Data Structures and Algorithms for ADT Disjoint Sets

Data: Elements $x_1, x_2, \cdots x_n$ (could be integers)

Organization: maintained as disjoint sets

$$S = \{ S_1, S_2, \cdots, S_k \}$$

- A set is non-empty
- A set is represented by one of its members
- The collection is dynamic (changes over time)

Our assumption (for the lab)

- when we declare ∃ 10 elements, the elements are named 0 1 2 3 4 5 6 7 8 9

ADT Disjoint-Set Operations: (union-find)

MakeSet(x):
- creates a new set containing only x
- x is a new element, not already in the set system.
- x is the representative.

Union(x, y) - executed as merge(x, y)
- result is that set containing x and set containing y are now one set

- what should the new representative be?

FindSet(x)

- returns a pointer (or name of) the representative of the unique set containing x

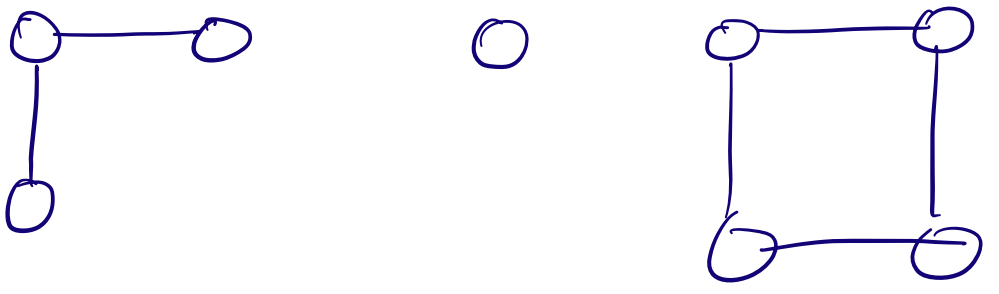
To analyze the running times of various implementations, we use two parameters:

n = number of MakeSet operations

m = number of MakeSet, Union, FindSet operations.

Eg Application: maintaining connectivity information about graph components.

(LANs, for example)



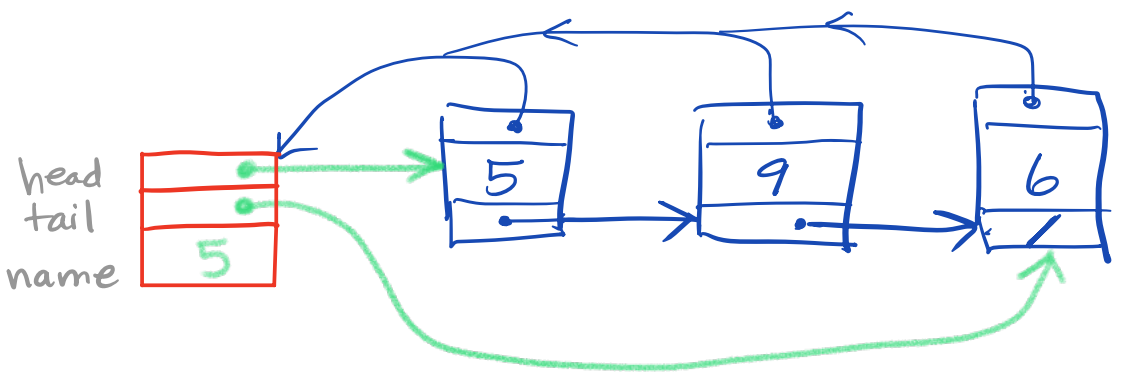
Representing Disjoint Sets as Arrays

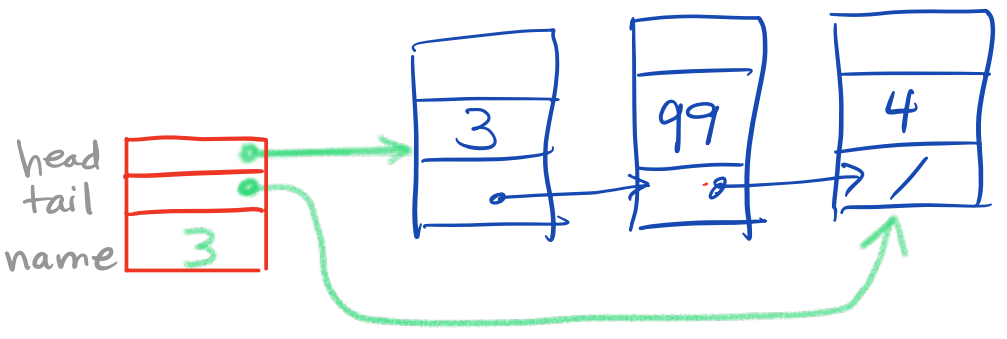
In our world, we don't directly makeSet(x) we establish how many elements there are in advance and then initialize them all to be in their own set.

