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 компонент связности в неориентированном графе

Выполнил Гущик Н.Ю.
студент группы
021704

Проверил Витязь В.С

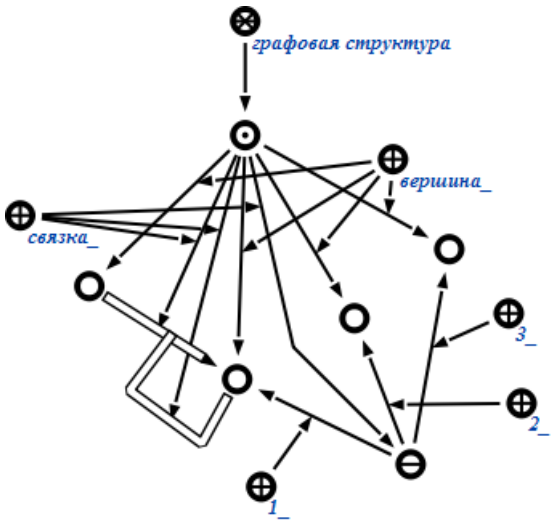
Минск 2021

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

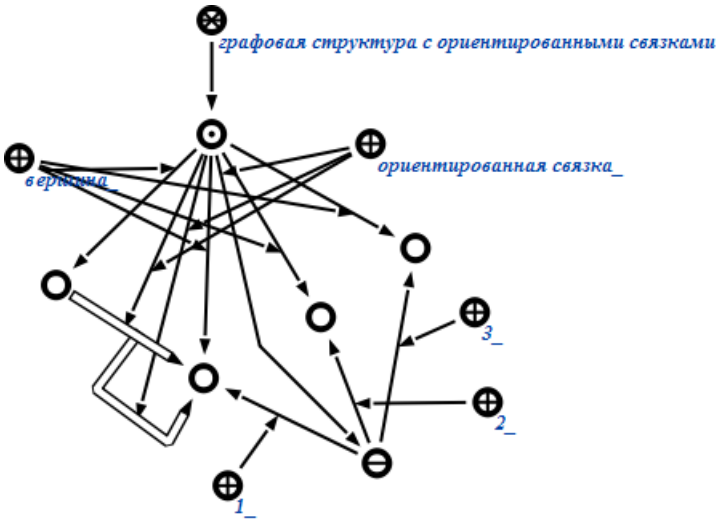
Задача: поиск компонент связности в неориентированном графе

1 Список понятий

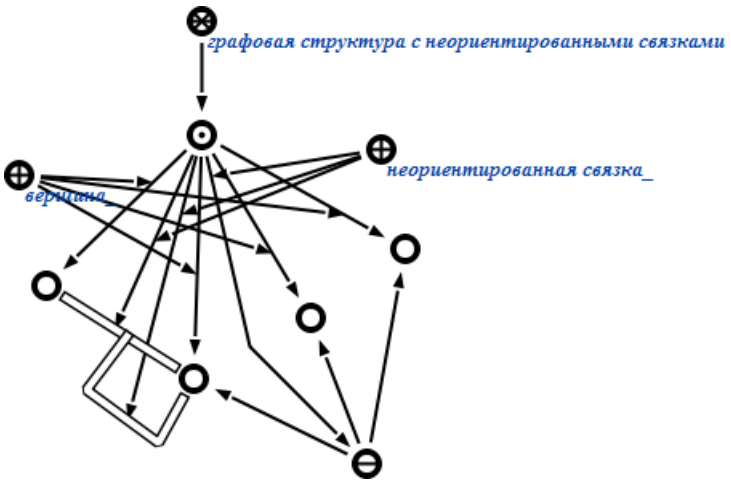
1. Графовая структура (абсолютное понятие) - это такая одноуровневая реляционная структура, объекты которой могут играть роль либо вершины, либо связки:
    a. Вершина (относительное понятие, ролевое отношение);
    b. Связка (относительное понятие, ролевое отношение).



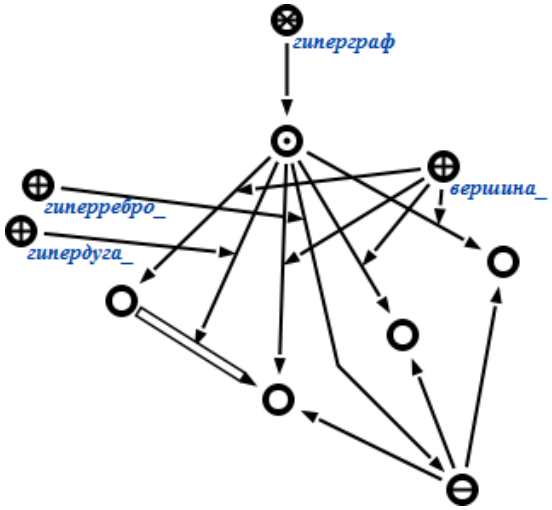
2. Графовая структура с ориентированными связками (абсолютное понятие)
    a. Ориентированная связка (относительное понятие, ролевое отношение) – связка, которая задается ориентированным множеством.



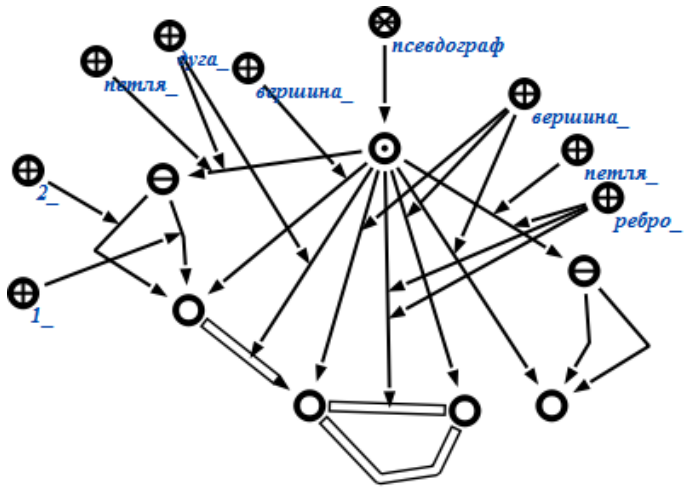
3. Графовая структура с неориентированными связками (абсолютное понятие)
    a. Неориентированная связка (относительное понятие, ролевое отношение) – связка, которая задается неориентированным множеством.



