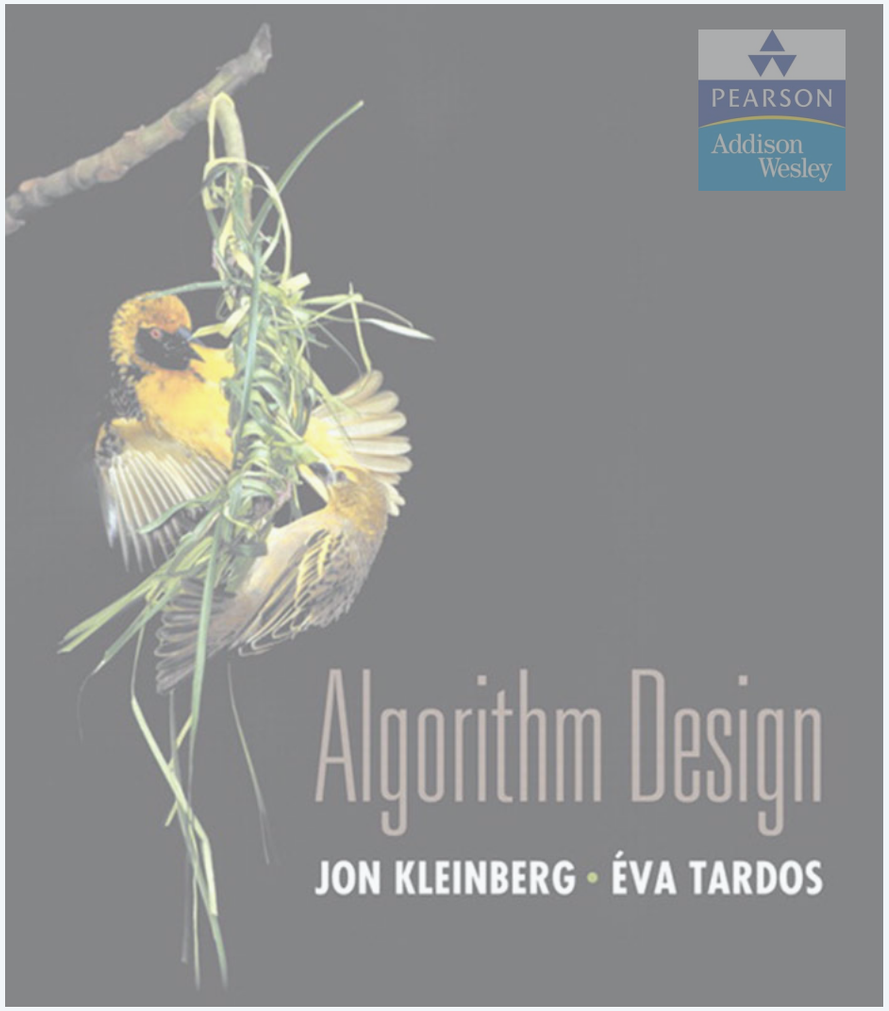
Greedy Algorithms II

- *Dijkstra's algorithm*
- *minimum spanning trees*
- *Prim, Kruskal, Boruvka*
- *single-link clustering*

Lecture slides by Kevin Wayne

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

Last updated on 27/11/2019 22:26



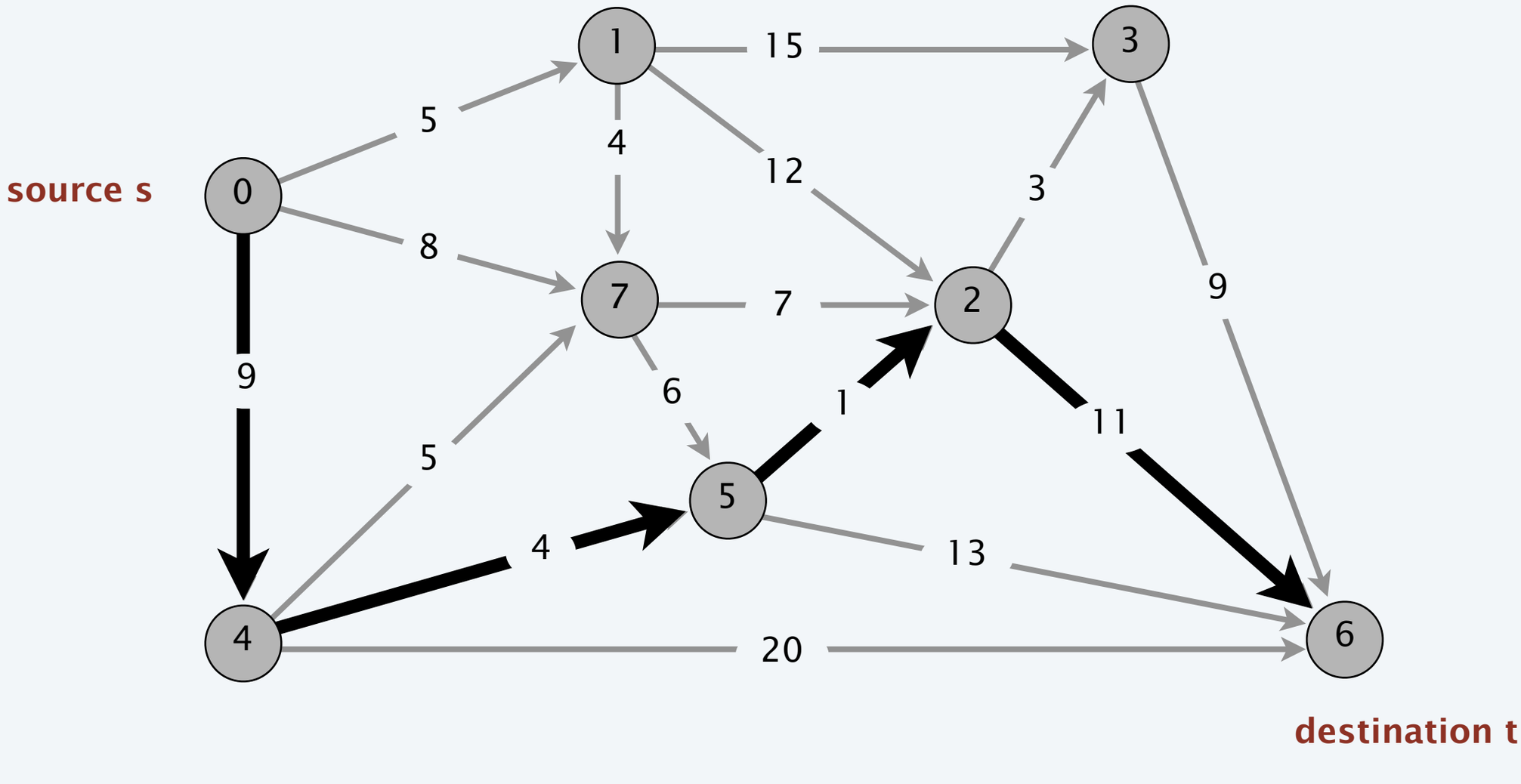
GREEDY ALGORITHMS II

- *Dijkstra's algorithm*
- *minimum spanning trees*
- *Prim, Kruskal, Boruvka*
- *single-link clustering*

SECTION 4.4

Single-pair shortest path problem

Problem. Given a digraph $G = (V, E)$, edge lengths $\ell_e \geq 0$, source $s \in V$, and destination $t \in V$, find a shortest directed path from s to t.

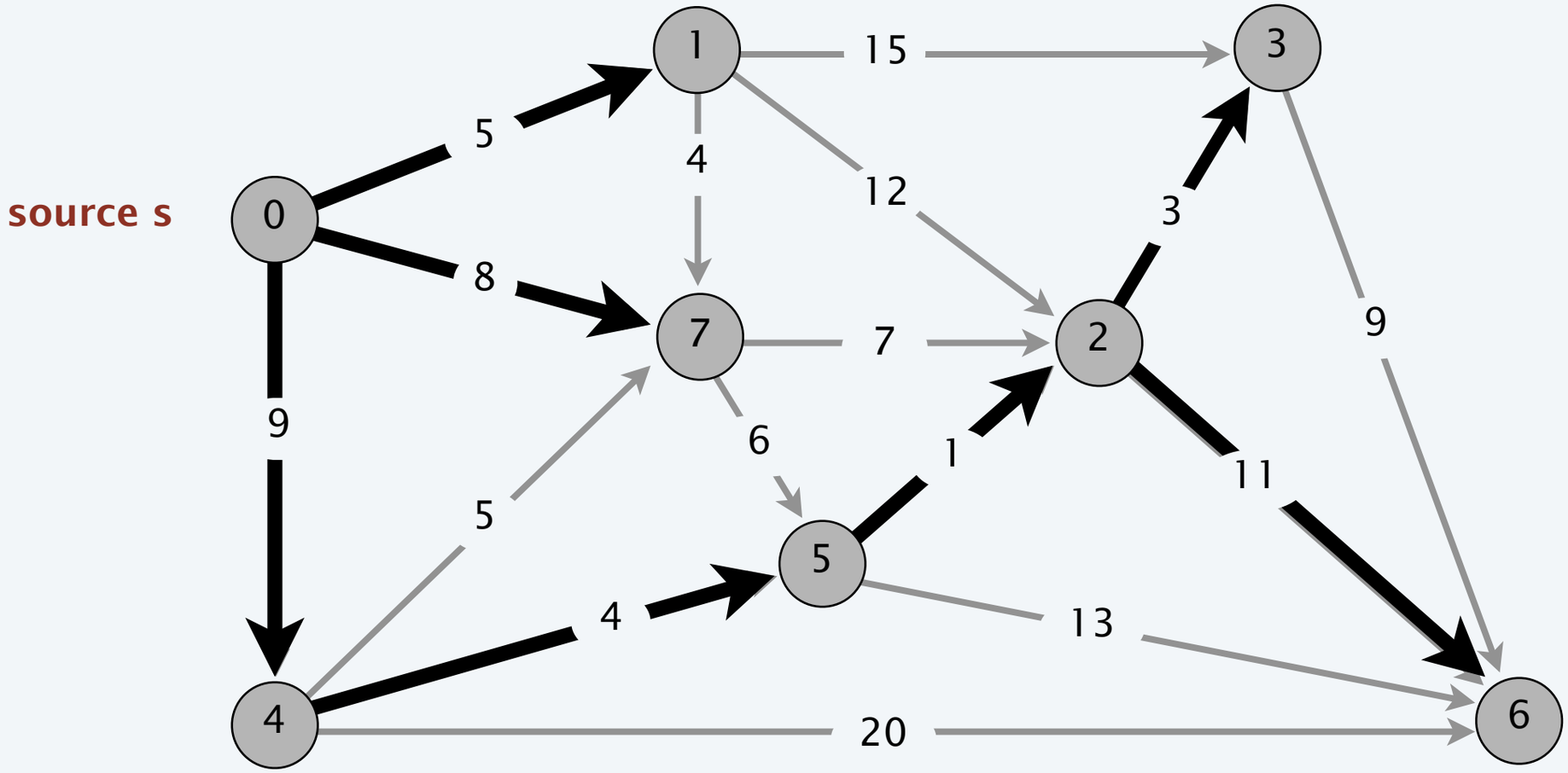


length of path = 9 + 4 + 1 + 11 = 25

Single-source shortest paths problem

Problem. Given a digraph $G = (V, E)$, edge lengths $\ell_e \geq 0$, source $s \in V$, find a shortest directed path from s to every node.

