Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №1**

По дисциплине «Математические основы интеллектуальных систем»

Тема: «Нахождение компонент связности неориентированного графа»

**Выполнил:**

Студент 2 курса

Группы ИИ-21

Литвинюк Т. В.

**Проверил:**

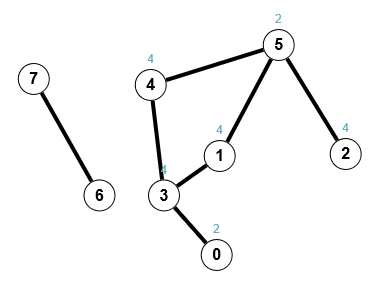
Козинский А. А.

Брест 2022

**Цель:** научитьсянаходить компонент связности неориентированного графа.

**Ход работы:**

**Задание 1**

Построить матрицу смежности и инцидентности для заданного графа. Изобразить граф.

Файл connection.txt:

(1,4),(2,4),(2,6),(3,6),(4,5),(5,6),(7,8)

Код программы:

#include <stdio.h>

int main() {

FILE \*file;

char numbers[100];

const int tops = 8, edges = 7; // кол-во вершин и ребер

int connections[edges][2];

file = fopen("connections.txt", "r");

fgets(numbers, 256, file);

char temp[] = "00";

int i = 0, t = 1, n = 0, m = 0;

while (i < 41){

while (numbers[++i] != ',')

temp[t--] = numbers[i];

connections[n][m++] = (temp[0] - 48) \* 10 + temp[1] - 48;

temp[0] = '0';

t = 1;

while (numbers[++i] != ')')

temp[t--] = numbers[i];

connections[n++][m--] = (temp[0] - 48) \* 10 + temp[1] - 48;

temp[0] = '0';

i += 2;

t = 1;

}

// Матрица смежности

int adjacencyMatrix[tops][tops];

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

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

adjacencyMatrix[i][j] = 0;

}

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

adjacencyMatrix[connections[i][0] - 1][connections[i][1] - 1] = 1;

adjacencyMatrix[connections[i][1] - 1][connections[i][0] - 1] = 1;

}

printf("{");

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

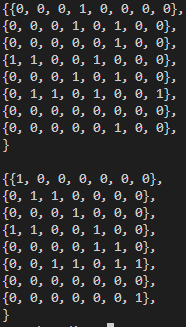
printf("{");

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

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

printf("\b\b},\n");

}

 printf("}\n\n");

// Матрица инцидентности

int incidenceMatrix[tops][edges];

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

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

incidenceMatrix[i][j] = 0;

}

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

incidenceMatrix[connections[i][0] - 1][i] = 1;

incidenceMatrix[connections[i][1] - 1][i] = 1;

}

printf("{");

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

printf("{");

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

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

printf("\b\b},\n");

}

printf("}\n");

return 0;

}

**Задание 2**

Используя поиск в ширину написать программу, определяющую число компонент связности графа. Методы представляются в виде функций.

#include <stdio.h>

#include <stdbool.h>

int bfs(int (\*graph)[8], int vertex){

bool visited[8], available[8], run = true;

for (int i = 0; i < 8; i++)

visited[i] = available[i] = false;

int number = 0;

while(run){

available[vertex] = visited[vertex] = true;

for (int c = 0; c < 8; c++){

for (int i = 0; i < 8; i++){

if (graph[vertex][i])

available[i] = true;

}

for (int i = 0; i < 8; i++){

if (available[i] && !visited[i])

{vertex = i; visited[i] = true; break;}

}

}

number++;

for (int i = 0; i < 8; i++){

if(!visited[i]) {vertex = i; run = true; break;}

else run = false;

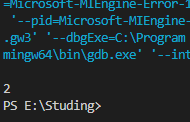
}

}

return number;

}

int main(){

 int graph[8][8] = {{0, 0, 0, 1, 0, 0, 0, 0},

{0, 0, 0, 1, 0, 1, 0, 0},

{0, 0, 0, 0, 0, 1, 0, 0},

{1, 1, 0, 0, 1, 0, 0, 0},

{0, 0, 0, 1, 0, 1, 0, 0},

{0, 1, 1, 0, 1, 0, 0, 0},

{0, 0, 0, 0, 0, 0, 0, 1},

{0, 0, 0, 0, 0, 0, 1, 0}};

printf("%d", bfs(graph,0));

}

**Задание 3**

Используя поиск в глубину написать программу, определяющую число компонент связности графа. Методы представляются в виде функций.

#include <stdio.h>

void DFS(int s, int max\_tops, int \*matrx[], char \*used;

used[s] = 1;

for (int i = 0; i < max\_tops; i++) {

if (used[i] == 0 && matrx[s-1][i-1] == 1)

DFS(i, max\_tops,matrx , used);

}

}

int main(){

int max\_tops = 8;

int graph[8][8] = {{0, 0, 0, 1, 0, 0, 0, 0},

{0, 0, 0, 1, 0, 1, 0, 0},

{0, 0, 0, 0, 0, 1, 0, 0},

{1, 1, 0, 0, 1, 0, 0, 0},

{0, 0, 0, 1, 0, 1, 0, 0},

{0, 1, 1, 0, 1, 0, 0, 0},

{0, 0, 0, 0, 0, 0, 0, 1},

{0, 0, 0, 0, 0, 0, 1, 0}};

char used[max\_tops];

int \*buf[max\_tops], cnt = 0;

for(int i = 0; i < max\_tops; i++){//цикл для получения возможности передачи двумерного

buf[i]=graph[i]; //массива в функцию

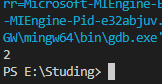
}

DFS(2,max\_tops,buf,used);

for (int i = 0; i < max\_tops; i++) {

if (used[i] == 0) {

DFS(i, max\_tops, buf, used);

 cnt++;

}

}

printf("%d", cnt);

return 0;

}

**Вывод:** в ходе лабораторной работы я научился работать с графами, научился находить компонент связности.