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

Представление и обработка информации в интеллектуальных системах

на тему

Поиск гамильтонова цикла в неориентированном графе

Выполнил: И. С. Прокопович

Студент группы

121703

Проверил: А. С. Загорский

Минск 2022

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

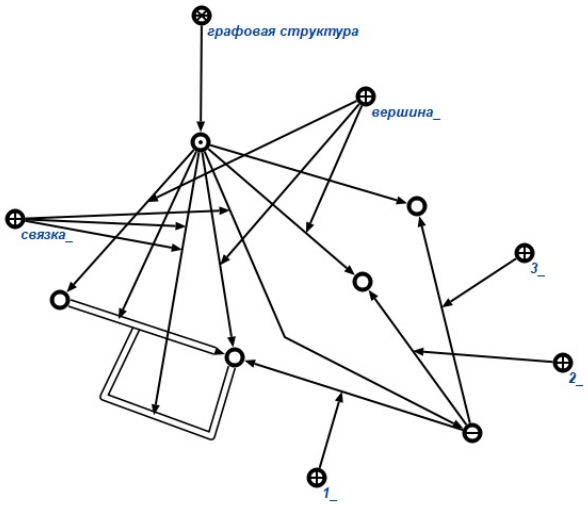
Задача: Найти гамильтонов цикл в неориентированном графе

1. Список понятий

1. Графовая структура (абсолютное понятие) - это такая одноуровневая реляционная структура, объекты которой могут играть роль либо вершины, либо связки:

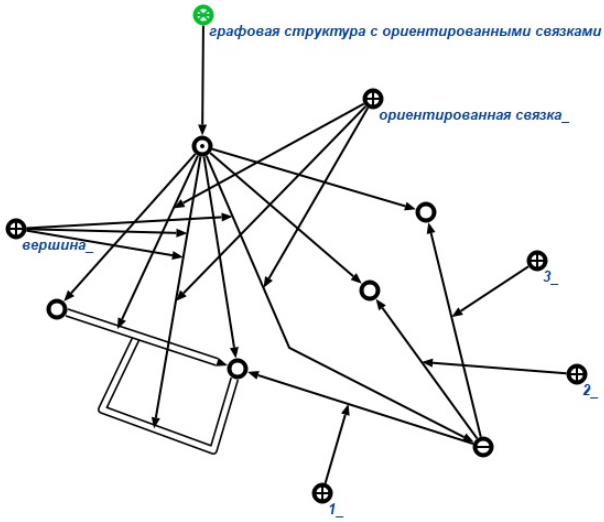
   (a) Вершина (относительное понятие, ролевое отношение);

   (b) Связка (относительное понятие, ролевое отношение).



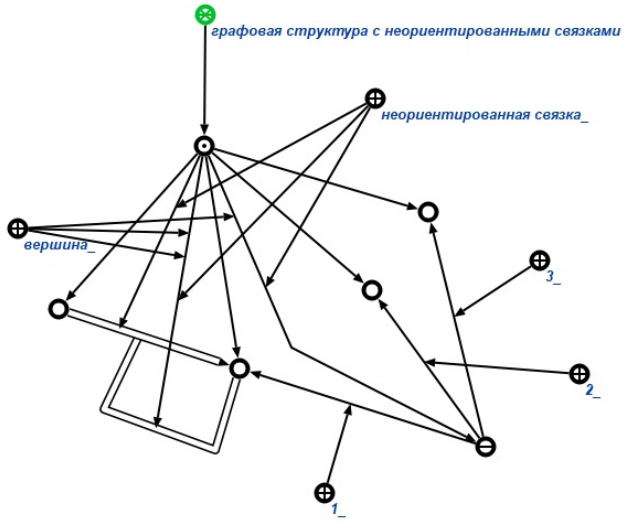
2. Графовая структура с ориентированными связками (абсолютное понятие)

   (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.

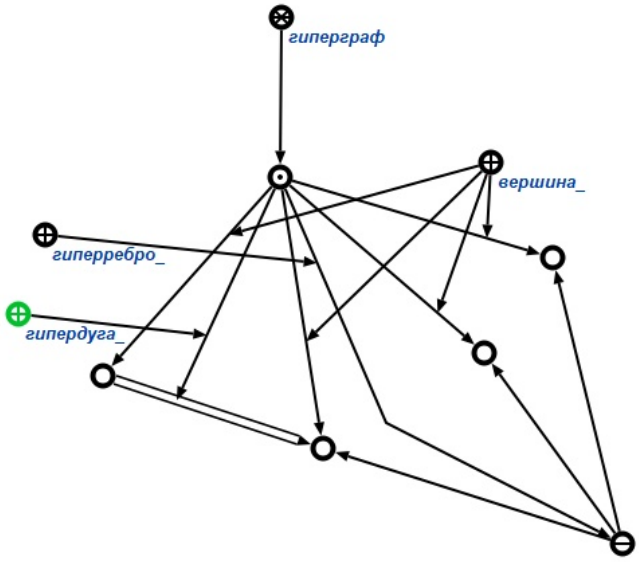


4. Гиперграф (абсолютное понятие) – это такая графовая структура, в которой связки могут связывать только вершины:

   (a) Гиперсвязка (относительное понятие, ролевое отношение)

   (b) Гипердуга (относительное понятие, ролевое отношение) – ориентированнаягипер-связка;

(c) Гиперребро (относительное понятие, ролевое отношение) – неориентированнаягипер-связка.



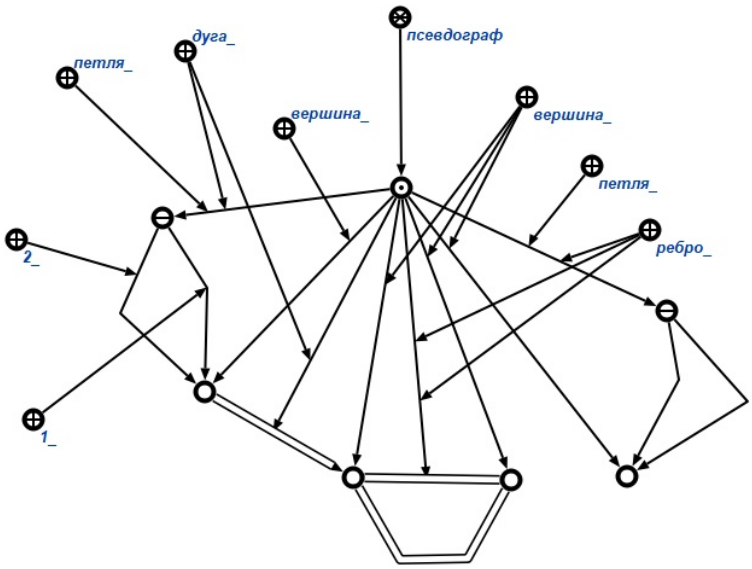
5. Псевдограф (абсолютное понятие) – это такой гиперграф, в котором все связки должны быть бинарными

