Министерство образования Республики Беларусь

Учреждение образования

“Белорусский государственный университет

информатики и радиоэлектроники”

Факультет информационных технологий и управления

Кафедра интеллектуальных информационных технологий

**РАСЧЕТНАЯ РАБОТА**

по дисциплине «Традиционные и интеллектуальные информационные системы»

на тему   
«Найти максимальный путь между заданными вершинами»

Выполнил Тесловский А.П.

студент группы

721702

Проверил Шункевич Д.В.

Минск 2018

**Цель:** Получить навыки формализации и обработки информации с использованием семантических сетей

**Задача:** Найти максимальный путь между заданными вершинами

**Список понятий:**

1. Графовая структура (абсолютное понятие) - это такая одноуровневая реляционная структура, объекты которой могут играть роль либо вершины, либо связки:
   1. Вершина (относительное понятие, ролевое отношение);
   2. Связка (относительное понятие, ролевое отношение).

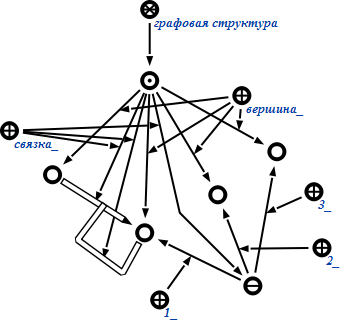


Рис. 1 Графовая структура

1. Ориентированный граф (абсолютное понятие) – это такой граф, в котором все связки являются дугами:
   1. Дуга (относительное понятие, ролевое отношение) – ориентированная связка.

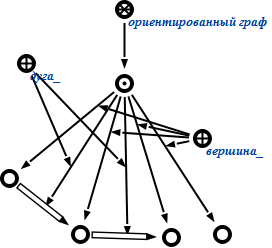


Рис. 2 Ориентированный граф

10. Взвешенный граф (абсолютное понятие) – граф, связкам которого поставлены в соответствие веса.

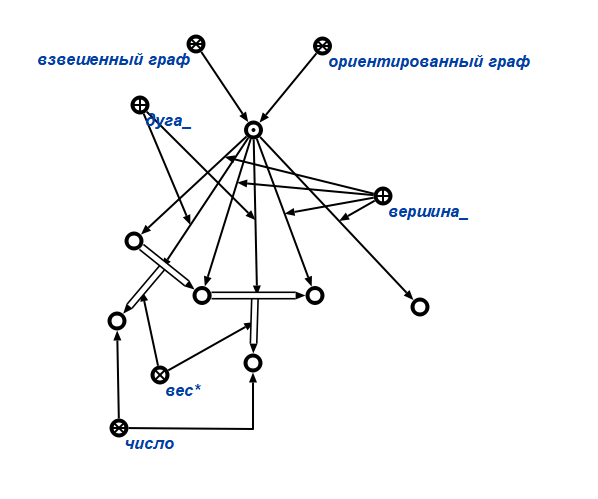


Рис. 3 Взвешенный граф

11. Маршрут (относительное понятие, бинарное ориентированное отношение) – это чередующаяся последовательность вершин и гиперсвязок в гиперграфе, которая начинается и кончается вершиной, и каждая гиперсвязка последовательности инцидентна двум вершинам, одна из которых непосредственно предшествует ей, а другая непосредственно следует за ней. В примере ниже показан маршрут A, CON1, C, CON2, D, CON3, B, CON1, A в гиперграфе.

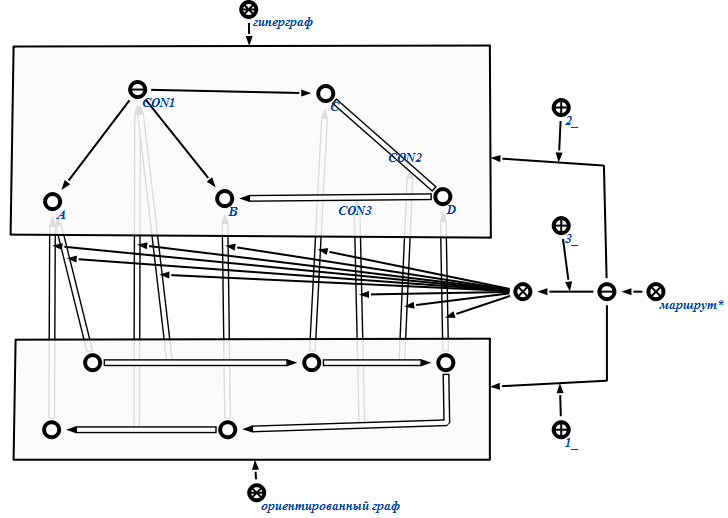


Рис. 4 Маршрут

12. Цепь (относительное понятие, бинарное ориентированное отношение) – это маршрут, все гиперсвязки которого различны. В примере ниже показана цепь A, CON1, C, CON2, D, CON3, B, CON4, A в гиперграфе.

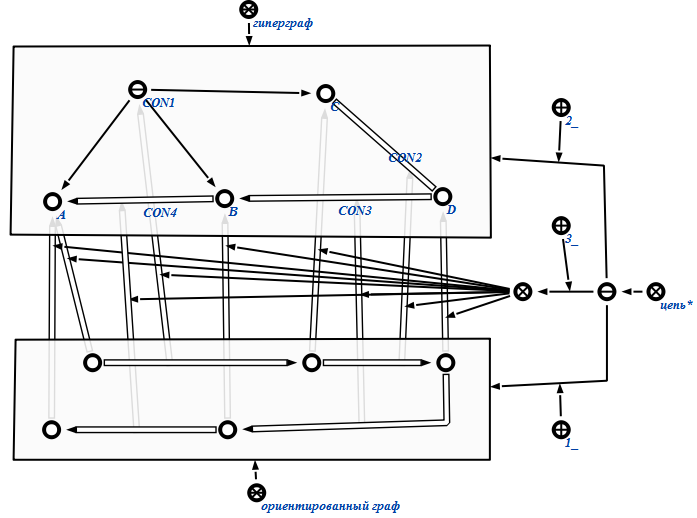


Рис. 5 Цепь

13. Простая цепь, путь (относительное понятие, бинарное ориентированное отношение) – это цепь, в которой все вершины различны. В примере ниже показан путь A, CON1, C, CON2, D, CON3, B в гиперграфе.

