Homework Assignment #2
Due: January 22, 2019 at 11:30 a.m.

1. In class, we discussed the union-find ADT, which stores a collection of disjoint sets, plus a label for each set, and provides three operations:

- CREATE(x) adds a new set $\{x\}$ to the collection,
- FIND(x) returns the label of the set containing element x, and
- UNION(x, y) combines the sets containing x and y into a single set.

We saw a simple implementation of this ADT. Each of the sets is stored in a linked list. Each node in the list has a pointer to the next element in the list and a head pointer that points to the first element of the list. The first node in the list stores two additional pieces of information: the length of the list and a tail pointer that points to the end of the list.

A CREATE(x) operation creates a new list of length 1.

A FIND(x) operation simply follows the head pointer from the node representing x and returns the item stored in the first node of x's list.

A UNION(x, y) operation first follows the head pointers of x and y to get to the heads of the lists containing x and y. If the two heads are the same, then the UNION is trivial and there is nothing to do. If the two heads are different, the shorter list is traversed, updating the head pointer of each node to point to the head of the longer list. The next pointer of the tail of the longer list is updated to point to the head of the shorter list. The tail pointer and length fields of the first node in the new, combined list are also updated.

We used an aggregate analysis in class to show that the total cost of a sequence of n operations on this data structure (starting from an empty data structure) is $O(n \log n)$ in the worst case. Recall that the idea of this analysis was to look at any one node u and show that, during the whole sequence of n operations, u is traversed by $O(\log n)$ UNIONS. (This is because the length of the list that u belongs to at least doubles each time u is traversed, so after u is traversed $\log_2 n$ times, the list u belongs to contains all n elements.) Summing up over all nodes u, we see that the total number of node traversals by all the UNIONS is $O(n \log n)$ since there are at most n nodes in the data structure. Since the time to do a non-trivial UNION is proportional to the number of nodes it traverses (in the shorter list), the total time for all the UNIONS is $O(n \log n)$.

(a) Redo the amortized analysis using the accounting method to show that the total cost of any sequence of n CREATE, FIND and UNION operations (starting from an empty data structure) is $O(n \log n)$.

(b) Show that this bound is tight. I.e., show that, there is a constant $c > 0$ such that for all n, there is a sequence of n operations (starting from an empty data structure) that takes at least $cn \log n$ steps. (We sketched this in class, but I just want you to do it carefully.)