Федеральное агентство по образованию

ФГБОУ ВО Уфимский государственный авиационный технический

университет

Кафедра ВМиК

Расчетно-графическая работа

По дисциплине «Алгоритмы и структуры данных»

Вариант 17

Выполнил: студент группы ПРО-222 Хусамов А. Р.

Проверил: Верхотурова Г. Н.

Уфа 2020

**Задание:** Орграф, заданный с помощью списков смежности, проверить на наличие

циклов и при их наличии вывести каждый цикл в виде вершин циклического пути.

**Определение:**

***Список******смежности*** — один из способов представления графа в виде коллекции списков вершин. Каждой вершине графа соответствует список, состоящий из «соседей» этой вершины

***Цикл*** – замкнутая цепь в графе.

***Цепь*** – маршрут, в котором все ребра различны.

***Маршрут*** – чередующая последовательность вершин и ребер графа.

**Алгоритм нахождения циклов в графе:**

При обходе используем два цвета вершин:

* “white” – не посещенная,
* “grey” – посещенная,

1. Запускаем DFS(int v) из каждой вершины

(где v – текущая вершина)

1. Подсчитываем кол-во смежных вершин с данной и записываем результат в переменную count.
2. Закрашиваем вершину в серый цвет.
3. Начинаем обход всех соседних вершин:
   1. Пока последний символ в строке с циклом не будет равен v, удаляем символы с конца. Прибавляем в конце строки « -> ».
   2. Если кол-во путей из этой вершины > 1 и это не первый путь, меняем цвета серых вершин, не находящихся в цикле, на белые, чтобы можно было найти и другие циклы. Т. к. в серые вершины функция не заходит.
   3. Если следующая вершина белая, прибавляем в цикл вершину и рекурсивно вызываем DFS для следующей вершины.
   4. Если цвет вершины серый и вершина равна стартовой, печатаем цикл в Result\_RichTextBox и добавляем к ней стартовую вершину.

Цикл существует, если поиск в глубину обнаруживает ребро, конец которого покрашен в серый цвет и совпадает с начальной вершиной.

**Входные данные:**

* Построенный пользователем невзвешенный орграф
* adjacency\_list [n, n] – список смежности из n элементов

(n – максимальное кол-во вершин орграфа)

**Вспомогательные данные:**

* struct Circle – структура, хранящая координаты и радиус круга
* color [n] – массив цветов вершин орграфа
* cycle – строка с циклом (без конечного значения)
* start\_ind – индекс начальной вершины
* count – кол-во смежных вершин
* changed – булевая переменная для проверки на окрашенность в серый
* from\_vertex – переменная, хранящая значение первой выбранной вершины
* to\_vertex – переменная, хранящая значение второй выбранной вершины

**Выходные данные:**

* Result\_RichTextBox – поле с выводом циклов орграфа
* В случае отсутствия циклов в орграфе будет выведено соответствующее сообщение

**Объявление глобальных переменных:**

#region Declaration of variables

struct Circle //структура Круг (координаты центра круга и радиус)

{

public int x;

public int y;

public int rad;

}

static int n = 15; //Максимальное кол-во вершин в орграфе

Circle[] circles = new Circle[n]; //массив кругов

int k = 0; //Кол-во вершин в орграфе

int[,] adjacency\_list = new int[n, n]; //Список смежности из n элементов

string[] color = new string[n]; //Массив цветов вершин орграфа

//Изначально все вершины белые

//При входе в вершину её цвет будет изменяться на серый

string cycle = ""; //Строка с циклом (без конечного значения)

int start\_ind; //Индекс начальной вершины

int from\_vertex = -1; //Переменная для запоминания выбранной вершины

#endregion

**Код модифицированного Поиска в глубину:**

void DFS(int v) //Модифицированная функция Поиска в глубину

{ //v — вершина, в которой мы сейчас находимся

int count = 0; //Переменная для подсчёта путей исходящих из этой вершины

for (int i = 0; i < k; ++i) //Считаем пути

if (adjacency\_list[v, i] == 0) //Если встречаем 0 в списке смежности, значит больше путей нет

{

count = i; //Запоминаем кол-во путей

break; //Выходим из цикла

}

color[v] = "grey"; //Закрашиваем вершину в серый цвет

for (int u = 0; u < count; ++u) //Проходим по всем "соседям" этой вершины

{

//Возвращаем строку до вершины, у которой несколько путей

//Это необходимо для верного вывода циклов

while (cycle[cycle.Length-1] != (v+49))

cycle = cycle.Remove(cycle.Length - 1);

cycle += " -> ";

if (count > 1 && u != 0) //Если кол-во путей из этой вершины > 1 и это не первый путь

change\_color(k); //Меняем цвета серых путей на белые, чтобы можно было найти и другие циклы

if (color[adjacency\_list[v, u] - 1] == "white") //Если следующая вершина белая

{

cycle += (adjacency\_list[v, u]) + " "; //Прибавляем в цикл вершину

DFS(adjacency\_list[v, u] - 1); //Вызываем DFS для следующей вершины

}

//Если цвет вершины серый и вершина равна стартовой, печатаем цикл в Result\_RichTextBox

if (color[adjacency\_list[v, u] - 1] == "grey" && adjacency\_list[v, u]-1 == start\_ind)

Result\_RichTextBox.Text += cycle + (start\_ind + 1) + "\n";

}

}

**Код функции изменения цветов вершин:**

void change\_color() //Изменяем цвета в орграфе

{

for (int j = 0; j < k; ++j) //Проходим по всем вершинам

{

bool changed = false; //Булевая переменная для проверки на окрашенность в серый

for (int i = 0; i < cycle.Length; ++i) //Проходим по всей строке с циклом

if (j+49 == cycle[i]) //Если встречаем вершину в цикле

{

color[j] = "grey"; //Красим её в серый цвет

changed = true; //Отмечаем это в булевой переменной

break; //Выходи из цикла

}

if (!changed) //Если вершина не была перекрашена

color[j] = "white"; //Красим её в белый цвет

}

}

**Обработчик нажатия на Canvas\_Panel:**

private void Canvas\_Panel\_MouseDown(object sender, MouseEventArgs e)//Обработчик нажатия на Полотно

{

Pen pen\_default = new Pen(Color.MediumBlue, 3); //карандаш для круга по умолчанию

Pen pen\_selected = new Pen(Color.Red, 3); //карандаш для выбранного круга

Pen pen\_for\_line = new Pen(Color.Black, 2); //карандаш для соединительных линий

pen\_for\_line.CustomEndCap = new AdjustableArrowCap(7, 13);//Добавляем в конец линии стрелочку

if (e.Button == MouseButtons.Right) //если нажата правая кнопка мыши

{

circles[k].rad = 15;

circles[k].x = e.X - circles[k].rad;

circles[k].y = e.Y - circles[k].rad;

Canvas\_Panel.CreateGraphics().DrawEllipse(pen\_default,

circles[k].x,

circles[k].y,

circles[k].rad \* 2,

circles[k].rad \* 2); //рисуем круг

var lb = new Label //текст для отображения номера круга на панели

{

Location = new Point(circles[k].x + circles[k].rad / 2 + 30, circles[k].y + circles[k].rad / 2 - 5), //местоположение текста

Text = (k + 1).ToString(), //номер круга

Font = new Font("Serif", 14),

AutoSize = true //размер "автоматический"

};

Canvas\_Panel.Controls.Add(lb); //добавление номера круга на панель

k++; //увеличиваем счётчик вершин

if (k > n)//Если кол-во вершин больше максимального

{

MessageBox.Show("Вы привысили максимальное кол-во вершин!");

Clear\_Button\_Click(sender, e);

}

}

if (e.Button == MouseButtons.Left) //если нажата левая кнопка мыши

{

if (from\_vertex == -1) //если пока не запомнили ни одну из вершин

{

for (int i = 0; i < k; i++) //проверяем попал ли щелчок мыши на какую-то из вершин

if (((e.X - circles[i].x - circles[i].rad) \*

(e.X - circles[i].x - circles[i].rad) +

(e.Y - circles[i].y - circles[i].rad) \*

(e.Y - circles[i].y - circles[i].rad))

< (circles[i].rad) \* (circles[i].rad))

{

from\_vertex = i;//если попали, то запоминаем вершину по которой щелкнули

Canvas\_Panel.CreateGraphics().DrawEllipse(pen\_selected,

circles[i].x,

circles[i].y,

circles[i].rad \* 2,

circles[i].rad \* 2); //выделяем круг

break;//выходим из цикла

}

}

else //если по одной из вершин уже раннее щелкнули(запомнили), то

{

int to\_vertex = -1; //создаём переменную для запоминания второй вершины

for (int i = 0; i < k; i++) //проверяем попал ли щелчок мыши на какую-то из вершин

if (((e.X - circles[i].x - circles[i].rad) \*

(e.X - circles[i].x - circles[i].rad) +

(e.Y - circles[i].y - circles[i].rad) \*

(e.Y - circles[i].y - circles[i].rad))

< (circles[i].rad) \* (circles[i].rad))

{

to\_vertex = i;//если попали, то запоминаем вершину по которой щелкнули

Canvas\_Panel.CreateGraphics().DrawEllipse(pen\_selected,

circles[i].x,

circles[i].y,

circles[i].rad \* 2,

circles[i].rad \* 2); //выделяем круг

break;//выходим из цикла

}

if ((to\_vertex != -1) && (from\_vertex != to\_vertex))//если щелкнули сначала по одной, а потом по другой вершине

{

//центр первой выделенной вершины

Point p1 = new Point(

circles[from\_vertex].x + circles[from\_vertex].rad,

circles[from\_vertex].y + circles[from\_vertex].rad);

//центр второй выделенной вершины

Point p2 = new Point(

circles[to\_vertex].x + circles[to\_vertex].rad,

circles[to\_vertex].y + circles[to\_vertex].rad);

//рисуем соединительную линию

Canvas\_Panel.CreateGraphics().DrawLine(pen\_for\_line, p1, p2);

//отмечаем в списке смежностей путь

int i;

for (i = 0; adjacency\_list[from\_vertex, i] != 0; ++i) ;

adjacency\_list[from\_vertex, i] = to\_vertex + 1; //Добавляем вершину в список смежности

AdjList\_RichTextBox.Text = "";//Очищаем RichTextBox со списком смежности, чтобы переписать его

for (i = 0; i < k; ++i)//Обновляем списки смежности в RichTextBox

{

AdjList\_RichTextBox.Text += (i + 1) + ":";

for (int j = 0; adjacency\_list[i, j] != 0; ++j)

AdjList\_RichTextBox.Text += " " + adjacency\_list[i, j];

AdjList\_RichTextBox.Text += "\n";

}

Canvas\_Panel.CreateGraphics().DrawEllipse(pen\_default,

circles[from\_vertex].x,

circles[from\_vertex].y,

circles[from\_vertex].rad \* 2,

circles[from\_vertex].rad \* 2);//Снимаем выделение

Canvas\_Panel.CreateGraphics().DrawEllipse(pen\_default,

circles[to\_vertex].x,

circles[to\_vertex].y,

circles[to\_vertex].rad \* 2,

circles[to\_vertex].rad \* 2);//Снимаем выделение

from\_vertex = -1;//сбрасываем значение для дальнейшей работы

}

}

}

}

**Обработчик нажатия на кнопку «Очистить список смежностей»:**

private void Clear\_Button\_Click(object sender, EventArgs e) //Возврат всех элементов панели к начальному положению

{

for (int i = 0; i < k; ++i) //Очистка списков смежностей

for (int j = 0; j < k; ++j)

adjacency\_list[i, j] = 0;

AdjList\_RichTextBox.Text = ""; //Очистка RichTextBox со списками смежностей

Result\_RichTextBox.Text = ""; //Очистка RichTextBox с результатом

Canvas\_Panel.Refresh(); //Обновление полотна

for (int i = 0; i < k; ++i) //Очистка массива структур

{

circles[i].x = 0;

circles[i].y = 0;

circles[i].rad = 0;

}

Canvas\_Panel.Controls.Clear(); //Очитска Полотна от лэйблов

k = 0; //Обнуляем кол-во вершин

}

**Обработчик нажатия на кнопку «Проверить на наличие циклов»:**

private void Result\_Button\_Click(object sender, EventArgs e)//Обработчик нажатия на кнопку результата