Assumption. There exists a path from s to every node.



shortest-paths tree

Shortest paths: quiz 1

Suppose that you change the length of every edge of G as follows. For which is every shortest path in G a shortest path in G′?

A. Add 17.

B. Multiply by 17.

C. Either A or B.

D. Neither A nor B.

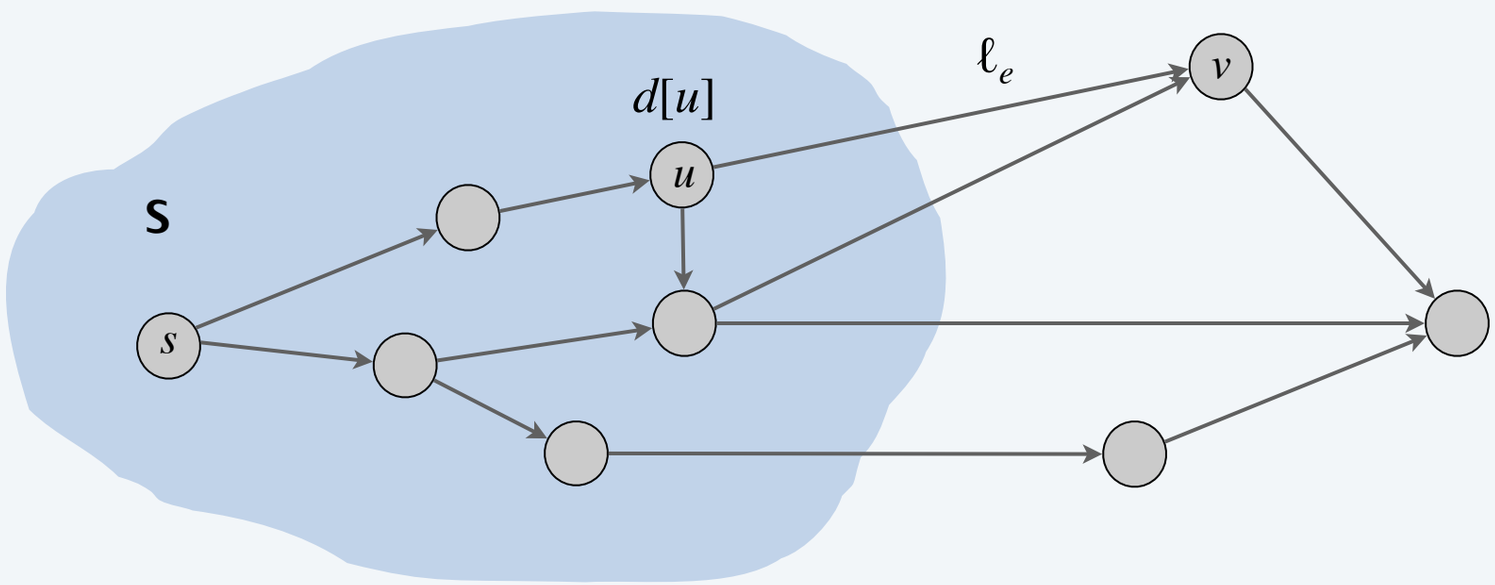
Dijkstra's algorithm (for single-source shortest paths problem)

Greedy approach. Maintain a set of explored nodes S for which algorithm has determined $d[u]$ = length of a shortest $s \rightsquigarrow u$ path.

- Initialize $S \leftarrow \{ s \}$, $d[s] \leftarrow 0$.
- Repeatedly choose unexplored node $v \notin S$ which minimizes

$$\pi(v) \;=\; \min_{e=(u,v)\,:\,u \in S} d[u] + \ell_e$$

the length of a shortest path from s to some node u in explored part S, followed by a single edge $e = (u, v)$



Dijkstra's algorithm (for single-source shortest paths problem)

Greedy approach. Maintain a set of explored nodes S for which algorithm has determined $d[u]$ = length of a shortest $s \rightsquigarrow u$ path.

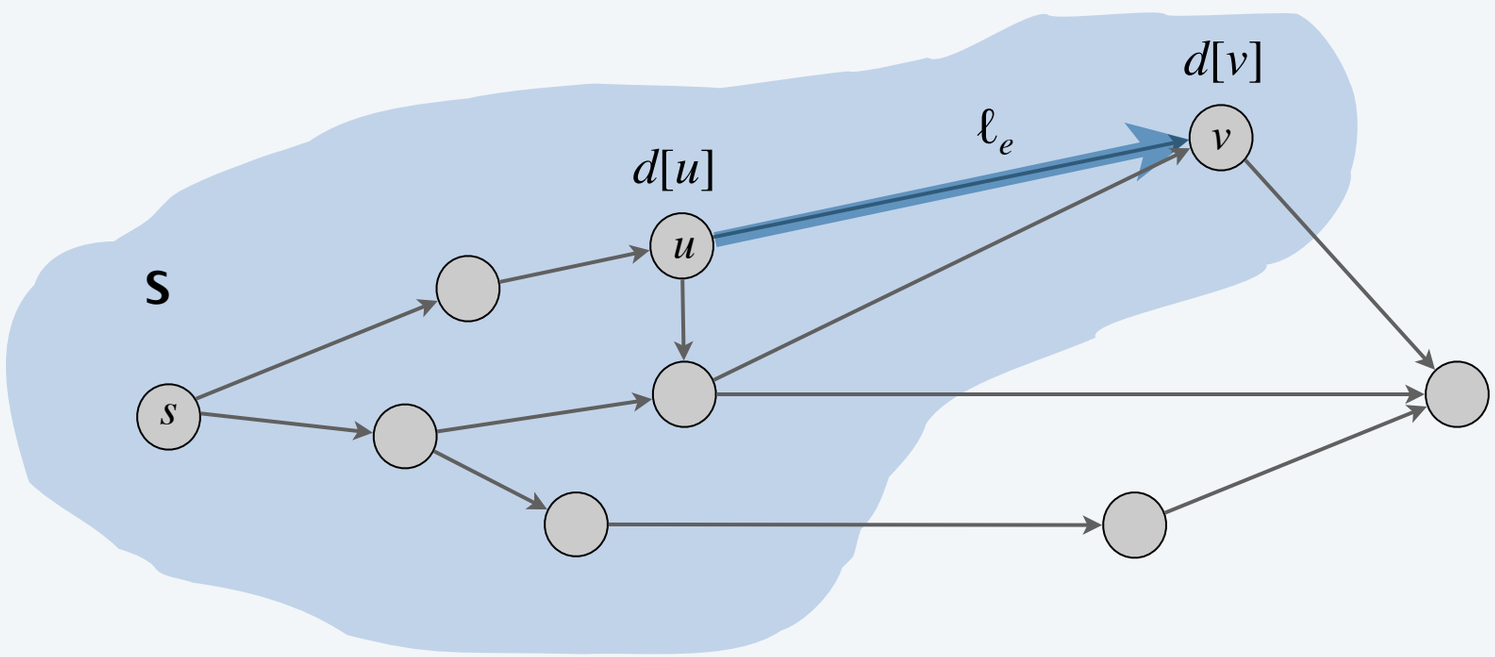
- Initialize $S \leftarrow \{ s \}$, $d[s] \leftarrow 0$.
- Repeatedly choose unexplored node $v \notin S$ which minimizes

$$\pi(v) \;=\; \min_{e=(u,v)\,:\,u \in S} d[u] + \ell_e$$

the length of a shortest path from s to some node u in explored part S, followed by a single edge $e = (u, v)$

add v to S, and set $d[v] \leftarrow \pi(v)$.

- To recover path, set $pred[v] \leftarrow e$ that achieves min.



Dijkstra's algorithm: proof of correctness

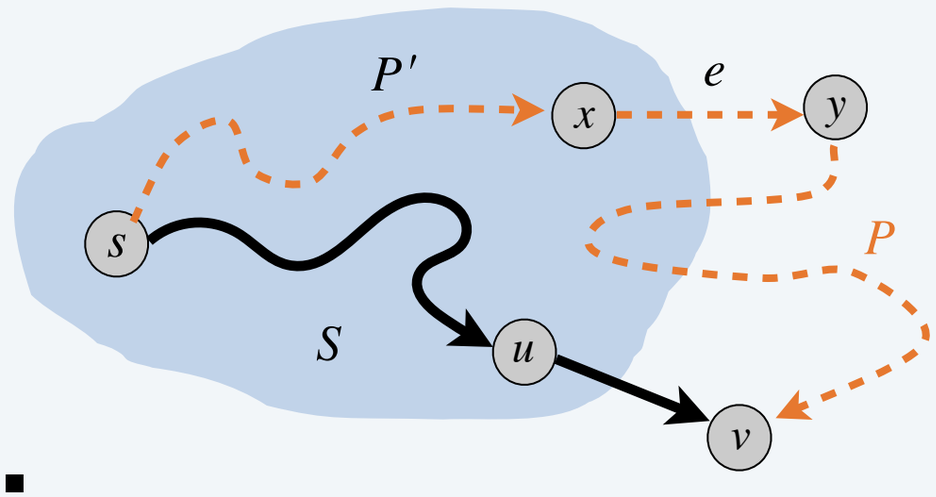
Invariant. For each node $u \in S$: $d[u]$ = length of a shortest $s \rightsquigarrow u$ path.

