**Задание:** Визуализация алгоритма Форда-Беллмана.

**Спецификация**

*Внешняя спецификация****.***

Входные данные: взвешенный ориентированный граф, выделенная вершина.

Промежуточные данные: картинка, соответствующая каждому шагу алгоритма.

Выходные данные: расстояния от выделенной вершины до остальных вершин в графе.

*Место и форма представления входных и выходных данных****.***

Граф:  
- представление графа в файле:  
[номер 1 вершины] [пробел] [номер 2 вершины] [пробел] [вес вершины],

- ввод графа вручную.   
 Пользователь двойным щелчком мыши добавляет вершину, которая изначально не связана с другими вершинами. Вершина становится выбранной, если по ней щелкнуть левой кнопкой мыши. Чтобы соединить две вершины, нужно их выбрать.

Выделенная вершина выбирается во время работы программы, перед запуском алгоритма, левым щелчком мыши.

Выходные данные содержатся в специально выделенной таблице, которая содержит расстояния от выделенной вершины до других, полученных на каждом шаге. Визуальное представление графа содержит выделенную вершину, расстояния до других вершин, ребра, входящие в путь, отделенные цветом от других. Вся эта информация находится в главном окне программы.

*Интерфейс*.

рис.1. Схема главного окна программы.

Меню

Рисунок графа

Таблица (1), содержащая все ребра графа

Таблица (2), содержащая расстояния от выделенной вершины до остальных (изначально неактивна)

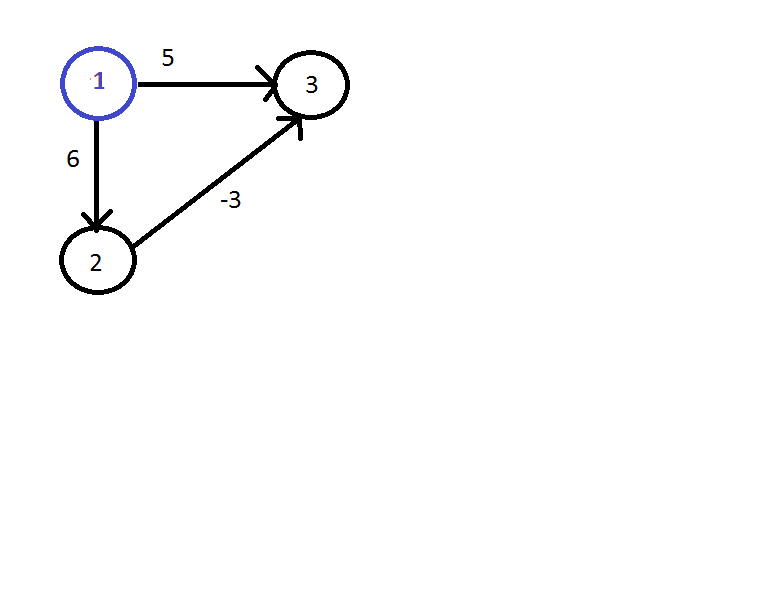
Кнопка «Показать алгоритм» («Сделать шаг»)

*Визуализация*. Выделенная вершина окрашивается в синий цвет.

