Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра “Вычислительная техника”

**Отчет**

по лабораторной работе № 9

по курсу “Логика и основы алгоритмизации в инженерных задачах”

на тему “Поиск расстояний в графе”

Выполнили студенты группы 22ВВС1:

Разин Д.С

Беккаревич К.А

Приняли:

Акифьев И. В.

Юрова О.В

Пенза 2023

**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 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.
2. Реализуйте процедуру поиска расстояний на основе обхода в глубину для графа, представленного списками смежности.
3. Оцените время работы реализаций алгоритмов поиска расстояний на основе обхода в глубину и обхода в ширину для графов разных порядков.

**Задание №9.1.2**

#define\_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

#include<queue>

#include<clocale>

#defineMAX\_VERTICES100

// Функция для генерации случайной матрицы смежности

voidgenerrand(intmatrix[MAX\_VERTICES][MAX\_VERTICES],intnumVertices){

inti,j;

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

for(j=i;j<numVertices;j++){

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

matrix[j][i]=matrix[i][j];

if(i==j)matrix[i][j]=0;

}

}

}

// Функция для поиска расстояний в графе с использованием BFS

voidbfs(intmatrix[MAX\_VERTICES][MAX\_VERTICES],intnumVertices,intstartNode){

boolvisited[MAX\_VERTICES]={false};

intdistances[MAX\_VERTICES]={0};

std::queue<int>q;

q.push(startNode);

visited[startNode]=true;

while(!q.empty()){

intcurrentNode=q.front();

q.pop();

for(inti=0;i<numVertices;i++){

if(matrix[currentNode][i]&&!visited[i]){

q.push(i);

visited[i]=true;

distances[i]=distances[currentNode]+1;

}

}

}

// Выводим расстояния от startNode до всех остальных узлов

for(inti=0;i<numVertices;i++){

printf("Расстояние от ребра %d до ребра %d: %d\n",startNode,i,distances[i]);

}

}

intmain(){

setlocale(LC\_ALL,"Russian");

srand(time(NULL));

intnumVertices;

printf("Введите количество вершин в графе: ");

scanf("%d",&numVertices);

intmatrix[MAX\_VERTICES][MAX\_VERTICES]={0};

generrand(matrix,numVertices);

printf("Матрица смежности графа:\n");

for(inti=0;i<numVertices;i++){

for(intj=0;j<numVertices;j++){

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

}

printf("\n");

}

intstartNode;

printf("Введите начальный узел для поиска расстояний: ");

scanf("%d",&startNode);

clock\_tstart=clock();

bfs(matrix,numVertices,startNode);

clock\_tend=clock();

doubletime\_spent5=(double)(end-start)/CLOCKS\_PER\_SEC;

printf("time adjmatrix v wiriny: %f s.\n",time\_spent5);

getchar();

getchar();

return0;

}

**Задание №9.1.3**

#define\_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

#include<queue>

#include<clocale>

#defineMAX\_VERTICES100

// Функция для генерации случайной матрицы смежности

voidgenerrand(intmatrix[MAX\_VERTICES][MAX\_VERTICES],intnumVertices){

inti,j;

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

for(j=i;j<numVertices;j++){

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

matrix[j][i]=matrix[i][j];

if(i==j)matrix[i][j]=0;

}

}

}

// Процедура для вывода списка смежности графа

voidprintAdjacencyList(intadjacencyList[MAX\_VERTICES][MAX\_VERTICES],intnumVertices){

printf("Список смежности графа:\n");

for(inti=0;i<numVertices;i++){

printf("Вершина %d: ",i);

for(intj=0;j<numVertices;j++){

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

printf("%d ",j);

}

}

printf("\n");

}

}

// Функция для поиска расстояний в графе с использованием BFS

voidbfs(intmatrix[MAX\_VERTICES][MAX\_VERTICES],intnumVertices,intstartNode){

boolvisited[MAX\_VERTICES]={false};

intdistances[MAX\_VERTICES]={0};

std::queue<int>q;

q.push(startNode);

visited[startNode]=true;

while(!q.empty()){

intcurrentNode=q.front();

q.pop();

for(inti=0;i<numVertices;i++){

if(matrix[currentNode][i]&&!visited[i]){

q.push(i);

visited[i]=true;

distances[i]=distances[currentNode]+1;

