Construction of unique Binary tree

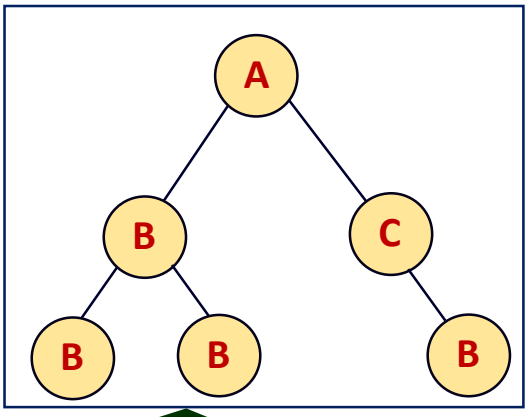
Let us suppose one of your friends **wants to send you a binary tree via a network.** So how he/she will send the binary tree?

Don't think that just take a snap-shot of that binary tree and send the picture. **We are not studying data structure to do that.** We have to think optimal.

We cannot just send the picture because this will waste the network bandwidth unnecessarily as **sending a picture consume more network bandwidth.**

So more optimal way is to write the traversal of binary tree in a text file and then send it. This will save a lot of network bandwidth.

Let us see the difference



Size of this image is approximate 16-18 KB

Inorder Travesal : 1234567

This will take 26 byte when stored in a text file

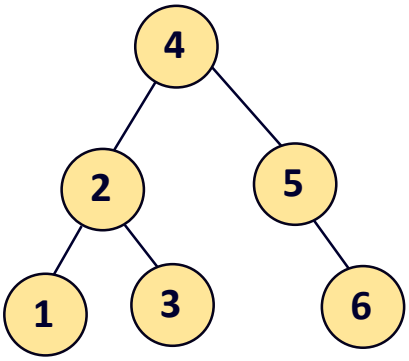
So your friend **must opt the second method** which is more optimal. Now let us suppose that you have received the inorder traversal. You don't know how original binary tree look alike. So can you able to reconstruct the original binary tree? If yes. Then please go ahead. ☺

Actually we cannot reconstruct original binary tree if we have only inorder traversal available. In general, **a inorder traversal does not uniquely define the structure of the tree.**

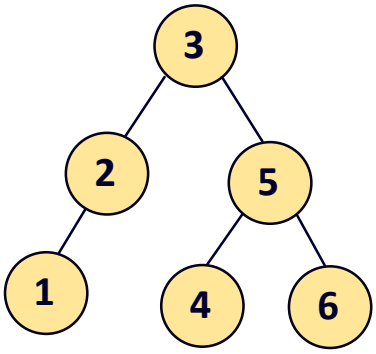
Let us see why we cannot reconstruct original binary tree if we have only inorder traversal available.

Inorder traversal that you have received from your friend is: **1 2 3 4 5 6 7**

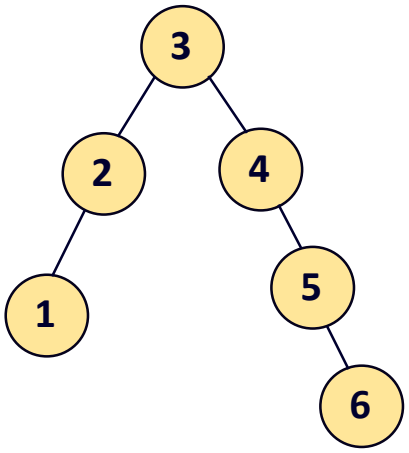
For this inorder traversal **multiple binary tree exist.** So this will **cause ambiguity that which binary tree your friend had sent to you.**



Inorder :1 2 3 4 5 6 7



Inorder :1 2 3 4 5 6 7



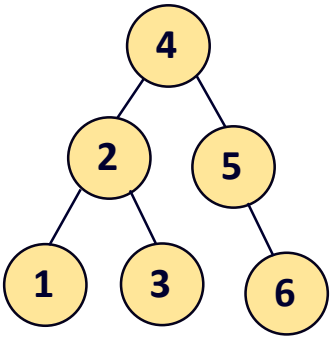
Inorder :1 2 3 4 5 6 7

So **which binary tree out of these three, your friend had sent you**. This cannot be predicted. The only information you had, is inorder traversal. But **more than one binary tree has same in-order traversal.**

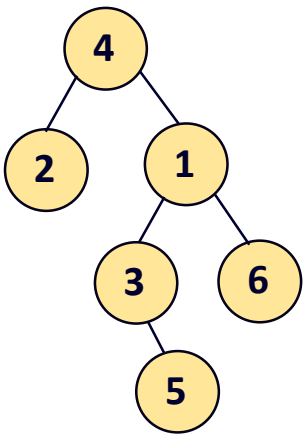
So unique binary tree cannot be constructed from a given inorder traversal.