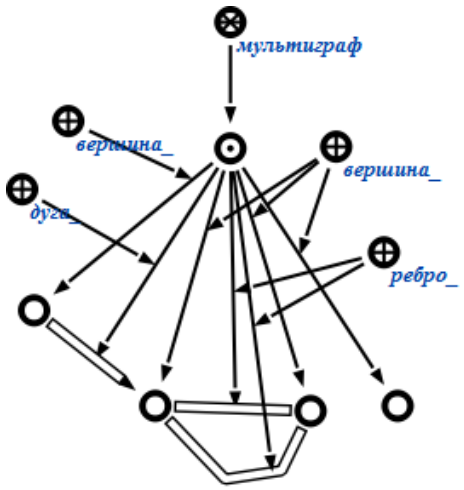
4. Гиперграф (абсолютное понятие) – это такая графовая структура, в которой связки могут связывать только вершины:
    a. Гиперсвязка (относительное понятие, ролевое отношение);
    b. Гипердуга (относительное понятие, ролевое отношение) – ориентированнаягиперсвязка;
    c. Гиперребро (относительное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



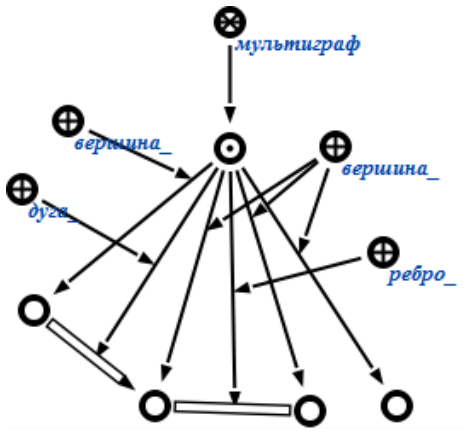
5. Псевдограф (абсолютное понятие) – это такой гиперграф, в котором все связки должны быть бинарными:
    a. Бинарная связка (относительное понятие, ролевое отношение) –гиперсвязка арности 2;
    b. Ребро (относительное понятие, ролевое отношение) – неориентированнаягиперсвязка;
    c. Дуга (относительное понятие, ролевое отношение) – ориентированная гиперсвязка;
    d. Петля (относительное понятие, ролевое отношение) – бинарная связка, у которой первый и второй компоненты совпадают.



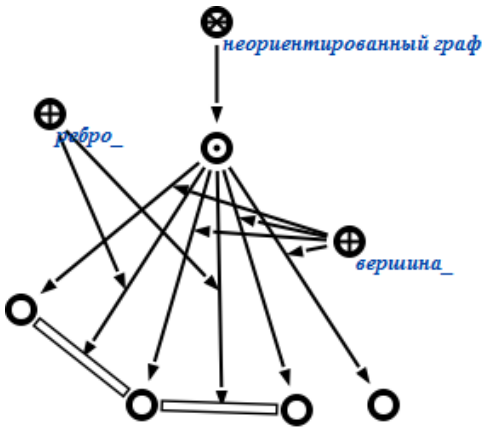
6. Мультиграф (абсолютное понятие) – это такой псевдограф, в котором не может быть петель:



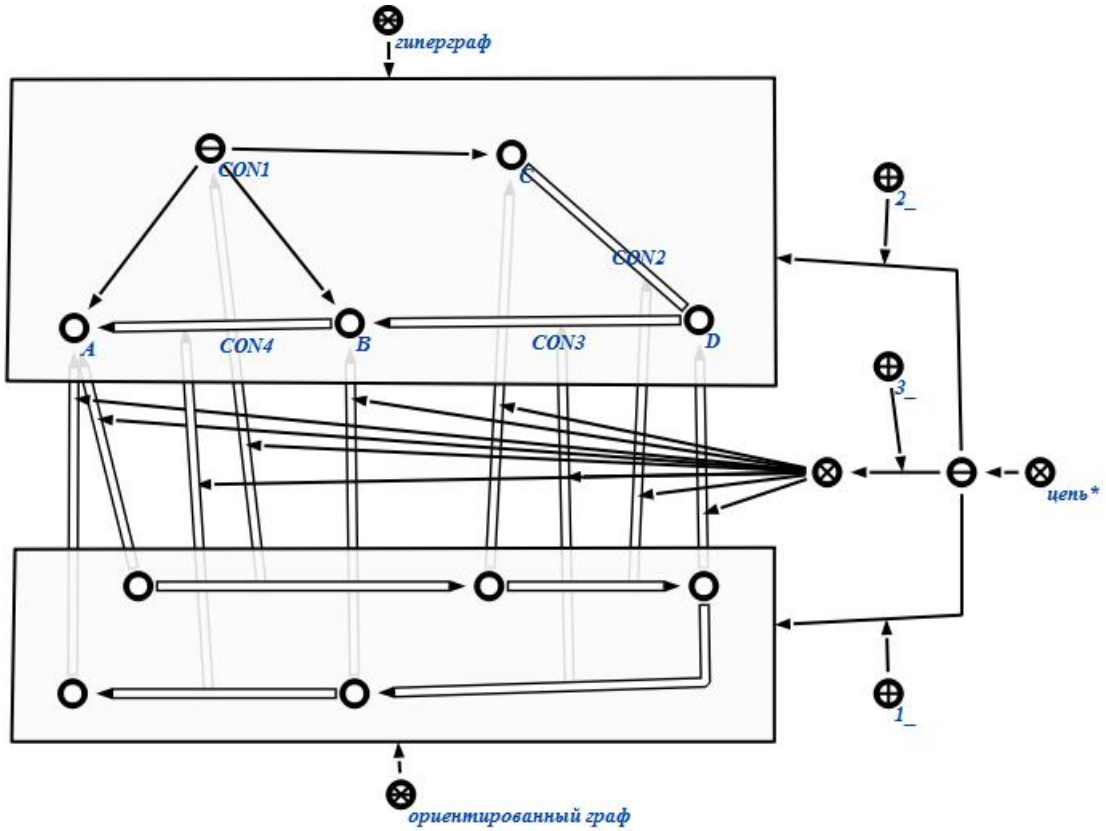
7. Граф (абсолютное понятие) – это такой мультиграф, в котором не может быть кратных связок, т.е. связок у которых первый и второй компоненты совпадают:



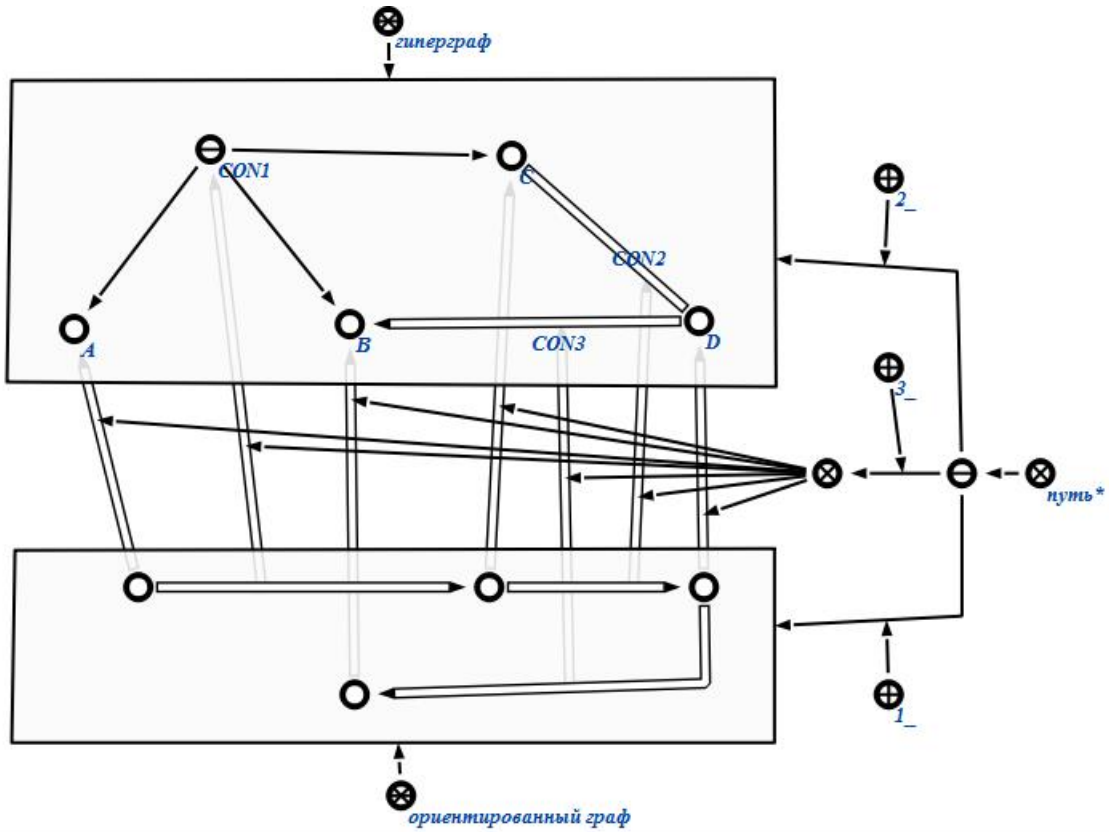
8. Неориентированный граф (абсолютное понятие) –это такой граф, в котором все связки являются ребрами:



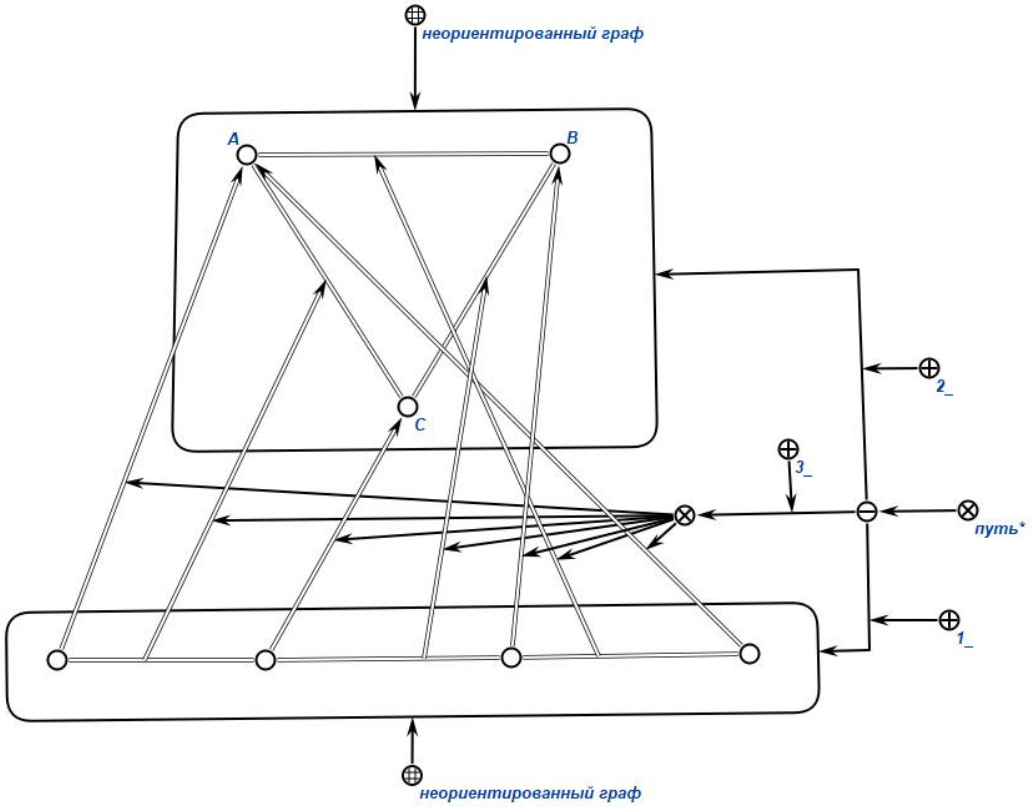
9. Цепь (относительное понятие, бинарное ориентированное отношение) – это маршрут, все гиперсвязки которого различны. В примере ниже показана цепь *A*, *CON1*, *C*, *CON2*, *D*, *CON3*, *B*, *CON4*, *A* в гиперграфе.



