МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
"МИРЭА - Российский технологический университет"

РТУ МИРЭА

Институт информационных технологий
Кафедра математического обеспечения и стандартизации ИТ

ОТЧЕТ
ПО ПРАКТИЧЕСКОЙ РАБОТЕ №2.5: Основные алгоритмы работы с графами.
ПО ДИСЦИПЛИНЕ
« СТРУКТУРЫ И АЛГОРИТМЫ ОБРАБОТКИ ДАННЫХ»

Выполнил студент группы ИКБО-01-21 Маров Г.А.

Принял старший преподаватель Туманова М.Б.

Практическая работа выполнена «17» октября 2022 г. _____ *(подпись студента)*

«Зачтено» « » сентября 2022 г. _____ *(подпись руководителя)*

Москва 2022

Цель: Получить знания и навыки работы с графами. Написать программу для поиска минимального расстояния между вершинами с помощью метода Беллмана-Форда и протестировать её на практике.

Ход работы

Индивидуальный вариант (24): Нахождение кратчайшего пути методом Беллмана-Форда.

Математическая модель решения:

Метод состоит из трех шагов:

1. На этом шаге инициализируются расстояния от исходной вершины до всех остальных вершин, как бесконечные, а расстояние до самого src принимается равным 0. Создается массив dist[] размера |V| со всеми значениями равными бесконечности, за исключением элемента dist[src], где src — исходная вершина.

2. Вторым шагом вычисляются самые короткие расстояния. Следующие шаги нужно выполнять |V|-1 раз, где |V| — число вершин в данном графе.

   - Произведите следующее действие для каждого ребра u-v:

     Если dist[v] > dist[u] + вес ребра uv, то обновите dist[v]

     dist [v] = dist [u] + вес ребра uv

3. На этом шаге сообщается, присутствует ли в графе цикл отрицательного веса. Для каждого ребра u-v необходимо выполнить следующее:

   - Если dist[v] > dist[u] + вес ребра uv, то в графе присутствует цикл отрицательного веса.

Идея шага 3 заключается в том, что шаг 2 гарантирует кратчайшее расстояние, если граф не содержит цикла отрицательного веса. Если мы снова переберем все ребра и получим более короткий путь для любой из вершин, это будет сигналом присутствия цикла отрицательного веса.

Как это работает? Как и в других задачах динамического программирования, алгоритм вычисляет кратчайшие пути снизу вверх. Сначала он вычисляет самые короткие расстояния, то есть пути длиной не более, чем в одно ребро. Затем он вычисляет кратчайшие пути длиной не более двух ребер и так далее. После i-й итерации внешнего цикла вычисляются кратчайшие пути длиной не более i ребер. В любом простом

пути может быть максимум |V|-1 ребер, поэтому внешний цикл выполняется именно |V|-1 раз. Идея заключается в том, что если мы вычислили кратчайший путь с не более чем i ребрами, то итерация по всем ребрам гарантирует получение кратчайшего пути с не более чем i + 1 ребрами.

Код программы с комментариями:

practice5.cpp:

```
#include <iostream>
#include <map>
#include <vector>
#include <string>
#include <iomanip>
#include <vector>
using namespace std;

// Class to represent a graph
class Graph {
private:
    int V;
    map<pair<string, string> , int> graph; // Map to store graph
    vector<string> vertices; // vector with all vertices
public:
    Graph(int vertices) {
        V = vertices; // No. of vertices
        for (int i = 0; i < V; i++) {
            string tmp;
            cout << "enter " << i + 1 << " vertex: ";
            cin >> tmp;
            this->vertices.push_back(tmp);
        }
    }

    // function to add an edge to graph
    void addEdge(string u, string v, int w) {
        graph[make_pair(u, v)] = w;
    }

    // utility function used to print the solution
    void printArr(map<string, int> dist) {
        cout << "vertex     distance from source" << endl;
        for (auto el : dist)
            cout << left << setw(20) << el.first << el.second << endl;
    }

    // The main function that finds shortest distances from src to
    // all other vertices using Bellman - Ford algorithm. The function
    // also detects negative weight cycle
    void BellmanFord(string src) {
        map<string, int> dist;

        // Step 1: Initialize distances from src to all other vertices
        // as ~inf
        for (string v : vertices)
            dist[v] = 10000000;
        dist[src] = 0;

        // Step 2: Relax all edges |V| - 1 times. A simple shortest
        // path from src to any other vertex can have at-most |V| - 1
```

```
        // edges
        for (int i = 0; i < V - 1; i++)
            // Update dist value and parent index of the adjacent vertices of
            // the picked vertex. Consider only those vertices which are still in
            // queue
            for (auto el : graph)
                if (dist[el.first.first] != 10000000 && dist[el.first.first] + el.second
< dist[el.first.second])
                    dist[el.first.second] = dist[el.first.first] + el.second;

        // Step 3: check for negative-weight cycles. The above step
             // guarantees shortest distances if graph doesn't contain
             // negative weight cycle. If we get a shorter path, then there
             // is a cycle.
        for (auto el : graph)
            if (dist[el.first.first] != 10000000 && dist[el.first.first] + el.second <
dist[el.first.second]) {
                cout << "Graph contains negative weight cycle";
                return;
            }

        // print all distance
        printArr(dist);
    }
};
int main()
{
    cout << "enter amount of vertices: ";
    int n;
    cin >> n;
    Graph* graph = new Graph(n);
    cout << "enter amount of edges: ";
    int n_edges;
    cin >> n_edges;
    for (int i = 0; i < n_edges; i++) {
        cout << "--- entering " << i + 1 << " edge ---" << endl;
        string u, v;
        int w;
        cout << "enter 1st vertex: ";
        cin >> u;
        cout << "enter 2nd vertex: ";
        cin >> v;
        cout << "enter edge weight: ";
        cin >> w;
        graph->addEdge(u, v, w);
    }
    cout << "enter src vertex to use Bellman Ford alg: ";
    string src;
    cin >> src;
    graph->BellmanFord(src);
}
```

