Санкт-Петербургский политехнический университет Петра Великого

Кафедра прикладной математики и информатики

**Дискретная математика**

Отчет по лабораторной работе № 2

Тема "Графы"

**Вариант 6 – «Алгоритм Флойда-Уоршалла»**

Выполнил студент гр. 5030102/20202 Тишковец С. Е.

Санкт-Петербург

2024

**Оглавление**

[**Постановка задачи** 3](#_Toc179830351)

[**Язык программирования** 3](#_Toc179830352)

[**Описание алгоритма** 3](#_Toc179830353)

[**Демонстрация алгоритма** 5](#_Toc179830354)

[**Формат данных** 7](#_Toc179830355)

[**Дополнительные требования** 7](#_Toc179830356)

[**Область применимости алгоритма** 8](#_Toc179830357)

[**Источники информации** 8](#_Toc179830358)

# **Постановка задачи**

*Вариант 3 – Алгоритм Флойда-Уоршалла*

На вход программе подаётся взвешенный орграф.

Дополнительные требования:

* указать сложность алгоритма и доказать, что она именно такая;
* сравнение работы алгоритма на различных допустимых входных данных: на каких графах алгоритм работает лучше, на каких – хуже, на каких – вообще не работает;
* объяснить почему был выбран тот или иной способ представления графов в программе.

# **Язык программирования**

Алгоритм был реализован с помощью **C++**

# **Описание алгоритма**

Алгоритм Флойда-Уоршалла:

1. Чтение матрицы смежности из файла input.txt и запись данных в двумерный массив
2. Определение размерности матрицы ***n*** – иными словами, количество вершин графа. В случае, когда граф отсутствует (размерность матрицы равна 0), программа заканчивает работу.
3. Создается двумерный массив ***distance*** для хранения конечного результата работы алгоритма и двумерный массив ***path*** для хранения непосредственно путей между всеми вершинами. На начальном этапе данные исходной матрицы копируются в массив ***distance****.* Значения ***path[i][i]*** инициализируются нулями, отсутствие пути обозначено в матрице как ***-1,*** остальные значения ***path[i][j] = i.***
4. Реализуется поиск кратчайшего пути между всеми вершинами графа с помощью трёх вложенных циклов по ***k, i, j*** (от ***0*** до ***n-1***):

- Внешний выбирает вершину ***k***, через которую мы пробуем найти лучший путь. Внутренние два перебирают пары **(*i*, *j)*** вершин, между которыми мы ищем минимальное расстояние.

- Для каждой пары ***(i, j)*** вершин соответственно возможны два случая:

* + ***k*** не является промежуточной вершиной в кратчайшем пути от ***i*** до ***j***. Мы сохраняем значения ***distance[i][j]***и ***path[i][j]***такими, какое оно есть.
  + ***k*** — промежуточная вершина в кратчайшем пути от ***i*** до ***j*** . Мы обновляем значение ***distance[i][j]*** как ***distance[i][k]*** + ***distance[k][j],*** если ***distance[i][j]*** > ***distance[i][k]*** + ***distance[k][j].*** Значение ***path[i][j]***обновляется как ***path[k][j]*.**

- Если ***distance[i][k]*** или ***distance[k][j]*** равны ***INF,*** это означает, что через данную вершину ***k*** пути между вершинами ***i*** и ***j*** не существует. Переход по несуществующим путям изначально не должен происходить. Поскольку данная реализация алгоритма использует в качестве значения ***INF*** конкретное число 999, а не общепринятое понятие бесконечности (то есть что-то недостижимое), нужно ставить дополнительные проверки на то, что ***distance[i][k]*** и ***distance[k][j]*** не равны ***INF.***

- Проверка на наличие в графе цикла отрицательного веса – то есть, когда расстояние от вершины ***i*** к вершине ***i*** меньше ***0***.

1. Вывод результата работы алгоритма на экран – выводятся матрица и пути между всеми вершинами (за исключением пути от вершины ***i*** к вершине ***i***).

# **Демонстрация алгоритма**

Изображение выглядит как линия, круг, диаграмма

Автоматически созданное описаниеПример 1. Рассмотрим следующий взвешенный орграф

Матрица смежности выглядит следующим образом

Матрица путей выглядит:

Пояснения: в графе 4 вершины, пронумерованных числами от 1 до 4. Граф задан матрицей смежности , вес ребра ***i - j*** хранится в . При отсутствии ребра ***i - j*** значение , также будем считать, что . В матрице отсутствие путей обозначено как **-1**, , остальные значения .

**Шаг 1:** Рассматриваем 1-ю вершину в качестве промежуточного узла и вычисляем значения ***distance[i][j]*** для каждой пары узлов ***(i, j)*** по формуле ***distance[i][j] = min(distance[i][j], distance[i][1] + distance[1][j]).***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 |
| 1 | 0 | 5 | -1 | INF |
| 2 | INF | ? | ? | ? |
| 3 | INF | ? | ? | ? |
| 4 | INF | ? | ? | ? |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 |
| 1 | 0 | 5 | -1 | INF |
| 2 | INF | 0 | INF | INF |
| 3 | INF | -2 | 0 | 3 |
| 4 | INF | 7 | INF | 0 |

Поскольку к вершине 1 не проведены направленные ребра, таблица после первого шага не будет отличаться от исходной. Значения ***P[i][j]***тоже остались неизменными.