   (a) Бинарная связка (относительное понятие, ролевое отношение) –гиперсвязка арности 2;

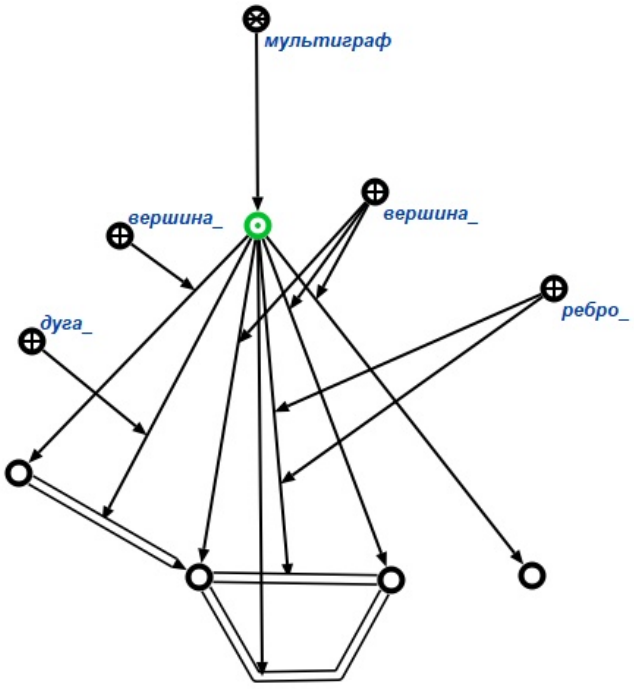
   (b) Ребро (относительное понятие, ролевое отношение) –неориентированнаягиперсвязка;

   (c) Дуга (относительное понятие, ролевое отношение) – ориентированная гиперсвязка;

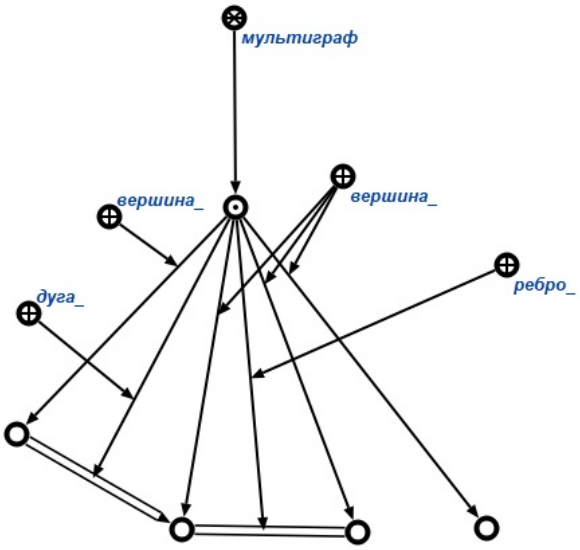
   (d) Петля (относительное понятие, ролевое отношение) – бинарная связка, у которой первый и второй компоненты совпадают.



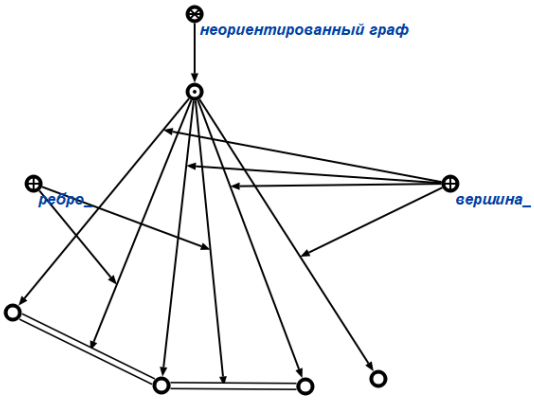
6. Мультиграф (абсолютное понятие) – это такой псевдограф, в котором не может быть петель:



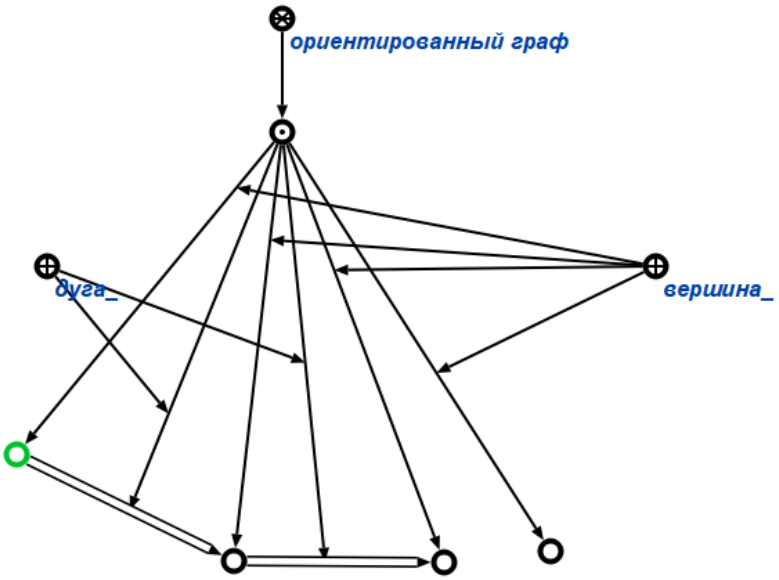
7. Граф (абсолютное понятие) – это такой мультиграф, в котором не может быть кратны связок, т.е. связок у которых первый и второй компоненты совпадают:



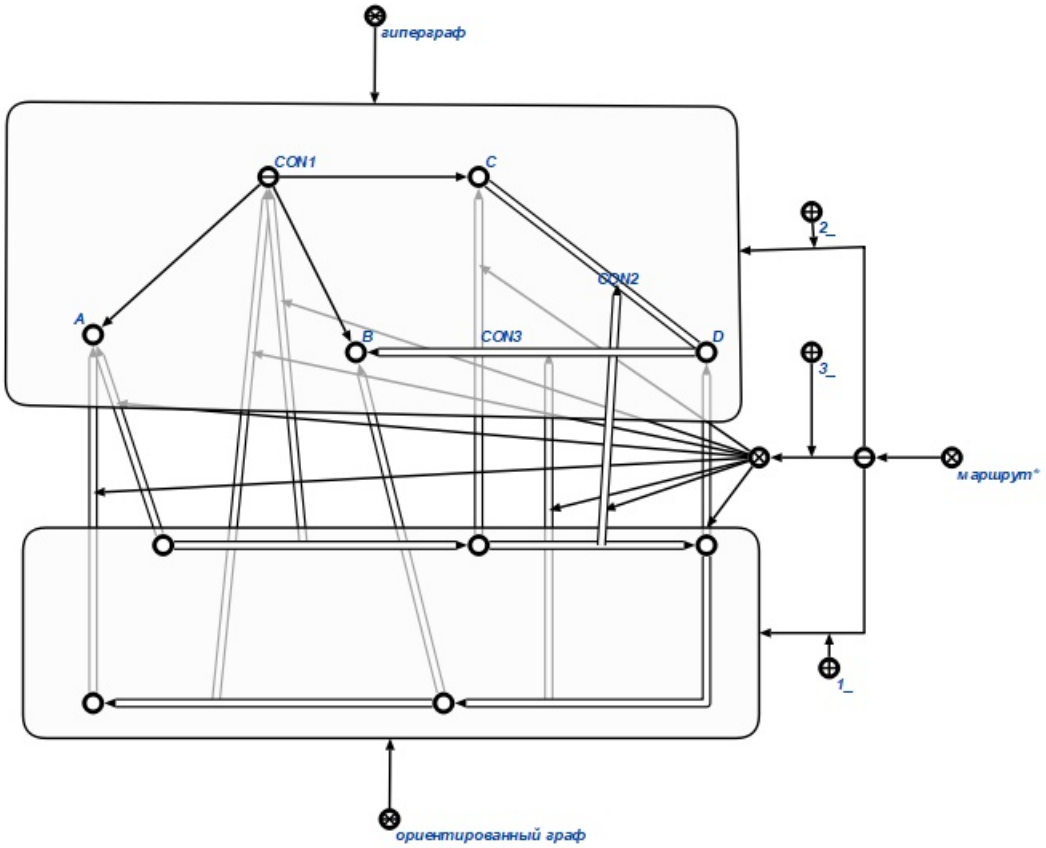
8. Неориентированный граф (абсолютное понятие) –это такой граф, в котором все связки являются ребрами:



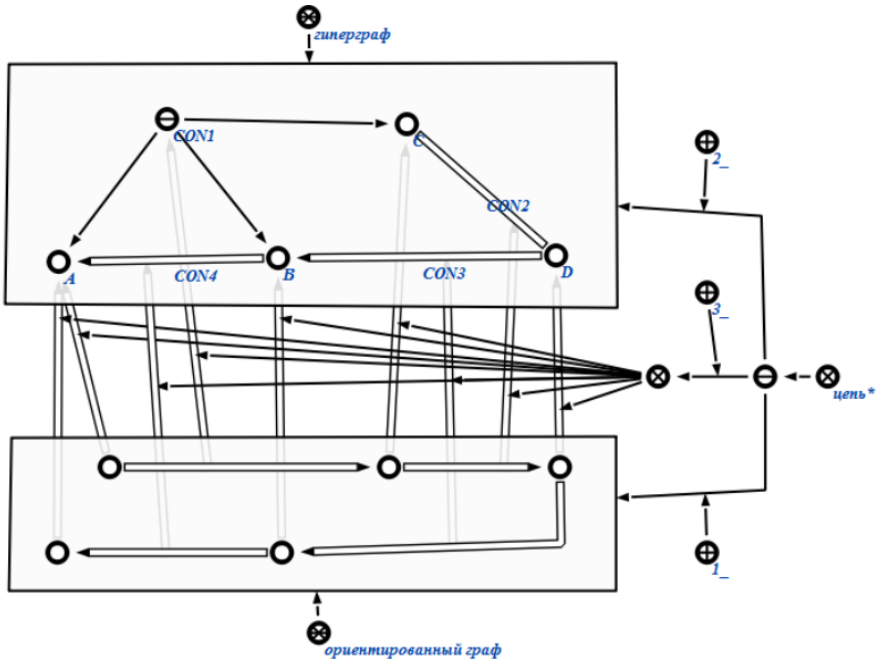
9. Ориентированный граф (абсолютное понятие) - это такой граф, в котором все связки являются дугами:



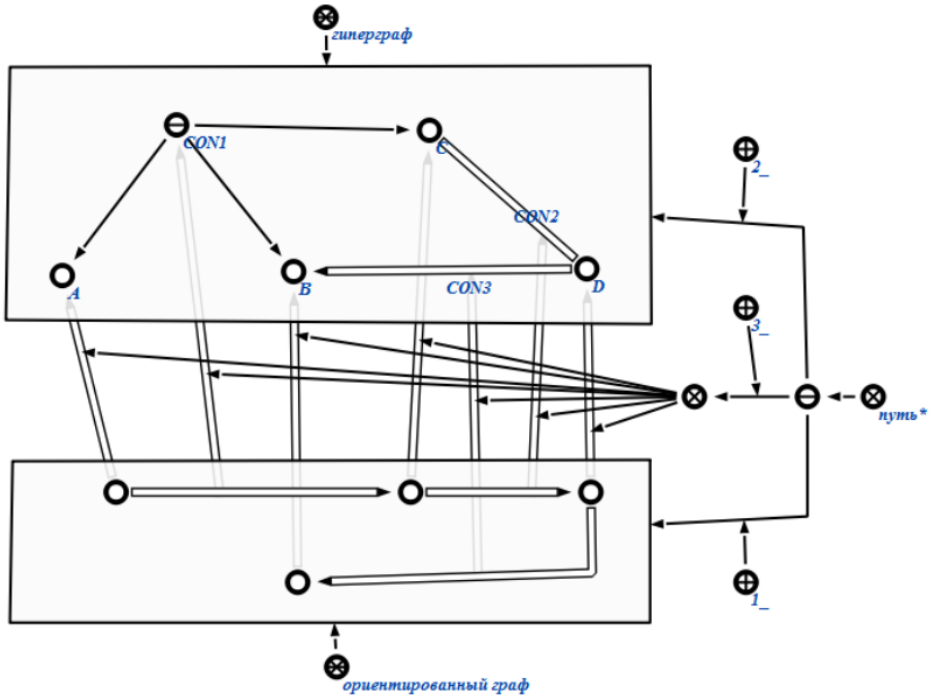
10. Маршрут (относительное понятие, бинарное ориентированное отношение) – это чередующаяся последовательность вершин и гиперсвязок в гиперграфе, которая начинается и кончается вершиной, и каждая гиперсвязка последовательности инцидентна двум вершинам, одна из которых непосредственно предшествует ей,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



