Министерство образование Республики Беларусь

Учреждение образования «Белорусский государственный технологический университет»

Факультет информационных технологий

Кафедра информационных систем и технологий

**Курс «Математическое программирование»**

**Отчет по лабораторной работе №5**

**Алгоритмы на графах**

**(алгоритмы поиска в ширину и глубину, топологическая сортировка)**

**Вариант 10**

Выполнила: студентка факультета ИТ

2 курс 4 группа

Прихач А. А.

Проверил: Бракович А. И.

Минск 2017

**Цель работы:** освоить сущность и программную реализацию: а) способов представления графов; б) алгоритмов поиска в ширину и глубину; в) алгоритма топологической сортировки графов.

**Ход выполнения работы**

**Задание 1.** Ориентированный граф G взять в соответствии с вариантом. Осуществить алгоритмы поиска в ширину и глубину, а также алгоритма топологической сортировки аналогично примерам, рассмотренным на лекциях. Оформить отчет, включив в него каждый шаг выполнения алгоритмов.

**Исходный граф:**



**Обход в ширину**

Для выполнения обхода введем следующие обозначения: Q – очередь вершин, С – массив окраски вершин, D – массив расстояний и P – массив предшествующих вершин.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Q | 0 |  |  |  |  |
| C | G | W | W | W | W |
| D | 0 | I | I | I | I |
| P | N | N | N | N | N |

**Шаг 1**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Q | 1 |  |  |  |  |
| C | B | G | W | W | W |
| D | 0 | 1 | I | I | I |
| P | N | 0 | N | N | N |

**Шаг 2**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Q | 2 |  |  |  |  |
| C | B | B | G | W | W |
| D | 0 | 1 | 2 | I | I |
| P | N | 0 | 1 | N | N |

**Шаг 3**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Q | 3 |  |  |  |  |
| C | B | B | B | G | W |
| D | 0 | 1 | 2 | 3 | I |
| P | N | 0 | 1 | 2 | N |

**Шаг 4**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Q | 4 |  |  |  |  |
| C | B | B | B | B | G |
| D | 0 | 1 | 2 | 3 | 4 |
| P | N | 0 | 1 | 2 | 3 |

**Шаг 5**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Q |  |  |  |  |  |
| C | B | B | B | B | B |
| D | 0 | 1 | 2 | 3 | 4 |
| P | N | 0 | 1 | 2 | 3 |

**Шаг 6**



**Конечный результат**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P | N | 0 | 1 | 2 | 3 |



**Обход в глубину**

Для выполнения обхода введем следующие обозначения: Q – очередь вершин, С – массив окраски вершин, D – массив, где для каждой вершины записывается время обнаружения (шаг окраски в серый цвет), P – массив предшествующих вершин, массив F предназначен для хранения времени фиксации (шага окраски в черный цвет) вершины, t – номер шага алгоритма.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| t | 1 |  |  |  |  |
| C | G | W | W | W | W |
| D | 1 | I | I | I | I |
| P | N | N | N | N | N |
| F | 0 | 0 | 0 | 0 | 0 |

**Шаг 1**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| t | 2 |  |  |  |  |
| C | G | G | W | W | W |
| D | 1 | 2 | I | I | I |
| P | N | 0 | N | N | N |
| F | 0 | 0 | 0 | 0 | 0 |

**Шаг 2**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| t | 3 |  |  |  |  |
| C | G | G | G | W | W |
| D | 1 | 2 | 3 | I | I |
| P | N | 0 | 1 | N | N |
| F | 0 | 0 | 0 | 0 | 0 |

**Шаг 3**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| t | 4 |  |  |  |  |
| C | G | G | G | G | W |
| D | 1 | 2 | 3 | 4 | I |
| P | N | 0 | 1 | 2 | N |
| F | 0 | 0 | 0 | 0 | 0 |

**Шаг 4**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| t | 5 |  |  |  |  |
| C | G | G | G | G | G |
| D | 1 | 2 | 3 | 4 | 5 |
| P | N | 0 | 1 | 2 | 3 |
| F | 0 | 0 | 0 | 0 | 0 |