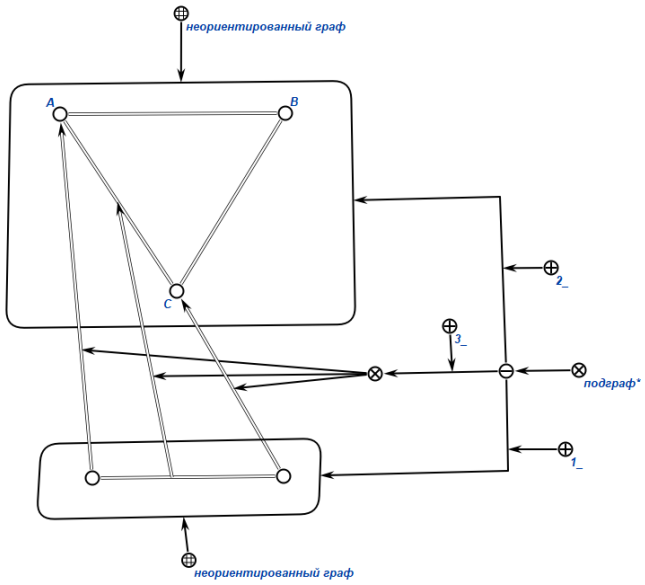
10. Простая цепь, путь (относительное понятие, бинарное ориентированное отношение) – это цепь, в которой все вершины различны. В примере ниже показан путь *A*, *CON1*, *C*, *CON2*, *D*, *CON3*, *B* в гиперграфе.



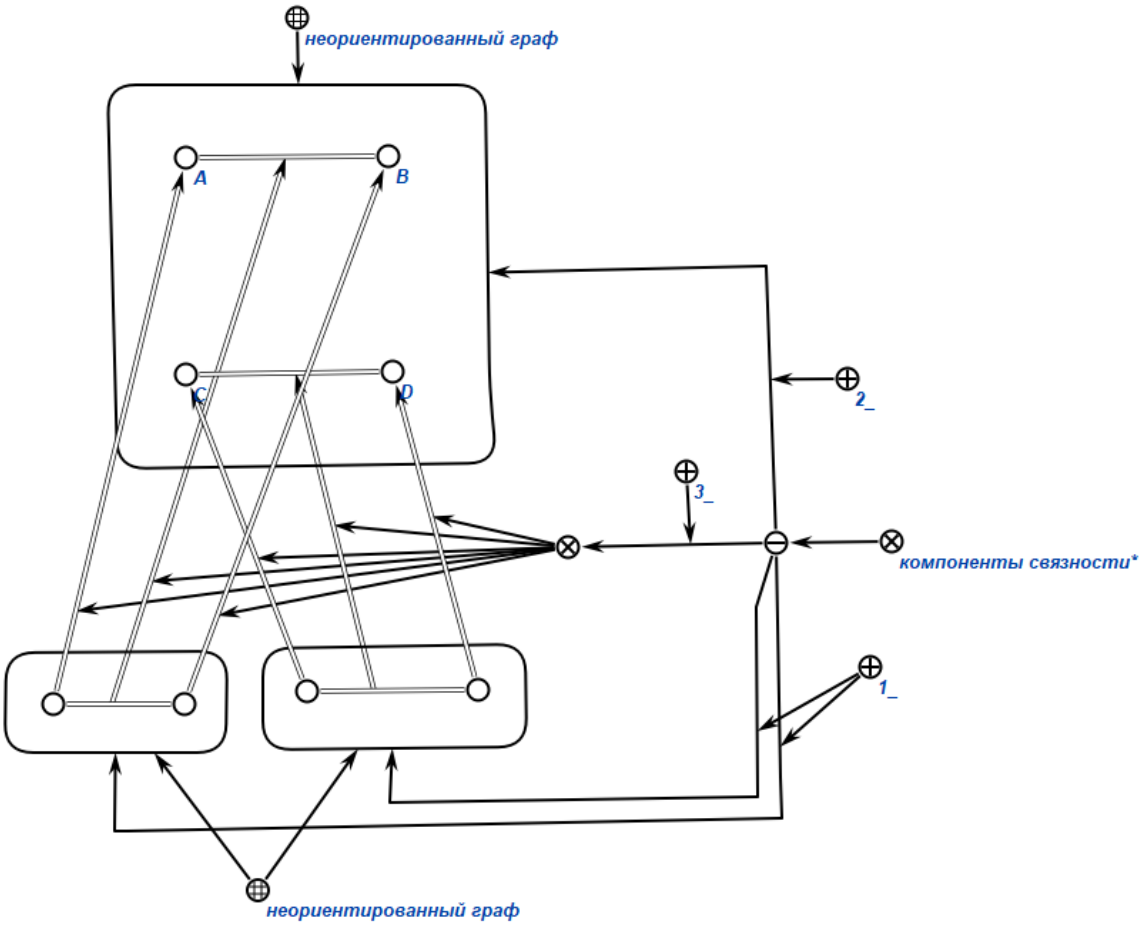
11. Связный граф (относительное понятие) – граф, содержащий ровно одну компоненту связности. Это означает, что между любой парой вершин этого графа существует как минимум один путь.



12. Подграф (относительное понятие) – граф, содержащий некое подмножество вершин данного графа и некое подмножество инцидентных им рёбер. В примере ниже показан подграф, состоящий из вершин A и C основного графа.



13. Компонента связности (относительное понятие) – максимальный (по включению) связный подграф графа. В примере ниже показаны две компоненты связности: A и B, C и D.