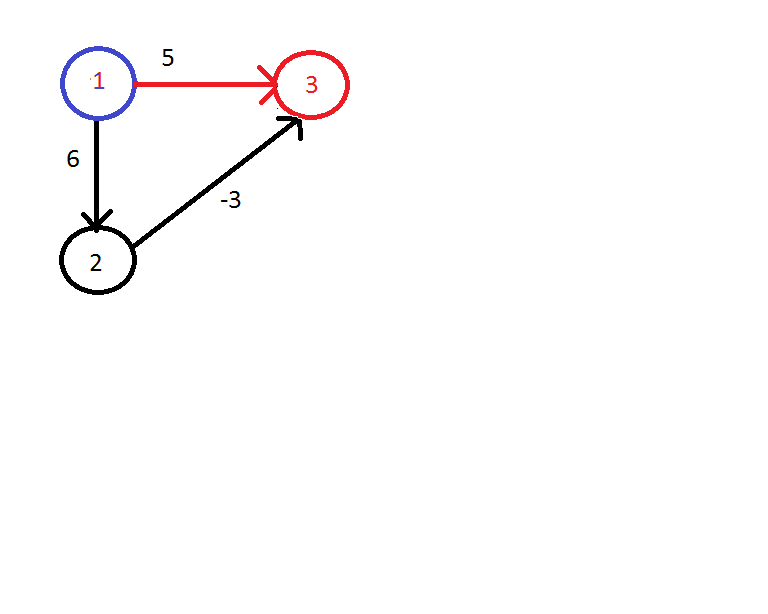
По нажатию кнопки «Сделать шаг»происходит следующее:  
- ослабляемое (релаксируемое) ребро и вершина, выделяются красным цветом,  
- ребра, входящие в путь, выделяются зеленым цветом,  
- в соответствии с изменениями на данном шаге изменяется таблица (2).

Рис. 2. Пример визуализации (первые несколько шагов).

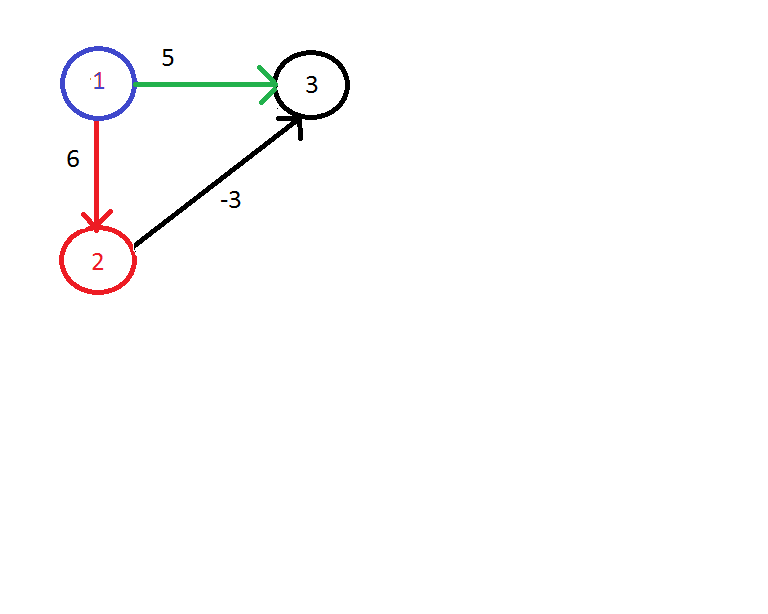
1 шаг.



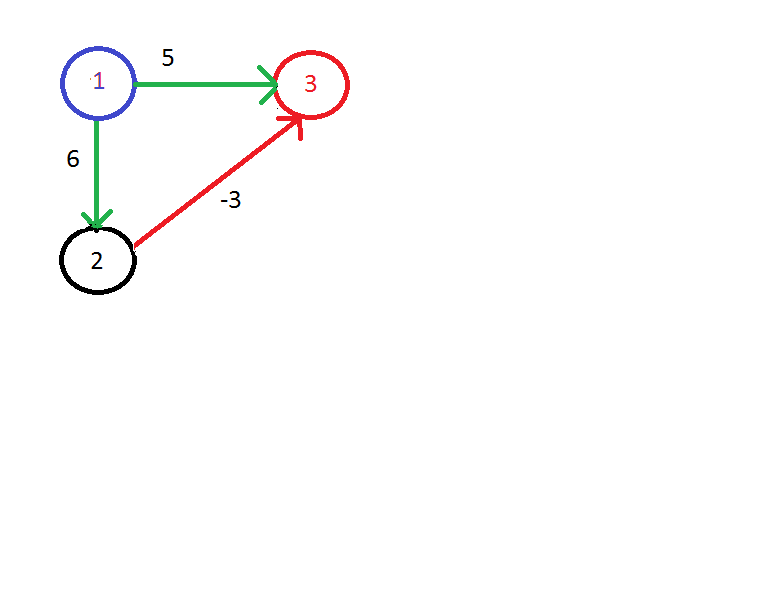
2 шаг.



