GeeksforGeeks Q&A



GeeksforGeeks





f Like Share Sign Up to see what your friends like.

Construct the cartesian tree

Given an inorder traversal of a cartesian tree, construct the tree.

Cartesian tree: is a heap ordered binary tree, where the root is greater than all the elements in the subtree.

Note: You may assume that duplicates do not exist in the tree.

Example:

```
Input : [1 2 3]
Return:
          3
        2
      1
asked Oct 11, 2015 by Swanky
📭 flag - 📭 Answer - 💬 comment
```

1 Answer



Inorder traversal: (Left tree) root (Right tree)

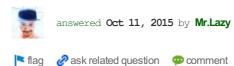
Note that the root is the max element in the whole array. Based on the info, It's easy to figure out the position of the root in inorder traversal. Thus, we can separate out the elements which go in the left subtree and right subtree. Once you have the inorder traversal for left subtree, you can recursively solve for left subtree. Same for right subtree.



```
// Author :: Gaurav Ahirwar
#include<bits/stdc++.h>
using namespace std;
struct node
    int data;
    struct node* left, *right;
typedef struct node* Node;
typedef struct node node;
Node newNode(int data) {
    Node temp = (Node)malloc(sizeof(struct node));
    temp->data = data;
temp->left = temp->right = NULL;
    return(temp);
}
void inorder(Node root) {
    if(!root) return;
    inorder(root->left);
    cout << root->data << " ";</pre>
    inorder(root->right);
}
Node buildTree(int in[], int start, int end) {
    if (start == end) {
        return newNode(in[start]);
    if (start > end) return NULL;
    // find max which will be the root.
    int maxVal = INT_MIN, maxIndex = -1;
    for (int i = start; i <= end; i++) {</pre>
        if (in[i] > maxVal) {
             maxVal = in[i];
             maxIndex = i;
        }
    }
    Node root = newNode(maxVal);
    root->left = buildTree(in, start, maxIndex - 1);
    root->right = buildTree(in, maxIndex + 1, end);
    return root;
}
Node solve(int in[], int n) {
    if (n == 0) return NULL;
    return buildTree(in, 0, n - 1);
}
int main() {
    int in[] = {1, 2, 3};
int n = sizeof in / sizeof in[0];
    Node root = solve(in, n);
    inorder(root);
    return 0;
}
```

Run on IDE

?

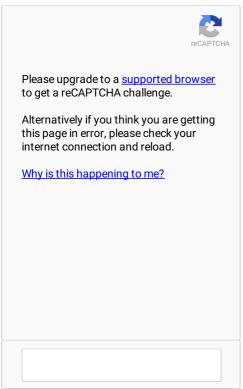


Your answer

Writing Code? Use the insert code snippet button (4th from last) for syntax highlighting



Paragraphs: 0, Words: 0	4
Your name to display (optional):	
☐ Email me at this address if my answer is selected or commented on:	
Privacy: Your email address will only be used for sending these notifications.	
Timady. Total official addition office for containing those floationion.	
Anti-spam verification:	



To avoid this verification in future, please $\underline{\text{log in}}$ or $\underline{\text{register}}$.

Add answer