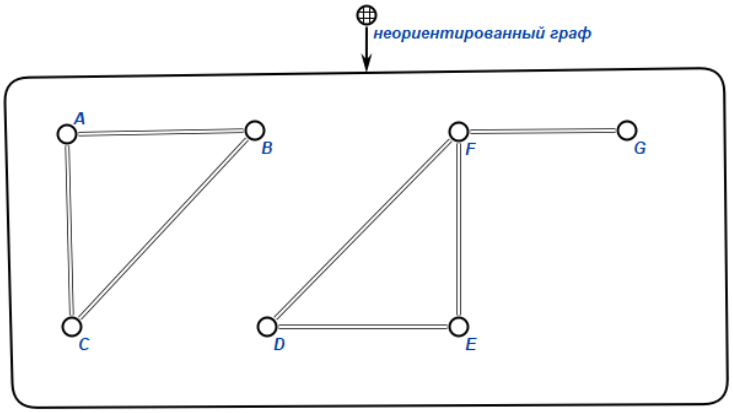
2 Тестовые примеры

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

2.1 Тест 1

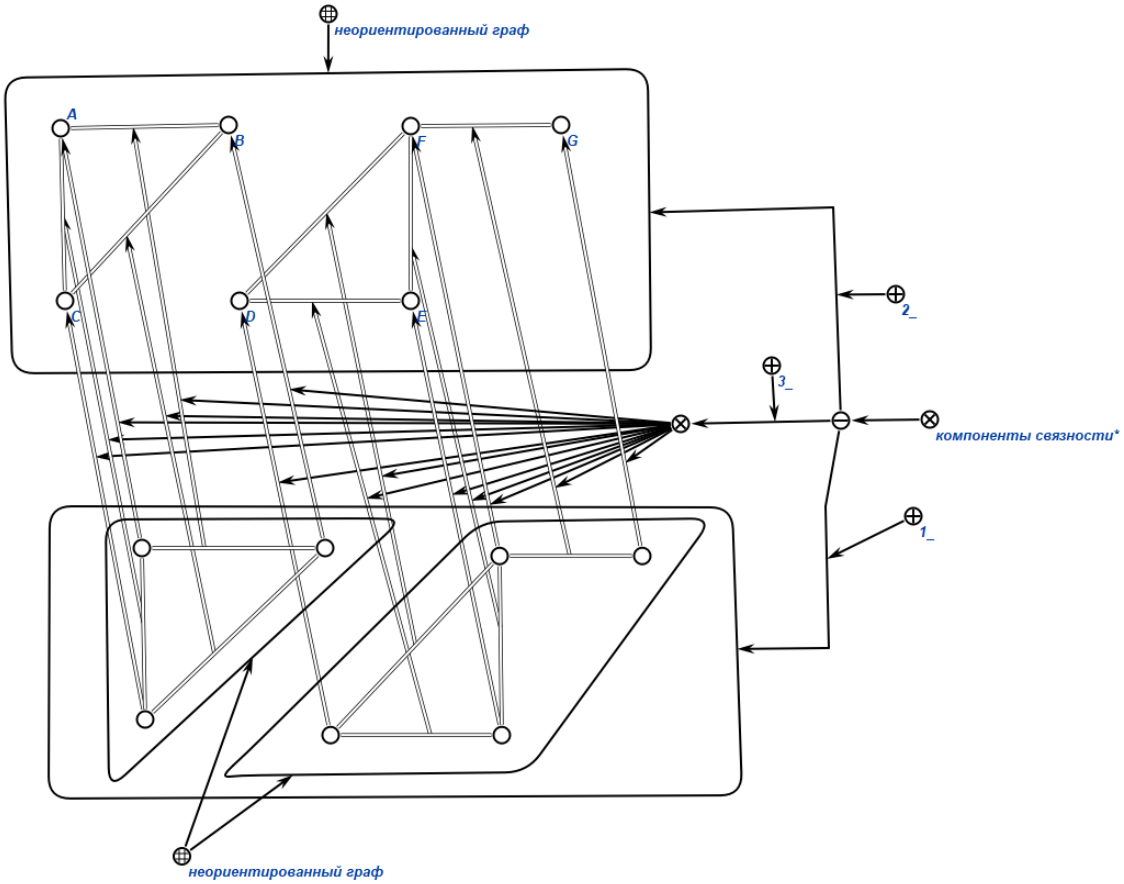
Вход:

Необходимо найти компоненты связности в неориентированном графе.



Выход:

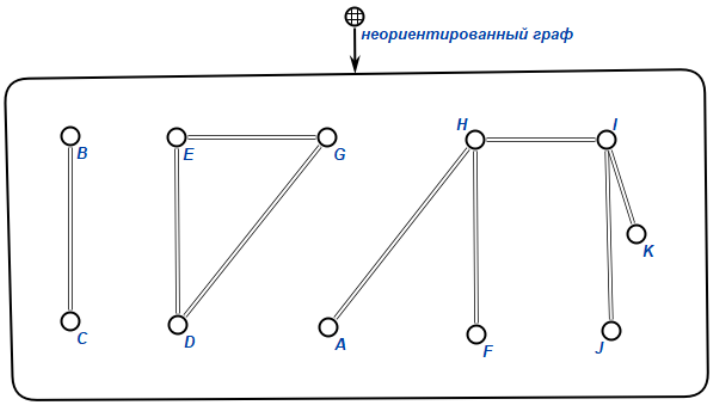
Будут найдены компоненты связности A, B, C и D, E, F, G.



