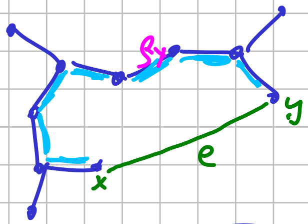
weighted graph $w: E \to \mathbb{R}$ WS06 injective

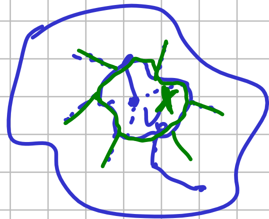
Minimum Spanning Tree : min $w(T)$

$T[x,y]$ $T[e]$

e is T-light $\equiv \exists f \in T[e]: w(f) > w(e)$

if $\exists e$ T-light $\Rightarrow$ T is not minimum (by exchange)
$\Leftarrow$





193x

Jarník's algorithm
(Dijkstra's ...)

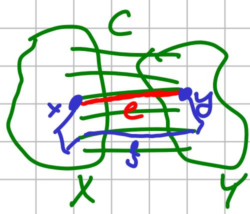
1. $T \leftarrow \{v_0\}$
2. While $|T| < n$:
3. Select lightest $uv \in E$ s.t. $u \in T$, $v \notin T$
4. $T \leftarrow T \cup \{e\}$

If G is connected, J.a. finds a spanning tree.

Cut Lemma (Blue lemma):

Let $C = E(X,Y)$ be an elementary cut,
$e \in C$ lightest edge of the cut
T an arbitrary MST.
Then $e \in T$.

Proof: Assume the contrary.
Let $f \in T[e] \cap C$.
$T' := T - f + e$ is another ST
& $w(f) > w(e) \Rightarrow w(T') < w(T)$ ↯



Thm: J.a. finds a MST.

Proof: By cut lemma, every edge added to T is in all MSTs.
So $T \subseteq$ every MST.
Since all spanning trees have $n-1$ edges,
T = every MST.

Uniqueness

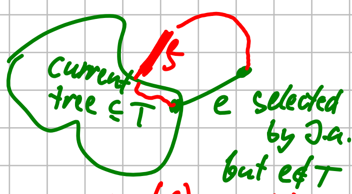
Thm: The MST is unique.

Thm: T is the MST $\Leftrightarrow$ there are no T-light edges.
$\Rightarrow$ ✓

↳ values of weights don't matter, only the order

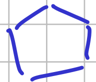
↳ we just use an edge comparison oracle (running in const. time)

if there are no T-light edges, then J.a. outputs T.
If not, stop J.a. at the first moment it selects an edge $e \notin T$:



$w(f) > w(e)$
$\Rightarrow$ e is T-light! ↯

Non-unique weights: multiple MSTs:



$w=1$

↳ "$\leq$" is not a linear order
→ break ties arbitrarily, e.g., compare $(w(e), id(e))$ lexicographically

---

Red-Blue (Meta) Algorithm

1. All edges uncolored.
2. Repeat as long as possible:
3. either: Find e lightest in some elementary cut & not blue and color it blue.] blue rule
   or: Find e heaviest on some cycle & not red and color it red.] red rule

Blue lemma: blue $\in$ MST

Red lemma: red $\notin$ MST

So no edge is re-colored.

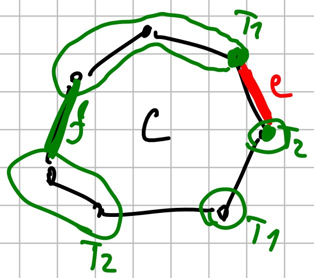
So the alg. stops after $\leq m$ steps.

Rainbow lemma: when the alg. stops, all edges are colored.

So blue edges $=$ MST.

Red lemma: If e is heaviest on some cycle C, then $e \notin$ MST.

Proof: By contradiction … $e \in T$, T is the MST



$T - e$ has two components T_1, T_2

There is some other $f \in C$ with one vertex in T_1 and the other in T_2.

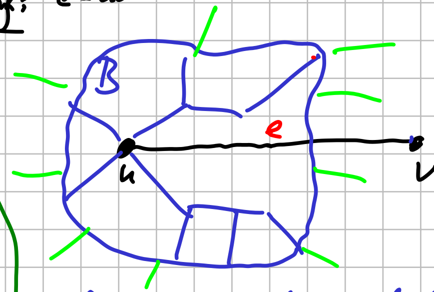
So $T' := T - e + f$ is again a ST

But $w(T') < w(T)$ ↯

Rainbow lemma:

If $\exists e \in E$ uncolored, the alg. can continue.

