Union Find

→ Data structure which maintains connected components

→ 3 operations

* **Make Union Find (S)**

  given a set S, create a partition made of singleton elements

* **Find (e)**

  Finds the connected component to which e belongs to

* **Union (C, C'):**

  merges the connected components C and C'

What are the connected components going to be called?
It will be labelled by some elt in connected comp.

Let $S = \{0, 1, 2, \ldots, n-1\}$

maintain an array **Components**

Components [i] = label of connected comp to which i belongs

Make Union Find (S) :

create an array

Component = [0, 1, ... , n-1]

component [i] = label of connected comp to which i belongs — O(n)

Find (e) : return component [e] — O(1)

merge (k, k') : want to merge the connected comps labelled k , k'

* walk through array and relabel all k labels to k' say. O(n)

O(mn)

m merge operations take

Better implementation

* As before, maintain an arr **Component**

Component [i] = label of connected comp to which i belongs

* Maintain 2 auxillary arrays. (**Members**, **size**)

Members [k] = [list of elements in conn. comp with label k]

size [k] = # of elements in conn comp with label k

Make Union Find (S) :

create an array

Component = [0, 1, ... , n-1]

Component [i] = label of connected comp to which i belongs

size = [1, 1 ... , 1]

members = [[0], [1], ... , [n-1]]

O(n)

Find (e) : return component [e]

O(1)

merge (k, k') : want to merge the connected comps labelled k, k'

* Let size(k) ≤ size(k')
* Update k label to k'

  for elem in members [k]:

    component [elem] = k'

  O(size[k])
* Update size [k'] = size [k] + size [k']

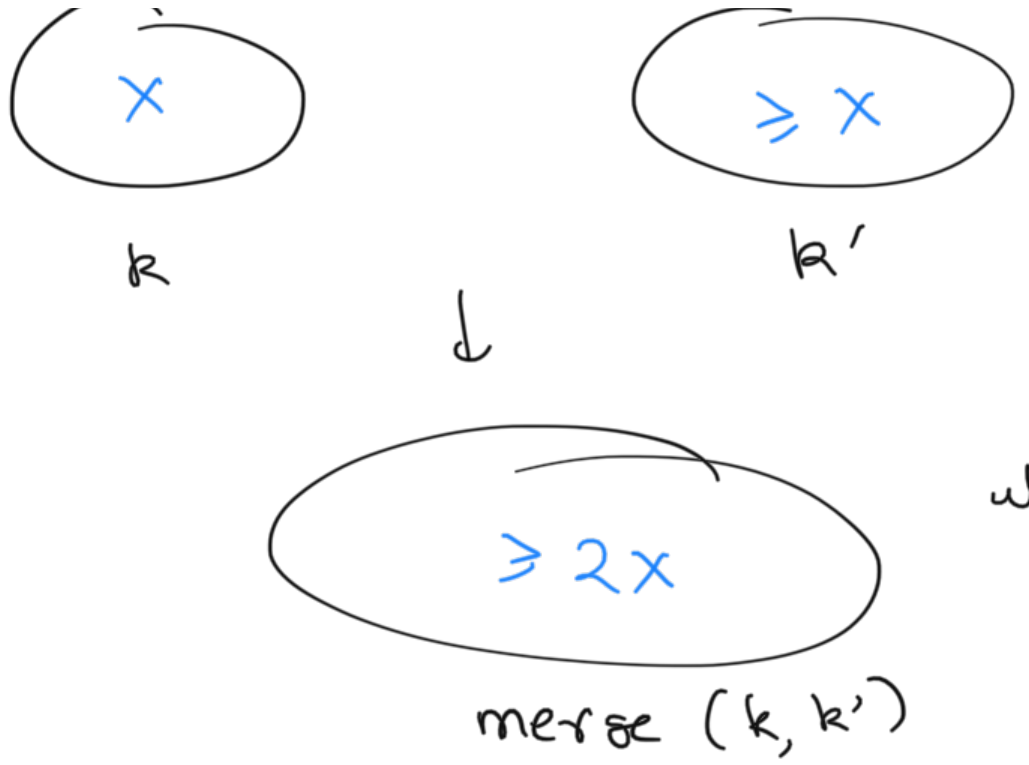
  O(1)
* append members [k] to members [k']

