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: И. Т. Батяновский
Преподаватель: А. А. Кухтичев
Группа: М8О-307Б

Дата:
Оценка:
Подпись:

Москва, 2020

Лабораторная работа №9

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

Задача:

Разработать программу на языке C или C++, реализующую указанный алгоритм согласно заданию:

Задан взвешенный ориентированный граф, состоящий из **n** вершин и **m** ребер. 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 ребер 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 ребер и циклов отрицательного веса.

Формат входных данных

В первой строке заданы $1 \leq \mathbf{n} \leq 10^5$, $1 \leq \mathbf{m} \leq 3 * 10^5$, $1 \leq \mathbf{start} \leq \mathbf{n}$ и $1 \leq$**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 -10^9 до 10^9.

Формат результата

Необходимо вывести одно число – длину кратчайшего пути между указанными вершинами. Если пути между указанными вершинами не существует, следует вывести строку **«No solution»** (без кавычек).

Примеры

Входные данные:	Результат работы:
5 6 1 5 1 2 2 1 3 0 3 2 -1 2 4 1 3 4 4 4 5 5	5

1 Описание алгоритма

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 $d[0 \dots n-1]$, 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 ∞.

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 ∞.

2 Реализация

Для алгоритма Форда-Беллмана, в отличие от многих других графовых алгоритмов, более удобно представлять граф в виде одного списка всех рёбер (а не n списков рёбер — рёбер из каждой вершины). В приведённой реализации заводится структура данных edge для ребра. Входными данными для алгоритма являются числа n, m, список e рёбер, и номер стартовой вершины v. Все номера вершин нумеруются с 0 по $n - 1$.

```
#include<bits/stdc++.h>

typedef long long int li;

using namespace std;

const li INF = LONG_MAX;

struct edge
{
  li u, v, weight;
};

int main()
{
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  li n, m, s, f;
  cin >> n >> m >> s >> f;
  vector<edge> e(m);

  for (int i = 0; i < m; ++i)
    cin >> e[i].u >> e[i].v >> e[i].weight;

  bool signal;
  vector<li> d(n + 1, INF);
  d[s] = 0;

  for (int i = 0; i < n; ++i)
  {
    signal = false;
    for (int j = 0; j < m; ++j)
    {
      if (d[e[j].u] < INF)
      {
        if (d[e[j].v] > d[e[j].u] + e[j].weight)
        {
          d[e[j].v] = d[e[j].u] + e[j].weight;
          signal = true;
```

```
      }
    }
  }
  if (signal == false) break;
 }

 if (d[f] == INF)
 {
  cout << "No solution\n";
 } else {
  cout << d[f] << '\n';
 }
 return 0;
}
```

3 Выводы

В данной лабораторной работе я изучил и реализовал полезный алгоритм поиска кратчайшего пути в графе. Необходимость понимания данного алгоритма уже была доказана мне в задачах по спортивному порграммированию.

Список литературы

[1] *Основные идеи реализации*
URL: `https://e-maxx.ru/algo/ford_bellman`

[2] *Первое знакомство с алгоритмом*
URL: `https://ru.wikipedia.org/wiki/Алгоритм_Беллмана_-_Форда`