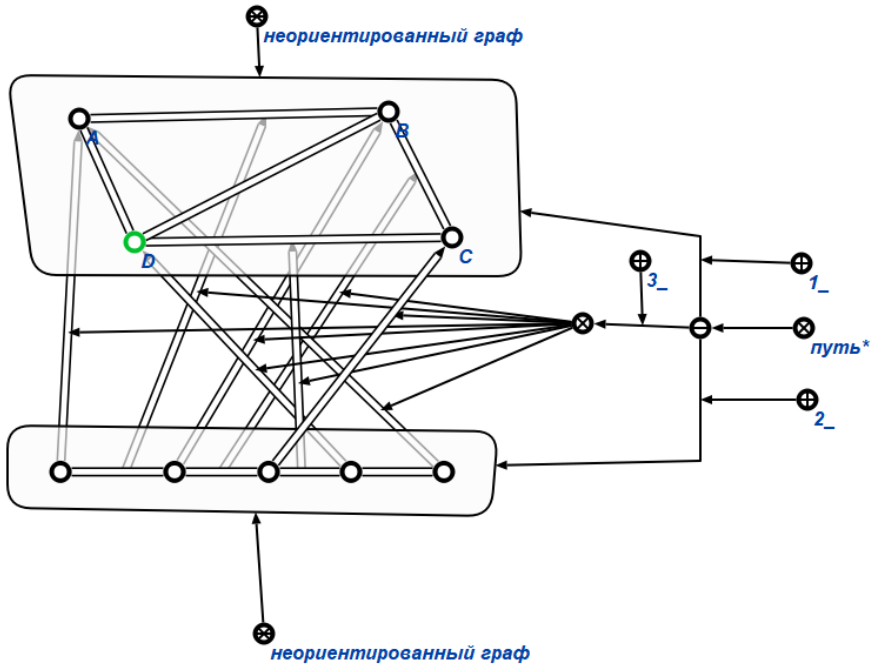
11. Цепь (относительное понятие, бинарное ориентированное отношение) – это маршрут, все гиперсвязки которого различны. В примере ниже показана цепь A, CON1, C, CON2, D, CON3, B, CON4, A в гиперграфе.



12. 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.



13. Гамильтонов цикл (относительное понятие, бинарное неориентированное отношение) – это замкнутый путь, который проходит через каждую вершину данного графа ровно по одному разу; то есть простой путь, в который входят все вершины графа. В примере ниже Гамильтонов цикл: A, B, C, D, A.



2. Алгоритм

1. Выбираем начальную вершину для поиска цикла.

2. В множество path записываем выбранную вершину.

3. Заполняем множество visit значениями 0 (false) - для остальных вершин (начальная вершина не включается в это множество).

4. Определяем соседний узел выбранной вершины который ещё не был посещён и вписываем его в множество path.

5. При переходе к следующей веришне, помечаем предыдущую вершину во множестве visit как посещенную - 1 (true).

6. Если мы выбрали вершину которая имеет связь только с уже посещенными вершинами причем остались еще непосещенные вершины мы возвращаемся на шаг назад, убирая добавленную вершину из множества path и меняем значение для этой вершины в множестве visit на непосещенную

7. Выбираем другую вершину отличную от той которую мы убирали

8. Если таким образом мы приходим к начальной вершине при выборе всех вершин связанных с начальной (2-ой элемент в множестве path), то гамильтонов цикл отсутсвует

9. Повторяем пункты 4, 5 до тех пор, пока все вершины не будут посещены

10. Выводим множество path (Гамильтонов цикл)

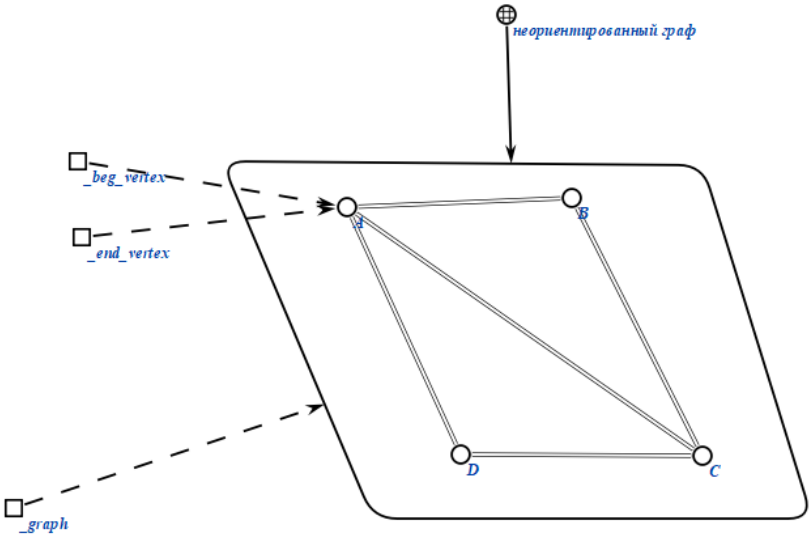
3. Тестовые примеры

3.1 Тест 1

Вход:

Необходимо найти Гамильтонов цикл для вершины A.

1. Задание входного графа, начальной и конечной вершины пути для работы алгоритма



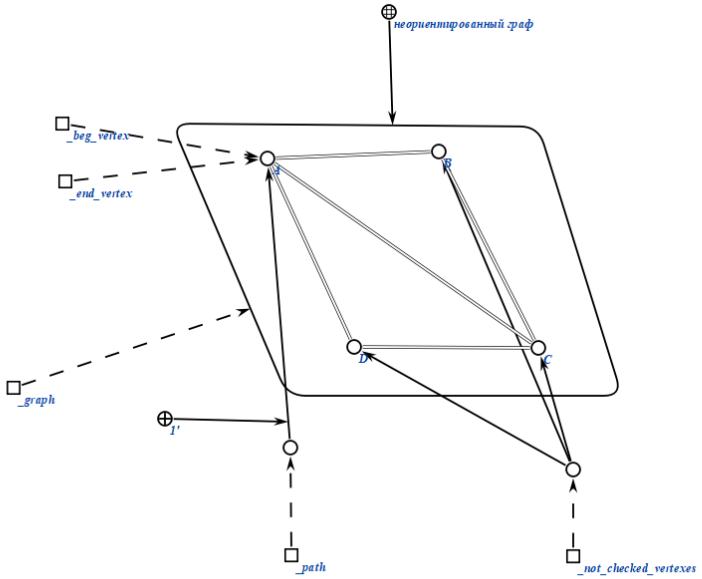
Переменные изменятся следующим образом:

• _graph получит в качестве значения sc-узел ориентированного графа;

• _beg_vertex получит в качестве значения вершину A, которая будет начальной для поиска минимального пути;

• _end_vertex получит в качестве значение вершину A, которая будет конечной для поиска минимального пути. Таким образом, из состояния sc-памяти на этом шаге вам должно быть ясно, что будет производиться поиск Гамильтонова цикла для вершины A.

