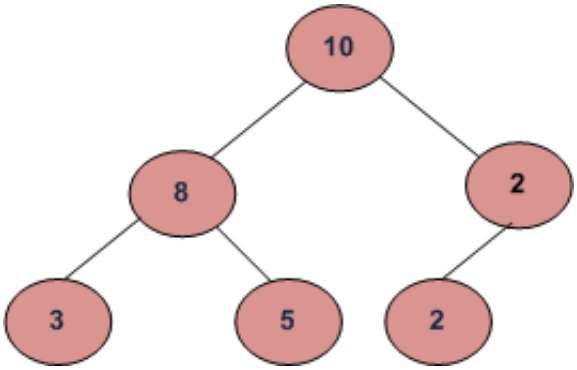
Check for Children Sum Property in a Binary Tree.

Given a binary tree, write a function that returns true if the tree satisfies below property.

For every node, data value must be equal to sum of data values in left and right children. Consider data value as 0 for NULL children. Below tree is an example



Algorithm:

Traverse the given binary tree. For each node check (recursively) if the node and both its children satisfy the Children Sum Property, if so then return true else return false.

Implementation:

```
/* Program to check children sum property */
#include <stdio.h>
#include <stdlib.h>

/* A binary tree node has data, left child and right child */
struct node
{
    int data;
    struct node* left;
    struct node* right;
};

/* returns 1 if children sum property holds for the given
    node and both of its children*/
int isSumProperty(struct node* node)
{
  /* left_data is left child data and right_data is for right child data*/
  int left_data = 0,  right_data = 0;

  /* If node is NULL or it's a leaf node then
     return true */
  if(node == NULL ||
     (node->left == NULL && node->right == NULL))
```

```
    return 1;
  else
  {
    /* If left child is not present then 0 is used
       as data of left child */
    if(node->left != NULL)
      left_data = node->left->data;

    /* If right child is not present then 0 is used
      as data of right child */
    if(node->right != NULL)
      right_data = node->right->data;

    /* if the node and both of its children satisfy the
       property return 1 else 0*/
    if((node->data == left_data + right_data)&&
        isSumProperty(node->left) &&
        isSumProperty(node->right))
      return 1;
    else
      return 0;
  }
}

/*
 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);
}

/* Driver program to test above function */
int main()
{
  struct node *root  = newNode(10);
  root->left         = newNode(8);
  root->right        = newNode(2);
  root->left->left   = newNode(3);
  root->left->right  = newNode(5);
  root->right->right = newNode(2);
  if(isSumProperty(root))
    printf("The given tree satisfies the children sum property ");
  else
    printf("The given tree does not satisfy the children sum property ");

  getchar();
  return 0;
}
```

Run on IDE

Time Complexity: O(n), we are doing a complete traversal of the tree.

As an exercise, extend the above question for an n-ary tree.

This question was asked by Shekhar.

Please write comments if you find any bug in the above algorithm or a better way to solve the same problem.

75 Comments Category: Trees

Related Posts:

- Find Count of Single Valued Subtrees
- Check if a given array can represent Preorder Traversal of Binary Search Tree
- Mirror of n-ary Tree
- Succinct Encoding of Binary Tree
- Construct Binary Tree from given Parent Array representation
- Symmetric Tree (Mirror Image of itself)
- Find Minimum Depth of a Binary Tree
- Maximum Path Sum in a Binary Tree

2 Average Difficulty : **2/5.0**
Based on **1** vote(s)

(Login to Rate)

Writing code in comment? Please use code.geeksforgeeks.org, generate link and share the link here.

vivekvivek • 15 hours ago

JAVA Code

```
public class BTCheck4ChildrenSumProp

{

public static Node newNode(int data)

{
```

```
Node node=new Node();

node.data=data;

node.left=null;

node.right=null;

return (node);

}
```

see more

• Reply • Share ›

Vikash Kumar • 2 months ago

we can also do it by level order traversal.

1 • Reply • Share ›

Shikha Gupta • 3 months ago

Havent checked for all test cases but following is my code:

```
#include<stdio.h>

#include<stdlib.h>

#include<math.h>

struct node

{

int data;

struct node *llink;

struct node *rlink;

};

int checkChildrenSumProperty(struct node *root)
```

see more

• Reply • Share ›

Abhi Shikha Gupta • 16 days ago

This will not work

```
struct node *root=newNode(10);
```

root->llink=newNode(8);

root->rlink=newNode(11);

root->llink->llink=newNode(5);

root->llink->rlink=newNode(1);

because (5+1) != 8 return -1;
and (-1+11)==10 that is true so output is satisfied
but output shouldn't be satisfied as 5+1 is not equal to 8.

u can keep a global variable flag=1 make it zero if it hits return -1
or pass this value by reference.