Тестирование

Тестирование программы приведено на рисунке 1.

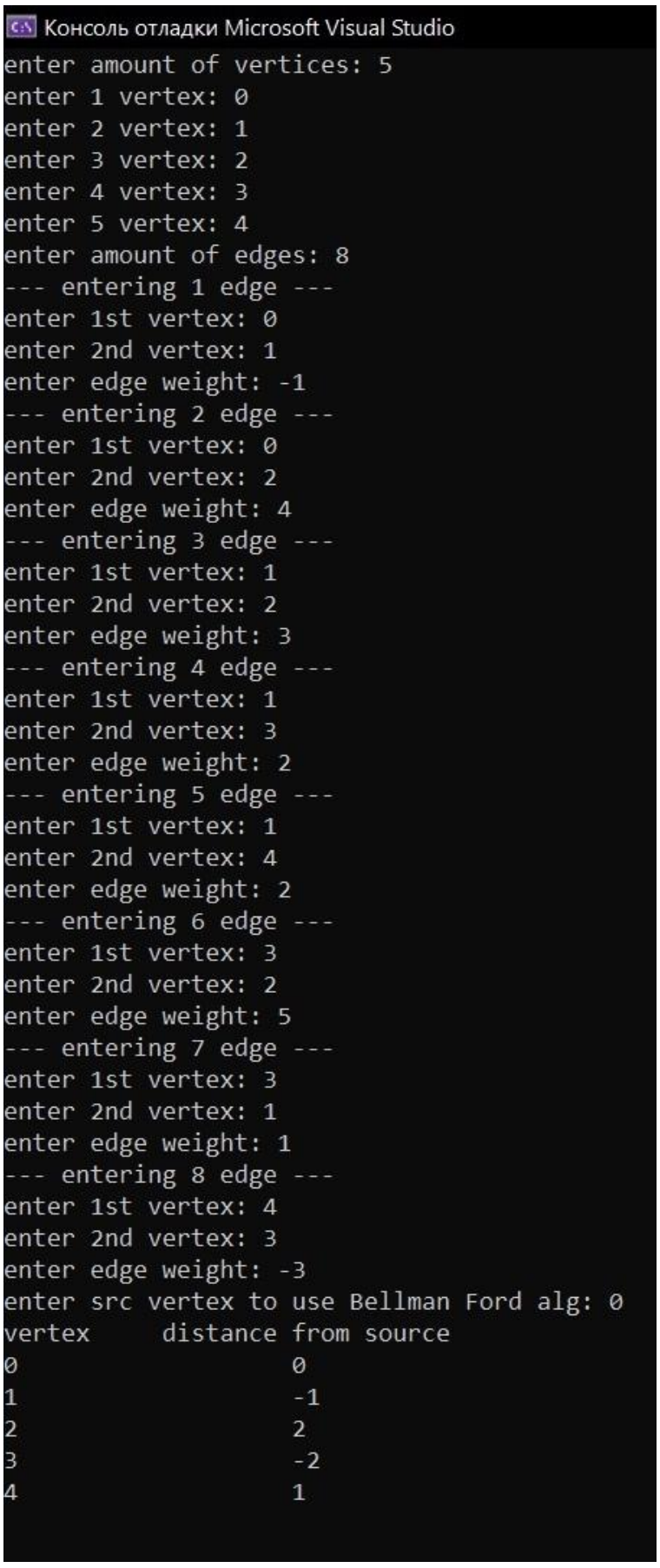

Рисунок 1 - тестирование программы

Таким образом, по результатам тестирования видно, что программа работает корректно.

Вывод

В результате выполнения данной работы мной были освоены навыки работы с графами. Мной был изучен
метод Беллмана-Форда для поиска кратчайшего расстояния между вершинами в взвешенном графе. Он был применён для решения заданий практической работы. Таким образом, мной был получен практический опыт использования метода.