A Randomized Linear-Time (Expected) Algorithm for Minimum Spanning Trees

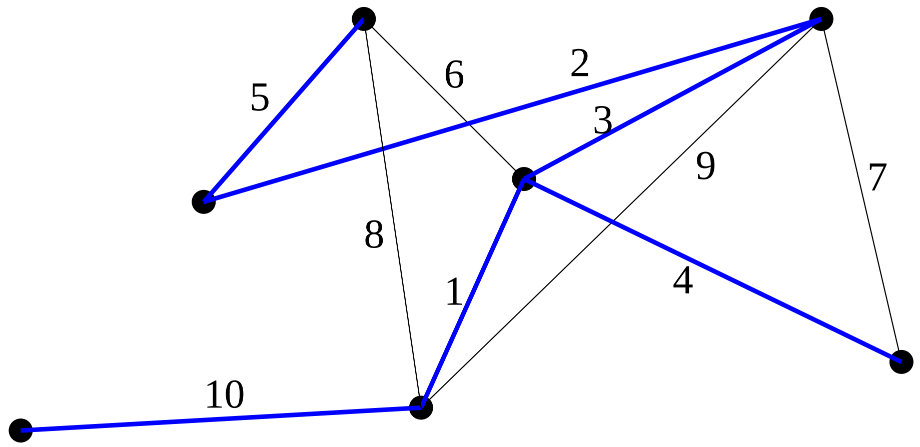
D. Karger, P. Klein, R. Tarjan

Presentation by:
Hal Burch (speech) and Mihai Budiu (slides)

Nov 10, 1998

Problem statement

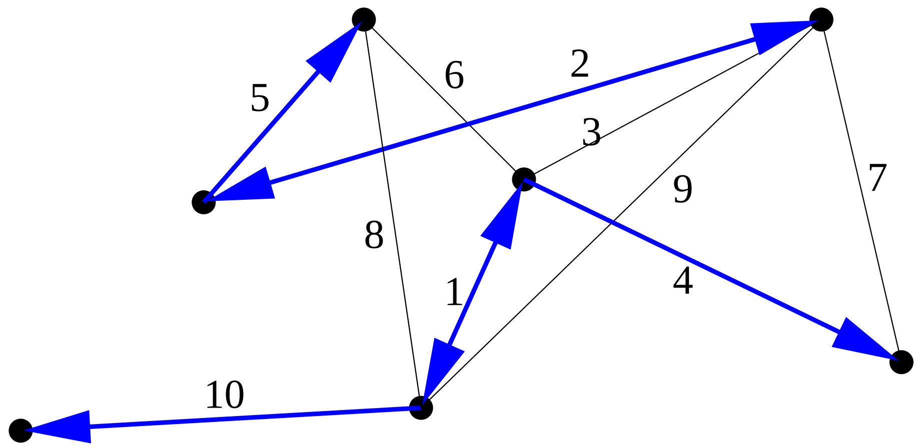
Given a connected undirected weighted graph $G = (V, E)$, find a connected set of edges MST with minimum total weight which spans all the nodes of the graph. $|V| = n, |E| = m$.



We can assume that edge weights are unique, by breaking ties in some arbitrary way. Then the MST is unique.

“Blue Edge” Property

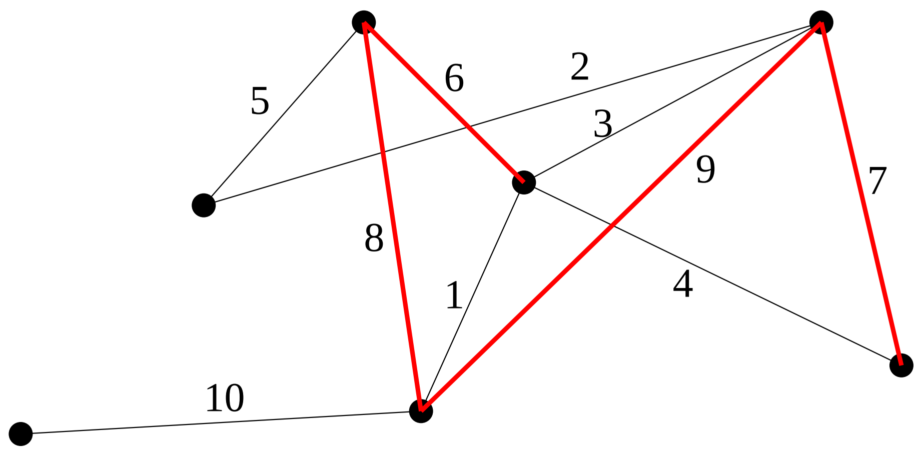
The minimum-weight edge incident to a node is always part of the MST.