If we used an array for each set, and each set could grow to contain n elements, then originally each element needs to be in its own arry of size n

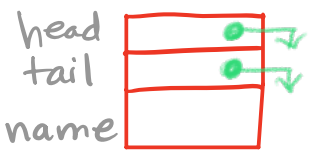
⇒

Representing Disjoint Sets as Linked Lists





MakeSet(x):



FindSet(x):

Union(x, y):

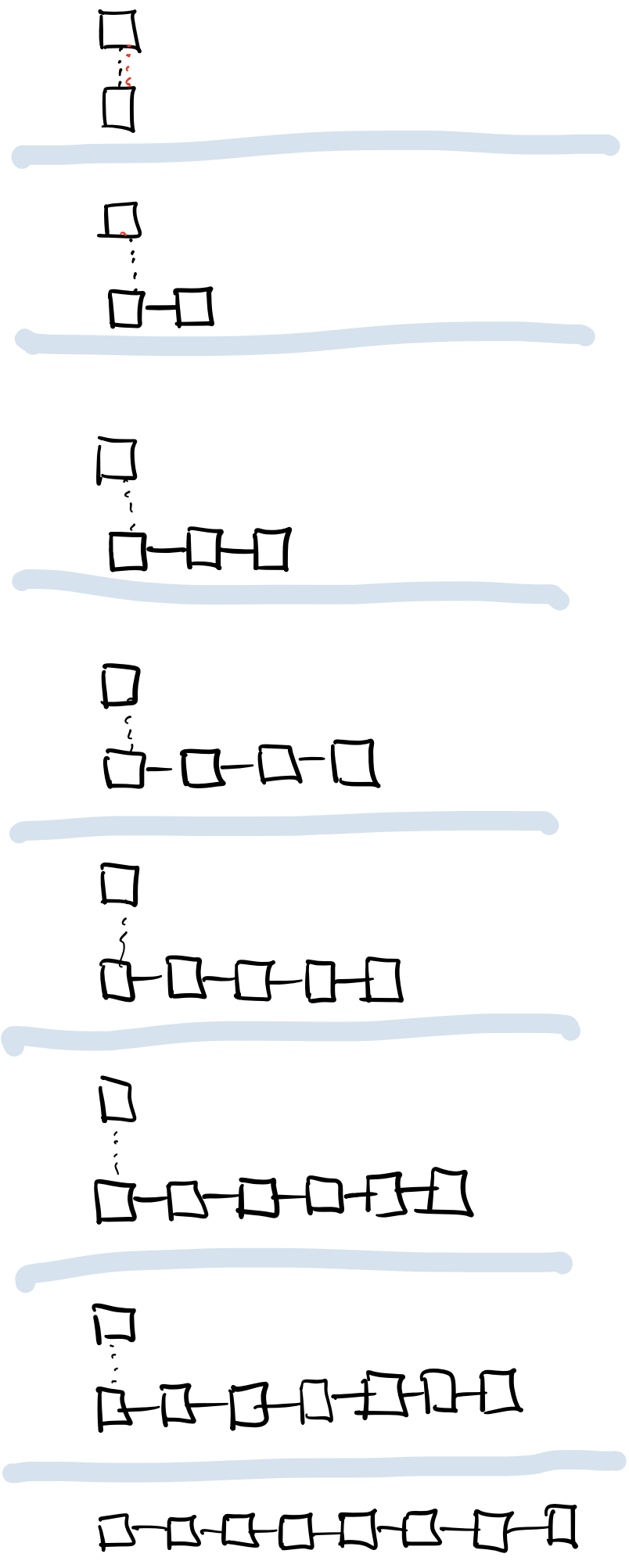
```
if (FindSet(x) ≠ Find set(y))
  while temp ≠ NULL {
    temp = x -> set -> head
    temp -> set = y -> set
    temp = temp -> next
  }
```

Ideas

First, what if we do no optimization.

Worst case # links

to reassign.



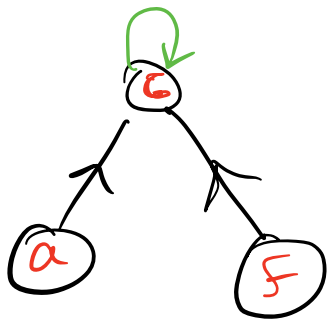
What is total worst-case running time for m ops, n if which are MakeSet?

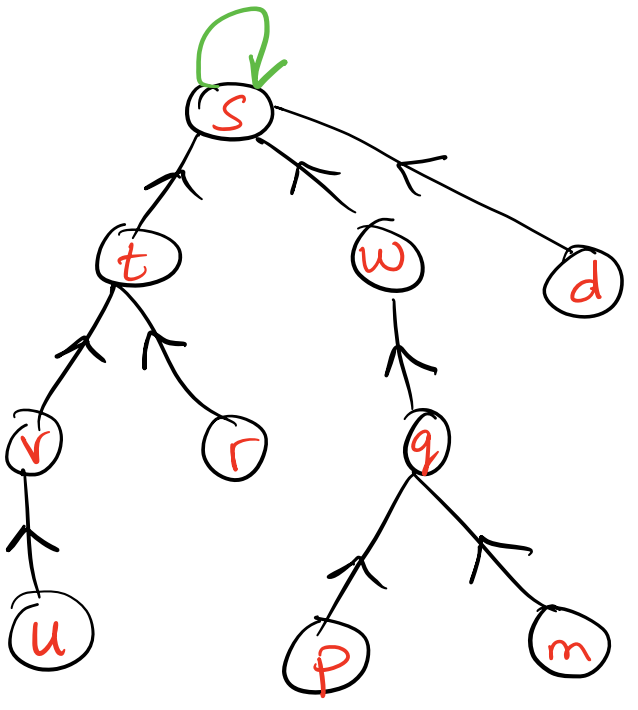
In worst case, $m \in O(n)$ so <u>amortized</u> over m ops

