Спецификация

**Тема:** Алгоритм Флойда- Уоршелла

**Группа:** 5303

**Студенты:** Матвеева А.И.

Нигай А.С.

Табунникова Н.Р.

**Теория**

Алгоритм Флойда-Уоршелла — алгоритм для нахождения кратчайших расстояний между всеми вершинами взвешенного [графа](http://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) без циклов с отрицательными весами с использованием метода динамического программирования. Алгоритм легко модифицировать таким образом, чтобы он возвращал не только длину кратчайшего пути, но и сам путь. Для этого достаточно завести дополнительный массив next, в котором будет храниться номер вершины, в которую надо пойти следующей, чтобы дойти из u в v по кратчайшему пути.

**Постановка задачи**

Дан взвешенный ориентированный [граф](https://neerc.ifmo.ru/wiki/index.php?title=%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D0%BD%D1%8B%D0%B5_%D0%BE%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F:_%D0%B3%D1%80%D0%B0%D1%84,_%D1%80%D0%B5%D0%B1%D1%80%D0%BE,_%D0%B2%D0%B5%D1%80%D1%88%D0%B8%D0%BD%D0%B0,_%D1%81%D1%82%D0%B5%D0%BF%D0%B5%D0%BD%D1%8C,_%D0%BF%D0%B5%D1%82%D0%BB%D1%8F,_%D0%BF%D1%83%D1%82%D1%8C,_%D1%86%D0%B8%D0%BA%D0%BB) G(V, E), в котором вершины пронумерованы от 1 до n.

\omega_{uv} = \begin{cases} \text{weight of }uv ,& \text{if } uv \in E \\ +\infty ,& \text{if } uv \notin E \end{cases}  
Требуется найти матрицу кратчайших расстояний d, в которой элемент d_{ij} либо равен длине кратчайшего пути из i в j, либо равен 0, если вершина j не достижима из i.

**Алгоритм**

Пусть вершины графа {\displaystyle G=(V,\;E),\;|V|=n}G=(V,E), |V|=n  пронумерованы от 1 до {\displaystyle n}n и введено обозначение   
{\displaystyle d\_{ij}^{k}} для длины кратчайшего пути от {\displaystyle i}i до {\displaystyle j}j, который кроме самих вершин {\displaystyle i,\;j}i, j проходит только через вершины {\displaystyle 1\ldots k}1..k. Очевидно, что {\displaystyle d\_{ij}^{0}} — длина (вес) ребра {\displaystyle (i,\;j)}(i,j), если таковое существует (в противном случае его длина может быть обозначена как {\displaystyle \infty }∞).

Существует два варианта значения {\displaystyle d\_{ij}^{k},\;k\in \mathbb {(} 1,\;\ldots ,\;n)}:

1. Кратчайший путь между {\displaystyle i,\;j}i, j не проходит через вершину {\displaystyle k}k, тогда {\displaystyle d\_{ij}^{k}=d\_{ij}^{k-1}}
2. Существует более короткий путь между {\displaystyle i,\;j}i, j, проходящий через {\displaystyle k}k, тогда он сначала идёт от {\displaystyle i}i до {\displaystyle k}k, а потом от {\displaystyle k}k до {\displaystyle j}i. В этом случае, очевидно, {\displaystyle d\_{ij}^{k}=d\_{ik}^{k-1}+d\_{kj}^{k-1}}

Таким образом, для нахождения значения функции достаточно выбрать минимум из двух обозначенных значений.

Тогда [рекуррентная](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BA%D1%83%D1%80%D1%81%D0%B8%D1%8F) формула для {\displaystyle d\_{ij}^{k}} имеет вид:

{\displaystyle d\_{ij}^{0}} — длина ребра {\displaystyle (i,\;j);}(i, j);

{\displaystyle d\_{ij}^{k}=\min(d\_{ij}^{k-1},\;d\_{ik}^{k-1}+d\_{kj}^{k-1}).}

Алгоритм Флойда-Уоршелла последовательно вычисляет все значения {\displaystyle d\_{ij}^{k},} {\displaystyle \forall i,\;j} для {\displaystyle k}k от 1 до {\displaystyle n}n. Полученные значения {\displaystyle d\_{ij}^{n}} являются длинами кратчайших путей между вершинами {\displaystyle i,\;j.}i, j.

Алгоритм работает за \Theta(n^3) времени и использует \Theta(n^2) памяти.

