**Міністерство Освіти І НАУКИ України**

**Національний університет "Львівська політехніка"**

**IКНІТ**

Кафедра **ПЗ**

### ЗВІТ

До лабораторної роботи № 9

**З дисципліни:** *“Алгоритми та структури даних”*

**На тему:** *“Нелінійні структури даних: червоно-чорні дерева”*

**Лектор:**

доц. каф. ПЗ

Коротєєва Т.О.

**Виконала:**

ст. групи ПЗ-26

Матолінець Л. А.

**Прийняв:**

асистент каф. ПЗ

Симець І.І.

« \_\_\_\_ » \_\_\_\_\_\_\_\_ 2022 р.

∑= \_\_\_\_\_ .

Львів – 2022

**Тема роботи:** Нелінійні структури даних: червоно-чорні дерева**.**

**Мета роботи**:  Ознайомитися з червоно-чорними деревами та отримати навички програмування алгоритмів, що їх обробляють.

**ІНДИВІДУАЛЬНЕ ЗАВДАННЯ**

**Варіант 8**

**Варіант 8:** N, M – дійсні; другий за значенням по спаданню елемент та його колір; батько та його колір.

**ХІД ВИКОНАННЯ**

using System;

using System.Runtime.CompilerServices;

namespace SortingApp

{

enum Color

{

Red,

Black

}

class RB

{

public class Node

{

public Color colour;

public Node left;

public Node right;

public Node parent;

public double data;

public Node() { }

public Node(double data) { this.data = data; }

public Node(Color colour) { this.colour = colour; }

public Node(double data, Color colour) { this.data = data; this.colour = colour; }

}

private Node root;

public String displayedData = "";

public String descendants = "";

public Node secondMinimal;

public RB() {

}

private void LeftRotate(Node X)

{

Node Y = X.right; // set Y

X.right = Y.left;//turn Y's left subtree into X's right subtree

if (Y.left != null)

{

Y.left.parent = X;

}

if (Y != null)

{

Y.parent = X.parent;//link X's parent to Y

}

if (X.parent == null)

{

root = Y;

}

if (X == X.parent.left)

{

X.parent.left = Y;

}

else

{

X.parent.right = Y;

}

Y.left = X; //put X on Y's left

if (X != null)

{

X.parent = Y;

}

}

private void RightRotate(Node Y)

{

Node X = Y.left;

Y.left = X.right;

if (X.right != null)

{

X.right.parent = Y;

}

if (X != null)

{

X.parent = Y.parent;

}

if (Y.parent == null)

{

root = X;

}

if (Y == Y.parent.right)

{

Y.parent.right = X;

}

if (Y == Y.parent.left)

{

Y.parent.left = X;

}

X.right = Y;//put Y on X's right

if (Y != null)

{

Y.parent = X;

}

}

public String getDescendants(Node node)

{

String str = "";

if (node == null)

{

str = "Nothing in the tree!";

}

if (node != null)

{

getDescendantsD(node);

}

str = descendants;

return str;

}

private void getDescendantsD(Node current)

{

if (current != null)

{

getDescendantsD(current.left);

descendants += current.data + ", ";

getDescendantsD(current.right);

}

}

public String DisplayTree()

{

String str = "";

if (root == null)

{

str ="Nothing in the tree!";

}

if (root != null)

{

InOrderDisplay(root);

}

str = displayedData;

return str;

}

public Node FindInTree(double key)

{

String str = "";

bool isFound = false;

Node temp = root;

Node item = null;

while (!isFound)

{

if (temp == null)

{

break;

}

if (key < temp.data)

{

temp = temp.left;

}

if (key > temp.data)

{

temp = temp.right;

}

if (key == temp.data)

{

isFound = true;

item = temp;

}

}

return temp;

}

public void Insert(double item)

{

Node newItem = new Node(item);

if (root == null)

{

root = newItem;

root.colour = Color.Black;

return;

}

Node Y = null;

Node X = root;

while (X != null)

{

Y = X;

if (newItem.data < X.data)

{

X = X.left;

}

else

{

X = X.right;

}

}

newItem.parent = Y;

if (Y == null)

{

root = newItem;

}

else if (newItem.data < Y.data)

{

Y.left = newItem;

}

else

{

Y.right = newItem;

}

newItem.left = null;

newItem.right = null;

newItem.colour = Color.Red;