2.2 Тест 2

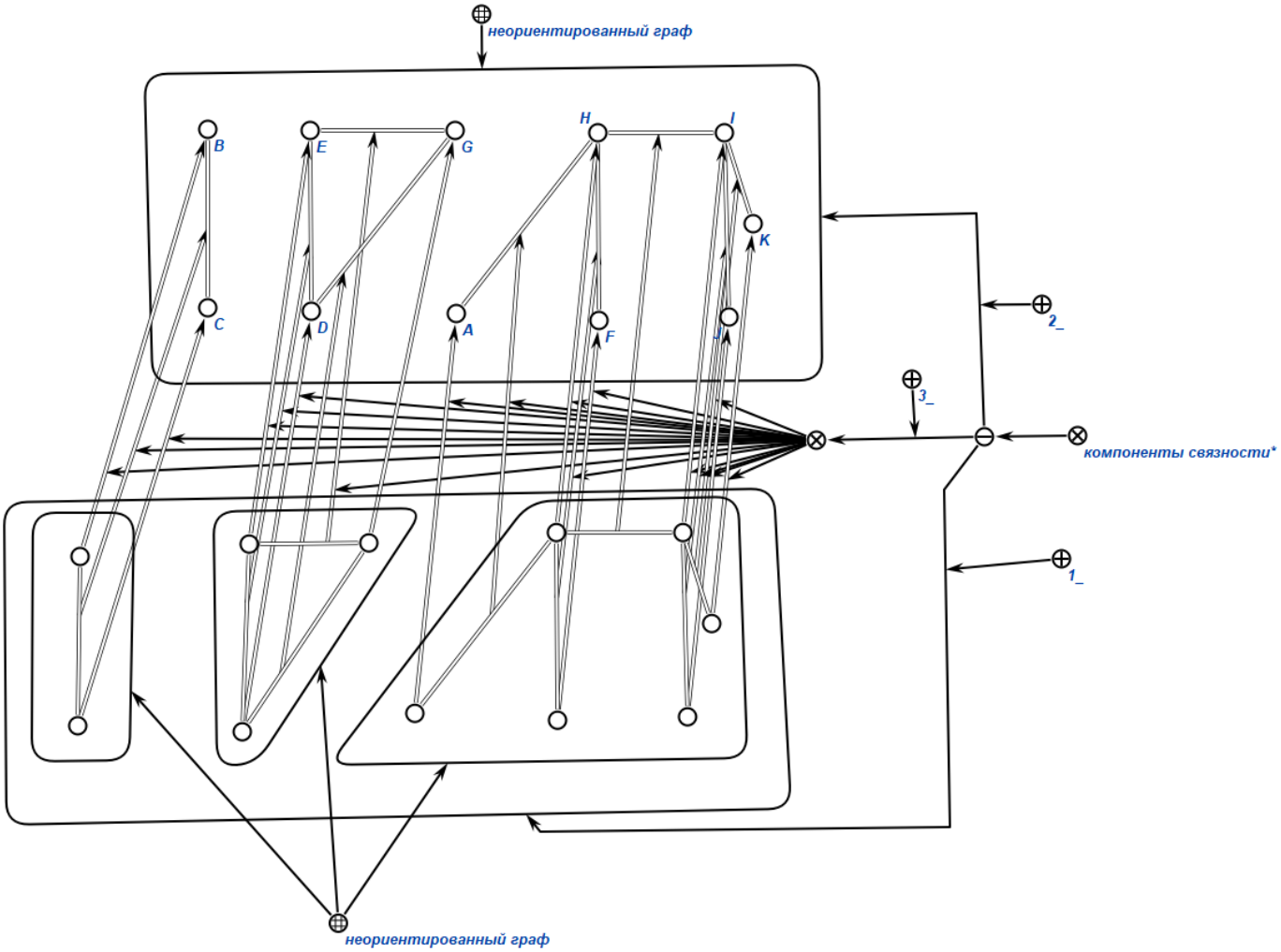
Вход:

Необходимо найти компоненты связности в неориентированном графе.



Выход:

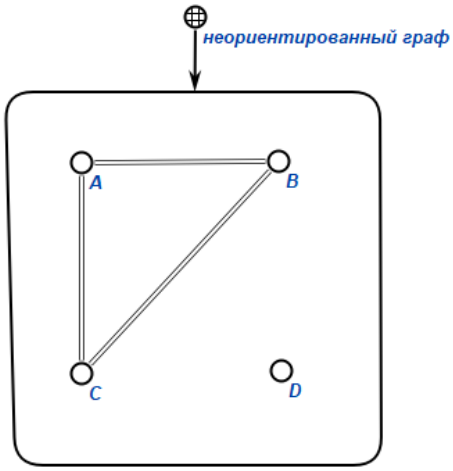
Будут найдены компоненты связности B, C; D, E, G и A, H, F, I, K, J.



2.3 Тест 3

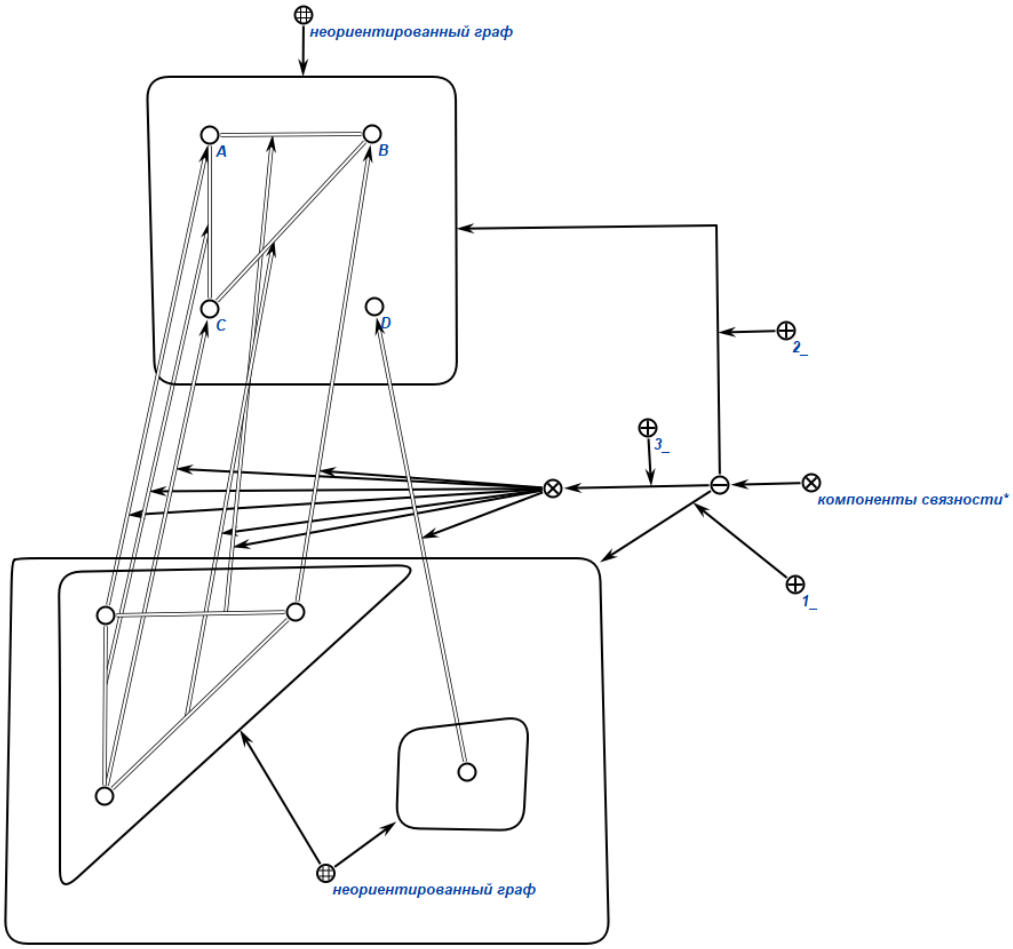
Вход:

Необходимо найти компоненты связности в неориентированном графе.



