

Consider the problem of adding delete functionality to the UNION-FIND data structure given in class. The original structure/algorithm supported

- INSERT in $O(1)$ worst case.
- UNION in $O(1)$ worst case.
- FIND in $O(\log^* n)$ amortized.

(n is a bound on how many elements are in the structure)

We want a Data Structure that supports INSERT, UNION, FIND and DELETE. We would like to achieve

- INSERT in $O(1)$ worst case.
- UNION in $O(1)$ worst case.
- FIND in $O(\log^* n)$ amortized.
- DELETE in $O(1)$ amortized.

(n is a bound on how many elements are in the structure, deleted elements are not considered to be in the structure even if their node is still there)

1. Consider the following structure/algorithm: For INSERT, UNION, FIND do what is done in the standard data structure. For DELETE, when you delete an element x you take ALL of x 's children and make them all children of $\text{parent}(x)$. Do the amortized analysis of UNION and of DELETE using the potential function given in class (You will find that you do not get the desired behaviour).
2. Consider the following structure/algorithm: For INSERT, UNION, FIND do what is done in the standard data structure. For DELETE, when you delete an element x you just mark it as deleted but it is still in the tree. The node is still in the tree, but it has no element. Do the amortized analysis of UNION and of DELETE using the potential function given in class (You will find that you do not get the desired behaviour).
3. Consider the following structure/algorithm: For INSERT, UNION, FIND do what is done in the standard data structure. For DELETE, when you delete an element x you just mark it as deleted but it is still in the tree; however, you keep track of how many elements in the entire structure are actually deleted nodes. When the number of deleted nodes is more than the number of valid nodes then the tree is reconstructed by making all of the valid nodes point to their roots, and removing all of the deleted nodes. INSERT and UNION are still $O(1)$, you don't have to show this.

- (a) Show how to reconstruct the tree in linear time.
- (b) Show that DELETE is $O(1)$ amortized
- (c) Show that FIND is $O(\log^* n)$ amortized. (remember, n counts only those elements that have not been removed or marked deleted).