Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЁТ**

по лабораторной работе №5

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Обход графа в ширину»

Выполнили:

студенты группы 20ВВ2

Киреев Б.П.

Верховский М.В.

Лукин В.Д.

Приняли:

д.т.н. Митрохин М.А.

к.т.н. доцент Юрова О.В.

Пенза 2021

Название: Обход графа в ширину.

Цель работы: Научиться осуществлять обход графа в ширину.

Лабораторное задание:

**Задание 1:**

1. Сгенерируйте (используя генератор случайных чисел) матрицу

смежности для неориентированного графа G. Выведите матрицу на экран.

2. Для сгенерированного графа осуществите процедуру обхода в ширину,

реализованную в соответствии с приведенным выше описанием. При

реализации алгоритма в качестве очереди используйте класс queue из

стандартной библиотеки С++.

3. Реализуйте процедуру обхода в ширину для графа, представленного списками смежности.

**Задание 2:**

1. Для матричной формы представления графов реализуйте алгоритм

обхода в ширину с использованием очереди, построенной на основе

структуры данных «список», самостоятельно созданной в лабораторной

работе № 3.

2. Оцените время работы двух реализаций алгоритмов обхода в ширину

(использующего стандартный класс queue и использующего очередь,

реализованную самостоятельно) для графов разных порядков.

**Задание 1, 2:**

Листинг:

Файл Lab5.cpp

#include <time.h>

#include <malloc.h>

#include <locale.h>

#include <iostream>

#include <queue>

#include <ctime>

#include <conio.h>

using namespace std;

int i, num = 0, time1, time2;

int\* vis = NULL, n, \*\* p = NULL;

int pogr = 0;

//Вершина / След.элемент; Номер

struct SmegnElem {

SmegnElem\* Nextelem;

int num;

};

//Список смежности

struct SmegnList {

SmegnElem\* First;

int Colo;

} \*\*SpecMatrix = NULL, \* SpecMatrixRezerve = NULL, \* OurQueue = NULL;

//Создание списка смежности

SmegnList\* CreateList() {

SmegnList\* groups = (SmegnList\*)malloc(sizeof(SmegnList));

groups->First = NULL;

groups->Colo = 0;

return groups;

}

//Добавление элемента в список смежности

void AddSmegElem(SmegnList\* group, int Chis) {

SmegnElem\* newItem = (SmegnElem\*)malloc(sizeof(SmegnElem));

newItem->Nextelem = NULL;

newItem->num = Chis;

if (group->Colo == 0) { //При отсутствии элементов записываем как первый

group->First = newItem;

group->Colo++;

return;

}

SmegnElem\* last = group->First; //При наличии других элементов начинаем перебор

while (last->Nextelem != NULL) {

last = last->Nextelem;

}

last->Nextelem = newItem;

group->Colo++;

}

//Отрисовка списка смежности

void PrintSmegElem(SmegnList\* groups) {

if (groups->Colo == 0) { //При размере = 0, у нас нет данных

cout << "Список пуст" << endl;

return;

}

SmegnElem\* current = groups->First;

while (current != NULL) { //Перебор структуры до последнего элемента(с выводом)

int value = current->num;

cout << value << " ";

current = current->Nextelem;

}

cout << endl;

}

void BFS(int\*\* p, int\* vis, int n, int num) {

queue <int> q;

int i;

q.push(num);

vis[num] = 1;

while (!q.empty()) {

num = q.front();

q.pop();

cout << num + 1 << " ";

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

if (vis[i] == 0 && p[num][i] == 1) {

q.push(i);

vis[i] = 1;

}

}

}

}

void VerticalSearchSmeg() {

queue <int> q;

SmegnElem\* Save = SpecMatrix[0]->First;

int VisControl = 1;

bool Checker = false, Checker2 = true;

for (int i = 0; i < n; i++) { vis[i] = 0; } //Обнуление проходимых вершин

Save = SpecMatrix[0]->First;

q.push(1);

vis[0] = 1;

cout << "1 ";

while (!q.empty()) {

Save = SpecMatrix[q.front() - 1]->First;

q.pop();

while (Save != NULL) {

Checker = true;

for (int i = 0; i < n; i++) { if (vis[i] == Save->num) { Checker = false; } }

if (Checker == true) { vis[VisControl] = Save->num; VisControl++; q.push(Save->num); cout << Save->num << " "; }

Save = Save->Nextelem;

