Traversing a Binary Tree

Traversing Binary Tree

There are 3 ways of traversing a binary tree T having root R.

1. Preorder Traversing

• Steps:

(a) Process the root R

(b) Traverse the left subtree of R in preorder.

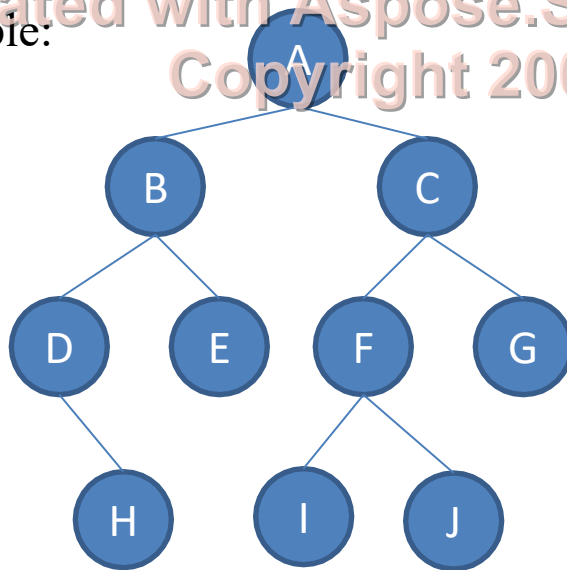(c) Traverse the right subtree of R in preorder.

• Example:

Figure: Binary Tree T

Preorder Traversal of T

A, B, D, H, E, C, F, I, J, G

2. Inorder Traversing

- Steps:

  (a) Traverse the left subtree of R in inorder.

  (b) Traverse the root R.

  (c) Traverse the right subtree of R in inorder.

- Example:
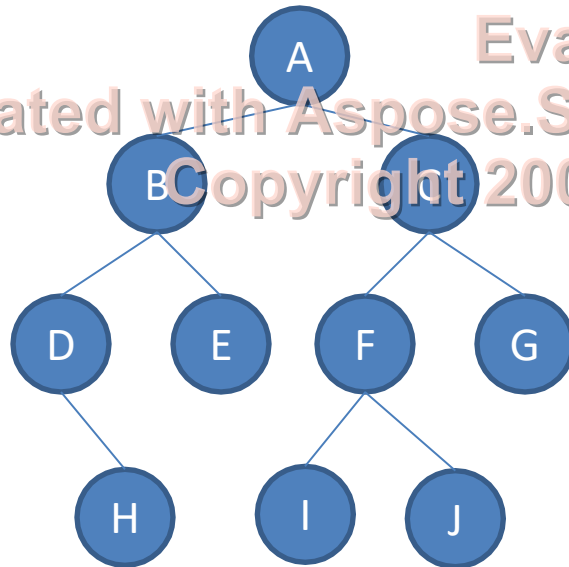
Inorder Traversal of T

D, H, B, E, A, I, F, J, C, G

Figure: Binary Tree T

3. Postorder Traversing

- Steps:

   (a) Traverse the left subtree of R in postorder.

   (b) Traverse the right subtree of R in postorder.

   (c) Traverse the root R.

- Example:
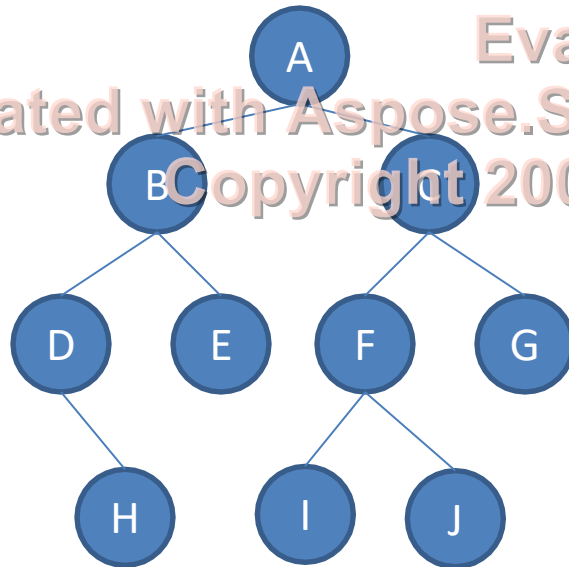
Postorder Traversal of T

H, D, E, B, I, J, F, G, C, A

Figure: Binary Tree T

4

Traversal Algorithms Using Stacks

Preorder Traversal Using Stack

Algorithm: Preorder_Traverse(Tree, Root, Stack)

(1)   Set Stack[0]=Null and Top=1 and Ptr=Root

(2)   Repeat steps (3) to (5) until Ptr ≠ NULL.

(3)   Process Ptr->Info.

(4)   if Ptr->Right ≠ NULL then set Stack[Top]=Ptr->Right and Top=Top+1

(5)   If Ptr->Left ≠ NULL then set Ptr=Ptr->Left

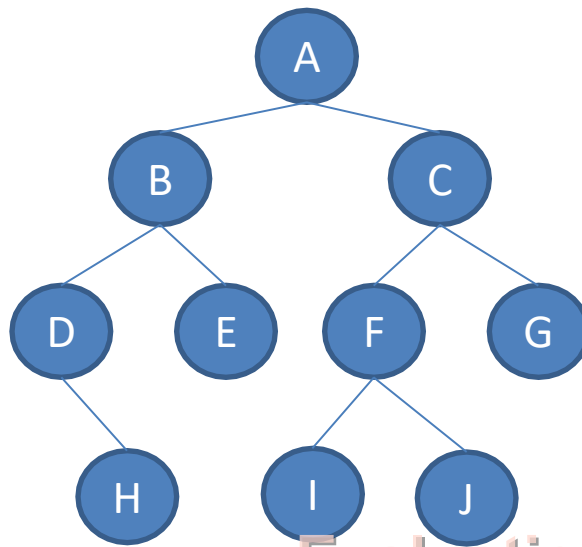   else Set Ptr=Stack[Top] and Top=Top-1

(6) Exit.

