Linked List Representation (1/2)

- Each set is a singly linked list, represented by an object with attributes
  - head: the first element in the list, assumed to be the set's representative, and
  - tail: the last element in the list.
- Objects may appear within the list in any order.
- Each object in the list has attributes for
  - the set member,
  - pointer to the set object, and
  - next.

Linked List Representation (2/2)

- MAKE-SET: create a singleton list.
- FIND-SET: follow the pointer back to the list object, and then follow the head pointer to the representative.
- UNION: a couple of ways to do it.
  - UNION(x, y)
  - Weighted-Union Heuristic

UNION(x, y)

- UNION(x, y): append y's list onto end of x's list. Use x's tail pointer to find the end.
  - Need to update the pointer back to the set object for every node on y's list.
  - If appending a large list onto a small list, it can take a while.

Operation	# objects updated
$\text{UNION}(x_2, x_1)$	1
$\text{UNION}(x_3, x_2)$	2
$\text{UNION}(x_4, x_3)$	3
$\text{UNION}(x_5, x_4)$	4
$\vdots$	$\vdots$
$\text{UNION}(x_n, x_{n-1})$	$n-1$
	$\Theta(n^2)$ total

Amortized time per operation = $\Theta(n)$.

Weighted-Union Heuristic

- Weighted-union heuristic: Always append the smaller list to the larger list. (Break ties arbitrarily.)
- A single union can still take $\Omega(n)$ time, e.g., if both sets have $n/2$ members.