Pf. [by induction on $|S|$]

Base case: $|S| = 1$ is easy since $S = \{ s \}$ and $d[s] = 0$.

Inductive hypothesis: Assume true for $|S| \geq 1$.

- Let v be next node added to S, and let (u, v) be the final edge.
- A shortest $s \rightsquigarrow u$ path plus (u, v) is an $s \rightsquigarrow v$ path of length $\pi(v)$.
- Consider any other $s \rightsquigarrow v$ path P. We show that it is no shorter than $\pi(v)$.
- Let $e = (x, y)$ be the first edge in P that leaves S, and let P' be the subpath from s to x.
- The length of P is already $\geq \pi(v)$ as soon as it reaches y:



$$\ell(P) \geq \ell(P') + \ell_e \geq d[x] + \ell_e \geq \pi(y) \geq \pi(v) \quad \blacksquare$$

- $\geq$: non-negative lengths
- $\geq$: inductive hypothesis
- $\geq$: definition of $\pi(y)$
- $\geq$: Dijkstra chose v instead of y

Dijkstra's algorithm: efficient implementation

Critical optimization 1. For each unexplored node $v \notin S$:
explicitly maintain $\pi[v]$ instead of computing directly from definition

$$\pi(v) \;=\; \min_{e=(u,v)\,:\,u \in S} d[u] + \ell_e$$

- For each $v \notin S$: $\pi(v)$ can only decrease (because set S increases).

- More specifically, suppose u is added to S and there is an edge $e = (u, v)$ leaving u. Then, it suffices to update:

$$\pi[v] \leftarrow \min \{\, \pi[v],\; \pi[u] + \ell_e) \,\}$$

recall: for each $u \in S$,
$\pi[u] = d[u]$ = length of shortest $s \rightsquigarrow u$ path

Critical optimization 2. Use a min-oriented priority queue (PQ)
to choose an unexplored node that minimizes $\pi[v]$.

Dijkstra's algorithm: efficient implementation

Implementation.

- Algorithm maintains $\pi[v]$ for each node v.
- Priority queue stores unexplored nodes, using $\pi[\cdot]$ as priorities.
- Once u is deleted from the PQ, $\pi[u]$ = length of a shortest $s \rightsquigarrow u$ path.

DIJKSTRA (V, E, ℓ, s)

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

Create an empty priority queue pq.

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

WHILE (IS-NOT-EMPTY(pq))

  $u \leftarrow$ DEL-MIN(pq).

  FOREACH edge $e = (u, v) \in E$ leaving u:

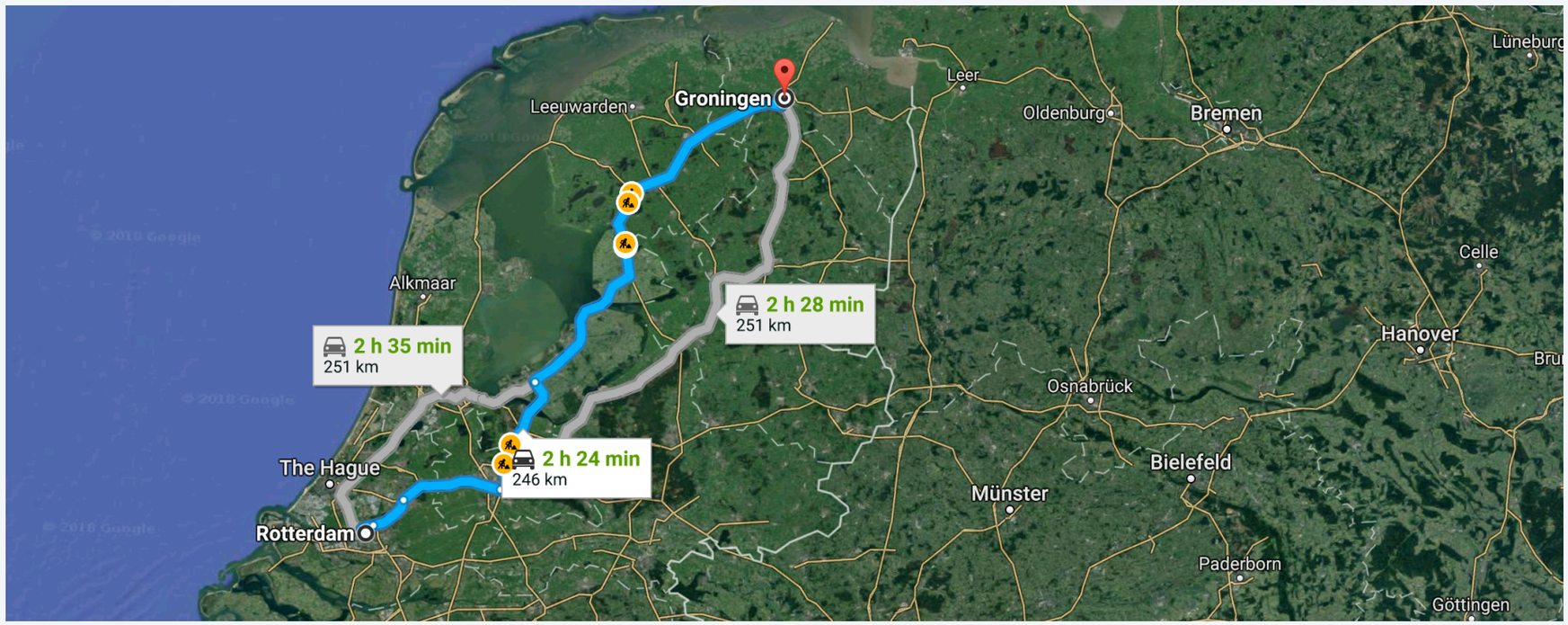
    IF $(\pi[v] > \pi[u] + \ell_e)$

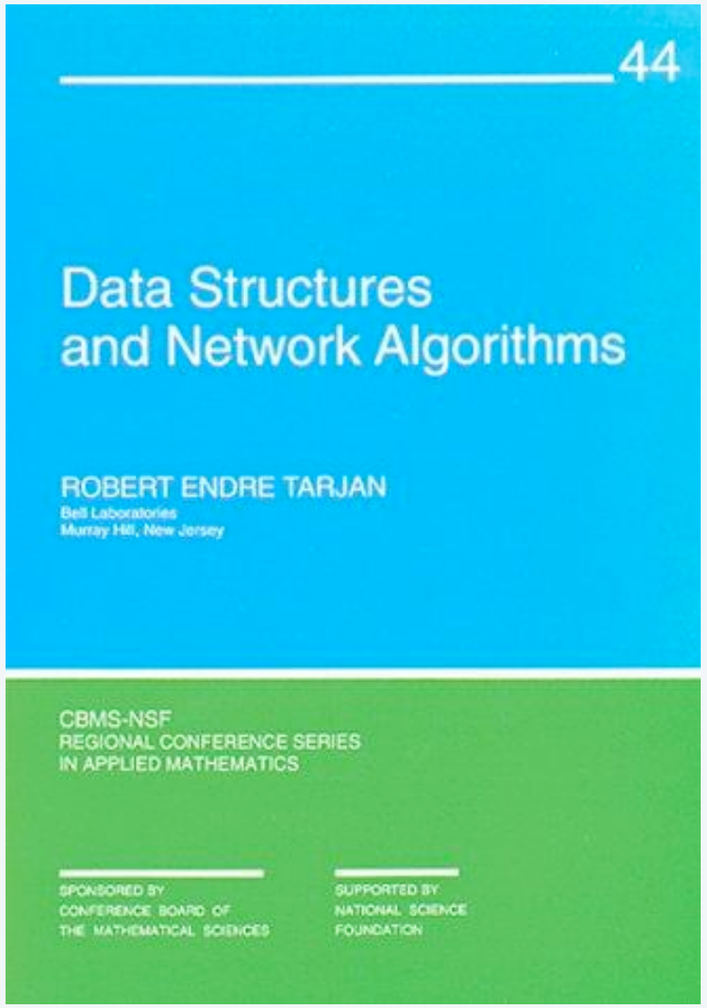
      DECREASE-KEY$(pq,\ v,\ \pi[u] + \ell_e)$.

      $\pi[v] \leftarrow \pi[u] + \ell_e\ ;\ pred[v] \leftarrow e.$

Edsger Dijkstra

> *“ What’s the shortest way to travel from Rotterdam to Groningen? It is the algorithm for the shortest path, which I designed in about 20 minutes. One morning I was shopping in Amsterdam with my young fiancée, and tired, we sat down on the café terrace to drink a cup of coffee and I was just thinking about whether I could do this, and I then designed the algorithm for the shortest path.”* — Edsger Dijsktra





SECTION 6.1

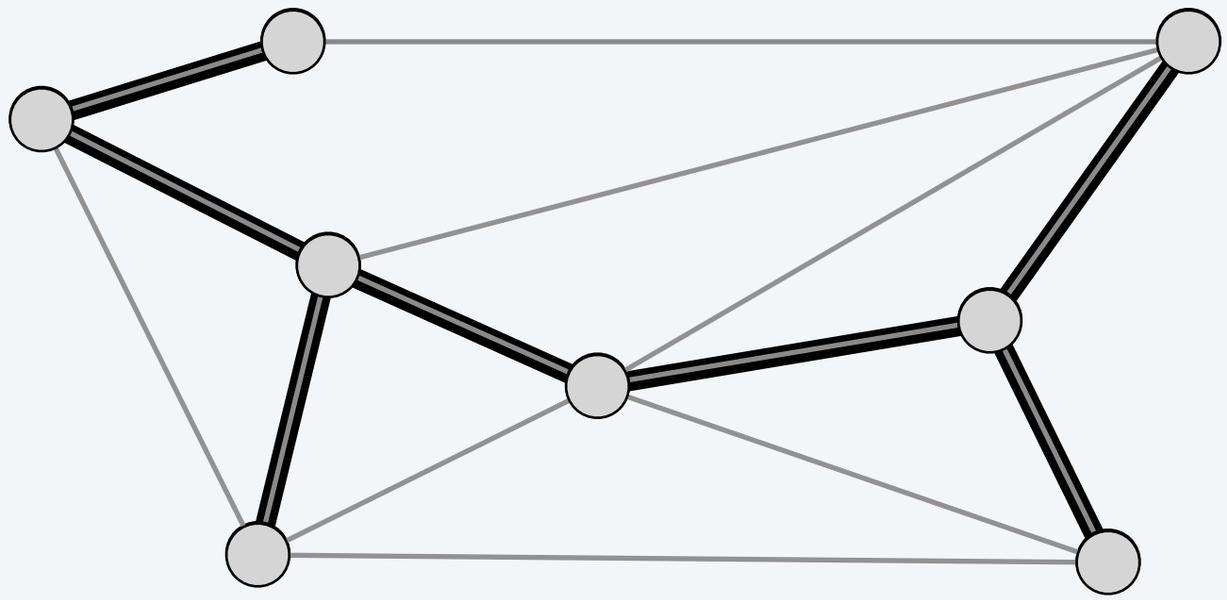
GREEDY ALGORITHMS II

- *Dijkstra's algorithm*
- *minimum spanning trees*
- *Prim, Kruskal, Boruvka*
- *single-link clustering*

Spanning tree definition

Def. Let $H = (V, T)$ be a subgraph of an undirected graph $G = (V, E)$.