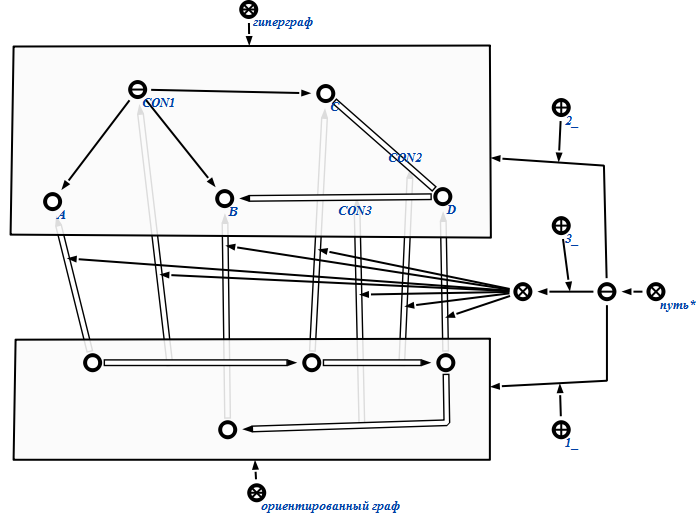


Рис. 6 Простая цепь

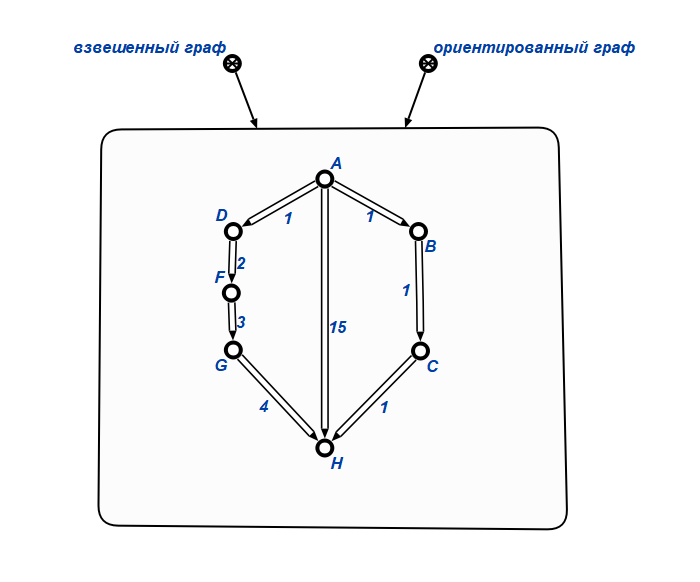
# Тестовые примеры

Во всех тестах и описании алгоритма ниже графы будет приведены в сокращенной форме со скрытыми ролями элементов графа.

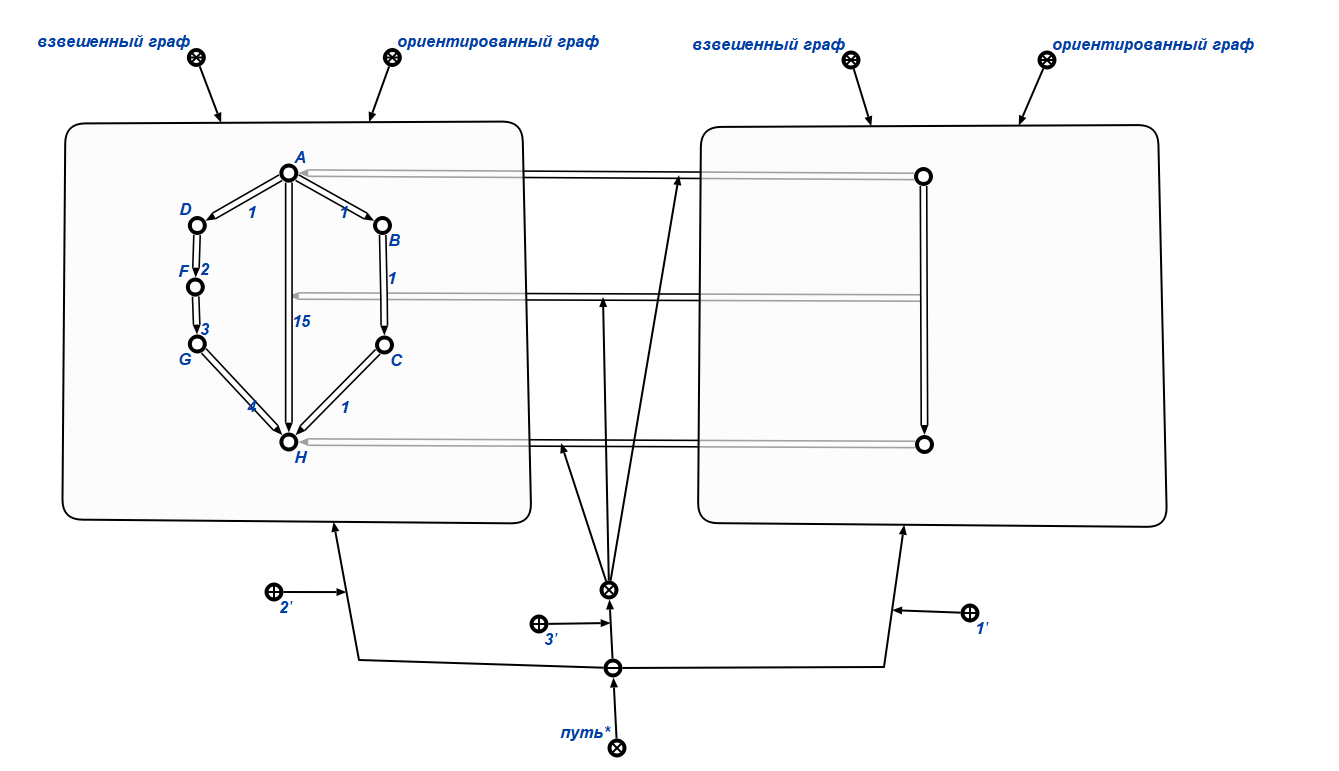
## Тест 1

Задача: найти максимальный путь между вершинами A и H.