**Реализация**

На вход программе подаётся граф, заданный в виде матрицы смежности — двумерного массива d[][] размера n×n, в котором каждый элемент задаёт длину ребра между соответствующими вершинами.

Требуется, чтобы выполнялось d[i][i]=0 для любых i.

**Псевдокод**

На каждом шаге алгоритм генерирует [матрицу](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D1%80%D0%B8%D1%86%D0%B0_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) {\displaystyle W,}W, .  
{\displaystyle w\_{ij}=d\_{ij}^{n}.}Матрица {\displaystyle W}W содержит длины кратчайших путей между всеми вершинами графа. Перед работой алгоритма матрица {\displaystyle W}W заполняется длинами рёбер графа (или запредельно большим *M*, если ребра нет).

**for** k = 1 **to** n

**for** i = 1 **to** n

**for** j = 1 **to** n

W[i][j] = min(W[i][j], W[i][k] + W[k][j])

**Особые случаи**

**Случай вещественных весов**

Если веса рёбер графа не целочисленные, а вещественные, то следует учитывать погрешности, неизбежно возникающие при работе с типами с плавающей точкой.

Применительно к алгоритму Флойда неприятным спецэффектом этих погрешностей становится то, что найденные алгоритмом расстояния могут уйти сильно в минус из-за накопившихся ошибок. В самом деле, если на первой фазе имела место ошибка \Delta, то на второй итерации эта ошибка уже может превратиться в 2 \Delta, на третьей — в 4 \Delta, и так далее.

Чтобы этого не происходило, сравнения в алгоритме Флойда следует делать с учётом погрешности:

**if** (d[i][k] + d[k][j] < d[i][j] - EPS)

d[i][j] = d[i][k] + d[k][j];

**Случай отрицательных циклов**

Если в графе есть циклы отрицательного веса, то формально алгоритм Флойда-Уоршелла неприменим к такому графу.

На самом же деле, для тех пар вершин i и j, между которыми нельзя зайти в цикл отрицательного вес, алгоритм отработает корректно.

Для тех же пар вершин, ответа для которых не существует (по причине наличия отрицательного цикла на пути между ними), алгоритм Флойда найдёт в качестве ответа какое-то число (возможно, сильно отрицательное, но не обязательно). Тем не менее, можно улучшить алгоритм Флойда, чтобы он аккуратно обрабатывал такие пары вершин и выводил для них, например, - \infty.

Для этого можно сделать, например, следующий критерий "не существования пути". Итак, пусть на данном графе отработал обычный алгоритм Флойда. Тогда между вершинами i и j не существует кратчайшего пути тогда и только тогда, когда найдётся такая вершина t, достижимая из i и из которой достижима j, для которой выполняется d[t][t]<0.

Кроме того, при использовании алгоритма Флойда для графов с отрицательными циклами следует помнить, что возникающие в процессе работы расстояния могут сильно уходить в минус, экспоненциально с каждой фазой. Поэтому следует принять меры против целочисленного переполнения, ограничив все расстояния снизу какой-нибудь величиной (например, - {\rm INF}).

# Требования к программе

Программа должна выполнять следующие действия:

* По введенной матрице смежности/ матрице инцидентности/ списку инцидентности отображать граф, работу алгоритма Флойда – Уоршелла;
* Генерировать новый граф;
* Иметь и при надобности выводить справку для пользователя.

Действия, возможные для разработки:

* График зависимости времени поиска пути от времени

**Эскиз интерфейса**

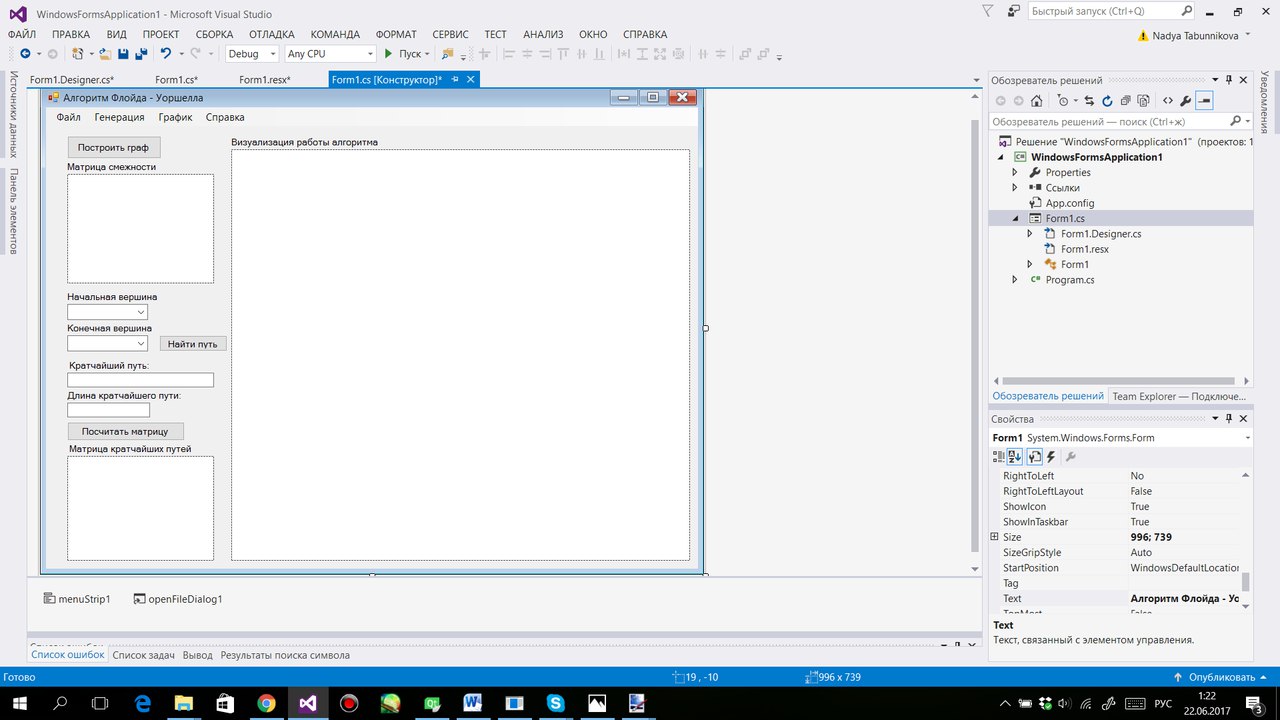


Рис. 1

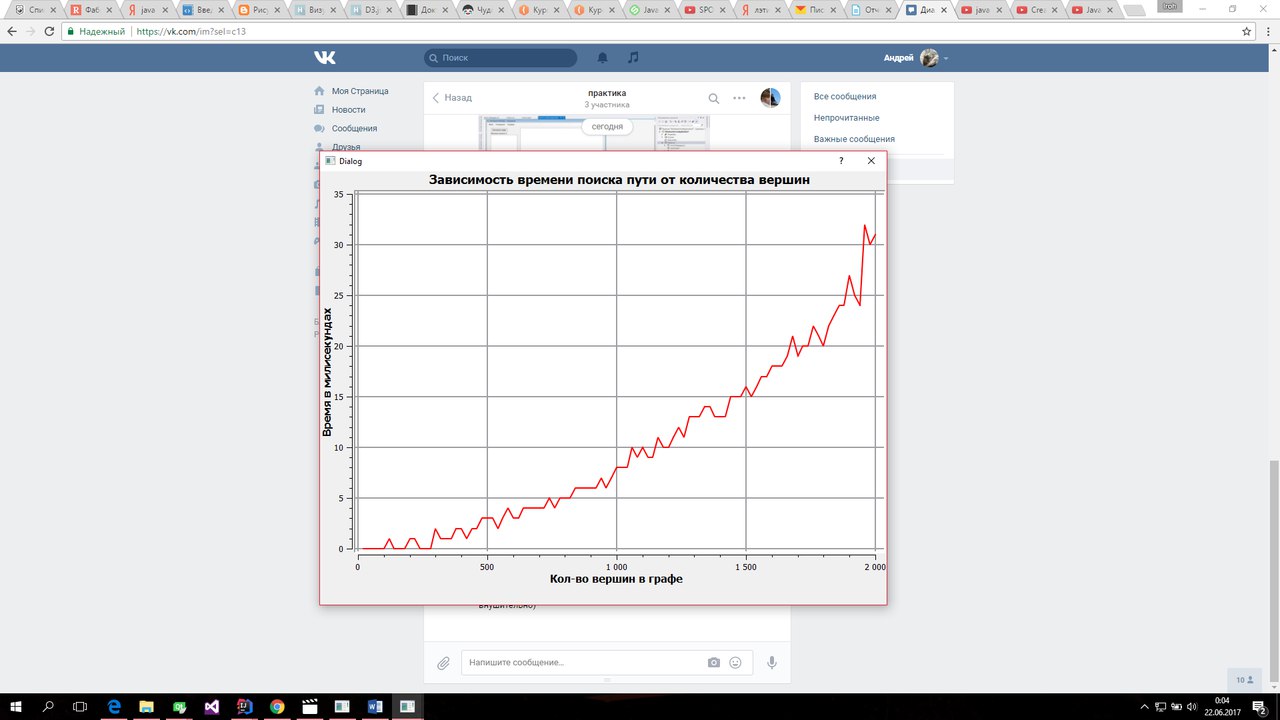


Рис. 2

**Работа меню:**

**Файл** – взять матрицу смежности из файла.

**Генерация** – сгенерировать матрицу смежности.

**Справка** – получить справку (руководство пользователя). Справка открывается в отдельном окне.

**График** – построить график зависимости времени поиска пути от количества вершин. График открывается в отдельном окне, будет разработан, если останется свободное время (Рис 2).

**Работа программы:**

Программа может выполнять следующие действия:

* построить граф по матрице смежности;
* найти кратчайший путь между двумя вершинами заданного графа;
* посчитать матрицу кратчайших путей для заданного графа;

В случае если останется дополнительное время для разработки:

* построить график зависимости времени поиска пути от количества вершин.

Для того чтобы построить граф, нужно ввести матрицу смежности в соответствующее поле (вручную, сгенерировать или взять из файла) и нажать кнопку **«Построить граф»**. Граф построится в поле для визуализации работы алгоритма.

Для того чтобы найти кратчайший путь между двумя вершинами заданного графа, нужно выбрать начальную и конечную вершину и нажать кнопку **«Найти путь»**. Кратчайший путь и его длина появятся в соответствующих полях.

Для того чтобы посчитать матрицу кратчайших путей для заданного графа, нужно нажать кнопку **«Посчитать матрицу»**. Матрица кратчайших путей появится в соответствующем поле.

За построение графика зависимости времени поиска пути от количества вершин отвечает кнопка меню **«График»**.

# Предварительная диаграмма классов:

Class ChartForm

Class Algorithm

Class Reader

Class Writer

Class Matrix

# Описание классов:

Class Matrix - класс, описывающий взвешенный ориентированный граф. Должен хранить матрицу смежности, кол-во вершин графа.

