TREETRAV

An Interactive PLATO Lesson on Binary Tree Traversals

Edwin W. Brown III

Computer Science Department

Indiana University

Bloomington, Indiana 47401

TECHNICAL REPORT No. 58

TREETRAV

AN INTERACTIVE PLATO LESSON ON BINARY TREE TRAVERSALS

EDWIN W. BROWN III

STUART C. SHAPIRO

NOVEMBER 22, 1976

TREETRAV

An Interactive PLATO Lesson on

Binary Tree Traversals

Edwin W. Brown III

Stuart C. Shapiro

Computer Science Department
Indiana University
Bloomington, Indiana 47401

PURPOSE

A basic concept taught in any data structures course is that of binary tree traversals. The recursive definitions of the various traversal orders are simple and straightforward, and for this reason the student may consider them easily mastered and not worthy of detailed coverage. However, when the student is presented with a binary tree for which he must provide the correct order of node visits corresponding to a particular traversal type, he will often encounter difficulties unless he has previously taken time to drill himself in this type of exercise. It is for this purpose that lesson TREETRAV is available for interactive use on the PLATO computer-based education system.

DESCRIPTION

Upon entering lesson TREETRAV (figure 1), the student is provided with a master index (figure 2) from which he can choose either to begin the lesson at the normal starting point or to proceed directly to a specific section. If he chooses the former, he will be presented with a brief review of the definition of a binary

tree, using the notation found in Section 2.3.1 of Knuth [1]. It is assumed that the student has had previous exposure to the general concepts of linked structures, trees, and more specifically, binary trees.

This is followed by a session in which the student designs a binary tree to be used later in learning the traversal orders. In this work area (figure 3), the student dictates the structure of his tree by entering commands that perform various functions.

leftn -- creates a left subtree to node n

rightn -- creates a right subtree to node n

deleten -- deletes the subtree with root node n

generate -- randomly generates an entire tree

replot -- replots the screen

The "n" above represents the numeric label of the node to which the student is referring (e.g., left4). To specify a position on the screen for placing a newly created subtree, the student presses keys marked with arrows until the cursor is positioned in a desired location.

When the student completes his tree, he is presented with the concept of traversing binary trees. Recursive definitions are given for the three types of traversal orders -- preorder, inorder, and postorder -- as stated in Knuth 2.3.1 [1]. The student is also given a set of abbreviations that correspond to the different steps that make up the traversal methods.

tL -- traverse the left subtree

tR -- traverse the right subtree

v -- visit the root

These abbreviations are to be used in the next two work areas where the student will be taught how to traverse his tree.

The student then proceeds to the second work area (figure 4), where his tree is replotted. Here he is given the opportunity to see it traversed by the lesson in any traversal order he chooses and at his own pace. During the traversal, each step that is about to be executed is displayed next to the appropriate node, using the abbreviations listed above. In this way a visible record is kept of the last step executed at a particular level before going to the next, so that upon return to an uncompleted level, it will be clear which step is to be executed next. Completion of all three steps at a certain level is indicated by displaying "ok" next to the appropriate node. As each node is visited it is circled, and its label is listed at the bottom of the screen, so that when the traversal is complete, a record of the order of visits is available. At the end of the traversal, the student is encouraged to try some other traversal orders on different trees until he feels confident of his understanding.

When he has decided that he fully understands the traversal definitions, the student may proceed to the third work area (figure 5), where he must enter each step to be executed rather than having it provided for him as it was in the last area. Both this and the next area are examples of the "monitored tasks" referred to in Shapiro [2]. That is, the student is constantly being monitored during the exercise -- if he enters the correct step it is executed,

otherwise he is informed of his error. At any time in this traversal session, as well as in the others, the student may press the HELP key to see the recursive definition for the traversal order that is being carried out. Again, upon completion of this section, he is urged to repeat it with other traversal orders.

