Ministry of Education and Science of Ukraine

National Technical University of Ukraine

«Kyiv Polytechnic Institute. Igor Sikorsky »

Faculty of Informatics and Computer Technologies

Department of Computer Engineering

LAB № 6

from the discipline "Theory of Algorithms"

on the topic «Binary search trees»

PERFORMED BY:

1st year student

group ІП-93

Dominskyi V.O.

The credit - 9311

Variant – 11

CHECKED:

Associate Professor of OT

c.t.s.,s.r.

Antoniuk А.І.

Kyiv - 2020

**TASK**

**Goal:**

transform an input binary tree into a binary search tree. Search for sums of consecutive nodes in a tree.

**Option task:**

1. Convert an input binary tree into a binary search tree.

A binary tree with a fixed structure (that is, links between nodes, their parent, and descendants) is fed into the input. You must rewrite the values ​​of the tree nodes so that:

a) their new values ​​were taken only from the set present in the input tree;

b) the internal structure of the tree (the links between the parent node and the descendant node) was maintained.

2. Search for sums of consecutive nodes in the tree

After the input tree has been converted to a binary search tree, the next task must be resolved. In addition, a certain number of S. is given. In the obtained binary search tree, it is necessary to find all such monotone paths (which do not necessarily come from the root, but all go from top to bottom) that the sum of the values ​​of the nodes belonging to the found paths is equal to S.

**CODE**

**using** System**;**

**using** System**.**Collections**.**Generic**;**

class BinaryTree

**{**

class Vertex

**{**

//index in vertex

**public** int Information**;**

//creating two parts of the vertex

**public** Vertex LeftPart**,** RightPart**;**

**}**

**public** static void Main**(**String**[]** Arguments**)**

**{**

//making binary tree

Vertex Parent **=** **null;**

Parent **=** insert**(**Parent**,** 11**);**

Parent **=** insert**(**Parent**,** 0**);**

Parent **=** insert**(**Parent**,** 25**);**

Parent **=** insert**(**Parent**,** 1**);**

Parent **=** insert**(**Parent**,** 20**);**

Parent **=** insert**(**Parent**,** 14**);**

Parent = insert(Parent, 5);

Parent = insert(Parent, 31);

//number,which we need to find

int FinalNumber = 19;

FindCombination(Parent, FinalNumber);

}

static Vertex NewVertex(int Information)

{

Vertex NotPermanent = new Vertex();

NotPermanent.Information = Information;

//dont make new children

NotPermanent.LeftPart = null;

NotPermanent.RightPart = null;

return NotPermanent;

}

static Vertex insert(Vertex Parent, int Key)

{

if (Parent == null)

{

//theres no numbers

return NewVertex(Key);

}

if (Key < Parent.Information)

{

//left wing of the tree is always<than the parent

Parent.LeftPart = insert(Parent.LeftPart, Key);

}

else

{

//right wing of the tree is always>than the parent

Parent.RightPart = insert(Parent.RightPart, Key);

}

return Parent;

}

static bool FindCombinationSecond(Vertex Parent, int FinalNumber, HashSet<int> Combination)

{

if (Parent == null)

{

//starting vertex

return false;

}

//we are inserting vertex value into combination

//checking difference between giving FinalNumber and vertex value in combination

//if it is found,then combination exists,otherwise-no

if (FindCombinationSecond(Parent.LeftPart, FinalNumber, Combination))

{

//these are vertexes,which go further

return true;

}

if (Combination.Contains(FinalNumber - Parent.Information))

{

//for example "19-11=8" and showing final combination

Console.WriteLine("One of the combinations is [" + (FinalNumber - Parent.Information) + ", " + Parent.Information + "]");

return true;

}

else

{

Combination.Add(Parent.Information);

}

return FindCombinationSecond(Parent.RightPart, FinalNumber, Combination);

}

static void FindCombination(Vertex Parent, int FinalNumber)

{

HashSet<int> Combination = new HashSet<int>();

if (!FindCombinationSecond(Parent, FinalNumber, Combination))

{

//if everything went really wrong

Console.WriteLine("There's no combinations :(");

}

}

}

**RESULTS OF THE PROGRAM WORK**

The input:

Array = [11, 0, 25, 1, 20, 14, 5, 31]

S = 19

Output array: One of the combinations is [5, 14]

**CONCLUSIONS**

I got acquainted with the topic of laboratory work.

Have acquired relevant work skills.

An appropriate test program has been developed.

Got the job of binary search trees. Considered several ways to set trees. I understood the benefits of implementing this algorithm