Cover Sept 17, 2026.

ADT Disjoint Sets

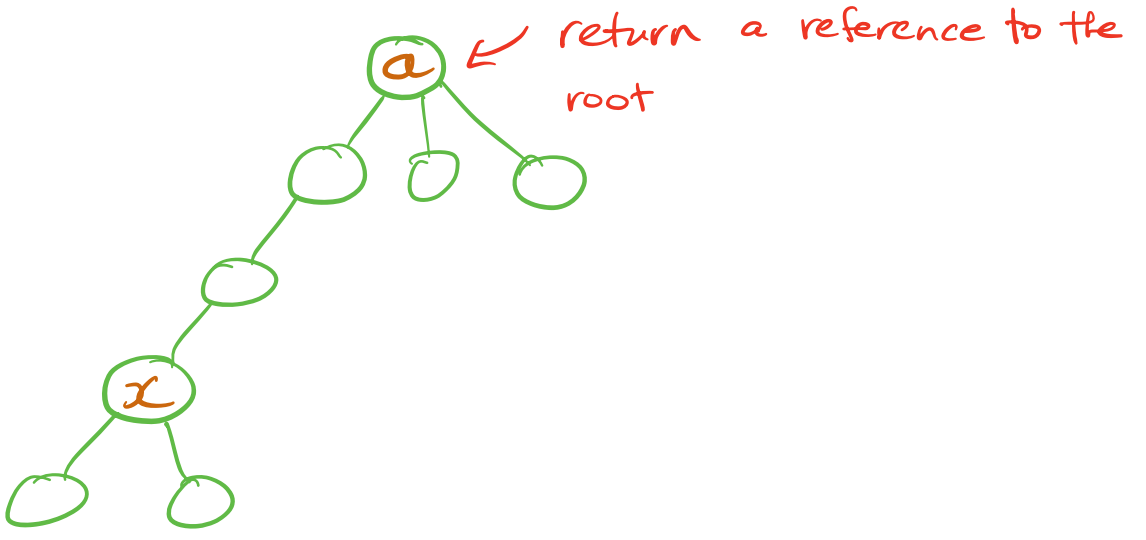
Represented by a Forest



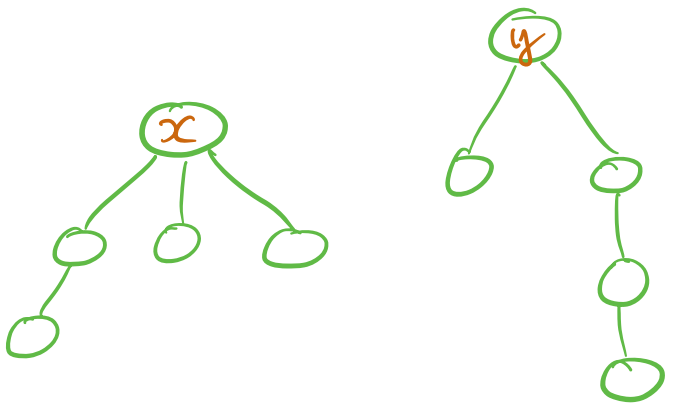


MakeSet(x)

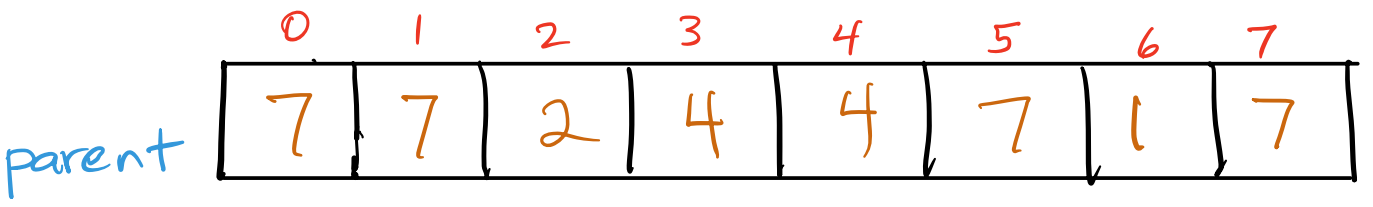
FindSet(x)



Union(x, y)



How to maintain a forest of trees containing elements when our operations are find and union (merge)?



$n = 8$