Now let us suppose **instead of sending inorder traversal**, your friend had **sent you a preorder traversal**. Now you have only the information is **preorder traversal as : 4 2 1 3 5 6**

So now can you able to reconstruct the original binary tree? No. let us see why?



Preorder : 4 2 1 3 5 6



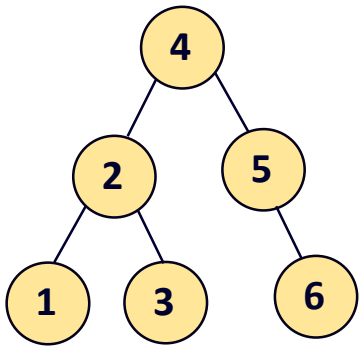
Preorder walk :

4 2 1 3 5 6

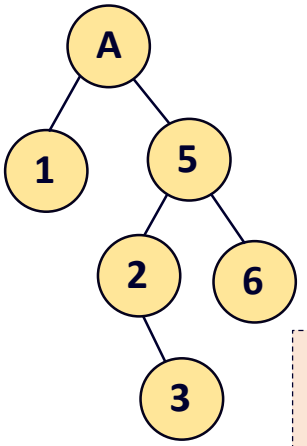
Both the tree have the same pre-order traversal. So the same ambiguity is also present in case of pre-order traversal.

So unique binary tree cannot be constructed from a given pre-order traversal.

Same ambiguity is present in case of post-order traversal also.



Postorder : 1 3 2 6 5 4



Postorder walk

1 3 2 6 5 4

So **we cannot construct unique binary tree** from a given **in-order or pre-order or post-order traversal.**

To uniquely construct a Binary Tree, In-order traversal together with either Post-order or Pre-order must be given.

We will discuss how to construct a unique binary tree from the given traversals if In-order traversal together with either Post-order or Pre-order must be given.

BST Construction from Given Traversals

As we know **we cannot construct unique binary tree** if **any one out of below** traversals are given individually.

- In-order Traversal
- Pre-order Traversal
- Post-order Traversal

To construct a unique binary Inorder traversal together with either Postorder or Preorder must be given.

Let's suppose one or two traversal are given, whether we can construct **Binary tree or not**.

Let's Rephrase.

We can Construct Unique Binary Tree if following are given:-

- **Inorder and Preorder.**
- **Inorder and Postorder.**
- **Inorder and Level-order.**

We cannot Construct Unique Binary tree if following are given:-

- **Pre-order Traversal**
- **Post-order Traversal**
- **Level-order Travesal**
- **In-order Traversal**
- **Postorder and Preorder.**
- **Levelorder and Postorder.**

So if any one out of four (Pre, Post, in-order or Level) are given, Binary Tree cannot be constructed.

Construction of unique BST from Pre-order and in-order

Let us suppose **we have given in-order and pre-order traversal of a binary tree.** We don't know how original binary tree look alike. **So we'll try to reconstruct original binary tree from given in-order and pre-order traversal.**

Example

You are given two traversal Inorder and Preorder of Binary seach tree as:
Inorder : g d h b e a f j c Preorder : a b d g h e c f j
What is height of original binary tree?

Solution

Inorder : g d h b e a f j c
Preorder : a b d g h e c f j

As we know first element in preorder traversal is always a root node. So we got our first clue that a is root tree.

Step 1

Scan the preorder traversal from left to right one element at a time. Use that currently scanned element in in-order traversal to get its left subtree and right subtree.

Pre-order : **a b d g h e c f j**

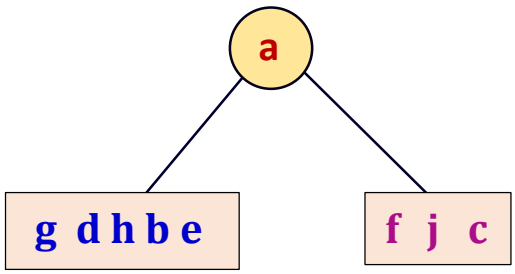
Scanning Preorder Traversal from left to right

a is root node so use element a in in-order traversal to get its right and left subtree.

