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 технологии»

на тему

Задача поиска простых цепей указанной длины во взвешенном ориентированном графе

Выполнил Голяницкий В.А.
студент группы
021704

Проверил Витязь В.С

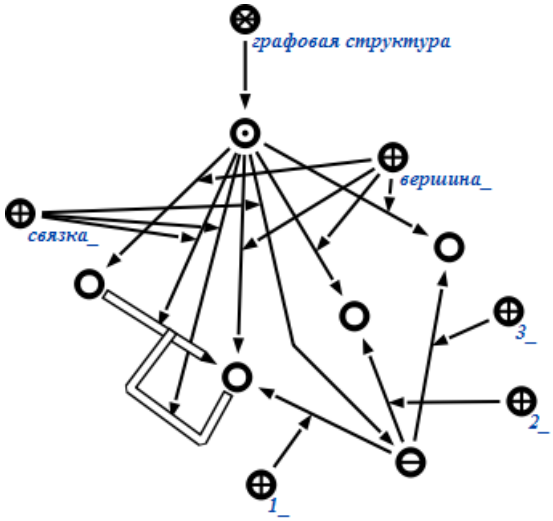
Минск 2021

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

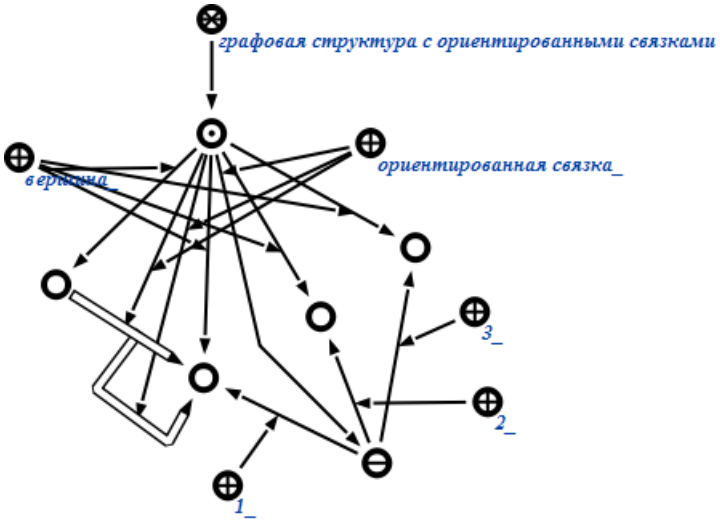
Задача: поиск простых цепей указанной длины во взвешенном ориентированном графе

1 Список понятий

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

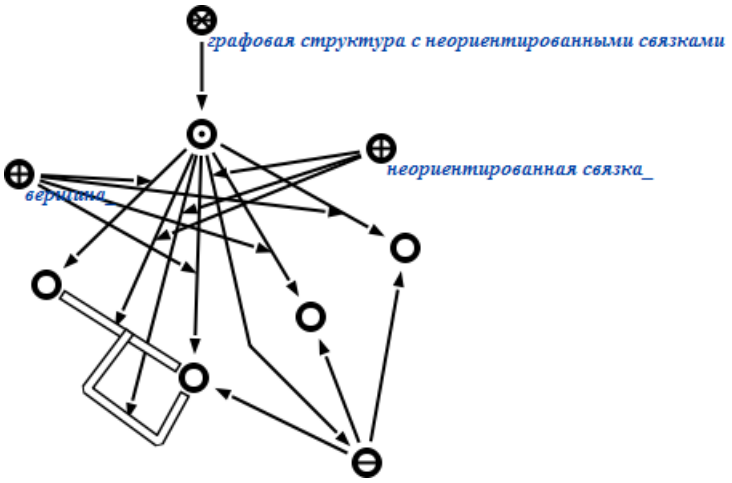


2. Графовая структура с ориентированными связками (абсолютное понятие)
    a. Ориентированная связка (относительное понятие, ролевое отношение) – связка, которая задается ориентированным множеством.

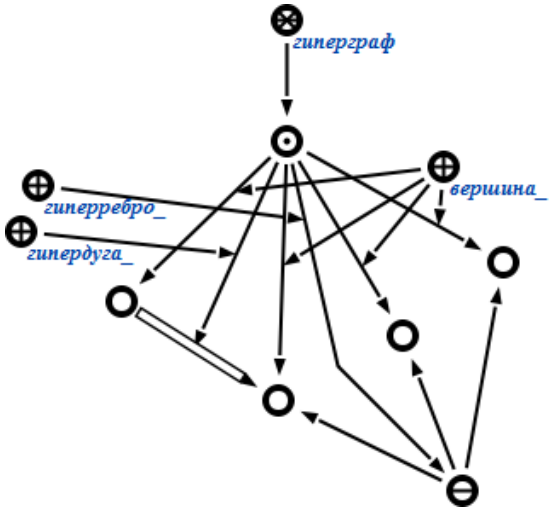


3. Графовая структура с неориентированными связками (абсолютное понятие)

a. Неориентированная связка (относительное понятие, ролевое отношение) – связка, которая задается неориентированным множеством.

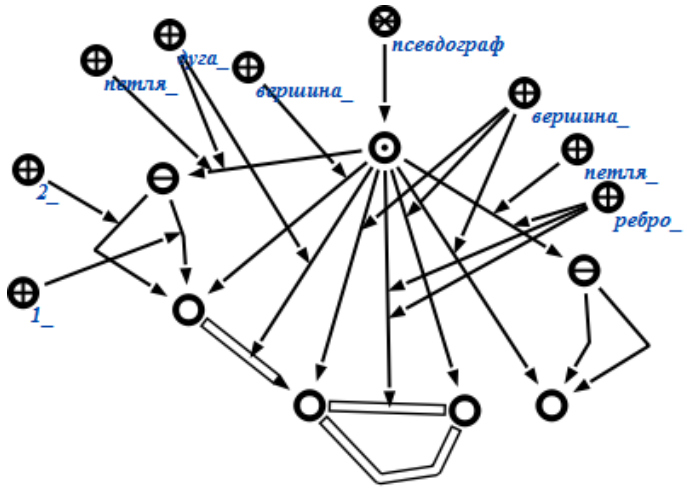


4. Гиперграф (абсолютное понятие) – это такая графовая структура, в которой связки могут связывать только вершины:
   a. Гиперсвязка (относительное понятие, ролевое отношение);
   b. Гипердуга (относительное понятие, ролевое отношение) – ориентированнаягиперсвязка;
   c. Гиперребро (относительное понятие, ролевое отношение) – неориентированнаягиперсвязка.

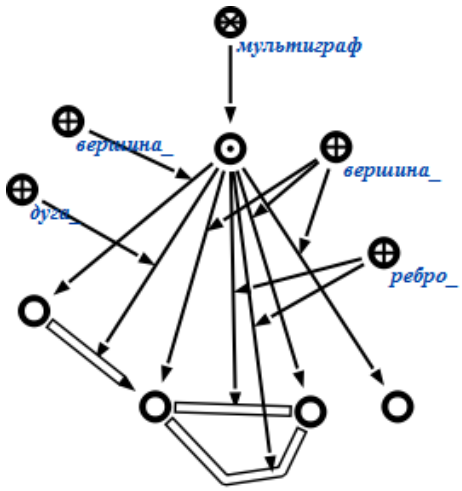


5. Псевдограф (абсолютное понятие) – это такой гиперграф, в котором все связки должны быть бинарными:
   a. Бинарная связка (относительное понятие, ролевое отношение) –гиперсвязка арности 2;
   b. Ребро (относительное понятие, ролевое отношение) – неориентированнаягиперсвязка;
   c. Дуга (относительное понятие, ролевое отношение) – ориентированная гиперсвязка;

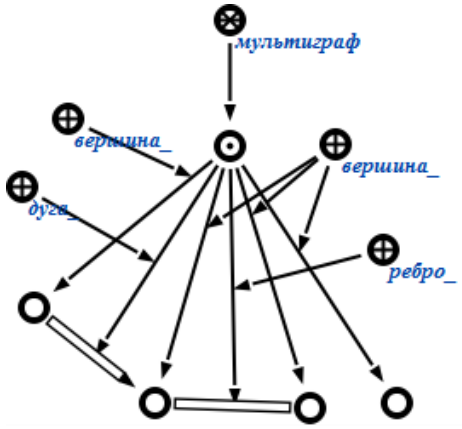
d. Петля (относительное понятие, ролевое отношение) – бинарная связка, у которой первый и второй компоненты совпадают.



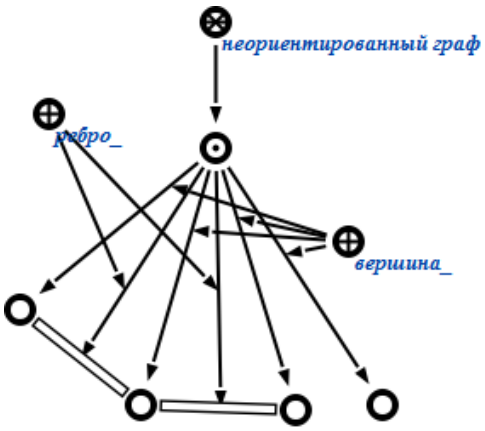
6. Мультиграф (абсолютное понятие) – это такой псевдограф, в котором не может быть петель:



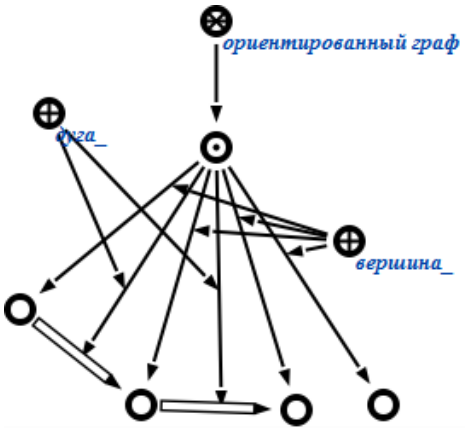
7. Граф (абсолютное понятие) – это такой мультиграф, в котором не может быть кратных связок, т.е. связок у которых первый и второй компоненты совпадают:



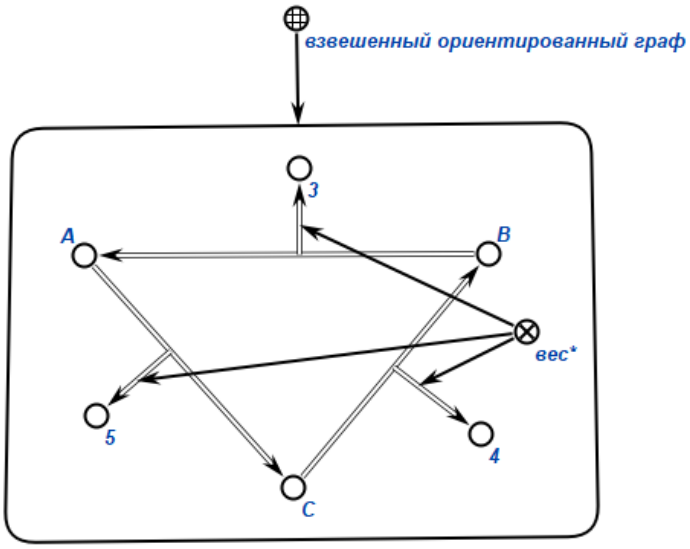
8. Неориентированный граф (абсолютное понятие) –это такой граф, в котором все связки являются ребрами:



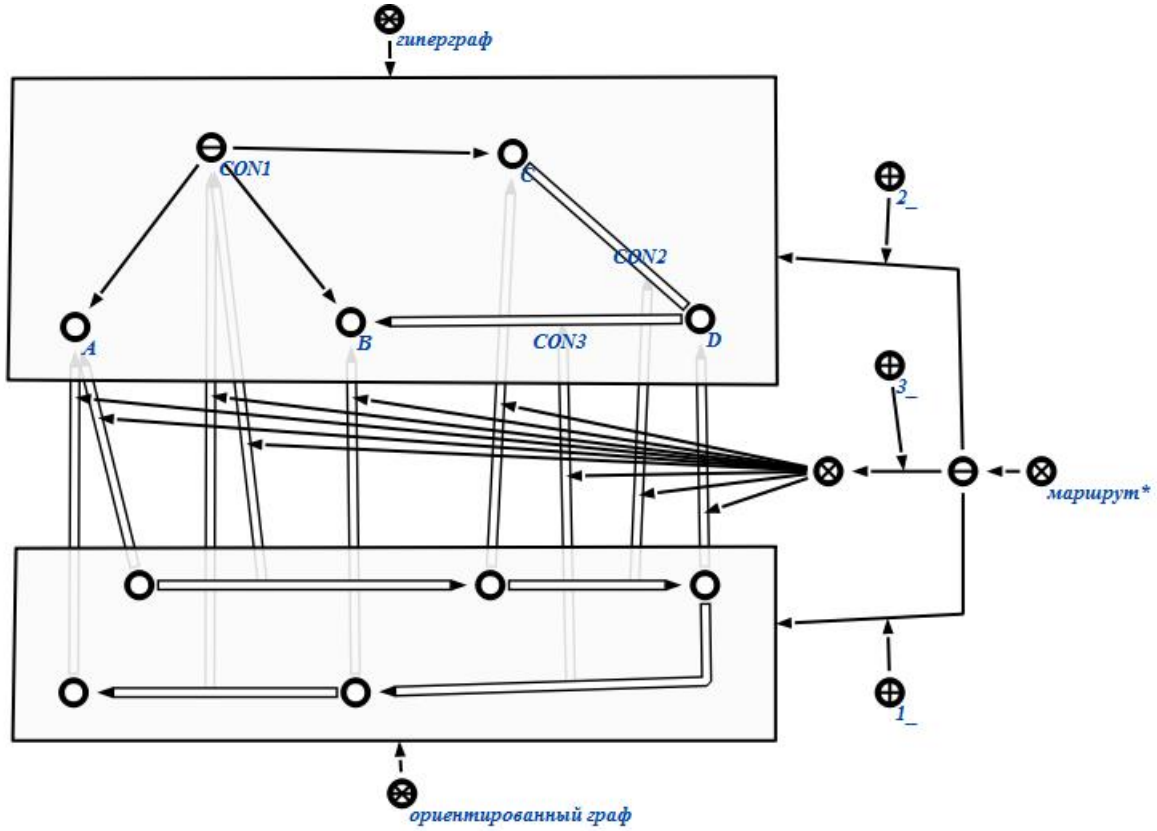
9. Ориентированный граф (абсолютное понятие) - это такой граф, в котором все связки являются дугами:



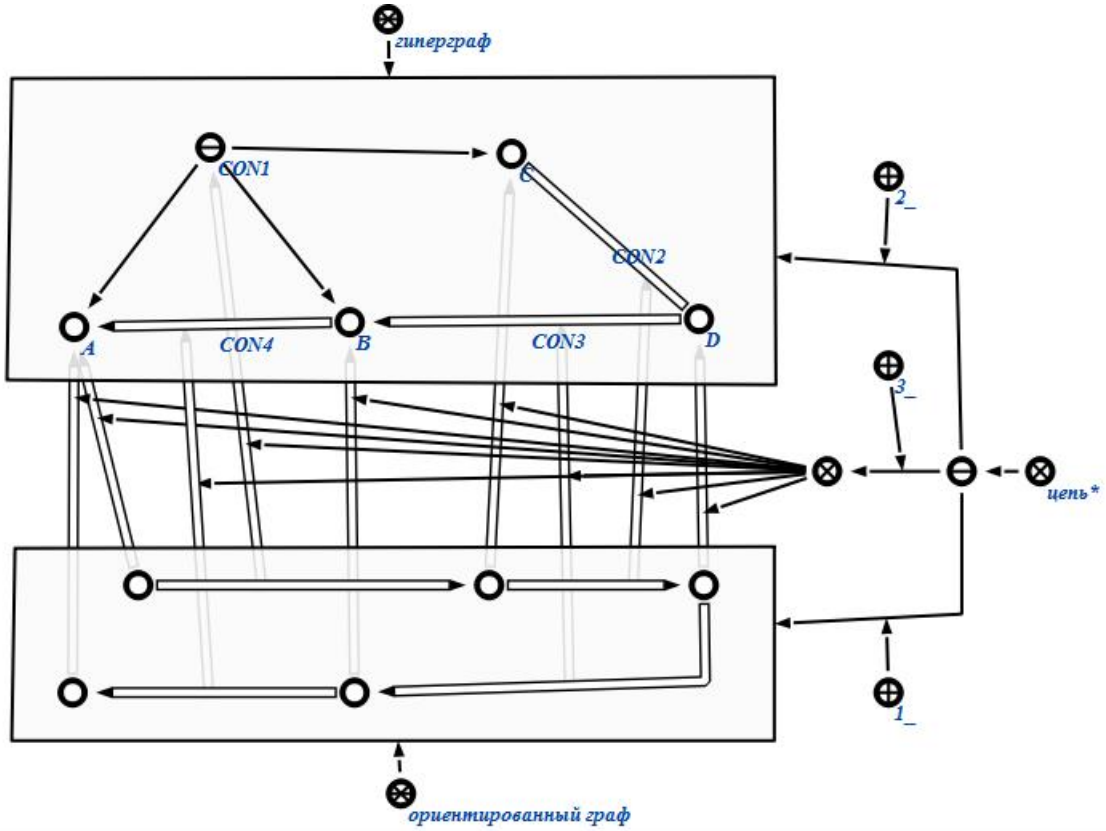
10. Взвешенный граф (абсолютное понятие) - это такой граф, в котором каждому ребру поставлено в соответствие некое значение (вес ребра):



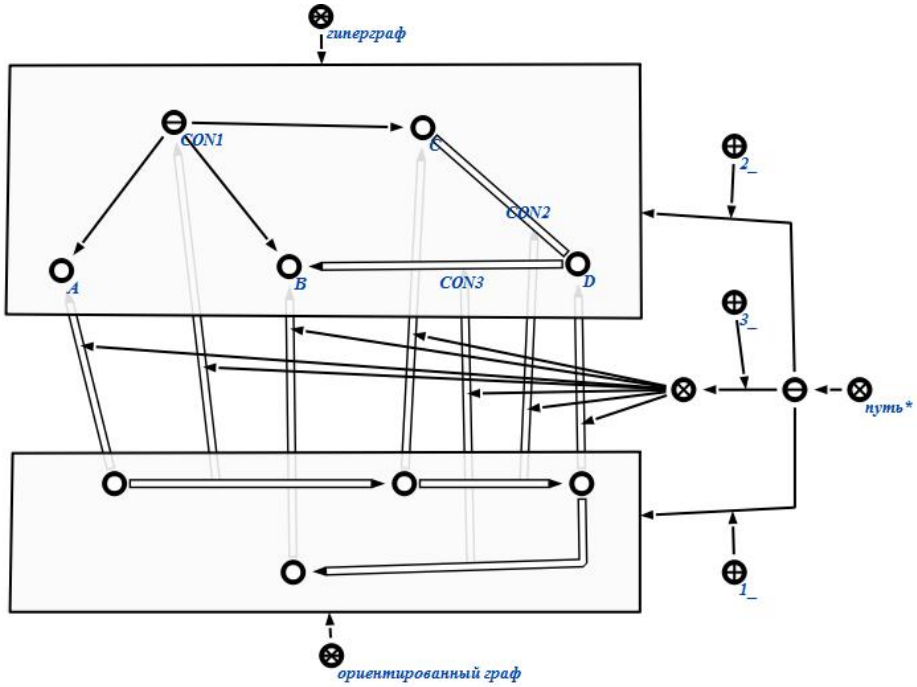
11. Маршрут (относительное понятие, бинарное ориентированное отношение) – это чередующаяся последовательность вершин и гиперсвязок в гиперграфе, которая начинается и кончается вершиной, и каждая гиперсвязка последовательности инцидентна двум вершинам, одна из которых непосредственно предшествует ей,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



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.



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.



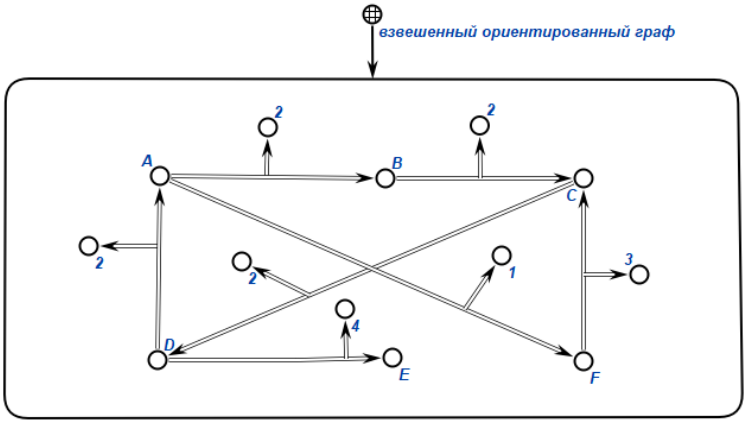
2 Тестовые примеры

Во всех тестах графы будет приведены в сокращенной форме со скрытыми ролями элементов графа.

2.1 Тест 1

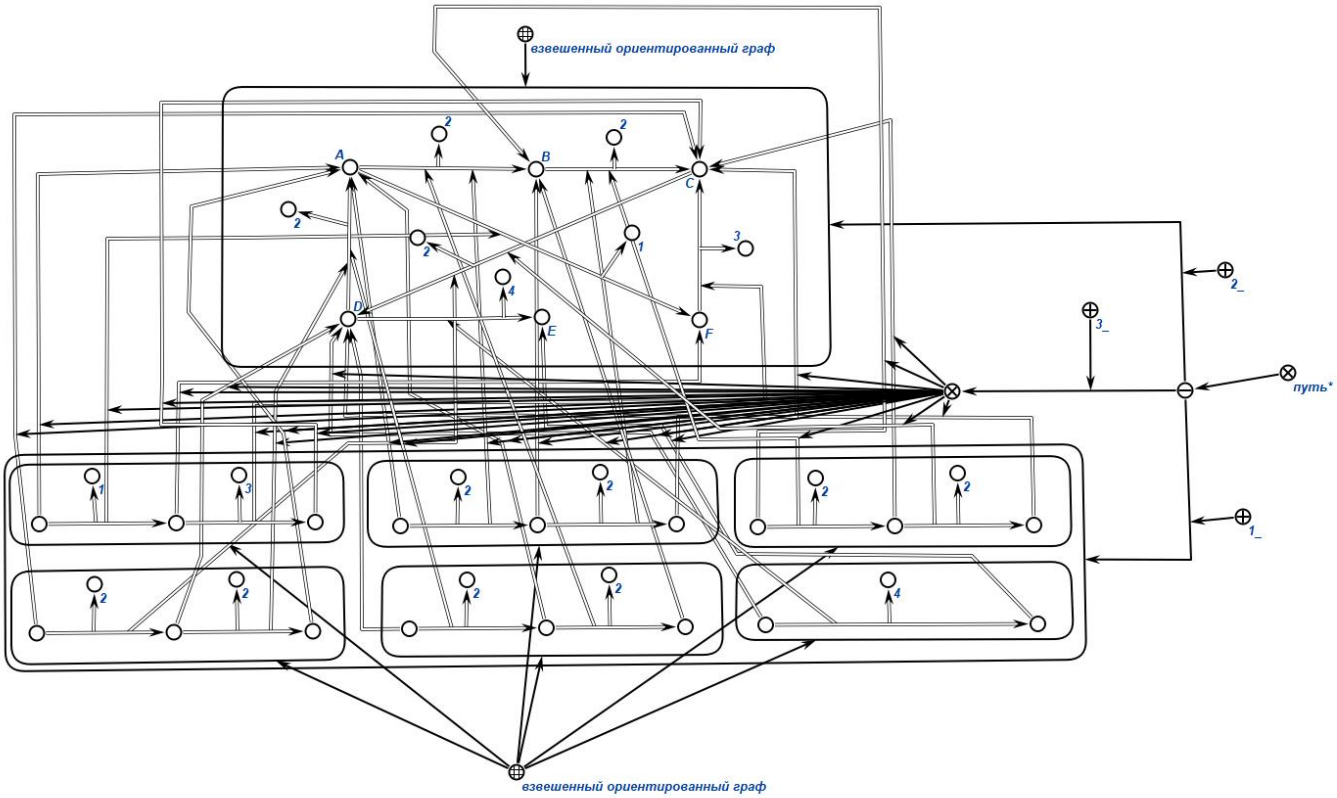
Вход:

Необходимо найти простые цепи длины 4.



