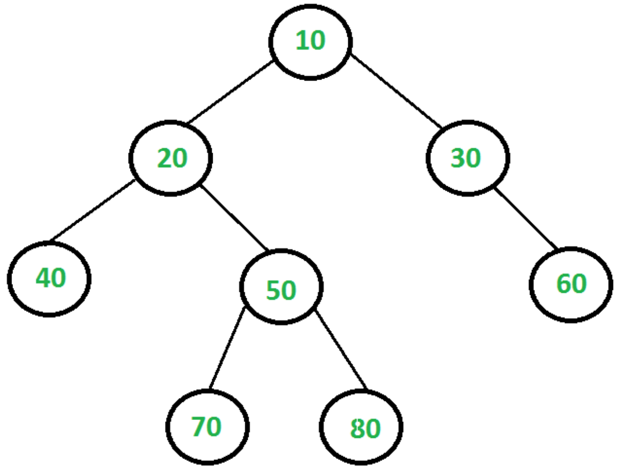
Dynamic Programming | Set 26 (Largest Independent Set Problem)

Given a Binary Tree, find size of the **L**argest **I**ndependent **S**et(LIS) in it. A subset of all tree nodes is an independent set if there is no edge between any two nodes of the subset.
For example, consider the following binary tree. The largest independent set(LIS) is {10, 40, 60, 70, 80} and size of the LIS is 5.



A Dynamic Programming solution solves a given problem using solutions of subproblems in bottom up manner. Can the given problem be solved using solutions to subproblems? If yes, then what are the subproblems? Can we find largest independent set size (LISS) for a node X if we know LISS for all descendants of X? If a node is considered as part of LIS, then its children cannot be part of LIS, but its grandchildren can be. Following is optimal substructure property.

1) Optimal Substructure:

Google™ Custom Search

GeeksforGeeks
53,523 people like GeeksforGeeks.

Interview Experiences
Advanced Data Structures
Dynamic Programming
Greedy Algorithms
Backtracking
Pattern Searching
Divide & Conquer
Mathematical Algorithms
Recursion
Geometric Algorithms

Let LISS(X) indicates size of largest independent set of a tree with root X.

```
     LISS(X) = MAX { (1 + sum of LISS for all grandchildren of X),
                     (sum of LISS for all children of X) }
```

The idea is simple, there are two possibilities for every node X, either X is a member of the set or not a member. If X is a member, then the value of LISS(X) is 1 plus LISS of all grandchildren. If X is not a member, then the value is sum of LISS of all children.

2) Overlapping Subproblems
Following is recursive implementation that simply follows the recursive structure mentioned above.

```
// A naive recursive implementation of Largest Independent Set problem
#include <stdio.h>
#include <stdlib.h>

// A utility function to find max of two integers
int max(int x, int y) { return (x > y)? x: y; }

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

// The function returns size of the largest independent set in a given
// binary tree
int LISS(struct node *root)
{
    if (root == NULL)
       return 0;

    // Caculate size excluding the current node
    int size_excl = LISS(root->left) + LISS(root->right);

    // Calculate size including the current node
    int size_incl = 1;
    if (root->left)
       size_incl += LISS(root->left->left) + LISS(root->left->right);
    if (root->right)
       size_incl += LISS(root->right->left) + LISS(root->right->right)
```

Popular Posts

- All permutations of a given string
- Memory Layout of C Programs
- Understanding “extern” keyword in C
- Median of two sorted arrays
- Tree traversal without recursion and without stack!
- Structure Member Alignment, Padding and Data Packing
- Intersection point of two Linked Lists
- Lowest Common Ancestor in a BST.
- Check if a binary tree is BST or not
- Sorted Linked List to Balanced BST

```
    // Return the maximum of two sizes
    return max(size_incl, size_excl);
}


// A utility function to create a node
struct node* newNode( int data )
{
    struct node* temp = (struct node *) malloc( sizeof(struct node) );
    temp->data = data;
    temp->left = temp->right = NULL;
    return temp;
}

// Driver program to test above functions
int main()
{
    // Let us construct the tree given in the above diagram
    struct node *root         = newNode(20);
    root->left                = newNode(8);
    root->left->left          = newNode(4);
    root->left->right         = newNode(12);
    root->left->right->left   = newNode(10);
    root->left->right->right  = newNode(14);
    root->right               = newNode(22);
    root->right->right        = newNode(25);

    printf ("Size of the Largest Independent Set is %d ", LISS(root));

    return 0;
}
```

Output:

```
Size of the Largest Independent Set is 5
```

Time complexity of the above naive recursive approach is exponential. It should be noted that the above function computes the same subproblems again and again. For example, LISS of node with value 50 is evaluated for node with values 10 and 20 as 50 is grandchild of 10 and child of 20. Since same suproblems are called again, this problem has Overlapping Subprolems property. So LISS problem has both properties (see this and this) of a dynamic programming problem. Like other typical Dynamic Programming(DP) problems, recomputations of same subproblems can be avoided by storing the solutions to subproblems and solving problems in bottom up manner.