2. Создание ориентированного множества пути и множества неиспользованных вершин



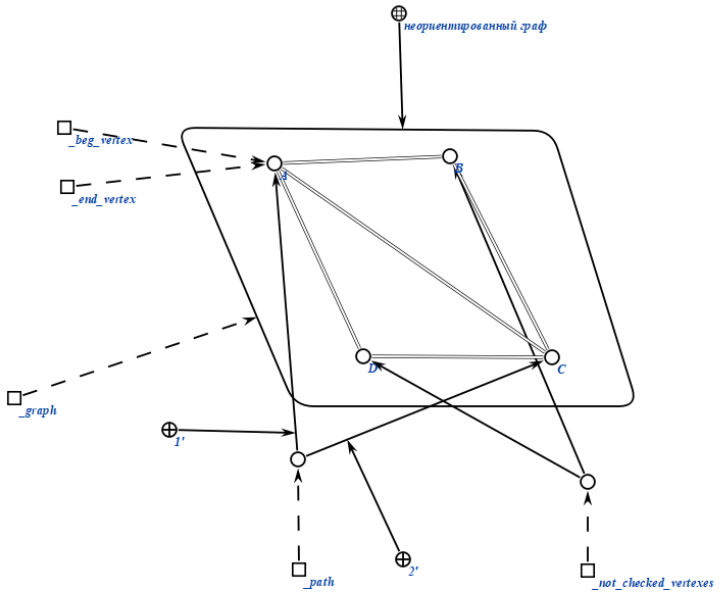
Переменная _path получит в качестве значения будущее ориентированное множество вершин в порядке следования в Гамильтоновом цикле обрабатываемого графа. Будем использовать его для проверки, используется ли проверяемая вершина уже в цикле. Добавляем первую вершину: это всегда будет вершина A, так как гамильтонов цикл мы в задаче всегда строим именно для этой вершины. Указываем ее порядок следования во множестве _path.

Переменная _not_checked_vertexes получит в качестве значения множество непроверенных вершин обрабатываемого графа (в это множество не включена начальная вершина пути A).

Каждый раз нам следует проверять, остались ли еще неиспользованные вершины. Мы не можем использовать одинаковые вершины несколько раз, так и не можем оставить неиспользованных вершин.

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

3. Добавление во множество используемых вершин вершины C

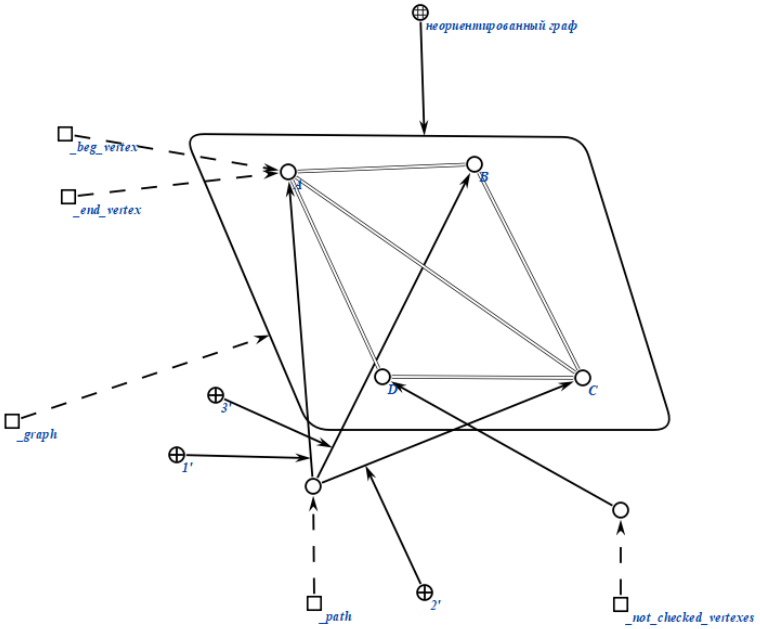


Из трех вершин, с которыми имеет связи выбираем первую - C. Добавляем ее во множество пути с порядковым номером 2.

Удаляем из множества неиспользованных вершин вершину C.

Вершина C имеет связь с двумя непосещенными вершинами:B и D.

4. Добавление во множество используемых вершин вершины B

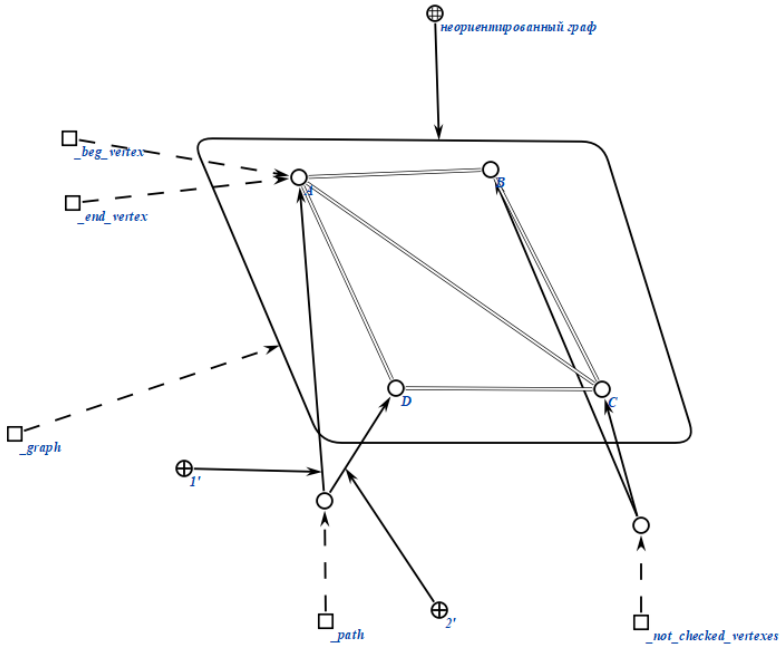


Добавляем вершину B во множество пути с порядковым номером 3.

Удаляем из множества неиспользованных вершин вершину B.

Вершина B имеет связь с вершинами: A и C. Проверка множества неиспользованных вершин показывает, во множестве остались элементы: вершина D. Следовательно, построив связь с вершиной A мы завершим цикл, но он не будет являться Гамильтоновым. Возвращаемся на шаг назад, убирая вершину B из _path. Вершина C ещё имеет связь с вершной D (так как при добавлении вершины D в _path происходит аналогичная ситуация). Возвращаемся на шаг назад, убирая вершину C из _path. Вершины C и B добавляем в список неиспользованных вершин. В итоге вернулись к выбору из трех вершин: B, C, D и выбираем D.

5. Добавление во множество используемых вершин вершины D

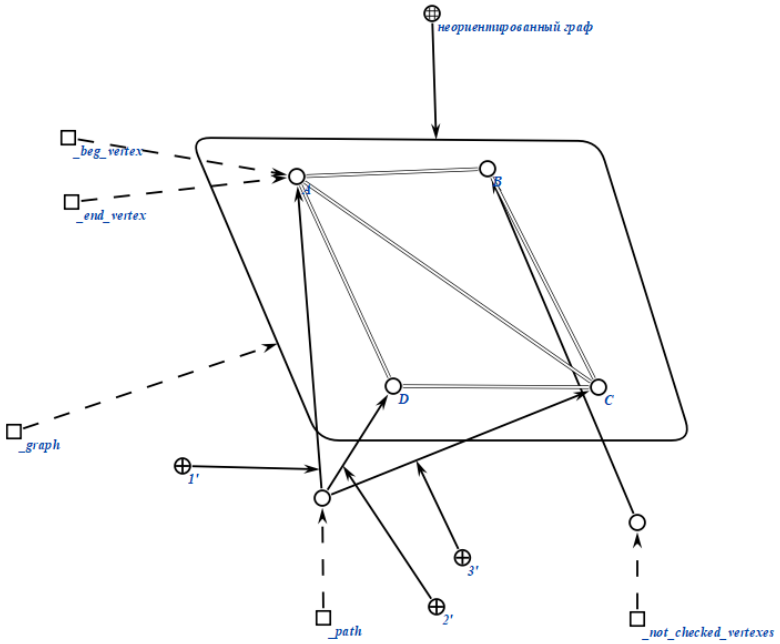


Из двух вершин, с которыми имеет связи выбираем вторую - D. Добавляем ее во множество пути с порядковым номером 2.