Выход:

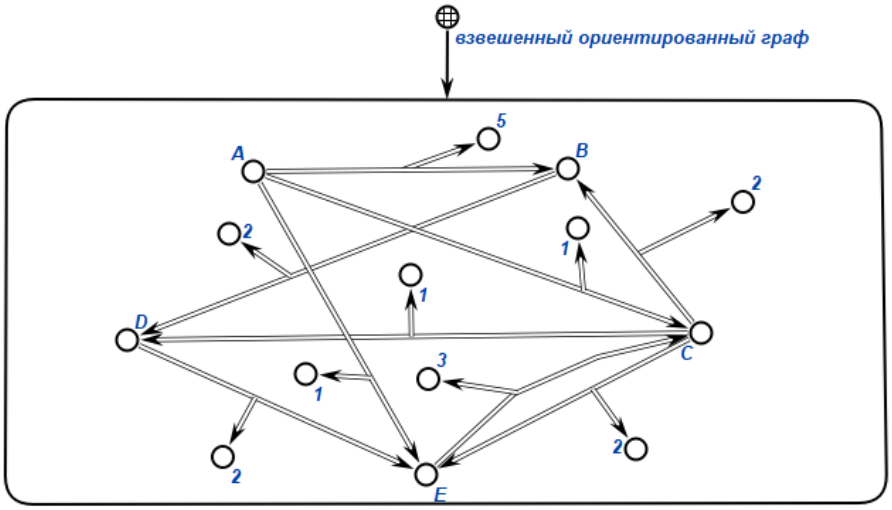
Будут найдены цепи *A,F,C; A,B,C; B,C,D; C,D,A; D,A,B; D,E*:



2.2 Тест 2

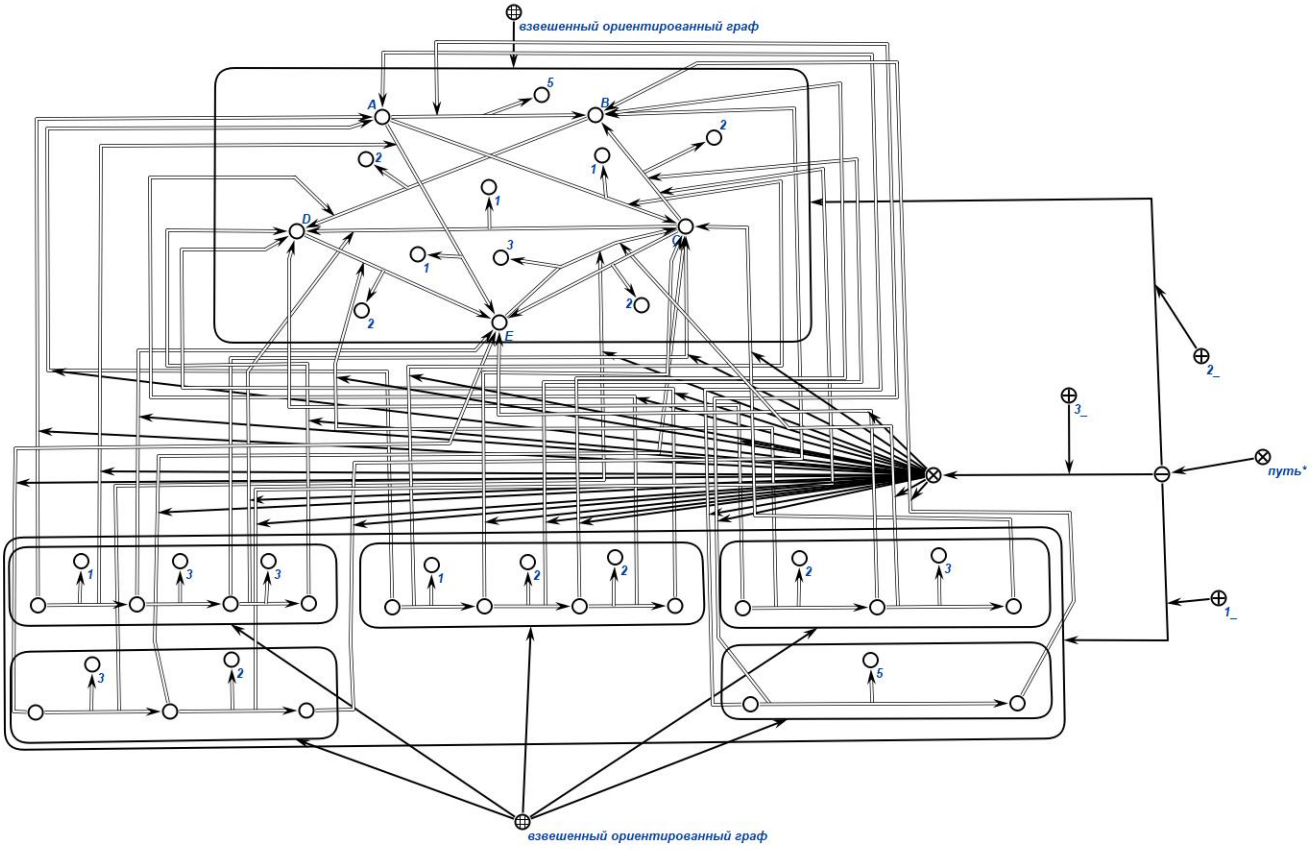
Вход:

Необходимо найти простые цепи длины 5.



Выход:

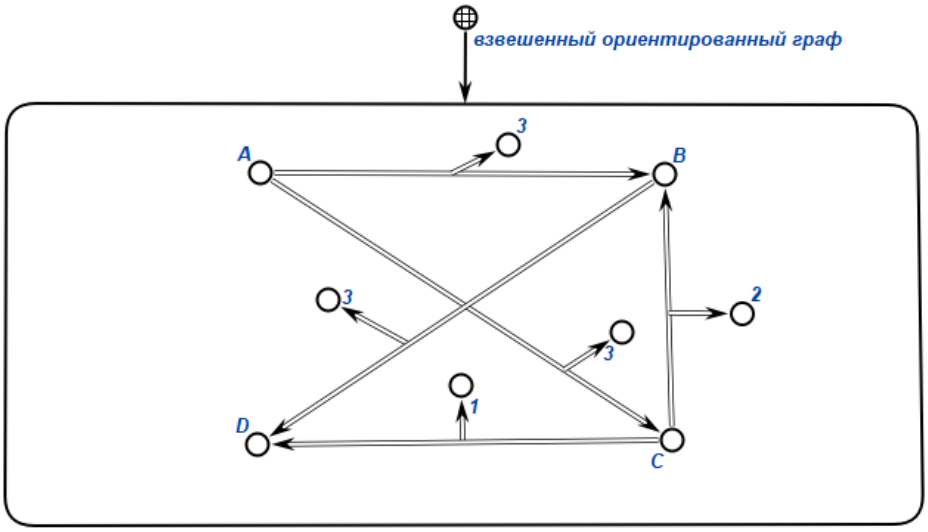
Будут найдены цепи *A,E,C,D; A,C,B,D; A,B; C,D,B; E,C,B*:



2.3 Тест 3

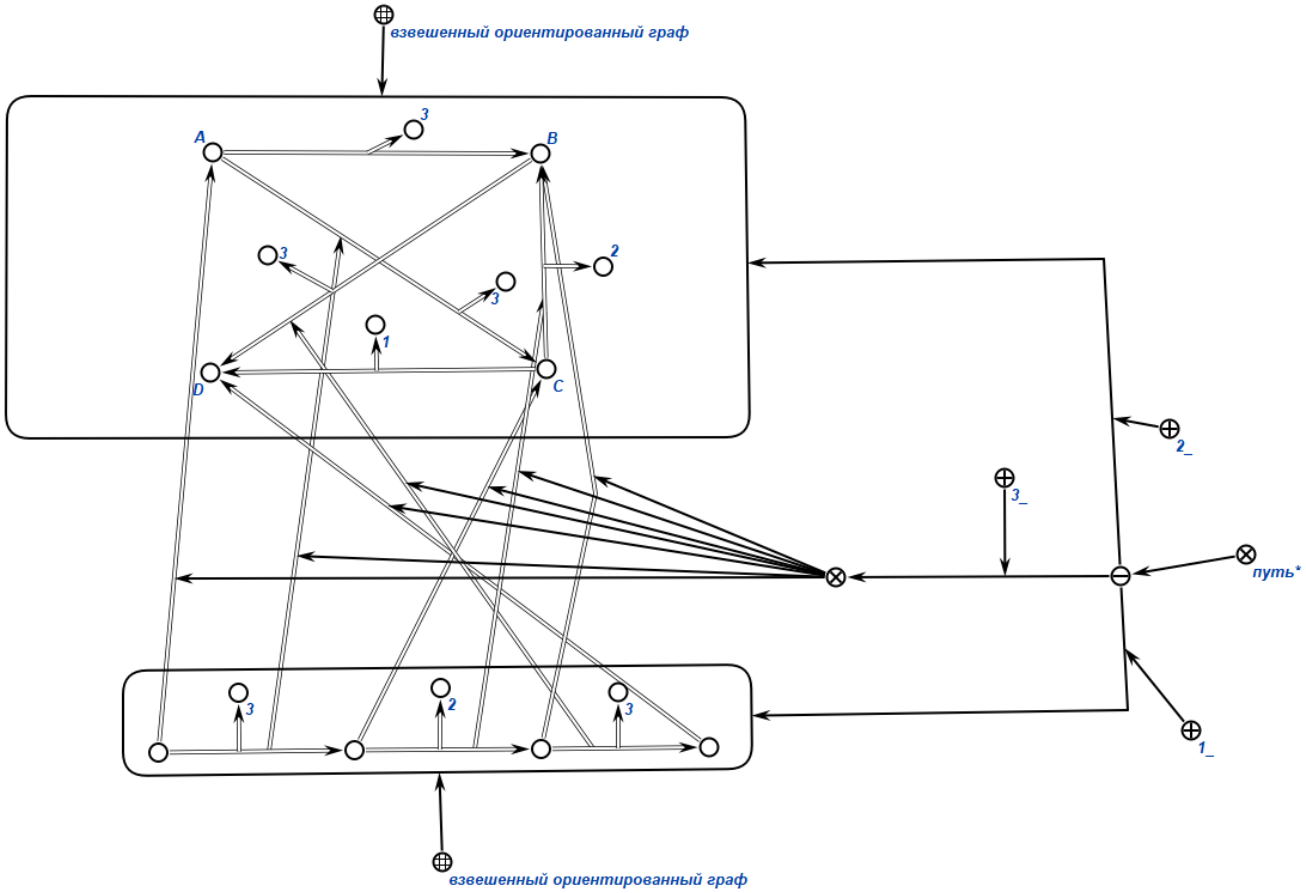
Вход:

Необходимо найти простые цепи длины 8.



Выход:

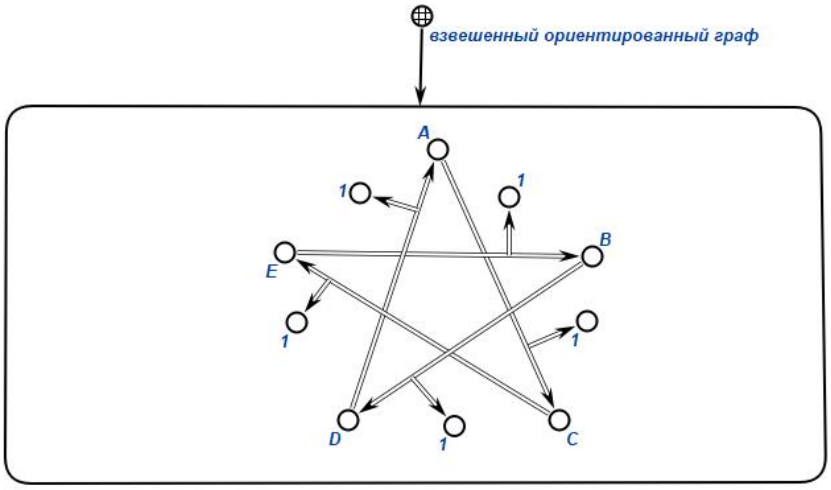
Будет найдена цепь *A,C,B,D*:



2.4 Тест 4

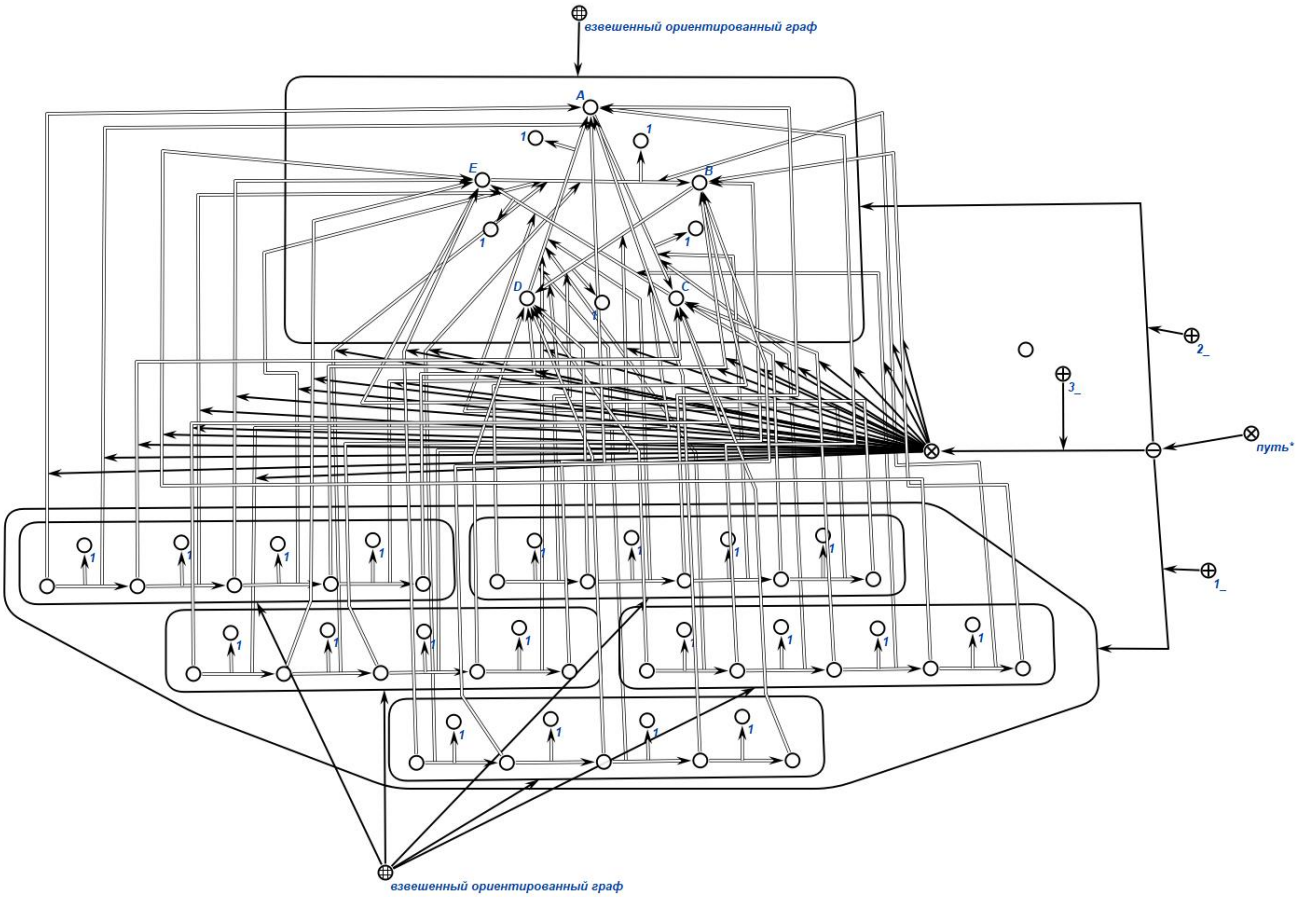
Вход:

Необходимо найти простые цепи длины 4.



Выход:

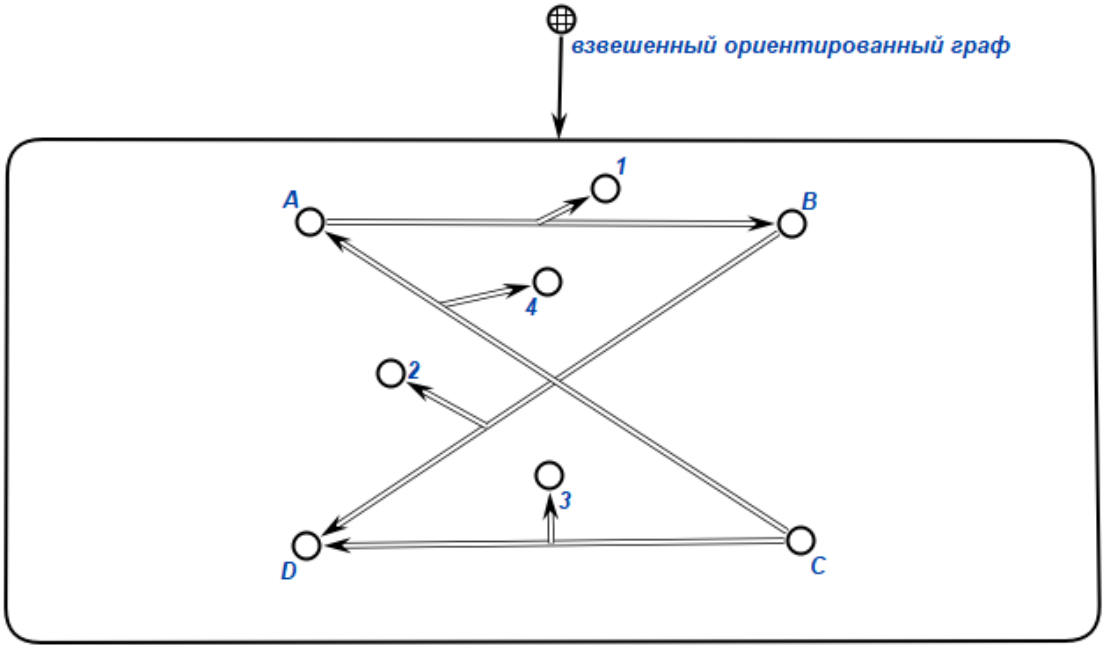
Будут найдены цепи *A,C,E,B,D; C,E,B,D,A; E,B,D,A,C; B,D,A,C,E; D,A,C,E,B* :



2.5 Тест 5

Вход:

Необходимо найти простые цепи длины 10.

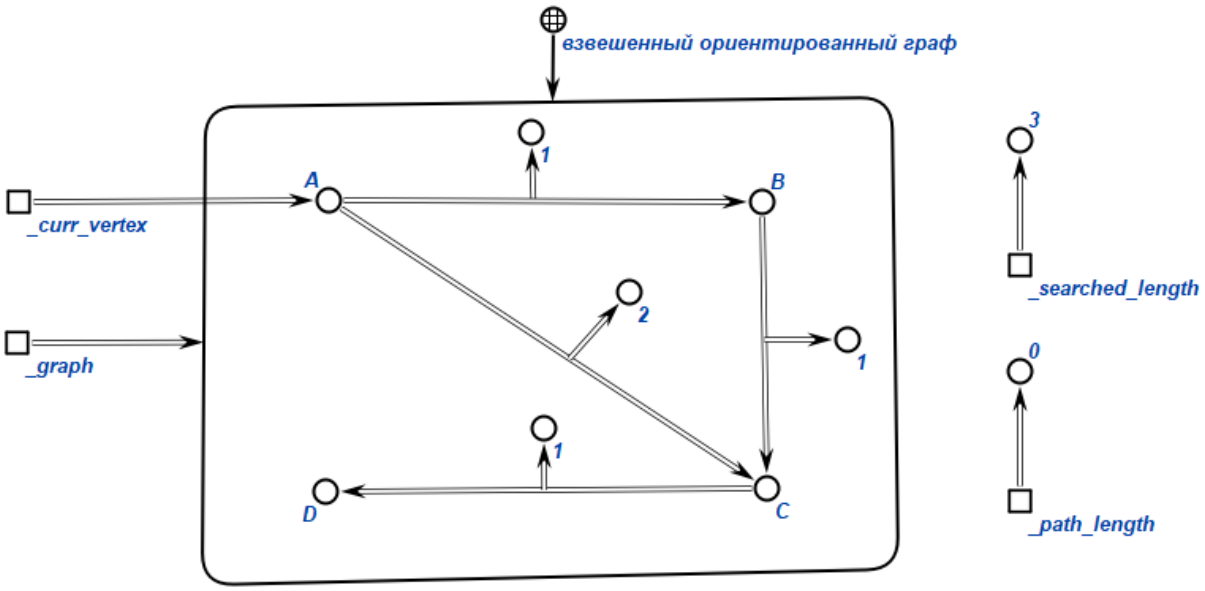


Выход:

Простой цепи длины 10 не существует. Программа должна вернуть ошибку вызывающему контексту.

3 Описание алгоритма

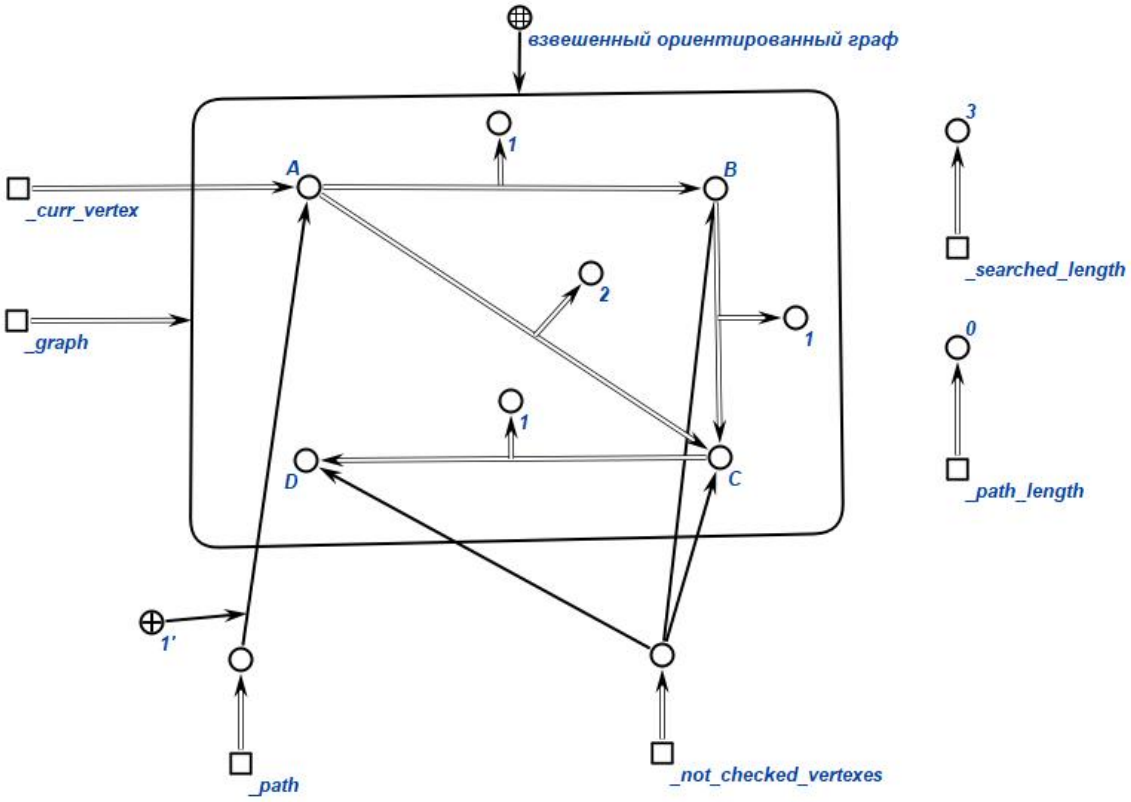
1. Задание входного графа, текущей рассматриваемой вершины пути, переменных для хранения длины искомого пути и длины текущего построенного пути для работы алгоритма



Переменные изменятся следующим образом:

- _graph получит в качестве значения sc-узел ориентированного взвешенного графа;
- _curr_vertex получит в качестве значения вершину A, которая является текущей в поиске пути заданной длины;
- _searched_length получит в качестве значение 3, что означает длину искомого пути
- _path_length получит в качестве значение 0, что означает длину текущего построенного пути. При добавлении в текущий путь новых вершин будем изменять значение _path_length и сравнивать его с _searched_length

2. Создание ориентированного множества пути и создание множества неиспользованных вершин



Переменная _path получит в качестве значения будущее ориентированное множество вершин в порядке следования в текущем пути. Будем использовать его для проверки, используется ли уже проверяемая вершина, и для хранения текущего исследуемого пути.