Выход:

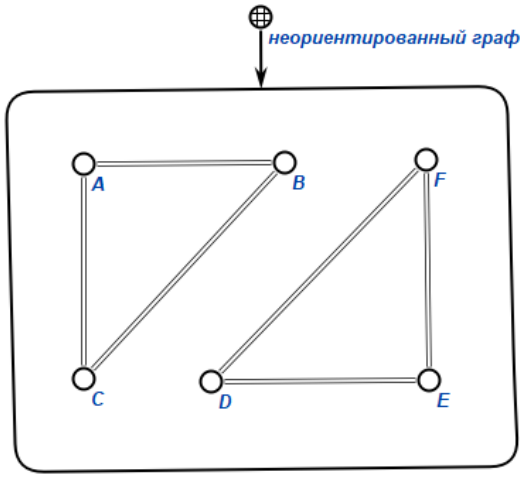
Будут найдены компоненты связности A, B, C и D.



2.4 Тест 4

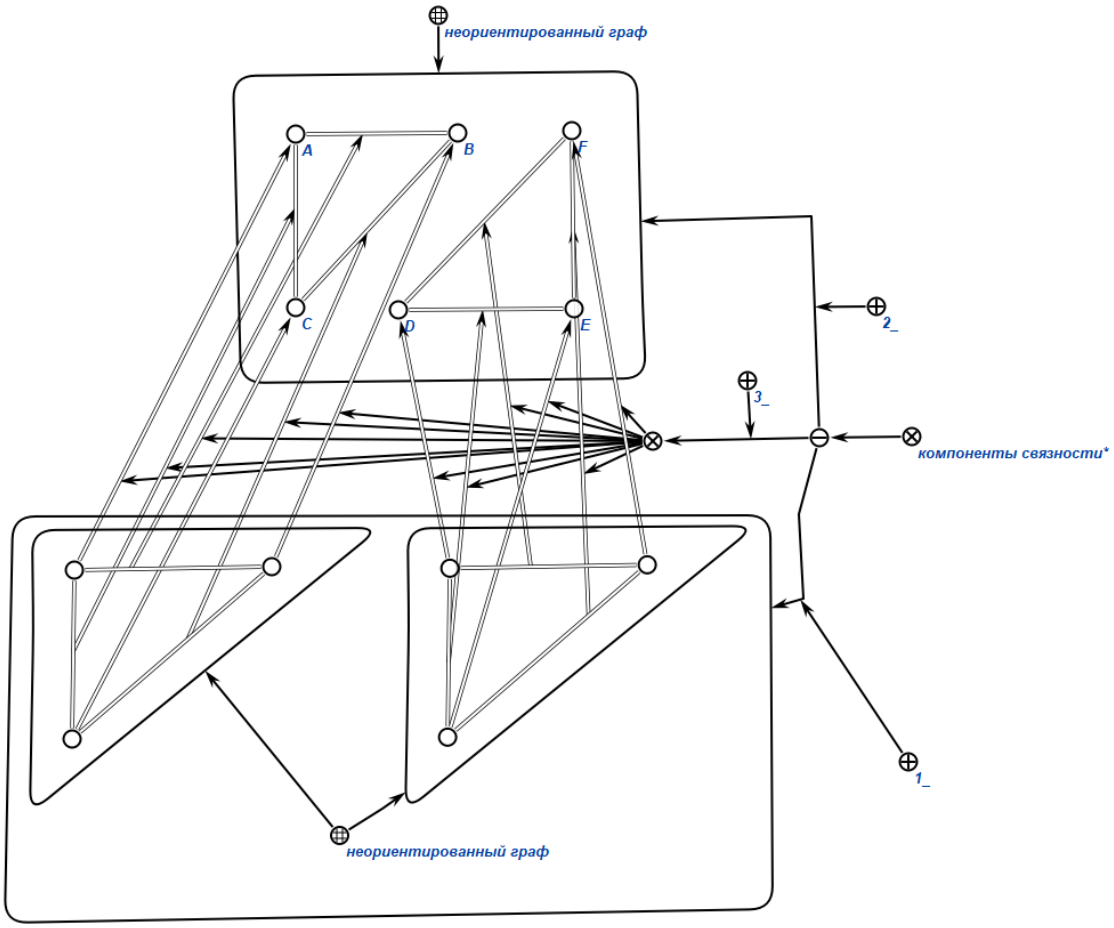
Вход:

Необходимо найти компоненты связности в неориентированном графе.



Выход:

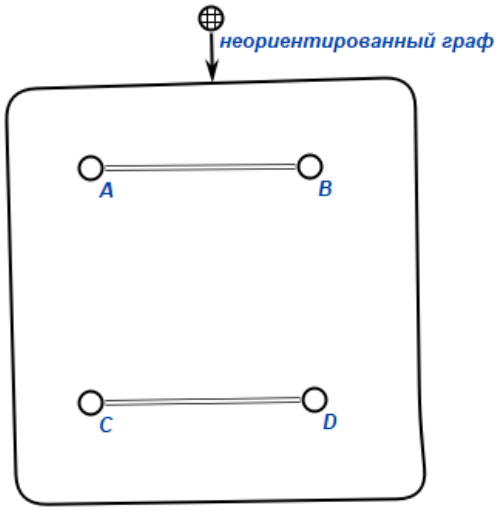
Будут найдены компоненты связности A, B, C и D, F, E.