Following is C implementation of Dynamic Programming based solution. In the following solution, an additional field 'liss' is added to tree nodes. The initial value of 'liss' is set as 0 for all nodes. The recursive function LISS() calculates 'liss' for a node only if it is not already set.

```
/* Dynamic programming based program for Largest Independent Set probl
#include <stdio.h>
#include <stdlib.h>

// A utility function to find max of two integers
int max(int x, int y) { return (x > y)? x: y; }

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

// A memoization function returns size of the largest independent set
//  a given binary tree
int LISS(struct node *root)
{
    if (root == NULL)
        return 0;

    if (root->liss)
        return root->liss;

    if (root->left == NULL && root->right == NULL)
        return (root->liss = 1);

    // Caculate size excluding the current node
    int liss_excl = LISS(root->left) + LISS(root->right);

    // Calculate size including the current node
    int liss_incl = 1;
    if (root->left)
        liss_incl += LISS(root->left->left) + LISS(root->left->right);
    if (root->right)
        liss_incl += LISS(root->right->left) + LISS(root->right->right

    // Return the maximum of two sizes
```

705 Subscribe

Recent Comments

Aman Hi, Why arent we checking for conditions...

Write a C program to Delete a Tree. · 35 minutes ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 38 minutes ago

Sanjay Agarwal bool tree::Root_to_leaf_path_given_sum(tree...

Root to leaf path sum equal to a given number · 1 hour ago

GOPI GOPINATH @admin Highlight this sentence "We can easily...

Count trailing zeroes in factorial of a number · 1 hour ago

newCoder3006 If the array contains negative numbers also. We...

Find subarray with given sum · 1 hour ago

newCoder3006 Code without using while loop. We can do it...

Find subarray with given sum · 1 hour ago

```
    root->liss = max(liss_incl, liss_excl);

    return root->liss;
}

// A utility function to create a node
struct node* newNode(int data)
{
    struct node* temp = (struct node *) malloc( sizeof(struct node) );
    temp->data = data;
    temp->left = temp->right = NULL;
    temp->liss = 0;
    return temp;
}

// Driver program to test above functions
int main()
{
    // Let us construct the tree given in the above diagram
    struct node *root         = newNode(20);
    root->left                = newNode(8);
    root->left->left          = newNode(4);
    root->left->right         = newNode(12);
    root->left->right->left   = newNode(10);
    root->left->right->right  = newNode(14);
    root->right               = newNode(22);
    root->right->right        = newNode(25);

    printf ("Size of the Largest Independent Set is %d ", LISS(root));

    return 0;
}
```

Output

```
Size of the Largest Independent Set is 5
```

Time Complexity: O(n) where n is the number of nodes in given Binary tree.

Following extensions to above solution can be tried as an exercise.
1) Extend the above solution for n-ary tree.

2) The above solution modifies the given tree structure by adding an additional field 'liss' to tree nodes. Extend the solution so that it doesn't modify the tree structure.

3) The above solution only returns size of LIS, it doesn't print elements of LIS. Extend the solution to print all nodes that are part of LIS.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Related Tpoics:

- Print a Binary Tree in Vertical Order | Set 2 (Hashmap based Method)
- Print Right View of a Binary Tree
- Red-Black Tree | Set 3 (Delete)
- Construct a tree from Inorder and Level order traversals
- Print all nodes at distance k from a given node
- Print a Binary Tree in Vertical Order | Set 1
- Interval Tree
- Check if a given Binary Tree is height balanced like a Red-Black Tree

Writing code in comment? Please use **ideone.com** and share the link here.

29 Comments GeeksforGeeks

Sort by Newest

Join the discussion…

hj • 3 months ago

Can this problem be seen as a variant of Graph Coloring Problem? The larges
color will form largest independent set.
Of course the graph coloring depends on the order in which vertices are seen
independent subsets and use for graph coloring?

1 •

aman hj • a month ago

Yes it can be done taking in consideration the notion of bipartite graph.

•

Cristian Florica • 4 months ago

What is the approach for the "Maximum Weighted Independent Set (MWIS) pr

•

prashant jha • 5 months ago

```
#include<iostream>
using namespace std;
struct node
{
node* lchild;
int data;
node* rchild;
node(int d)
{
```

```
data=d;
lchild=NULL;
rchild=NULL;
}
}*root=NULL;
void create(node* &root,int d)
{
int n;
if(d==-1)
```

see more

Nitesh • 5 months ago

LIS will always contain leaves? is there any case in which it will not??

# if it always contains leafs then a better solution is

```
# include<stdio.h>
# include<iostream>
# include<stdlib.h>
int arr[100],t;
struct node
{ int data,flag;
struct node *left, *right;
};
int lic(node *root)
{
if(root==NULL)
return 0;
else
{
```

see more

Guru Gorantla Nitesh • 3 months ago

It need not contain leaves all the time. For example if a tree has a deptl
leaves. Let this dth row contain only one leaf then your LISS need not c
when we consider an example we tend to take a almost balanced bina
number of nodes in that depth increases with d(if almost balanced) . S
leaf.