{

for (int i = 0; i < k; ++i) //Для всех вершин в орграфе

{

start\_ind = i; //Запоминаем начальную вершину

for (int j = 0; j < k; ++j) //Все вершины "перекрашиваем" в белый цвет

color[j] = "white";

cycle = (i+1) + " "; //Очищаем строку с циклом

DFS(i); //Вызываем DFS для i-той вершины

}

if (Result\_RichTextBox.Text == "")//Если RichTextBox с результатом пуст, после предыдущего шага

Result\_RichTextBox.Text = "Циклы не найдены!"; //То орграф не имеет циклов

}

**Анализ порядка сложности алгоритма по управляющим структурам:**

**Функция change\_color:**

void change\_color()

{

for (int j = 0; j < k; ++j)

{

…

for (int i = 0; i < cycle.Length; ++i)

if (j+49 == cycle[i])

{

O(K\*C)

O(C)

…

break;

}

}

}

Сложность функции = O(K\*C),

где C – длина строки, K – кол-во вершин в орграфе

**Функции DFS:**

void DFS(…)

{

for (int i = 0; i < k; ++i)

if (adjacency\_list[v, i] == 0)

{

...

O(K)

break;

}

...

for (int u = 0; u < count; ++u)

{

while (cycle[cycle.Length-1] != (v+49))

O(C)

...

if (count > 1 && u != 0)

O(K\*C)

change\_color();

O(U(K\*C))

if (color[adjacency\_list[v, u] - 1] == "white")

{

...

O(K+M)

DFS(adjacency\_list[v, u] - 1);

}

...

}

}

Сложность функции = O(U(K\*C)),

где U – кол-во смежных вершин, K – кол-во вершин орграфа, C – длина строки

**Обработчик нажатия на кнопку результата:**

private void Result\_Button\_Click(…)

{

for (int i = 0; i < k; ++i)

{

...

O(KU(K\*C))

for (int j = 0; j < k; ++j)

O(K)

...

O(U(K\*C))

DFS(i);

}

...

}

Сложность события = O(KU(K\*C)),

где K – кол-во вершин в орграфе, M - общее количество рёбер, C – длина строки

**Обработчик нажатия на Полотно:**

private void Canvas\_Panel\_MouseDown(object sender, MouseEventArgs e)

{

...

if ()

{

O(K)

for (int i = 0; i < k; i++)

...

}

else

{

...

O(K)

for (int i = 0; i < k; i++)

...

if ()

{

...

O(U)

for (i = 0; adjacency\_list[from\_vertex, i] != 0; ++i) ;

...

for (i = 0; i < k; ++i)

{

O(K)

...

}

...

}

}

}

Сложность события = O(K)

