**МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КПІ»**

**Кафедра**

**Автоматизованих систем обробки інформації та управління**

**Звіт**

**з комп’ютерного практикуму №6**

**з дисципліни «Технології створення програмних продуктів»**

**Виконав: Михайленко О.М.**

**Студент гр. ІС-41, ФІОТ, 3 курс,**

**залікова книжка IC-4118**

Київ 2017

Розглядається фрагмент коду, що відповідає за розфарбування графу методом стягування

Вхідні дані:матриця суміжності, boolean

Вихідні дані: масив з кольорами вершин, цілі невід’ємні числа

Примітка: замість РГР з курсу ТСПП для тестування була взята курсова робота з курсу ДРДО, тому що РГР, що являє собою графічний редактор, тому тісно пов’язаний із графічним інтерфейсом користувача та не містить жодного класу чи методу, що не пов’язаний з Windows Forms, які не підлягають unit-тестуванню.

**Фрагмент коду**

static public int[] Color(bool[,] matrix)

{

bool[,] copy\_adj = new bool[matrix.GetLength(0), matrix.GetLength(1)];

for (int i = 0; i < matrix.GetLength(0); i++)

{

for (int j = 0; j < matrix.GetLength(1); j++) //copy of adjacency matrix

{

copy\_adj[i, j] = matrix[i, j];

}

}

int[] color = new int[copy\_adj.GetLength(0)];

int currentColor = 1;

for(int i = 0; i < copy\_adj.GetLength(0); i++)

{

if(color[i] == 0) //if vertex is not colored

{

color[i] = currentColor;

for (int j = i + 1; j < copy\_adj.GetLength(1); j++) // for pair of vertexes

{

if (copy\_adj[i, j] == false) //if vertexes are not connected

{

color[j] = color[i];

for (int k = 0; k < copy\_adj.GetLength(1); k++)

{

copy\_adj[i, k] = copy\_adj[i, k] || copy\_adj[j, k]; //contracting vertexes

copy\_adj[j, k] = true; //remouving vertex j

if (copy\_adj[k, j]) //reconecting edges of j to i

copy\_adj[k, i] = true;

copy\_adj[k, j] = true; //remouving vertex j

}

}

}

currentColor++;

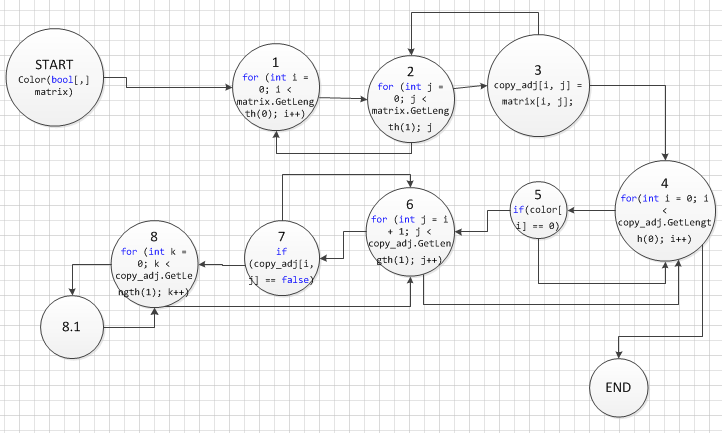
}

}

return color;

}

**Потоковий граф**



Визначення цикломатичної складності потокового графу:

1. V(G) = 8 Регіонів;
2. V(G) = 17 дуг – 11 вузлів + 2 = 8;
3. V(G) = 7 пред. вуз. + 1 = 8.

**Множина незалежних лінійних шляхів**

Визначення усіх можливих шляхів

Шлях 1: start – 1– 2 – 3 – 4 – end – граф не містить жодного елемента

Шлях 2: start – 1– 2 – 3 – 4 – 5– 6 – 7 – 8–8.1 – 8 – 6 – 4 – end – граф не містить жодної грані

Шлях 3: start – 1– 2 – 3 – 4 – 5 – 6 – 4– end – граф повний

Шлях 4: start – 1– 2 – 3 – 4 – 5– 6 – 7 – 8–8.1 – 8 – 6 – 4 – end – повний шлях

**Тестові варіанти**

У таблиці приведені тестові варіанти, складені відповідно до класів еквівалентності:

Таблиця 1 – Варіанти тестів для різних шляхів у потоковому графі

| **Група тестів** | **Тест** | **Вхідні дані** | **Очікуваний результат** |
| --- | --- | --- | --- |
| Normal work | public void ColorTest() | bool[,] m = { { false, true }, { true, false } }; | flag == true; |
| Normal work | public void FullConnectionsTest() | bool[,] m = { { true, true, true }, { true, true, true }, { true, true, true } }; | flag == true; |
| Empty graph | public void EmptyTest() | bool[,] m = {}; | flag == true; |
| No edges | public void EmptyConnectionsTest() | bool[,] m = { { false, false, false }, { false, false, false }, { false, false, false } }; | flag == true; |

Реалізуємо ці тести в окремому файлі вихідного коду, використовуючи фреймворк Microsoft UnitTesting:

using Microsoft.VisualStudio.TestTools.UnitTesting;

using Map\_coloring;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Map\_coloring.Tests

{

[TestClass()]

public class ContractionColoringTests

{

[TestMethod()]

public void ColorTest()

{

bool[,] m = { { false, true }, { true, false } };

int[] color = ContractionColoring.Color(m);

int[] expected = { 1, 2 };

bool flag = true;

for (int i = 0; i < expected.GetLength(0); i++)

if (expected[i] != color[i])

flag = false;

Assert.AreEqual(flag, true);

}

[TestMethod()]

public void EmptyTest()

{

bool[,] m = {};

int[] color = ContractionColoring.Color(m);

int[] expected = {};

bool flag = true;

for (int i = 0; i < expected.GetLength(0); i++)

if (expected[i] != color[i])

flag = false;

Assert.AreEqual(flag, true);

}

[TestMethod()]

public void EmptyConnectionsTest()

{

bool[,] m = { { false, false, false }, { false, false, false }, { false, false, false } };

int[] color = ContractionColoring.Color(m);

int[] expected = {1, 1, 1 };

bool flag = true;

for (int i = 0; i < expected.GetLength(0); i++)

if (expected[i] != color[i])

flag = false;

Assert.AreEqual(flag, true);

}

[TestMethod()]

public void FullConnectionsTest()

{

bool[,] m = { { true, true, true }, { true, true, true }, { true, true, true } };

int[] color = ContractionColoring.Color(m);

int[] expected = { 1, 2, 3 };

bool flag = true;

for (int i = 0; i < expected.GetLength(0); i++)

if (expected[i] != color[i])

flag = false;

Assert.AreEqual(flag, true);

}

}

}

Результат тестування:

