CSL765: Introduction to (Logic and Functional) Programming
I semester 2019-20

Recovery From Inorder Traversals

As discussed in the document Rambling through Woods on a Sunny Morning, a binary tree can be recovered from it's preorder (or postorder) traversal. Although the same approach can not be used to recover the tree from inorder traversal. The issue arises due to the difference in type of information that is abstracted by these traversals. Preorder (or postorder) traversal does not hide the parent-child relation viz. a parent node will always appear before a child node in preorder traversal (after in postorder traversal); although the parity (number of children) of the parent node is lost. However inorder traversal does not have any such guarantee. For the same reason infix expressions require "cosmetic sugar" such as parentheses, associativity and precedence of operators to enable parsing of expressions unambiguously while prefix (or postfix) expressions need only the arity of each operator.

Problem Statement

Considering the discussion above, design and implement an *efficient* algorithm to recover the binary tree from an inorder traversal (which contains the minumum "cosmetic sugar"). Your task is to document both the traversal algorithm that you use and the solution and thought process for the final out come in a single LaTeX-noweb file. Make sure that you document your final thought processes in the development of your solutions, proofs, algorithms and code. Since each noweb document can handle only a single programming language, decide beforehand whether you are going to program in Standard ML or OCaml.

What you have to do

1. Decide whether to program in Standard ML or OCaml
2. Create a LaTeX-noweb file `<entry-number>.nw` document in which you document your solution along with your thought and design decisions as you go towards your final solution.
3. Create the binary-tree module (with an appropriate signature and structure) that are mutually compatible and implement it in SML or OCaml.
4. Use the same datatype for binary trees that is discussed in Rambling through Woods on a Sunny Morning.
5. Design an *efficient* function `inorder` which yields the inorder traversal with the minimum amount of information that is required for recovering the binary tree.
6. Design an *efficient* function `inorderInverse` to recover the binary tree from the inorder traversal.
7. Discuss the complexity and amount of extra information required by your solution.
8. It should be possible to directly run `noweave` to yield a `<entry-number>.tex` file and then pdflatex to yield a `<entry-number>.pdf` document with all the solutions (including the code).
9. It should be possible to directly run `notangle` to yield all the relevant files (including the test files) for each program that can be run. Each complete program file should be named as follows.
   `Standard ML: <entry-number>-<program-name>-complete.sml`
   `OCaml: <entry-number>-<program-name>-complete.ml`
   and it should be possible to run it directly. Hence the various tests on which it is run are very much part of the file with appropriate "use" statements in them.

Submission Instruction

Submit a `noweb` file on Moodle `https://moodle.iitd.ac.in`

Important Notes

1. Do not change any of the names given in the signature. You are not even allowed to change upper-case letters to lower-case letters or vice-versa.
2. You may define any new functions you like besides those mentioned in the signature.
3. Follow the input output specification as given. We will be using automated scripts to extract your formatted documents and code files and to execute the code for evaluation. In case of mismatch, you might be awarded zero marks.
4. The evaluator will be reading your formatted documents (including the source code) so you must explain your algorithms, proofs and code so that the evaluator can understand every aspect of what you have implemented.