Ввод:

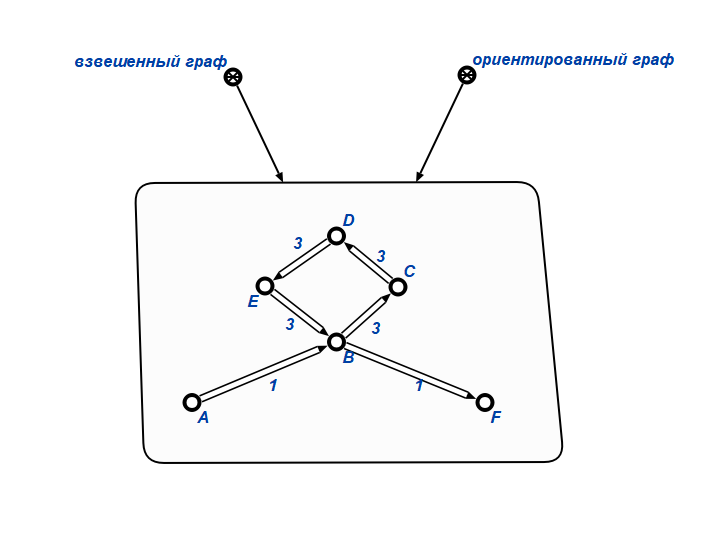


Вывод:

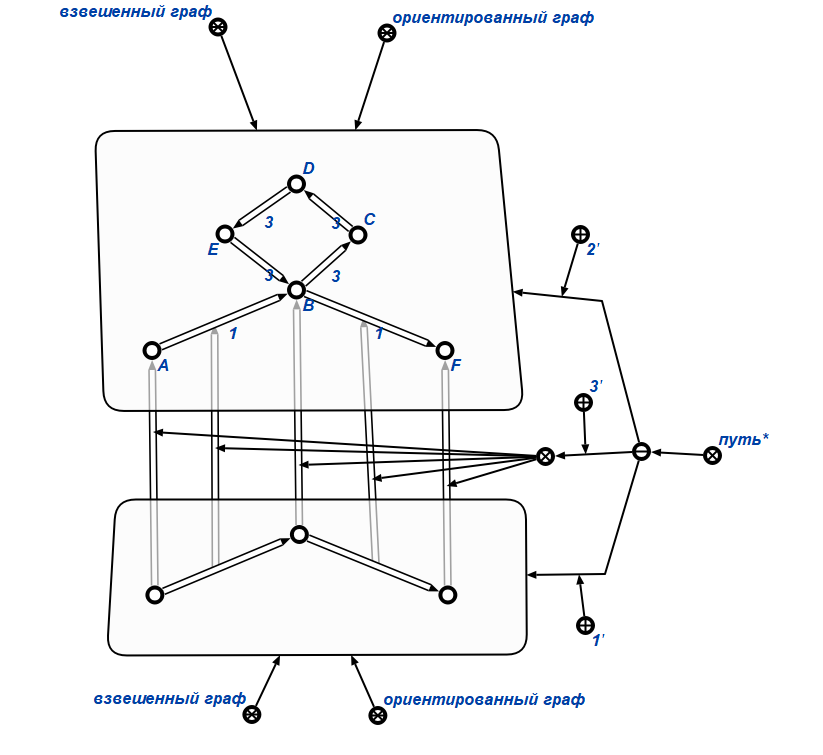


## Тест 2

Задача: найти максимальный путь между вершинами A и F.

Ввод: 

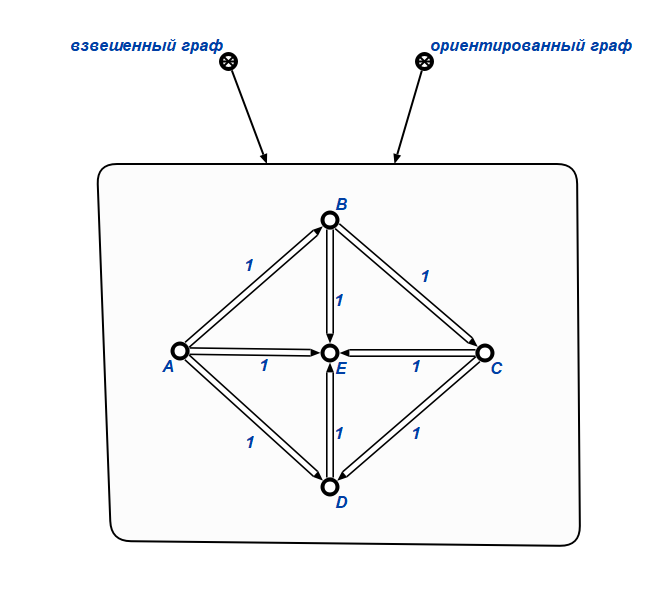
Вывод:



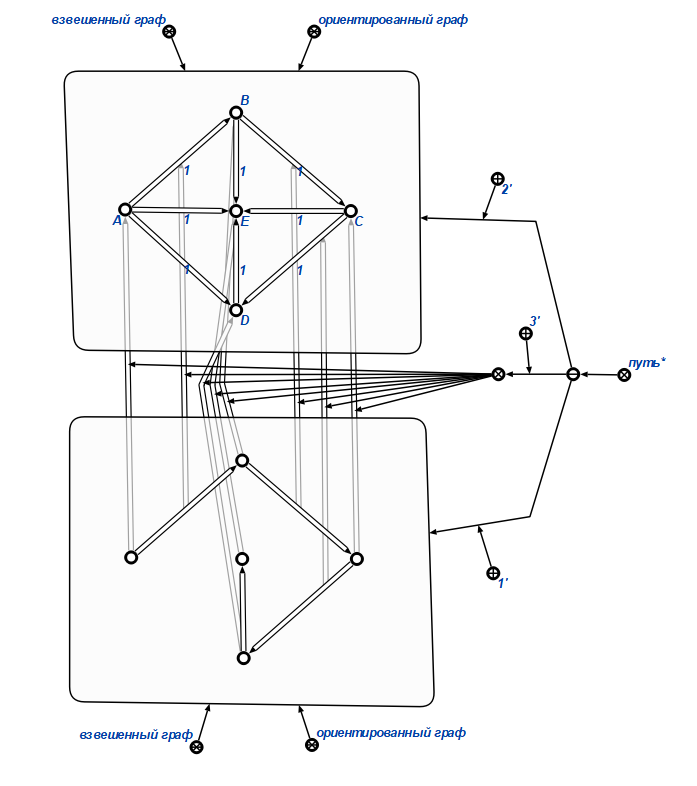
## Тест 3

Задача: найти максимальный путь между вершинами A и E.

