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: В. В. Косогоров
Преподаватель: А. А. Кухтичев
Группа: М8О-206Б

Дата:
Оценка:
Подпись:

Москва, 2020

Лабораторная работа №9

Задача: Задан взвешенный ориентированный граф, состоящий из n вершин и m рёбер. Вершины пронумерованы целыми числами от 1 до n. Необходимо найти длину кратчайшего пути из вершины с номером start в вершину с номером finish при помощи алгоритма Беллмана-Форда. Длина пути равна сумме весов рёбер на этом пути. Обратите внимание, что в данном варианте веса ребер могут быть отрицательными, поскольку алгоритм умеет с ними работать. Граф не содержит петель, кратных рёбер и циклов отрицательного веса.

1 Описание

В отличие от алгоритма Дейкстры, этот алгоритм применим также и к графам, содержащим рёбра отрицательного веса. Впрочем, если граф содержит отрицательный цикл, то, понятно, кратчайшего пути до некоторых вершин может не существовать (по причине того, что вес кратчайшего пути должен быть равен минус бесконечности).[1].

Для алгоритма Беллмана-Форда, в отличие от многих других графовых алгоритмов, более удобно представлять граф в виде одного списка всех рёбер (а не n списков рёбер из каждой вершины).

2 Исходный код

Для хранения информации о каждом ребре графа создадим структуру Edge, в которой будет находиться инофрмация о вершинах, инциндентных данному ребру, и его весе. В векторе будем хранить минимальные расстояния от вершины с номером start до каждой вершины графа. Произведём n - 1 релаксаций, каждый раз уменьшая расстояния в векторе, если в ходе алгоритма было найдено расстояние, меньшее предыдущего.
В конце работы алгоритма в векторе будут храниться наименьшие расстояния от вершины start до каждой вершины, и нам остаётся выбрать лишь вершину с номером finish.

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

const long INF = std::numeric_limits<long>::max();

struct Edge {
    unsigned long from, to;
    long weight;

    friend std::istream& operator>>(std::istream& is, Edge& e);
};

std::istream& operator>>(std::istream& is, Edge& e) {
    unsigned long from, to;
    is >> from;
    is >> to;
    e.from = from - 1;
    e.to = to - 1;
    is >> e.weight;
    return is;
}

long BellmanFord(const std::vector<Edge>& edges, const unsigned long n, const unsigned long
                 const unsigned long start, const unsigned long finish) {
    std::vector<long> minDistFromStart(n, INF);
    minDistFromStart[start] = 0;
    for (unsigned long relaxN = 1; relaxN <= n - 1; ++relaxN) {
        bool relaxation = false;
```

```
        for (auto edge : edges) {
            if (minDistFromStart[edge.from] < INF) {
                if (minDistFromStart[edge.from] + edge.weight < minDistFromStart[edge.to]) {
                    minDistFromStart[edge.to] = minDistFromStart[edge.from] + edge.weight;
                    relaxation = true;
                }
            }
        }
        if (relaxation == false) {
            break;
        }
    }

    return minDistFromStart[finish];
}

int main() {
    std::ios_base::sync_with_stdio(false);

    unsigned long n, m, start, finish;
    std::cin >> n >> m >> start >> finish;
    --start;
    --finish;

    std::vector<Edge> edges(m);
    for (auto i = 0; i < m; ++i) {
        std::cin >> edges[i];
    }

    long ans = BellmanFord(edges, n, m, start, finish);
    if (ans == INF) {
        std::cout << "No solution" << std::endl;
    } else {
        std::cout << ans << std::endl;
    }

    return 0;
}
```

3 Консоль

```
 .../da_labs/lab_09  ./a.out
5 6 1 5
1 2 2
1 3 0
3 2 -1
2 4 1
3 4 4
4 5 5
5


 .../da_labs/lab_09  ./a.out
5 6 1 5
1 2 34
1 3 54
3 2 424
2 3 443
3 1 1323
4 2 123
No solution
```

4 Тест производительности

Релаксация производится n - 1 раз, где n - количество вершин графа. В этом цикле релаксация производится для каждого ребра. Количество рёбер в полном графе равно $\frac{n(n-1)}{2}$, то есть $O(n^2)$.
Получается, алгоритм Беллмана-Форда работает за $O(n^3)$.

5 Выводы

Выполнив восьмую лабораторную работу по курсу «Дискретный анализ», я реализовал алгоритм Беллмана-Форда поиска кратчайшего пути в графе. Этот алгоритм использует динамическое программирование, для каждого ребра проводя релаксации, чтобы найти оптимальное решение для задачи.

Список литературы

[1] *Bellman-Ford*
URL: `https://en.wikipedia.org/wiki/BellmanFord_algorithm` (дата обращения: 10.06.2020)