

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:

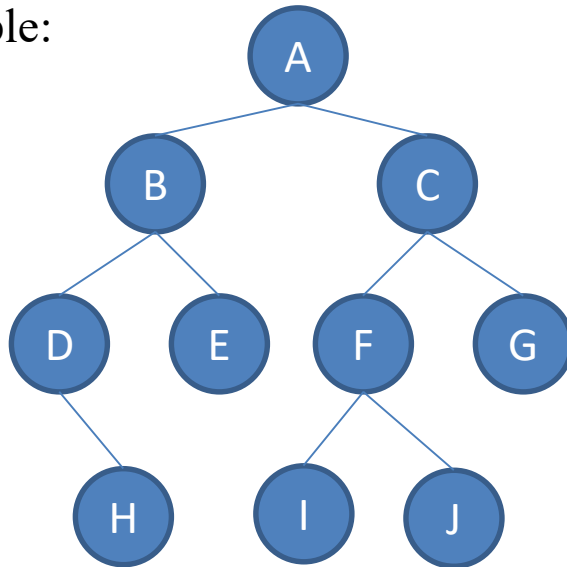


Figure: Binary Tree T

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:

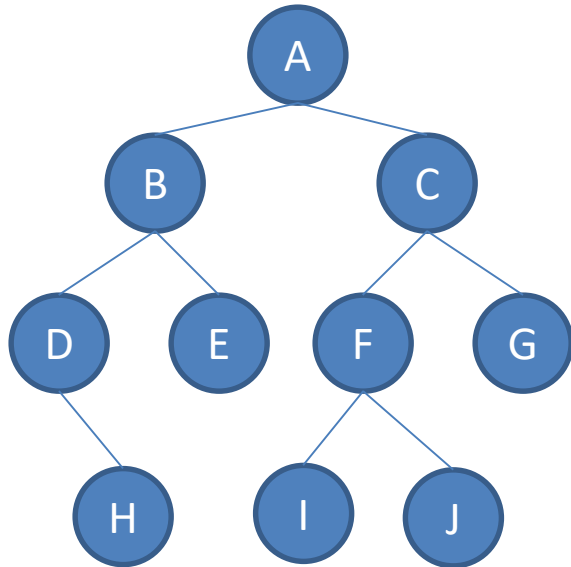


Figure: Binary Tree T

Inorder Traversal of T

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

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:

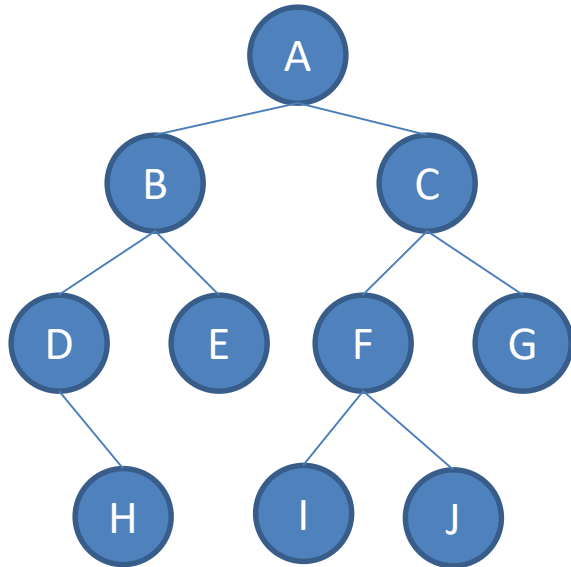


Figure: Binary Tree T

Postorder Traversal of T

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

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 \neq NULL
- (3) Process Ptr->Info.
- (4) if Ptr->Right \neq NULL then set Stack[Top]=Ptr->Right and Top=Top+1
- (5) If Ptr->Left \neq NULL then set Ptr=Ptr->Left
 else Set Ptr=Stack[Top] and Top=Top-1
- (6) Exit.

Example:

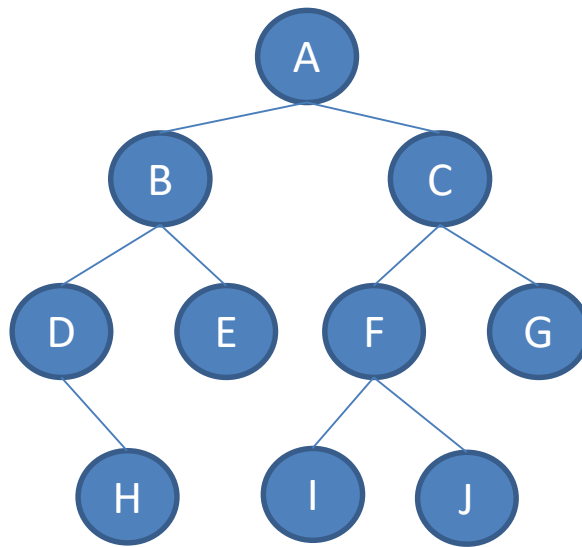


Figure: Binary Tree T

1. Initially $\text{Ptr} := A$ and $\text{Stack}: \emptyset$
2. Proceed down the left-most path rooted at $\text{Ptr} = A$
 - i. Process A, Push C onto Stack. $\text{Stack}: \emptyset, C$
 - ii. Process B, Push E onto Stack. $\text{Stack}: \emptyset, C, E$
 - iii. Process D, Push H onto Stack. $\text{Stack}: \emptyset, C, E, H$
3. Pop the Stack and Set $\text{Ptr} := H$. $\text{Stack}: \emptyset, C, E$
4. Proceed down the left-most path rooted at $\text{Ptr} = H$
 - i. Process H