Ввод:

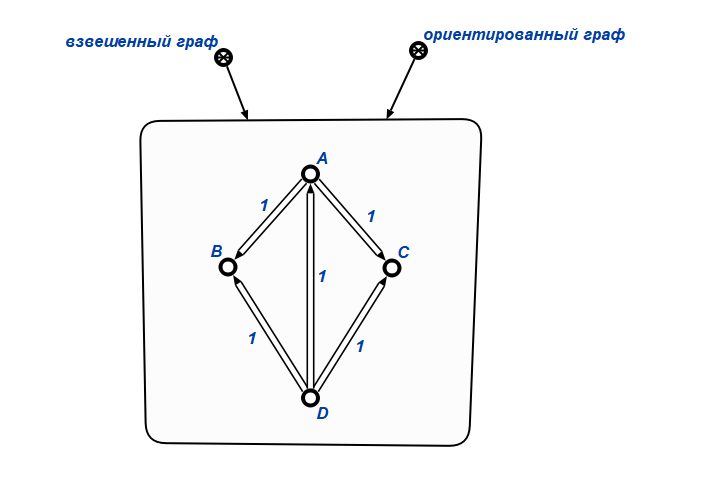


Вывод:



## Тест 4

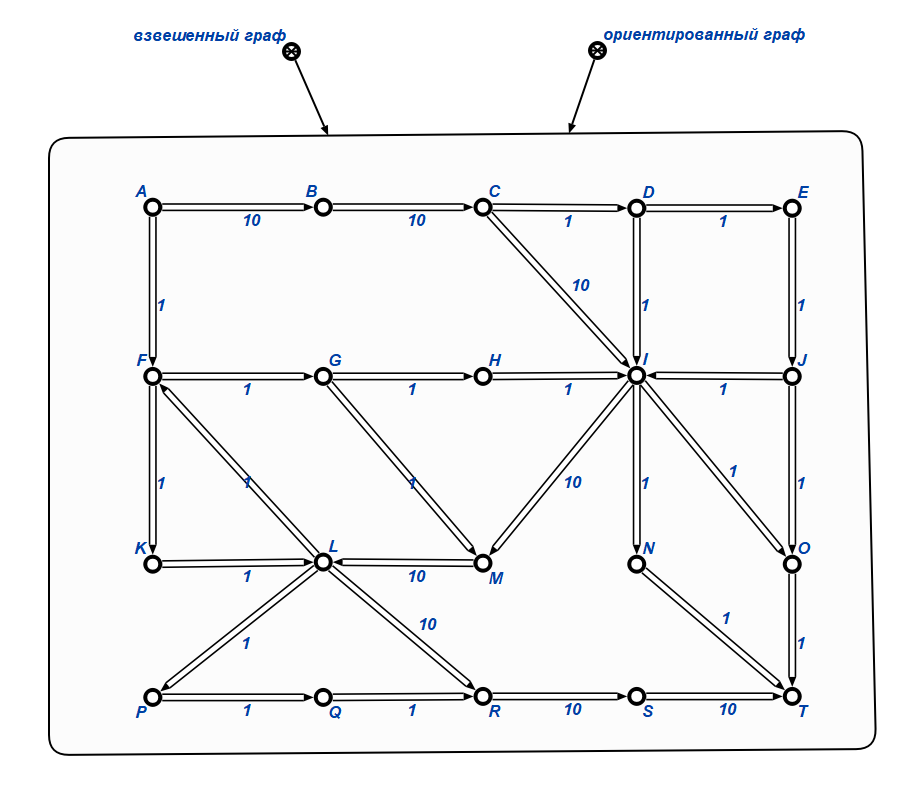
Задача: найти максимальный путь между вершинами A и D.

Ввод: 

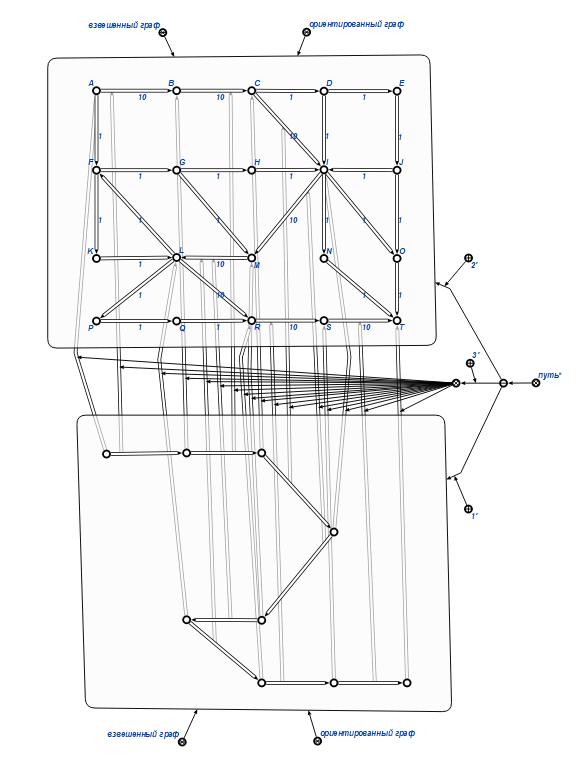
Вывод: Путь не существует.

## Тест 5

Задача: найти максимальный путь между вершинами A и T.

Ввод: 

Вывод:



**Алгоритм**

Для выполнения задачи будем использовать следующие переменные:

\_graph – исходный граф.

\_start и \_destination – начальная и конечная точки пути соответственно.

\_this – проверяема вершина.

\_path – текущий путь.

\_ pathways – множество, содержащее все найденные пути от \_start до \_destination.

\_next – множество, содержащее для каждой вершины все достижимые из неё вершины.

1.

Пользователь задаёт граф, он заносится в переменную \_graph.

