Министерство образования Республики Беларусь
Учреждение образования
“Брестский Государственный технический университет”
Кафедра ИИТ

Лабораторная работа №3
По дисциплине “Математические основы интеллектуальных систем”
Тема: “Нахождение кратчайшего пути в графе”

Выполнил:
Студент 2 курса
Группы ИИ-21
Кирилович А. А.
Проверил:
Козинский А. А.

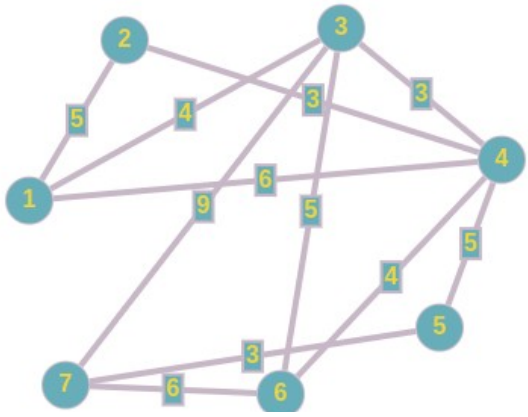
Брест 2022

Цель: научиться находить кратчайший путь во взвешенном графе с помощью алгоритмов Дейкстры и Флойда-Уоршалла.

Ход работы:

Задание 1

Алгоритмом Дейкстры вычислить кратчайшие пути от вершины x_1 ко всем вершинам графа.
Граф:
{1,2},{1,3},{1,4},{2,4},{3,4},{3,6},{3,7},{4,5},{4,6},{5,7},{6,7}
[5,4,6,3,3,5,9,5,4,3,6]




```
#include <iostream>
#include "../graph_LIB.hh"

int main() {
	convert c;
	std::string file_path = "connections.txt";
	std::vector<int> nodes = c.reading_file(file_path);
	file_path = "weights.txt";
	std::vector<int> weights = c.reading_file(file_path);

	int max_node = c.count_of_nodes(nodes);
	std::vector<std::vector<int>> adjacencyMatrix =
c.adjancy(nodes, weights, max_node);

	alg search;
	int start = 0;
	std::vector<int> distance;
	std::vector<int> path(max_node);
	distance = search.Dejkstra(adjacencyMatrix, start, path);
	std::vector<int> way;
	for (int i = 0; i < max_node; i++) {
		std::cout << std::endl << start << "->" << i << " : " << distance[i];
		std::cout << "\tPath:";
		search.thisIsTheWay(path, start, i, way);
		for (int j = 0; j < way.size(); j++) {
			std::cout << way[j];
		}
	way.clear();
	}
}
```

Реализация функции Dejkstra из файла Graph_LIB:

```
std::vector<int> Dejkstra(std::vector<std::vector<int>> &adjacencyMatrix, int start, std::vector<int>
&path) {
	int max_node = adjacencyMatrix.size();
	std::vector<int> distance(max_node);
	std::vector<int> visited(max_node);
	for (int i = 0; i < max_node; i++) {
		distance[i] = INT_MAX;
		visited[i] = 0;
	}
	distance[start] = 0;
	for (int i = 0; i < max_node; i++) {
		int min = INT_MAX;
		int min_index = 0;
		for (int j = 0; j < max_node; j++) {
			if (visited[j] == 0 && distance[j] <= min) {
				min = distance[j];
				min_index = j;
			}
		}
		visited[min_index] = 1;
		for (int j = 0; j < max_node; j++) {
			if (visited[j] == 0
			&& adjacencyMatrix[min_index][j] != 0
			&& distance[min_index] != INT_MAX
			&& distance[min_index] + adjacencyMatrix[min_index][j] < distance[j]) {
				distance[j] = distance[min_index] + adjacencyMatrix[min_index][j];
```

```
0->0 : 0        Path:00
0->1 : 5        Path:01
0->2 : 4        Path:02
0->3 : 6        Path:03
0->4 : 11       Path:034
0->5 : 9        Path:025
0->6 : 13       Path:026
```

```
                        path[j] = min_index;
                    }
                }
        }

        path[start] = start;
        return distance;
}
```

