# Minimum Spanning Trees II

Recall: Boruvka's algo:

- Start with no blue edges (edges added to MST)
  Every single vertex is a blue tree
- Repeat
  - for each blue tree, find the min weight outgoing edge
  - color all such edges blue

  until all blue edges form a single tree

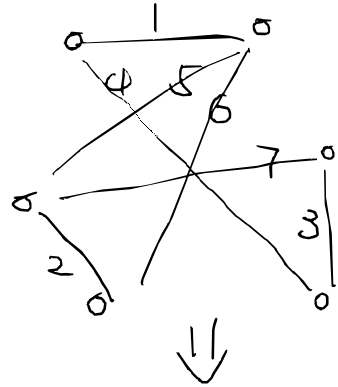
$m := \#$ edges, $n := \#$ vertices

Boruvka's algorithm runs in $O(m \log n)$ time.

- finding the min weight outgoing edges & merging the blue trees & relabeling for each vertex, which blue tree it belongs to takes $O(m)$ time
- # blue trees reduces by a factor of $\geq 2$ each round, at most $O(\log n)$ rounds.

A "heuristic" improvement: **Cleanup**

- Suppose there are multiple edges between the same pair of blue trees.
  It suffices to only keep the cheapest.
- Equivalently, we can contract each blue tree into a supernode, and remove duplicated edges.




May remove some edges so that future rounds can be faster!

Best hope: cleanup reduces

$m \to m/2 \to m/4 \to \cdots$

then running time would be (not always true)

$m + m/2 + \cdots = O(m)$ instead of

$m + m + \cdots = O(m \log n)$

Karger, Klein, Tarjan: randomized MST algo with expected $O(m)$ time.

Idea: use MST (or MS forest) of a random subgraph to "prune" more edges in each round.

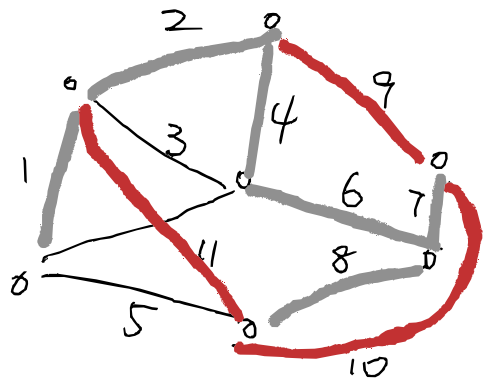
Suppose we are given a spanning tree $T$ (not necessarily min)

- if for any edge $(u,v)$, all edges on path $u \to v$ in $T$ have smaller weight than $(u,v)$,

then $(u,v)$ can be colored red (pruned).

Ideally, if $T$ is MST,
- can color all other edges red (done!)

KKT: find a good $T$ that is cheaper to compute.



# Randomized MST algo.

Repeat

If #edges > $c \cdot$ #(super)nodes  (c is a const, e.g., c=5 works)

Do a "thinning" step

Run one Boruvka round.

Cleanup by contracting blue trees

## Thinning step:

Sample each edge in the graph with prob. 1/2,

Compute the MSF $T'$ of this random subgraph recursively.

Remove edges $(u,v)$ s.t.

$u$ and $v$ are connected in $T'$

$(u,v)$ more expensive than all edges on path $u \to v$ in $T'$.

→ can be done in $O(m)$ time. nontrivial. omit in this lecture.

**Lemma:** A thinning step reduces #edges to $\leq 2 \cdot$ #supernodes in expectation.

Proof. $T'$ is built:

- sample each edge w.p. $1/2$
- add sampled edges in increasing order by weights to $T'$, as long as no cycle is created.

This is equivalent to:

- process ALL edges in inc order by weights,
  - if the edge does not create a cycle, then add it to $T'$ w.p. $1/2$.

only these edges remain.

However, only #supernodes many edges can be added to $T'$.

$\mathbb{E}[\#\text{coin tosses}] \leq 2 \cdot \#\text{supernodes}$. □

Thm: Expected running time is $O(m)$.

Proof. Let $b := \#(\text{super})\text{nodes}$

$e := \#(\text{remaining})\text{edges}$

$R(e)$ be running time on input w/ $e$ edges.

If $e \le c \cdot b$, (no thinning) ← add $\ge b/2$ blue edges & contract

$$R(e) \le O(e) + R(e - b/2)$$

$$\le O(e) + R((1 - 1/2c) \cdot e).$$

$$\le O(e) + R(0.9e) \qquad (c = 5)$$

If $e > c \cdot b$.

$$R(e) \le O(e) + R(e/2) + R(2 \cdot b)$$

$$\le O(e) + R(e/2) + R(\tfrac{2}{c} \cdot e)$$

$$\le O(e) + R(e/2) + R(0.4e) \qquad (c = 5)$$

Can inductively prove:

$R(e) \le C' \cdot e$ for a large const $C'$.

□