Construct Special Binary Tree from given Inorder traversal

Given Inorder Traversal of a Special Binary Tree in which key of every node is greater than keys in left and right children, construct the Binary Tree and return root.

Examples:

```
Input: inorder[] = {5, 10, 40, 30, 28}
Output: root of following tree
         40
       /   \
      10     30
     /         \
    5          28

Input: inorder[] = {1, 5, 10, 40, 30,
                    15, 28, 20}
Output: root of following tree
          40
        /   \
       10     30
      /         \
     5          28
    /          /  \
   1         15    20
```

The idea used in Construction of Tree from given Inorder and Preorder traversals can be used here. Let the given array is {1, 5, 10, 40, 30, 15, 28, 20}. The maximum element in given array must be root. The elements on left side of the maximum element are in left subtree and elements on right side are in right subtree.

```
        40
      /       \
   {1,5,10}   {30,15,28,20}
```

We recursively follow above step for left and right subtrees, and finally get the following tree.

```
         40
       /   \
      10     30
     /         \
    5          28
   /          /  \
   1         15    20
```

Algorithm: buildTree()
1) Find index of the maximum element in array. The maximum element must be root of Binary Tree.
2) Create a new tree node 'root' with the data as the maximum value found in step 1.
3) Call buildTree for elements before the maximum element and make the built tree as left subtree of 'root'.
5) Call buildTree for elements after the maximum element and make the built tree as right subtree of 'root'.
6) return 'root'.

Implementation: Following is the implementation of the above algorithm.

C/C++

```
/* program to construct tree from inorder traversal */
#include<stdio.h>
#include<stdlib.h>

/* A binary tree node has data, pointer to left child
```

```
   and a pointer to right child */
struct node
{
    int data;
    struct node* left;
    struct node* right;
};

/* Prototypes of a utility function to get the maximum
   value in inorder[start..end] */
int max(int inorder[], int strt, int end);

/* A utility function to allocate memory for a node */
struct node* newNode(int data);

/* Recursive function to construct binary of size len from
   Inorder traversal inorder[]. Initial values of start and end
   should be 0 and len -1.  */
struct node* buildTree (int inorder[], int start, int end)
{
    if (start > end)
        return NULL;

    /* Find index of the maximum element from Binary Tree */
    int i = max (inorder, start, end);

    /* Pick the maximum value and make it root */
    struct node *root = newNode(inorder[i]);

    /* If this is the only element in inorder[start..end],
       then return it */
    if (start == end)
        return root;

    /* Using index in Inorder traversal, construct left and
       right subtress */
    root->left  = buildTree (inorder, start, i-1);
    root->right = buildTree (inorder, i+1, end);

    return root;
}

/* UTILITY FUNCTIONS */
/* Function to find index of the maximum value in arr[start...end] */
int max (int arr[], int strt, int end)
{
    int i, max = arr[strt], maxind = strt;
    for(i = strt+1; i <= end; i++)
    {
        if(arr[i] > max)
        {
            max = arr[i];
            maxind = i;
        }
    }
    return maxind;
}

/* Helper function that allocates a new node with the
   given data and NULL left and right pointers. */
struct node* newNode (int data)
{
    struct node* node = (struct node*)malloc(sizeof(struct node));
    node->data = data;
    node->left = NULL;
    node->right = NULL;

    return node;
}

/* This funtcion is here just to test buildTree() */
void printInorder (struct node* node)
{
```

```
    if (node == NULL)
        return;

    /* first recur on left child */
    printInorder (node->left);

    /* then print the data of node */
    printf("%d ", node->data);

    /* now recur on right child */
    printInorder (node->right);
}

/* Driver program to test above functions */
int main()
{
   /* Assume that inorder traversal of following tree is given
         40
       /   \
      10     30
     /         \
    5          28 */

    int inorder[] = {5, 10, 40, 30, 28};
    int len = sizeof(inorder)/sizeof(inorder[0]);
    struct node *root = buildTree(inorder, 0, len - 1);

    /* Let us test the built tree by printing Insorder traversal */
    printf("\n Inorder traversal of the constructed tree is \n");
    printInorder(root);
    return 0;
}
```

Java

```
// Java program to construct tree from inorder traversal

/* A binary tree node has data, pointer to left child
   and a pointer to right child */
class Node
{
    int data;
    Node left, right;

    Node(int item)
    {
        data = item;
        left = right = null;
    }
}

class BinaryTree
{
    Node root;

    /* Recursive function to construct binary of size len from
       Inorder traversal inorder[]. Initial values of start and end
       should be 0 and len -1.  */
    Node buildTree(int inorder[], int start, int end, Node node)
    {
        if (start > end)
            return null;

        /* Find index of the maximum element from Binary Tree */
        int i = max(inorder, start, end);

        /* Pick the maximum value and make it root */
        node = new Node(inorder[i]);

        /* If this is the only element in inorder[start..end],
```

```
         then return it */
        if (start == end)
            return node;

        /* Using index in Inorder traversal, construct left and
         right subtress */
        node.left = buildTree(inorder, start, i - 1, node.left);
        node.right = buildTree(inorder, i + 1, end, node.right);

        return node;
    }

    /* UTILITY FUNCTIONS */

    /* Function to find index of the maximum value in arr[start...end] */
    int max(int arr[], int strt, int end)
    {
        int i, max = arr[strt], maxind = strt;
        for (i = strt + 1; i <= end; i++)
        {
            if (arr[i] > max)
            {
                max = arr[i];
                maxind = i;
            }
        }
        return maxind;
    }

    /* This funtcion is here just to test buildTree() */
    void printInorder(Node node)
    {
        if (node == null)
            return;

        /* first recur on left child */
        printInorder(node.left);

        /* then print the data of node */
        System.out.print(node.data + " ");

        /* now recur on right child */
        printInorder(node.right);
    }

    public static void main(String args[])
    {
        BinaryTree tree = new BinaryTree();

        /* Assume that inorder traversal of following tree is given
             40
            /   \
          10     30
         /         \
        5          28 */
        int inorder[] = new int[]{5, 10, 40, 30, 28};
        int len = inorder.length;
        Node mynode = tree.buildTree(inorder, 0, len - 1, tree.root);

        /* Let us test the built tree by printing Inorder traversal */
        System.out.println("Inorder traversal of the constructed tree is ");
        tree.printInorder(mynode);
    }
}

// This code has been contributed by Mayank Jaiswal
```

Output:

```
 Inorder traversal of the constructed tree is
 5 10 40 30 28
```

Time Complexity: O(n^2)