**Шаг 5**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| t | 6 |  |  |  |  |
| C | G | G | G | G | B |
| D | 1 | 2 | 3 | 4 | 5 |
| P | N | 0 | 1 | 2 | 3 |
| F | 0 | 0 | 0 | 0 | 6 |

**Шаг 6**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| t | 7 |  |  |  |  |
| C | G | G | G | B | B |
| D | 1 | 2 | 3 | 4 | 5 |
| P | N | 0 | 1 | 2 | 3 |
| F | 0 | 0 | 0 | 7 | 6 |

**Шаг 7**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| t | 8 |  |  |  |  |
| C | G | G | B | B | B |
| D | 1 | 2 | 3 | 4 | 5 |
| P | N | 0 | 1 | 2 | 3 |
| F | 0 | 0 | 8 | 7 | 6 |

**Шаг 8**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| t | 9 |  |  |  |  |
| C | G | B | B | B | B |
| D | 1 | 2 | 3 | 4 | 5 |
| P | N | 0 | 1 | 2 | 3 |
| F | 0 | 9 | 8 | 7 | 6 |

**Шаг 9**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| t | 10 |  |  |  |  |
| C | B | B | B | B | B |
| D | 1 | 2 | 3 | 4 | 5 |
| P | N | 0 | 1 | 2 | 3 |
| F | 10 | 9 | 8 | 7 | 6 |

**Шаг 10**



**Конечный результат**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P | N | 0 | 1 | 2 | 3 |



**Топологическая сортировка**

Для топологической сортировки граф должен быть ацикличным, т.е. в нашем графе, мы должны избавиться от циклов.

**Шаг 1**

|  |
| --- |
|  |



**Шаг 2**

|  |
| --- |
|  |



**Шаг 3**

|  |
| --- |
|  |



**Шаг 4**

|  |
| --- |
|  |



**Шаг 5**

|  |
| --- |
|  |



**Шаг 6**

|  |
| --- |
| 4 |



**Шаг 7**

|  |  |
| --- | --- |
| 3 | 4 |



**Шаг 8**

|  |  |  |
| --- | --- | --- |
| 2 | 3 | 4 |



**Шаг 9**

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |



**Шаг 10**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 |



**Конечный результат**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 |



**Задание 2.** Осуществить программную реализацию алгоритмов на C++. Разработать структуры AMatrix и АList для представления ориентированного графа матричным и списковым способом. Разработать функции преобразования из одного способа представления в другой. Разработать функцию BFS обхода вершин графа, используя метод поиска в ширину. Продемонстрировать работу функции. Копии экрана вставить в отчет.

//AMatrix.h

#pragma once

#include <iostream>

#include <vector>

#include <queue>

#include <algorithm>

#define INF INT\_MIN

#define PLM INT\_MAX

class AMatrix

{

private:

std::vector<std::vector<int>> matrix;

public:

AMatrix(std::vector<std::vector<int>> matrix);

std::vector<std::vector<int>> MatrixToList();

void Show();

};

//Alist.h

#pragma once

#include <iostream>

#include <vector>

#include <queue>

#include <algorithm>

#define INF INT\_MIN

#define PLM INT\_MAX

class AList

{

private:

std::queue<int> queue;

std::vector<int> color;

bool loop = false;

std::vector<int> topologicalSort;

void GoToV(int v);

void GoToTopologicalSort(int v);

std::vector<std::vector<int>> list;

public:

AList(std::vector<std::vector<int>> list);

std::vector<std::vector<int>> ListToMatrix();

void BFS();

void DFS();

void TopologicalDFS();

void Show();

};

Преобразование из матричного представления в списковое:

std::vector<std::vector<int>> AMatrix::MatrixToList()

{

std::vector<std::vector<int>> list;

for (int i = 0; i < this->matrix.size(); i++)

{

std::vector<int> temp;

for (int j = 0; j < this->matrix[i].size(); j++)

{

if (this->matrix[i][j] > 0)

{

temp.push\_back(j);