}

}

}

void VerticalFullSearchSmeg() {

SmegnElem\* Save = SpecMatrix[0]->First, \* Dopo;

int VisControl = 1;

bool Checker = false;

for (int i = 0; i < n; i++) { vis[i] = 0; } //Обнуление проходимых вершин

OurQueue = CreateList();

AddSmegElem(OurQueue, 1);

Save = SpecMatrix[OurQueue->First->num - 1]->First;

vis[0] = 1;

cout << "1 ";

while (OurQueue->First != NULL) {

Save = SpecMatrix[OurQueue->First->num - 1]->First;

Dopo = OurQueue->First->Nextelem;

OurQueue->First = Dopo;

OurQueue->Colo--;

while (Save != NULL) {

Checker = true;

for (int i = 0; i < n; i++) { if (vis[i] == Save->num) { Checker = false; } }

if (Checker == true) {

vis[VisControl] = Save->num;

VisControl++;

AddSmegElem(OurQueue, Save->num);

cout << Save->num << " ";

}

Save = Save->Nextelem;

}

}

}

int main() {

setlocale(LC\_ALL, "Rus");

srand(time(0));

cout << "Введите размерность графа:";

cin >> n;

vis = (int\*)calloc(n, 3);

p = (int\*\*)calloc(n, 3);

SpecMatrix = (SmegnList\*\*)calloc(n, sizeof(SmegnList\*\*));

SpecMatrixRezerve = (SmegnList\*)calloc(n, sizeof(SmegnList\*));

SpecMatrixRezerve = CreateList();

cout.precision(3 \* n);

vis = (int\*)malloc(sizeof(int) \* n);

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

p[i] = (int\*)calloc(n, 3);

vis[i] = 0;

SpecMatrix[i] = CreateList(); //Создаём список смежности

AddSmegElem(SpecMatrix[i], i + 1); //Вставляем в начало каждого списка смежности номер отвечающей вершины

}

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

for (int m = 0 + pogr; m < n; m++) {

p[i][m] = 0 + rand() % 2;

p[m][i] = p[i][m];

p[i][i] = 0;

}

pogr++;

}

for (int i = 0; i < n; i++) { //Создание списков смежности из матрицы смежности

for (int m = 0; m < n; m++) { if (p[i][m] == 1) { AddSmegElem(SpecMatrix[i], m + 1); } }

}

cout << "\nМатрица 1:" << endl;

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

for (int m = 0; m < n; m++) { cout << p[i][m] << " "; }

cout << endl;

}

cout << "\nСписки смежности для Мартицы №1:" << endl; //Отрисовка списков смежности

for (int i = 0; i < n; i++) { PrintSmegElem(SpecMatrix[i]); }

cout << "\nОбход в ширину(По матрице / Очередь): ";

time1 = clock();

VerticalSearchSmeg();

time2 = clock();

cout << "\nВремя: " << ((float)time2 - (float)time1) / CLOCKS\_PER\_SEC << "\n" << endl;

time1 = time2;

cout << "\nОбход в ширину(По списку смежности / Список): ";

VerticalFullSearchSmeg();

time2 = clock();

cout << "\nВремя: " << ((float)time2 - (float)time1) / CLOCKS\_PER\_SEC << "\n" << endl;

return 0;

}

Пояснительный текст к программе:

Данная программа осуществляет обход графа, представленного матрицей смежности, и обход графа, представленного списками смежности, в ширину. Работа алгоритма продолжается до тех пор, пока существуют непосещённые вершины. Оценили время работы двух алгоритмов обхода в ширину.

Результаты работы программы:

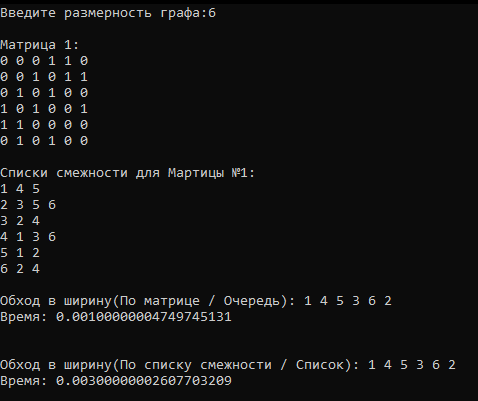


Рисунок №1 – Результат работы программы.

Время обхода в ширину по матрице/очередь: 0,01c

Время обхода в ширину по списку смежности / список: 0,003с

Вывод: При выполнении данной лабораторной работы мы смогли научиться осуществлять обход графа в ширину.