The blue edges form a spanning forest of G.

“Red Edge Property”

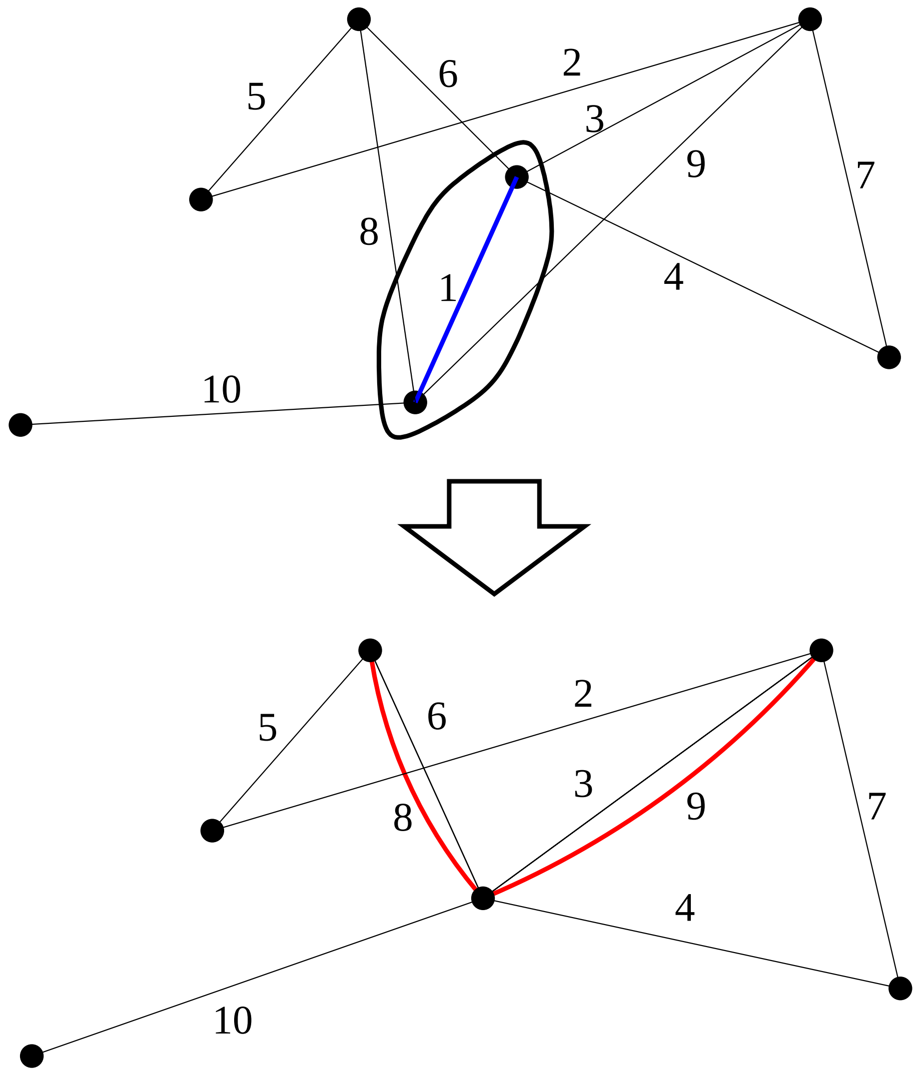
The MST never includes the heaviest edge in a cycle.



The classical algorithms do not use this property. KBT, by using this, achieves a good running time (expected linear).

A Borůvka Step

For all blue edges, “collapse” the two nodes at the ends of it. Delete self-loops, isolated vertices and multiple-edges (keep lightest).

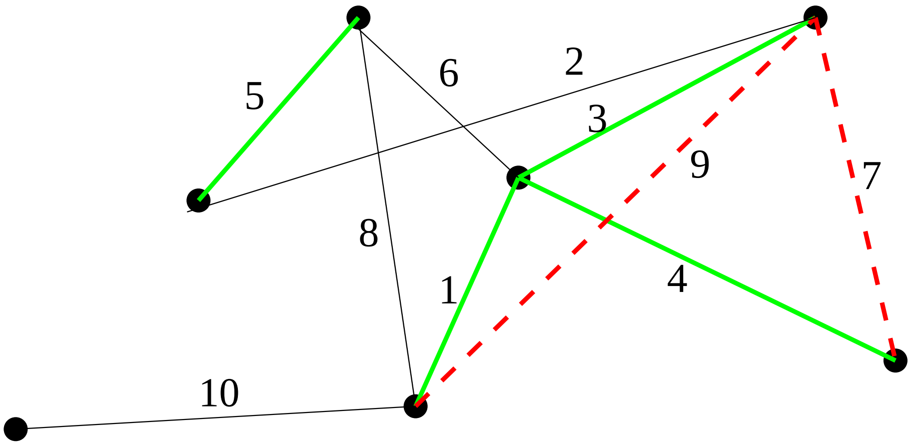


F-heavy and F-light edges

Given an arbitrary forest $F \subset G$ (thick, green edges in figure) of a graph G, an edge $e \in G$ is:

F-**heavy** if the path in F between its end-points contains only edges lighter than itself (if there's no path in F, the max weight is considered to be ∞); such edges are shown in dotted red lines.

