Algorithm Design and Implementation

Principle of Algorithms V

Minimum Spanning Trees

Guoqiang Li

School of Software, Shanghai Jiao Tong University

Minimum Spanning Trees

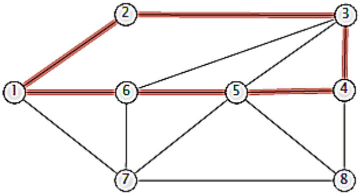
Cycles

Definition

A path is a sequence of edges which connects a sequence of nodes.

Definition

A cycle is a path with no repeated nodes or edges other than the starting and ending nodes.



path $P = \{(1,2),(2,3),(3,4),(4,5),(5,6)\}$

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

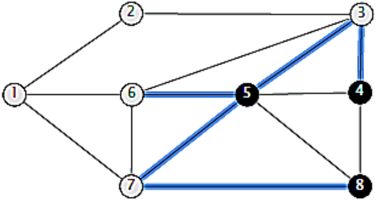
Cuts

Definition

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

Definition

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



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

Quiz 1

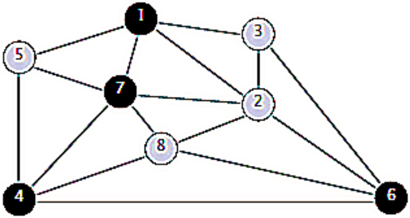
Consider the cut $S = \{1, 4, 6, 7\}$. Which edge is in the cutset of S?

A. S is not a cut (not connected)

B. 1-7.

C. 5-7.

D. 2-3.



Quiz 2

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

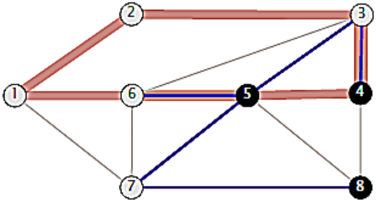
B. 2

C. not 1

D. an even number

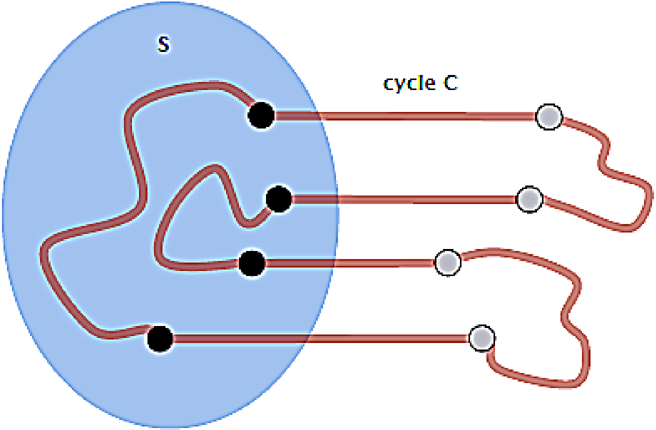
Cycle-cut intersection

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



$$
\begin{aligned}
\text{cycle } C &= \{(1,2),(2,3),(3,4),(4,5),(5,6),(6,1)\} \\
\text{cutset } D &= \{(3,4),(3,5),(5,6),(5,7),(8,7)\} \\
\text{intersection } C \cap D &= \{(3,4),(5,6)\}
\end{aligned}
$$

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



Spanning tree definition

Definition

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.

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.

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.

Minimum spanning tree (MST)

Definition

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.

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

Applications

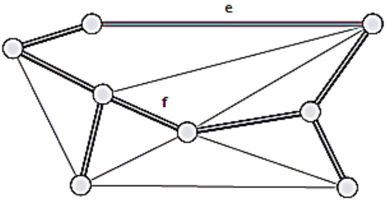
MST is fundamental problem with diverse applications.

- 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).

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.



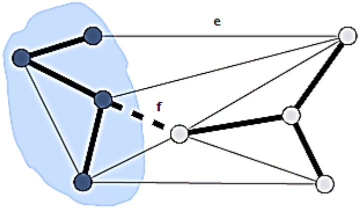
graph $G = (V, E)$
spanning tree $H = (V, T)$

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.

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.

Proof. [by induction on number of iterations]

Base case. No edges colored $\Longrightarrow$ 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.

Proof. [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
  - $e \in T^* \Rightarrow$ not red
  - blue rule $\Rightarrow$ e not blue and $c_e \geq c_f$
- Thus, $T^* \cup \{f\} - \{e\}$ satisfies invariant.

