CS 170 Algorithms
Fall 2014 David Wagner

Worksheet 10/10

1. Amortized running time of a bunch of Finds, if we don't call Union

Imagine we have a set of trees on n nodes, built by some sequence of calls to MakeSet and Union.

(a) Before we begin, put \$1 on each node that's not a root and not a child of a root. This money has to come from somewhere, so let's borrow some money from the bank. How much money do we have to borrow from the bank, in terms of n, in the worst case?

(b) Alice is going to make a sequence of n calls to Find— but no calls to Union. Let's charge her a \$2 service fee for each call to Find. Assume it costs us \$1 for each pointer we follow while executing Find, and the rest of our operations are free.

Consider her ith call to Find. Suppose her ith call to Find is Find(x) where x is a root or x's parent is a root. How much does it cost us to execute Find(x)? Is the \$2 service fee enough to pay for the work we do? How much profit do we have left over?

(c) Suppose her ith call to Find is Find(y) where we have to follow a parent pointer $k \geq 2$ times to get from y to the root. So, executing Find(y) will cost us \$$k$. Sadly, the \$2 service fee isn't enough to pay for this. Where can we find enough money to pay for this? Is it guaranteed to be there when we need it, given how path compression works? How much of the service fee will we have left over as profit?

(d) What's the total amount of profit we'll have accumulated, after a sequence of n Find operations? Do we have enough to pay back our loan from the bank, if the bank doesn't charge us any interest?

2. Amortized running time

If the total time to execute any sequence of n calls to MakeSet, Union, and Find is at most $O(n \lg^* n)$, what is the amortized running time of Union? of Find?