Московский авиационный институт
(национальный исследовательский университет)

Факультет компьютерных наук и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №9 по курсу «Дискретный анализ»

Студент:	О. А. Мезенин
Преподаватель:	А. А. Кухтичев
Группа:	М8О-306Б-21
Дата:	
Оценка:	
Подпись:	

Москва, 2023

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 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.

1 Описание

Алгоритм Джонсона предназначен для вычисления кратчайших путей между всеми парами вершин [1]. Он эффективен для разреженных графов и умеет работать с отрицательными рёбрами. Алгоритм Джонсона в ходе выполнения использует алгоритмы Беллмана-Форда и Дейкстры.
Опишем сам алгоритм:

1. Создать вершину s и присоединить её ко всем остальным вершинам дугами с весом 0.

2. Запустить алгоритм Беллмана-Форда для графа вместе с вершиной s из вершины s. Если в процессе выполнения алгоритма будет найден отрицательный цикл, то алгоритм Джонсона закончен.

3. Изменить весовую функцию: $\omega'_{i,j} = \omega_{i,j} + h(i) - h(j)$, где $h(x)$ – минимальные расстояния от вершины s до остальных вершин, найденные алгоритмом Беллмана-Форда. Таким образом, мы избавимся от отрицательных весов.

4. Убрать вершины s и запустить алгоритм Дейкстры для всех вершин.

5. Ответ для каждого ребра будет следующий: $\delta_{i,j} = \delta'_{i,j} - h(i) + h(j)$, где $\delta'_{x,y}$ – значения, вычисленные алгоритмом Дейкстры.

Пусть V – количество вершин, E – количество рёбер в графе. Сложность алгоритма составит $O(V^2 \log V + VE \log V)$. При $E \sim V$ – $O(V^2 \log V)$.

2 Исходный код

Функция `main` считывает входные данные, строит граф, вызывает функцию `Johnson` и выводит ответ.

```
int main() {
    std::ios::sync_with_stdio(false);
    std::cin.tie(0);

    int n, m;
    std::cin >> n >> m;
    TGraph g;
    for (int i = 0; i < m; ++i) {
        int u, v, w;
        std::cin >> u >> v >> w;
        if (g.find(u-1) == g.end()) {
            g[u-1] = std::vector<TNode>();
        }
        g[u-1].emplace_back(v-1, w);
    }

    try {
        TGraph answer = Johnson(g, n);
        std::vector<std::vector<long>> matrix(n, std::vector<long>(n, LONG_MAX));
        for (int i = 0; i < n; ++i) {
            for (TNode &p: answer[i]) {
                matrix[i][p.v] = p.w;
            }
        }
        for (int i = 0; i < n; ++i) {
            for (int j = 0; j < n; ++j) {
                if (matrix[i][j] == LONG_MAX)
                    std::cout << "inf ";
                else
                    std::cout << matrix[i][j] << " ";
            }
            std::cout << '\n';
        }
    } catch (TNegativeCycle &e) {
        std::cout << "Negative cycle\n";
    }
    return 0;
}
```

Реализация алгоритма Джонсона:

```
TGraph Johnson(const TGraph &g, int n) {
    TGraph answer;
    TGraph gd = g;
    std::vector<long> h;
```

```
    gd[n] = std::vector<TNode>(n);
    for (int i = 0; i < n; ++i) {
        gd[n][i].v = i;
        gd[n][i].w = 0;
    }

    bool negativeCycleNotExists = BellmanFord(gd, n+1, n, h);
    if (!negativeCycleNotExists)
        throw TNegativeCycle();

    for (int i = 0; i < n+1; ++i) {
        if (gd.find(i) == gd.end())
            continue;
        for (TNode &node: gd[i]) {
            node.w += h[i] - h[node.v];
        }
    }
    for (int i = 0; i < n; ++i) {
        std::vector<long> delta = Dijkstra(gd, n, i);
        answer[i] = std::vector<TNode>();
        for (int j = 0; j < n; ++j) {
            answer[i].emplace_back(j, delta[j] == LONG_MAX ? delta[j] : delta[j] + h[j] - h[i]);
        }
    }
    return answer;
}
```

dijkstra.cpp	
std::vector<long> Dijkstra(TGraph &g, int n, int start)	Алгоритм Дейкстры
bellman_ford.cpp	
bool BellmanFord(TGraph &g, int n, int start, std::vector<long> &h)	Алгоритм Беллмана-Форда

3 Консоль

```
aprold@SAI:~/Documents/GitHub/MAI-DA/lab9$ make lab9
g++ -std=c++2a -pedantic -Wall -Wextra -Werror main.cpp johnson.cpp bellman_ford.cpp
dijkstra.cpp -o lab9.out
aprold@SAI:~/Documents/GitHub/MAI-DA/lab9$ ./lab9.out
5 4
1 2 -1
2 3 2
1 4 -5
3 1 1
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

Тесты производительности представляют из себя следующее: алгоритм Джонсона будет сравниваться с алгоритмом Флойда-Уоршелла. Будет два теста:

1. $n = 1000, m = 1000$
2. $n = 1000, m = 999000$

Все веса равны 1.

```
aprold@SAI:~/Documents/GitHub/MAI-DA/lab9$ make benchmark
g++ -std=c++2a -pedantic -Wall -Wextra -Werror benchmark.cpp johnson.cpp bellman_ford
dijkstra.cpp -o benchmark.out
aprold@SAI:~/Documents/GitHub/MAI-DA/lab9$ ./benchmark.out <tests/test1
Johnson time: 327175us
FloydWarshall time: 4336279us
aprold@SAI:~/Documents/GitHub/MAI-DA/lab9$ ./benchmark.out <tests/test2
Johnson time: 37005784us
FloydWarshall time: 20131167us
```

Первый тест оказался быстрее с алгоритмом Джонсона, второй – с алгоритмом Флойда-Уоршелла. Временная сложность алгоритма Флойда-Уоршелла составляет $O(n^3)$, а алгоритма Джонсона – $O(n^2 \log n + nm \log n)$, что при $n \sim m$ (первый тест) есть $O(n^2 \log n)$, но при $n^2 \sim m$ (второй тест) есть $O(n^3 \log n)$.

5 Выводы

Выполнив девятую лабораторную работу по курсу «Дискретный анализ», узнал про алгоритм Джонсона.
Алгоритм Джонсона предназначен для вычисления кратчайших путей между всеми парами вершин. Он эффективен для разреженных графов и умеет работать с отрицательными рёбрами. Тесты также показали, что для полных графов лучше использовать другой алгоритм, например, алгоритм Флойда-Уоршелла.

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 Джонсона*
URL: `https://neerc.ifmo.ru/wiki/index.php?title=Алгоритм_Джонсона` (дата обращения: 05.04.2023)