Goal: An O(m)-time algorithm

without bit manipulation of edge weights

Boruvka's algorithm with contraction:

If G contains at least two vertices:

select cheapest edge incident to each vertex;

Contract all selected edges;

Recur on contracted graph.

If contraction preserves sparsity ($m = O(n)$),

this algorithm runs in $O(n) = O(m)$ time

on sparse graphs.

E.g. planar graphs

How to handle non-sparse graphs?

Thinning: remove all but O(n) edges by finding edges that can't be in the minimum spanning tree.

How to thin?

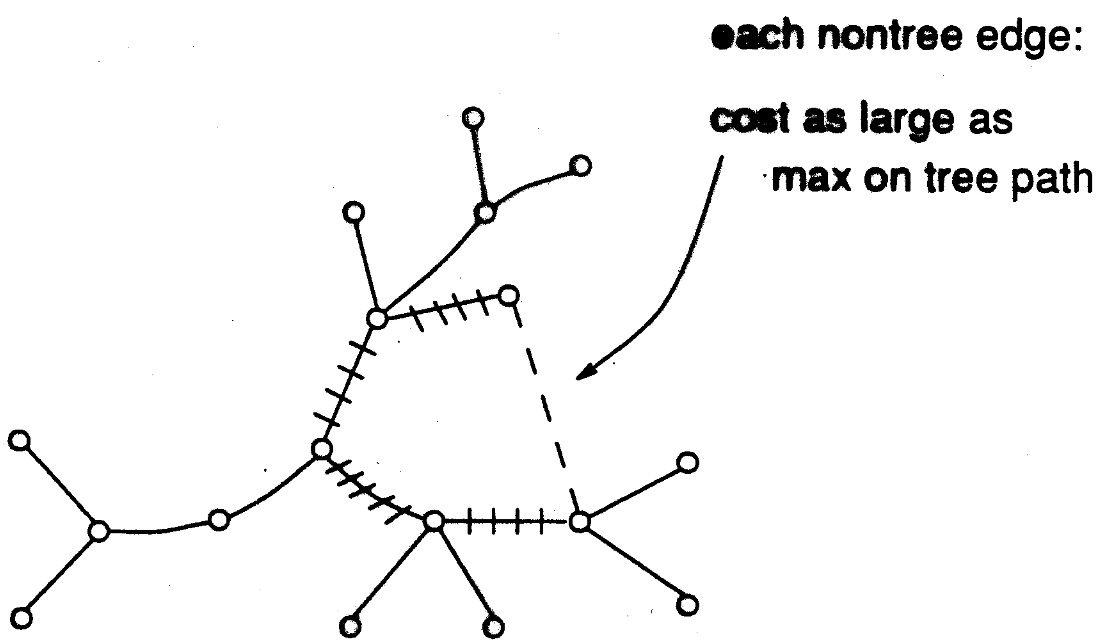
Verification:

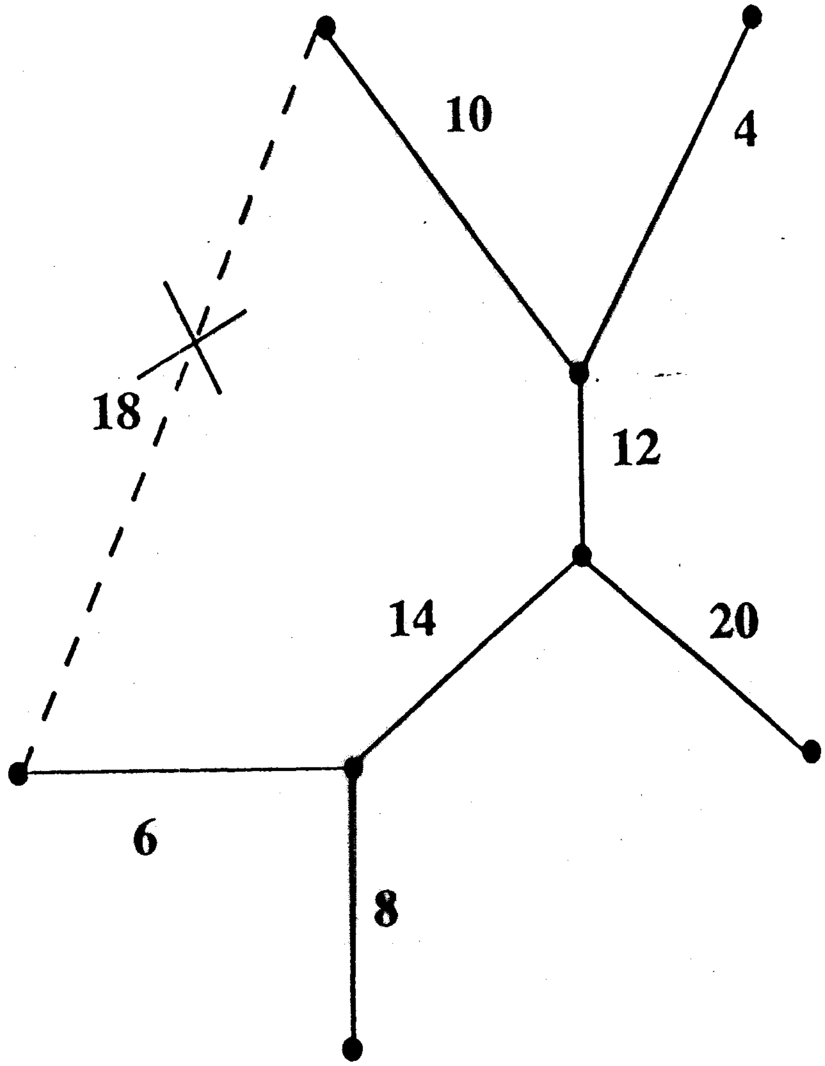
Given a spanning tree, is it minimum?

Thinning: Given a spanning tree, delete any non-tree edge larger than every edge on tree path joining its ends (red rule).

If all non-tree edges can be thinned, tree is verified.

Verification





10
4
18
12
14
20
6
8

History of Verification Algorithms

Tarjan, 1979	$O(m\,\alpha\,(m,n))$ time
Komlos, 1984	$O(m)$ comparisons
Dixon, Rauch, Tarjan, 1992	$O(m)$ time
King, 1993	$O(m)$ time (simplified)

All these algorithms will thin.

Thinning by Random Sampling (1993)

Select half the edges at random.

Build a minimum spanning forest of the sample.

Thin.

How many edges remain?

Karger:	O(nlogn) on average
Klein, Tarjan:	< 2n on average

Minimum Spanning Forest Algorithm

If # edges/ # vertices < 5, then

(Boruvka step) Select the cheapest edge incident to each vertex.

Contract all selected edges.

Recur on contracted graph.

Else

(Sampling and Thinning Step) Sample the edges, each with probability 1/2.

Construct a minimum spanning forest of the sample, recursively.

Thin using this forest.

Recur on Thinned Graph

Analysis

Boruvka step

$m < 5n$ implies $m' < 9m/10$ since at least

$n/2$ edges are contracted

$T(m) = O(m) + T(9m/10)$

Thinning Step

$m > 5n$ implies $2n < 2m/5$

$T(m) = O(m) + T(m/2) + T(2m/5)$

where $T(m/2)$ and $T(2m/5)$ are expected time

$T(m) = O(m)$ by induction

Bound on Number of Edges Not Thinned

Let $e_1, e_2, \ldots, e_m$ be the edges, in increasing cost.

Run the following variant of Kruskal's algorithm.

Inititalize $F = \emptyset$.

Process the edges in order.

To process e_i, flip a coin to see if e_i is in the sample.

If e_i forms a cycle with edges in F, discard it as thinned.

Otherwise, if e_i is sampled, add e_i to F. (Whether or not e_i is sampled, it is not thinned.)

F is the minimum spanning forest of the sample.

How many edges are not thinned?

The only relevant coin flips are those on unthinned edges, each of which has a chance of 1/2 of adding an edge to F (a success).

There can be at most n-1 successes.

For there to be more than k unthinned edges, the first k relevant coin flips must give at most n-2 successes.

The chance of this is at most

$$\left(\frac{1}{2}\right)^k \sum_{i=0}^{n-2} \binom{k}{i} < \left(\frac{1}{2}\right)^k \sum_{i=0}^{n} \binom{k}{i}$$

In particular, the average number of unthinned edges is at most 2n.

Open Problems

Deterministic O(m)?

Simpler verification?

Other applications?

- **directed spanning trees?**
- **shortest paths?**