}

}

list.push\_back(temp);

}

return list;

}

Преобразование в матричное представление:

std::vector<std::vector<int>> AList::ListToMatrix()

{

std::vector<std::vector<int>> matrix;

for (int i = 0; i < this->list.size(); i++)

{

std::vector<int> temp;

for (int j = 0; j < this->list.size(); j++)

{

temp.push\_back(INF);

}

matrix.push\_back(temp);

}

for (int i = 0; i < this->list.size(); i++)

{

for (int j = 0; j < this->list[i].size(); j++)

{

if (this->list[i][j] == i)

{

matrix[i][i] = PLM;

}

else

{

if (((matrix[i][this->list[i][j]]) == -1) && ((matrix[this->list[i][j]][i]) == 1))

{

matrix[i][this->list[i][j]] = PLM;

matrix[this->list[i][j]][i] = PLM;

}

else

{

matrix[i][this->list[i][j]] = 1;

matrix[this->list[i][j]][i] = -1;

}

}

}

}

return matrix;

}

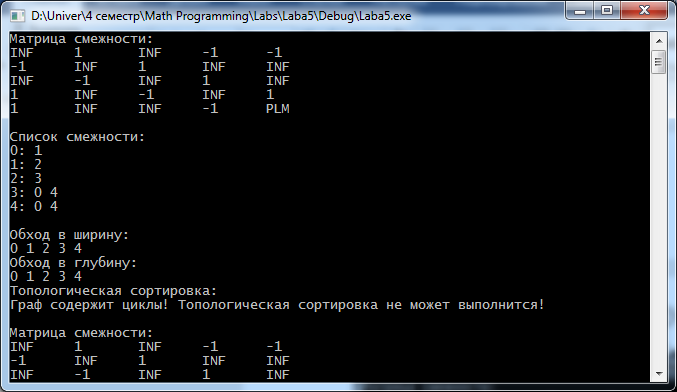


Рисунок 1 – Результат выполнения преобразований

Поиск в ширину:

void AList::BFS()

{

this->color.clear();

while (!this->queue.empty())

{

this->queue.pop();

}

for (int i = 0; i < this->list.size(); i++)

{

this->color.push\_back(0);

}

std::cout << "Обход в ширину:" << std::endl;

for (int i = 0; i < this->list.size(); i++)

{

if ((this->color[i] == 1) && (this->color[i] == 2))

{

continue;

}

this->queue.push(i);

this->color[i] = 1;

while (!this->queue.empty())

{

i = this->queue.front();

this->queue.pop();

this->color[i] = 2;

std::cout << i << " ";

for (int j = 0; j < this->list[i].size(); j++)

{

if (((this->color[this->list[i][j]]) == 2) || ((this->color[this->list[i][j]]) == 1))

{

continue;

}

this->queue.push(this->list[i][j]);

this->color[this->list[i][j]] = 1;

}

}

}

}

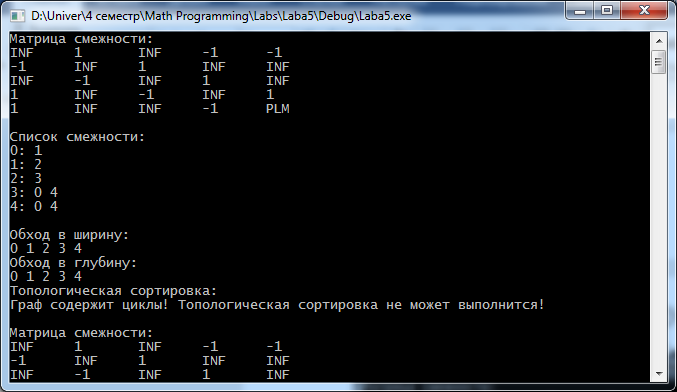


Рисунок 2 – Поиск в ширину

**Задание 3.** Разработать функцию DFS обхода вершин графа, используя метод поиска глубину. Продемонстрировать работу функции. Копии экрана вставить в отчет.