In-order : g d h b e a f j c

a is the root of the tree

g d h b e are in the left subtree of a **f j c are in the right subtree of a**



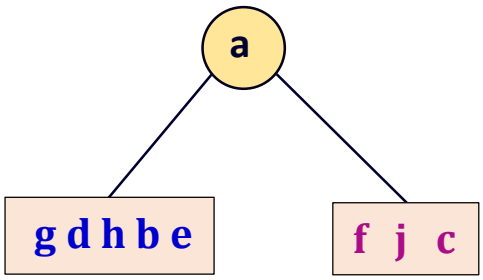
Step 2

Next element under scan is b. Use currently scanned **element b** in left sub-tree of a to get further its left sub-tree and right sub-tree.

Preorder : **a b d g h e c f j**

Current element under scan i.e **element b**

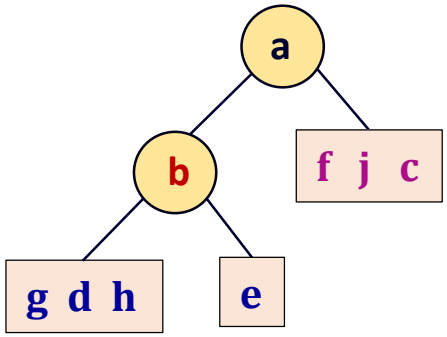
b is root node in the left subtree of a.

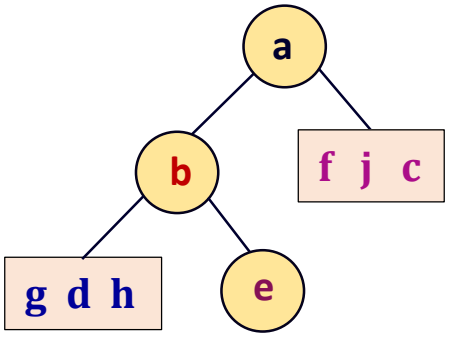


Left Subtree of a : g d h b e

b is the root

g d h are in the left subtree of b **e i is in the right subtree of b**

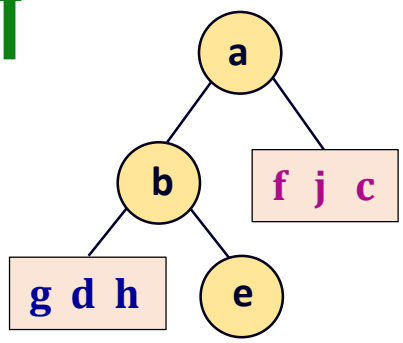




Step 3

Next element under scan is d. Use currently scanned **element d** in left sub-tree of b to get further its left sub-tree and right sub-tree.

Preorder : **a b d g h e c f j**

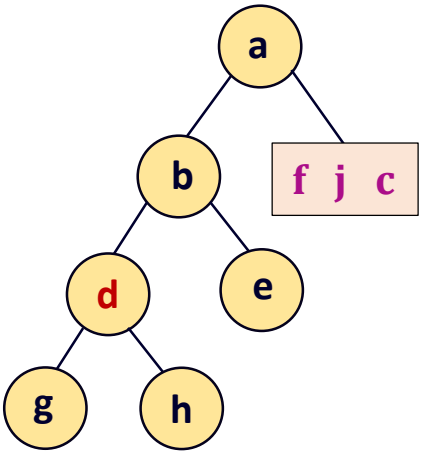


d is root node in the left subtree of b.

Left Subtree of b : **g d h**

d is the root

g is in the left subtree of d **h is in the right subtree of d**

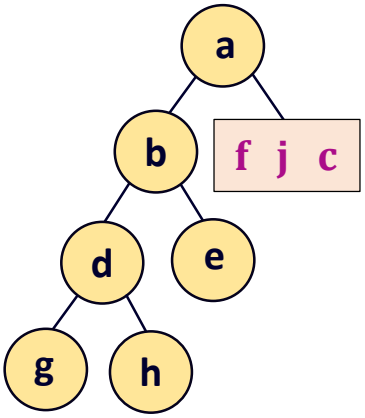


Step 4

Next scanned element is g , and **g has no left or right child.** So stop and go for scanning next element.

Preorder : **a b d g h e c f j**

g is a single value node. Node g has no left or right child. So move to next step.

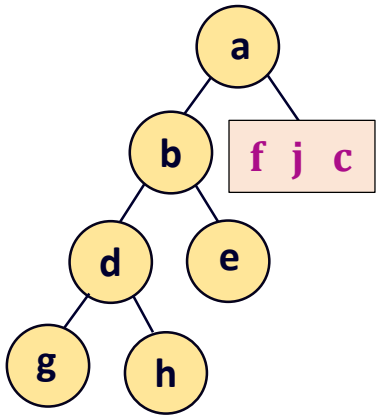


Step 5,6

Next scanned elements are h and e. Element h and e has no left or right child. So stop and go for scanning next element.