• Reply • Share ›

Akhilesh Kumar ➚ Shikha Gupta • 2 months ago

Pls check Test Case
root=newNode(12);

root->llink=newNode(8);

root->rlink=newNode(4);

root->llink->llink=newNode(6);

root->llink->rlink=newNode(2);

root->rlink->llink=newNode(1);

root->rlink->rlink=newNode(3);

root->llink->llink->rlink=newNode(5);

root->rlink->rlink->llink=newNode(7);

root->rlink->rlink->rlink=newNode(9);

• Reply • Share ›

Ankur Goyal ➚ Shikha Gupta • 3 months ago

yes, it will work for all the cases :)

1 • Reply • Share ›

Akhilesh Kumar ➚ Ankur Goyal • 2 months ago

Please check the Test Case

root=newNode(12);

root->llink=newNode(8);

root->rlink=newNode(4);

root->llink->llink=newNode(6);

root->llink->rlink=newNode(2);

root->rlink->llink=newNode(1);

root->rlink->rlink=newNode(3);

root->llink->llink->rlink=newNode(5);

root->rlink->rlink->llink=newNode(7);

root->rlink->rlink->rlink=newNode(9);

• Reply • Share ›

Shikha Gupta ➚ Ankur Goyal • 3 months ago

Thanks Ankur :)

• Reply • Share ›

Amitava Mozumder • 3 months ago

the short and most efficient version of above prog

```
int isChildrenSum(struct node * t)
{
if(!t)
return 1;
if(!t->left&&!t->right)
return 1;
return (isChildrenSum(t->left)&&(isChildrenSum(t->right))&&
(t->data == ((t->left)?t->left->data:0 )+ ((t->right)?t->right->data:0 ) ));
}
```

• Reply • Share ›

Hardeep Sidhu ➚ Amitava Mozumder • 3 months ago

how to call this function??

• Reply • Share ›

Amitava Mozumder ➚ Hardeep Sidhu • 3 months ago

@hardeep just call it with the root node ,
(isChildrenSum (root))? cout<< "yes" : cout<<"No";

• Reply • Share ›

Abhishek Somani ➚ Amitava Mozumder • 3 months ago

I have compilers:

Dev C++ 5.7

Codeblocks

Turbo C/C++

Borland and many more.

And it runs fine babe.

See this link :

http://ideone.com/UmkPq5

And please dont upvote your own answer !!Bubyeee

• Reply • Share ›

Amitava Mozumder ➚ Abhishek Somani • 3 months ago

of course it will run fine !! you added return 0 when i said , and you changes your && to || when i pointed it out , c'mon dude , just accept you were wrong don't argue with me and correct your code at the same time #facepalm

• Reply • Share ›

Saket More • 4 months ago

https://ideone.com/8A2sJb

• Reply • Share ›

Saket More ➚ Saket More • 4 months ago

Just like the postorder traversal..

• Reply • Share ›

Abhishek Somani • 4 months ago

Why not use the most easy Postorder Traversal for this :-

```
int isChildrenSum(struct node * T)

{

int left,right;

if(T)

{
```

left=isChildrenSum(T->left);

right=isChildrenSum(T->right);

if(T->data==(left+right)||(T->left!=NULL&&T->right!=NULL))

return 1;

}
return 0;

}

NOTE:
Considers both NULL and leaf nodes as trivially satisfying the property :)
Thanks :D
1 • Reply • Share ›

Amitava Mozumder Abhishek Somani • 3 months ago
the code is wrong, compile and see for yourself -_-
• Reply • Share ›

Abhishek Somani Amitava Mozumder • 3 months ago
Can you give me such case bro!
• Reply • Share ›

Amitava Mozumder Abhishek Somani • 3 months ago
just run your code against any tree, the answer is always the data of the root node , maybe because you didn't put "return 0" anywhere,
the left and right variable can only hold value 1 , so checking t->data == (1+1)&&(1 or 0) makes no sense to me

check out my version , it's the most easiest way without using any variables
• Reply • Share ›

Abhishek Somani Amitava Mozumder • 3 months ago
Check the code again!
• Reply • Share ›

