Path Compression, no Weighted Union

The goal this week is to analyze the amortized complexity of the Disjoint Set Forest data structure with the path compression heuristic but without union by rank. This is a chance to understand the power of path compression, on one hand, and to get practice with the potential function method, on another.

Problem:
Consider the data structure for DISJOINT SETS ADT that represents each set by a tree, performs $\textsc{Link}(x, y)$, where x and y are roots, by making y a child of x, and does path compression as part of $\textsc{Find-Set}(x)$ by making x and each of its ancestors point to the root of the tree to which they belong.

Prove that the amortized complexity of $\textsc{Union}$ and $\textsc{Find-Set}$ is $O(\log n)$, starting from the initial configuration consisting of n singleton sets.

Solution:
As discussed in class, since we can turn any $\textsc{Union}$ operation into two $\textsc{Find-Set}$ operations and at most one $\textsc{Link}$ operation, we will assume that we have a sequence of $\textsc{Link}$ and $\textsc{Find-Set}$ operations only.

$\textsc{Find-Set}(x)$ takes time proportional to the depth d of x in its tree, i.e. its distance from the root of the tree. Let's define the actual cost of $\textsc{Find-Set}(x)$ to be $1 + d$, i.e. cost of 1 for each node traversed on the path from x to the root, including x and the root themselves. $\textsc{Link}(x, y)$ takes constant time, and we define its cost to be 1.

We define the potential function

$$\Phi(S) = \sum_{x \in \cup S} \log_2 w(x),$$

where $w(x)$ is the number of nodes in the subtree rooted at x (including x).

For all $x \in \cup S$, the subtree rooted at x contains at least one node, so $w(x) \geq 1$ and, hence, $\log_2 w(x) \geq 0$. Therefore $\Phi(S) \geq 0$. Initially, $w(x) = 1$ for all $x \in \cup S$, so $\Phi(S) = 0$. Therefore, to bound the amortized complexity, we just need to bound the change in potential for each operation.

Let us consider $\textsc{Link}(x, y)$ first. Recall that x and y are both roots of their respective trees, and that, since we are not doing union by rank, x becomes the root of the new tree. Then the only node whose weight changes is x. Therefore the change in potential is $\Delta\Phi \leq \log_2 w'(x)$ where $w'(x)$ is the new weight of x. Since there are only n elements, $w'(x) \leq n$, so $\Delta\Phi \leq \log_2 n$. Thus, the amortized cost of this operation is $\log_2 n + 1 = O(\log n)$.

The worst case cost of FIND-SET(x), where x is of depth at most 1 is at most 2 and the weight of no node changes, so the amortized cost is equal to the actual cost, i.e. is 2.

Finally, consider the cost of FIND-SET(x), where x is of depth $d > 1$. Let $x = x_0, x_1, \ldots, x_d$ be the sequence of ancestors of x, where x_d is the root of the tree. The only nodes whose weights change are $x_1, \ldots, x_{d-1}$. Specifically, $w'(x_i) = w(x_i) - w(x_{i-1})$ for $i = 1, \ldots, d-1$. Hence $\log w'(x_i) < \log w(x_i)$, so the weight changes all decrease the potential. We will divide the nodes $x_1, \ldots, x_{d-1}$ into two groups: those for which the potential decreases sufficiently to "pay for" their cost, and those for which the potential decrease is not large enough. To bound the amortized complexity, we will argue that the second group is small.

Let $I = \{i \mid w(x_i) \geq 2w(x_{i-1})\} = \{j_1, \ldots, j_k\}$, where $j_1 < \cdots < j_k$. Since x_{j_r-1} is an ancestor of $x_{j_{r-1}}$, it follows that $w(x_{j_r}) \geq 2w(x_{j_r-1}) \geq 2w(x_{j_{r-1}})$. Hence, by induction, $w(x_{j_k}) \geq 2^k w(x_{j_1})$. Since $n \geq w(x_{j_k})$ and $w(x_{j_1}) \geq 1$, we get $k \leq \log_2 n$.

If $w(x_i) < 2w(x_{i-1})$, then $w'(x_i) < w(x_i)/2$ and $\log w'(x_i) < (\log w(x_i)) - 1$. Thus the change in potential is $\Delta\Phi < -(d-1-k)$, so the amortized cost of FIND-SET(x) is at most

$$(1 + d) - (d - 1 - k) = 2 + k \leq 2 + \log_2 n = O(\log n).$$

Second Problem, if you have time.

Suppose now that all LINK operations come before all FIND-SET operations. (Again we do not use union by rank but we do use path compression.) Prove that with this ordering, the amortized complexity of the operations is $O(1)$.

Solutions: We define the actual costs of the operations as before. We define a new potential function. Let $\phi(x)$ for any node x in the forest be equal to 1 if x has depth at least 2, or if it has depth 1 and has never been traversed by a FIND-SET; otherwise $\phi(x) = 0$. Then the potential Φ is the sum of all potentials $\phi(x)$ over $x \in \bigcup S$.

Initially every node has depth 0, so the potential is 0. Moreover, the potential is obviously non-negative. Again we can bound the potential change to bound the amortized complexity.

Before any FIND-SET operation, $\phi(x) = 1$ for all x of depth at least 1. Therefore, the potential of any set/tree is equal to the number of nodes in the tree minus 1, as the only node with potential 0 is the root. The potential Φ of the entire data structure then equals $n - k$, where k is the number of sets. Each LINK(x, y) operation increases the potential by 1, so the amortized complexity of the operation is 2.

Consider next a FIND-SET(x) operation. Let again $x_0 = x, x_1, \ldots, x_d$ be the nodes on the path from x to the root, with x_d being the root. Then $\phi(x_{d-i}) = 1$ for any $i \geq 2$ before the operation. Moreover the potential of x_i for all $0 \leq i \leq d$ is 0 after the operation, as all nodes except x_d become children of x_d, i.e. have depth 1, and have been traversed by FIND-SET(x). Therefore, the change in potential is $\Delta\Phi = -\max\{0, d-1\} \leq 1-d$. Since the actual cost of FIND-SET(x) is $1 + d$, the amortized cost is at most $1 + d + 1 - d = 2$.