In Step 5 Preorder : **a b d g h e c f j**

In Step 6 Preorder : **a b d g h e c f j**



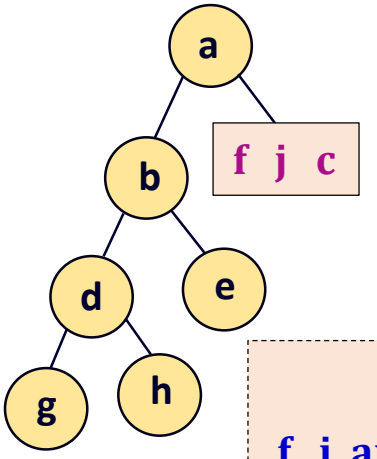
Step 7

Next element under scan is c. Use currently scanned **element c** in right sub-tree of a to get further its left sub-tree and right sub-tree.

Preorder : **a b d g h e c f j**

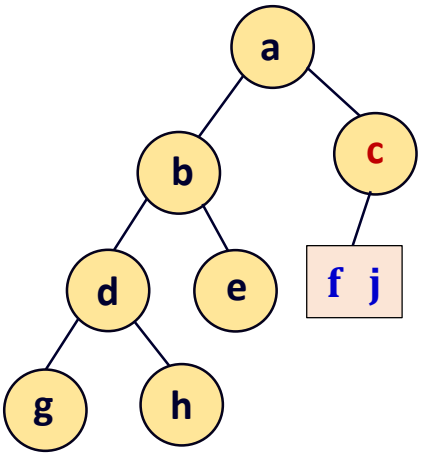
c is root node in the right subtree of a.

Right Subtree of a : f j c



c is the root

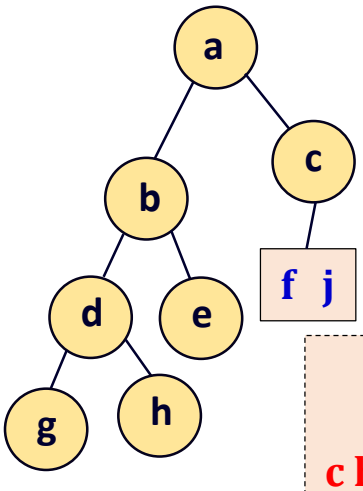
f j are in the left subtree of c **c has no right subtree**



Step 8

Next element under scan is f. Use currently scanned **element f** in left sub-tree of c to get further its left sub-tree and right sub-tree(if any).

Preorder : **a b d g h e c f j**

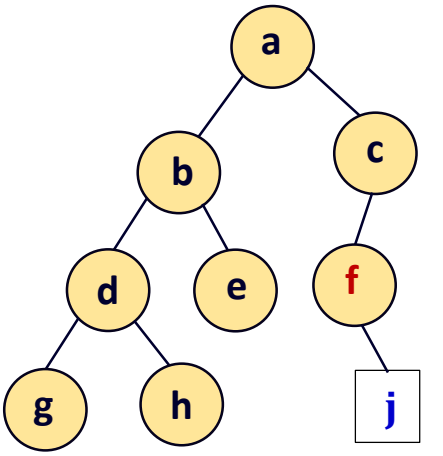


f is root node in the left subtree of c.

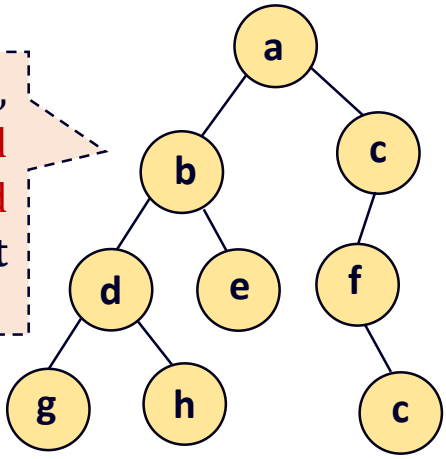
Right Subtree of c : f j

f is the root

c has no right child **j is in the right subtree of f**



This is original binary tree, which we have reconstructed from given inorder and preorder traversal. **So height of this binary tree is 3.**



Example 2

Construct a binary tree whose preorder traversal is 5 7 4 8 6 1 9 2 3 and inorder traversal is 4 7 5 6 1 8 2 9 3. The post-order traversal of the binary tree is ?