Удаляем из множества неиспользованных вершин вершину D.

Вершина D имеет связь с единственной вершиной: C.

6. Добавление во множество используемых вершин вершины C

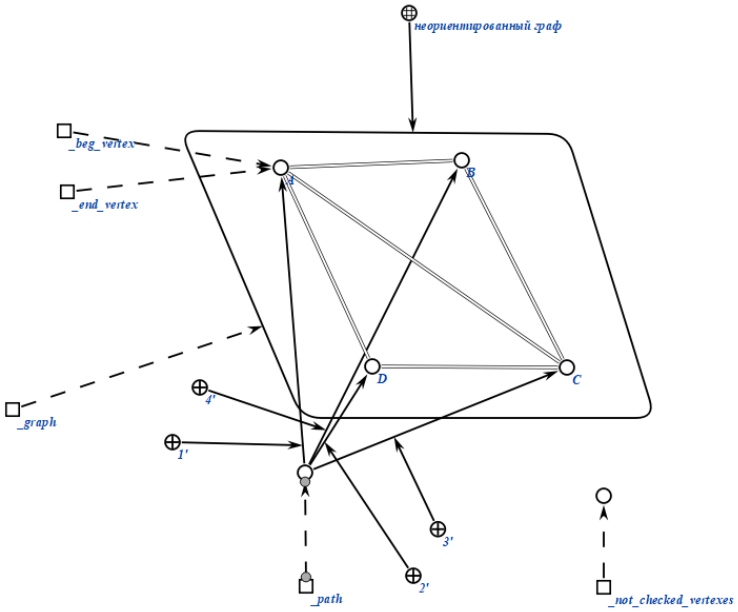


Добавляем вершину C во множество пути с порядковым номером 3.

Удаляем из множества неиспользованных вершин вершину C.

Вершина C имеет связь с единственной непосещенной вершиной: B.

7. Добавление во множество используемых вершин вершины B

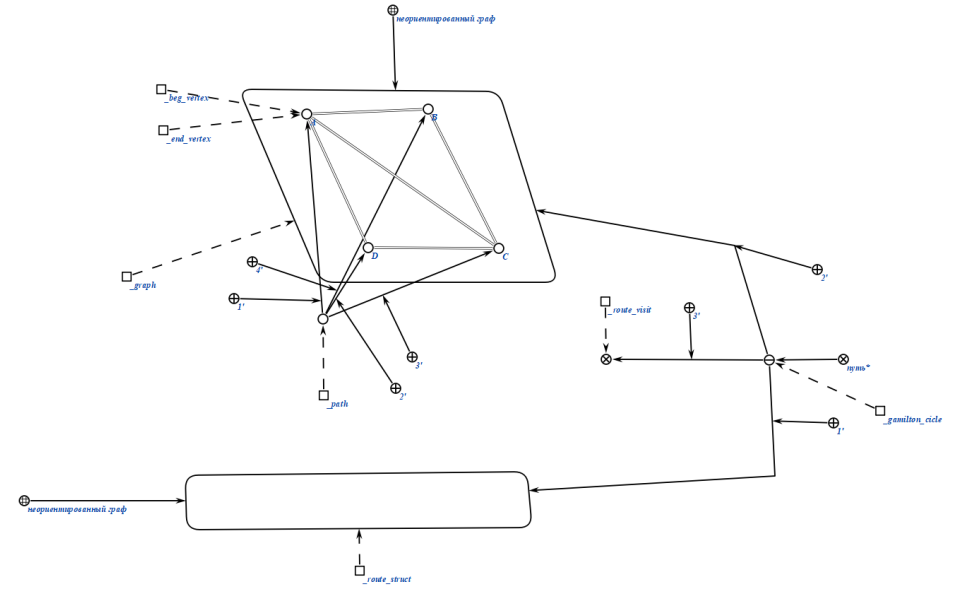


Добавляем вершину B во множество пути с порядковым номером 4.

Удаляем из множества неиспользованных вершин вершину B.

Вершина B имеет связь с единственной вершиной: A.

8. Создание связки отношения путь*

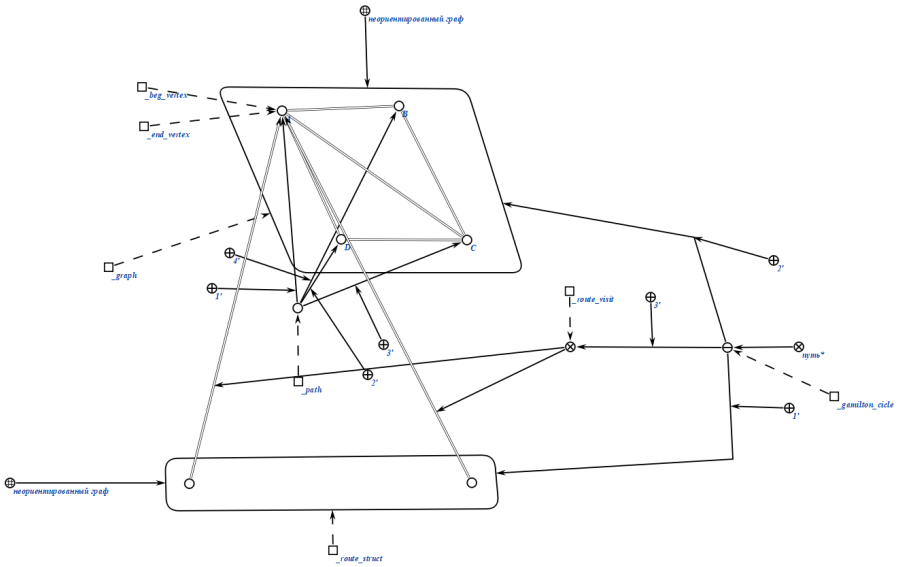


Можем получить ответ, так как множество неиспользованных вершин осталось пустым и было удалено, а мы пришли в вершину, которая хранится в переменной _end_vertex.

Начинаем генерацию Гамильтонова цикла.

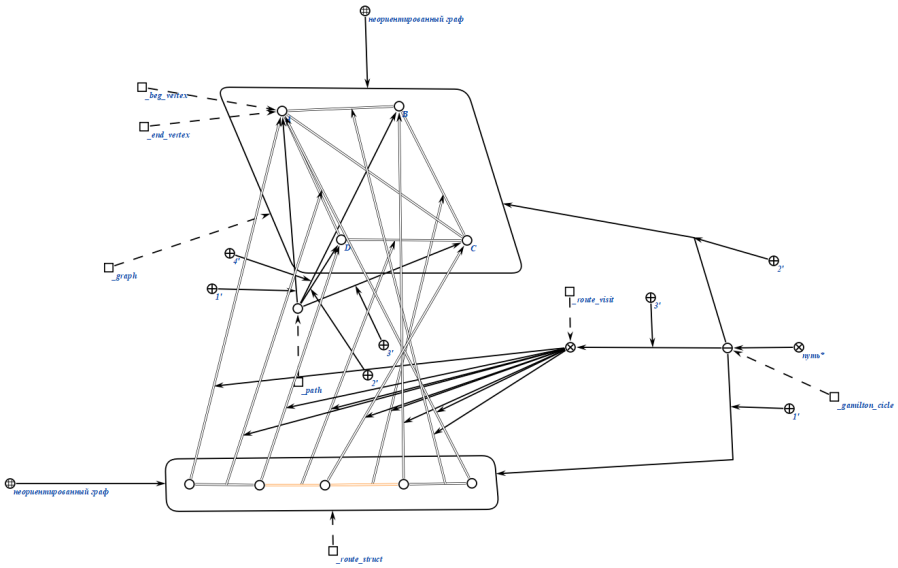
Создадим связку отношения путь* и установим ее в качестве значения переменной _gamilton_cicle. Переменная _route_struct получит в качестве значения ориентированный граф структуры пути, а переменная _route_visit – отношения посещения.

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



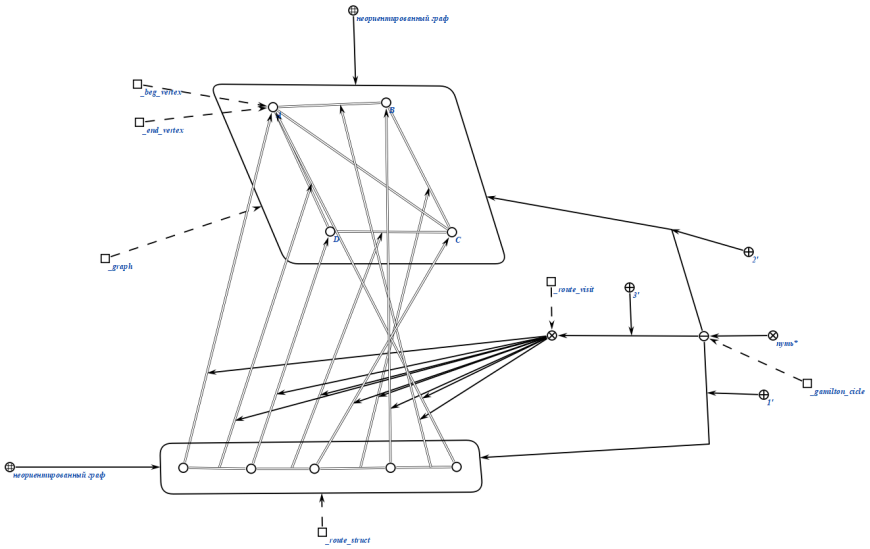
Добавим в структуру генерируемого пути посещение начальной и конечной вершины A.

10. Добавление в структуру пути посещения всех элементов ориентированного множества _path в порядке их следования в нем. Создание посещения для связывающих ребер



Добавим в структуру генерируемого пути посещение вершин D, C и B в порядке их следования в ориентированном множестве _path. Создание посещения для связывающих ребер и вершин.

11. Удаление ориентированного множества _path. Результат работы алгоритма



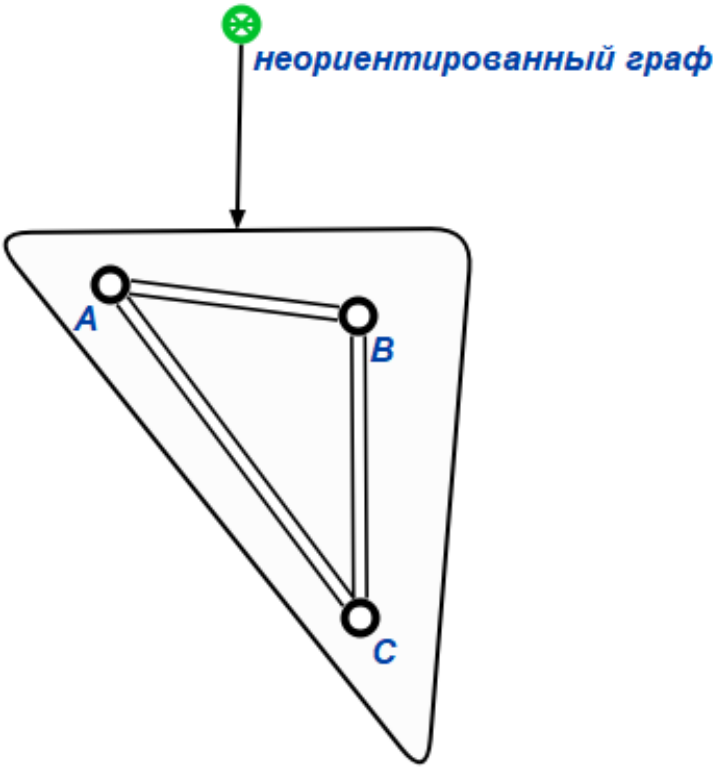
Выход:

Будет найден Гамильтонов цикл A, D, C, B, A:

3.2 Тест 2

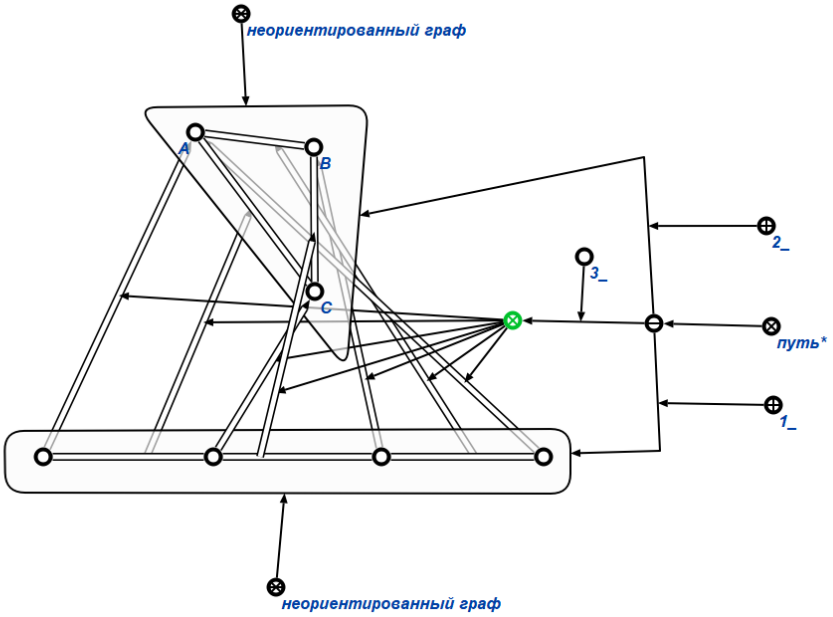
Вход:

Необходимо найти Гамильтонов цикл для вершины A.