The final work area (figure 6) consists of a randomly generated tree for which the student must provide the correct order of node visits that corresponds to a traversal order of his choosing. He does this by simply entering the numeric label of each node as it is to be visited, with no record kept of the traversal steps occurring between visits. This provides a true test of what he has learned, for in order to succeed, he must be able to mentally keep track of the steps to be executed and the levels completed. As in the previous session, the student is monitored for the accuracy of his answers and has access to the HELP key.

When he leaves the final work area, the student is given a score (percentage of correct first choices) that reflects his performance during the exercise. At this point he is given the option of repeating the last exercise with a new randomly generated tree, or returning to the master index (figure 2), from which he can go to any portion of the lesson. The index may be accessed at any point in the lesson, thus allowing the student complete freedom to enter any work area he wishes. A comment facility is also provided in order that the student may voice any opinions or questions he may have about the lesson.

IMPLEMENTATION

The structure of the student's tree is stored as an array of screen coordinates corresponding to the positions of the nodes. The indexes of this array serve as labels for the nodes, and are also used in the pointer fields to designate the left and right links of each node.

The randomly generated trees are created by using a random number generator (system provided) to decide whether to create a left, a right, or no subtree, and then consulting a table containing screen coordinates of possible node positions to determine where to place the generated subtrees. A system of tagging and level indexing is used to prevent placing two nodes at the same position, or placing a subtree of a node at a higher level than itself.

The traversals are accomplished using a simple stacking algorithm, in which the node label corresponding to an uncompleted level is placed on a stack before proceeding to the next level. When all steps at a particular level are completed, the top of the stack is removed and that level returned to. An empty stack indicates completion of the traversal. A flowchart describing the algorithm in more detail appears in figure 7.

REMARKS

It is hoped that lesson TREETRAV will find a genuine use by those engaged in teaching a data structures course and whose students have access to a PLATO terminal. The lesson can be found listed under Information Processing in the main index of PLATO computer science lessons (lesson CSLESSONS). In writing the lesson, attention was given to the criteria set forth in Shapiro [2]

for a well-structured CAI (Computer-Assisted Instruction) lesson. Every effort will be made to review the lesson regularly, read the users' comments, and make appropriate modifications.

REFERENCES

1. Knuth, Donald E., The Art of Computer Programming, Vol. 1: Fundamental Algorithms, Second Edition, Addison-Wesley, Reading, Massachusetts, 1973.

2. Shapiro, Stuart C., PLATO Lessons for a Data Structures Course, Technical Report No. 15, Computer Science Department, Indiana University, August, 1974.