(A) 4 7 1 6 2 3 9 8 5

(B) 4 7 1 6 2 3 9 8 5

(C) 4 7 1 6 2 3 9 8 5

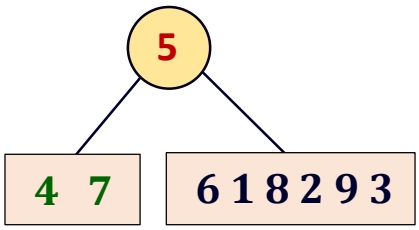
(D) 4 7 1 6 2 3 9 8 5

Solution

Step 1

Preorder traversal : 5 7 4 8 6 1 9 2 3

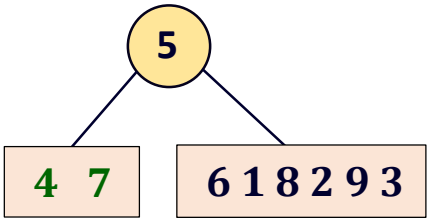
Inorder traversal : 4 7 5 6 1 8 2 9 3

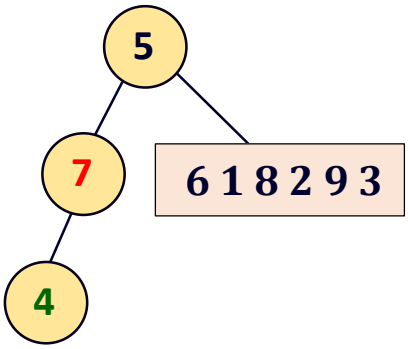


Step 2

Preorder traversal : 5 7 4 8 6 1 9 2 3

Left subtree of 5: 4 7

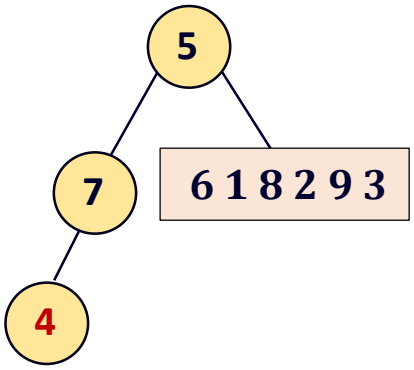




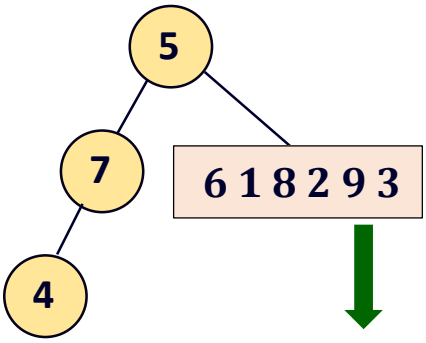
Step 3

Preorder traversal : 5 7 4 8 6 1 9 2 3

Node 4 don't have any child (right or left)



