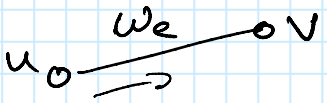
Minimum Spanning Tree (MST)

weighted edges

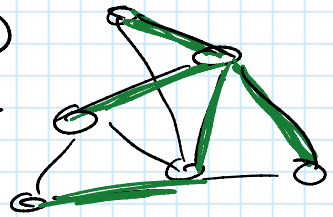
edge e : weight w_e



graph $G=(V, E, w)$
spanning tree
with min. total weight

for simplicity: weights unique (distinct)

↳ MST unique

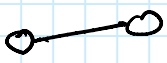


Goal: Constr. MST using a distr. algorithm

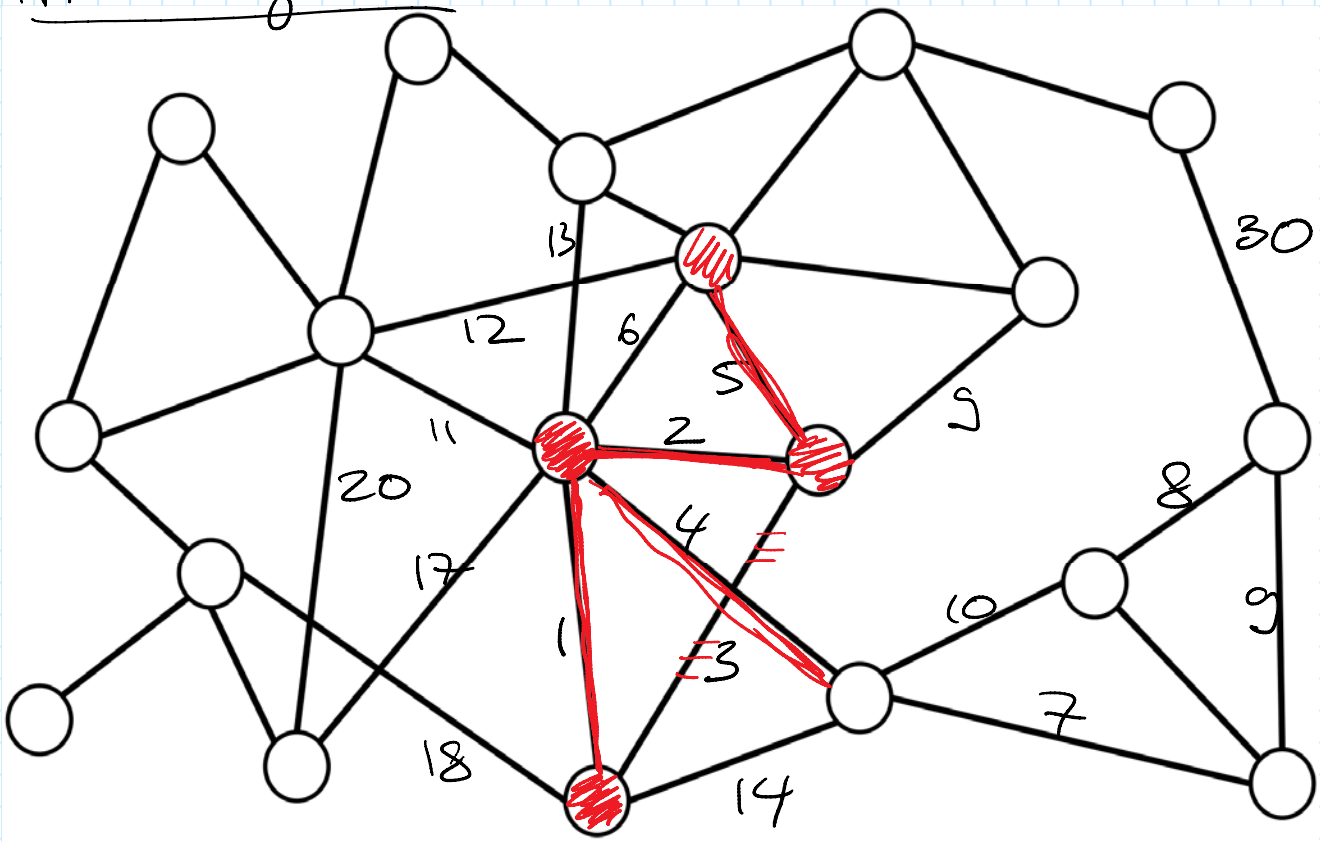
Sequentially: greedy!