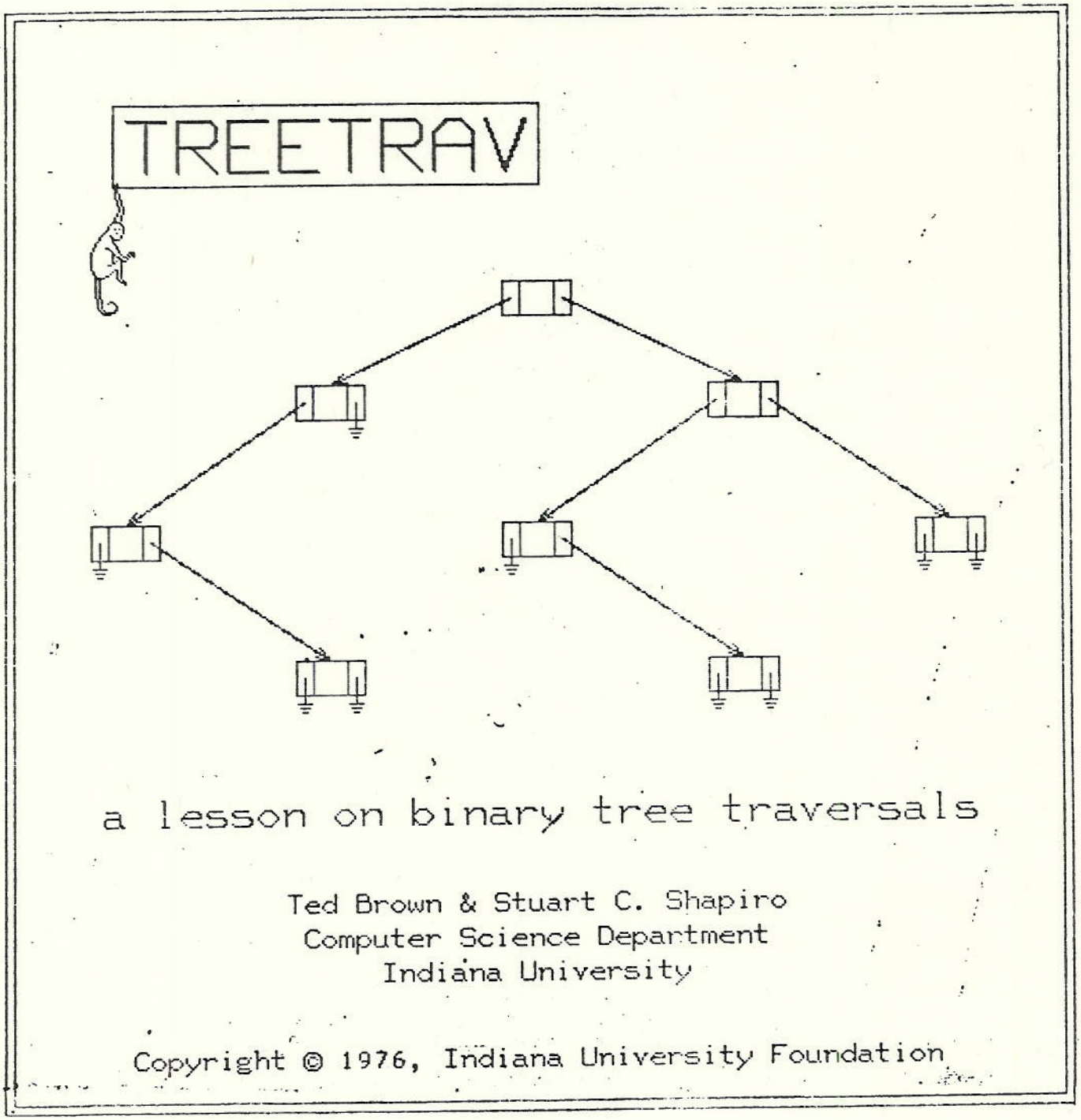


figure 1

Title Page

TREETRAV INDEX

Press the number of the section you would like to choose:

1) Introduction: You should start here if this is your first time in this lesson. This will allow you to progress through the entire lesson in proper sequence, which is necessary before you can understand how to use sections 2 through 5 below.

2) Tree Laboratory: To build or modify your binary tree.

3) Traversal Laboratory I: To traverse your tree with each step displayed before being executed.

4) Traversal Laboratory II: To traverse your tree by entering each step before it is executed.

5) Traversal Laboratory III: To test your knowledge of the traversal orders by entering the correct order of node visits for a randomly generated tree.

6) Comment: To leave a comment about the lesson.

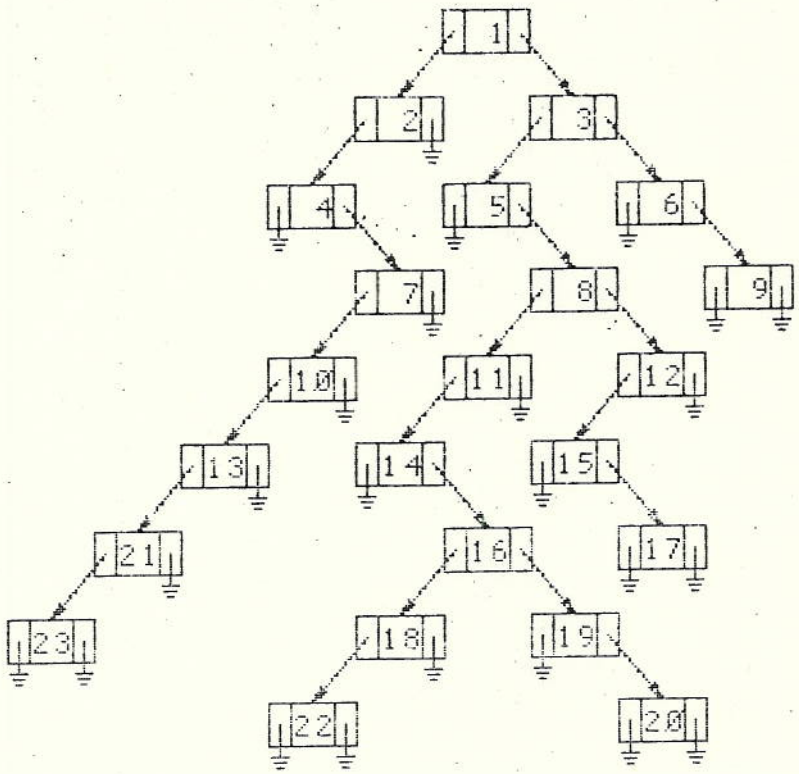
7) Exit: To return to the lesson you came from.