Behind D Walls • 5 months ago

recursion code goes into infinite loop for keys 1-50 or more elements why?

Guest • 6 months ago

is the solution nothing but the root and all the leaves possible for a tree...i knov
there r two levels but else for other cases ,my conclusion is right,isnt it?? pls h

1

hj Guest • 3 months ago

I was thinking the same too.. but on second thought, this in fact forms
from the case where there are two levels.
Consider the case where there are 5 levels.. (assume complete tree) r
3rd level can also be included in the largest independent set.

Sreenivas Doosa • 7 months ago

Awesome logic duuude :) Appreciate it..!

Gaurav Gulzar • 9 months ago

/*Try this and please reply if u find anything wrong*/.

```
//Utility Function
int _LISS(Node *root, int *count) {.
if(! root)
return 0;

if(!(_LISS(root->left, count) + _LISS(root->right, count))) {.
(*count)++;
return 1;
}

return 0;
}

int LISS(Node *root) {.
int count = 0;.
_LISS(root,&count);
return count;
}
```

gaurav • 9 months ago

```
[sourcecode language="C++"]
/* Paste your code here (You may delete these lines if not writing code) */

//
// LongestIndSet.cpp
//
//
// Created by Gaurav Gulzar on 08/08/13.
//
```

```
//

#include
using namespace std;

typedef struct tNode
{
int data;
struct tNode* left;
struct tNode* right;
```

see more

RAHUL23 • 9 months ago

CAN ANYONE TELL HOW TO PRINT MAXIMUM LIST SET??
IT WOULD BE OF GR8 HELP

THANKS IN ADVANCE

1

sudhanshu • 10 months ago

Another method O(n) :-

1) Perform level order traversal and in an array(declared globally), increment t

2) So you get a count of the number of nodes corresponding to each level in a

3) Now, get the max subset sum ensuring adjacent levels(cells) aren't picked

This way we can also print them easily

Takes O(n) time overall.

saurabh ➜ sudhanshu • 6 months ago

in the given example 60 and 70 are from adjacent levels but yet they fo
you will end up with answer 4 for the same case.

Born Actor • 10 months ago

```
#include <iostream>
#include<string>
#include<sstream>
#include<iomanip>
#include <stdio.h>
#include <math.h>
#include <vector>
#include <stdlib.h>
using namespace std;
int count=0; //count variable counts the number of times the recursiv
class node
{
        public:
        node*l;
        node *r;
        node *p;
        int value;
```

see more

illuminati • 11 months ago

one of the few problems solved with memoisation on GFG.

Karan Verma • 11 months ago

If the root&#039s children are the leaves themselves,then it&#039ll be wrong.

abhishek08aug • a year ago

Intelligent :D

Abhijeet • a year ago

Isn't the size of largest independent set equal MAX(#of nodes at even depths,
based solution will do with the O(n) complexity.

1

Niks • a year ago

```
// A naive recursive implementation of Largest Independent Set probler
#include <stdio.h>
#include <stdlib.h>

// A utility function to find max of two integers
int max(int x, int y) { return (x > y)? x: y; }

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

static int index = 0;
```

see more

apsc • a year ago

Can someone please tell how to extend it to print nodes of LIS, also without ch
we use memoization. Thanks.

Santhosh • a year ago

Here's my take on the problem. Simple traversal based solution. Let me know
missed.

This works from bottom up. I assume that all leaf nodes are part of the solutio

Algo: all nodes contain a bool print.

1) Postorder traversal. If the node is root, set print=true
2) If node->left is null check if right node print value.
3) set current->print = !node->right->print.
4) If node->right is null set current->print=!node->left->print
5) If there are both left and right nodes, check if they are printed.
6) If any one of them is printed then skip current node, else print it.

```
void postorder(struct node* node)
{
     if (node == NULL)
        return;
```

see more

Srinath ➦ Santhosh • a year ago

I am sure pretty sure this approach is correct,a lot more intuitive and e

^ | ˅ •

Santhosh ➦ Santhosh • a year ago

If this is found to be logically correct, can be easily extended to get cou
structure of node then a simple solution would be to push the bool valu
number of children of the node and compare them.

^ | ˅ •

sreeram • a year ago

Without modifying original tree

```
[#include
#include

int max(int x, int y) { return (x > y)? x: y; }

struct node
{
int data;
struct node *left, *right;
};
int LISSUtil(struct node *root,int *lcref,int *rcref)
{
if (root == NULL){
*lcref=*rcref=0;
return 0;}
```

see more

op • a year ago

What if the max arguments happens to be same. Then in that case we will ha
add 90 to inorder last node i.e. 60. then [40, 70, 80, 10, 60] and [40, 70, 80, 30,
Your programs works fine.
But at the time of printing we need to take care of this case, if it is required to c

viki • a year ago

There is no need to have these lines in second solution. I mean, removing thes
Cheers

if (root->left == NULL && root->right == NULL)
return (root->liss = 1);

bond • a year ago

its a recent tc div2 hard problem :D