Homework 2.4.1

Deepanshu
2019CS50427

Ans => • From implementation of union, we just have to update the links and thus union takes constant time i.e O(1)

• for find, in worst case analysis all the links could be linear i-e in form of a single list where reaching to the node for establishing link can take linear time because we would have to traverse whole list. Thus find takes linear time i-e O(n) in worst case.

b. Consider the example

① ② ——— ⓝ

1. union(find(1), find(2)) : ① → ② ③ ④ ——— ⓝ
2. union(find(1), find(3)) : ① → ② → ③ ④ ⑤ ——— ⓝ

and so on

Here, we are traversing whole list first to join two sets that takes O(n) time.

c. If we directly keep pointer to the element to be linked then, it would be O(1) but but but
But then to update that pointer, we have to traverse each node.
So, for each set only, we can maintain a pointer which might work in cases with more components with lesser nodes in each component.
We may also perform rotation, or maybe decide a pattern of assigning the 'leader of set' to make its access faster.

Deepanshu 2019055042 7

Page :
Date :

Homework 4-2

Ans:- After using union-by-height method, the height of the trees can be limited to O(log n).

- Union still takes constant time i.e O(1) since it still require updating links and some variable values that might be storing the height.

- find would now take O(log n) as reaching the leader of set (root here) can be achieved in traversing up a tree of height O(log n).

- for updating this information, we basically need information about the size of tree which we can store in the root of each tree.

while comparing the two heights (say h_1 & h_2) final height of combined tree will be either h_2 or h_1+1 depending on the situation.
So, we will update that value everytime we merge two trees into one.