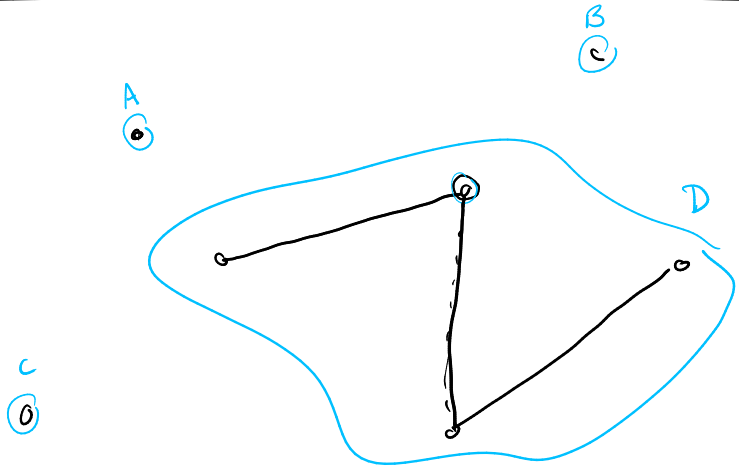
Kruskal's alg. for MST

Kruskal(G): — G weighted, undirected graph
    Sort edges by weight
    for each edge (u,v) smallest to biggest:
        if it doesn't make a cycle:
            add (u,v) to T.



What we need:
- Start every vertex in its own singleton set.
- Ask a vertex which set it is in (FIND)
- Union two sets into one (UNION)

Called a disjoint set or union-find data structure.

Given this:

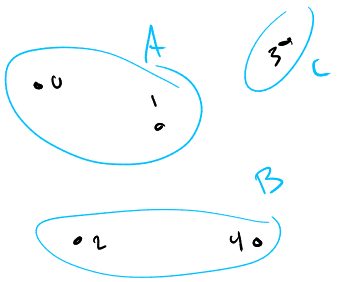
Kruskal(G):
    T ← empty set of edges. $\Theta(1)$
    Sort edges of G by weight $\Theta(E \lg V)$
    U ← new union-find structure with each vertex in singleton set. $T_{new}(V)$
    for each edge e=(u,v) from smallest → biggest: $O(E)$
        if U.find(u) ≠ U.find(v): → $O(T_{find}(V) \cdot E)$ total
            Add e to T $\Theta(1)$
            U.union(u,v) → $\Theta(T_{union}(V) \cdot V)$ total

Total: $\Theta(E \lg V) + T_{new}(V) + O(T_{find}(V) \cdot E) + \Theta(T_{union}(V) \cdot V)$

want find + union to run in $\lg V$ or better.

How to implement union-find?

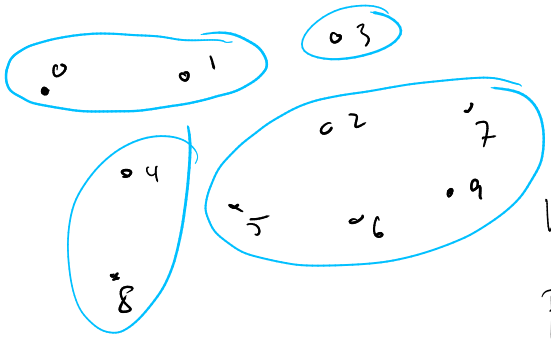
- Dictionary mapping vertex → set?

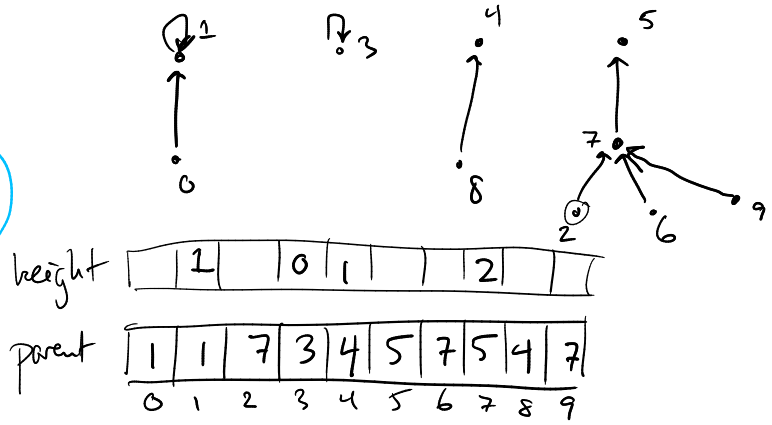


$\{ 0 \to A,\ 1 \to A,\ 2 \to B,\ 3 \to C,\ 4 \to B \}$

- create: $O(n)$
- find: $O(1)$
- union: $O(n)$!!

Idea: each vertex will map to a "parent" which is in the same set. A vertex which maps to itself is the "root" and is used as the name of the set.





- Find: look up a vertex's parent, keep going until finding a root (vertex that points to itself).
- Union: look up roots of both, make the root of the shorter tree point to the root of the taller.
  - if heights are equal, choose arbitrarily and increment height of new root.