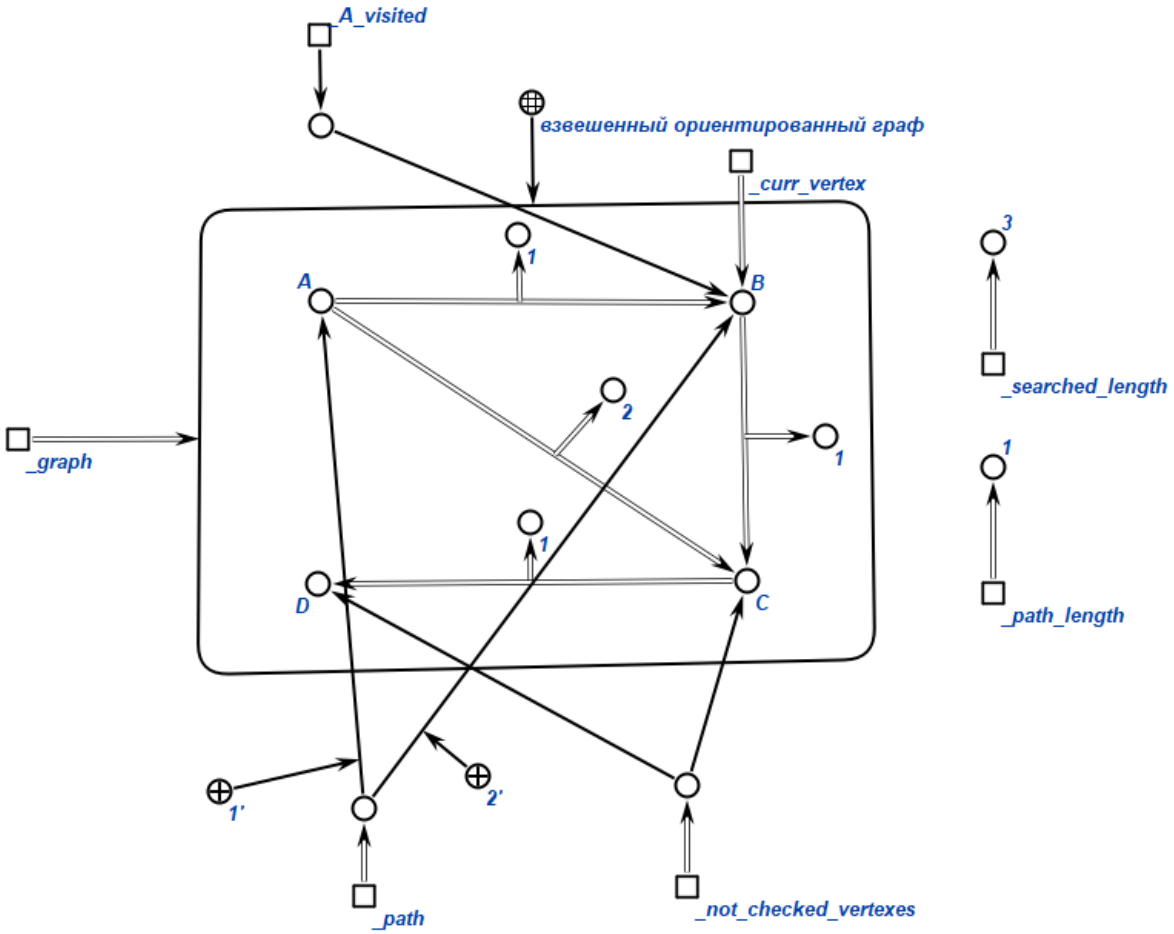
Добавляем первую вершину: вершину A. Указываем ее порядок следования во множестве _path.

Переменная _path_length получает текущую длину пути. Так как путь сейчас состоит из одной начальной вершины, значение длины равно нулю.

Переменная _not_checked_vertexes получит в качестве значения множество непроверенных вершин обрабатываемого графа (в это множество не включена начальная вершина пути A). Мы не можем использовать одинаковые вершины несколько раз.

По множеству неиспользованных вершин проверяем, с какими вершинами имеет связи вершина A. Это вершины B и C.

3. Добавление во множество используемых вершин вершины B



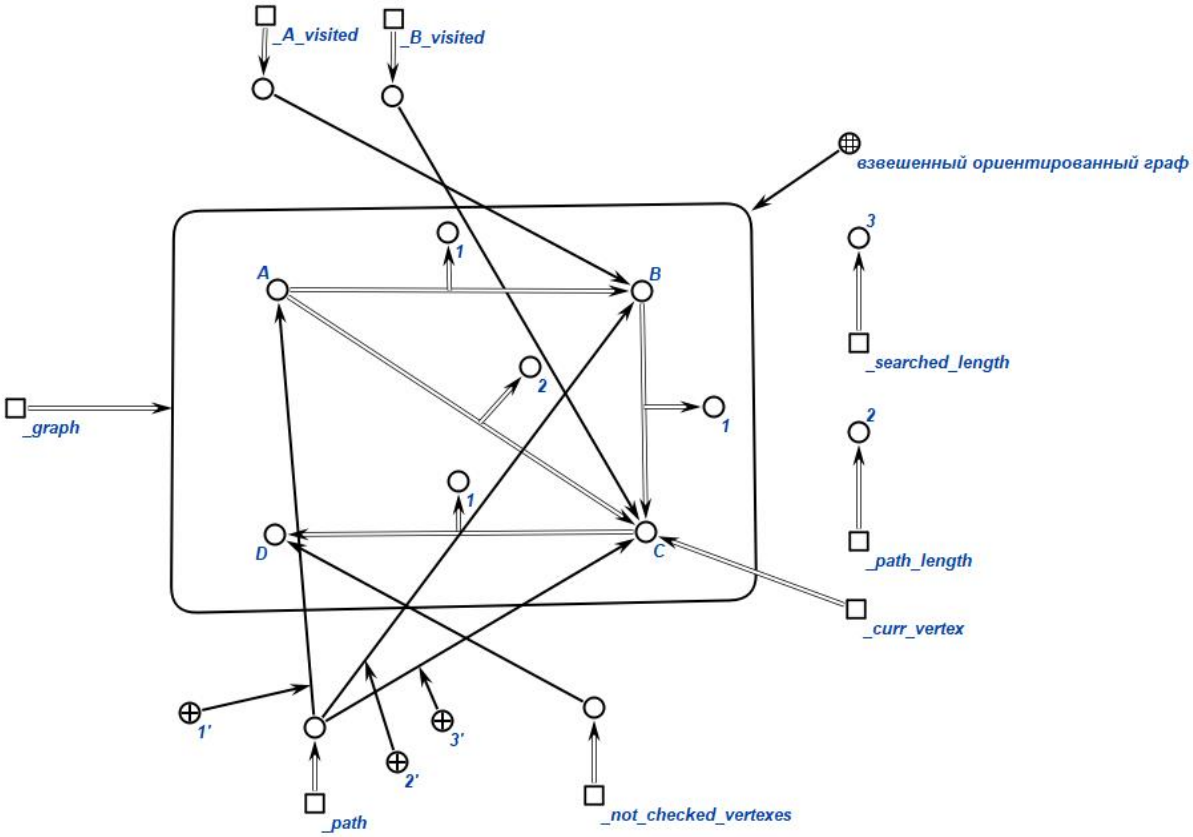
Из двух вершин, с которыми имеет связи выбираем первую - B. Добавляем ее во множество пути с порядковым номером 2. Создаем множество посещенных из вершины A вершин и добавляем туда вершину B. Будем использовать это множество для проверки, какие вершины из текущей мы уже посещали, чтобы при возвращении к вершине не начинать проверку того же самого пути. Значение переменной _curr_vertex изменим на B.

Длину пути изменяем на значение веса ребра AB. Длина текущего пути – 1. 1<3, поэтому продолжаем построение пути.

Удаляем из множества неиспользованных вершин вершину B.

Вершина B имеет связь с вершиной C.

4. Добавление во множество используемых вершин вершины C



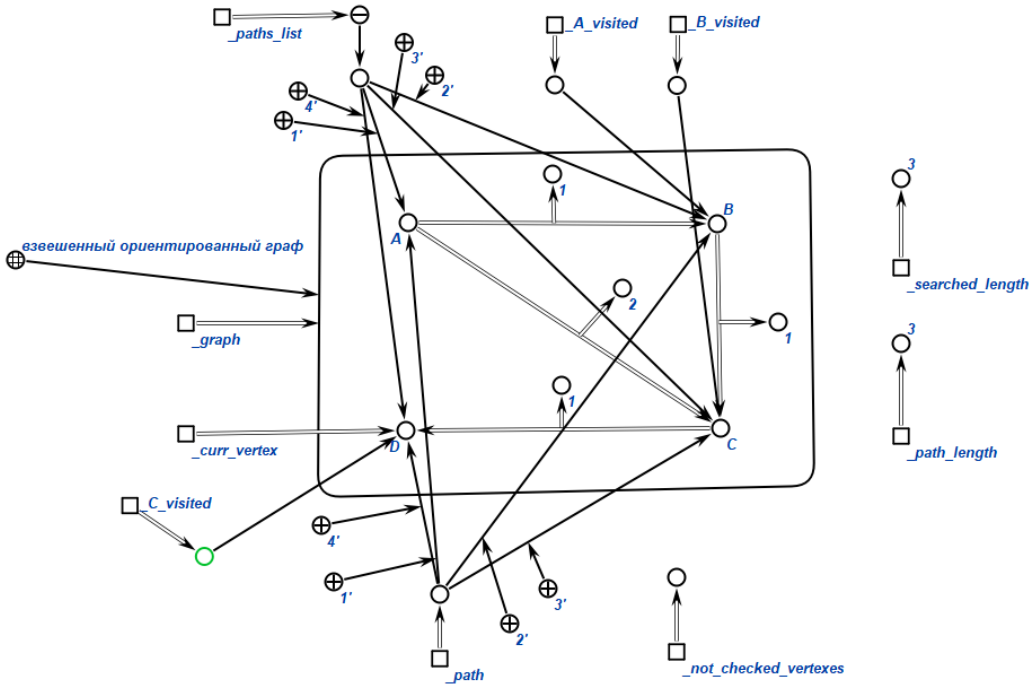
Добавляем вершину C во множество пути с порядковым номером 3. Создаем множество посещенных из вершины B вершин и добавляем туда вершину C. Значение переменной _curr_vertex изменим на C.

Удаляем из множества неиспользованных вершин вершину C.

Изменяем длину текущего пути на значение веса ребра BC. Длина текущего пути - 2. 2<3, поэтому продолжаем построение пути.

Вершина C имеет связь с единственной вершиной: D.

5. Добавление во множество используемых вершин вершины D и создание ориентированного множества полученного пути искомой длины



Добавляем вершину D во множество пути с порядковым номером 4. Создаем множество посещенных из вершины C вершин и добавляем туда вершину D. Значение переменной _curr_vertex изменим на D.

Удаляем из множества неиспользованных вершин вершину D.

Изменяем длину текущего пути на значение веса ребра CD. Длина текущего пути - 3. 3=3, поэтому построение текущего пути искомой длины считаем завершенным.

Мы пришли в финальную вершину, так как она не является начальной ни для одного ребра и множество неиспользованных вершин пусто.

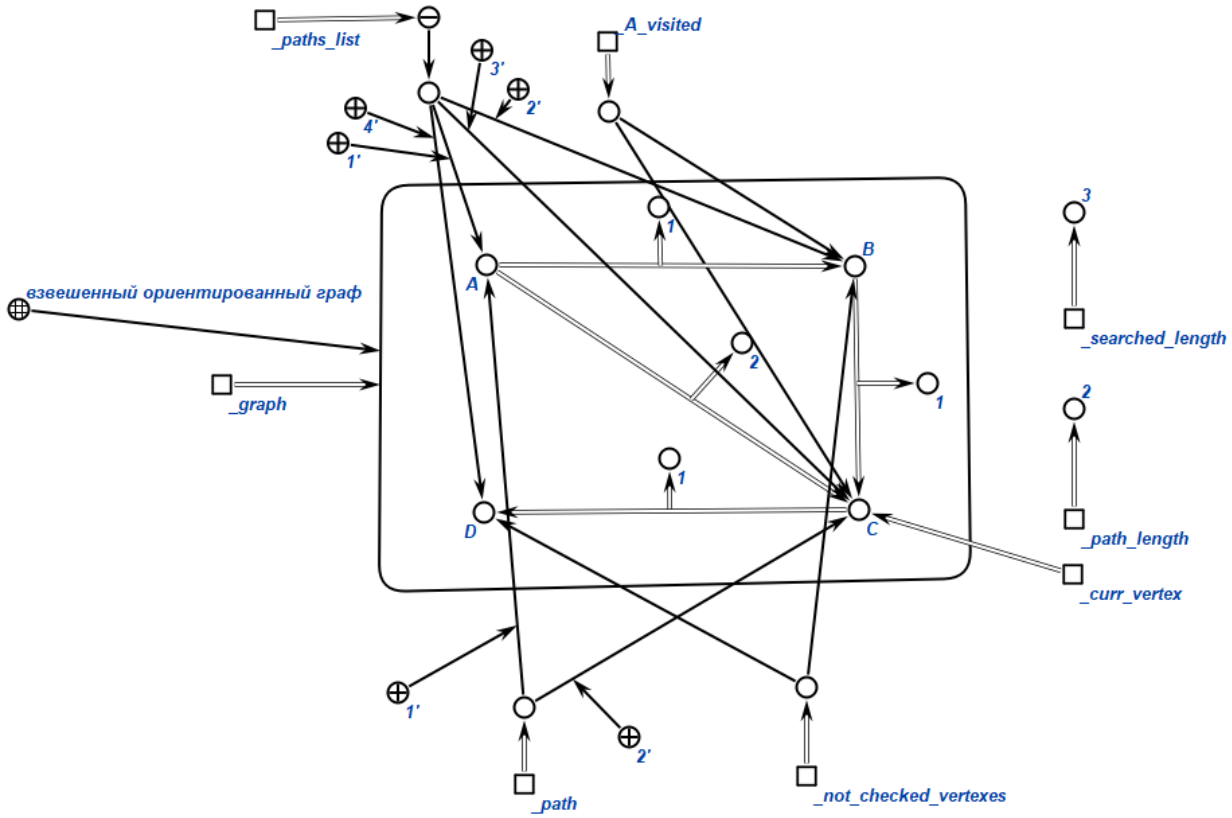
Переносим все значения ориентированного множества _path во множество, хранящее уже готовый путь искомой длины и являющееся элементом списка таких множеств _paths_list, с сохранением порядка следования элементов.