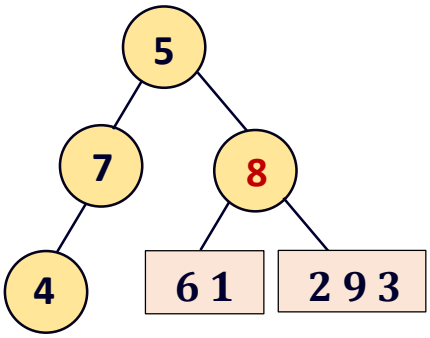
Step 4



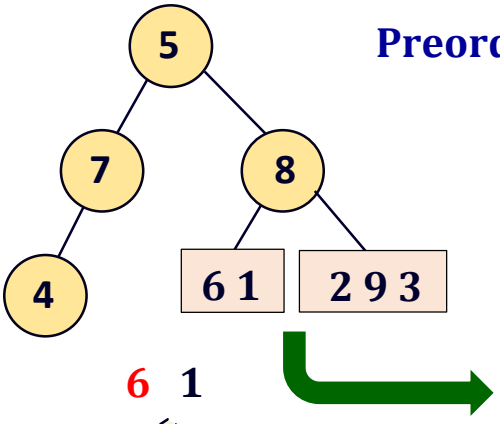
Preorder traversal : 5 7 4 8 6 1 9 2 3

6 1 8 2 9 3

Right subtree of node 5.
So 8 become the root in
Right Subtree of node 5



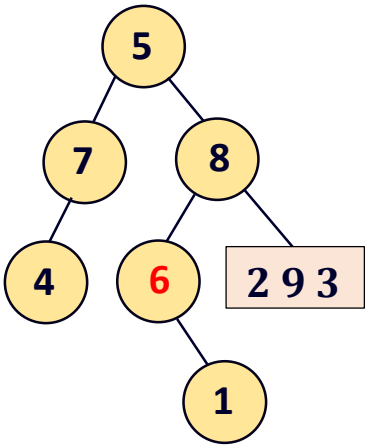
Step 5



Preorder traversal : 5 7 4 8 6 1 9 2 3

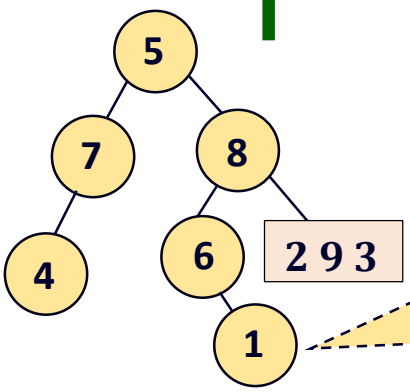
6 1

Left subtree of node 8.
So 6 become the root in
left Subtree of node 8



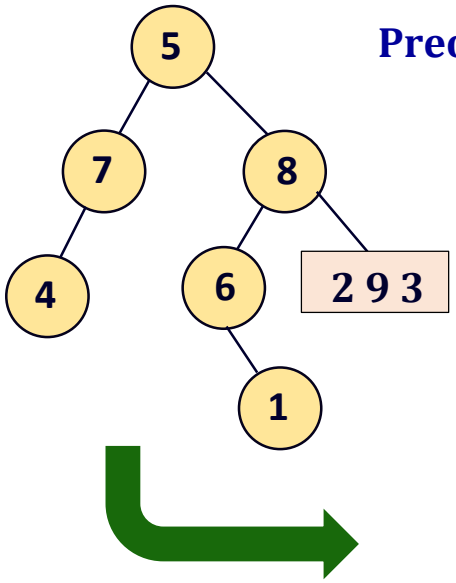
Step 6

Preorder traversal : 5 7 4 8 6 1 9 2 3



Node 1 don't have any child (right or left),so move to next step

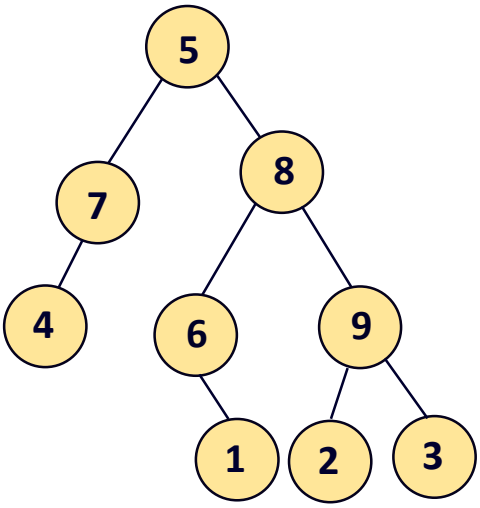
Step 7



Preorder traversal : 5 7 4 8 6 1 9 2 3

Right subtree of node 8.
Node 9 become the root
in Right Subtree of node 5

2 9 3



This is final binary tree which we have reconstructed from given inorder and preorder.

So post-order traversal is given as:

4 7 1 6 2 3 9 8 5

Example

The inorder traversal of a binary tree is d b e a f c g and pre-order traversal of same binary tree is as: a b d e c f g. The post-order traversal of the binary tree is