I) start at some node u
grow tree from u

⟹ always add min. weight edge
⟹ make sure that we do not close cycles



Prim's Algorithm



$e_1, \ldots, e_{i-1}$

T is not an MST

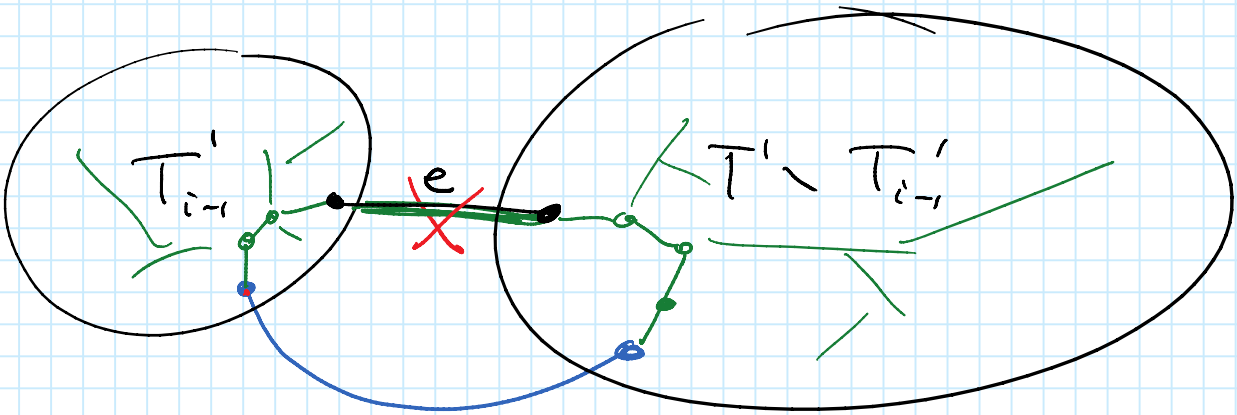
$\Rightarrow \exists$ min. i s.t. no MST that contains edges $e_1, \ldots, e_i$

MST T' : T' contains $e_1, \ldots, e_{i-1}$

T' doesn't contain e_i

T'_{i-1} : subtree of T' induced by $e_1, \ldots, e_{i-1}$

T' $\quad e \neq e_i$



e_i $\quad w_{e_i} \leq w_e$

$<$ (distinct weights)

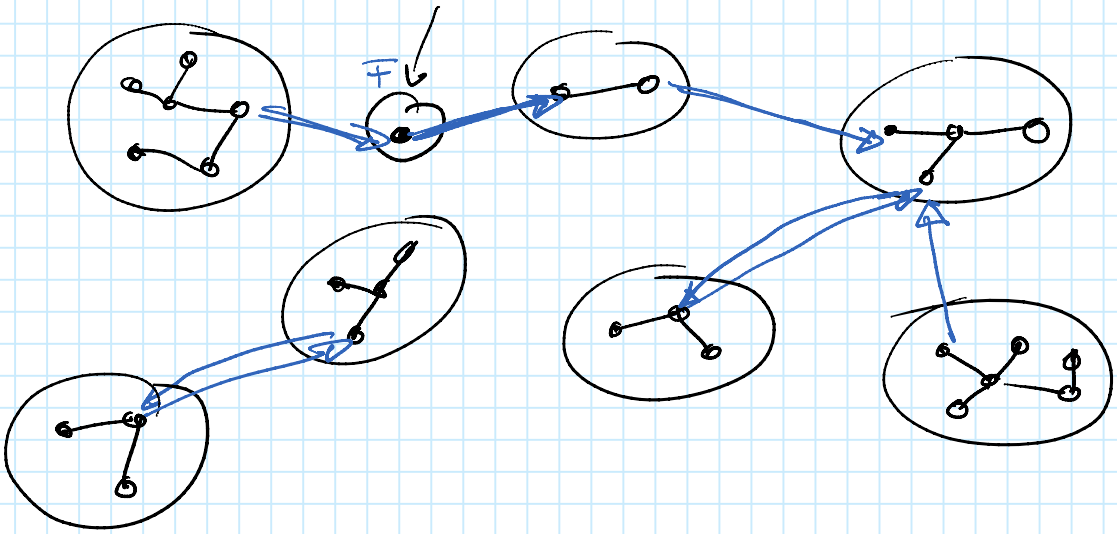
Kruskal's Algorithm

always pick the edge with min. weight which doesn't close a cycle

$\Rightarrow$ also works ...