Amitava Mozumder Abhishek Somani • 3 months ago
yup ran it again now the ans is 0 every time , don't you have a compiler ?
can't you see for a leaf node both the left and right variable holds 0

can't you see for a leaf node both the left and right variable holds 0 ; so the && with anything zero is always zero
plz forget your code and do this ⇓⇓

```
int isChildrenSum(struct node * t)
{
if(!t)
return 1;
if(!t->left&&!t->right)
return 1;
return (isChildrenSum(t->left)&&(isChildrenSum(t->right))&&
(t->data == ((t->left)?t->left->data:0 )+ ((t->right)?t->right->data:0 )
));
}
```

• Reply • Share ›

SlickHackz ➚ Amitava Mozumder • 19 days ago

pretty slick..

you can combine the 1st and 2nd ifs, such that :

```
int isChildSum(struct node *root)
{
if(!root || (!root->left && !root->right))
return 1;

return (isChildrenSum(t->left)&&(isChildrenSum(t->right))&&
(t->data == ((t->left)?t->left->data:0 )+ ((t->right)?t->right->data:0 )
));
}
```

• Reply • Share ›

Shantanu • 4 months ago

```
int isSumProperty(struct node* node)

{

int ldata=0,rdata=0;

if(node==NULL||node->left==NULL && node->right==NULL)

return 1;

else

{
```

```
if(node->left)

ldata=node->left->data;

if(node->right)

rdata=node->right->data;

return((node->data==ldata+rdata)

&& isSumProperty(node->left)

&& isSumProperty(node->right));

}

}
```

3 • Reply • Share ›

Ashish Singh • 5 months ago

http://ideone.com/IwrOeC

• Reply • Share ›

prashant patel • 5 months ago

one simple implementation can be:

```
int isSumProperty(Node* root){

if(root==null) return 0;

int left=isSumProperty(root->left);

int right=isSumProperty(root->right);

if((left==0 && right==0) || root->data == left + right) return root->data;
else return 0;
}
```

• Reply • Share ›

Hitesh Lalwani prashant patel • 3 months ago

the best one dude!! ;)

• Reply • Share ›

sick_master • 5 months ago

```
bool isSumProperty ( struct node * node )
{
if ( node == NULL || ( node->left == NULL && node->right == NULL ) )
return true ;
```

return true ;
return (isSumProperty (node->left) && isSumProperty (node->right) && node->data == (node->left->data) + (node->right->data)) ;
}

• Reply • Share ›

Mitul Agrawal • 6 months ago

Use a boolean variable as one of the method arguments and maintain state rather than checking for left and right child in every recursive call.

• Reply • Share ›

aakash kaushik • 6 months ago

static boolean isOrderedTree(Node root) {
if (root == null || (root.leftChild == null && root.rightChild == null))
return true;
if (getNodeData(root) == (getNodeData(root.leftChild) + getNodeData(root.rightChild))){
return true && isOrderedTree(root.leftChild) && isOrderedTree(root.rightChild);
}

return false;
}

static int getNodeData(Node node) {

if (node == null) {
return 0;
}
return node.data;
}

1 • Reply • Share ›

Jerry Goyal • 8 months ago

```
int sum(node* node){
        if(!node)
        return 1;
        if(node->left==NULL&&node->right==NULL){
                return 1;
         }
         return ( (root->data==(root->left?root->left->data:0)+(root->right?root->righ
         sum(node->left)&&sum(node->right) );


 }
```

• Reply • Share ›

neo ➚ Jerry Goyal • 8 months ago

if node is null and

if root->right or root->left is null , then there will be segmentation fault in your code.

• Reply • Share ›

Jerry Goyal ➚ neo • 8 months ago

thnks for pointing out..now it's correct.

• Reply • Share ›

surbhijain93 • 8 months ago

```
bool i(struct node* node)

{

if(node==NULL)

return ;

if(node->left==NULL && node->right==NULL)

return true;

if(node->data==((node->left)?node->left->data:0)+((node->right)?node->right->data:0))

return ( true && i(node->left) && i(node->right));

return false;

}
```

• Reply • Share ›

Yeshwanth Selvaraj • 8 months ago

```
int isSumProperty(struct node * root)
{
static int i=1;
if(root== NULL)
return 1;
if (root->data != root->left->data + root->right->data)
{
return !i;
}
return isSumProperty(root->left) && isSumProperty(root->right) && i;
}
```

• Reply • Share ›

Utkarsh Mishra • 9 months ago

it return value of root if all done else 0

```
int sumtree(node* root)
{
int right,left,sum;
if (root==NULL)
return 0;
else
{
left=sumtree(root->llink);
right=sumtree(root->rlink);
sum=left+right;
if(sum==0)
return root->data;
else if(sum==root->data)
return root->data;
else return 0;
}
}
```

• Reply • Share ›

surbhijain93 • 9 months ago

2) This is giving run time error.Please tell why

```
int check(struct node* node)

{ int sum,flag;

if(node==NULL || (node->left==NULL && node->right==NULL)){

return 1;

}

printf("%d %d",node->left->data,node->right->data);

if(node->left==NULL)

{

sum=node->right->data;

}
```

see more

• Reply • Share ›

Pankaj Kushwaha surbhijain93 • 5 months ago

just add blow line in your main function , it will work:

root->right->left = newNode(<any_number>);

you was derefrencing a NULL pointer , that's why it was giving a segmentation fault....

