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. П. Будникова
Преподаватель:	А. А. Кухтичев
Группа:	М8О-207Б-19
Дата:	
Оценка:	
Подпись:	

Москва, 2021

Лабораторная работа №6

Задача: Разработать программу на языке C или C++, реализующую указанный алгоритм согласно заданию.
Вариант алгоритма: 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 $\mathbf{1 \leqslant n \leqslant 2000, 1 \leqslant m \leqslant 4000}$. В следующих **m** строках записаны ребра. Каждая строка содержит три числа – номера вершин, соединенных ребром, и вес данного ребра. Вес ребра – целое число **от** $\mathbf{-10^9}$ **до** $\mathbf{10^9}$.
Формат результата: Если граф содержит цикл отрицательного веса, следует вывести строку **"Negative cycle"** (без кавычек). В противном случае следует вывести 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

Как говорится в [1]: «Алгоритм Джонсона — позволяет найти кратчайшие пути между всеми парами вершин взвешенного ориентированного графа. Данный алгоритм работает, если в графе содержатся рёбра с положительным или отрицательным весом, но отсутствуют циклы с отрицательным весом. » Данный алгоритм использует в себе два алгоритма - алгоритм Беллмана Форда и алгоритм Дейкстры. По [2]: «Алгоритм Беллмана — Форда — алгоритм поиска кратчайшего пути во взвешенном графе. » и [3]: « Алгоритм Дейкстры (англ. Dijkstra's algorithm) - алгоритм на графах, изобретённый нидерландским учёным Эдсгером Дейкстрой в 1959 году. Находит кратчайшие пути от одной из вершин графа до всех остальных. Алгоритм работает только для графов без рёбер отрицательного веса. »
Алгоритм Беллмана-Форда решает задачу поиска минимального расстояния от одной вершины до всех остальных за O$(n * m)$, где n - количество вершин графа, m -множество ребер графа. То сложность решения поставленной задачи была бы O$(n^2 * m)$, поэтому алгоритм Джонсона сначала делает граф без отрицательных дуг из поданного графа, а потом с помощью алгоритма Дейкстры, сложность которого O(n^2), находит кратчайшие пути от каждой вершины до всех остальных. Сначала создается дополнительная вершина, с ребрами, вес которых 0, в каждую вершину заданного графа. Далее с помощью алгоритма Беллмана-Форда находятся все кратчайшие пути из новой вершины во все остальные. Высчитываются новые веся для всех ребер по формуле: $newLen(A, B) = len(A, B) + bf(A) - bf(B)$, где $bf(X)$ - кратчайший путь от новой добавленной вершины до вершины X, посчитанное с помощью алгоритма Беллмана-Форда . Таким образом граф больше не содержит отрицательных дуг. Также при работе алгоритма Беллмана-Форда граф проверяется на наличие отрицательных циклов, если такие содержаться, то соответствующее сообщение выводится на экран и программа завершает свою работу. Новая вершина никак не влияет на наличие отрицательных вершин в начальном графе, так как в нее не идет ни одна дуга. После того, как все веса дуг были пересчитаны, с помощью алгоритма Дейкстры находятся кратчайшие пути для каждой вершины во все остальные.

2 Исходный код

Функции	
bool BellmanFord (std::vector<long long> &bf, const Vertex &edges, const int &n)	Функция, выполняющая алгоритм Беллмана-Форда
std::vector<std::pair<long long,bool» Dijkstra(int curV, const VertexL &vertexes)	Функция, выполняющая алгоритм Дейкстры

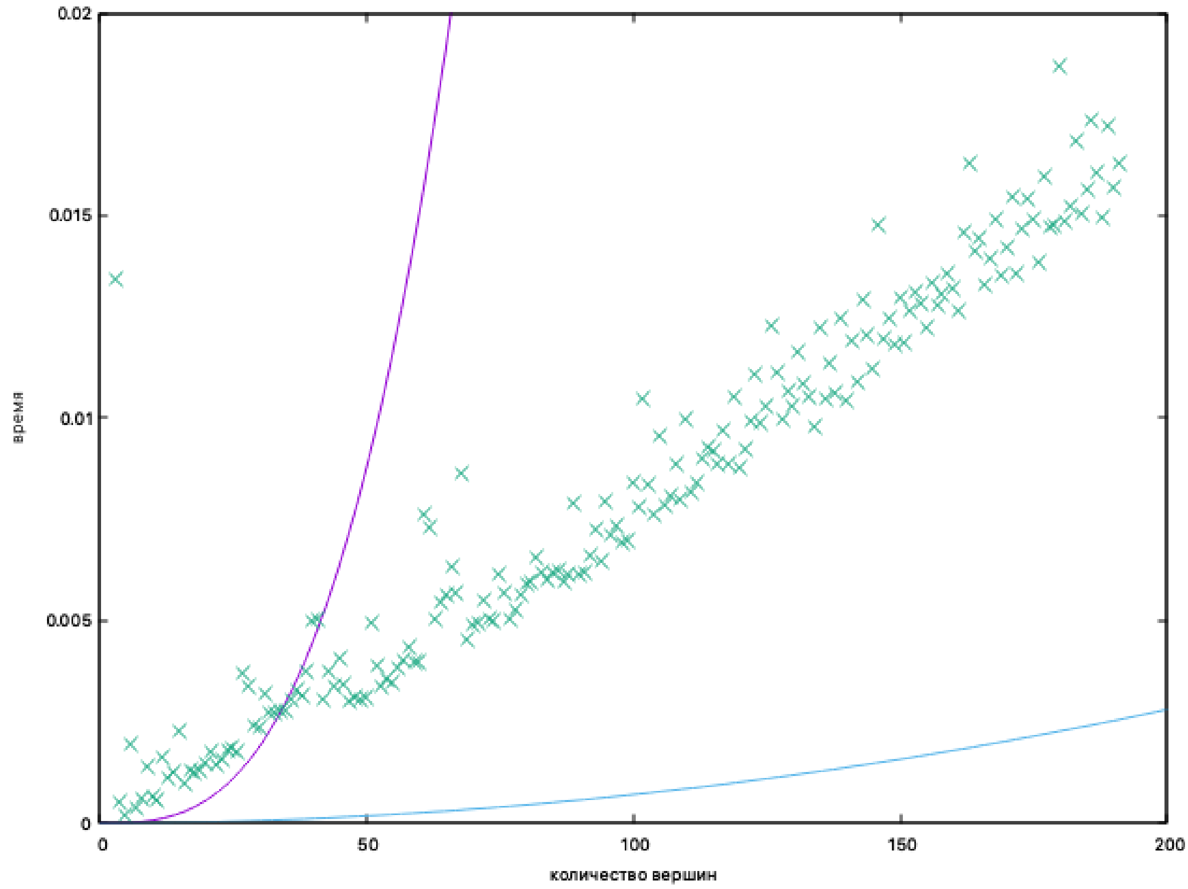
```cpp
int main() {
    std::ios::sync_with_stdio(false);
    int n = 0, m = 0;
    int num1, num2, w;
    std::cin >> n >> m;
    if (m == 0 && n == 0) return 0;
    if (m == 0) { Print(n); return 0;}
    Vertex edges(n);
    for (int i = 0; i < m; ++i) {
        std::cin >> num1 >> num2 >> w;
        edges[num1 - 1].push_back({num2 - 1, w});
    }
    //BellmanFord
    std::vector<long long> bf(n, 0);
    if (!BellmanFord(bf, edges, n)) {
        std::cout << "Negative cycle" << std::endl;
        return 0;
    }
    VertexL vertexes(n);
    for(int i = 0; i < edges.size(); ++i) {
        for (std::pair<int, int> e : edges[i]) vertexes[i].push_back({e.first, e.second
             + bf[i] - bf[e.first]});
    }
    //Dijkstra
    std::vector<std::pair<long long, bool>> d;
    for (int i = 0; i < n; ++i) {
        d = Dijkstra(i, vertexes);
        for (int j = 0; j < d.size(); ++j) {
            if (d[j].first == __LONG_LONG_MAX__) {
                std::cout << "inf";
            } else {
                if (i == j) std::cout << 0;
                else std::cout << d[j].first + bf[j] - bf[i];
            }
            if (j < d.size() - 1) std::cout << " ";
        }
        std::cout << std::endl;
    }
    return 0;}
```

3 Консоль

```
Lera:l9 valeriabudnikova$ cat test.txt
5 4
1 2 -1
2 3 2
1 4 -5
3 1 1
Lera:l9 valeriabudnikova$ ./lab9 <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

Сложность алгоритма Джонсона для данной задачи: O($n^2 * m + n(n^2 + m)$), где n - количество вершин, а m - количество ребер, но так как в программе используется структура set, для которой вставка происходят за O($log(n)$), а поиск наименьшего элемента за O(1), то сложность O($n^2 * m + n^2 * (log(n) + m)$), тк m < n(n - 1), ≈O($n^2 log(n)$). Постоим график зависимости времени от входных данных. В входных данных заданы такие графы, что количество ребер составляет примерно треть от количества вершин.
На графике также в качестве сравнения приведены квадратичная и кубическая функция.



5 Выводы

Выполнив девятую лабораторную работу по курсу «Дискретный анализ», я научилась реализовывать алгоритм Джонсона, включая Беллмана-Форда и Дейкстры для нахождения минимальных путей в графе. Также я попрактиковалась в использовании различных структур данных в c++ и их методах. В данной лабораторной работе я также повторила перегрузки операторов, так как для тестирования программы необходимо было выводить различные структуры на экран, что сильно бы повысило неразборчивость кода, если не использовать перегрузки операторов вывода.

Список литературы

[1] *Алгоритм Джонсона — Википедия.*
URL: `https://ru.wikipedia.org/wiki/Алгоритм_Джонсона` (дата обращения: 18.06.2020).

[2] *Алгоритм Беллмана-Форда — Википедия.*
URL: `https://ru.wikipedia.org/wiki/Алгоритм_Беллмана_-_Форда` (дата обращения: 18.06.2020).

[3] *Алгоритм Дейкстры — Википедия.*
URL: `https://ru.wikipedia.org/wiki/Алгоритм_Дейкстры` (дата обращения: 18.06.2020).