29 Oct 2021

UNION FIND (for a partitioned set)

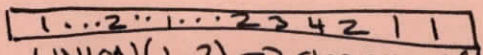
→ init w/ a set of sets

→ after that, we allow 2 operations:

1) UNION - merge 2 sets
2) FIND - find the "id" of a set

First 2 Data Structures

① Fast Find



UNION(1, 2) ⇒ change all 1's to 2's

→ union: must update all ets of the set $\Theta(n)$

→ find: $\Theta(1)$

② Fast Union

→ each conn. comp. is a rooted tree. Root node is the id.

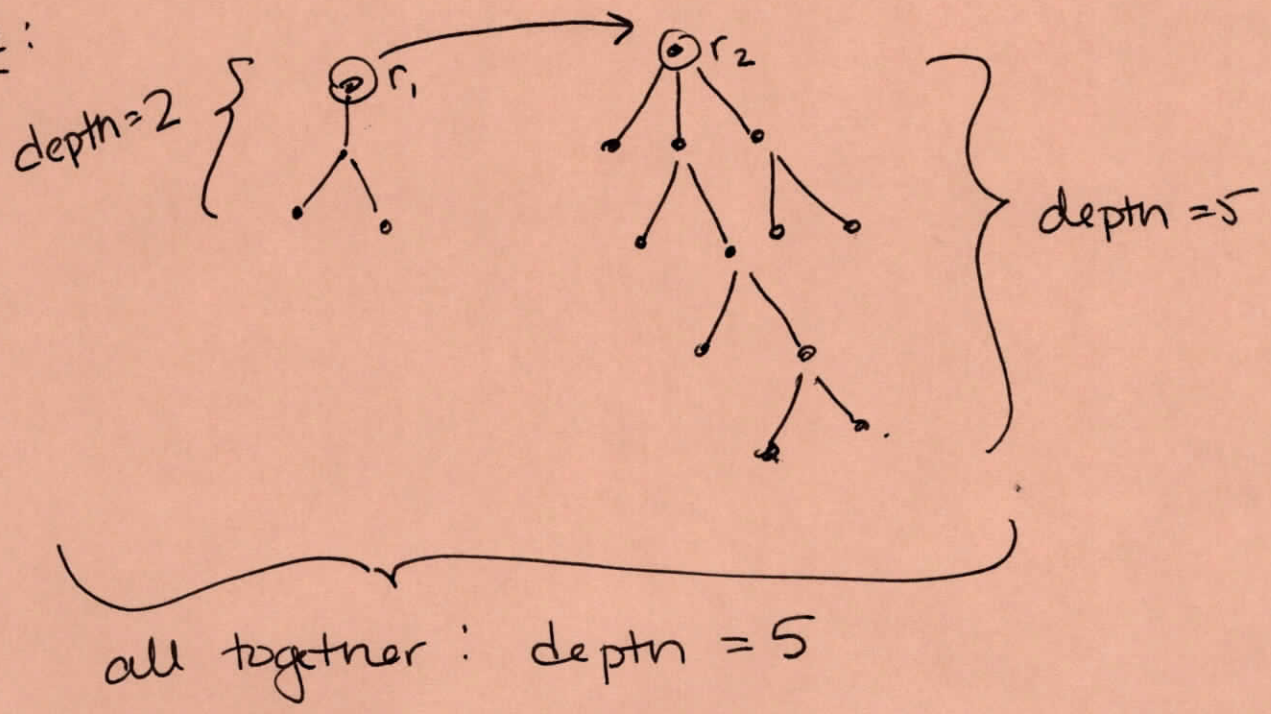
→ union: change one of roots to pt to other (as long as we know roots, that's $\Theta(1)$)

→ FIND: need to walk up tree to find root. $\Theta(n)$

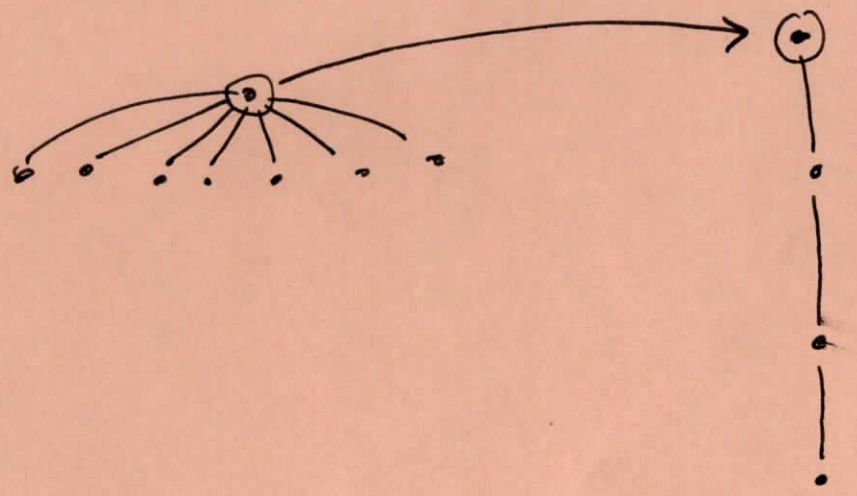
Let's consider Fast union with 2 heuristics / improvements

Improvement 1

example:



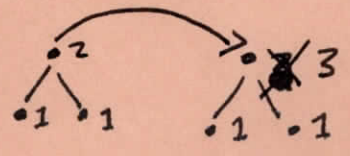
note: we care about depth, not the # of nodes:



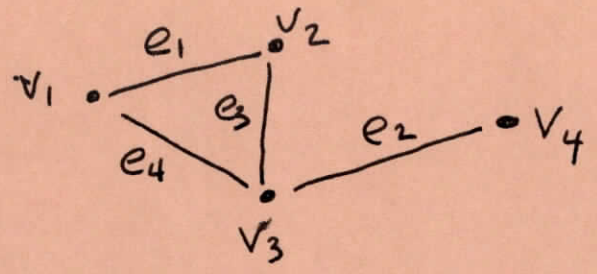
How do we do this?