Поиск в глубину:

void AList::DFS()

{

this->color.clear();

for (int i = 0; i < this->list.size(); i++)

{

this->color.push\_back(0);

}

std::cout << "Обход в глубину:" << std::endl;

for (int i = 0; i < this->list.size(); i++)

{

this->GoToV(i);

}

}

void AList::GoToV(int v)

{

if (this->color[v] == 2)

{

return;

}

this->color[v] = 2;

std::cout << v << " ";

for (int i = 0; i < this->list[v].size(); i++)

{

this->GoToV(this->list[v][i]);

}

}

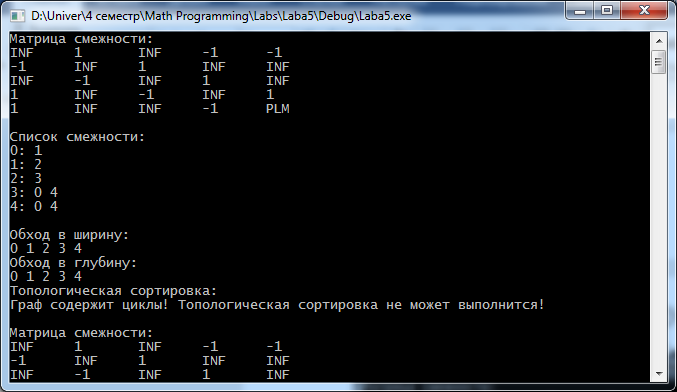


Рисунок 3 – Поиск в глубину

**Задание 4.** Доработайте функцию DFS, для выполнения топологической сортировки графа. Продемонстрировать работу функции. Копии экрана вставить в отчет.

Топологическая сортировка:

void AList::TopologicalDFS()

{

this->color.clear();

this->loop = false;

this->topologicalSort.clear();

for (int i = 0; i < this->list.size(); i++)

{

this->color.push\_back(0);

}

for (int i = 0; i < this->list.size(); i++)

{

this->GoToTopologicalSort(i);

}

if (this->loop == false)

{

std::reverse(topologicalSort.begin(), topologicalSort.end());

}

std::cout << "Топологическая сортировка:" << std::endl;

if (this->loop == true)

{

std::cout << "Граф содержит циклы! Топологическая сортировка не может выполнится!" << std::endl;

}

else

{

for (int i = 0; i < this->list.size(); i++)

{

std::cout << this->topologicalSort[i] << " ";

}

std::cout << std::endl;

}

}

void AList::GoToTopologicalSort(int v)

{

if (this->color[v] == 2 || this->loop == true)

{

return;

}

else if (this->color[v] == 1)

{

this->loop = true;

return;

}

else

{

this->color[v] = 1;

for (int i = 0; i < this->list[v].size(); i++)

{

this->GoToTopologicalSort(this->list[v][i]);

}

this->color[v] = 2;

this->topologicalSort.push\_back(v);

}

}

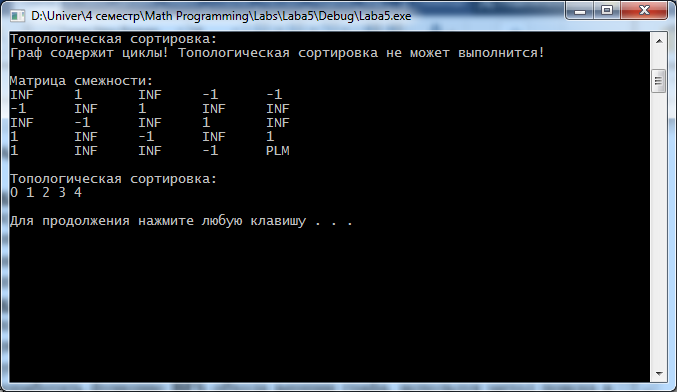


Рисунок 4 – Топологическая сортировка

**Вывод**

В ходе выполнения данной лабораторной работы была освоена сущность и программная реализация способов представления графов (матричный и списковый). Были реализованы алгоритмы поиска в ширину и глубину, а также алгоритм топологической сортировки графов как программно, так и вручную.