• Reply • Share ›

surbhijain93 • 9 months ago

Is this correct?

```
int sum(struct node* new)
{
int data;
static int flag=0;
if(new->left==NULL && new->right==NULL)
return;
size(new->left);
size(new->right);
if(new->left==NULL)
{
data=new->right->data;
}
else if( new->right==NULL)
{
data=new->left->data;
}
else
```

see more

• Reply • Share ›

Agam • a year ago

A Shorter Solution :

```
int csum(struct node *root)
{
if(root == NULL)
return 0;
if(root->left == NULL && root->right == NULL)
return root->data;
if(root->data == (csum(root->left)+csum(root->right)))
return root->data;
else
```

```
return 0;
}
```

• Reply • Share ›

don • a year ago

```
/* Program to check children sum property */
#include <stdio.h>
#include <stdlib.h>

/* A binary tree node has data, left child and right child */
struct node
{
int data;
struct node* left;
struct node* right;
};

/* returns 1 if children sum property holds for the given
node and both of its children*/
int isSumProperty(struct node* node)
{
int l=0,r=0;
if(node==NULL)
```

see more

1 • Reply • Share ›

sahil • a year ago

```
bool checksumproperty(struct tree* root)

{

if(root==NULL || root->left==NULL && root->right==NULL)

{

return 1;

}

if((root->data==root->left->data+root->right->data))

{

return checksumproperty(root->left);

return checksumproperty(root->right);
```

return checkSumProperty(root->right);

return 1;

}

return 0;

}

2 • Reply • Share ›

Avata This comment was deleted.

halfcoder Guest • 5 months ago

stop wondering..and start thinking

• Reply • Share ›

Prakhar • a year ago

```
int isSumProperty(node *root)
{

if(root == NULL)
{
return 1;
}

int left = isSumProperty(root->left);
int right = isSumProperty(root->right);

int t = 0;

int data_left = (root->left == NULL)?0:root->left->data;
int data_right = (root->right == NULL)?0:root->right->data;

if(root->data == data_left + data_right || data_left + data_right == 0 )
t = 1;

return (left && right && t ) ;

}
```

• Reply • Share ›

ryan • a year ago

code for checksum

```
bool checksumprop(struct Node* node)
{
```

```
{
if(node==NULL)
return false;
int num=0;
num=check(node);
if(num!=0)
return true;
else
return false;
}
int check(struct Node*node)
{
if(node->left==NULL && node->right==NULL)
return (node->data);
int l=0,r=0;
if(node->left)
l=check(node->left);
if(node->right)
r=check(node->right);
if(l+r==node->data)
return(node->data);
else
return 0;
}
```

• Reply • Share ›

JustThinking • a year ago

How to solve if property is something like this

Sum of data values in entire left sub tree + Sum Of data values in entire right Sub tree should be equal to node data.

• Reply • Share ›

ajayv JustThinking • a year ago

that is SumTree..

see this.. http://www.geeksforgeeks.org/c...

1 • Reply • Share ›

Sachin • a year ago

I have implemented above problem for n-ary tree.

```
import java.util.ArrayList;

public class NaryTree {
```

```
public static void printNaryTree(Node root){

if(root == null){

return;

}

System.out.println(root.data);

if(root.child != null){

for(Node n : root.child){

printNaryTree(n);
```

see more

• Reply • Share ›

<HoldOnLife!#> • a year ago

If a tree has just a root node then the left data is zero and right data is zero but root->data!=0 but it is satisfying children sum property according to solution?

1 • Reply • Share ›

Guest <HoldOnLife!#> • a year ago

@<HoldOnLife!#> All leaf nodes satisfy the children sum property. Its required as termination condition for recursion. Root node can be treated as a leaf node, having no left or right children.

```
/* If node is NULL or it's a leaf node then return true */
if(node == NULL || (node->left == NULL && node->right == NULL))
return 1;
```

Its return above in the code.

1 • Reply • Share ›

Load more comments

Subscribe Add Disqus to your site Privacy