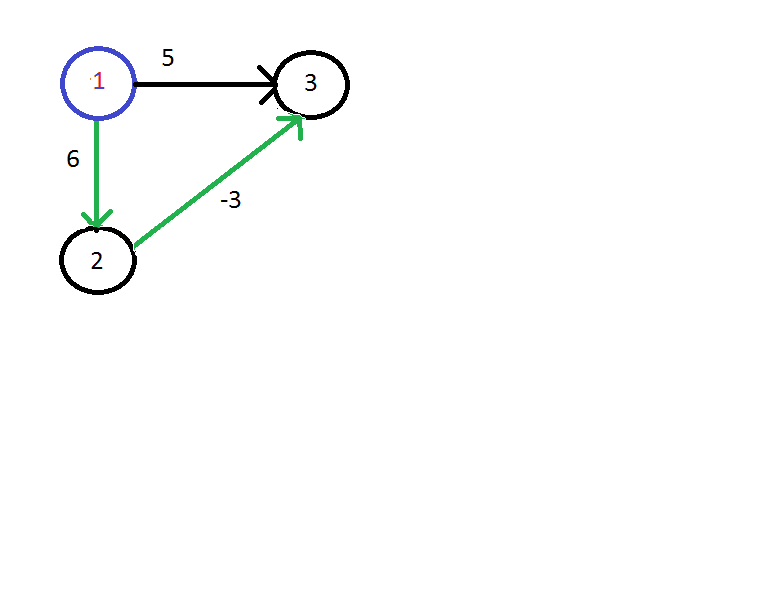
3 шаг.



4 шаг.



5 шаг.



*Внутренняя спецификация.*

Представление графа в программе, состоящего из N вершин и M ребер – список вершин, пронумерованных от 1 до N, и список ребер. Ребро – две вершины и вес этого ребра.  
Над графом определены базовые функции: добавление вершины, добавление ребра.

Описание структуры данных на псевдокоде:

class Graph  
{  
 int N; *//число вершин*  
 int M; *//число ребер*  
 struct {  
 vert a; *//номер вершины, из которой исходит ребро* vert b; *//номер вершины, в которую входит ребро*  
 int w; *//вес ребра*  
 } edge[M];  
*//базовые функции*  
 void addVert(); //*добавление вершины*  
 void addEdge(vert u, vert v, int weight); //*добавление ребра u → v с весом weight*  
}

**План разработки**

|  |  |  |
| --- | --- | --- |
| Дата | Что должно быть готово | Ответственное лицо |
| 1 неделя | | |
| Среда | Внешняя спецификация, частично внутренняя, план разработки | Совместно. |
| Четверг | Репозиторий | Андреев. |
| Пятница | Прототип графического интерфейса. | Вихляев В. |
| Суббота | Пошаговая реализация алгоритма Ф-Б без графического интерфейса. | Осипов В. |
| 2 неделя | | |
| Понедельник | 1 версия приложения  Можно обойтись без:  - полной визуализации (допускается вывод только правильного ответа),  - корректности ввода (касается ввода из файла), - без рисования вручную /ввода из файла (должно быть сделано одно из двух)  - правильной реакции программы на неправильные действия (например, добавление вершины при выполнении алгоритма),  - красивого интерфейса (неполное меню, справка и т.д.). | Реализация интерфейса – Вихляев В.  Вывод графа на экран и ввод графа вручную – Андреев Н.  Реализация алгоритма – Осипов В. |
| Вторник | 1 версия приложения (если не был сделан) |
| Среда | 2 версия |
| Четверг | 2 версия + отчет, обновленный репозиторий |
| Пятница | Должно быть сдано/показано. |