Example:

Figure: Binary Tree T

1. Initially, Ptr := A and Stack: Ø

2. Proceed down the left-most path rooted at Ptr = A.

   i.   Process A, Push C onto Stack.   Stack: Ø, C

   ii. Process B, Push E onto Stack.   Stack: Ø, C, E

   iii. Process D, Push H onto Stack.   Stack: Ø, C, E, H

3. Pop the Stack and Set Ptr := H.   Stack: Ø, C, E

4. Proceed down the left-most path rooted at Ptr = H

   i. Process H

5. Pop the Stack and Set Ptr := E   and   Stack: Ø, C

6. Proceed down the left-most path rooted at Ptr = E

    i. Process E

7. Pop the Stack and Set Ptr := C and   Stack: Ø

8. Proceed down the left-most path rooted at Ptr = C

    i.   Process C, Push G onto Stack.   Stack: Ø, G

    ii. Process F, Push J onto Stack.   Stack: Ø, G, J

    iii. Process I

9. Pop the Stack and Set Ptr := J and   Stack: Ø, G

10. Proceed down the left-most path rooted at Ptr = J

    i. Process J

11. Pop the Stack and Set Ptr := G   and Stack: Ø

12. Proceed down the left-most path rooted at Ptr = G

    i. Process G

13. Pop the Stack and set Ptr := Ø and Exit.

Preorder Traversal of T: A, B, D, H, E, C, F, I, J, G

2. Inorder Traversal Using Stack

Algorithm: Inorder_Traverse(Tree, Root, Stack)

(1)   Set Stack[0]=NULL and Top=1 and Ptr=Root

(2)   Repeat while Ptr ≠ NULL

        (a) Set Stack[Top]=Ptr and Top=Top+1

        (b) Set PTR=Ptr->Left

(3)   Set Ptr=Stack[Top] and Top = Top -1

(4)   Repeat steps 5 to 7 while Ptr ≠ NULL

(5)   Process Ptr->Info

(6)   If Ptr->Right ≠NULL then set Ptr=Ptr->Right and go to step 2.

(7)   Set Ptr=Stack[Top] and Top=Top-1

(8) Exit
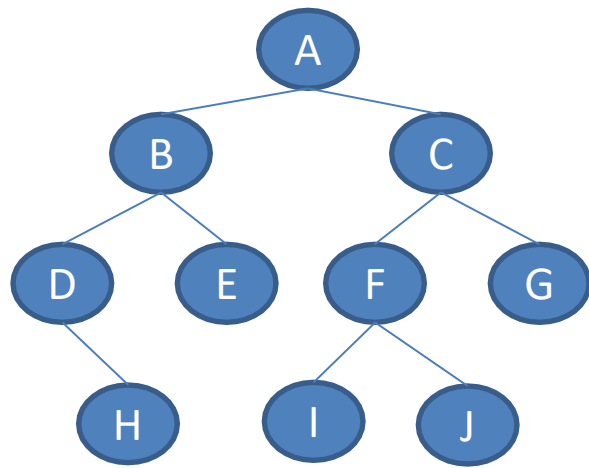
Example:

Figure: Binary Tree T

1. Initially Ptr := A and Stack: Ø

2. Proceed down the left-most path rooted at Ptr = A, pushing A, B, D onto Stack.

3.   Stack: Ø, A, B, D

4. Pop the Stack and Set Ptr := D.   Stack: Ø, A, B

5. Process D. Set Ptr := H. Proceed down the left-most path rooted at Ptr = H, pushing H onto Stack.   Stack: Ø, A, B, H

6. Pop the Stack and Set Ptr := H.   Stack: Ø, A, B

7. 6.   Process H.

8. Pop the Stack and Set Ptr := B.   Stack: Ø, A

9. Process B. Set Ptr:= E .Proceed down the left-most path rooted at Ptr = E, pushing E onto Stack.   Stack: Ø, A, E

10. Pop the Stack and Set Ptr := E.   Stack: Ø, A

11. Process E.

12. Pop the Stack and Set Ptr := A.   Stack: Ø

13. Process A. Set Ptr:= C. Proceed down the left-most path rooted at Ptr = C, pushing C, F, I onto Stack.   Stack: Ø, C, F, I

14. Pop the Stack. Set Ptr := I   .   Stack: Ø, C, F

15. Process I.

16. Pop the Stack. Set Ptr := F.   Stack: Ø, C, F

17. Process F. Set Ptr := J. Proceed down the left-most path rooted at Ptr = J, pushing J
     onto Stack.   Stack: Ø, C, J

18. Pop the Stack. Set Ptr := J.   Stack: Ø, C

19. Process J.

20. Pop the Stack. Set Ptr := C.   Stack: Ø

21. Process C. Set Ptr := G. Proceed down the left-most path rooted at Ptr = G, pushing G

    onto Stack.   Stack: Ø, G

22. Pop the Stack. Set Ptr := G.   Stack: Ø

23. Process G.

24. Pop the Stack. Set Ptr := Ø and Exit.

Inorder Traversal of T: D, H, B, E, A, I, F, J, C, G

Assignment

Write an algorithm that will traverse a binary tree in postorder traversal using stack. Discuss the algorithm using example.

END!!!