F-**light** if not (thick green or thin black).



For any forest F, no F-heavy edge can be in the MST.

The Randomized Algorithm

Repeat the following until only one node is left:

1. Do three rounds of Borůvka steps. Let C be the set of collapsed edges.

2. Pick each remaining edge with probability 1/2; let the graph obtained be H.

3. On H run recursively the MST algorithm; obtain a forest F.

4. Find and delete all F-heavy edges from G.

5. Compute recursively T, the MST of the resulting graph, G. Return $T \cup C$.

□

Correctness

Proof:

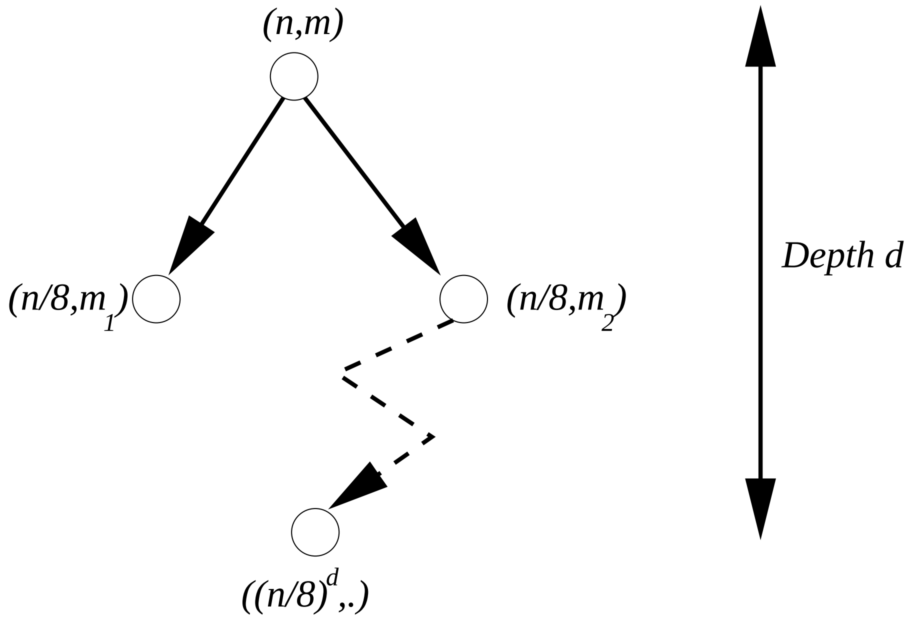
Each edge of C is blue, so it is part of the MST.

Each eliminated edge is F-heavy, so it is not in the MST.

T is part of the MST by induction.

The algorithm terminates, because each Borůvka reduction reduces the vertex set to 1/8-th of its initial size.$\square$

The Call Graph



$d \leq \log_8 n$.

Total vertices processed: $\sum_{d=0}^{\infty} 2^d n/8^d = O(n)$.

$m_1 + m_2 \leq m + |F| - 7n/8 \leq m$.

Timing Analysis (1): Worst Case

Theorem: The worst case running-time is $O(\min(n^2, m \log n))$, the same as the classical algorithm.

Proof: In the call graph, the time for each step is bounded by a constant times the number of edges. We estimate the total number of edges seen in the whole call-graph.

- At depth d each subproblem has $\leq n/8^d$ nodes and there are no multiple edges. Each subproblem thus deals with $< (n/8^d)^2/2$ edges. Therefore, the total number of edges processed $\leq \sum_{d=0}^{\infty} 2^d (n/8^d)^2/2 = O(n^2)$.

