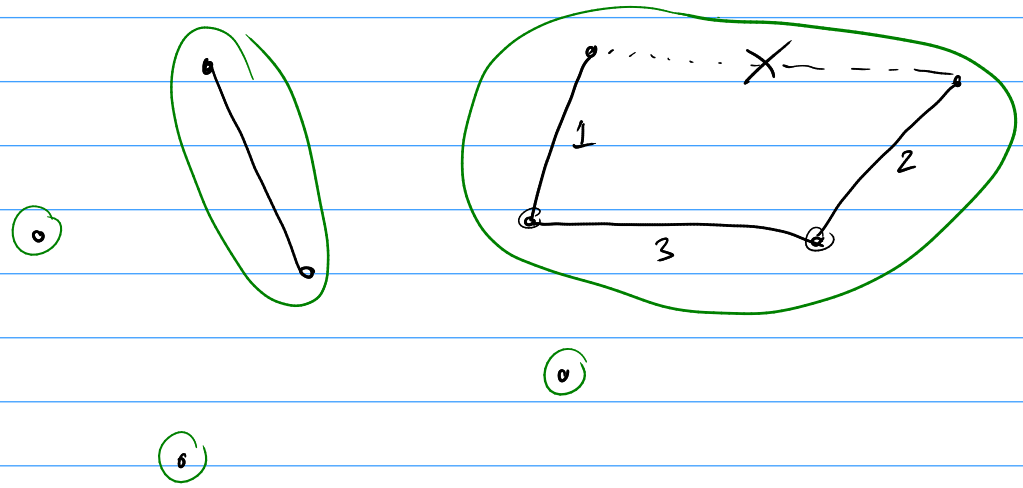
Kruskal's Algorithm

Sort the edges smallest → biggest
For each edge:
    Pick it if it doesn't make a cycle. ⟸



We want a data structure to keep track of sets of connected vertices. It should support the following operations:

- <u>union</u> — union two sets containing 2 given vertices.
- <u>find</u> — given a vertex, which set is it in?
- initialize — every vertex is in its own singleton set.

Often called a "disjoint set" data structure, aka "union-find".

```
Sort the edges smallest → biggest        ← Θ(E lg V)
For each edge (u,v):
    if find(u) != find(v):               ← Θ(E · T_find)
        pick edge (u,v)
        union(u,v).                      ← Θ(V · T_union)
```

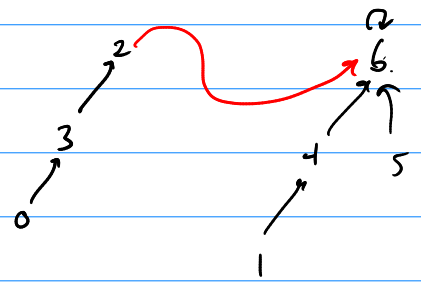
Overall: $\Theta(E \lg V + E \cdot T_{find} + V \cdot T_{union})$.

Goal: T_{find} and T_{union} $O(\lg V)$?

<u>Implementing union-find.</u>

Idea: keep a dictionary where every item points to another item which is in the same set. (its "parent").

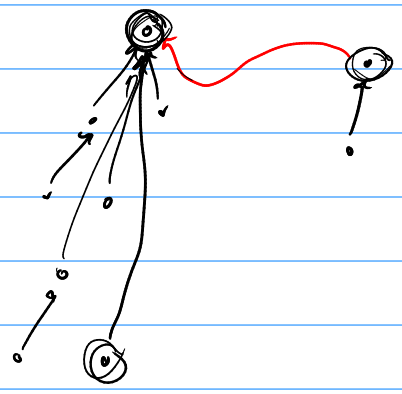
	0	1	2	3	4	5	6
p:	3	4	~~2~~ 6	2	6	6	6



Each set is stored as a tree, with each vertex mapped to its parent. The root of each tree is name of the set.

Union is now constant.

Find — $O(n)$ worst case? $O(\lg n)$ if we build trees carefully.



Idea: keep track of the <u>height</u> of each tree, by keeping a dict mapping nodes to height (height of non-root nodes doesn't matter).

When we union:

- Always make the node w/ smaller height child of the node w/ greater height.
- If the 2 trees have equal height, make either one the parent, increment its height.

Claim: using this scheme, find takes $O(\lg n)$.

- Path compression: when doing find, update everything along the path to point directly to the root.
  → Subsequent calls to find will be faster!

Fact: with path compression, <u>find</u> takes, on average, $\Theta(\lg^* n)$ time.

$\lg^* n$ = # of times to repeat lg before reaching 1.

$n = 2^{2^2} = 2^4 = 16, \quad \lg^*(n) = 3.$

$n = 2^{2^{2^2}} = 2^{2^4} = 2^{16} = 32{,}768, \quad \lg^*(n) = 4$

$n = 2^{2^{2^{2^2}}} = 2^{32{,}768} =$ HUGE!

$\Rightarrow \lg^*(n)$ is ≤ 5 for all #s in our universe.