(A) d e b f g c a

(B) e d b g f c a

(C) e d b f g c a

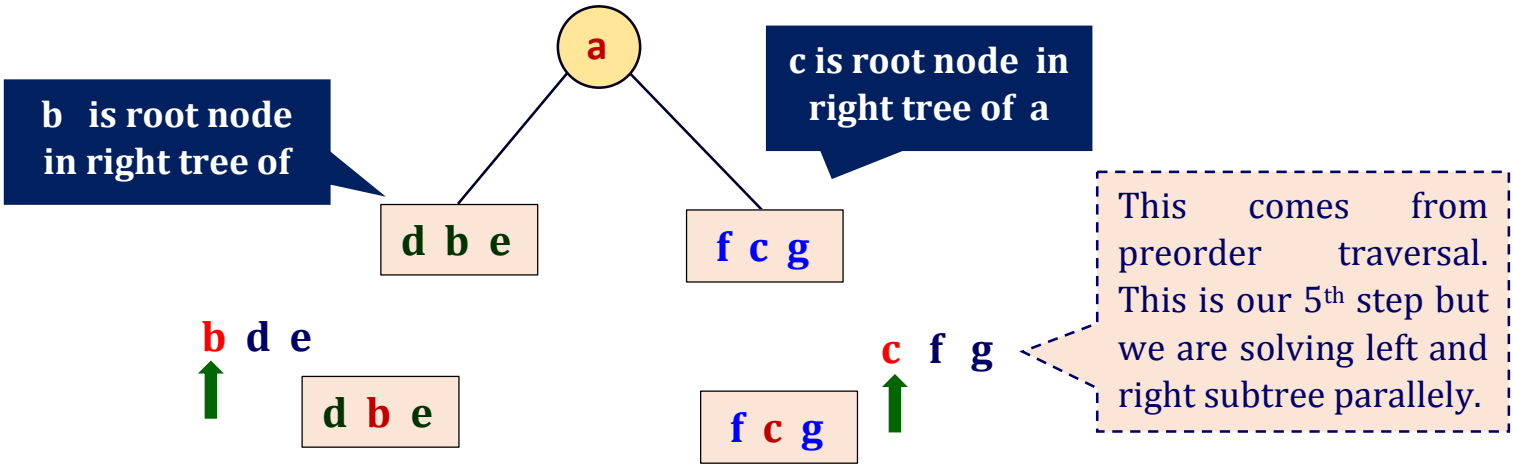
(D) d e f g b c a

Solution

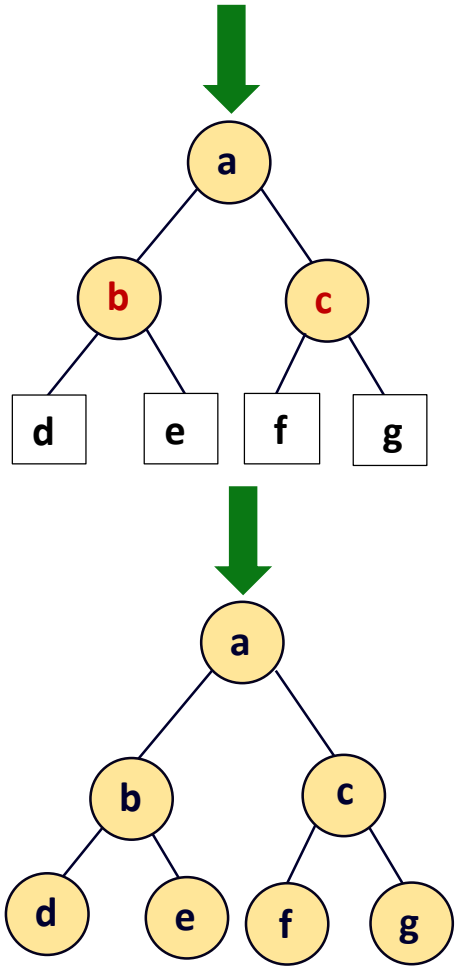
First Try to solve this Example by yourself. Don't worry its solution is also given. We will follow the same approach to solve this problem also as we have solved above two problems but for a competition point of view we have to choose a fast approach to solve any king problem. **So see how to do it fast.**

Pre-order Traversal : a b d e c f g

Inorder Traversal : d b e a f c g



Right subtree of **node a** contain **three element f c and g.** Now for these three elements, write the same element from inorder traversal in the order as they present in inorder traversal. (Both for left and right subtree of a)



Post-order traversal of above binary tree : d e b f g c a

So correct option is A

Construction of unique BST from Postorder and inorder

Construction of binary tree from given Postorder and inorder traversal is same as that constructing binary tree from preorder and inorder, but in case of postorder traversal we scan the given traversal from **right to left** and in case of preorder we scan the given traversal from **left to right.** So I will not discuss in much detail. We will solve the right and left subtree in parallel. I hope you will not face any problem.

Example:-

You are given two traversal Inorder and Postorder of Binary tree as:

Inorder : 3 2 8 5 7 9 6 4 1 Postorder : 2 3 5 8 6 1 4 9 7

Binary search tree with its Preorder traversal is ?

(A) d e b f g c a (B) e d b g f c a

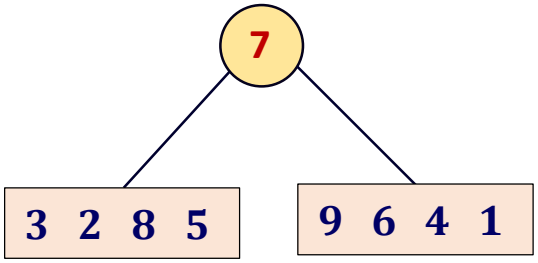
(C) e d b f g c a (D) d e f g b c a

Solution

As we know last element in preorder traversal is always a root node. So we got our first clue that a is root tree.

