A New Look to Traversal Algorithms Using Set Construct Data Structure

Vineet Kumar Sharma
Department of Computer Science & Engineering
Krishna Institute of Engineering & Technology
Ghaziabad, UP, India

Adesh Kumar Pandey
Department of Computer Science & Engineering
Krishna Institute of Engineering & Technology
Ghaziabad, UP, India

Dr. V.K. Srivastava
Department of Computer Science & Engineering
Krishna Institute of Engineering & Technology
Ghaziabad, UP, India

Abstract—**Tree traversal refers to the process of visiting or examining or updating each node in a tree data structure, exactly once, in a systematic way. Such traversals are classified by the order in which the nodes of the tree are visited. This paper presents a new and innovative technique using which traversing in trees as well as in graph becomes extremely easy and using this technique explanation & understanding of traversing in trees & graphs also becomes easy using set construct.**

Keywords— **Tree, Graph, Traversing algorithm, set construct, adjacency matrix**

I. INTRODUCTION

Tree & Graph traversing has been a great challenge for the students and research scholars in the field of computer science. Both are the perfect examples where recursion technique is deeply involved. Compared to linear data structures like linked list and one dimension assay, which have only one state of traversal, tree traversing can be done in various ways. Starting from the root 8 the binary tree there are three main steps that can be performed and the order in which they are performed defines the traversal type. These steps are performing an action on the current node, named as visiting the node, traversing to the left child node & then traversing to the right child node. Thus the process is most easily described through recursion.

The following techniques are the conventional methodologies for tree traversing. To traverse a non empty binary tree in preorder the following operations have to be recursively performed at each mode, starting with the root node. It is also known as depth first search.

1. Visit the node
2. Traverse the left sub tree.
3. Traverse the right sub tree.

To traverse a non empty binary tree in inorder, perform the following operations recursively at each node. It is also called symmetric traversal.

1. Traverse the left sub tree
2. Visit the node.
3. Traverse the right sub tree.

To traverse a non empty binary tree in post order, perform the following operations recursively at each node. It is also called level order traversing, where we visit every node on a level before going to a lower level.

1. Traverse the left sub tree
2. Traverse the right sub tree.
3. Visit the root.

II. PROPOSED METHODOLOGY

The technique using set construct presented here requires the following keywords & notations.

A. Tree

Tree is an ordered set of subtrees T(Xi) where Xi are the nodes of the tree defined by ordering them in almost complete binary tree manner of cardinality n as:

$$\{T(X_1), T(X_2), T(X_3) \ldots\ldots, T(X_n)\}$$

In almost complete binary tree manner, for a level tree of depth'd' node X_1 is at level 0 and node X_2 is at level 1, which is the left most node. On traversing from left to right we reach node X_n of level/depth'd'.

B. Subtree

It is the set of three members N, L, R defined as {N, L, R} where

i) node : N – Parent/leaf/root
ii) left child : L – null/leaf/subtree

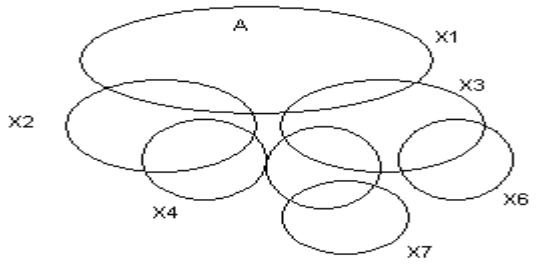
right child : R – null/leaf/subtree

Traversal set: T(N) is an ordered sub tree for the node N according to one of the tree traversal methods. i.e.

(i) Pre order Traversal – {N, L, R}
(ii) Post order Traversal – {L, R, N}
(iii) In order Traversal – {L, N, R}

Result set R: It is an ordered set and its members are the nodes of the tree, after execution of the program members are in required path of traversal.

A. *C. Marking of sub* trees



B. ***Fig. 1 Marking of sub* trees**

Work set W: An ordered set; members are nodes, L or R of T.
Algorithm for tree traversal in binary tree

Step 1: Initialize R and W to null
Step 2: Set y = X_1
Step 3: W ← T(y) UW
Step 4: if the left member of W

Case 1: node, move it to the set R and go to step 4
Case 2: subtree, set y = parent node of this subtree and goto step 3
Case 3: null, exit

Explanation of algorithm:- The algorithm starts traversing from the root of the tree. In inorder traversing technique L, N, R concept is used where L, N, R represents left, Node, & right respectively.

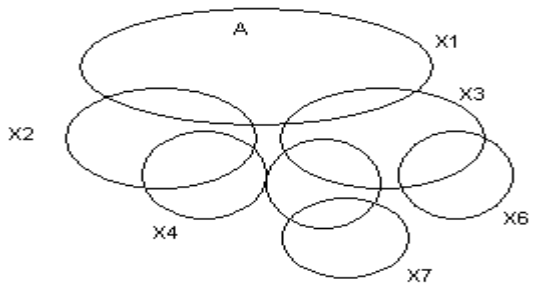


Fig. 2 Traversing in subtrees

For traversing we come first of all at – X_1. Because inorder follows LNR, traversing in X_1, would result X_2 50 X_3. Now we start traversing in X_2. Traversing in X_2 generates X_4 40. Then traversing in X4 only traverse node 35.
So X_2 50 X_3 generates
X_4 40 50 X_5 60 X_6
35, 40, 50, 58, 59, 60, 65

Similarly we can easily traverse in preorder & post order style.