H is a spanning tree of G if H is both acyclic and connected.



graph G = (V, E)

spanning tree H = (V, T)

Minimum spanning trees: quiz 3

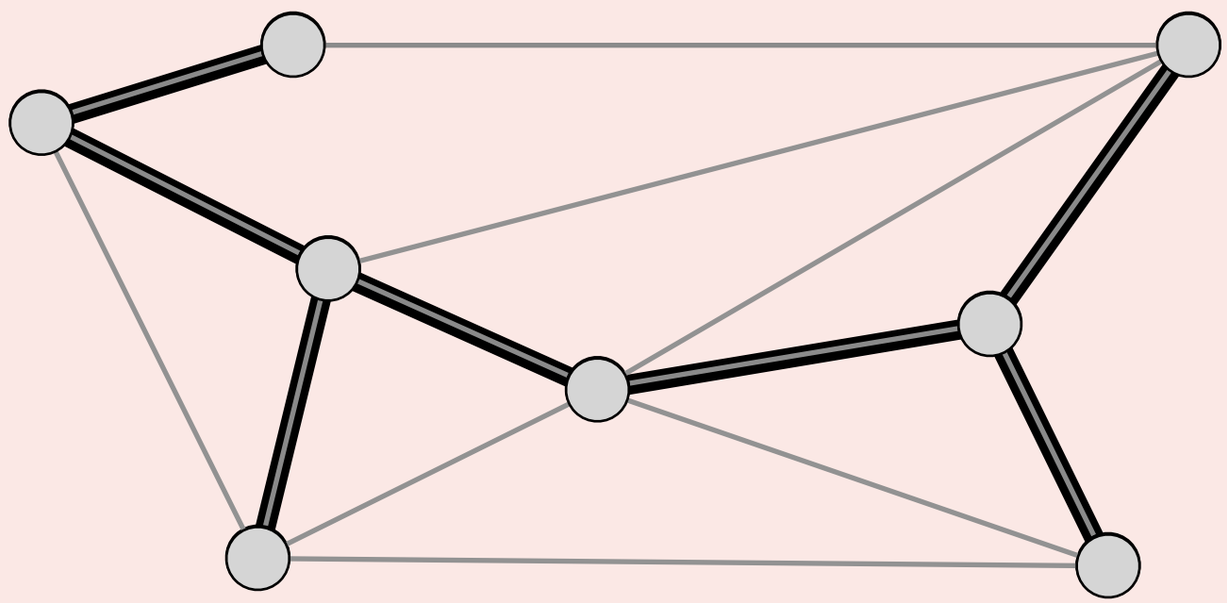
Which of the following properties are true for all spanning trees H?

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.



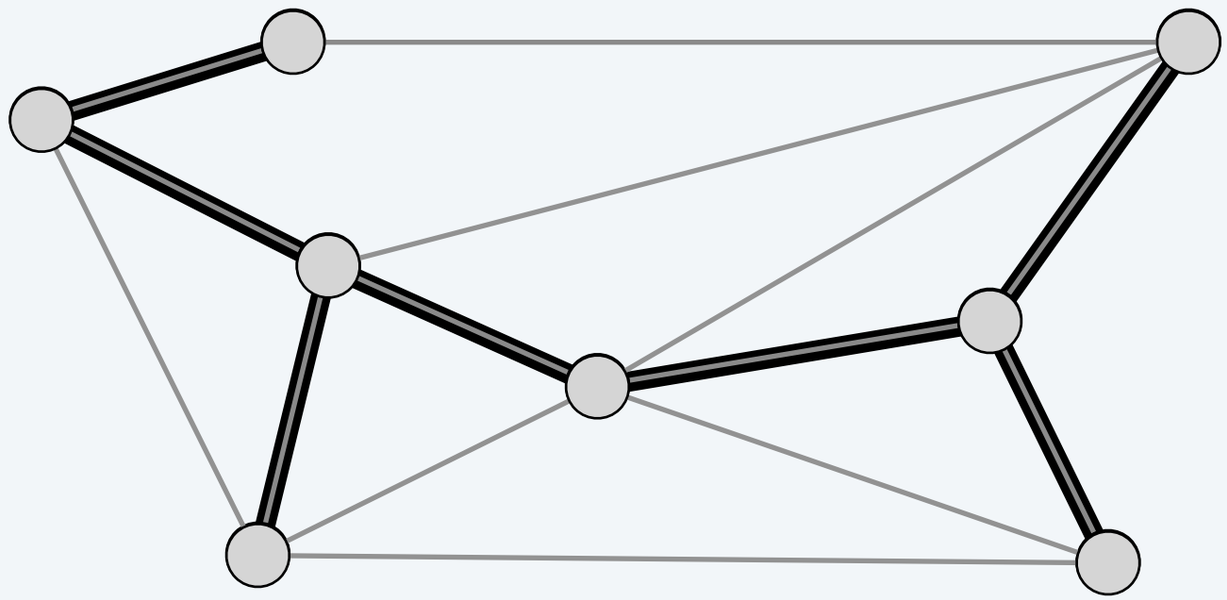
graph G = (V, E)

spanning tree H = (V, T)

Spanning tree properties

Proposition. Let $H = (V, T)$ be a subgraph of an undirected graph $G = (V, E)$.
Then, the following are equivalent:

- H is a spanning tree of G.
- H is acyclic and connected.
- H is connected and has $|V| - 1$ edges.
- H is acyclic and has $|V| - 1$ edges.
- H is minimally connected: removal of any edge disconnects it.
- H is maximally acyclic: addition of any edge creates a cycle.

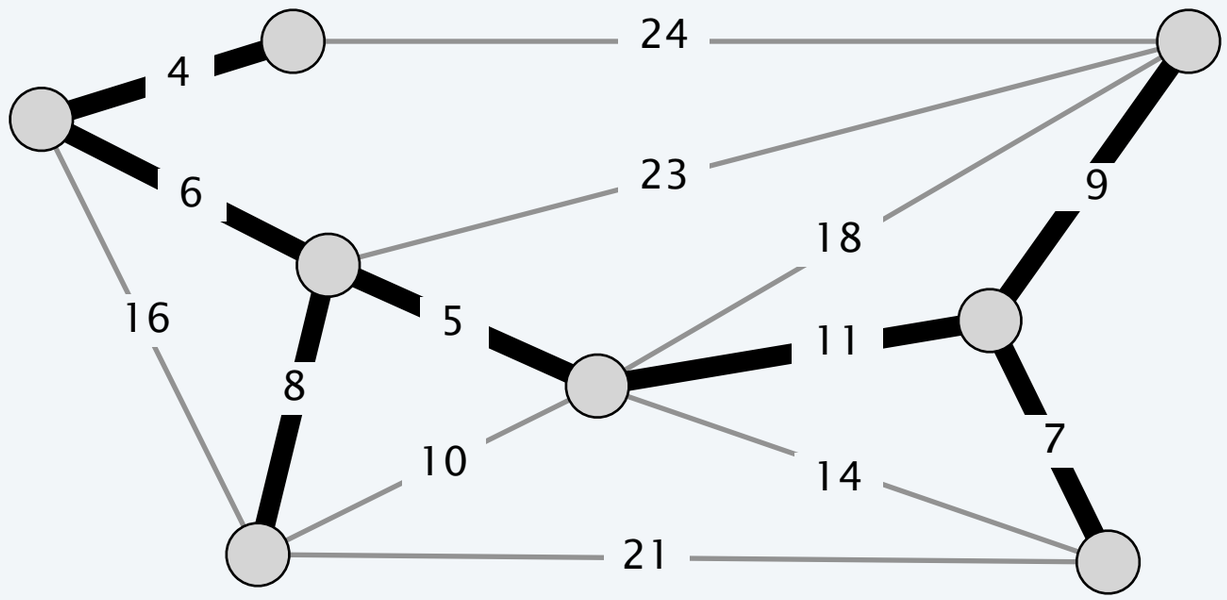


graph G = (V, E)

spanning tree H = (V, T)

Minimum spanning tree (MST)

Def. Given a connected, undirected graph $G = (V, E)$ with edge costs c_e, a minimum spanning tree (V, T) is a spanning tree of G such that the sum of the edge costs in T is minimized.