Удаляем вершину D из множества пути и добавляем во множество неиспользованных вершин. Уменьшаем длину текущего пути на 1. Не удаляем вершину D из _C_visited.

Вершина C не имеет других ребер, кроме как с вершиной D. Удаляем вершину C из множества пути и добавляем во множество неиспользованных вершин. Уменьшаем длину текущего пути на 1. Не удаляем вершину C из _B_visited. Удаляем _C_visited.

Вершина B не имеет других ребер, кроме как с вершиной C. Удаляем вершину B из множества пути и добавляем во множество неиспользованных вершин. Уменьшаем длину текущего пути на 1. Не удаляем вершину B из _A_visited. Удаляем _B_visited. Вершина A имеет еще одну неисследованную связь с вершиной C.

6. Добавление во множество используемых вершин вершины C



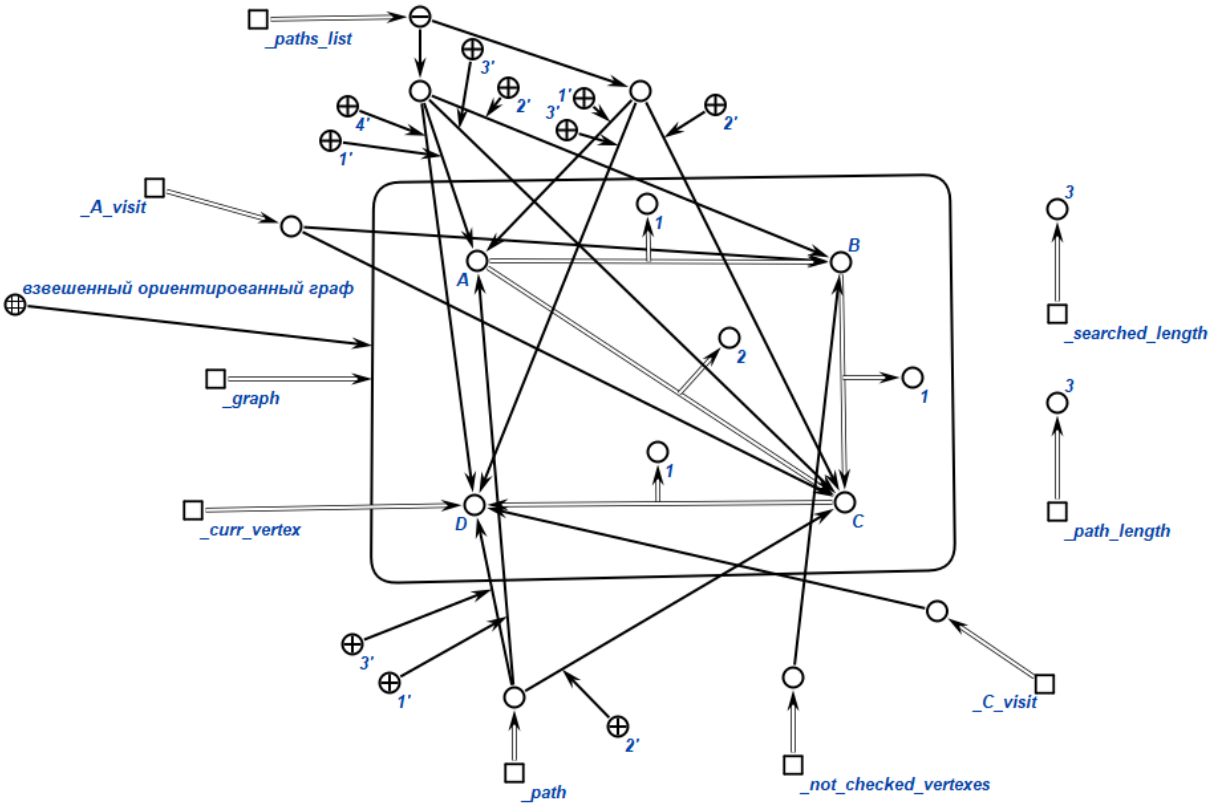
Добавляем вершину C во множество пути с порядковым номером 2. Добавляем C в _A_visited. Значение переменной _curr_vertex изменим на C.

Удаляем из множества неиспользованных вершин вершину C.

Изменяем длину текущего пути на значение веса ребра AC. Длина текущего пути - 2. 2<3, поэтому продолжаем построение пути.

Вершина C имеет связь с единственной вершиной: D.

7. Добавление во множество используемых вершин вершины D



Добавляем вершину D во множество пути с порядковым номером 3. Создаем множество посещенных из вершины C вершин и добавляем туда вершину D. Значение переменной _curr_vertex изменим на D.

Удаляем из множества неиспользованных вершин вершину D.

Изменяем длину текущего пути на значение веса ребра CD. Длина текущего пути - 3. 3=3, поэтому построение текущего пути искомой длины считаем заершенным.

Мы пришли в финальную вершину, так как она не является начальной ни для одного ребра и множество неиспользованных вершин пусто.

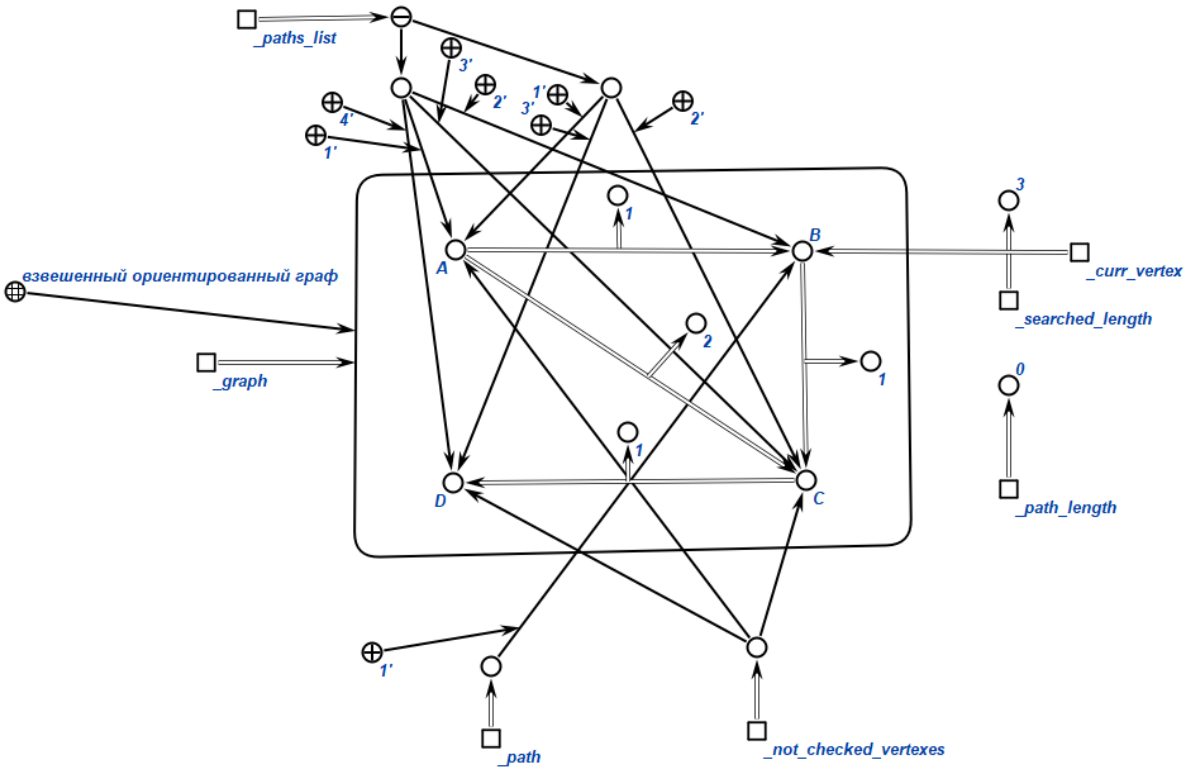
Переносим все значения ориентированного множества _path во множество, хранящее уже готовый путь искомой длины и являющееся элементом списка таких множеств _paths_list, с сохранением порядка следования элементов.

Удаляем вершину D из множества пути и добавляем во множество неиспользованных вершин. Уменьшаем длину текущего пути на 1. Не удаляем вершину D из _C_visited.

Вершина C не имеет других ребер, кроме как с вершиной D. Удаляем вершину C из множества пути и добавляем во множество неиспользованных вершин. Уменьшаем длину текущего пути на 2. Не удаляем вершину C из _A_visited. Удаляем _C_visited.

Вершина A не имеет других ребер, кроме как с вершинами C и B. Удаляем вершину A из множества пути и добавляем во множество неиспользованных вершин. Удаляем _A_visited.

8. Добавление во множество используемых вершин вершины B и построение пути из данной вершины



Начинаем построение путей искомой длины со следующей вершины - B. Добавляем ее во множество пути с порядковым номером 1. Длина текущего пути, состоящего из единственной вершины, – 0. Удаляем из множества неиспользованных вершин вершину B. Вершина B имеет связь с вершиной C.

Добавляем вершину C во множество пути с порядковым номером 2. Создаем множество посещенных из вершины B вершин и добавляем туда вершину C. Значение переменной _curr_vertex изменим на C. Удаляем из множества неиспользованных вершин вершину C.

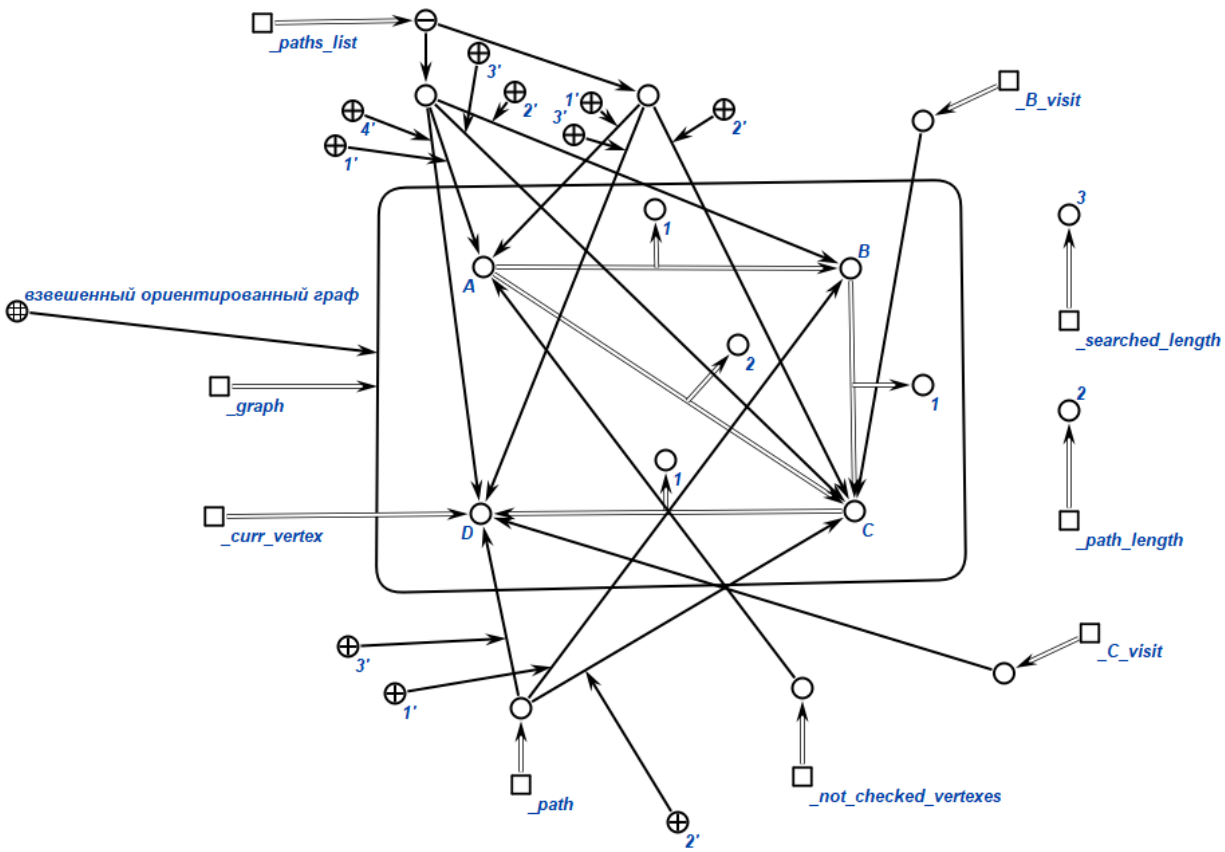
Изменяем длину текущего пути на значение веса ребра BC. Длина текущего пути - 1. 1<3, поэтому продолжаем построение пути.