A: Add a "rank" variable. Initialize to 1.

- If I add smaller to larger, rank doesn't change
- If I add 2 of same rank, rank increases by 1.



Example:



groups:

Walk through UF on this graph using Root union + IMPROVEMENT 1.

initially

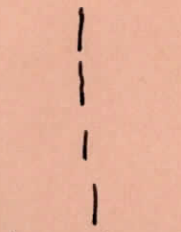
	v_1	v_2	v_3	v_4
parent	1	2	3	4
rank of root	1	1	1	1

← only care if it is a root.

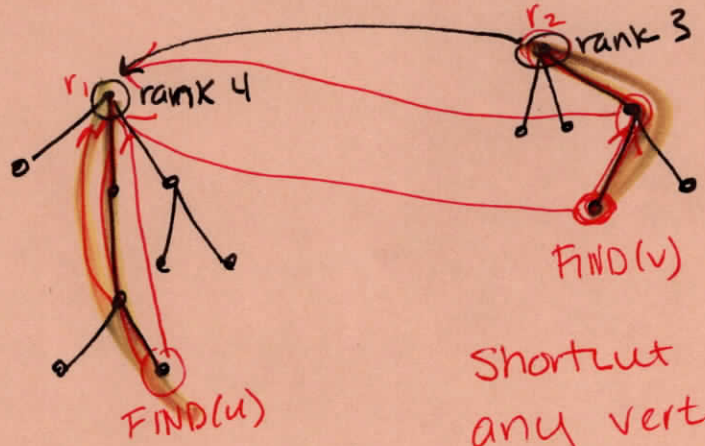
IMPROVEMENT 2 "make bushy"

want:

bushes good



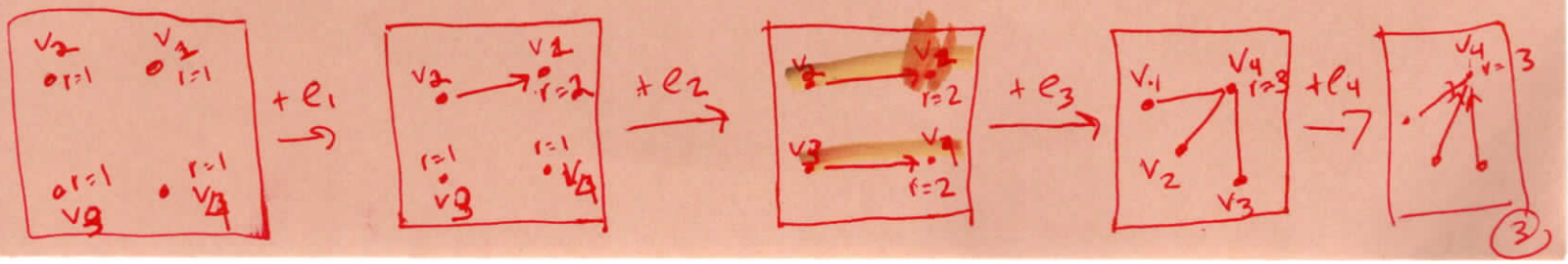
bamboo bad



Shortcut to the new root for any vertex I see on the initial find

Now, run through same example, with both improvements!

(work in groups)



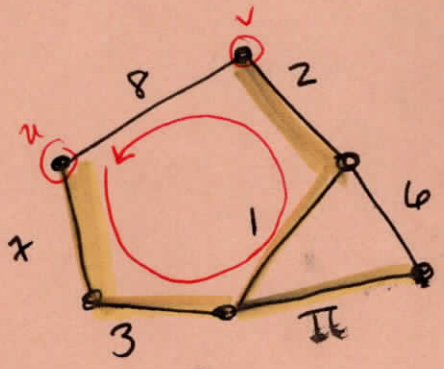
Given a weighted graph $G = (V, E, \varphi: E \to \mathbb{R}_{\geq 0})$, find a tree on $|V|$ vertices of minimal total weight.

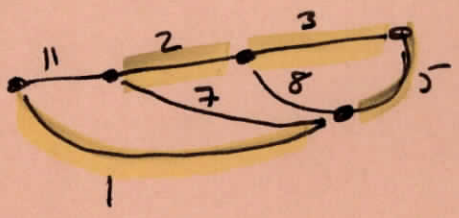
$T \subseteq G$ such that $V_T = V$ and $E_T \subseteq E$

(V_T, E_T) $\quad$ (V, E, φ)

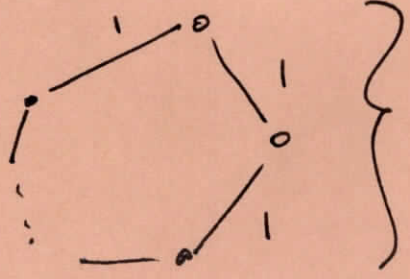
where $\varphi(T) := \sum_{e \in E_T} \varphi(e)$ is minimized.

example:





Q: Is the MST unique? NO
or is it just a MST? YES



cycle graph
w/ n edges / n verts
$\Rightarrow$ n MST's!