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

Proof. [by induction on number of iterations]

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

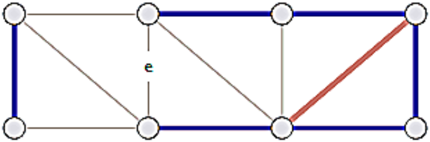
- let C be chosen cycle, and let e be edge colored red.
- if $e \notin T^*$, then T^* still satisfies invariant.
- Otherwise, consider fundamental cutset D by deleting e from T^*.
- let $f \in D$ be another edge in C.
- f is uncolored and $c_e \geq c_f$ since
  - $f \notin T^* \Rightarrow f$ not blue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

Theorem

The greedy algorithm terminates. Blue edges form an MST.

Proof. 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

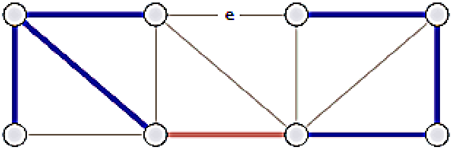
Greedy algorithm: proof of correctness

Theorem

The greedy algorithm terminates. Blue edges form an MST.

Proof. 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

Prim, Kruskal, Borůvka

Prim's algorithm

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

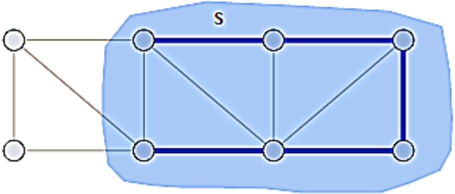
Repeat $n - 1$ times:

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

Theorem

Prim's algorithm computes an MST.

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



Prim's algorithm: implementation

```
Prim(V,E,c)
S ← ∅; T ← ∅;
s ← any node in V;
for each v ≠ s do π[v] ← ∞; pred[v] ← null; π[s] ← 0 ;
MakeQueue(pq);
for each v ∈ V do Insert(pq,v,π[v]) ;
while IsNotEmpty((pq)) do
    u ←DeleteMin(pq);
    S ← S ∪ {u}; T ← T ∪ {pred[u]};
    for each edge e = (u, v) ∈ E with v ∉ S do
        if c_e < π[v] then
            Decreasekey(pq,v,c_e);
            π[v] ← c_e; pred[v] ← e;
        end
    end
end
```

Prim's algorithm: analysis

Theorem

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

Proof.

By priority queue implementation.

Kruskal's algorithm

Consider edges in ascending order of cost:

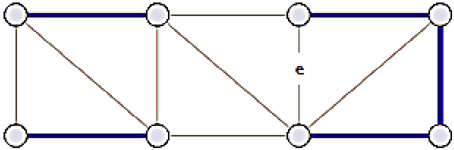
- Add to tree unless it would create a cycle.

Theorem

Kruskal's algorithm computes an MST.

Proof. 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.
- Case 2: both endpoints of e in different blue trees.
  $\Rightarrow$ color e blue by applying blue rule to cutset defined by either tree.



Kruskal's algorithm: implementation

```
Kruskal(V,E,c)
Sort m edges by cost and renumber so that
  c(e_1) ≤ c(e_2) ≤ ... ≤ c(e_m);
T ← ∅;
for each v ∈ V do  MakeSet(v);
for i = 1 to m do
    (u, v) ← e_i;
    if FindSet(u) ≠ FindSet(v) then
        T ← T ∪ {e_i};
        Union(u,v);
    end
end
Return T;
```

Kruskal's algorithm: analysis

Theorem

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

- Sort edges by cost.
- Use disjoint set data structure to dynamically maintain connected components.

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.

Proof. 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
- Case 2. [deleting edge e disconnects T]
  $\Rightarrow$ apply blue rule to cutset D induced by either component

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.

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.

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

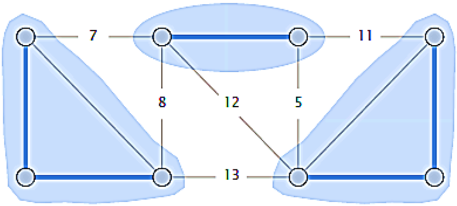
Repeat until only one tree.

- 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*

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



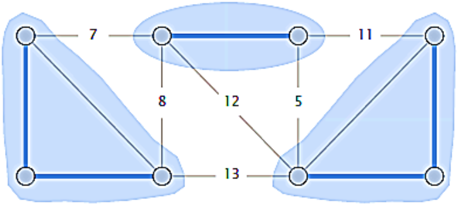
Borůvka's algorithm: implementation

Theorem

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

Proof.

