For Kruskal, we need:
- init each component to a single vertex
- check if 2 verts in same comp
- merge components, given verts in 2em

plan: each comp has a capt
  each vertex in comp knows capt

-init: each vert is its own capt  $O(V)$
- check: check capt(u) = capt(v)  $O(1)$
- merge:
less work to change smaller comp

still worst case is \( O(V) \) for each merge

but really need to know how long total of \( V-1 \) merges take — most are not worst case

work is all changing caps
- count, for each vert, how many cap changs it can have

each cap change represents \( 2x \) size of component
- how many times can a vert comp size double?

\[
\log_2 V
\]

each of \( V \) verts can have \( \log_2 V \) cap changes,

so \( V-1 \) merges do \( O(V \cdot \log V) \) work total
- amortized cost of a merge is \( O(\log V) \)
Kruskal

sort edges by weight \( O(E \log E) = O(E \log V) \;

for each edge \((u, v)\) in \( E \):

if \( \text{comp}(u) \neq \text{comp}(v) \):
    add \((u, v)\)  \( E \cdot O(1) = O(E) \)
merge \((u, v)\)  \( O(V \log V) \) total

\( \text{total} \ O(E \log V) \)