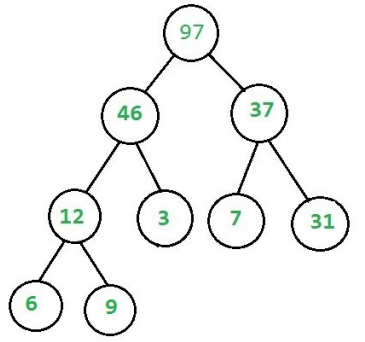
Check if a given Binary Tree is Heap

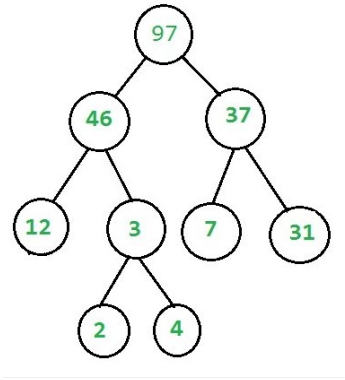
Given a binary tree we need to check it has heap property or not, Binary tree need to fulfill following two conditions for being a heap –

1. It should be a complete tree (i.e. all levels except last should be full).
2. Every node's value should be greater than or equal to its child node (considering max-heap).

For example this tree contains heap property –



While this doesn't –



We strongly recommend you to minimize your browser and try this yourself first.

We check each of the above condition separately, for checking completeness isComplete and for checking heap isHeapUtil function are written.
Detail about isComplete function can be found here.

isHeapUtil function is written considering following things –

1. Every Node can have 2 children, 0 child (last level nodes) or 1 child (there can be at most one such node).
2. If Node has No child then it's a leaf node and return true (Base case)
3. If Node has one child (it must be left child because it is a complete tree) then we need to compare this node with its single child only.
4. If Node has both child then check heap property at Node at recur for both subtrees.
   Complete code.

Below is C implementation.

```
/* C program to checks if a binary tree is max heap ot not */
#include<stdio.h>
#include<stdlib.h>
#include<stdbool.h>

/*  Tree node structure */
struct Node
{
    int key;
    struct Node *left;
```

```
    struct Node *right;
};

/* Helper function that allocates a new node */
struct Node *newNode(int k)
{
    struct Node *node = (struct Node*)malloc(sizeof(struct Node));
    node->key = k;
    node->right = node->left = NULL;
    return node;
}

/* This function counts the number of nodes in a binary tree */
unsigned int countNodes(struct Node* root)
{
    if (root == NULL)
        return (0);
    return (1 + countNodes(root->left) + countNodes(root->right));
}

/* This function checks if the binary tree is complete or not */
bool isCompleteUtil (struct Node* root, unsigned int index,
                     unsigned int number_nodes)
{
    // An empty tree is complete
    if (root == NULL)
        return (true);

    // If index assigned to current node is more than
    // number of nodes in tree, then tree is not complete
    if (index >= number_nodes)
        return (false);

    // Recur for left and right subtrees
    return (isCompleteUtil(root->left, 2*index + 1, number_nodes) &&
            isCompleteUtil(root->right, 2*index + 2, number_nodes));
}

// This Function checks the heap property in the tree.
bool isHeapUtil(struct Node* root)
{
    //  Base case : single node satisfies property
    if (root->left == NULL && root->right == NULL)
        return (true);

    //  node will be in second last level
    if (root->right == NULL)
    {
        //  check heap property at Node
        //  No recursive call , because no need to check last level
        return (root->key >= root->left->key);
    }
    else
    {
        //  Check heap property at Node and
        //  Recursive check heap property at left and right subtree
        if (root->key >= root->left->key &&
            root->key >= root->right->key)
            return ((isHeapUtil(root->left)) &&
                    (isHeapUtil(root->right)));
        else
            return (false);
    }
}

//  Function to check binary tree is a Heap or Not.
bool isHeap(struct Node* root)
{
    // These two are used in isCompleteUtil()
    unsigned int node_count = countNodes(root);
    unsigned int index = 0;
```

```
    if (isCompleteUtil(root, index, node_count) && isHeapUtil(root))
        return true;
    return false;
}

// Driver program
int main()
{
    struct Node* root = NULL;
    root = newNode(10);
    root->left = newNode(9);
    root->right = newNode(8);
    root->left->left = newNode(7);
    root->left->right = newNode(6);
    root->right->left = newNode(5);
    root->right->right = newNode(4);
    root->left->left->left = newNode(3);
    root->left->left->right = newNode(2);
    root->left->right->left = newNode(1);

    if (isHeap(root))
        printf("Given binary tree is a Heap\n");
    else
        printf("Given binary tree is not a Heap\n");

    return 0;
}
```

Output:

```
Given binary tree is a Heap
```