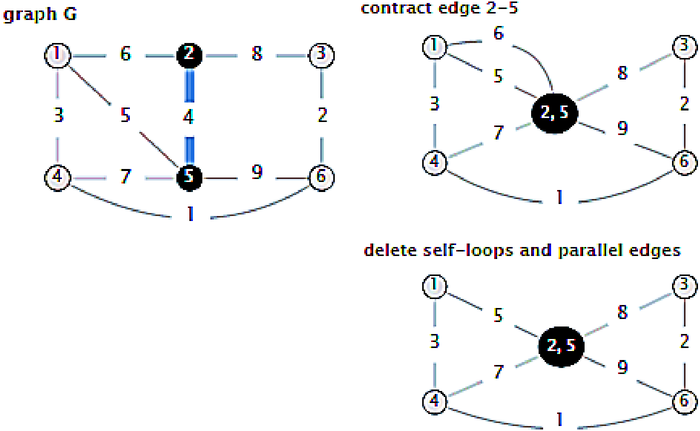
- To implement a phase in $O(m)$ time:
  - compute connected components of blue edges
  - for each edge $(u, v) \in E$, check if u and v are in different components; if so, update each component's best edge in cutset
- $\leq \log_2 n$ phases since each phase (at least) halves total # components.



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.

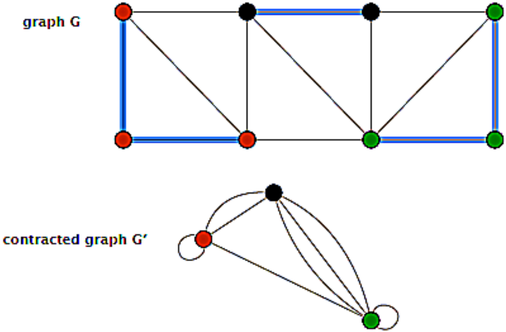


Q. How to contract a set of edges?

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.

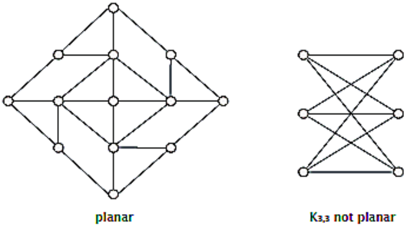
Goal. $O(m + n)$ time.



Borůvka's algorithm on planar graphs

Theorem

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



Proof.

- 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 $\leq cn + cn/2 + cn/4 + cn/8 + \cdots = O(n)$.

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.

Proof. Special case of the greedy algorithm.

Theorem

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

Proof.

- 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.

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

year	worst case	discoverec 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	$(m \log(\log^* n))$	Gabow-Galil-Spencer-Tarjan
1997	$O(m\alpha(n) \log \alpha(n))$	Chazelle
2000	$O(m\alpha(n))$	Chazelle
2002	asymptotically optimal	Pettie-Ramachandran
20xx	$O(m)$	???

deterministic compare-based MST algorithms

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
$(2^16, 2^5 536]$	5

Theorem (Fredman-Willard 1990)

$O(m)$ *in word RAM model.*

Theorem (Dixon-Rauch-Tarjan 1992)

$O(m)$ *MST verification algorithm.*

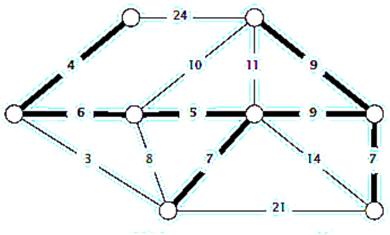
Theorem (Karger-Klein-Tarjan 1995)

$O(m)$ *randomized MST algorithm.*

Minimum bottleneck spanning tree

Problem. Given a connected graph G with positive edge costs, find a spanning tree that minimizes the most expensive edge.

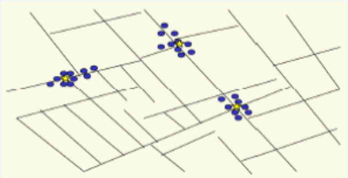
Goal. $O(m \log m)$ time or better.



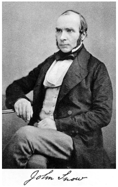
Single-Link Clustering

Clustering

Goal. Given a set U of n objects labeled $p_1, \cdots, p_n$, 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.
- Machine learning

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]

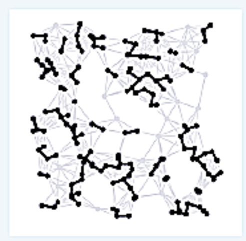
Spacing. Min distance between any pair of points in different clusters.

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.

Proof.

- Let C denote any other clustering $C_1, \dots, Ck$.
- 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 $(k-1)$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.

Quiz 5

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

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.