Distributed MST Algorithm

fragments



Fragment F

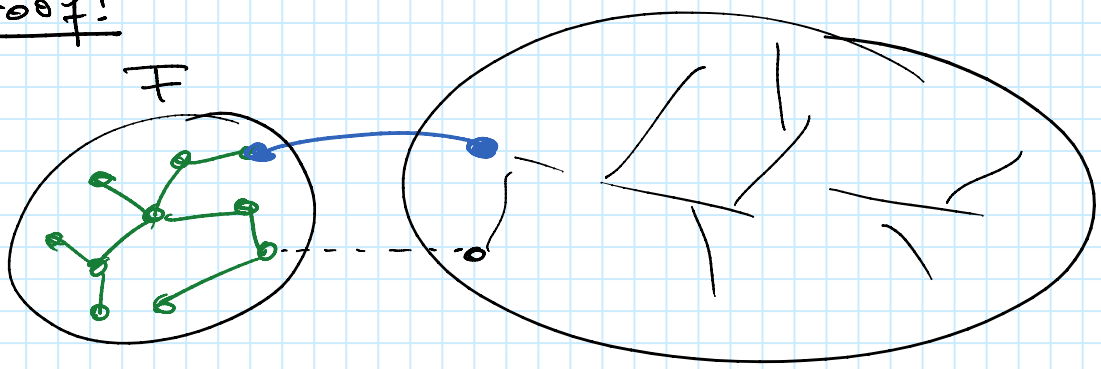
blue edge of F

min. weight edge connecting F with $V \setminus F$ (some other fragment)

Claim:

$\forall$ fragm. F, the blue edge of F is in MST

Proof:



□

GHS Algorithm

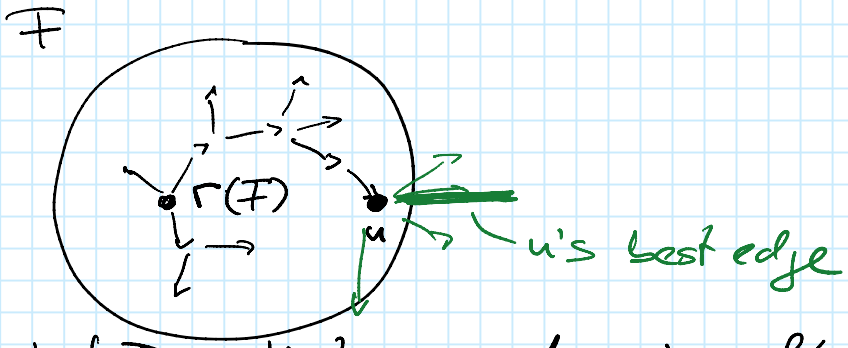
Gallager, Humblet, Spira

Fragment F:

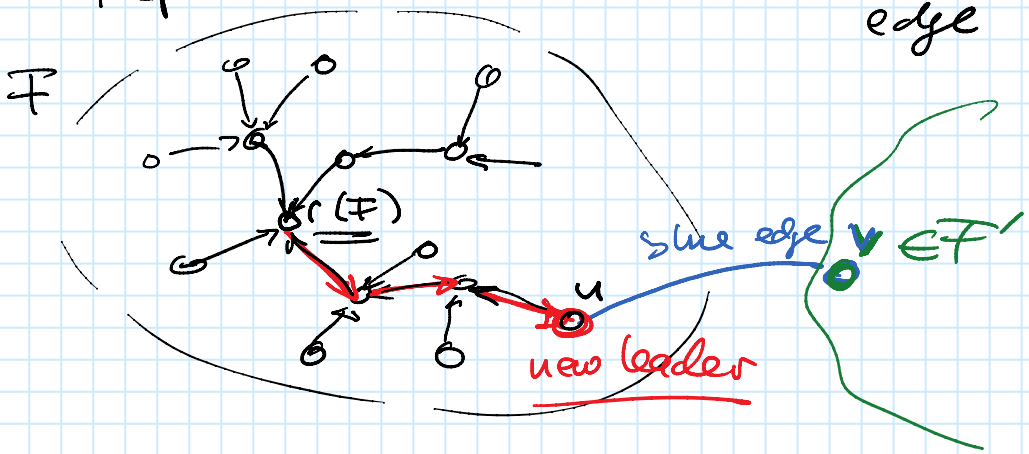
- rooted tree
- ID of root = ID of fragment F
- every node of F know ID of F
- root: leader of fragment