Пользователь определяет \_start и \_destination для поиска пути.

Создаётся множество \_next.

\_this = \_start.

2.

Вершина \_this заносится в \_path.

Если \_this != start

Занесённая в \_path вершина соединяется ориентированной дугой с предыдущей вершиной в \_path.

Если \_this = \_destination

переходим к пункту 3.

Если множество \_next для текущей вершины \_this не пусто

Для вершины \_this берём из множества \_next вершину, перемещаем \_this на эту вершину, удаляем её из \_next, переходим к пункту 2.

Иначе

Если this = start

Переходим к пункту 4.

Иначе

Перемещаем \_this на предыдущую вершину в \_path, удаляем две последние вершины из \_path, переходим к пункту 2.

3.

Заносим \_path во множество \_pathways, удаляем \_this из \_next, перемещаем \_this на предыдущую вершину в \_path, удаляем две последние вершины из \_path, переходим к пункту 2.

4.

Удаляем переменные \_this, \_start, \_destination и множество \_next.

Из множества \_pathways выбираем самый длинный путь, выводим его на экран.

Удаляем множество \_pathways.

**Пример выполнения алгоритма в sc-памяти:**

1. Заданный граф заносится в переменную \_graph.

В \_start и \_destination заносятся соответствующие вершины.

Для заданного графа создаётся множество \_next.

Вершина А заносится в\_this.

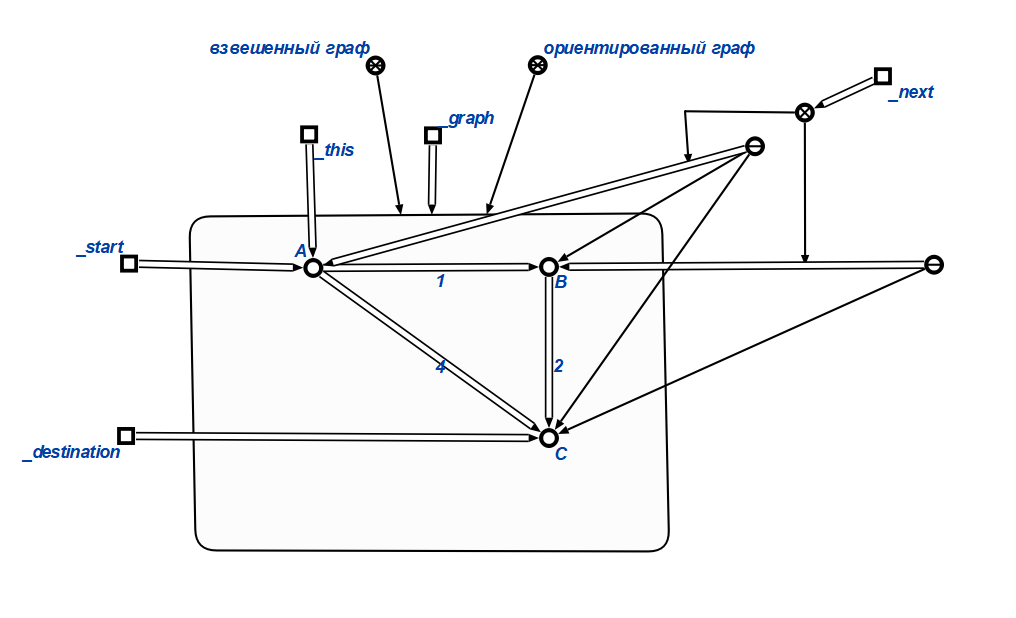


Иллюстрация пункта 1 алгоритма

2. Вершина А заносится в \_path.

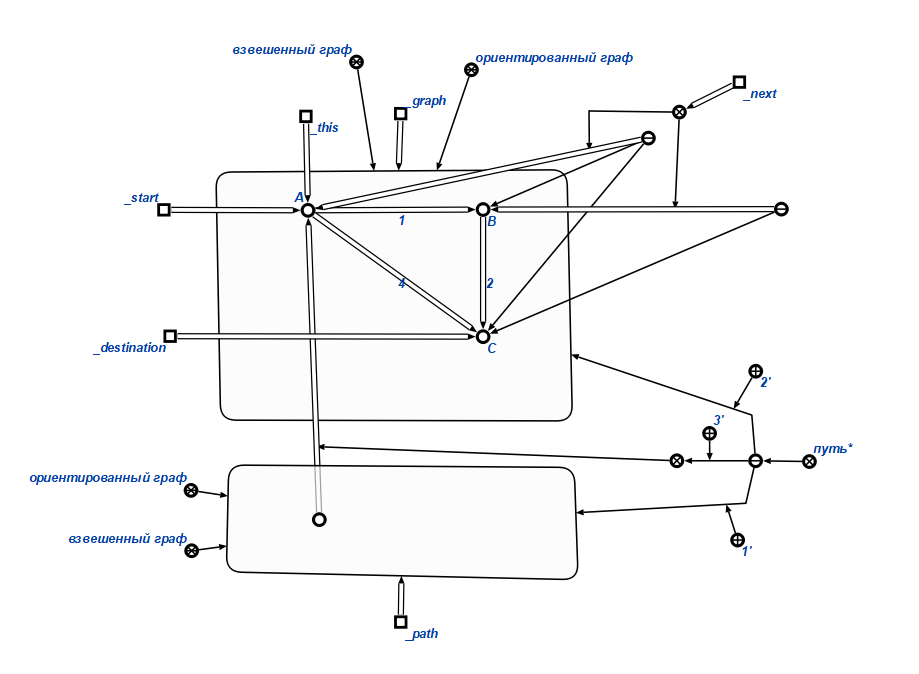


Иллюстрация пункта 2 алгоритма

3. \_this перемещается на вершину В из \_next.

B удаляется из \_next для А.

B заносится в \_path и соединяется дугой с А.

