**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ  
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ  
«ЛЭТИ» ИМ. В.И.УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра МО ЭВМ**

**СПЕЦИФИКАЦИЯ**

|  |  |  |
| --- | --- | --- |
| Студенты гр. 5304 |  | Божко С.В. |
|  |  | Ильясов Е. |
|  |  | Смурова А.И. |
| Преподаватель |  | Герасимова Т.В. |

Санкт-Петербург

2017

# Цель

Реализовать алгоритм Дейкстры для поиска наикратчайшего пути от одной вершины до всех остальных на базе языка Java для взвешенного ориентированного графа. Для удобства использования программы реализовать демонстрацию решения, а также понятный интерфейс.

# Описание алгоритма

Пусть дан взвешенный ориентированный граф *G*(*V*, *E*) без дуг отрицательного веса.

Задача: Найти кратчайшие пути от некоторой вершины *А* графа *G* до всех остальных вершин этого графа.

Каждой вершине из *V* сопоставим метку — минимальное известное расстояние от этой вершины до *А*. Алгоритм работает пошагово — на каждом шаге он «посещает» одну вершину и пытается уменьшать метки. Работа алгоритма завершается, когда все вершины посещены.

Инициализация. Метка самой вершины *А* полагается равной 0, метки остальных вершин — бесконечности. Это отражает то, что расстояния от *А* до других вершин пока неизвестны. Все вершины графа помечаются как не посещённые вершины.

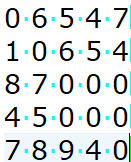
Шаг алгоритма. Если все вершины посещены, алгоритм завершается. В противном случае, из ещё не посещённых вершин выбирается вершина *U*, имеющая минимальную метку. Мы рассматриваем всевозможные маршруты, в которых *U* является предпоследним пунктом. Вершины, в которые ведут рёбра из *U*, назовём соседями этой вершины. Для каждого соседа вершины *U*, кроме отмеченных как посещённые, рассмотрим новую длину пути, равную сумме значений текущей метки *U* и длины ребра, соединяющего *U* с этим соседом. Если полученное значение длины меньше значения метки соседа, заменим значение метки полученным значением длины. Рассмотрев всех соседей, пометим вершину *U* как посещённую и повторим шаг алгоритма.

# Формат входных и выходных данных

Входные данные:

Ввод графа можно осуществить двумя способами:

1. Граф задается матрицей смежности из файла .txt в виде:

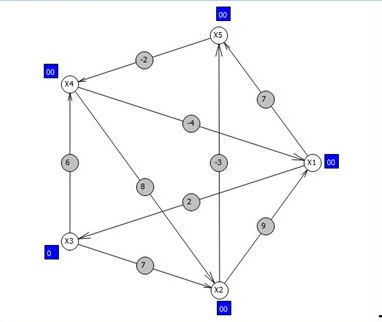


2. С помощью кнопки «Добавить вершину» пользователь добавляет необходимое количество вершин, а с помощью кнопки «Добавить ребро» указывает связь между двумя вершинами и стоимость этого ребра.

Начальная вершина при запуске поиска решения.

Выходные данные:

Граф визуализируется в окне в виде:

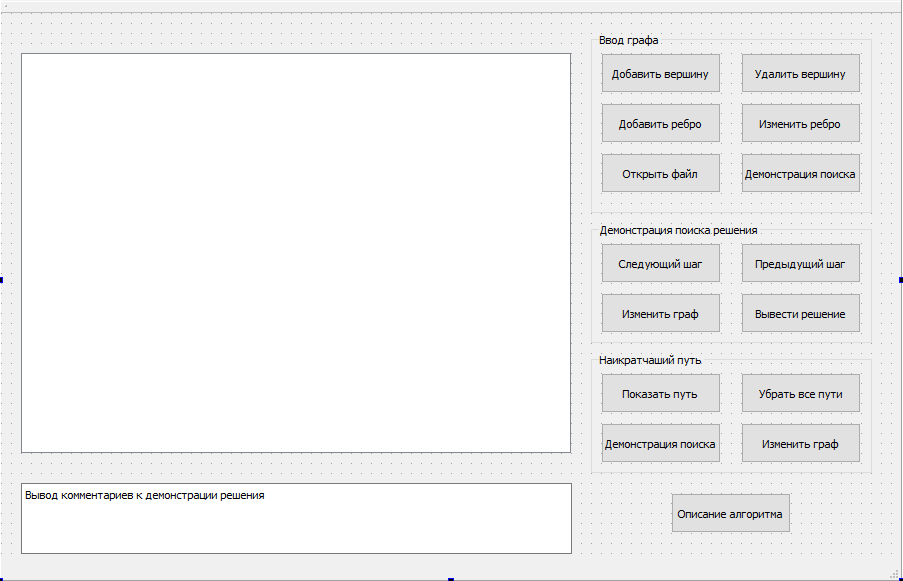


Графический пошаговый вывод алгоритма. Стоимости кратчайших путей до каждой вершины от начальной, а также кратчайший путь от начальной вершины до вершины с введенным номером.