Press SHIFT-BACK at any time to return to this index.

figure 2

Master Index

TREE LABORATORY

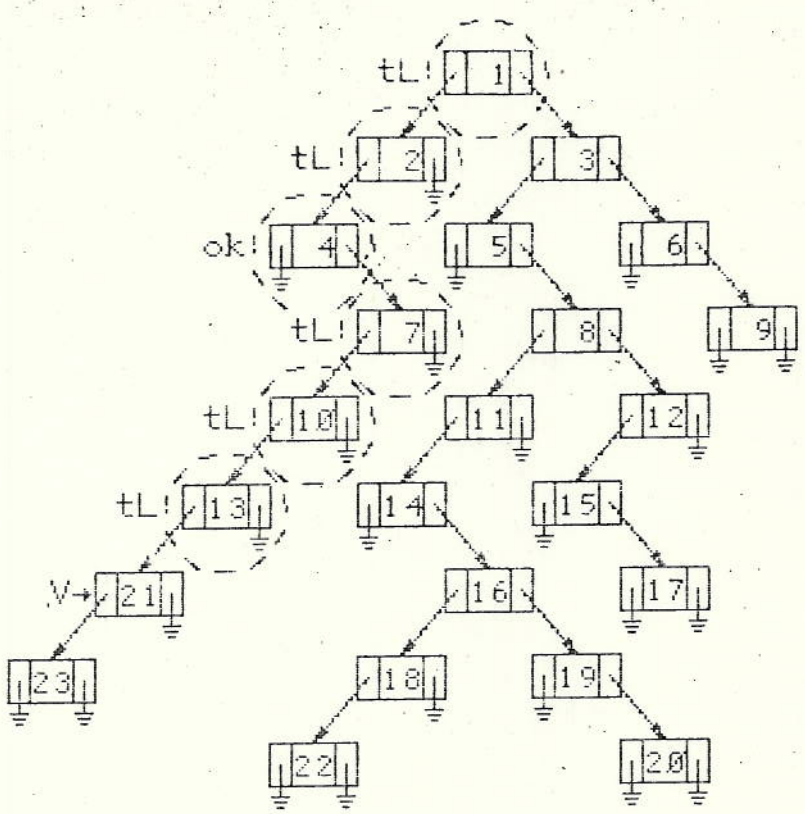


options: leftn, rightn, deleten, generate, replot, end
> left21 no
node 21 already has a left subtree