**Шаг 2:** Рассматриваем 2-ю вершину в качестве промежуточного узла и вычисляем значения ***distance[i][j]*** для каждой пары узлов ***(i, j)*** по формуле ***distance[i][j] = min(distance[i][j], distance[i][2] + distance[2][j]).***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 |
| 1 | ? | 5 | ? | ? |
| 2 | INF | 0 | INF | INF |
| 3 | ? | -2 | ? | ? |
| 4 | ? | 7 | ? | ? |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 |
| 1 | 0 | 5 | -1 | INF |
| 2 | INF | 0 | INF | INF |
| 3 | INF | -2 | 0 | 3 |
| 4 | INF | 7 | INF | 0 |

Здесь ситуация другая: ко 2-ой вершине проведены направленные ребра, однако от 2-ой вершины таковых «стрелочек» нет – поэтому результат второго шага не вносит изменений в общую картину. Значения ***P[i][j]***тоже остались неизменными.

**Шаг 3:** Рассматриваем 3-ю вершину в качестве промежуточного узла и вычисляем значения ***distance[i][j]*** для каждой пары узлов ***(i, j)*** по формуле ***distance[i][j] = min(distance[i][j], distance[i][3] + distance[3][j]).***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 |
| 1 | ? | ? | -1 | ? |
| 2 | ? | ? | INF | ? |
| 3 | INF | -2 | 0 | 3 |
| 4 | ? | ? | INF | ? |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 |
| 1 | 0 | -3 | -1 | 2 |
| 2 | INF | 0 | INF | INF |
| 3 | INF | -2 | 0 | 3 |
| 4 | INF | 7 | INF | 0 |

Замечаем, что в двух ячейках значения изменились в меньшую сторону.

Матрица примет следующий вид:

**Шаг 4:** Рассматриваем 4-ю вершину в качестве промежуточного узла и вычисляем значения ***distance[i][j]*** для каждой пары узлов ***(i, j)*** по формуле ***distance[i][j] = min(distance[i][j], distance[i][4] + distance[4][j]).***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 |
| 1 | ? | ? | ? | 2 |
| 2 | ? | ? | ? | INF |
| 3 | ? | ? | ? | 3 |
| 4 | INF | 7 | INF | 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 |
| 1 | 0 | -3 | -1 | 2 |
| 2 | INF | 0 | INF | INF |
| 3 | INF | -2 | 0 | 3 |
| 4 | INF | 7 | INF | 0 |

В результате 4-го шага матрица осталась той же, что и после 3-го шага. Значения ***P[i][j]***тоже остались неизменными.

# **Формат данных**

Входные данные: матрица смежности считывается из текстового файла. В первой строке указана размерность матрицы – количество ребер в графе, далее построчно, начиная со второй строки, в файле записаны соответствующие строки матрицы. Все символы текстового файла являются целыми числами.

Выходные данные: результатом выполнения программы являются матрица такой же размерности, как и исходная, и список всех кратчайших путей в графе. Матрица выводится в консоль в стандартном виде.

# **Дополнительные требования**

* **Временная сложность:** O(), где n — количество вершин в графе. Поскольку мы запускаем три вложенных цикла, каждый из которых перебирает ***n*** символов, получаем сложность
* **Способ представления графа** в данной программе – матрица смежности – был выбран не спроста. Матрица смежности является наиболее простым и оптимальным способом представления взвешенных орграфов. Реализация, например, списка смежности для взвешенных графов является уже более трудоемкой. Время доступа для проверки наличия ребра в матрице смежности является постоянным, а в списке смежности — линейным.
* **Сравнение работы алгоритма на различных допустимых входных данных:**

**-** временнаясложностьработы алгоритма одинакова для всех взвешенных орграфов, поскольку алгоритм проходится в каждом из трех циклах по всем значениям от ***0*** до ***n-1.*** Исключение составляет случай, когда матрица смежности ориентированного графа будет симметричной – тогда можно рассматривать не все элементы матрицы, а только те, что лежат, например, выше главной диагонали (и на самой диагонали, естественно, тоже).

**-** учитывая замечание выше, стоит рассмотреть применимость алгоритма к плотным и разреженным графам. Эффективность работы программы будет высокой как раз для более плотного графа, поскольку в нем количество ребер превышает число вершин.

**-** алгоритм Флойда-Уоршалла завершит свое выполнение с ошибкой в случае, если у графа будет обнаружен хотя бы один отрицательный цикл.

# **Область применимости алгоритма**

1. Область применения:

Алгоритм Флойда-Уоршалла является фундаментальным алгоритмом в информатике и теории графов. Он используется для поиска кратчайших путей между всеми парами узлов во взвешенном графе. Этот алгоритм достаточно эффективен и может обрабатывать графы как с положительными, так и с отрицательными весами ребер, что делает его универсальным инструментом для решения широкого спектра сетевых и связных проблем.

Также, данный алгоритм может быть использован для обнаружения в графе отрицательных циклов.

1. Случаи сбоя:

- Предельное значение ***INF,*** как правило, является конкретным числом. В случае, если вес ребра в графе будет превышать предельное значение, программа сработает не корректно.

- Нарушение формата входных данных: например, если в файле не все элементы матрицы смежности будут заполнены, то алгоритм выдаст некорректный результат.

- Наличие у графа отрицательных циклов: при наличии таких циклов программа завершит выполнение с ошибкой.

# **Источники информации**

* <https://www.techiedelight.com/ru/pairs-shortest-paths-floyd-warshall-algorithm/>
* <https://www.geeksforgeeks.org/floyd-warshall-algorithm-dp-16/>