Proof: $e = uv$



$B :=$ Subgraph reachable from u by blue edges

If $v \in B$: red rule on e

Else: consider cut $E(B, \bar{B})$

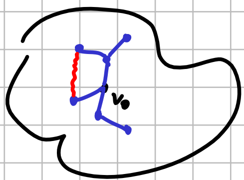
cut $\neq \emptyset$ because $e \in$ cut

cut has no blue edges

blue rule on this cut ∎

Classical MST algs

① Jarník grows a blue tree

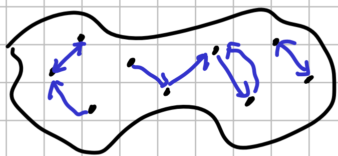


basic: $O(n \cdot m)$

with a heap: $O(m \log n)$

② Borůvka (192x)

blue forest



in every step, every tree select the lightest incident edge

\# steps $\leq \log n$

(in a step, # trees decreases at least twice)

time $O(m \cdot \log n)$

③ Kruskal's Alg.

Sort edges by weight: $w(e_1) < w(e_2) < \ldots < w(e_m)$

$T \leftarrow \emptyset$

For $i = 1 \ldots m$:

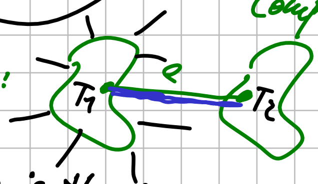
If $T + e_i$ is acyclic: $T \leftarrow T + e_i$

(Find — m times; Union — n times)

endpoints of e_i lie in different components of T

adding edge:

Comp. of T



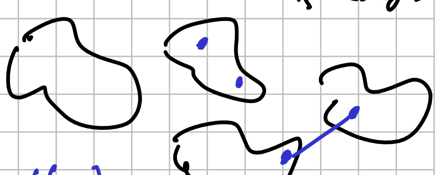
e is the first edge of the cut considered

$\Rightarrow$ e is lightest of cut

naive: $O(m \cdot n)$

better: use Union-Find data structure

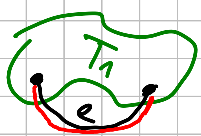
D.S. for maintaining connected components under insertion of edges



Find(x,y): are x,y in the same component?

Union(x,y): adds xy to E

dropping edge:

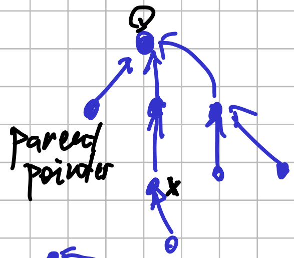


e is considered after all of T_1

$\Rightarrow$ e is heaviest on the cycle in $T_1 + e$

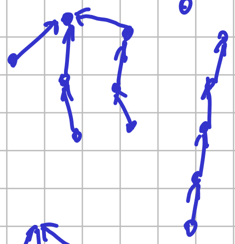
Union-Find: Represent each component by a shrub (rooted tree oriented towards the root)

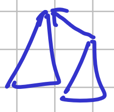
$V(\text{shrub}) \Leftrightarrow V(\text{component})$



Find(x,y): Locates roots of shrubs & compares them.

Union(x,y): Locates the roots x', y'
If $x' \neq y'$: add edge between x', y' arbitrarily



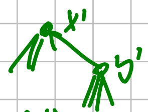


$2^r \quad 2^r$

$2^r + 2^r = 2^{r+1}$

Union by Rank

rank : $V \to \mathbb{N}$
at the start: rank(*) ← 0
In Union:



if $r(x') > r(y')$:
make x' parent of y'
& keep the ranks

If $r(x') = r(y')$:
Choose arbitrarily,
$r(\text{new root})$ ++

Ranks of roots = shrub heights

A shrub with root of rank r contains at least 2^r vertices

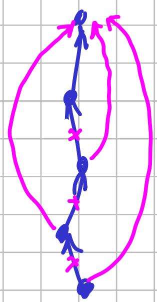
↳ ranks ≤ log n

→ heights ≤ log n

guarantees U & F in $O(\log n)$ time

Kruskal's alg. runs in $O(m \log n)$ time

Path Compression



Claims: ① P.C. without U. by R. → $O(\log n)$ amort. time per op.

② both P.C. & U. by R. → $O(\alpha(n)) \leq O(\log^* n)$ amort.

↑ the inverse Ackermann function

for path of length ℓ: $O(\ell)$ to find the root
$O(\ell)$ to compress the path

$2 \uparrow k := \left.2^{2^{2^{\cdot^{\cdot^{2}}}}}\right\}k$

$2 \uparrow 1 = 2$

$2 \uparrow (k+1) = 2^{2 \uparrow k}$

$\log^* n := \min k : 2 \uparrow k \geq n$