figure 3

Tree Building Work Area

TRAVERSAL LABORATORY I



1 2 4 7 10 13

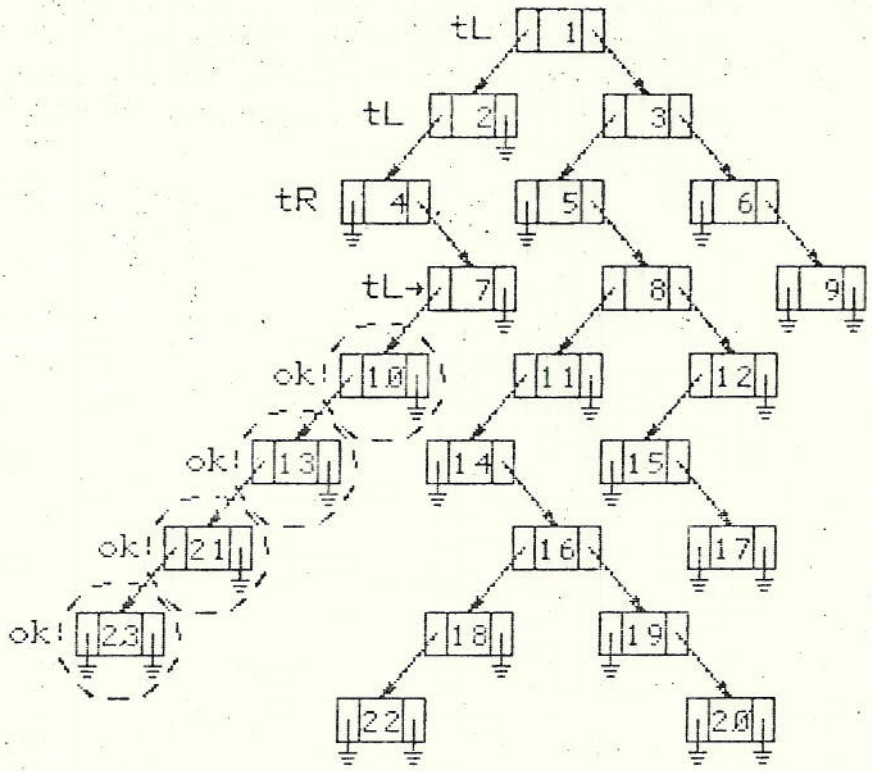
press NEXT to execute next step in PREORDER

(press HELP to see correct sequence of steps)

figure 4

First Traversal Work Area

TRAVERSAL LABORATORY II



23 21 13 10

enter next step to be executed in POSTORDER

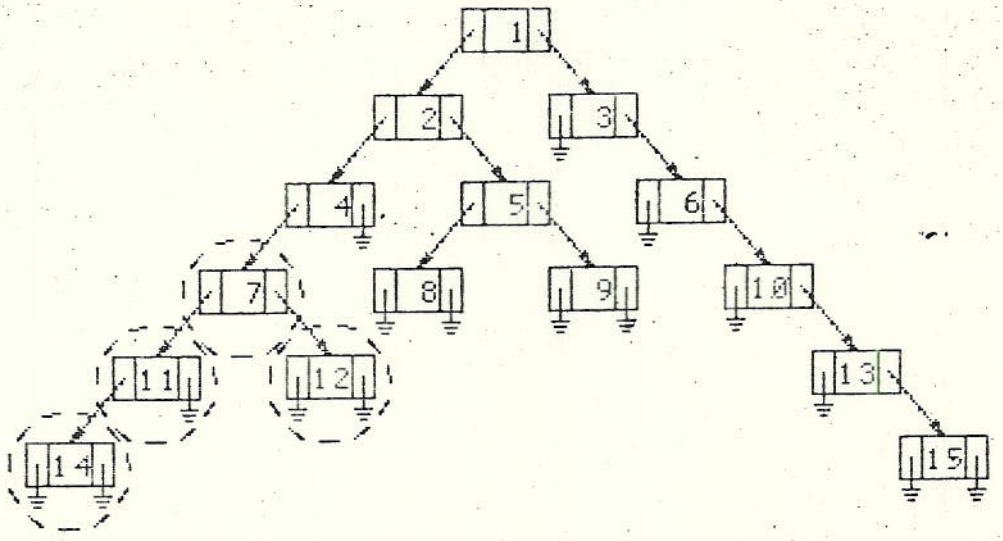
» tR

(press HELP to see correct sequence of steps)

figure 5

Second Traversal Work Area

TRAVERSAL LABORATORY III



14 11 7 12

enter number of next node to be visited in INORDER.