MST cost = 50 = 4 + 6 + 8 + 5 + 11 + 9 + 7

Cayley's theorem. The complete graph on n nodes has n^{n-2} spanning trees.

can't solve by brute force

Minimum spanning trees: quiz 4

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. $c'(e) = 17 \times c(e).$

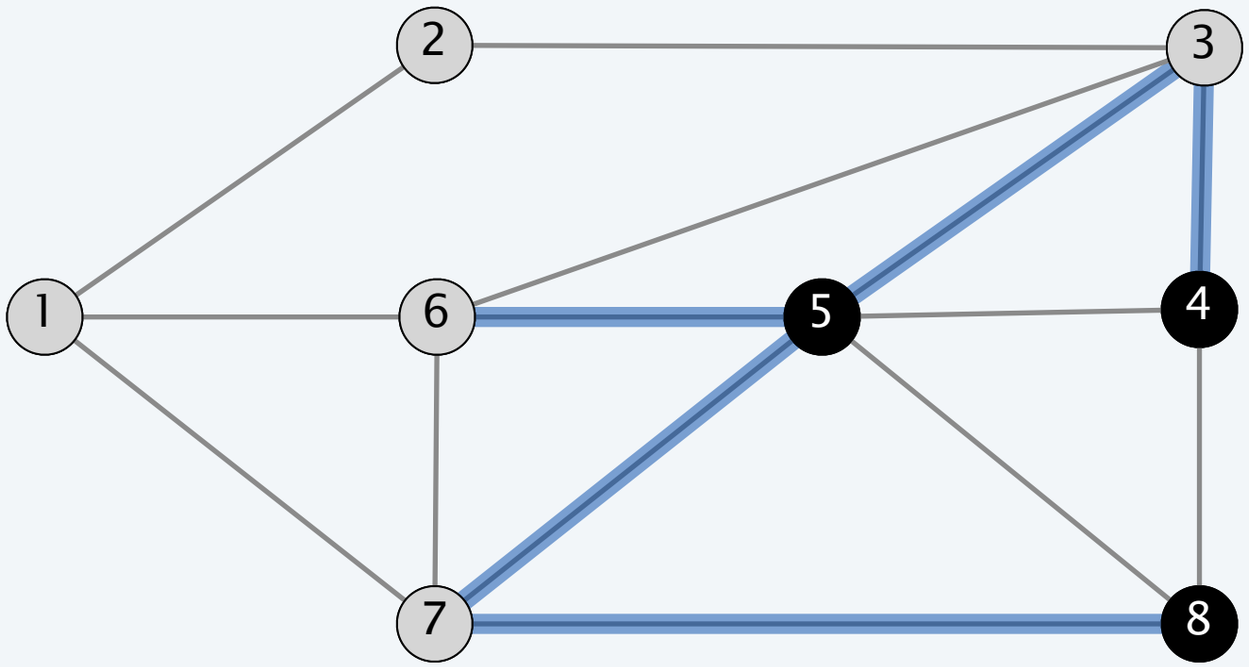
C. $c'(e) = \log_{17} c(e).$

D. All of the above.

Cuts

Def. A cut is a partition of the nodes into two nonempty subsets S and $V - S$.

Def. The cutset of a cut S is the set of edges with exactly one endpoint in S.



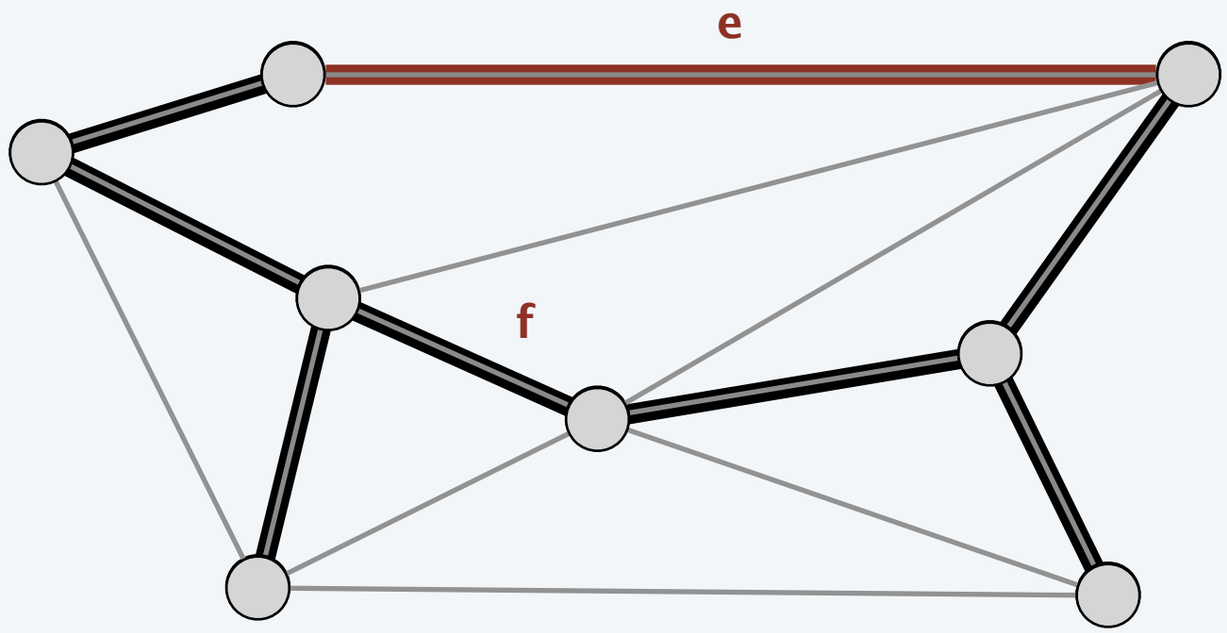
cut S = { 4, 5, 8 }

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

Fundamental cycle

Fundamental cycle. Let $H = (V, T)$ be a spanning tree of $G = (V, E)$.

- For any non tree-edge $e \in E$: $T \cup \{ e \}$ contains a unique cycle, say C.
- For any edge $f \in C$: $T \cup \{ e \} - \{ f \}$ is a spanning tree.



graph G = (V, E)

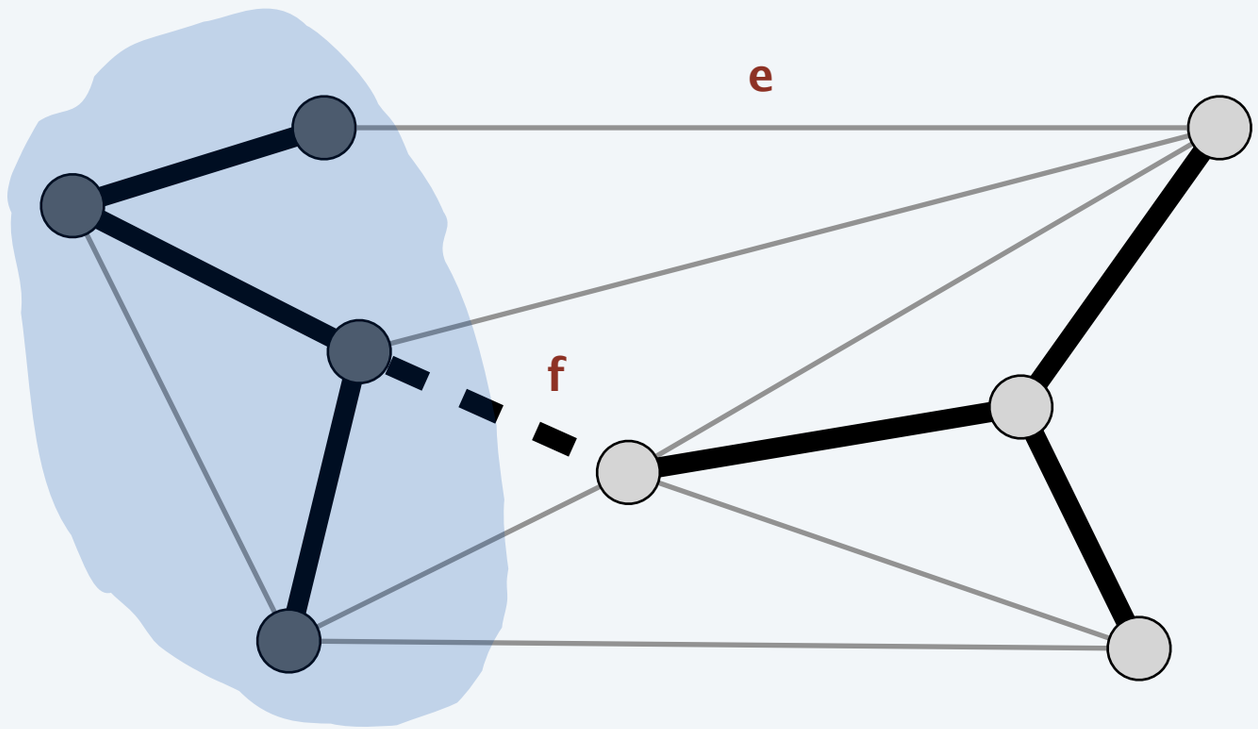
spanning tree H = (V, T)

Observation. If $c_e < c_f$, then (V, T) is not an MST.

Fundamental cutset

Fundamental cutset. Let $H = (V, T)$ be a spanning tree of $G = (V, E)$.

- For any tree edge $f \in T$: $T - \{ f \}$ contains two connected components.
  Let D denote corresponding cutset.
- For any edge $e \in D$: $T - \{ f \} \cup \{ e \}$ is a spanning tree.



graph G = (V, E)

spanning tree H = (V, T)

Observation. If $c_e < c_f$, then (V, T) is not an MST.

The greedy algorithm

Red rule.

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

Blue rule.

- Let D be a cutset with no blue edges.
- Select an uncolored edge in D of min cost 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.

Greedy algorithm: proof of correctness

Color invariant. There exists an MST (V, T^*) containing every blue edge and no red edge.

Pf. [by induction on number of iterations]

Base case. No edges colored $\Rightarrow$ every MST satisfies invariant.

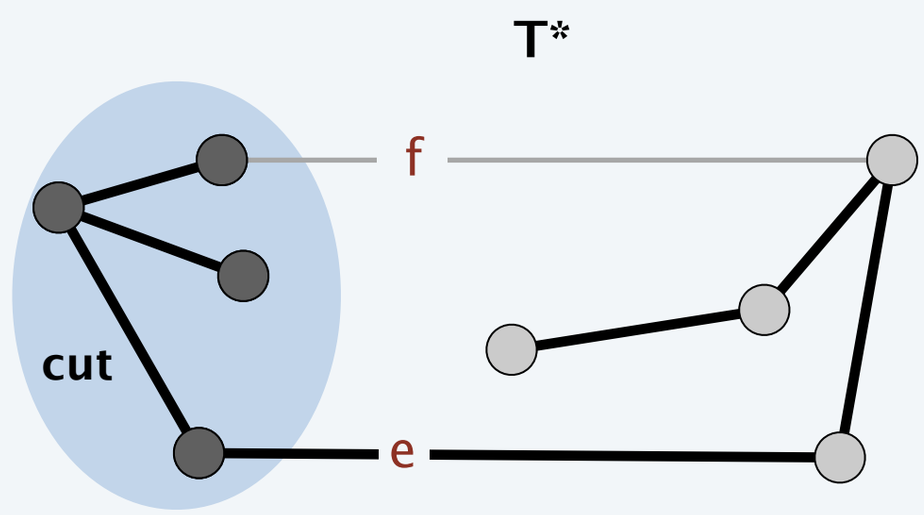
Greedy algorithm: proof of correctness

Color invariant. There exists an MST (V, T^*) containing every blue edge and no red edge.