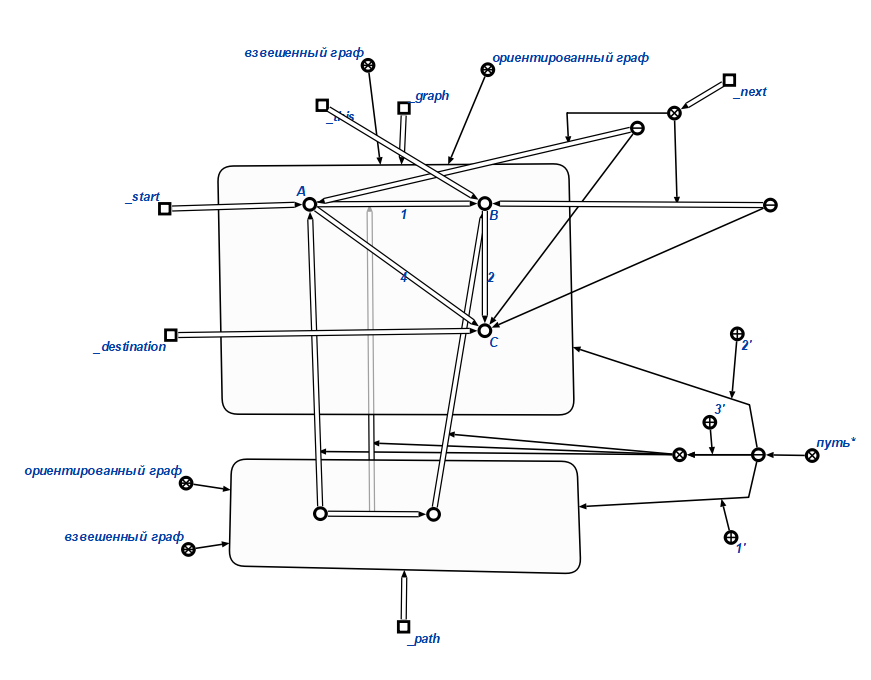
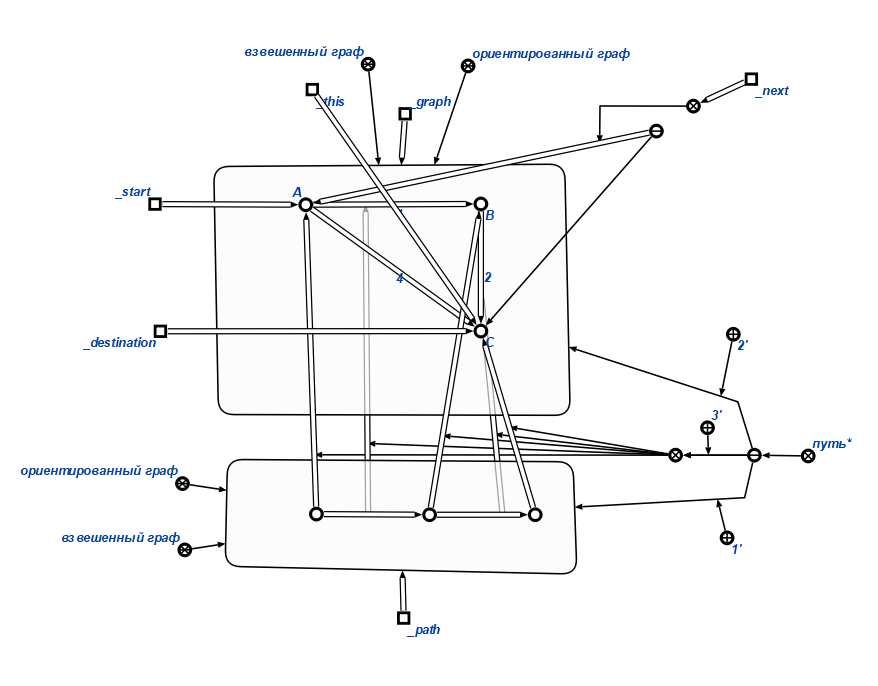


Иллюстрация пункта 3 алгоритма

4. \_this перемещается на вершину С из \_next.

С удаляется из \_next для В.

С заносится в \_path и соединяется дугой с B.

Иллюстрация пункта 4 алгоритма

5. Так как вершина С = \_destination, то заносим \_path в \_pathways.

Вершина С удаляется из \_path.

\_this возвращается на вершину B.