> 4

(press HELP to see correct sequence of steps)

figure 6

Third Traversal Work Area

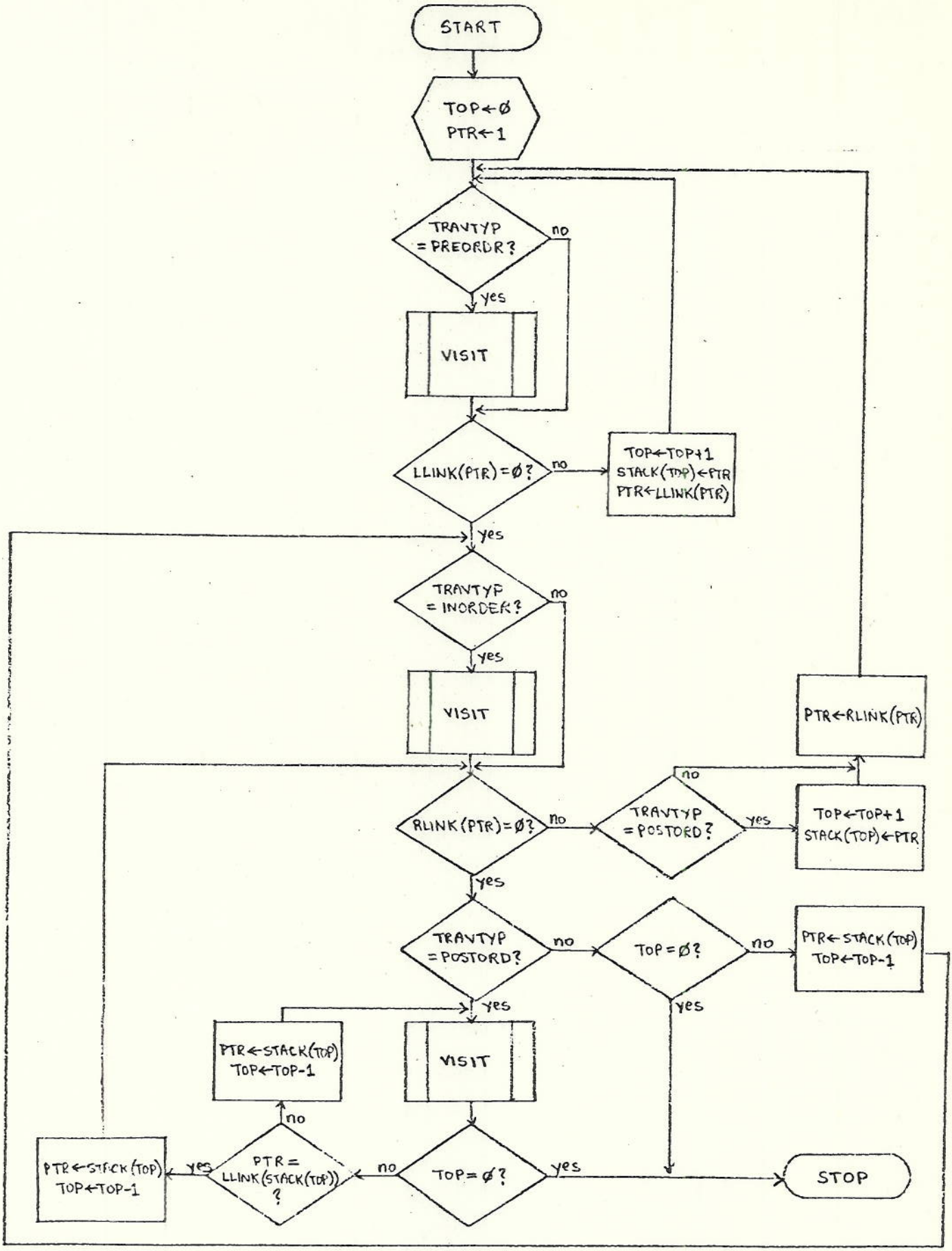


figure 7

Traversal Algorithm