Пензенский государственный университет

Кафедра «Вычислительная техника»

ОТЧЕТ

по лабораторной работе № 9

по дисциплине: "Логика и основы алгоритмизации в инженерных задачах"

## на тему: "Поиск расстояний в графе"

Выполнили студенты гр. 22ВВП2:

Корнилов В.М.

Самофалова А.В.

Горбунов Д.А.

Приняли:

Акифьев И.В.

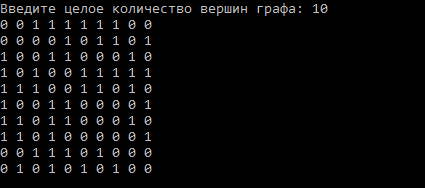
Юрова О.В.

Пенза, 2023

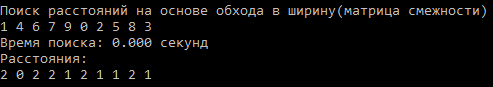
### **Цель:** изучить принцип нахождения вершин графа, используя методы обхода графа в глубину и в ширину.

**Задание 1**

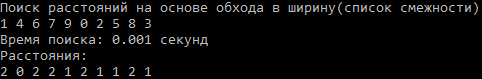
1. Сгенерируйте (используя генератор случайных чисел) матрицу смежности для неориентированного графа *G*. Выведите матрицу на экран.



1. Для сгенерированного графа осуществите процедуру поиска расстояний, реализованную в соответствии с приведенным выше описанием. При  реализации алгоритма в качестве очереди используйте класс **queue** из стандартной библиотеки С++.

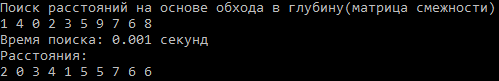


**3.**\* Реализуйте процедуру поиска расстояний для графа, представленного списками смежности.

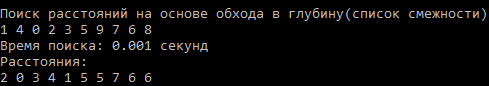


**Задание 2\***

1. Реализуйте процедуру поиска расстояний на основе обхода в глубину.



1. Реализуйте процедуру поиска расстояний на основе обхода в глубину для графа, представленного списками смежности.



1. Оцените время работы реализаций алгоритмов поиска расстояний на основе обхода в глубину и обхода в ширину для графов разных порядков.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Алгоритм/набор данных** | **В ширину (матрица смежности)** | **В ширину (список смежности)** | **В глубину (матрица смежности)** | **В глубину (список смежности)** |
| 10 | 0.000 | 0.001 | 0.001 | 0.001 |
| 100 | 0.004 | 0.003 | 0.005 | 0.003 |
| 500 | 0.016 | 0.018 | 0.017 | 0.019 |
| 1000 | 0.020 | 0.032 | 0.029 | 0.037 |
| 2000 | 0.069 | 0.092 | 0.073 | 0.098 |

На не большом количестве вершин алгоритмы работают за примерно одинаковое время. При увеличении порядка графа алгоритм на основе обхода матрицы смежности в ширину работает быстрее, чем на основе обхода матрицы смежности в глубину. Алгоритмы поиска расстояний в списке смежности работают на всех наборах данных примерно одинаково.

**Вывод:** изучили принцип нахождения вершин графа, используя методы обхода графа в глубину и в ширину.

**Приложение А (Листинг)**

**File 30BALLOV.cpp**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

#include<stdlib.h>

#include<windows.h>

#include<queue>

#include<time.h>

#include<iostream>

using namespace std;

typedef struct Node {

int ver;

struct Node\* next;

} Node;

void addver(Node\*\* List, int x, int y) {

Node\* newNode = (Node\*)malloc(sizeof(Node));

newNode->ver = y;

newNode->next = NULL;

if (List[x] == NULL) {

List[x] = newNode;

}

else {

Node\* temp = List[x];

while (temp->next != NULL) {

temp = temp->next;

}

temp->next = newNode;

}

return;

}

void vivod(Node\*\* List, int n) {

int i;

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

Node\* temp = List[i];

printf("Список смежности вершины %d ", i);

while (temp) {

printf(" -> %d ", temp->ver);

temp = temp->next;

}

printf("\n");

}

return;

}

int BFSD(int\*\* A, int\* dist, int n, int v) {

queue<int> q;

q.push(v);

dist[v] = 0;

while (!q.empty()) {

v = q.front();

printf("%d ", v);

q.pop();

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

if (A[v][i] == 1 && dist[i] == -1) {

q.push(i);

dist[i] = dist[v]+1;

}

}

}

return 0;