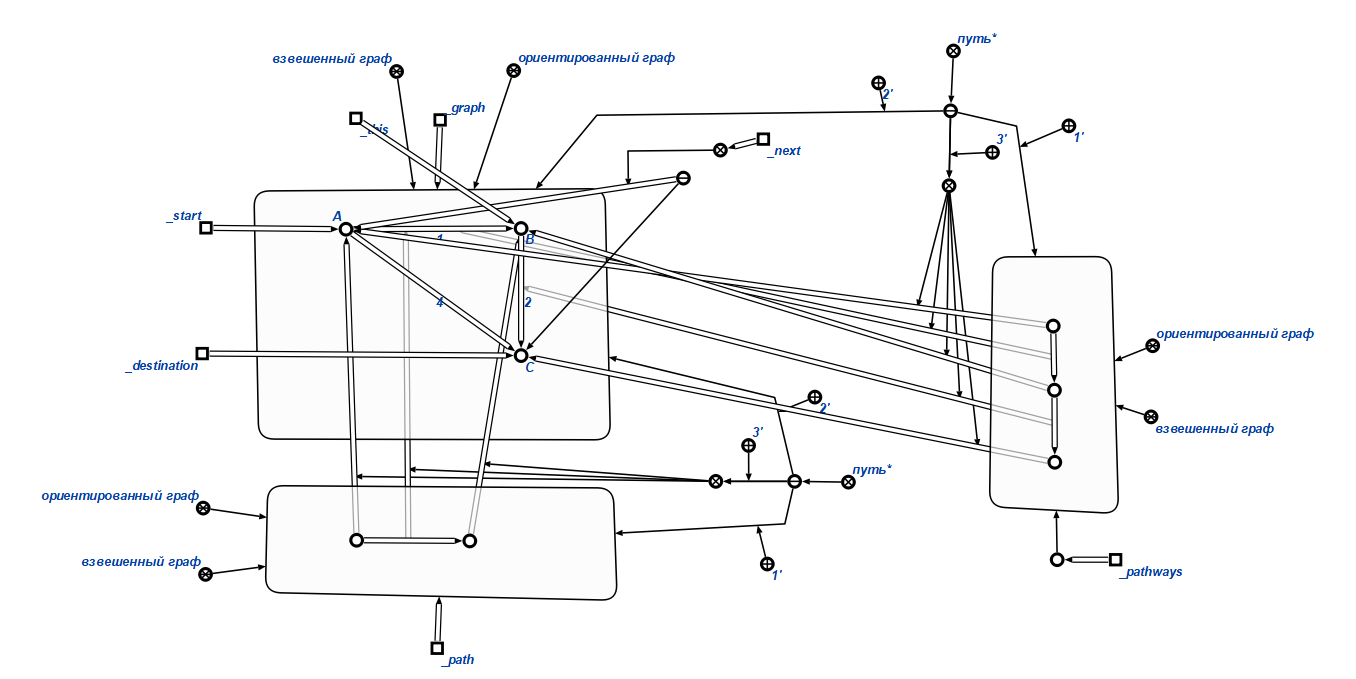


Иллюстрация пункта 5 алгоритма

6. Так как для вершины В \_next не содержит других вершин, то В удаляется из \_path и \_this переносится на вершину А.

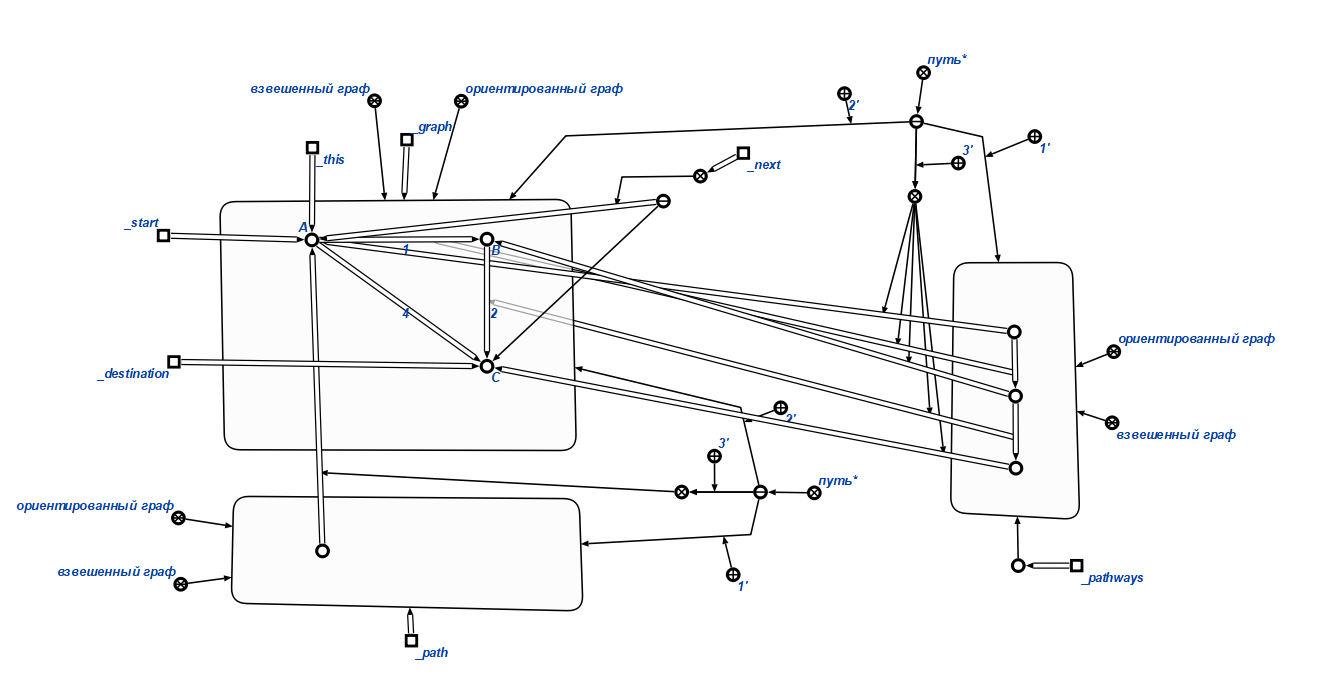


Иллюстрация пункта 6 алгоритма

7. Для вершины А в \_next есть вершина С, значит

\_this перемещается на вершину С из \_next.

С удаляется из \_next для А.

С заносится в \_path и соединяется дугой с А.

