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, int k) из каждой вершины

(где v – текущая вершина, k – кол-во вершин в орграфе)

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:**

* adjacency\_list [n, n] – список смежности из n элементов

(n – максимальное кол-во вершин орграфа)

**Вспомогательные данные:**

* 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

**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

static int n = 15; //Максимальное 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

#endregion

**Код модифицированного Поиска в глубину:**

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

{ //v — вершина, в которой мы сейчас находимся, k - кол-во вершин в орграфе

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

for (int i = 0; i < n; ++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, k); //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(int k) //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 нажатий на клавишу в TextBox:**

private void Adjacency\_TextBox\_KeyDown(object sender, KeyEventArgs e) //Обработчик нажатия на TextBox с вводом данных

{

if (e.KeyCode == Keys.Enter) //Если было нажатие на Enter

{

int ind = Convert.ToInt32(Top\_NumericUpDown.Value) - 1; //Индекс

string copy\_adj = "";

if (adjacency\_list[ind, 0] == 0) //Если для этой вершины еще нет путей

{

int i = 0;

string numb = ""; //Храним значение очередного числа в строковом представлении

copy\_adj = Adjacency\_TextBox.Text;

while (Adjacency\_TextBox.Text != "" && Adjacency\_TextBox.Text != " ") //Пока TextBox не пуст и не имеет одного символа = пробелу

{

if (Adjacency\_TextBox.Text[0] == ' ' && numb != "") //Если первый элемент TextBox - пробел и numb не пуст

{

if (numb.Length > 1) //Если число не однозначное

{

for (int j = 0; j < numb.Length; ++j) //Преобразуем строкоове значение в числовое

adjacency\_list[ind, i] += Convert.ToInt32(numb[j]) \* (int)Math.Pow(10, numb.Length - 1 - j);

}

else //Если число однозначное

adjacency\_list[ind, i] = Convert.ToInt32(numb);

++i;

numb = ""; //Обнуляем numb

Adjacency\_TextBox.Text = Adjacency\_TextBox.Text.Remove(0, 1); //Удаляем первый символ из TextBox

}

else //Если первый элемент TextBox - не пробел или numb пуст

{

numb += Adjacency\_TextBox.Text[0]; //Добавялем в numb цифру

Adjacency\_TextBox.Text = Adjacency\_TextBox.Text.Remove(0, 1); //Удаляем первый символ из TextBox

}

}

if(Adjacency\_TextBox.Text == "" && numb != "") //Если TextBox пуст и numb не пуст

if (numb.Length > 1) // Если число не однозначное

{

for (int j = 0; j < numb.Length; ++j) //Преобразуем строкоове значение в числовое

adjacency\_list[ind, i] += Convert.ToInt32(numb[j]) \* (int)Math.Pow(10, numb.Length - 1 - j);

}

else

adjacency\_list[ind, i] = Convert.ToInt32(numb); //Если число однозначное

}

else //Если список смежности для этой вершины не пуст

//Выводим об этом сообщение

Adjacency\_TextBox.Text = "Вы уже вводили данные для этой вершины! Смените вершину.";

//Выводим список смежности для данной вершины

AdjList\_RichTextBox.Text += "Вершина №" + Top\_NumericUpDown.Value + ": " + copy\_adj + "\n";

//Если вершина не равна кол-ву всех вершин в орграфе

if (Top\_NumericUpDown.Value != CountTop\_NumericUpDown.Value)

Top\_NumericUpDown.Value++; //Переходим к следуеющей вершине

}

}

**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 < n; ++i) //Очистка списков смежностей

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

adjacency\_list[i, j] = 0;

Adjacency\_TextBox.Text = ""; //Очистка TextBox, куда вводим данные

Top\_NumericUpDown.Value = 1; //Возврат значения NumericUpDown к 1

CountTop\_NumericUpDown.Value = 1; //Возврат значения NumericUpDown к 1

Alert\_Label.Text = ""; //Очистка Label с предупреждением

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

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

}

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

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

{

int k = Convert.ToInt32(CountTop\_NumericUpDown.Value); //k - кол-во вершин в орграфе

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, k); //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

**Функции DFS:**

void DFS(…)

{

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

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

{

...

O(N)

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(k);

O(U(K\*C))

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

{

...

O(K+M)

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

}

...

}

}

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(K(K+M))

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

O(K)

...

O(K+M)

DFS(i, k);

}

...

}

Сложность события = O(K(K+M)),

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

**Обработчик нажатия на клавиши в TextBox:**

private void Adjacency\_TextBox\_KeyDown(…)

{

...

while(Adjacency\_TextBox.Text != "" && Adjacency\_TextBox.Text != " ")

{

...

for (int j = 0; j < numb.Length; ++j)

O(N)

...