Postorder : 2 3 5 8 6 1 4 9 7

Inorder : 3 2 8 5 7 9 6 4 1



From given postorder

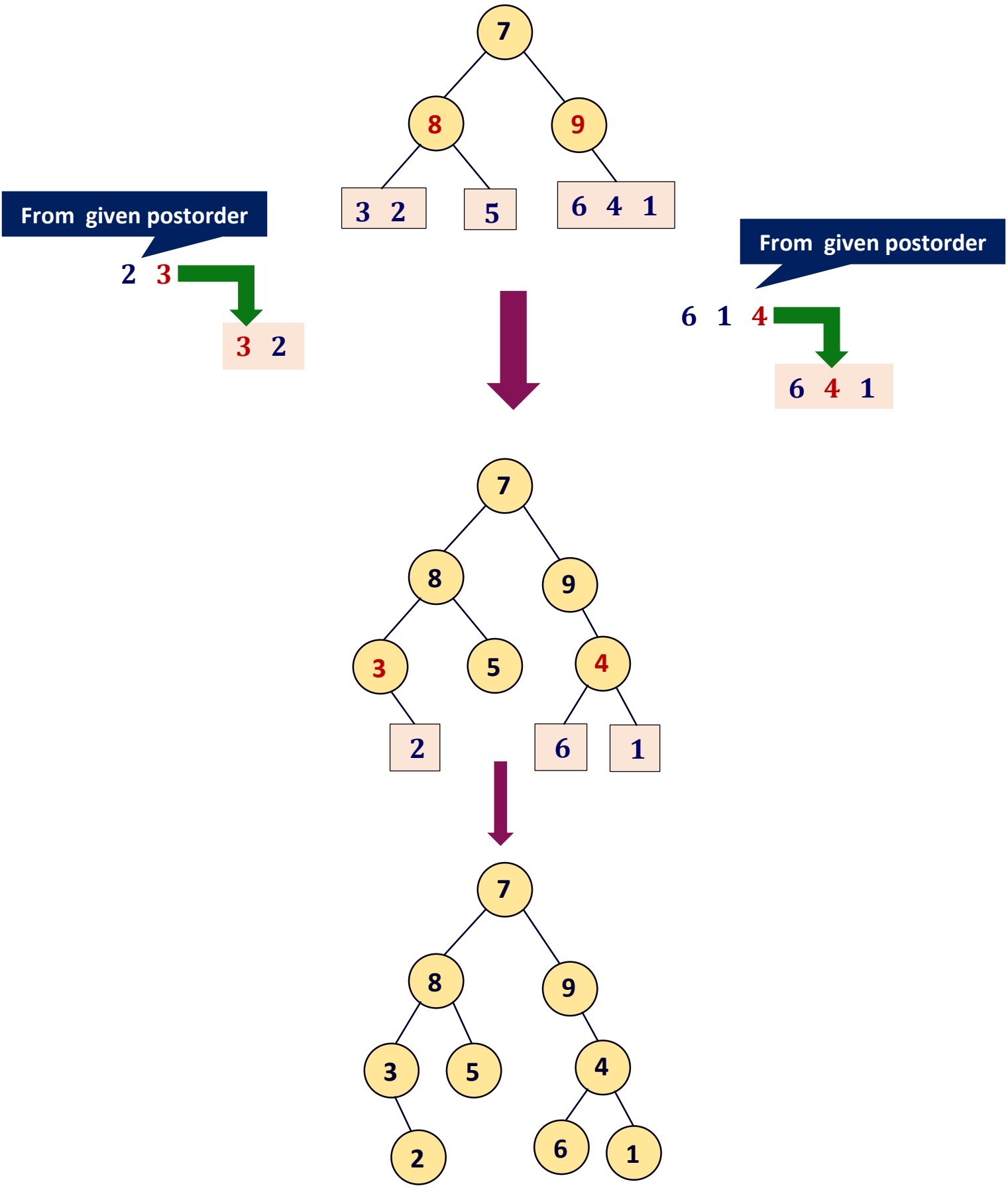
2 3 5 8

3 2 8 5

From given postorder

6 1 4 9

9 6 4 1



Preorder traversal is: 7 8 3 2 5 9 4 6 1

✓ **So the correct option is B**