}

}

}

// ВыводимрасстоянияотstartNodeдовсехостальныхузлов

for(inti=0;i<numVertices;i++){

printf("Расстояние от ребра %d дор ебра %d: %d\n",startNode,i,distances[i]);

}

}

intmain(){

setlocale(LC\_ALL,"Russian");

srand(time(NULL));

intnumVertices;

printf("Введите количество вершин в графе: ");

scanf("%d",&numVertices);

intmatrix[MAX\_VERTICES][MAX\_VERTICES]={0};

generrand(matrix,numVertices);

printf("Матрица смежности графа:\n");

for(inti=0;i<numVertices;i++){

for(intj=0;j<numVertices;j++){

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

}

printf("\n");

}

printAdjacencyList(matrix,numVertices);

intstartNode;

printf("Введите начальный узел для поиска расстояний: ");

scanf("%d",&startNode);

clock\_tstart=clock();

bfs(matrix,numVertices,startNode);

clock\_tend=clock();

doubletime\_spent5=(double)(end-start)/CLOCKS\_PER\_SEC;

printf("time adjmatrix v wiriny: %f s.\n",time\_spent5);

return0;

}

**Задание 9.2.1.**

#define\_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

#include<stdlib.h>

#include<clocale>

#include<time.h>

#defineMAX\_VERTICES400

#defineINFINITY99999

// Генерация случайной матрицы смежности

voidgenerrand(intmatrix[MAX\_VERTICES][MAX\_VERTICES],intnumVertices){

inti,j;

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

for(j=i;j<numVertices;j++){

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

matrix[j][i]=matrix[i][j];

if(i==j)matrix[i][j]=0;

}

}

}

voiddepthFirstSearch(intmatrix[MAX\_VERTICES][MAX\_VERTICES],boolvisited[MAX\_VERTICES],intdistance[MAX\_VERTICES],intcurrentVertex,intnumVertices){

visited[currentVertex]=true;

for(inti=0;i<numVertices;i++){

if(matrix[currentVertex][i]==1&&!visited[i]){

distance[i]=distance[currentVertex]+1;// Увеличиваемрасстояниена 1

depthFirstSearch(matrix,visited,distance,i,numVertices);

}

}

}

intmain(){

setlocale(LC\_ALL,"Russian");

srand(time(NULL));

intnumVertices;

printf("Введите количество вершин в графе: ");

scanf("%d",&numVertices);

intadjacencyMatrix[MAX\_VERTICES][MAX\_VERTICES];

generrand(adjacencyMatrix,numVertices);

printf("Матрица смежности:\n");

for(inti=0;i<numVertices;i++){

for(intj=0;j<numVertices;j++){

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

}

printf("\n");

}

boolvisited[MAX\_VERTICES];

intdistance[MAX\_VERTICES];

for(inti=0;i<numVertices;i++){

visited[i]=false;

distance[i]=INFINITY;// Исходно все расстояния равны бесконечности

}

intstartVertex;

printf("Введите стартовую вершину (от 0 до %d): ",numVertices-1);

scanf("%d",&startVertex);

clock\_tstart=clock();

if(startVertex>=0&&startVertex<numVertices){

distance[startVertex]=0;// Расстояниеотначальнойвершиныдосамойсебяравно 0

depthFirstSearch(adjacencyMatrix,visited,distance,startVertex,numVertices);

printf("Результат обхода в глубину, начиная с вершины %d:\n",startVertex);

for(inti=0;i<numVertices;i++){

printf("Расстояние от вершины %d до вершины %d: %d\n",startVertex,i,distance[i]);

}

}

else{

printf("Некорректный ввод стартовой вершины.\n");

}

clock\_tend=clock();

doubletime\_spent5=(double)(end-start)/CLOCKS\_PER\_SEC;

printf("time adjmatrix v glybiny: %f s.\n",time\_spent5);

return0;

