Московский авиационный институт
(национальный исследовательский университет)

**Факультет информационных технологий и прикладной математики**

**Кафедра вычислительной математики и программирования**

# Лабораторная работа №9 по курсу «Дискретный анализ»

| | |
|---|---|
| Студент: | М. А. Бронников |
| Преподаватель: | А. А. Кухтичев |
| Группа: | М8О-207Б |
| Дата: | |
| Оценка: | |
| Подпись: | |

**Москва, 2019**

# Лабораторная работа №9

**Задача:**

Разработать программу на языке C или C++, реализующую указанный алгоритм. Формат входных и выходных данных описан в варианте задания.

**Вариант задания:** Поиск кратчайшего пути между парой вершин алгоритмом Беллмана-Форда.

**Входные данные:** В первой строке заданы $1 \leq n \leq 105, 1 \leq m \leq 3 * 105, 1 \leq start \leq n, 1 \leq finish \leq n$. В следующих m строках записаны ребра. Каждая строка содержит три числа - номера вершин, соединенных ребром, и вес данного ребра. Вес ребра - целое число от -109 до 109.

**Выходные данные:** Необходимо вывести одно число - длину кратчайшего пути между указанными вершинами. Если пути между указанными вершинами не существует, следует вывести строку "No solution"(без кавычек).

# 1 Описание

Как сказано в [1]: «Пусть дан ориентированный взвешенный граф $G$ с $n$ вершинами и $m$ рёбрами, и указана некоторая вершина $v$. Требуется найти длины кратчайших путей от вершины $v$ до всех остальных вершин.».

« Мы считаем, что граф не содержит цикла отрицательного веса. Случай наличия отрицательного цикла будет рассмотрен ниже в отдельном разделе.

Заведём массив расстояний d[0 ... n-1], который после отработки алгоритма будет содержать ответ на задачу. В начале работы мы заполняем его следующим образом: d[v] = 0, а все остальные элементы d[] равны бесконечности $\infty$.

Сам алгоритм Форда-Беллмана представляет из себя несколько фаз. На каждой фазе просматриваются все рёбра графа, и алгоритм пытается произвести релаксацию (relax, ослабление) вдоль каждого ребра (a,b) стоимости c. Релаксация вдоль ребра — это попытка улучшить значение d[b] значением d[a] + c. Фактически это значит, что мы пытаемся улучшить ответ для вершины b, пользуясь ребром (a,b) и текущим ответом для вершины a.

Утверждается, что достаточно n-1 фазы алгоритма, чтобы корректно посчитать длины всех кратчайших путей в графе (повторимся, мы считаем, что циклы отрицательного веса отсутствуют). Для недостижимых вершин расстояние d[] останется равным бесконечности $\infty$» [1]

Пользуясь описанным алгоритмом мы найдем кратчайшие пути до всех вершин, после чего выведем ответ для интересующей нас вершины.

# 2 Исходный код

```
#include <iostream>
#include <vector>

long long int INF = 10000000000000001;

struct edge {
    int a;
    int b;
    int cost;
};

int main() {
    std::ios::sync_with_stdio(false);
    int N = 0, M = 0, start, finish;
    std::cin >> N >> M >> start >> finish;
    std::vector<edge> e(M);
    std::vector<long long int> d(N, INF);
    d[start - 1] = 0;
    int a, b, cost;

    for(int i = 0; i < M; ++i) {
        std::cin >> a >> b >> cost;
        e[i].a = a - 1;
        e[i].b = b - 1;
        e[i].cost = cost;
    }
    for(int j = 0; j < N - 1; ++j) {
        bool any = false;
        for(int i = 0; i < e.size(); ++i) {
            if(d[e[i].a] < INF) {
                if(d[e[i].b] > d[e[i].a] + e[i].cost) {
                    d[e[i].b] = d[e[i].a] + e[i].cost;
                    any = true;
                }
            }
        }
        if(!any) break;
    }

    if(d[finish - 1] == INF) {
        std::cout << "No solution\n";
    } else {
        std::cout << d[finish - 1] << '\n';
    }
    return 0;
}
```

# 3 Консоль

```
(base) max@max-X550CC:~/DA/lab9$ ls
da_lab9.pdf  main.cpp
(base) max@max-X550CC:~/DA/lab9$ g++ -std=c++17 -o run -pedantic  main.cpp
(base) max@max-X550CC:~/DA/lab9$ ./run
5 6 1 5
1 2 2
1 3 0
3 2 -1
2 4 1
3 4 4
4 5 5
5
(base) max@max-X550CC:~/DA/lab9$ rm run
(base) max@max-X550CC:~/DA/lab9$ exit
```

# 4 Выводы

Благодаря девятой лабораторной работе по курсу «Дискретный анализ», я, наконец, познакомился с основными вариантами поиска по графам.

Эта лабораторная работа оказалась одрной из самых простых за курс. Наибольшие неудобсва мне в доставило то, что я не знал о возможности отключения синхронизации потоков ввода/вывода из-за чего я получал *TIME LIMIT* на 5 тесте лабораторной. Тем не менее задание все равно показалось мне простым, ведь алгоритм, реализованный мной реализуется буквально в 10 строк

Я рад, что теперь в моем арсенале появился такой элегантный и мощный метод, поиск по графам, ведь без знаний о нем не может обойтись любой хороший программист.

# Список литературы

[1] *Алгоритм Форда-Беллмана - Maximal*
URL: http://www.e-maxx-ru.1gb.ru/algo/ford$_bellman$( : 16.10.2019).