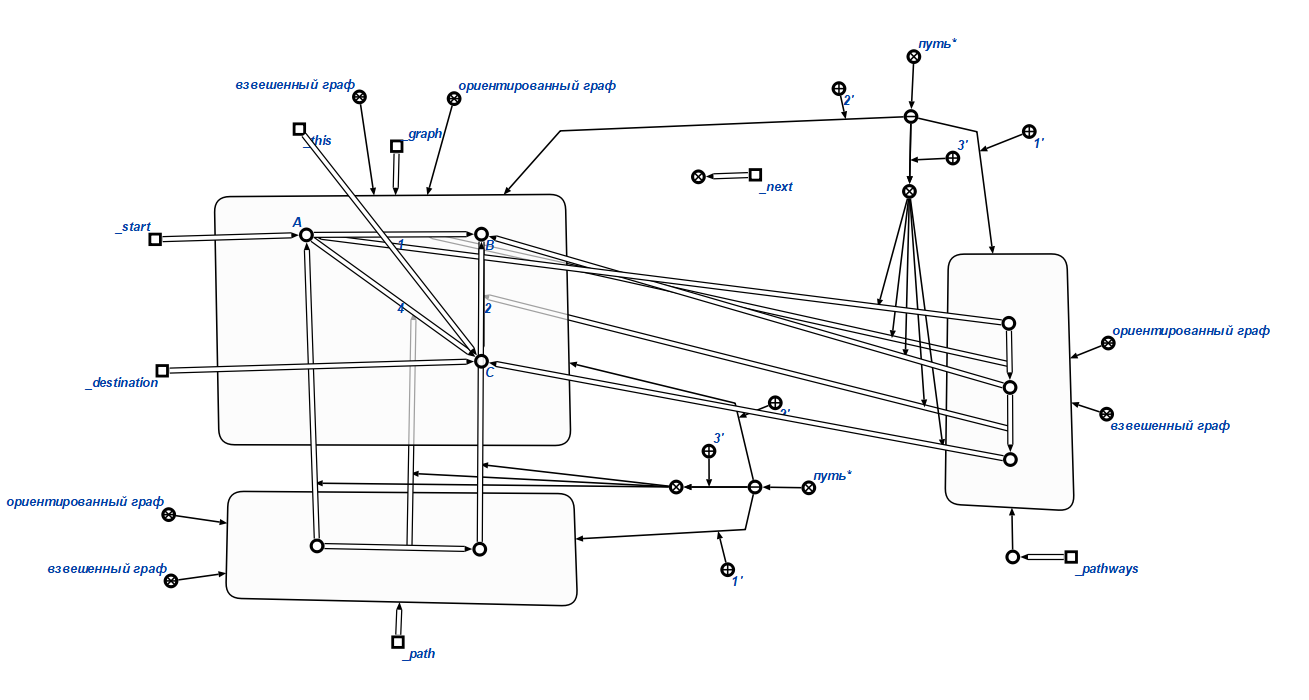


Иллюстрация пункта 7 алгоритма

8. Так как вершина С = \_destination, то заносим \_path в \_pathways.

Вершина С удаляется из \_path.

\_this возвращается на вершину А.

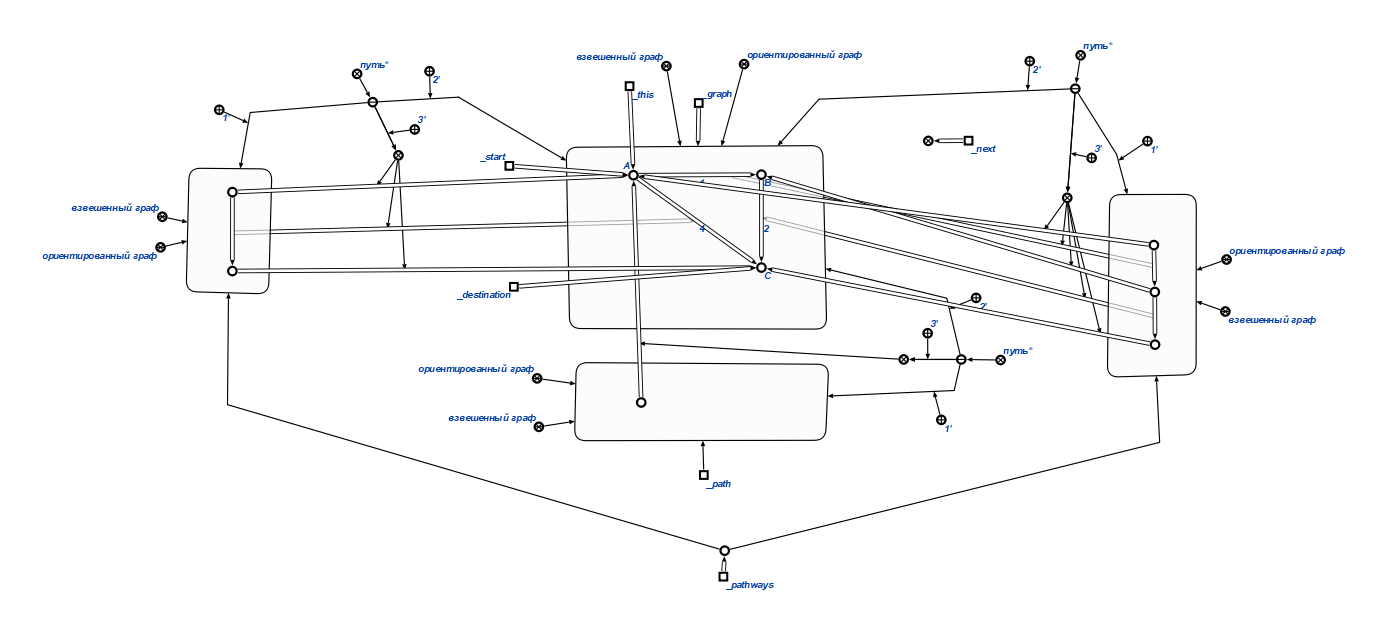


Иллюстрация пункта 8 алгоритма

9. Так как вершина А = \_start, а множество \_next для А пусто, то удаляем лишние переменные \_next, \_path, \_start, \_destination, \_this

Выбираем максимальный путь из \_pathways, этот путь является решением поставленной задачи.

Удаляем \_pathways.

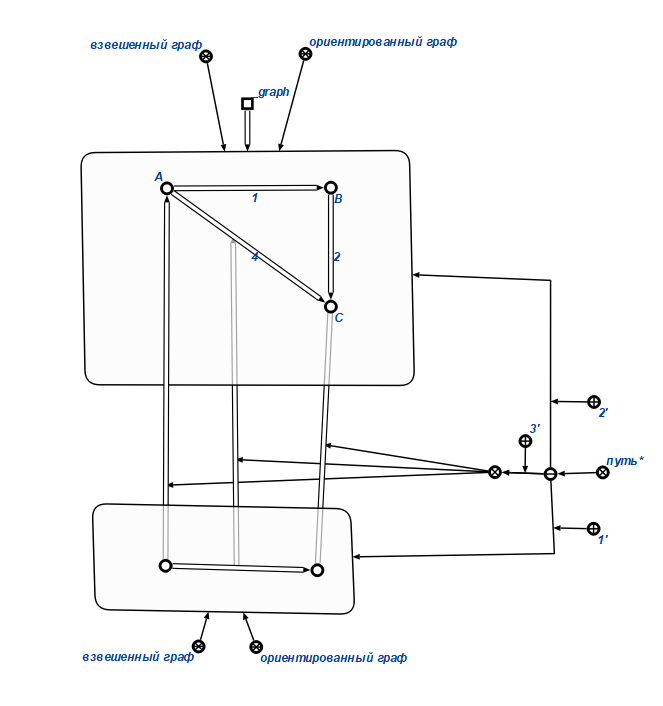


Иллюстрация пункта 9 алгоритма

# Список литературы

**OSTIS GT** [В Интернете] // База знаний по теории графов OSTIS GT. - 2011 r.. - http://ostisgraphstheo.sourceforge.net/index.php/Заглавная\_страница.

**Харарри Ф.** Теория графов [Книга]. - Москва : Едиториал УРСС, 2003.