Вершина C имеет связь с единственной вершиной: D.

Добавляем вершину D во множество пути с порядковым номером 3. Создаем множество посещенных из вершины C вершин и добавляем туда вершину D. Значение переменной _curr_vertex изменим на D. Удаляем из множества неиспользованных вершин вершину D.

Изменяем длину текущего пути на значение веса ребра CD. Длина текущего пути - 2.

9. Добавление в структуру пути посещения начальной и конечной вершины генерируемого пути и связывающего ребра



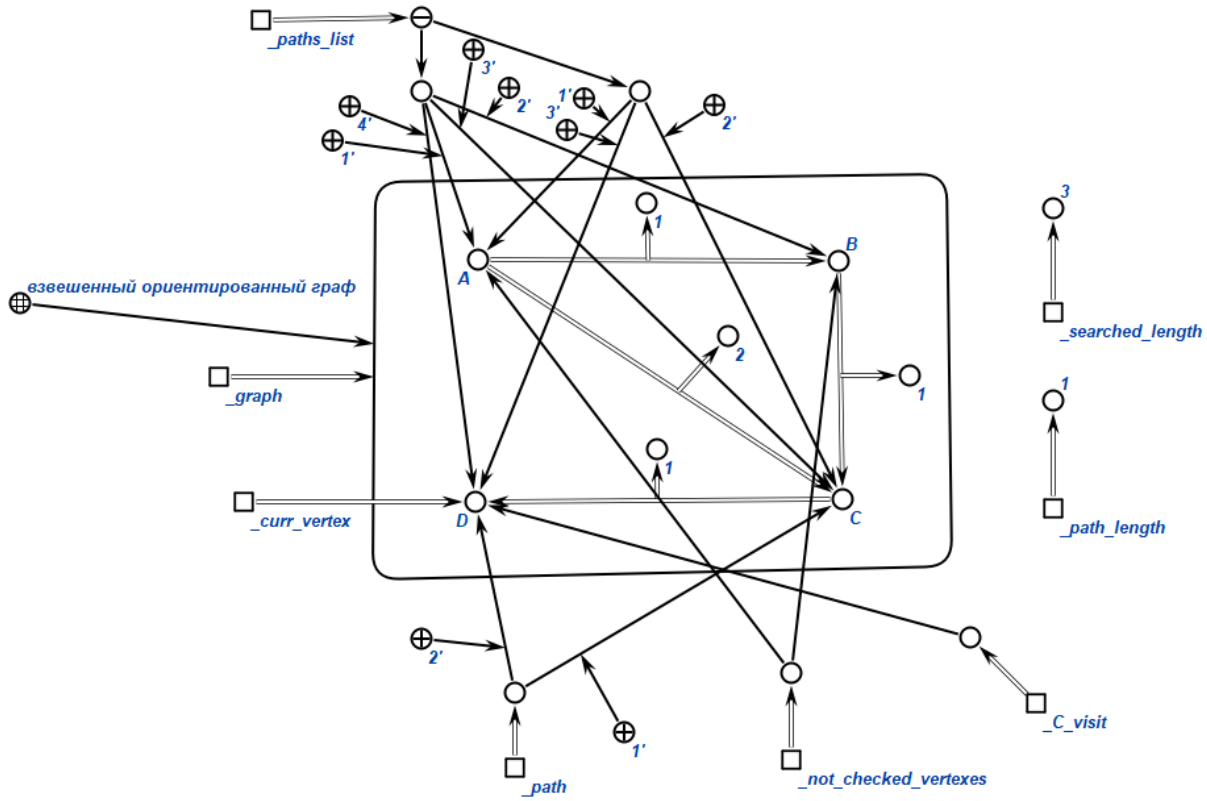
Мы пришли в финальную вершину, так как она не является начальной ни для одного ребра. Продолжать построение пути не является возможным. Текущая длина пути 2, что не является искомой длиной.

Удаляем вершину D из множества пути и добавляем во множество неиспользованных вершин. Уменьшаем длину текущего пути на 1. Не удаляем вершину D из _C_visited.

Вершина C не имеет других ребер, кроме как с вершиной D. Удаляем вершину C из множества пути и добавляем во множество неиспользованных вершин. Уменьшаем длину текущего пути на 1. Не удаляем вершину C из _B_visited. Удаляем _C_visited.

Вершина B не имеет других ребер, кроме как с вершиной C. Удаляем вершину B из множества пути и добавляем во множество неиспользованных вершин. Удаляем _B_visited.

10. Добавление во множество используемых вершин вершины C и построение пути из данной вершины

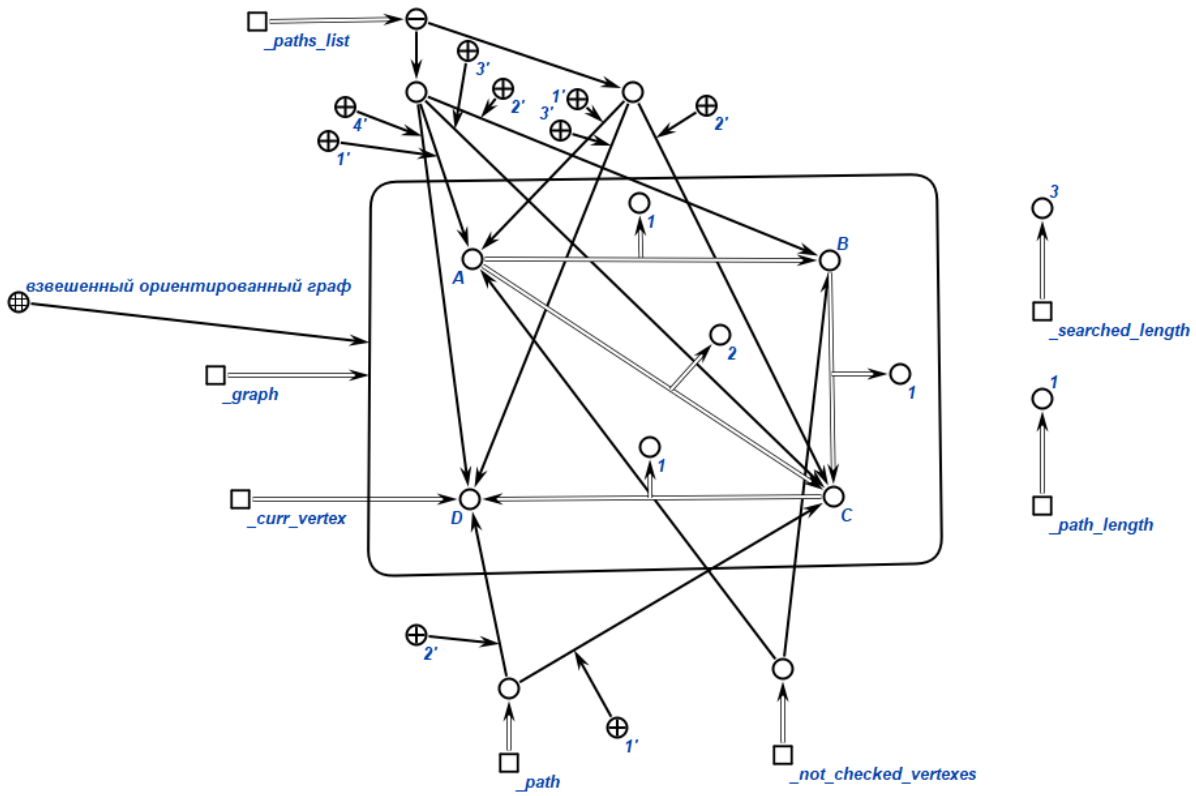


Начинаем построение путей искомой длины со следующей вершины - C. Добавляем ее во множество пути с порядковым номером 1. Значение переменной _curr_vertex изменим на C. Длина текущего пути, состоящего из единственной вершины, – 0. Удаляем из множества неиспользованных вершин вершину C.

Вершина C имеет связь с единственной вершиной: D.

Добавляем вершину D во множество пути с порядковым номером 2. Создаем множество посещенных из вершины C вершин и добавляем туда вершину D. Значение переменной _curr_vertex изменим на D. Удаляем из множества неиспользованных вершин вершину D. Изменяем длину текущего пути на значение веса ребра CD. Длина текущего пути - 1.

11. Добавление в структуру пути посещения начальной и конечной вершины генерируемого пути и связывающего ребра

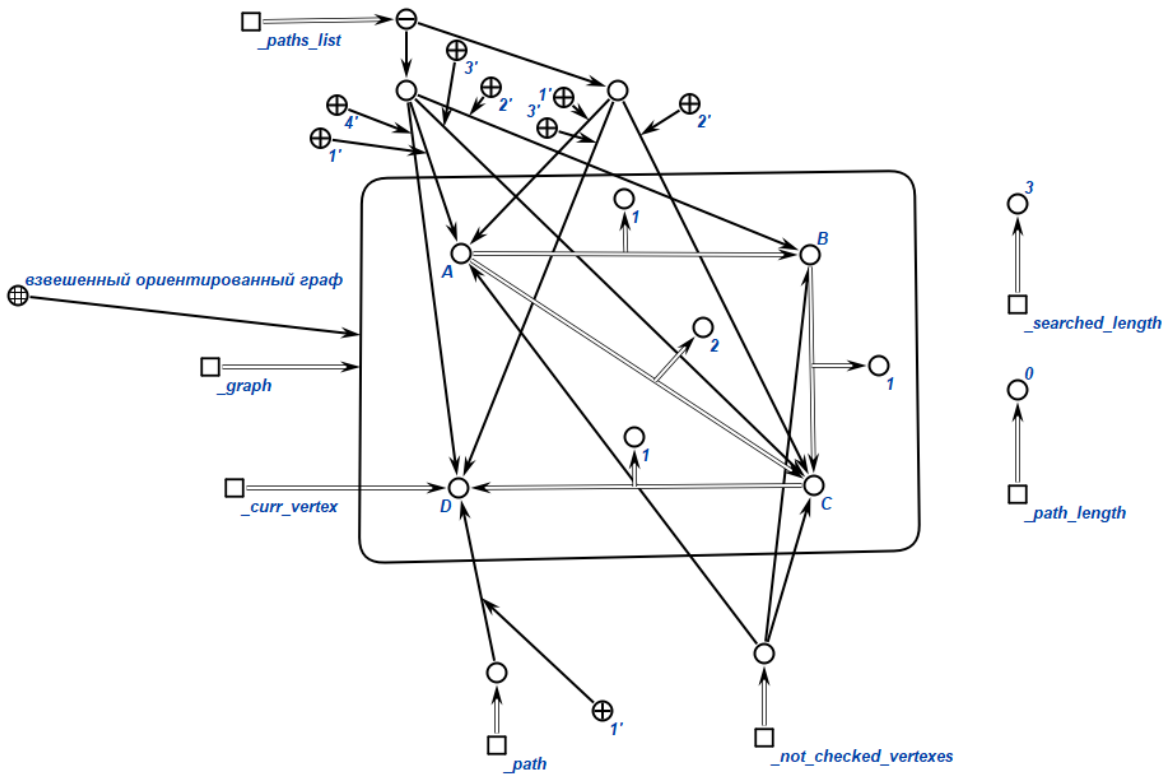


Мы пришли в финальную вершину, так как она не является начальной ни для одного ребра. Продолжать построение пути не является возможным. Текущая длина пути 1, что не является искомой длиной.

Удаляем вершину D из множества пути и добавляем во множество неиспользованных вершин. Уменьшаем длину текущего пути на 1. Не удаляем вершину D из _C_visited.