Pf. [by induction on number of iterations]

Induction step (blue rule). Suppose color invariant true before blue rule.

- let D be chosen cutset, and let f be edge colored blue.
- if $f \in T^*$, then T^* still satisfies invariant.
- Otherwise, consider fundamental cycle C by adding f to T^*.
- let $e \in C$ be another edge in D.
- e is uncolored and $c_e \geq c_f$ since
  - $e \in T^* \Rightarrow e$ not red
  - blue rule $\Rightarrow e$ not blue and $c_e \geq c_f$
- Thus, $T^* \cup \{ f \} - \{ e \}$ satisfies invariant.



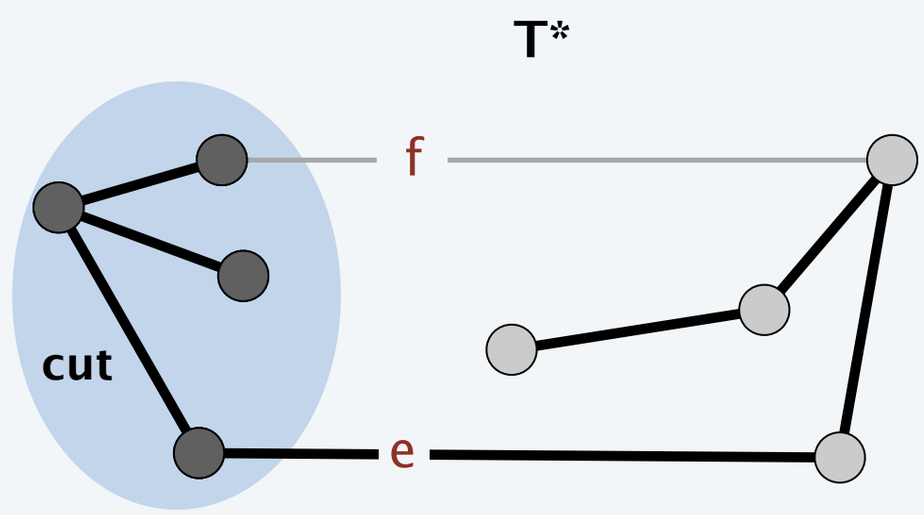
Greedy algorithm: proof of correctness

Color invariant. There exists an MST (V, T^*) containing every blue edge and no red edge.

Pf. [by induction on number of iterations]

Induction step (red rule). Suppose color invariant true before red rule.

- let C be chosen cycle, and let e be edge colored red.
- if $e \notin T^*$, then T^* still satisfies invariant.
- Otherwise, consider fundamental cutset D by deleting e from T^*.
- let $f \in D$ be another edge in C.
- f is uncolored and $c_e \geq c_f$ since
  - $f \notin T^* \Rightarrow f$ not blue
  - red rule $\Rightarrow f$ not red and $c_e \geq c_f$
- Thus, $T^* \cup \{ f \} - \{ e \}$ satisfies invariant. ▪



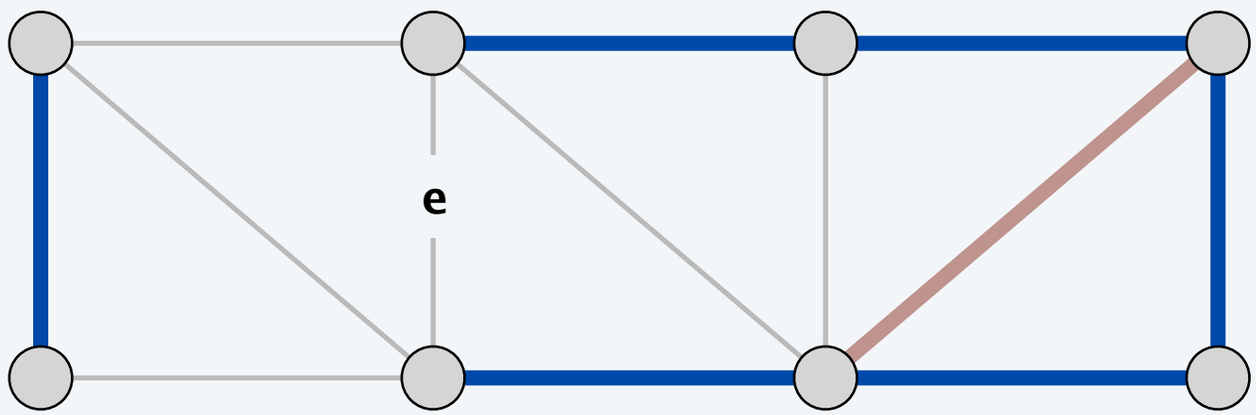
Greedy algorithm: proof of correctness

Theorem. The greedy algorithm terminates. Blue edges form an MST.

Pf. We need to show that either the red or blue rule (or both) applies.

- Suppose edge e is left uncolored.
- Blue edges form a forest.
- Case 1: both endpoints of e are in same blue tree.

  $\Rightarrow$ apply red rule to cycle formed by adding e to blue forest.



Case 1

Greedy algorithm: proof of correctness

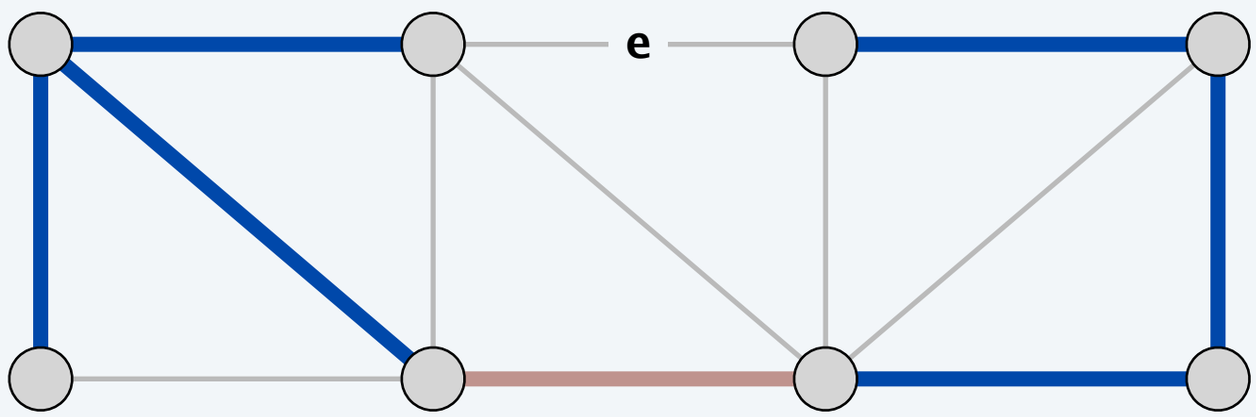
Theorem. The greedy algorithm terminates. Blue edges form an MST.

Pf. We need to show that either the red or blue rule (or both) applies.

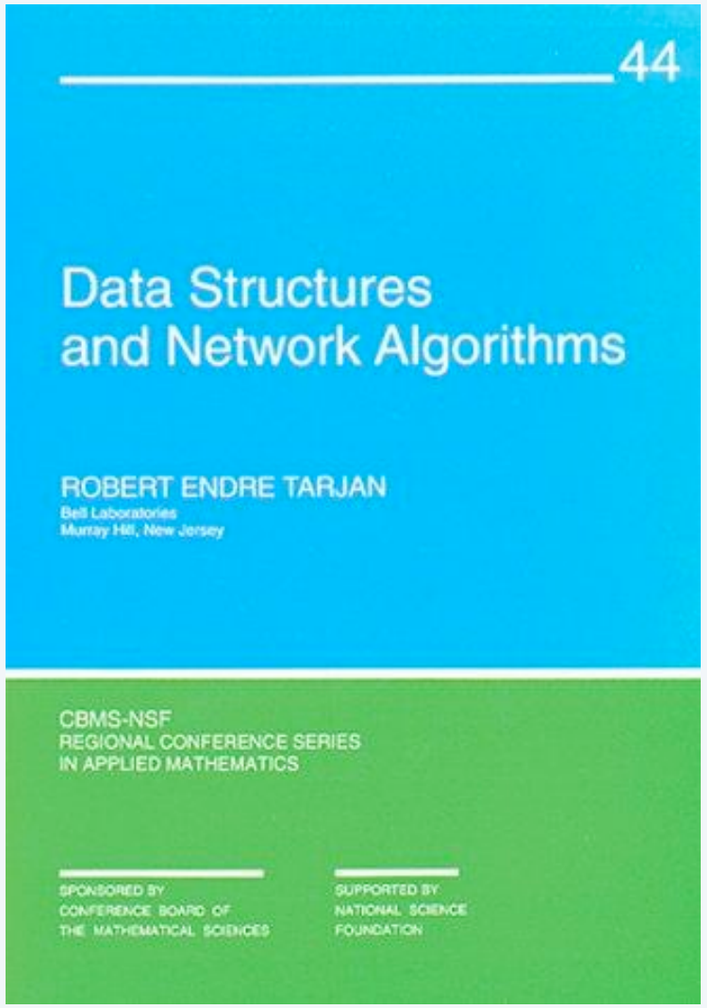
- Suppose edge e is left uncolored.
- Blue edges form a forest.
- Case 1: both endpoints of e are in same blue tree.

  $\Rightarrow$ apply red rule to cycle formed by adding e to blue forest.
- Case 2: both endpoints of e are in different blue trees.

