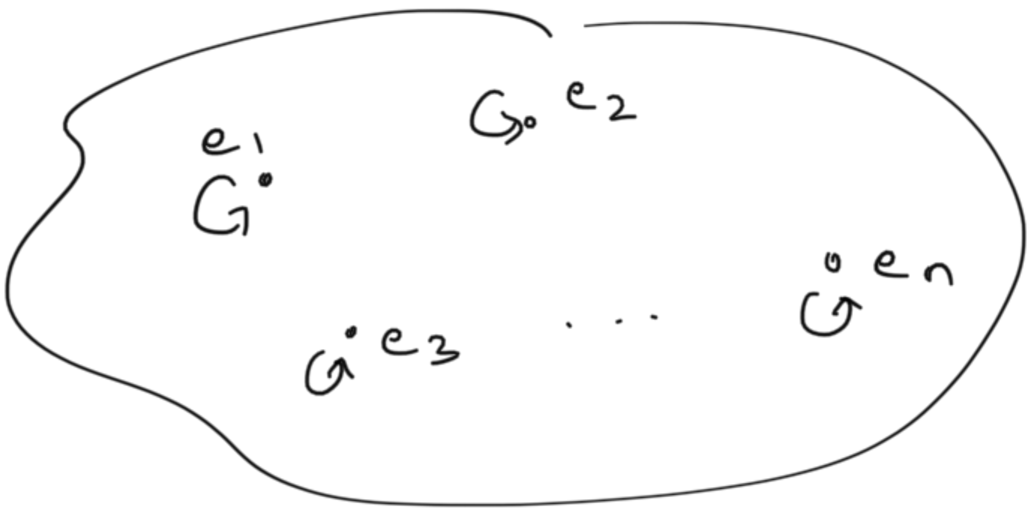
Union Find - another implementation

set S



* elements + arrows from elements to itself

* Interpret $\circ\, e_i \circlearrowleft$ as label (conn. comp ∩ e_i) $= e_i$

* Interpret $e_i \longrightarrow e_j$ as

label (conn. comp ∩ e_i) = label (conn. comp ∩ e_j)

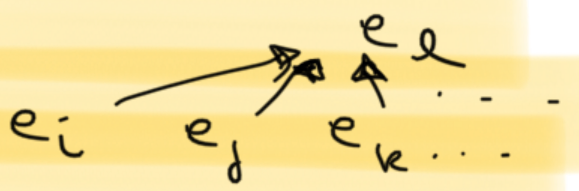
Find (e_i) :

* $e_i \longrightarrow e_j \longrightarrow e_k \cdots \longrightarrow e_l \circlearrowleft$ " label to e_i O(log n)

* set label (e_i) = e_j

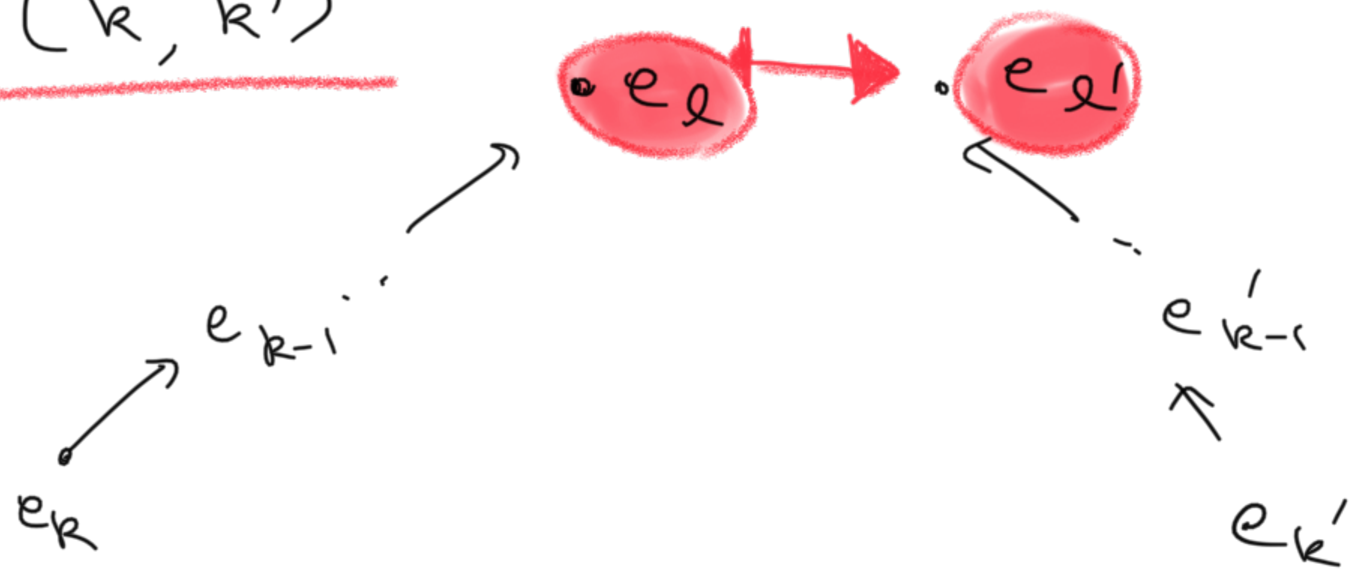
** path compression

* Infact reset



← makes it more efficient!

merge (k, k')



O(1)

look up ↘

* Draw arrow from $\overset{e_{\ell} \,\parallel}{\text{label}(e_k)}$ to

look up → $\underset{\overset{\parallel}{e_{\ell'}}}{\text{label}(e_{k'})}$ if $\text{size}(e_{\ell}) < \text{size}(e_{\ell'})$

* update size of $e_{\ell'}$

* So maintain auxillary data: label(element), size(label(element)).

more complicated amortized analysis

with * * * ① First find : $O(\log n)$

② Then $O(1)$ for nodes involved in first find . . .

n finds ⇝ $O(n \underline{\underline{\alpha(n)}}) \approx O(n)$

↑

Inverse Ackermann function, grows very slowly . . .

$\alpha(n) \leq 4$ for practical values of n apparently