---

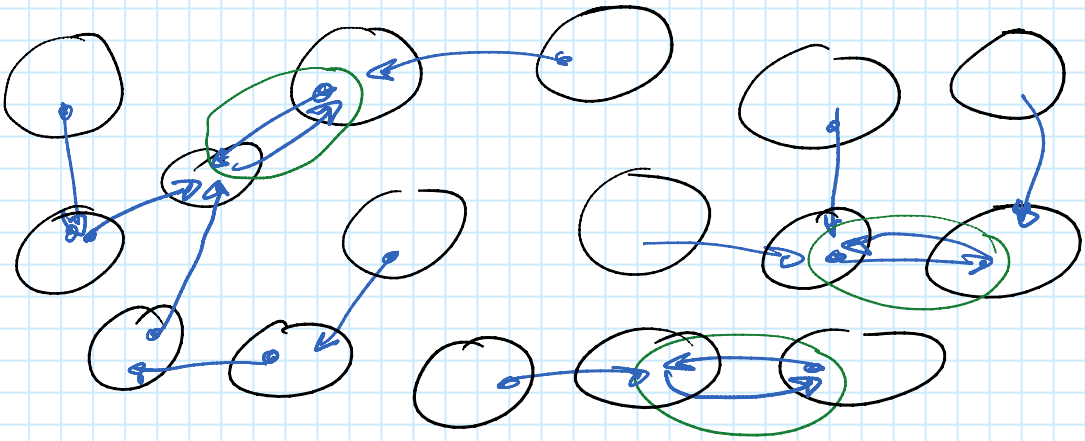
1. Find blue edges

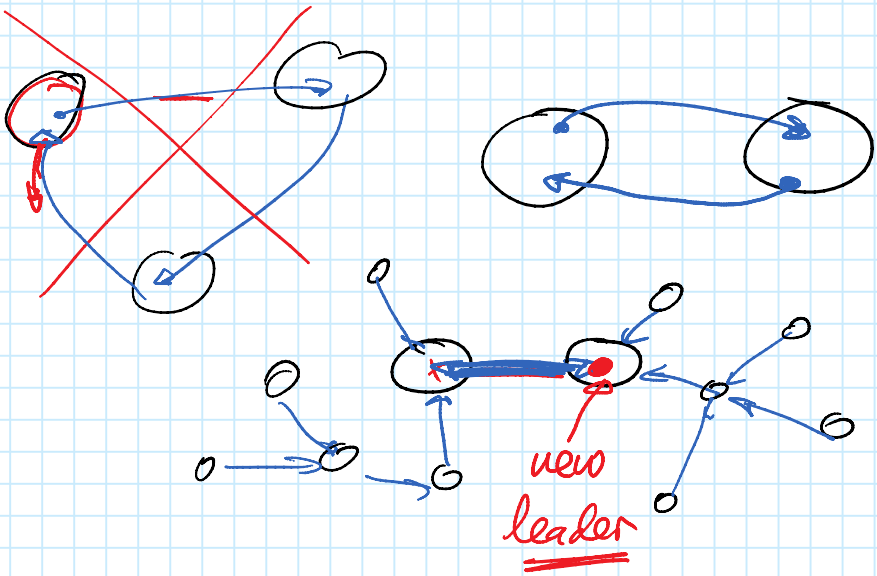


root of F initiates search using flooding
nodes check neighborhood
nodes report back using convergecast (echo)
(assume u has the blue edge)
root reports to u that it has the blue edge



2. Connect Fragments
(add blue edges)





Algorithm 3.5 GHS (Gallager–Humblet–Spira)

1: Initially each node is the root of its own fragment. We proceed in phases:
2: **repeat**
3: All nodes learn the fragment IDs of their neighbors. ← $O(1)$
4: The root of each fragment uses flooding/echo in its fragment to determine the blue edge $b = (u, v)$ of the fragment. $O(n)$
5: The root sends a message to node u; while forwarding the message on the path from the root to node u all parent-child relations are inverted {such that u is the new temporary root of the fragment} $O(n)$
6: node u sends a merge request over the blue edge $b = (u, v)$. $O(1)$
7: **if** node v also sent a merge request over the same blue edge $b = (v, u)$ **then**
8: either u or v (whichever has the smaller ID) is the new fragment root
9: the blue edge b is directed accordingly
10: **else**
11: node v is the new parent of node u
12: **end if**
13: the newly elected root node informs all nodes in its fragment (again using flooding/echo) about its identity } $O(n)$
14: **until** all nodes are in the same fragment (i.e., there is no outgoing edge)

GHS:
time: $O(n \log n)$
msg: $O(m + n \log n)$

Time Complexity:
$O(\log n)$ phases
1 phase: $O(n)$
total time compl. $O(n \log n)$

Awerbuch: $O(n)$

Kutten, Peleg: $O(D + \sqrt{n} \cdot \log^* n)$ ⟸

$\Omega(D + \sqrt{\frac{n}{\log n}})$

Message Complexity:
1 phase:
- floodings/echos: $O(n)$
- exploring neighborhood: $O(m)$

overall: $O(m \log n)$