5. Pop the Stack and Set $\text{Ptr} := E$ and Stack: \emptyset, C
6. Proceed down the left-most path rooted at $\text{Ptr} = E$
 - i. Process E
7. Pop the Stack and Set $\text{Ptr} := C$ and Stack: \emptyset
8. Proceed down the left-most path rooted at $\text{Ptr} = C$
 - i. Process C, Push G onto Stack. Stack: \emptyset, G
 - ii. Process F, Push J onto Stack. Stack: \emptyset, G, J
 - iii. Process I
9. Pop the Stack and Set $\text{Ptr} := J$ and Stack: \emptyset, G
10. Proceed down the left-most path rooted at $\text{Ptr} = J$
 - i. Process J
11. Pop the Stack and Set $\text{Ptr} := G$ and Stack: \emptyset
12. Proceed down the left-most path rooted at $\text{Ptr} = G$
 - i. Process G
13. Pop the Stack and set $\text{Ptr} := \emptyset$ 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 \neq 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 \neq NULL
- (5) Process Ptr->Info
- (6) If Ptr->Right \neq NULL then set Ptr=Ptr->Right and go to step 2.
- (7) Set Ptr=Stack[Top] and Top=Top-1
- (8) Exit

Example:

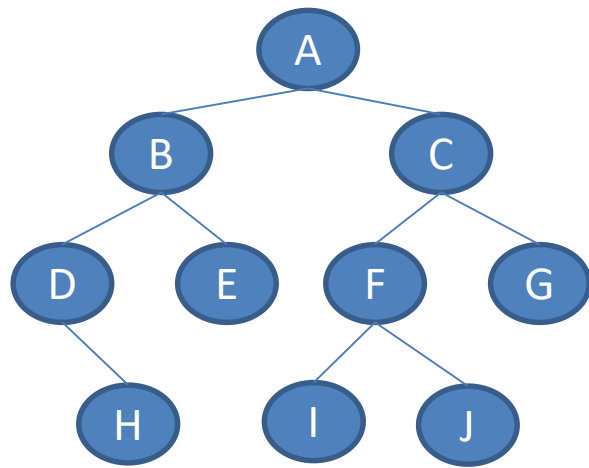


Figure: Binary Tree T

1. Initially $\text{Ptr} := A$ and $\text{Stack}: \emptyset$
2. Proceed down the left-most path rooted at $\text{Ptr} = A$, pushing A, B, D onto Stack.
3. $\text{Stack}: \emptyset, A, B, D$
4. Pop the Stack and Set $\text{Ptr} := D$. $\text{Stack}: \emptyset, A, B$
5. Process D. Set $\text{Ptr} := H$. Proceed down the left-most path rooted at $\text{Ptr} = H$, pushing H onto Stack. $\text{Stack}: \emptyset, A, B, H$
6. Pop the Stack and Set $\text{Ptr} := H$. $\text{Stack}: \emptyset, A, B$
7. 6. Process H.
8. Pop the Stack and Set $\text{Ptr} := B$. $\text{Stack}: \emptyset, A$
9. Process B. Set $\text{Ptr} := E$. Proceed down the left-most path rooted at $\text{Ptr} = E$, pushing E onto Stack. $\text{Stack}: \emptyset, A, E$

10. Pop the Stack and Set $\text{Ptr} := \text{E}$. Stack: \emptyset, A
11. Process E.
12. Pop the Stack and Set $\text{Ptr} := \text{A}$. Stack: \emptyset
13. Process A. Set $\text{Ptr} := \text{C}$. Proceed down the left-most path rooted at $\text{Ptr} = \text{C}$, pushing C, F, I onto Stack. Stack: $\emptyset, \text{C}, \text{F}, \text{I}$
14. Pop the Stack. Set $\text{Ptr} := \text{I}$. Stack: $\emptyset, \text{C}, \text{F}$
15. Process I.
16. Pop the Stack. Set $\text{Ptr} := \text{F}$. Stack: $\emptyset, \text{C}, \text{F}$
17. Process F. Set $\text{Ptr} := \text{J}$. Proceed down the left-most path rooted at $\text{Ptr} = \text{J}$, pushing J onto Stack. Stack: $\emptyset, \text{C}, \text{J}$
18. Pop the Stack. Set $\text{Ptr} := \text{J}$. Stack: \emptyset, C
19. Process J.
20. Pop the Stack. Set $\text{Ptr} := \text{C}$. Stack: \emptyset
21. Process C. Set $\text{Ptr} := \text{G}$. Proceed down the left-most path rooted at $\text{Ptr} = \text{G}$, pushing G onto Stack. Stack: \emptyset, G
22. Pop the Stack. Set $\text{Ptr} := \text{G}$. Stack: \emptyset
23. Process G.
24. Pop the Stack. Set $\text{Ptr} := \emptyset$ 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!!!