# Примерный вид интерфейса

Примерный вид интерфейса

*Рисунок 1. Интерфейс программы*



Описание интерфейса программы:

1. Сцена, на которой будет рисоваться введенный граф и процесс поиска решения.
2. В окне под сценой будет показываться дополнительная информация о ходе решения, чтобы был понятен каждый шаг решения.
3. Первая группа кнопок для ввода и изменения графа. Эта группа кнопок доступна только если происходит редактирование графа. Во время редактирования графа остальные две группы не доступны.
   1. Кнопка «Добавить вершину» - добавляет еще одну вершину в граф
   2. Кнопка «Удалить вершину» - удаляет вершину с введенным номером и все ребра инцидентные ей.
   3. Кнопка «Добавить ребро» - добавляет ребро. Вызывается окошко, в котором вводятся номера начальной и конечной вершин, а также стоимость ребра.
   4. Кнопка «Изменить ребро» - изменяет стоимость или удаляет ребро.
   5. Кнопка «Открыть файл» - для считывания данных из файла.
   6. Кнопка «Демонстрация поиска» - делает активным вторую группу кнопок, а эта группа кнопок оказывается заблокированной.
4. Вторая группа кнопок для демонстрации поиска решения задачи. Во время демонстрации поиска остальные две группы кнопок недоступны. При активации данной группы кнопок в окне под сценой будут выводится комментарии к каждому шагу решения.
   1. Кнопка «Следующий шаг» - показывает следующий шаг в решении
   2. Кнопка «Предыдущий шаг» - показывает предыдущий шаг в решении
   3. Кнопка «Изменить граф» - делает активным первую группу кнопок, а эта группа кнопок оказывается заблокированной.
   4. Кнопка «Вывести решение» - делает активным третью группу кнопок, а эта группа кнопок оказывается заблокированной.
5. Третья группа кнопок для демонстрации решения задачи. Во время демонстрации решения остальные две группы кнопок недоступны. При активации данной группы кнопок в окне под сценой будут выведены пути от начальной вершины до всех остальных вершин.
   1. Кнопка «Показать путь» - запрашивает номер вершины, до которой показать решение и показывает путь на сцене.
   2. Кнопка «Убрать все пути» - убирает нарисованный путь на сцене и возле каждой вершины показывать стоимость наикратчайшего пути от начальной и предыдущую вершину для этого пути.
   3. Кнопка «Изменить граф» аналогична кнопке из второй группы.
   4. Кнопка «Демонстрация пути» аналогична кнопке из первой группы.
6. Кнопка «Описание алгоритма» выводит описание алгоритма.

# Визуализация графа и поиска решения

Визуализация графа заключается в том, чтобы все введенные данные пользователем отображались на нем, т.е. на графе должны быть так же изображены стоимости ребер.

Визуализация поиска решения. Рассматривается каждый шаг и при его демонстрации другим цветом выделяются две вершины и ребро, которое их связывает. Также отмечается, изменилось ли наикратчайшее расстояние или нет.

Визуализация решения. После ввода конечной вершины, находятся наикратчайшие пути и ребра и вершины, входящие в эти пути, выделяются другим цветом.

# Структуры данных

Для реализации поставленной задачи используется 3 структуры данных:

1. struct Vertex – это описание вершины графа.

*Таблица 1. Свойства структуры Vertex*

|  |  |
| --- | --- |
| **Свойство структуры** | **Описание свойства** |
| int number | Номер вершины |
| pair <int, int> coordinate | Координаты вершины для демонстрации графа |
| int distance | Кратчайшее расстояние до начальной вершины |
| vector<int> previousVertex | Список предыдущих вершин для восстановления всех наикратчайших путей до данной вершины |

1. struct Edge – это описание ребра графа

*Таблица 2. Описание свойств структуры Edge*

|  |  |
| --- | --- |
| **Свойства структуры** | **Описание свойства** |
| Vertex first | Вершина, из которой выходит ребро |
| Vertex second | Вершина, в которую входит ребро |
| int weight | Стоимость ребра |

1. struct Graph – это описание графа

*Таблица 3. Описание свойств структуры Graph*

|  |  |
| --- | --- |
| **Свойства структуры** | **Описание свойства** |
| vector <Vertex> vertex | Список всех вершин графа |
| vector <Edge> edge | Список всех ребер графа |

# Псевдокод алгоритма

**void** dijkstra(s)**:**

{

**for** v \in V

{ d[v] = \infty

used[v] = *false*

}

d[s] = 0

**for** i \in V

{ v = *null*

**for** j \in V // найдем вершину с минимальным расстоянием

{ **if** !used[j] **and** (v == *null* **or** d[j] < d[v])

v = j

}

**if** d[v] == \infty

**break**

used[v] = *true*

**for** e : исходящие из *v* рёбра // произведём релаксацию по всем рёбрам, исходящим из *v*

{ **if** d[v] + e.len < d[e.to]

d[e.to] = d[v] + e.len

}

}

}

# План работы

26.06.2017.

Создание прототипа (интерфейс программы, один из способов ввода графа, визуализация графа)

28.06.2017.

Работающая программа без контроля последовательности действий пользователя.

30.06.2017.

Финальная версия программы с контролем последовательности действий пользователя. Пояснительная записка

# Распределение обязанностей

Реализация интерфейса пользователя - Смурова А.И.

Реализация алгоритма - Божко С.В.

Визуализация алгоритма - Ильясов Е.