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: П. А. Харьков
Преподаватель: А. А. Кухтичев
Группа: М8О-206Б-19
Дата:
Оценка:
Подпись:

Москва, 2021

Лабораторная работа №9

Задача: 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 $1 \le n \le 10^5$, $1 \le m \le 3 * 10^5$, $1 \le start \le n$ и $1 \le finish \le n$. 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 результатов: Необходимо вывести одно число – длину кратчайшего пути между указанными вершинами. Если пути между указанными вершинами не существует, следует вывести строку «No solution».

1 Описание

Требуется написать реализацию алгоритма поиска кратчайшего пути с помощью алгоритма Форда-Беллмана. Этот алгоритм позволяет найти кратчайший путь за $O(n * m)$, где n - количество узлов, а m - количество ребер. Этот алгоритм особенно полезен, если в графе может содержаться отрицательный цикл - хотя задача поиска минимального пути уже не имеет смысла. Для того, чтобы определить, что в графе есть отрицательный цикл необходимо немного модифицировать алгоритм.

2 Исходный код

Сначала я считываю количество вершин, рёбер, начальную вершину и конечную вершину. Затем сохраняю все рёбра и их цены в вектор *edges* и завожу вектор *distances*, в котором буду хранить минимальные цены путей из начальной вершины в i-тую вершину.

Затем на каждой фазе я прохожу по всем ребрам и пытаюсь улучшить результат для конечной точки. Если за всю фазу ни один результат не был улучшен, значит мы получили наилучший результат и можно завершать работу.

В конце я проверяю, если значение в векторе *distances* для конечной вершины не изменилось, значит, вершина недостижима из начальной вершины. Если же изменилось, то мы нашли минимальный путь.

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <cmath>

using namespace std;
using TLl = long long;

const TLl MAX_VAL = pow(10, 15);

struct TEdge {
  TLl From, To, Price;
};

int main(){
    ios::sync_with_stdio(false);
    cin.tie(NULL);
    cout.tie(NULL);

    int n, m, start, finish;
  cin >> n >> m >> start >> finish;
  start--;
    finish--;

  vector<TEdge> edges(m);
  TLl from, to, price;
  for (int i = 0; i < m; i++){
    cin >> from >> to >> price;
    from--;
        to--;
    edges[i].From = from;
        edges[i].To = to;
        edges[i].Price = price;
  }
```

```
    vector<TLl> distances(n, MAX_VAL);
  distances[start] = 0;

  for (int i = 1; i < n; i++){
    bool changed = false;
    for (const TEdge& e : edges){
      if(distances[e.To] > distances[e.From] + e.Price){
        distances[e.To] = distances[e.From] + e.Price;
        changed = true;
      }
    }

    if(changed == false){
      break;
    }
  }

  if (distances[finish] == MAX_VAL){
    cout << "No solution" << '\n';
    }else{
    cout << distances[finish] << '\n';
    }

    return 0;
}
```

3 Консоль

```
p.kharkov$ make
g++ -std=c++14 -pedantic -Wall -Wextra -O2 src/laba9.cpp -o laba9
p.kharkov$ cat tests/main
5 6 1 5
1 2 2
1 3 0
3 2 -1
2 4 1
3 4 4
4 5 5
p.kharkov$ ./laba9 <tests/main
5
```

4 Тест производительности

Тест производительности представляет из себя следующее: поиск кратчайшего пути полным перебором сравнивается с поиском кратчайшего пути с помощью алгоритма Беллмана-Форда. Первый тест состоит из 10 вершин и 100 ребер, второй - из 10 вершин и 120 ребер, а третий - из 10^5 вершин и 10^7 ребер:

```
p.kharkov$ ./benchmark <tests/01.t
Naive solution time: 122ms
FordBellman solution time: 0ms
p.kharkov$ ./benchmark <tests/02.t
Naive solution time: 769ms
FordBellman solution time: 0ms
p.kharkov$ ./benchmark <tests/03.t
FordBellman solution time: 671ms
```

Как видно, что алгоритм Беллмана-Форда значительно быстрее работает, чем наивный. В данном случае сложность наивного алгоритма $O(n^m)$, в то время, как сложность алгоритма Беллмана-Форда - $O(n * m)$.

5 Выводы

Выполнив девятую лабораторную работу по курсу «Дискретный анализ», я научился реализовывать алгоритм поиска кратчайшего пути в графе с помощью алгоритма Форда-Беллмана. Одним из преимуществом этого алгоритма, является то, что он может искать путь даже в графах с циклами отрицательного веса. Поиск кратчайшего пути в графе используется в различных задачах - как мне кажется, самой популярной из них это нахождение маршрута в обычных картах.

Список литературы

[1] Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн. *Алгоритмы: построение и анализ, 2-е издание.* — Издательский дом «Вильямс», 2007. Перевод с английского: И. В. Красиков, Н. А. Орехова, В. Н. Романов. — 1296 с. (ISBN 5-8459-0857-4 (рус.))

[2] *Алгоритм Форда-Беллмана.*
URL: `https://neerc.ifmo.ru/wiki/index.php?title=Алгоритм_Форда-Беллмана` (дата обращения: 03.05.2021).