- At each depth d, the total number of edges processed in all subproblems is at most m. Thus there are no more than $m \log_8 n$ total edges processed.$\Box$

Timing Analysis (2): the Expected Size of H

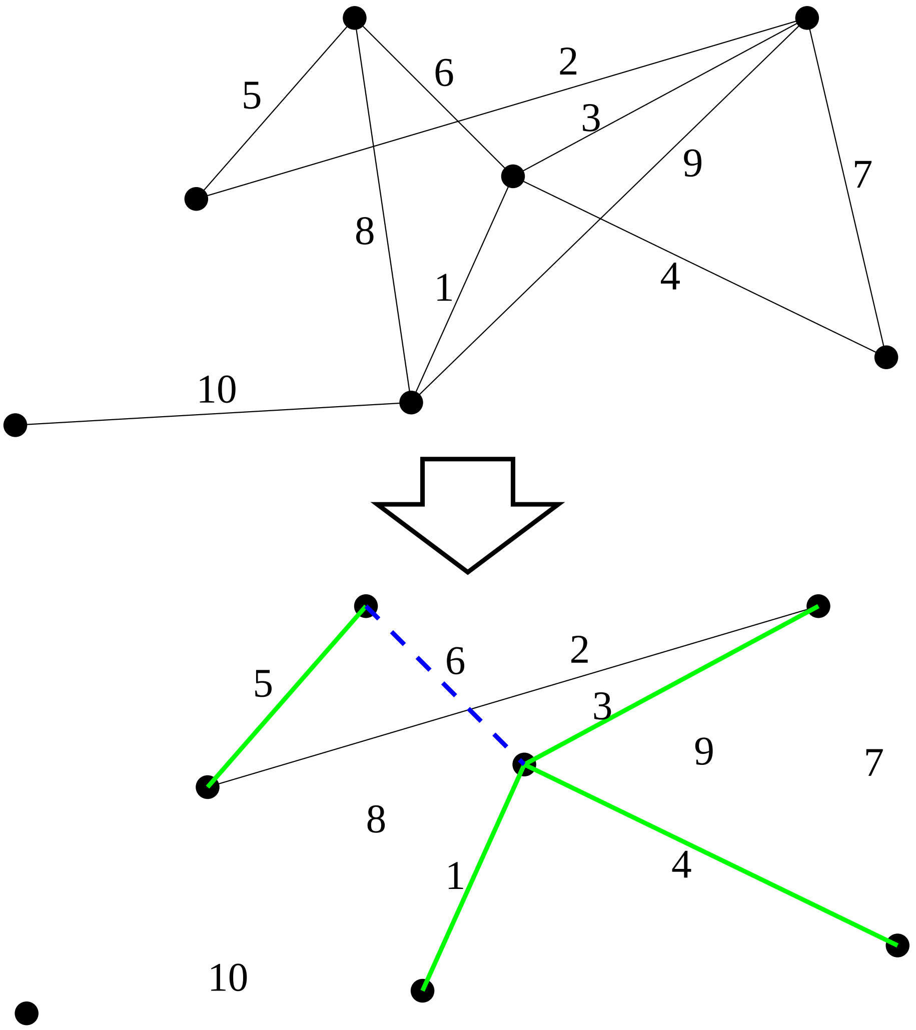
Sampling Lemma: Let H be a subgraph of G formed by selecting each edge with probability p, and let F be the MST of H. The expected number of F-light edges in G is at most n/p.

Proof: WLOG we can consider edges in G in increasing weight order (see figure on next slide). When edge e's turn comes, we have built a forest F'. Let L be the set of F-light edges.

If e is F'-heavy, it will be F-heavy. If e is F'-light, it will be F-light.

Let us assume e is F'-light. With probability p, we add e to F'. In the end $|F| < n$. Each $e \in L$ belongs to F with probability p. So $\mathbf{E}[|F|] = \mathbf{E}[|L|]p$. It follows $\mathbf{E}[|L|] > n/p$.$\square$

The Sampling Lemma Illustrated



This shows the graph after the 5 lightest edges have been sampled. The thick green edges are in H (and in F), the thin ones aren't. When 6 is considered it is F-light, so we add it to F with probability 1/2. 7 though will be F-heavy.

Timing Analysis (3): a Quick Estimation

Theorem: The expected running time is $O(n+m)$.

Proof: All Borůvka steps take linear amortized time.

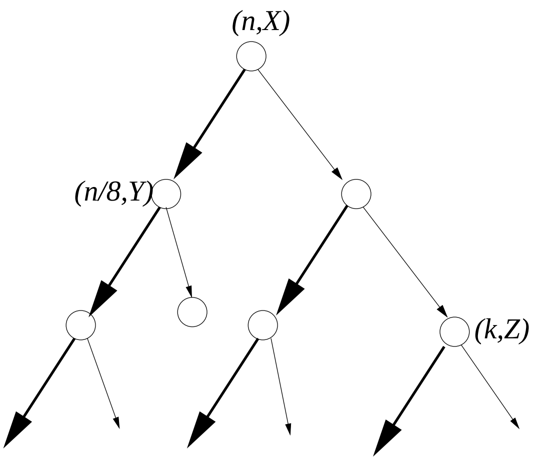
The sampling step can be done in linear time too. The expected result graph size is $n/8, m/2$.