Вершина C не имеет других ребер, кроме как с вершиной D. Удаляем вершину C из множества пути и добавляем во множество неиспользованных вершин. Удаляем _C_visited.

12. Добавление во множество используемых вершин вершины D и построение пути из данной вершины



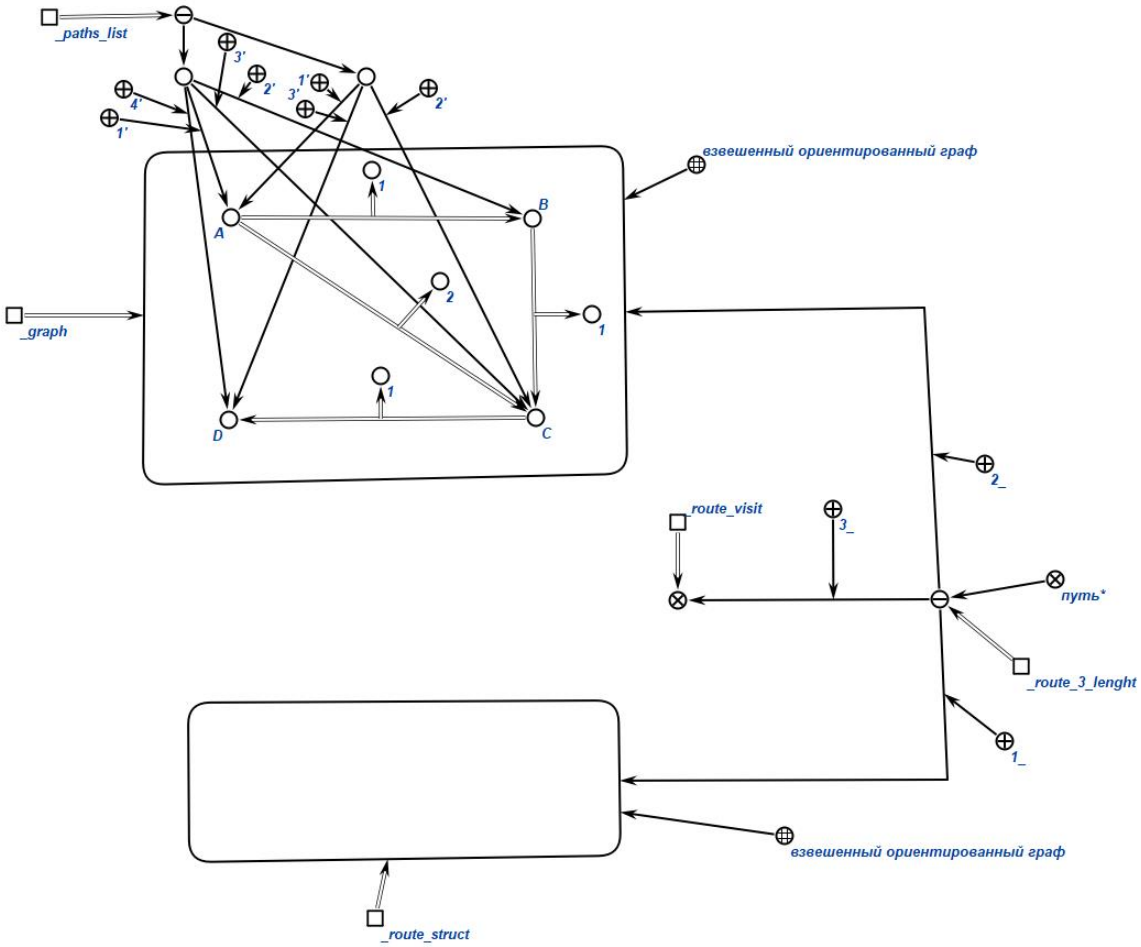
Начинаем построение путей искомой длины со следующей вершины - D. Добавляем ее во множество пути с порядковым номером 1. Значение переменной _curr_vertex изменим на D. Длина текущего пути, состоящего из единственной вершины, – 0. Удаляем из множества неиспользованных вершин вершину D.

Мы пришли в финальную вершину, так как она не является начальной ни для одного ребра. Продолжать построение пути не является возможным. Текущая длина пути 0, что не является искомой длиной.

Построение путей искомой длины завершено, так как мы рассмотрели все вершины графа в качестве начальных.

Множество текущего пути, множество неиспользованных вершин, переменные, хранящие длины пути, переменную текущей рассматриваемой вершины удаляем за ненадобностью.

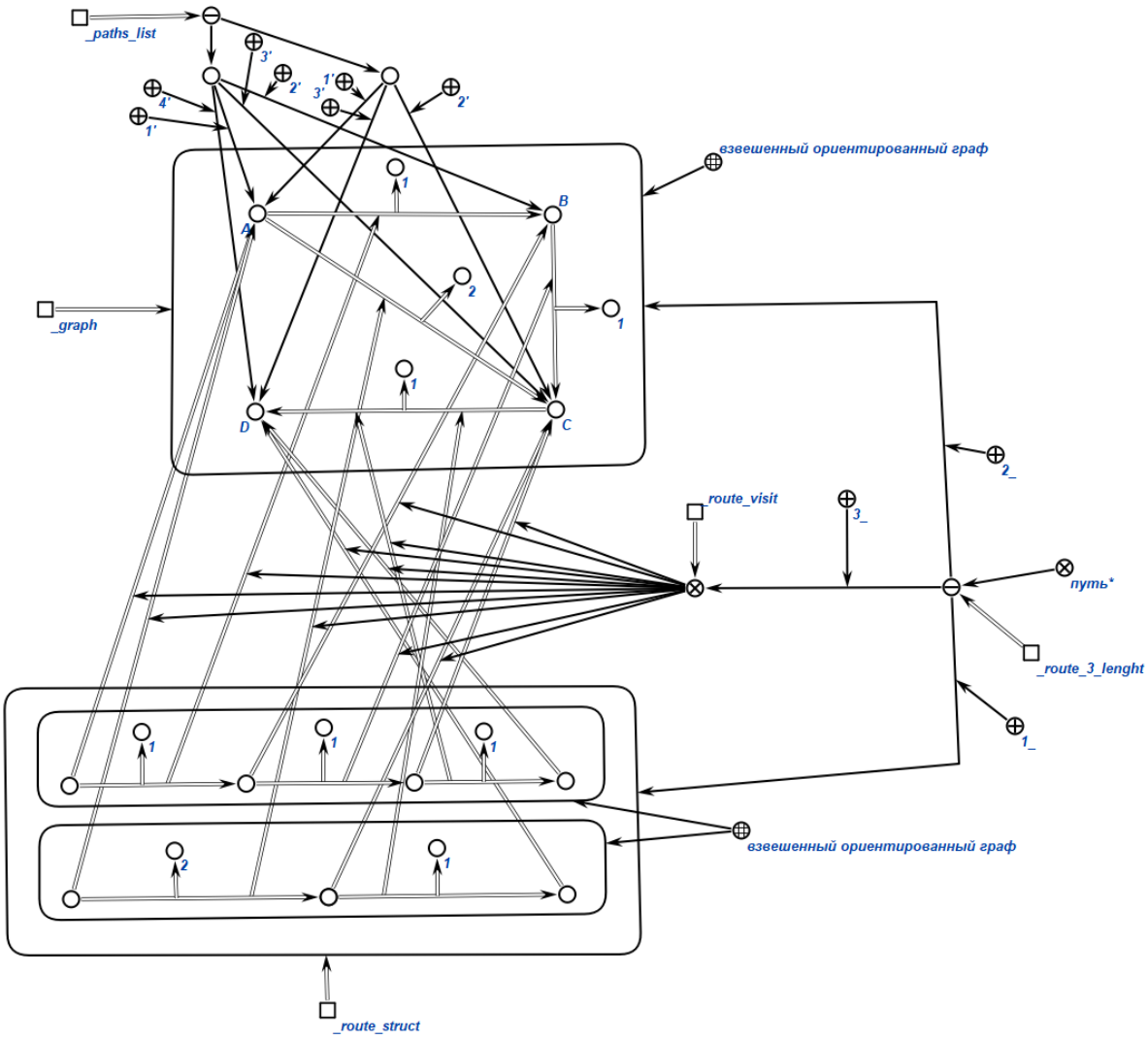
13. Создание связки отношения путь*



Создадим связку отношения путь* и установим ее в качестве значения переменной _route_3_length.

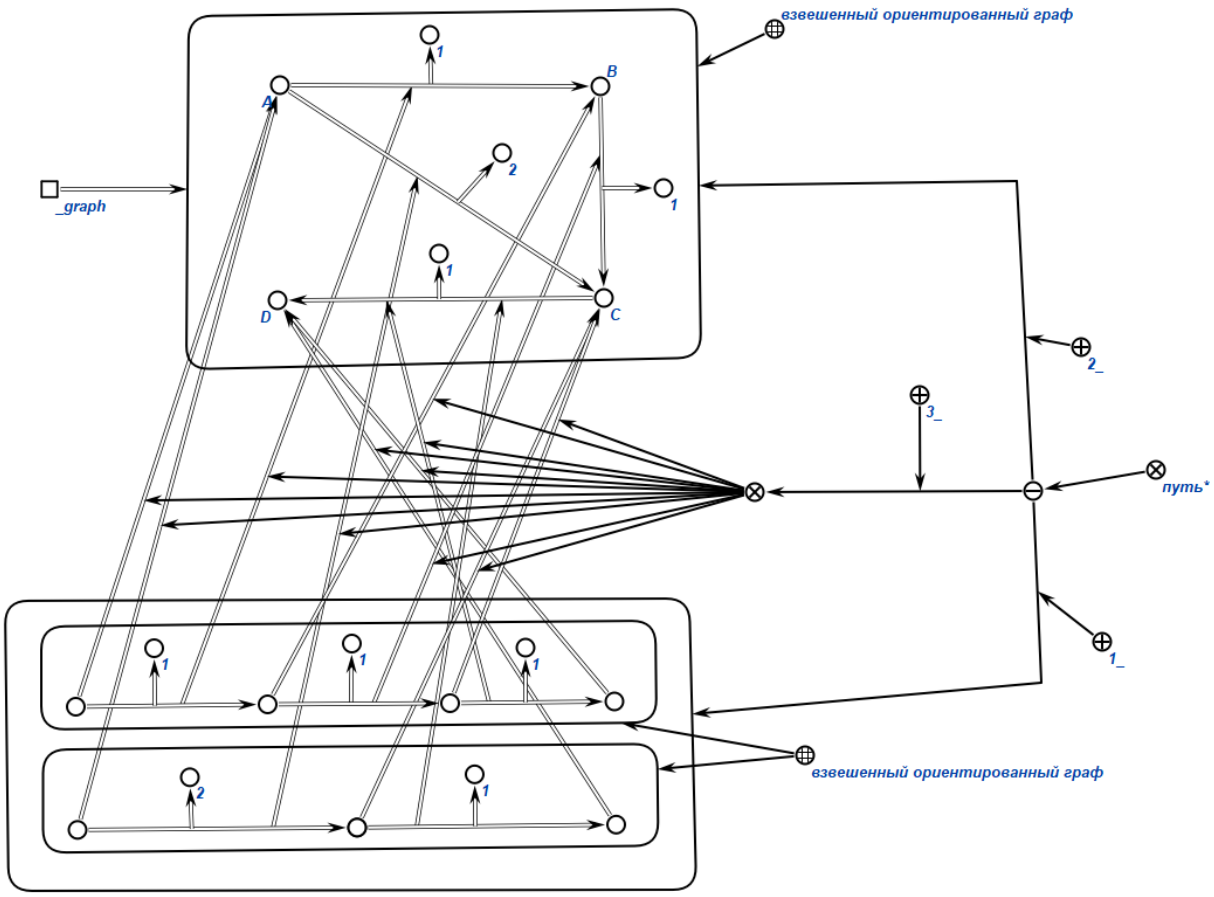
Переменная _route_struct получит в качестве значения взвешенный ориентированный граф структуры пути, а переменная _route_visit – отношения посещения.

14. Добавление в структуру пути посещения начальной и конечной вершины генерируемого пути (в нашем случае одинаковая вершина A)



Добавим в структуру два пути, хранящихся в списке путей _path_list, с созданием посещений вершин D, C, A и B в порядке их следования в ориентированных множествах. Создание посещения для связывающих ребер.

15. Удаление ориентированного списка _path_list и хранящихся в нем множеств. Результат работы алгоритма



4 Список литературы

OSTIS GT [В Интернете] // База знаний по теории графов OSTIS GT. - 2011 г.. - http://ostisgraphstheo.sourceforge.net/index.php/Заглавная_страница.

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