Prof. Fumble claims he has a method to support DELETE’s in the tree-structured UNION-FIND algorithm in a way that allows a sequence of \( n \) operations (UNION, FIND and DELETE) in \( O(n G(n)) \) time. His method is described by the following pseudo-code and figure.

**DELETE(\( i \))**

for each child \( c \) of the node for \( i \)

\[
\text{Parent}(c) \leftarrow \text{Parent}(i)
\]

remove the node for \( i \).

You point out two problems with Prof. Mumble’s method.

**Question 1 (4 points):** DELETE(\( i \)) won’t work correctly for all items. Why?

The problem is if \( i \) is the root of a tree, then Parent(\( c \)) will be set incorrectly.

(Note the problem occurs whether “rootness” is indicated by Parent(\( i \)) = \( i \) or Parent(\( i \)) = nil.)

**Question 2 (6 points):** A sequence of \( n \) operations might actually take \( O(n^2) \) time. Why?

The problem is the for-each loop. We don’t have a way to find just the children of node \( i \) without iterating through all the nodes in the graph.

[Note 1: I should have said \( \Omega(n^2) \) time, since, for example, \( n \cdot G(n) \) is in \( O(n^2) \).]

[Note 2: There might be a way to build a specific shape of tree that takes a long time to do repeated DELETEs, but I haven’t found one yet.]