  $\Rightarrow$ apply blue rule to cutset induced by either of two blue trees. ▪



Case 2



SECTION 6.2

GREEDY ALGORITHMS II

- *Dijkstra's algorithm*
- *minimum spanning trees*
- *Prim, Kruskal, Boruvka*
- *single-link clustering*

Prim's algorithm

Initialize S = any node, $T = \emptyset$.

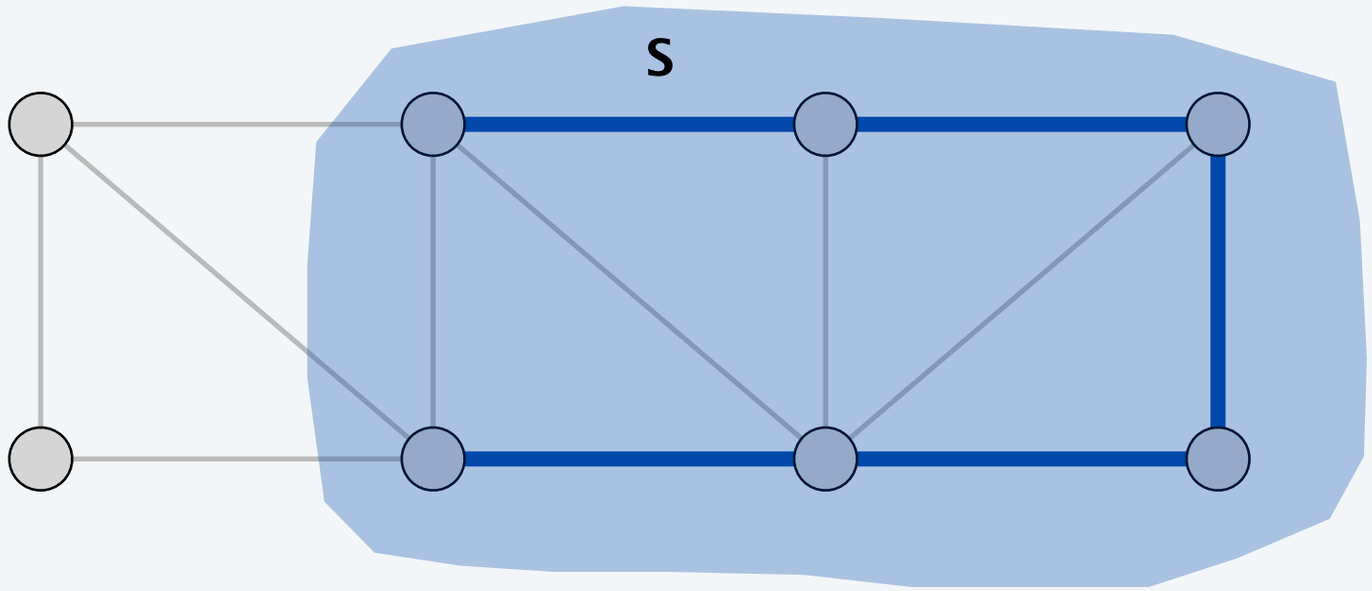
Repeat $n - 1$ times:

- Add to T a min-cost edge with one endpoint in S.
- Add new node to S.

by construction, edges in cutset are uncolored

Theorem. Prim's algorithm computes an MST.

Pf. Special case of greedy algorithm (blue rule repeatedly applied to S). ▪



Kruskal's algorithm

Consider edges in ascending order of cost:

- Add to tree unless it would create a cycle.

Theorem. Kruskal's algorithm computes an MST.

Pf. Special case of greedy algorithm.

- Case 1: both endpoints of e in same blue tree.

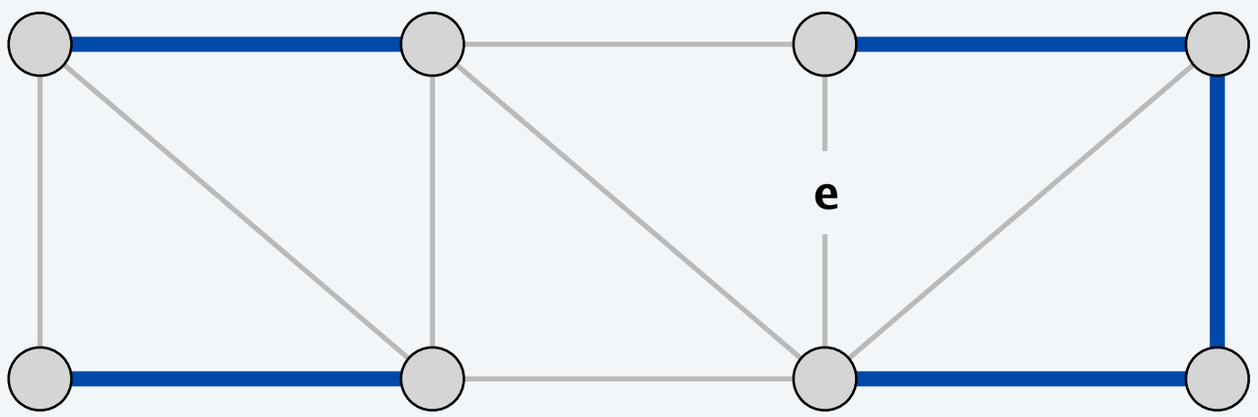
  $\Rightarrow$ color e red by applying red rule to unique cycle.

all other edges in cycle are blue

- Case 2: both endpoints of e in different blue trees.

  $\Rightarrow$ color e blue by applying blue rule to cutset defined by either tree. ▪

no edge in cutset has smaller cost
(since Kruskal chose it first)



Reverse-delete algorithm

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

- 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]

  $\Rightarrow$ apply red rule to cycle C formed by adding e to another path in T between its two endpoints

  no edge in C is more expensive
  (it would have already been considered and deleted)

- Case 2. [deleting edge e disconnects T]

  $\Rightarrow$ 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 cost and color it red.

Blue rule.

- Let D be a cutset with no blue edges.
- Select an uncolored edge in D of min cost 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, ...