IV. TRAVERSING IN GRAPH

This innovative technique using set construct can also be implemented in the traversing of graph. For traversing in graph several keywords & terminology are required.

Adjacency Matrix

The adjacency matrix of finite graph G on n vertices (nodes) is the n x n matrix where the non diagonal entry a_{ij} is the number of edges from vertex i to vertex j, and the diagonal entry a_{ii} are assumed to be ordered from V_1 to V_n.

The nxn matrix A, in which
A_{ij} = 1 if there exists a path from V_i to V_j
A_{ij} = 0 otherwise

Adjacency set A(N)
N (say i^{th} vertex) node; members which are adjacent nodes to N and currently present (say j^{th} vertex) i.e. a_{ij} = 1 in the adjacency matrix i^{th} row at the time of execution i.e. from the current states of the adjacency matrix.

Update adjacency matrix
For the member/node (say j^{th} vertex) which is included in the adjacency set A(N), set all is to zero in the column vector for that node in the adjacency matrix.

Result set R
An ordered set; members are the nodes of the graph. After execution of the program, members are in the path of traversal from the given node.

Work set W
An ordered set; members are nodes of the graph.
Preorder Traversal
A U M
Postorder Traversal
W U A

V. PROPSED ALGORITHM

Traverse in the given graph from a given source say y.

Construct the adjacency matrix A. then
Step 1: set W to null.
Step 2: initialize R with y as first member and update adjacency matrix.
Step 3: get A(y) and update adjacency matrix.
Step 4: apply the required traversal scheme i.e.
W = AUW or W = WUA
Step5: if the left member of W
Node (say Z), move Z to R
Set y = z and go to step 3.
Else null, exit

A. Explanation of algorithm

Suppose the following graph is given to us & we have to traverse with in it. A is given as the source vertex.

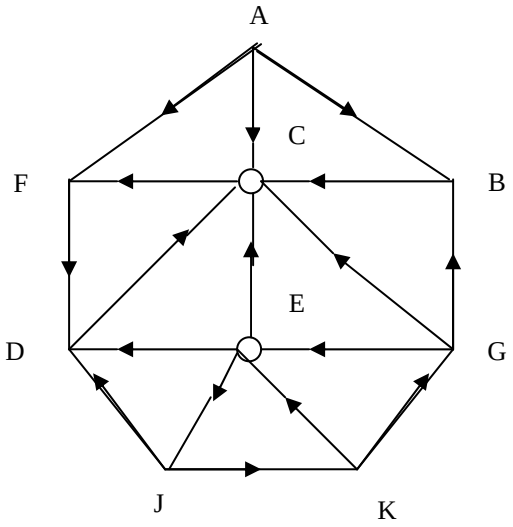


Fig. 3 Connected graph

	A	B	C	D	E	F	G	J	K
A		1	1			1			
B			1				1		
C						1			
D			1						
E			1	1				1	
F				1					
G			1		1				
J				1					1
K					1		1		

Fig. 4 Adjacency *Matrix*

We stand traversing from A. We put A in R, its adjacency set as is B, C & F because A connects to B, C & F vertices. The same becomes the workset and the column of B, C &F becomes zero.

R = {A}
A$ = {B, C, F}
W$ = {B, C, F}

1. Now B vertex has been extracted from WS and it is inserted into the result set. The adjacency set is now {G}
R = {A, B}
A$ = {G} Because B connects to vertex G and C but C previously column has converted all 1 to 0 & that W$ = {C, F} is why we are not considering C again.
Now W$ = WSUA
W$ = {C, F, G}
Now G is introduced in W$ so we converted the 1's of G column into 0.

2. Now we extract C from WS and put it in R
R = {A, B, C}
A$ = { }
Because C does not connect to any vertex.
Previously WS = {F, G}
Now WS is same i.e. {F,G}

3. Now we extract from WS and put it into R
R = {A, R, C,F}
A$ = {D}
Previously WS was {G}
New WS = {G, D}
Here we introduce G is WS so we make all the 1's of this column to 0.

4. Now G is extracted from WS and it is put in R
R = {A, B, C, F, G}
A$ = {E}
Previous W$ = {D}
New W$ = {D, E}
Because E is introduced in WS to we make all 1's of this column to 0.

5. Now D is being extracted from WS, We put it in R
R = {A, B, C, F, G, D}
AS = { }
Previous W$ = {E}
New W$ = {E}

6. Now we extract E from WS & put it in R
R = {A, B, C, F, G, D, E}
AS = {J}
Previous W$ = { }
New W$ = {J}
Because J is introduced in WS so we convert all 1's of its column to 0.

7. Now we extract J from WS & put it in R
R = {A, B, C, F, G, D, E, J}
AS = {K}
Previous W$ = { }
New W$ = {K}
Because K is introduced in WS so we convert all 1's of its column to 0.

8. Now we extract K from WS & put it in R
R = {A, B, C, F, G, D, E, J, K}
AS = { }

Previous W$ = { }
New W$ = { }
Now this time WS becomes null so algorithm terminates at this point.

VI. CONCLUSIONS

The study presented an innovative technique for traversing in trees and graph. Using this technique any body can easily implement the concept of traversing on any complicated graph or on tree. This paper also opens the doors to utilize the set construct technique for the lots of more complicated algorithms and by this way they can be understood and operated in an easy and better way.

REFERENCES

[1] An Intoduction to Algorithms, Thomas H. Corman, PHI publication
[2] Data Structures Using C and C++, Tanenbaum, PHI publication
[3] Data Structures and Program Design in C, Kruse, Pearson Edition