Recall: Kruskal's alg.

Sort edges smallest → biggest $\Theta(E \lg V)$

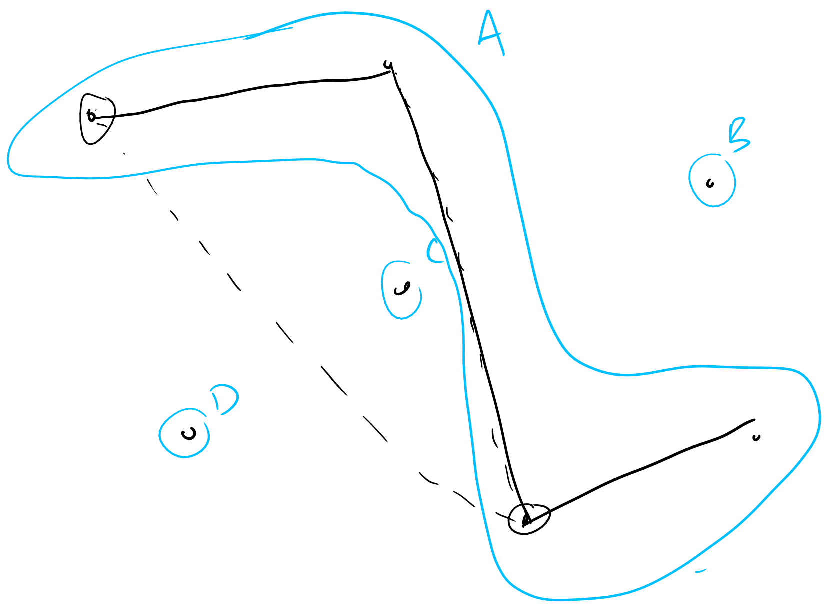
for each edge e: ← $\Theta(E)$ ×

Can we do better? → If choosing would not complete a cycle: ← DFS, $\Theta(V)$

(since we only search through tree)

choose it.

Total: $\Theta(E \lg V) + \Theta(EV) = \Theta(VE)$.



- All vertices start in singleton set.
- Every time we add an edge, union the sets.

Operations we need:

- Start all vertices in singleton sets
- Union two sets together (UNION)
- Query a vertex to see which set it is in (FIND)

This is called a disjoint set data structure, or a union-find data structure.

Kruskal(G):

$T \leftarrow$ empty set of edges.

Sort edges by weight — $\Theta(E \lg V)$

$U \leftarrow$ initialize union-find structure with all vertices in singleton sets. — $\Theta(T_{init}(V))$

for each edge $e = (u,v)$: — $\Theta(E)$

  if $U.find(u) \neq U.find(v)$: — $\Theta(2 \cdot T_{find}(V))$ $E\times$

    Add e to T — $O(1)$

    $U.union(u,v)$ — $\Theta(V \cdot T_{union}(V))$

assume $O(V)$

Total: $\Theta(E \lg V + T_{init}(V) + E \cdot \underline{T_{find}(V)} + V\, T_{union}(V))$

We'd love T_{find} and T_{union} to be $O(\lg V)$!

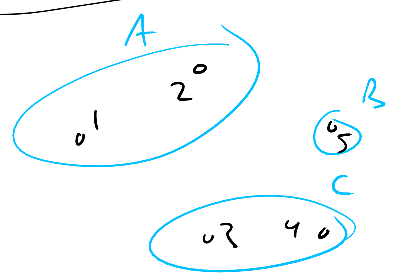
If they were, it would be $\Theta(E \lg V)$.

How to implement union-find?

One idea: dictionary mapping vertices $\rightarrow$ set labels.

Find: $O(1)$:)

Union: $O(n)$:(



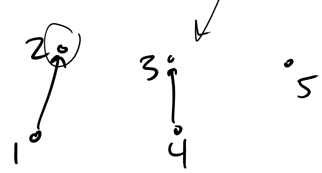
$1 \rightarrow A$
$2 \rightarrow A$
$3 \rightarrow C$
$4 \rightarrow C$
$5 \rightarrow B$

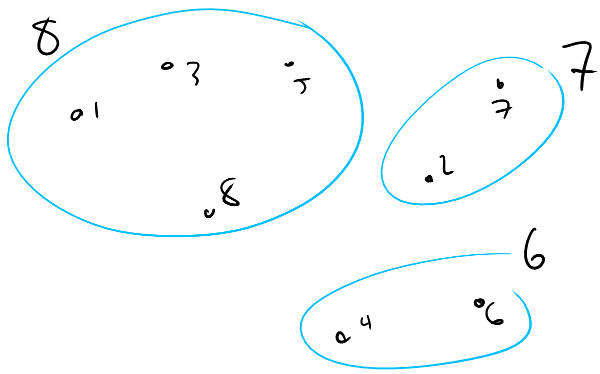
Idea: "lazy set labels".

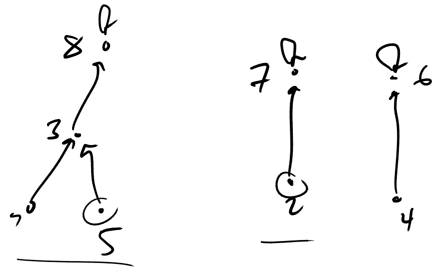
Instead of having each vertex point immediately to its set label, vertices could also point to other vertices in the same set.

e.g.

$1 \rightarrow 2$
$2 \rightarrow 2$
$5 \rightarrow 5$
$4 \rightarrow 3$
$3 \rightarrow 3$.







We will store a dictionary where each vertex maps to its "parent".
A vertex which is its own parent is the root of a tree; we will
use such root vertices as labels for their set.

- Find: just keep querying parent, grandparent etc. until finding a root.
- Union: Find root of both, make one point to the other.

    In particular, make the shorter tree point to the taller.

    how to know?

Keep a second dictionary mapping each root to the height of its tree.

- when unioning different-height trees, make shorter child of taller.
- when unioning same-height trees, make one child of other and increment height of new root.