Реализация функции thisIsTheWay из файла Graph_LIB:

```
void thisIsTheWay(std::vector<int> &path, int start, int end, std::vector<int> &way) {
        do {
                way.push_back(end);
                end = path[end];
        } while (end != start);
        way.push_back(start);

        int n = way.size();
        for (int i = 0; i < n / 2; i++) {
                std::swap(way[i], way[n - 1 - i]);
        }
}
```

Задание 2

Алгоритмом Флойда-Уоршолла вычислить кратчайшие пути от вершины x_1 ко всем вершинам графа.

```
#include <iostream>
#include <iomanip>
#include "../graph_LIB.hh"

int main() {
        convert c;
        std::string file_path = "connections.txt";
        std::vector<int> nodes = c.reading_file(file_path);
        file_path = "weights.txt";
        std::vector<int> weights = c.reading_file(file_path);

        int max_node = c.count_of_nodes(nodes);
        std::vector<std::vector<int>> adjacencyMatrix = c.adjancy(nodes, weights, max_node);

        alg search;
        int start = 0;
        std::vector<std::vector<int>> distance;
        std::vector<std::vector<int>> path(max_node, std::vector<int>(max_node));
        distance = search.FloydWarshall(adjacencyMatrix, path);

        std::vector<int> way;
        for (int i = 0; i < max_node; i++) {
                std::cout << std::endl << start << "->" << i << " : " << distance[start][i];
                std::cout << "\tPath:";
                search.thisIsTheWay(path, start, i, way);
                for (int j = 0; j < way.size(); j++) {
                        std::cout << way[j];
                }
                way.clear();
        }
}
```

Реализация функции FloydWarshall из файла Graph_LIB:

```
std::vector<std::vector<int>> FloydWarshall(std::vector<std::vector<int>> &adjacencyMatrix,
std::vector<std::vector<int>> &path) {
        int max_node = adjacencyMatrix.size();
        std::vector<std::vector<int>> distance(max_node, std::vector<int>(max_node));
        for (int i = 0; i < max_node; i++) {
                for (int j = 0; j < max_node; j++) {
                        if (adjacencyMatrix[i][j] == 0 && i != j) {
```

```
                        distance[i][j] = INT_MAX;
                }
                else {
                        distance[i][j] = adjacencyMatrix[i][j];
                }
        }
    }
    for (int k = 0; k < max_node; k++) {
        for (int i = 0; i < max_node; i++) {
                for (int j = 0; j < max_node; j++) {
                        if (distance[i][k] != INT_MAX
                        && distance[k][j] != INT_MAX
                        && distance[i][k] + distance[k][j] < distance[i][j]) {
                                distance[i][j] = distance[i][k] + distance[k][j];
                                path[i][j] = k;
                        }
                }
        }
    }

    for (int i = 0; i < max_node; i++) {
        path[i][i] = i;
    }
    for (int i = 0; i < max_node; i++) {
        for (int j = 0; j < max_node; j++) {
                if (path[i][j] == 0) {
                        path[i][j] = i;
                }
        }
    }
    return distance;
}
```

```
0->0 : 0        Path:00
0->1 : 5        Path:01
0->2 : 4        Path:02
0->3 : 6        Path:03
0->4 : 11       Path:034
0->5 : 9        Path:025
0->6 : 13       Path:026
```

Реализация функции thisIsTheWay из файла Graph_LIB:

```
void thisIsTheWay(std::vector<std::vector<int>> &path, int start, int end, std::vector<int> &way) {
    do {
        way.push_back(end);
        end = path[start][end];
    } while (end != start);
    way.push_back(start);

    int n = way.size();
    for (int i = 0; i < n / 2; i++) {
        std::swap(way[i], way[n - 1 - i]);
    }
}
```

Вывод: в ходе лабораторной работы я научился находить кратчайшие пути в графах от любых вершин до любых алгоритмом Дейкстры и алгоритмом Флойда-Уоршалла.