где K – кол-во вершин, U – кол-во смежных вершин

**Пример работы программы:**

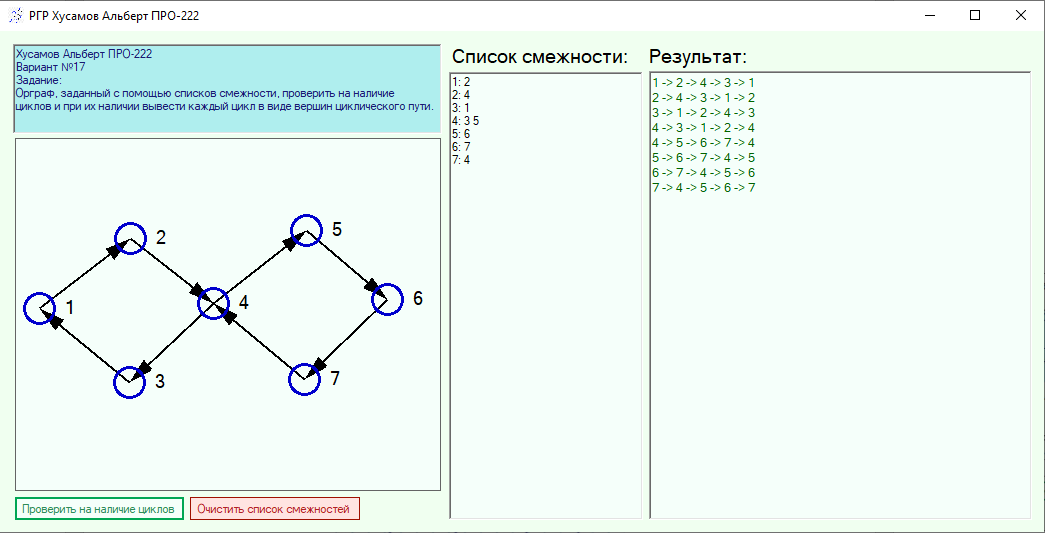


Рис. 1 Демонстрация работы программы

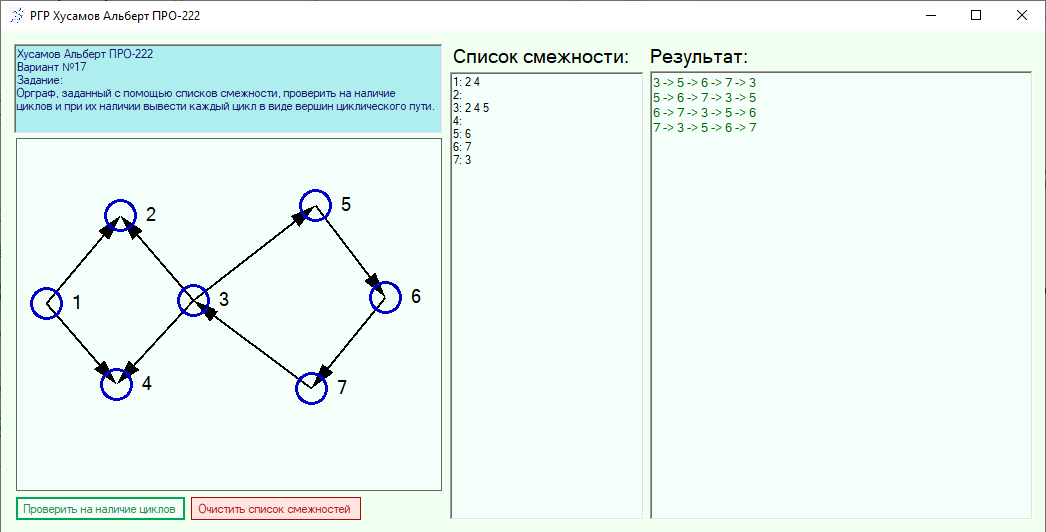


Рис. 2 Демонстрация работы программы

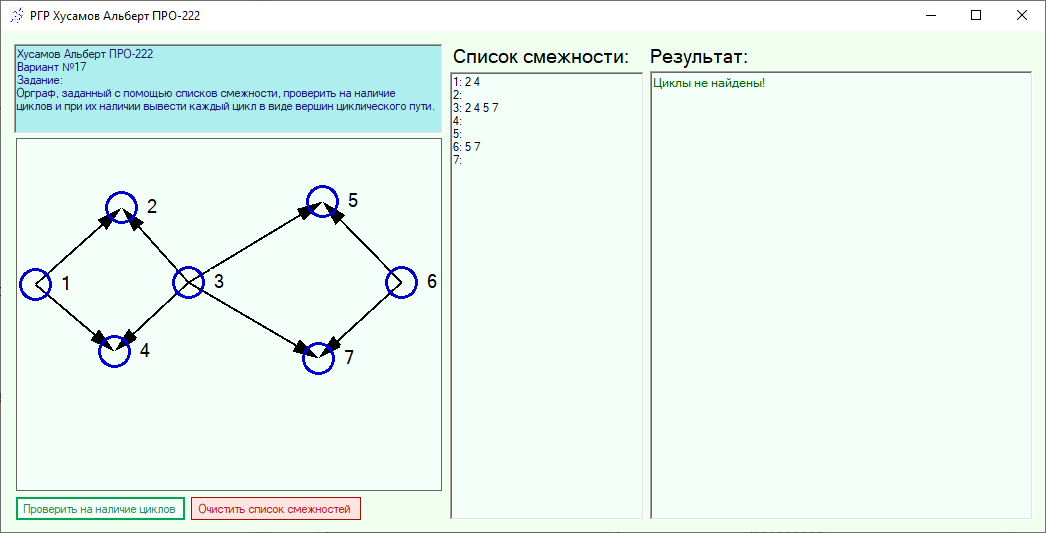


Рис. 3 Демонстрация работы программы

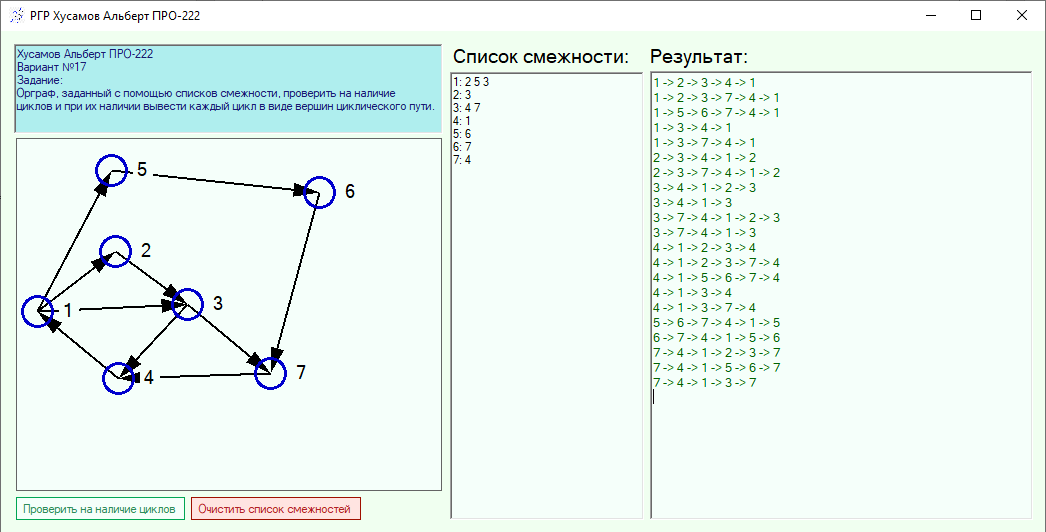


Рис. 4 Демонстрация работы программы

СПИСОК ЛИТЕРАТУРЫ

1. Алексеев, В.Е. Графы и алгоритмы. Структуры данных. Модели вычислений. Москва: БИНОМ, Лаборатория Знаний, 2011,–320 с. ISBN 978-5-94774-543-6.

2.Ф.А. Новиков. Дискретная математика для программистов. СПб: Питер, 2000.

3. Б.С. Хусаинов. Структуры и алгоритмы обработки данных. Примеры на языке Си: Учебное пособие.– М.: Финансы и статистика, 2004.– 464 с.–ISBN 5-279-02775-8.

4. Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн Алгоритмы: построение и анализ = Introduction to Algorithms. — 2-е изд. — М.: «Вильямс», 2006. — С. 1296.

5. Майника Э. Алгоритмы оптимизации на сетях и графах. 1981.

6. Евстигнеев В.А. Применение теории графов в программировании. 1985.

7. Райли Д. Абстракция и структуры данных. Вводный курс. 1993.

8. Вирт, Н. Алгоритмы и структуры данных. Новая версия для Оберона [электронный ресурс]. Москва: ДМК ПРЕСС, 2010.–272 с.– ISBN 978-5-94074-584-6.

9. Липский В. Комбинаторика для программистов. 1988.

10. Свами М., Тхуласираман К. Графы, сети и алгоритмы. 1984.

11.Ф.Препарата, М.Шеймос. Вычислительная геометрия. М.: «Мир», 1989.

12. Ахо А.В., Хопкрофт Д., Ульман Д. «Структуры данных и алгоритмы». М.: Вильямс, 2000

13. В.М. Бондарев, В.И. Рублинецкий, Е.Г. Качно. "Основы программирования". Харьков, "Фолио", Ростов-на-Дону, "Феникс". 1997.

14. Уильям Топп, Уильям Форд. Структуры данных в C++. 1999.

15. С.Д. Кондратьева. "Введение в структуры". Москва, МГТУ им. Баумана, 2000.