Class Algorithm – класс алгоритма, агрегирует матрицу смежности, поддерживает поиск кратчайших путей в графе алгоритмом Флойда – Уоршелла, содержит матрицу кратчайших путей, метод нахождения этой матрицы, метод нахождения кратчайшего пути между двумя вершинами.

Class ChartForm – класс графика зависимости времени работы алгоритма от размера графа, наследуется от Algorithm, содержит метод построения и отображения графика.

Class Reader – содержит статические методы прочтения из файла матрицы смежности / матрицы инцидентности / списка инцидентности графа и преобразования матрицы инцидентности, списка инцидентности в матрицу смежности.

Class Writer - содержит статические методы записи в файл данных, полученных в результате работы программы, таких как матрица кратчайших путей, кратчайший путь.

# Формат входных данных:

Для получения корректного результата программе нужна матрица смежности / матрица инцидентности / список инцидентности в виде последовательности целых чисел.

# План разработки:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Создание прототипа, 26 июня | 1-ая версия, 28 июня | 2-ая (конечная) версия, 30 июня |
| Нигай А.С. | Разработка класса BaseGraph:  реализация визуализации графов. Разработка GraphWithAlg методов класса поиска всех кратчайших путей, поиски кратчайшего пути между двумя вершинами для графов с неотрицательными весами рёбер. | Визуализация работы алгоритма Флойда – Уоршелла, разработка метода генерации графа.  Разработка метода нахождения циклов отрицательного веса. | Разработка класса Schedule, методов построения и отображения графика. |
| Табунникова Н.Р. | Разработка системы ввода/вывода из файла. | Разработка механизмов обеспечивающих корректность вводимых данных. | Разработка метода преобразования списка инцидентности в матрицу смежности. |
| Матвеева А.И. | Разработка системы ввода/вывода из GUI. | Разработка механизмов обеспечивающих корректность вводимых данных. | Разработка метода преобразования матрицы инцидентности в матрицу смежности. |