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: Е. С. Пищик
Преподаватель: А. А. Кухтичев
Группа: М8О-206Б

Дата:
Оценка:
Подпись:

Москва, 2021

Лабораторная работа №9

Задача: Вариант №6.

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 длины кратчайших путей между всеми парами вершин при помощи алгоритма Джонсона. 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 и кратных ребер.

Формат входных данных: В первой строке заданы 1 <= n <= 2000, 1 <= m <= 4000. 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: Если граф содержит цикл отрицательного веса, следует вывести строку "Negative cycle"(без кавычек). В противном случае следует выести матрицу из n строк и n столбцов, где j-е число в i-й строке равно длине кратчайшего пути из вершины i в вершину j. Если такого пути не существует, на соответствующей позиции должно стоять слово "inf"(без кавычек). Элементы матрицы в одной строке разделяются пробелом.

1 Описание

Для написания алгоритма Джонсона нам понадобится написать еще два алгоритма, алгоритм Беллмана-Форда и Дейкстры. Сам алгоритм таков, если алгоритм Беллмана-Форда возвращает ложь, то у нас отрицательный цикл, иначе используем алгоритм Дейкстры, если нет отрицательных дуг, иначе формируем новый орграф с такими же кратчайшими путями, но без отрицательных дуг и снова запускаем алгорит Дейкстры.

2 Исходный код

johnson.cpp	
bool operator<(TEdge const& p1, TEdge const& p2)	Оператор сравнения двух объектов ребер графа.
void Deikstra(TMatrix const& gr, size_t const& node, TMatrix& dist, size_t const& n)	Алгоритм Дейкстры.
bool BellmanFord(TGraph const& gr, size_t const& node, TMatrix& dist)	Алгоритм Беллмана-Форда.
bool Johnson(TGraph const& gr, TMatrix& dist)	Алгоритм Джонсона.
main.cpp	
int main()	Основная функция, собирающая все части программы в единое целое.

```
structures.hpp
#pragma once
#include <iostream>
#include <vector>
#include <queue>

typedef std::vector<std::vector<int64_t>> TMatrix;

struct TEdge
{
    size_t from;
    size_t to;
    int64_t weigth;
};

struct TGraph
{
    size_t v, e;
    std::vector<TEdge> edges;
    TGraph() {}
    TGraph(size_t n, size_t m) : v(n), e(m) {}
};
```

```
johnson.hpp
#pragma once
#include "structures.hpp"
bool operator<(TEdge const&, TEdge const&);
void Deikstra(TMatrix const&, size_t const&, TMatrix&, size_t const&);
bool BellmanFord(TGraph const&, size_t const&, TMatrix&);
bool Johnson(TGraph const&, TMatrix&);
```

3 Консоль

```
pe4eniks@pe4eniks-HP-Laptop-14-dk0xxx:~/solution$ cat test.txt
5 4
1 2 -1
2 3 2
1 4 -5
3 1 1
pe4eniks@pe4eniks-HP-Laptop-14-dk0xxx:~/solution$ ./solution<test.txt
0 -1 1 -5 inf
3 0 2 -2 inf
1 0 0 -4 inf
inf inf inf 0 inf
inf inf inf inf 0
```

4 Тест производительности

Тест состоит из ввода графа и запуска алгоритма Джонсона 50000, 100000 и 200000 раз.

```
pe4eniks@pe4eniks-HP-Laptop-14-dk0xxx:~/solution$ ./benchmark
Enter graph:
5 4
1 2 -1
2 3 2
1 4 -5
3 1 1
Enter test count:
50000
Time for Johnson on 50000 tests: 3 seconds

pe4eniks@pe4eniks-HP-Laptop-14-dk0xxx:~/solution$ ./benchmark
Enter graph:
5 4
1 2 -1
2 3 2
1 4 -5
3 1 1
Enter test count:
100000
Time for Johnson on 100000 tests: 7 seconds

pe4eniks@pe4eniks-HP-Laptop-14-dk0xxx:~/solution$ ./benchmark
Enter graph:
5 4
1 2 -1
2 3 2
1 4 -5
3 1 1
Enter test count:
200000
Time for Johnson on 200000 tests: 14 seconds
```

Алгоритм работает за O(n*m+n^2*lnn), в нашем случае n =m -> O(n^2*lnn), что лучше чем алгоритм Флойда-Уоршела, имеющего сложность O(n^3) и явно лучше наивного перебора за O(n^4).

5 Выводы

Выполнив девятую лабораторную работу по курсу «Дискретный анализ», я познакомился с алгоритмами на графах, такими как алгоритм Дейкстры, Беллмана-Форда и Джонсона, узнал в каких случаях нужно использовать алгоритм Джонсона и почему нельзя использовать Дейкстру(при наличии отрицательных ребер), для чего в алгоритме Джонсона нужен алгоритм Беллмана-Форда(для нахождения отрицательного цикла).

Список литературы

[1] Джонсон Habr

[2] Джонсон Wiki