}

**Задание 9.2.2.**

#define\_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

#include<stdlib.h>

#include<clocale>

#include<time.h>

#defineMAX\_VERTICES100

intgraph[MAX\_VERTICES][MAX\_VERTICES];

voidaddEdge(intfrom,intto){

graph[from][to]=1;

graph[to][from]=1;// Учтем симметрию для неориентированного графа

}

voidinitializeGraph(intvertices){

for(inti=0;i<vertices;++i){

for(intj=0;j<vertices;++j){

graph[i][j]=0;

}

}

}

voidprintAdjacencyList(intvertices){

printf("Список смежности:\n");

for(inti=0;i<vertices;++i){

printf("%d: ",i);

for(intj=0;j<vertices;++j){

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

printf("%d ",j);

}

}

printf("\n");

}

}

voiddfs(intcurrentVertex,intvertices,boolvisited[],intdistances[]){

visited[currentVertex]=true;

for(inti=0;i<vertices;++i){

if(graph[currentVertex][i]==1&&!visited[i]){

distances[i]=distances[currentVertex]+1;

dfs(i,vertices,visited,distances);

}

}

}

voidfindDistancesDFS(intstartVertex,intvertices,intdistances[]){

boolvisited[MAX\_VERTICES]={false};

distances[startVertex]=0;

dfs(startVertex,vertices,visited,distances);

}

intmain(){

intvertices,edges;

setlocale(LC\_ALL,"Russian");

srand(time(NULL));

// Введите количество вершин и рёбер

printf("Введите количество вершин графа: ");

scanf("%d",&vertices);

printf("\nВведите количество рёбер графа: ");

scanf("%d",&edges);

initializeGraph(vertices);

// Заполнение списка рёбрами автоматически

for(inti=0;i<edges;++i){

intfrom=rand()%vertices;

intto;

do{

to=rand()%vertices;

}while(from==to);

addEdge(from,to);

}

printf("Матрица смежности:\n");

for(inti=0;i<vertices;i++){

for(intj=0;j<vertices;j++){

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

}

printf("\n");

}

printAdjacencyList(vertices);

intstartVertex;

// Введите начальную вершину

printf("Введите начальную вершину: ");

scanf("%d",&startVertex);

intdistances[MAX\_VERTICES];

for(inti=0;i<vertices;++i){

distances[i]=INT\_MAX;

}

findDistancesDFS(startVertex,vertices,distances);

// Выведите расстояния от начальной вершины до всех остальных

printf("Расстояния от вершины %d до всех остальных (по обходу в глубину):\n",startVertex);

for(inti=0;i<vertices;++i){

if(distances[i]==INT\_MAX){

printf("До вершины %d: Недостижима\n",i);

}

else{

printf("Довершины %d: %d\n",i,distances[i]);

}

}

getchar();

getchar();

return0;

}

**Результаты работы программ**

Изображение выглядит как текст, снимок экрана, программное обеспечение, компьютер

Автоматически созданное описание

**Рисунок 1 - Результат работы программы lab9.12**

Изображение выглядит как текст, снимок экрана, программное обеспечение, компьютер

Автоматически созданное описание

**Рисунок 2 - Результат работы программы lab9.1.3**

Изображение выглядит как текст, снимок экрана, программное обеспечение, компьютер

Автоматически созданное описание

**Рисунок 3 - Результат работы программы lab9.2.1**

Изображение выглядит как текст, снимок экрана, программное обеспечение, компьютер

Автоматически созданное описание

**Рисунок 4 - Результат работы программы lab9.2.2**

**Задание 9.2.3**

**Обход в глубину:**

**20 ребер:** 

**100 ребер:** 

**200 ребер:** 

**300 ребер:** 

**400 ребер**: 

**Обход в ширину:**

**20 ребер:** 

**100 ребер:** 

**200 ребер:**

**300 ребер:** 

**400 ребер:** 

**Итог:** по представленным данным видно, что поиск длин между вершинами в графе эффективнее выполняется при помощи поиска в глубину.

**Вывод**

В результате выполнения лабораторной работы были успешно реализованы алгоритмы построения матрицы смежности, усвоены и практически проверены навыки в алгоритме реализации поиска расстояний в графе методом обхода графа в ширину и в глубину, также сравнили скорость работы программы на разных размерах матриц и выявили какой из методов быстрее.