O(N2)

for (int j = 0; j < numb.Length; ++j)

O(N)

...

...

}

}

Сложность события = O(N2)

**Пример работы программы:**

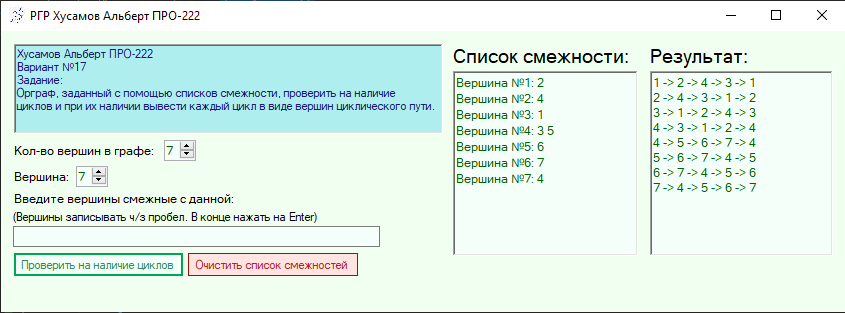


Рис. 1 Демонстрация работы программы

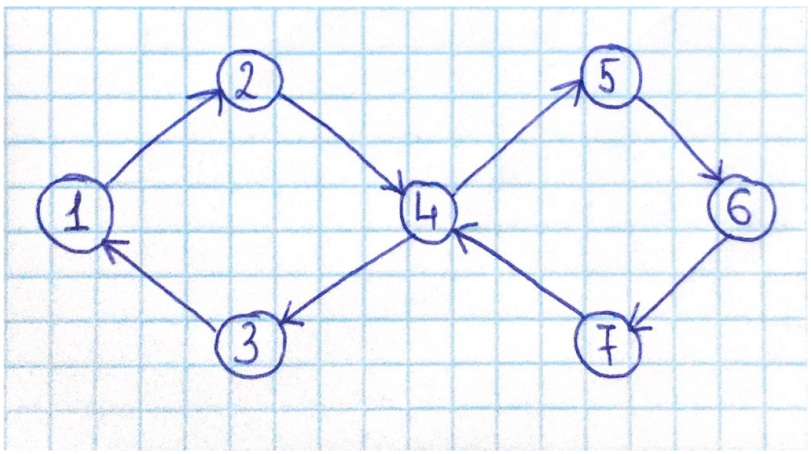


Рис. 2 Орграф

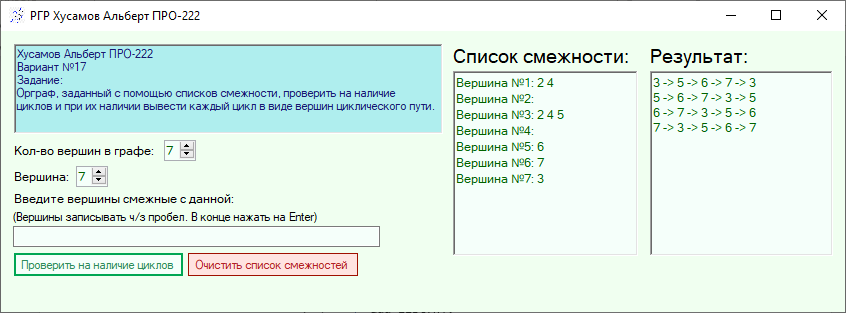


Рис. 3 Демонстрация работы программы

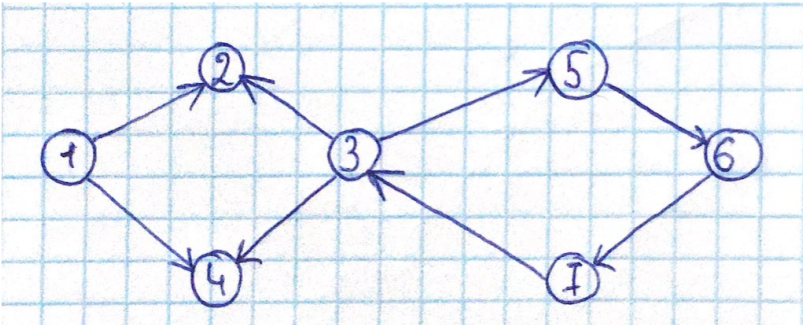