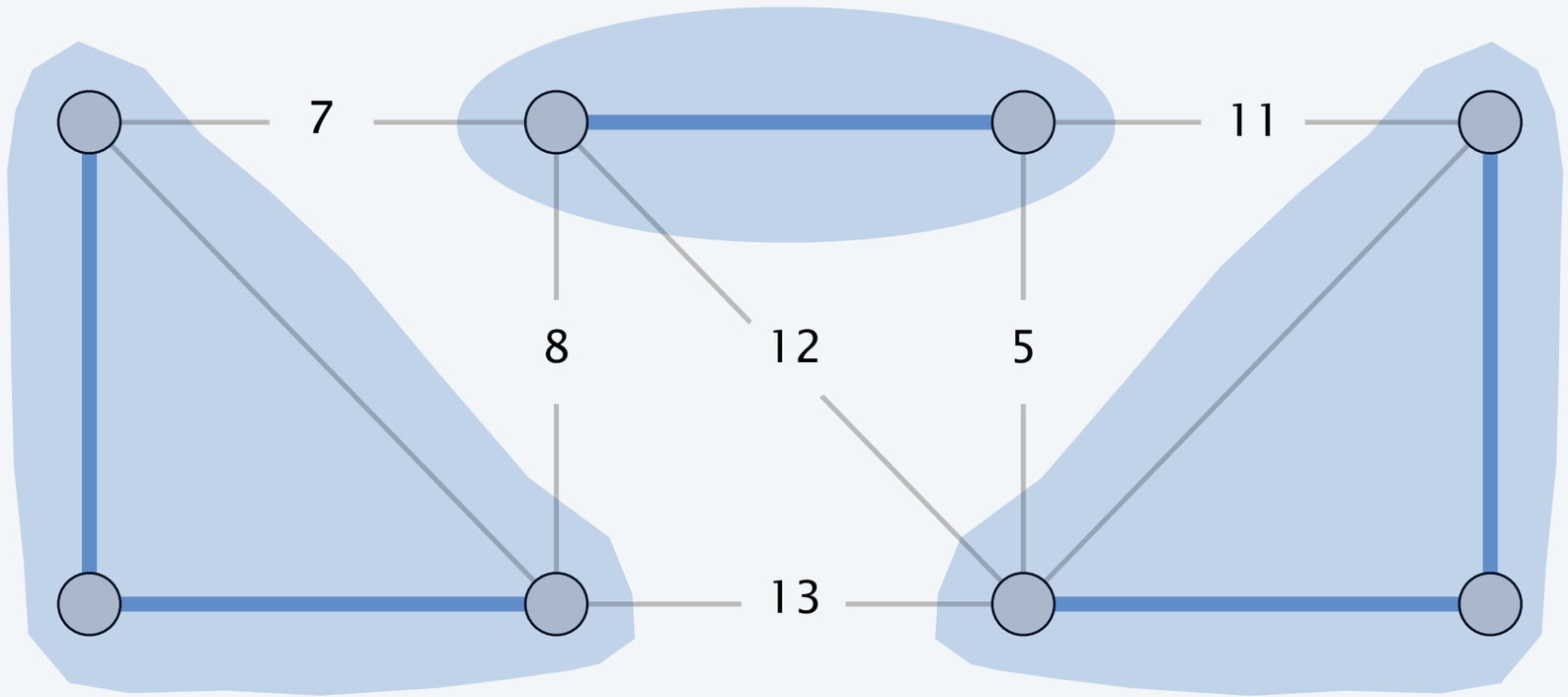
Borůvka's algorithm

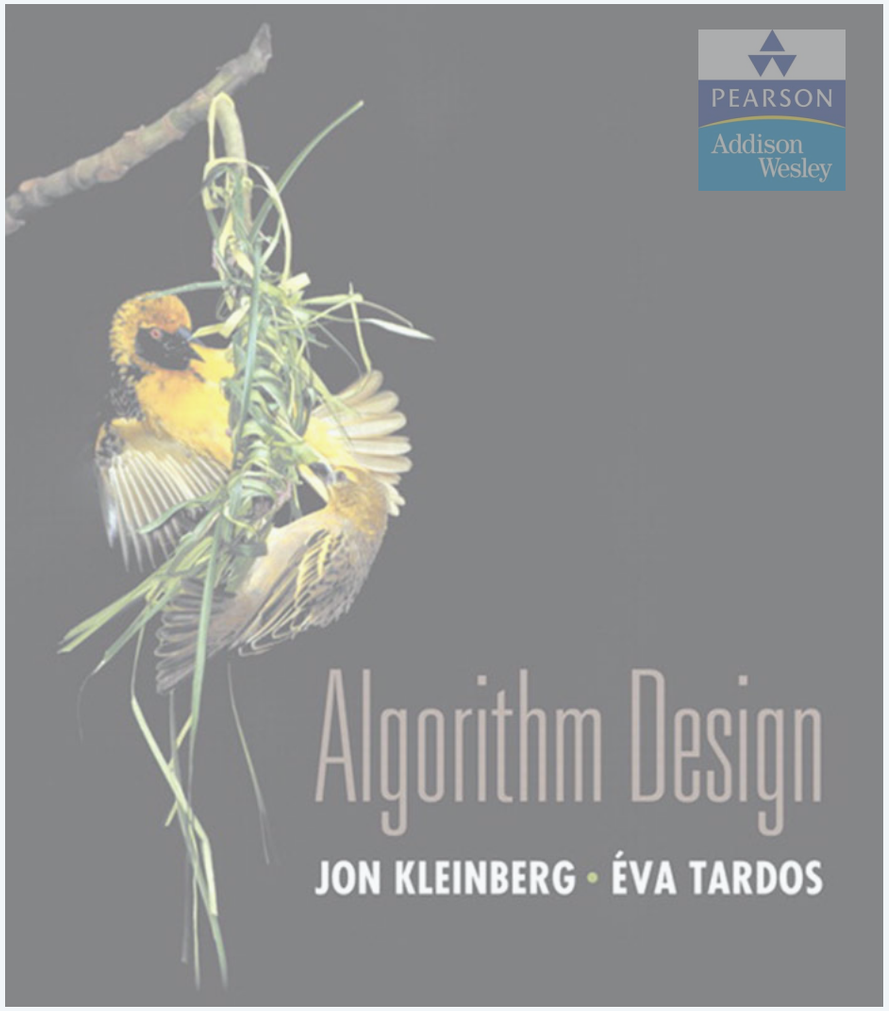
Repeat until only one tree.

- 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). ▪





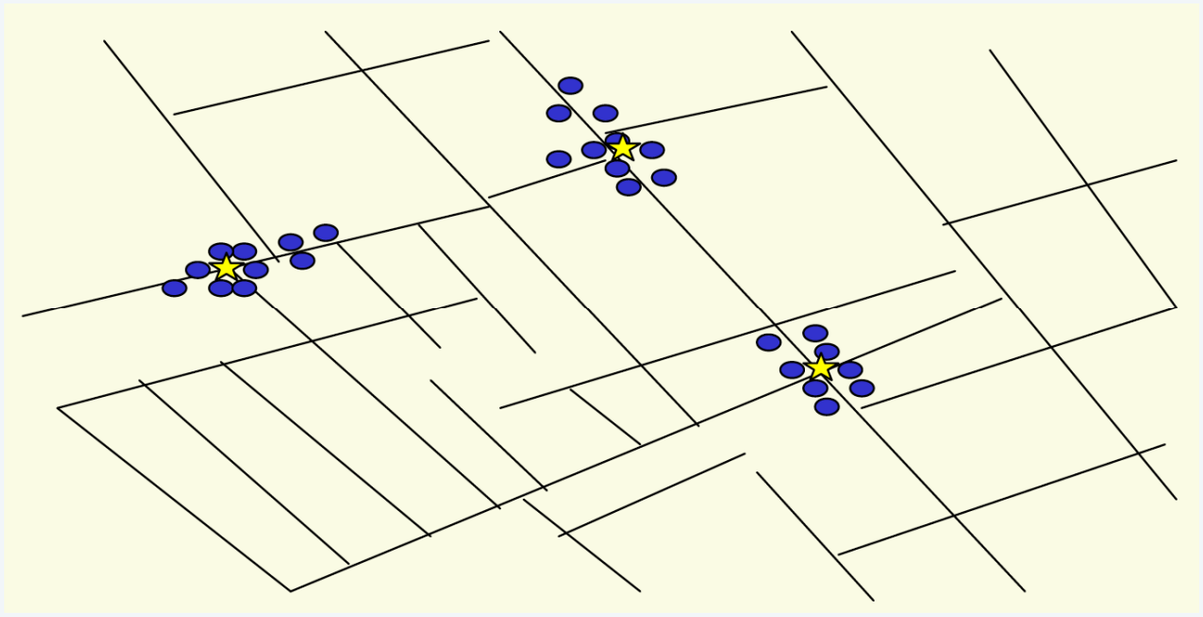
GREEDY ALGORITHMS II

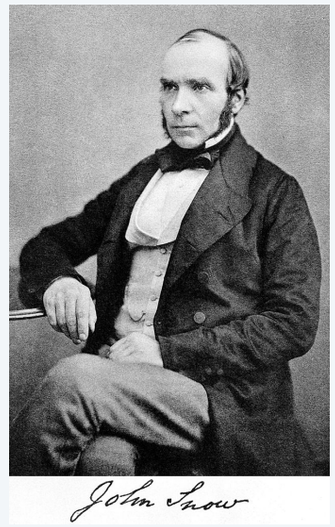
- *Dijkstra's algorithm*
- *minimum spanning trees*
- *Prim, Kruskal, Boruvka*
- *single-link clustering*

SECTION 4.7

Clustering

Goal. Given a set U of n objects labeled $p_1, \ldots, p_n$, 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

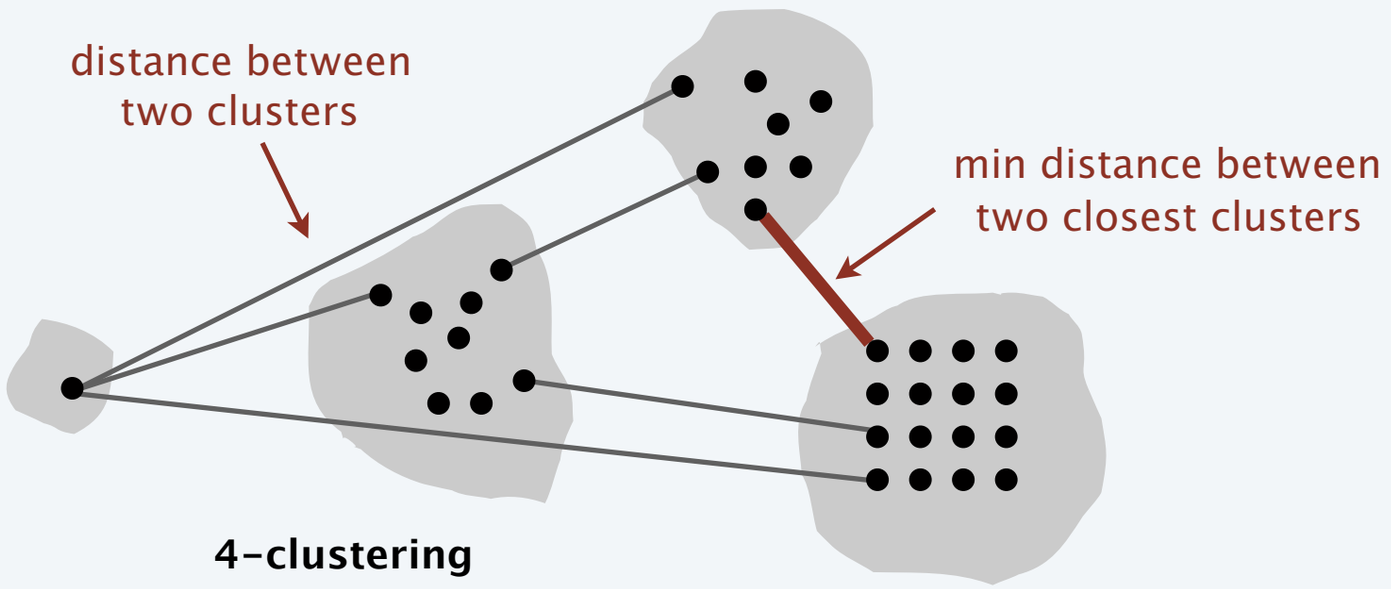
k-clustering. Divide objects into k non-empty groups.

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.

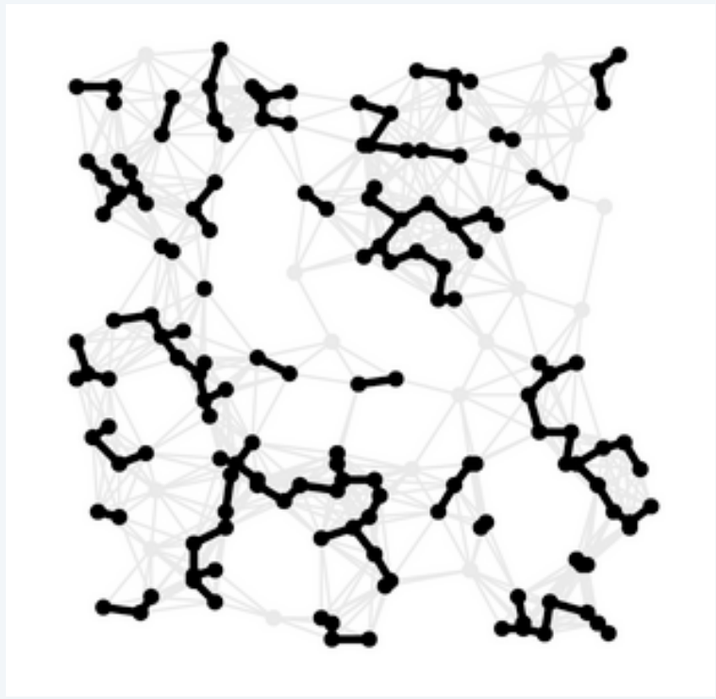
Goal. Given an integer k, find a k-clustering of maximum spacing.



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 $C_1^*, \ldots, C_k^*$ 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 $C_1, \ldots, C_k$.
- 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.
- Spacing of C^* = length d^* of the $(k-1)^{\text{st}}$ longest edge in MST.
- Edge (p, q) has length $\le d^*$ since it was added by Kruskal.
- Spacing of C is $\le d^*$ since p and q are in different clusters. ▪

this is the edge Kruskal would have added next had we not stopped it