InsertFixUp(newItem);

}

private void InOrderDisplay(Node current)

{

if (current != null)

{

InOrderDisplay(current.left);

displayedData += current.data + ", ";

InOrderDisplay(current.right);

}

}

private void InsertFixUp(Node item)

{

while (item != root && item.parent.colour == Color.Red)

{

if (item.parent == item.parent.parent.left)

{

Node Y = item.parent.parent.right;

if (Y != null && Y.colour == Color.Red)

{

item.parent.colour = Color.Black;

Y.colour = Color.Black;

item.parent.parent.colour = Color.Red;

item = item.parent.parent;

}

else

{

if (item == item.parent.right)

{

item = item.parent;

LeftRotate(item);

}

item.parent.colour = Color.Black;

item.parent.parent.colour = Color.Red;

RightRotate(item.parent.parent);

}

}

else

{

Node X = null;

X = item.parent.parent.left;

if (X != null && X.colour == Color.Black

item.parent.colour = Color.Red;

X.colour = Color.Red;

item.parent.parent.colour = Color.Black;

item = item.parent.parent;

}

else

{

if (item == item.parent.left)

{

item = item.parent;

RightRotate(item);

}

item.parent.colour = Color.Black;

item.parent.parent.colour = Color.Red;

LeftRotate(item.parent.parent);

}

}

root.colour = Color.Black;

}

}

public String getSecondMinimal()

{

String str = "";

double value = 0;

Node current = root;

while (current.left != null)

{

current = current.left;

}

Node secondSmallest = current.parent;

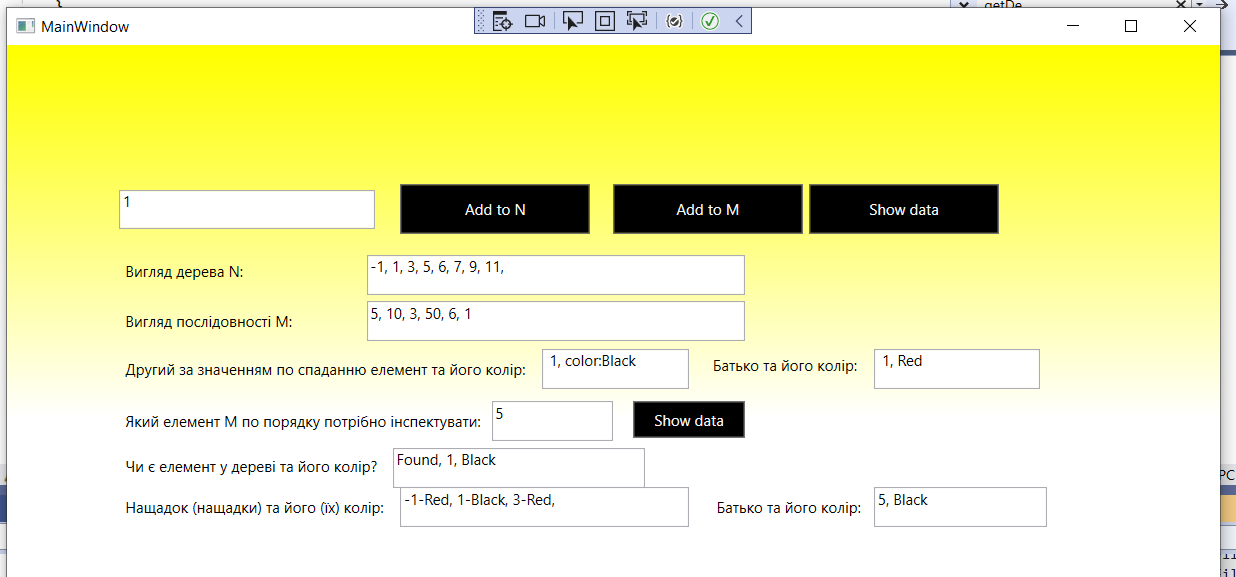
str += " " + (secondSmallest.data) + ", color:" + secondSmallest.colour;

return str;

}

}

}



*Рис. 1 Результат виконання програми*

**Висновок**

На цій лабораторній роботі я познайомилася з нелінійними структурами даних, зокрема червоно-чорними деревами, реалізувала основні функції з ними, також виконала індивідуальне завдання, внаслідок чого отримала навички роботи з алгоритмами, які обробляють це червоно-чорне дерево.