The algorithm finding F-heavy edges can be carried in linear time. The resulting graph size is $n/8, n/4$.

$T(n,m) \leq T(n/8, m/2) + T(n/8, n/4) + c(n+m)$ with solution $2c(n+m)$.$\square$

Timing Analysis: a Refined Analysis

Theorem: The expected running time is $O(m)$.



Proof: In the figure X, Y, Z are r.v. $\mathbf{E}[Y|X = k] \leq k/2$ by sampling. By linearity we have $\mathbf{E}[Y] \leq \mathbf{E}[X]/2$.

On a left-path from a call with k edges, $\sum_{d=0}^{\infty} k/2^d = 2k$ expected edges are visited.

By sampling lemma a right-child has $\mathbf{E}[Z] \leq k/2$ expected edges. Total expected edges seen in the rightmost path: $\sum_{d=1}^{\infty} 2^{d-1} n/8^d/2 = O(n)$. Total complexity is thus $O(m + 2n)$.$\Box$

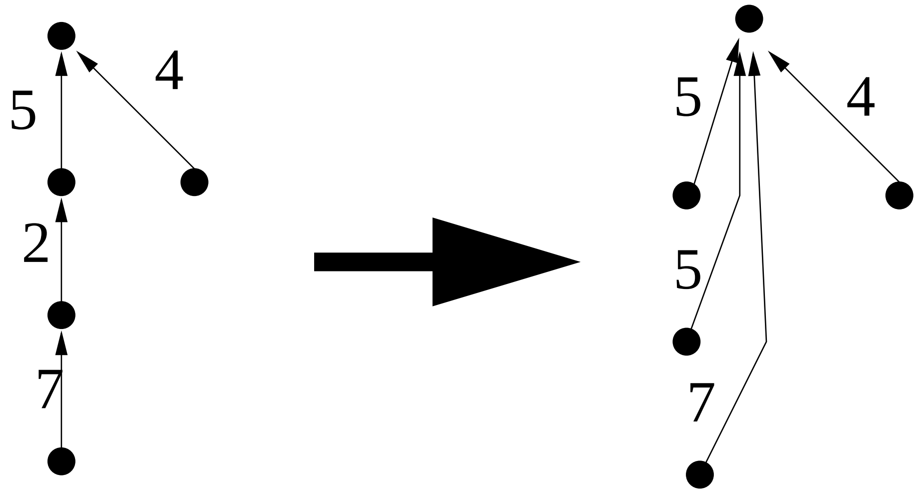
The Quality of the Solution

Theorem: The algorithm runs in $O(m)$ time with probability $1 - e^{-\Omega(m)}$.

Proof: Using Chernoff bounds.$\square$

Finding F-heavy edges

This can be achieved in linear time. The algorithm is rather intricate. We will present Tarjan's $O(n + m\alpha(m, n))$ algorithm, which is used by Dixon, Rauch, and Tarjan's linear algorithm. The algorithm is based on a slight adaptation of Tarjan's Union-Find data structure:



Basic Algorithm

1. Perform a depth-first traversal of each component of F.

2. At each node, first recurse on each of its children. For each $e \in G$ adjacent to the current node, if the other endpoint of e has not been traversed, then ignore e. Otherwise, find the least common ancestor of the endpoints of e by a find of Tarjan's Union-Find structure and "associate" e with this LCA.

3. Union the node with its parent, by an edge of weight equal to the weight of the edge between it and its parent.

Detecting F-Heaviness

The "associated" edges of each node are those edges whose LCA of the endpoints are that node. During the traversal of the LCA, the adaptation of Union-Find gives the maximum weight edge from the two endpoints to this LCA, which can be used to determine whether that edge is F-heavy or F-light.

Reading List

O Borůvka, "O jistém problému minimálním", *Práca Moravské Přírodovědcké Společnosti* 3, 1926.

David R. Karger, Philip N. Klein, Robert E. Tarjan, "A Randomized Linear-Time Algorithm to Find Minimum Spanning Trees" *Journal of the ACM*, 1995.

V. King "A Simpler Minimum Spanning Tree Verification Algorithm", *Algorithmica* 18, 1997: 263-270.

R. Cole, P. N. Klein, R. E. Tarjan "A Linear-Work Parallel Algorithm for Finding Minimum Spanning Trees", *Proc. 6th Symposium on Parallel Algorithms and Architectures*, 1994.