Выход:

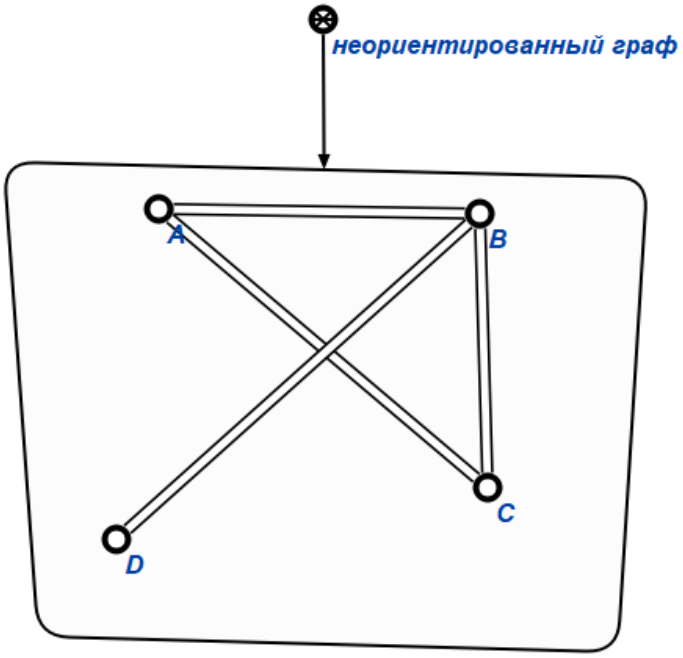
Будет найден Гамильтонов цикл A, C, B, A:



3.3 Тест 3

Вход:

Необходимо найти Гамильтонов цикл для вершины A.



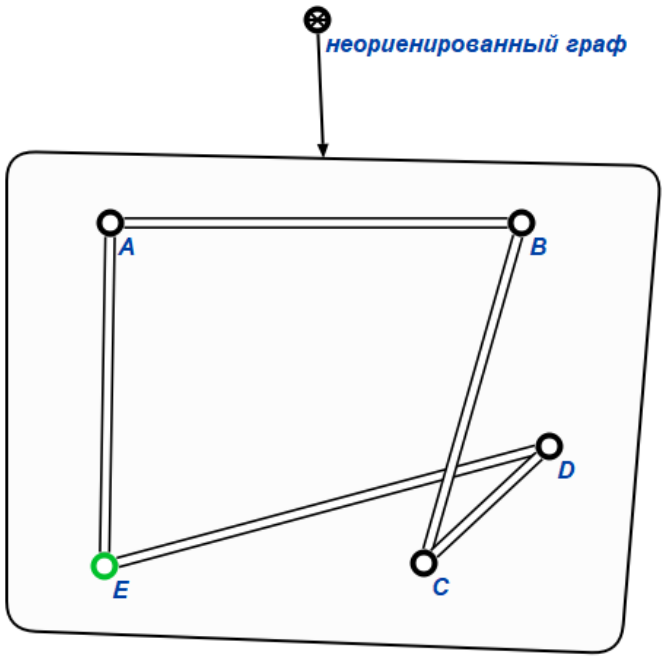
Выход:

Гамильтонова цикла для вершины A не существует. Программа должна вернуть ошибку вызывающему контексту.

3.4 Тест 4

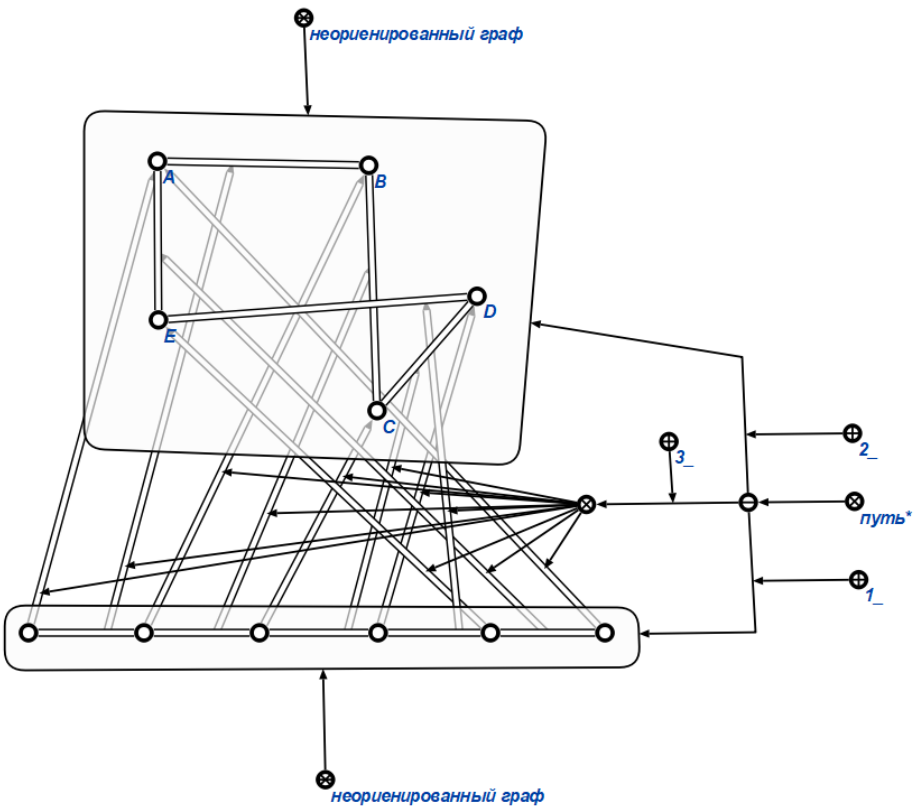
Вход:

Необходимо найти Гамильтонов цикл для вершины A.



Выход:

Будет найден Гамильтонов цикл A, B, C, D, E, A:



4. Вывод

В ходе данной расчетной работе мы получили навыки формализации и обработки с использованием семантических сетей, а также научились находить гамильтонов цикл в неориентированном графе

5. Список литературы

[1] OSTIS GT [В Интернете] // База знаний по теории графов OSTIS GT. - 2011 г..
- http://ostisgraphstheo.sourceforge.net/index.php.
[2] Харарри Ф. Теория графов [Книга]. - Москва : Едиториал УРСС, 2003