}

int BFSD2(Node\*\* List, int\* dist, int n, int v) {

queue<int> q;

q.push(v);

dist[v] = 0;

while (!q.empty()) {

v = q.front();

printf("%d ", v);

q.pop();

Node\* temp = List[v];

while (temp != NULL) {

int i = temp->ver;

if (dist[i] == -1) {

q.push(i);

dist[i] = dist[v] + 1;

}

temp = temp->next;

}

}

return 0;

}

void del(Node\*\* List, int n) {

int i;

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

Node\* temp = NULL;

Node\* head = List[i];

temp = head;

if (head != NULL){

temp = head->next;

}

free(head);

}

}

void DFSD(int\*\* A, int\* vis, int n, int v, int d) {

vis[v] = d;

printf("%d ", v);

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

if ((A[v][i] == 1) && (vis[i] == -1)) {

d = vis[v] + 1;

DFSD(A, vis, n, i, d);

}

}

}

void DFSD2(Node\*\* List, int\* vis, int ver, int d) {

vis[ver] = d;

printf("%d ", ver);

struct Node\* temp = List[ver];

while (temp != NULL) {

int cv = temp->ver;

if (vis[cv]==-1) {

d = vis[ver] + 1;

DFSD2(List, vis, cv, d);

}

temp = temp->next;

}

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int n = 0;

int\*\* A;

int\* dist;

int\* dist1;

int\* vis;

int\* vis1;

int v = 0;

int err;

printf("Введите целое количество вершин графа: ");

err=scanf("%d", &n);

if ((err == 0) || (n <= 0)) {

printf("Ошибка! Введите число заново\n");

return main();

}

dist = (int\*)calloc(n, sizeof(int));

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

dist[i] = -1;

}

srand(time(NULL));

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

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

A[i] = (int\*)malloc(sizeof(int) \* n);

}

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

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

A[i][j] = rand() % 2;

A[j][i] = A[i][j];

if (i == j) {

A[i][j] = 0;

}

}

}

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

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

printf("%d ", A[i][j]);

}

printf("\n");

}

printf("\n");

printf("Введите номер вершины от 0 до %d, с которой хотите начать определение расстояний : ", n-1);

err=scanf("%d", &v);

if ((err == 0) || (v < 0) || (v > n-1)) {

printf("Ошибка! Введите число заново\n");

return main();

}

Node\*\* list1 = (Node\*\*)malloc(n \* sizeof(Node\*));

int i, j;

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

list1[i] = NULL;

}

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

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

if (A[i][j] == 1) {

addver(list1, i, j);

}

}

}

dist1 = (int\*)calloc(n, sizeof(int));

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

dist1[i] = -1;

}

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

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

vis[i] = -1;

}

vis1 = (int\*)calloc(n, sizeof(int));

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

vis1[i] = -1;

}

printf("\n");

printf("\n");

vivod(list1, n);

printf("\n");

time\_t start1 = clock();

printf("\nПоиск расстояний на основе обхода в ширину(матрица смежности)\n");

BFSD(A, dist, n, v);

time\_t stop = clock();

double time2 = ((double)stop - (double)start1) / CLOCKS\_PER\_SEC;

printf("\nВремя поиска: %5.3f секунд", time2);

printf("\nРасстояния:\n");

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

printf("%d ", dist[i]);

}

printf("\n");

printf("\nПоиск расстояний на основе обхода в ширину(список смежности)\n");

start1 = clock();

BFSD2(list1, dist1, n, v);

stop = clock();

time2 = ((double)stop - (double)start1) / CLOCKS\_PER\_SEC;

printf("\nВремя поиска: %5.3f секунд", time2);

printf("\nРасстояния:\n");

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

printf("%d ", dist1[i]);

}

printf("\n");

printf("\nПоиск расстояний на основе обхода в глубину(матрица смежности)\n");

start1 = clock();

DFSD(A, vis, n, v, 0);

stop = clock();

time2 = ((double)stop - (double)start1) / CLOCKS\_PER\_SEC;

printf("\nВремя поиска: %5.3f секунд", time2);

printf("\nРасстояния:\n");

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

printf("%d ", vis[i]);

}

printf("\n");

printf("\nПоиск расстояний на основе обхода в глубину(список смежности)\n");

start1 = clock();

DFSD2(list1, vis1, v, 0);

stop = clock();

time2 = ((double)stop - (double)start1) / CLOCKS\_PER\_SEC;

printf("\nВремя поиска: %5.3f секунд", time2);

printf("\nРасстояния:\n");

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

printf("%d ", vis1[i]);

}

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

free(A[i]);

free(A);

free(dist);

free(dist1);

free(vis);

free(vis1);

del(list1, n);

return 0;

}