Рис. 4 Орграф

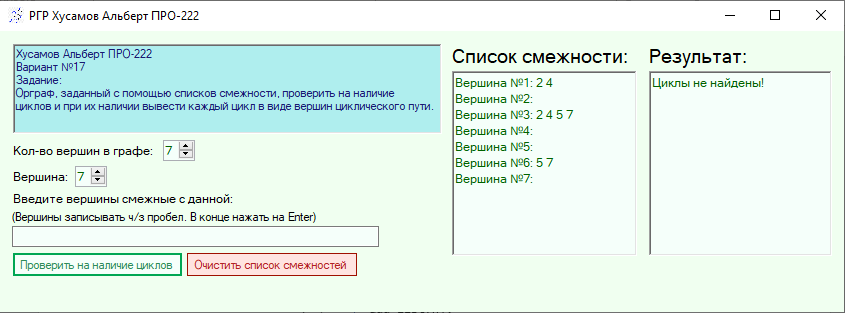


Рис. 5 Демонстрация работы программы

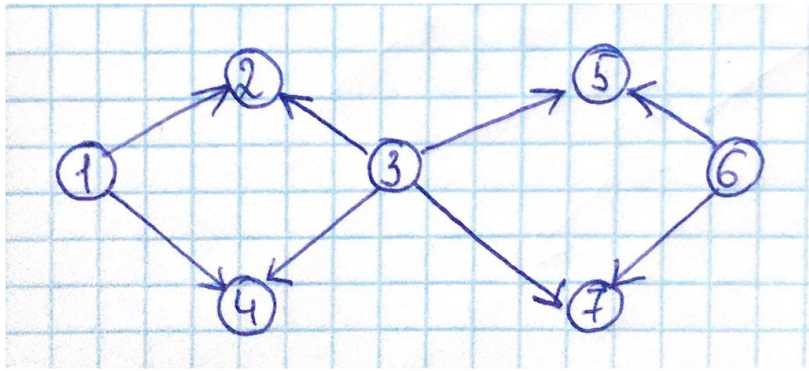


Рис. 6 Орграф

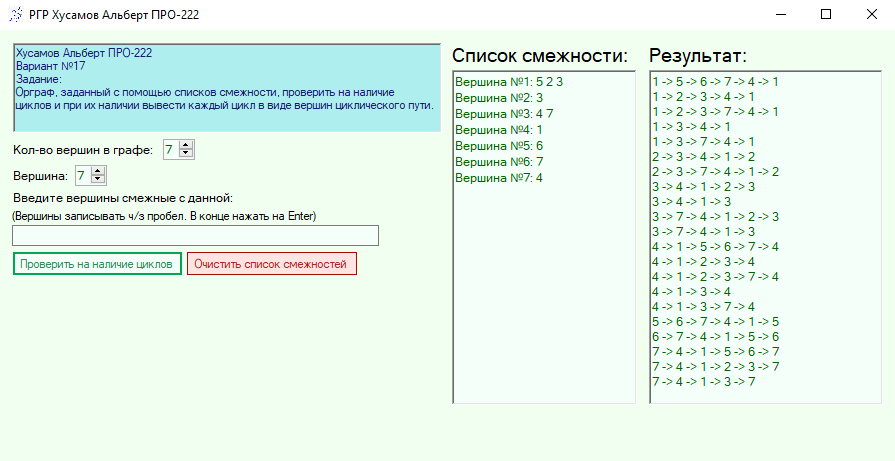


Рис. 7 Демонстрация работы программы

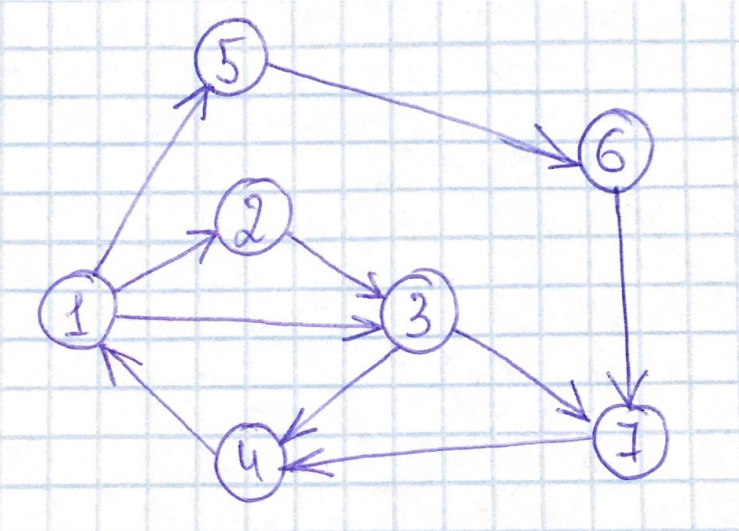


Рис. 8 Орграф

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.