  O(size[k])

Complexity analysis

* we always merge smaller component into larger component

*

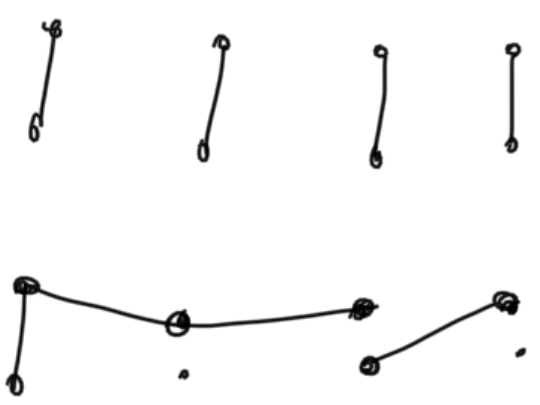


x = size (k) ≤ size (k')

relabelled component size atleast doubles!

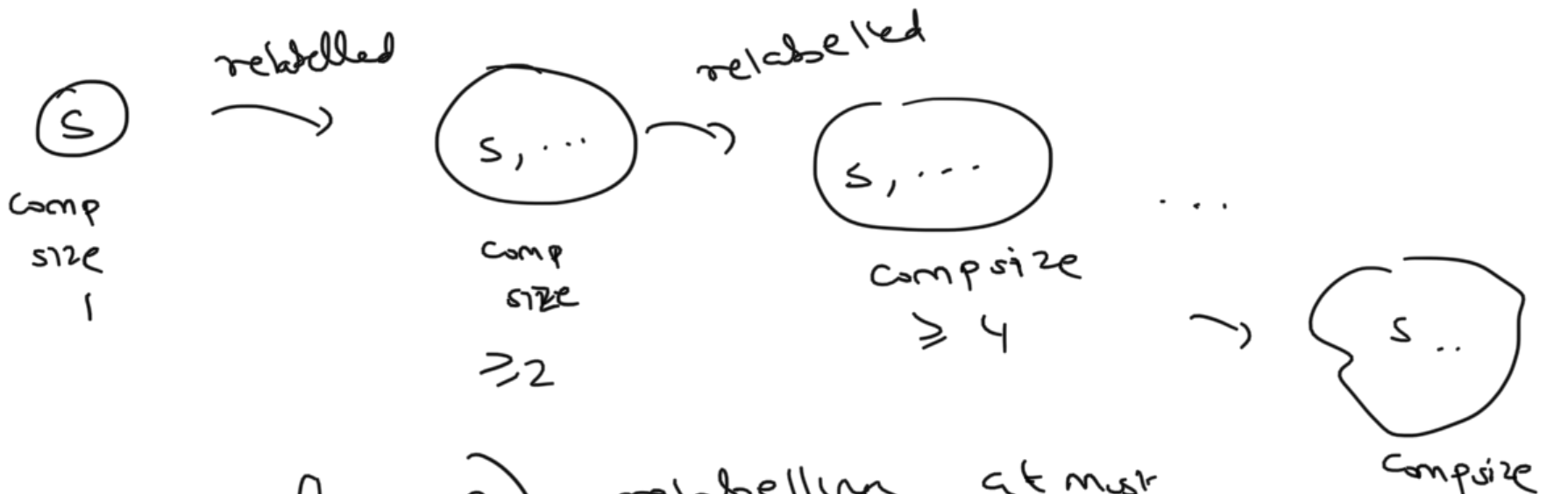
* Initially, all components are singletons
* Let there be m merge operations.

  Over all how many different elements have been "touched"? ≤ 2m



* size of components after m merges ≤ 2m

*



O(log m) relabellings at most for an element s

- at most 2m elements "touched" in m merge operations
- each element relabelled at most O(log m) times
- so O(m log m) / m merge operations
- so amortized complexity = O(log m) / merge operation