2.5 Тест 5

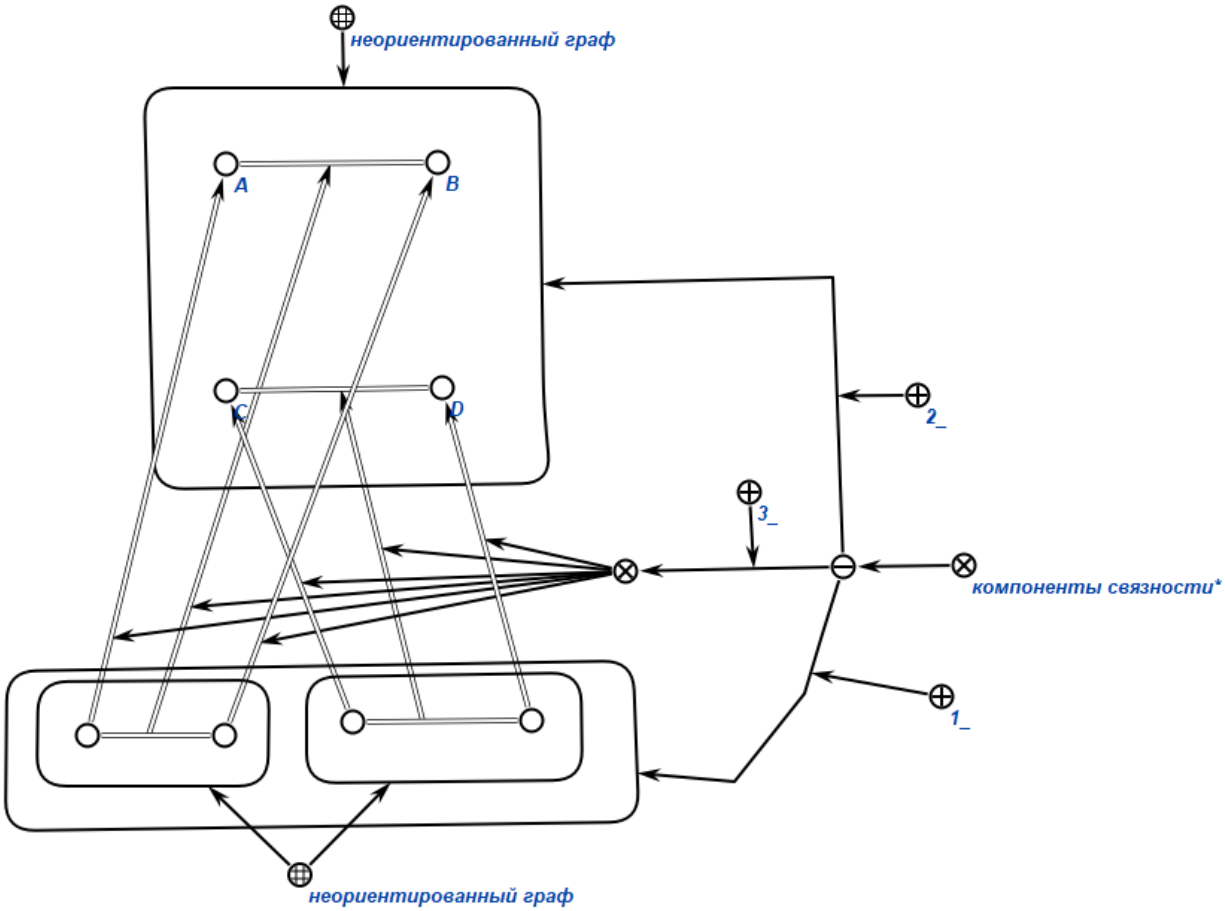
Вход:

Необходимо найти компоненты связности в неориентированном графе.



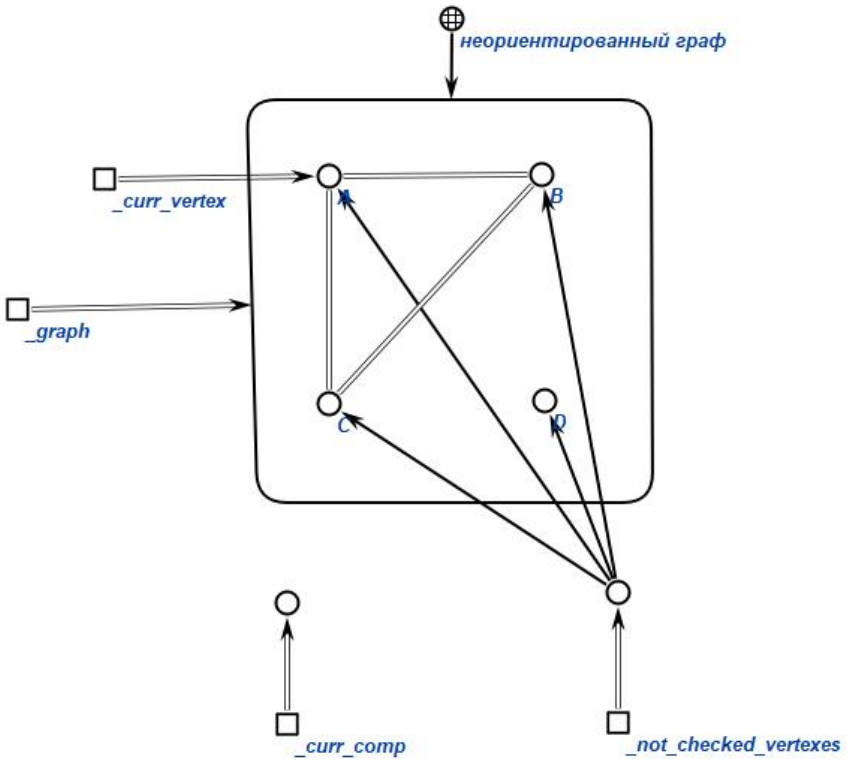
Выход:

Будут найдены компоненты связности A, B и D, C.



3 Описание алгоритма

1. Задание входного графа, неориентированных множеств неиспользованных вершин и текущей компоненты связности



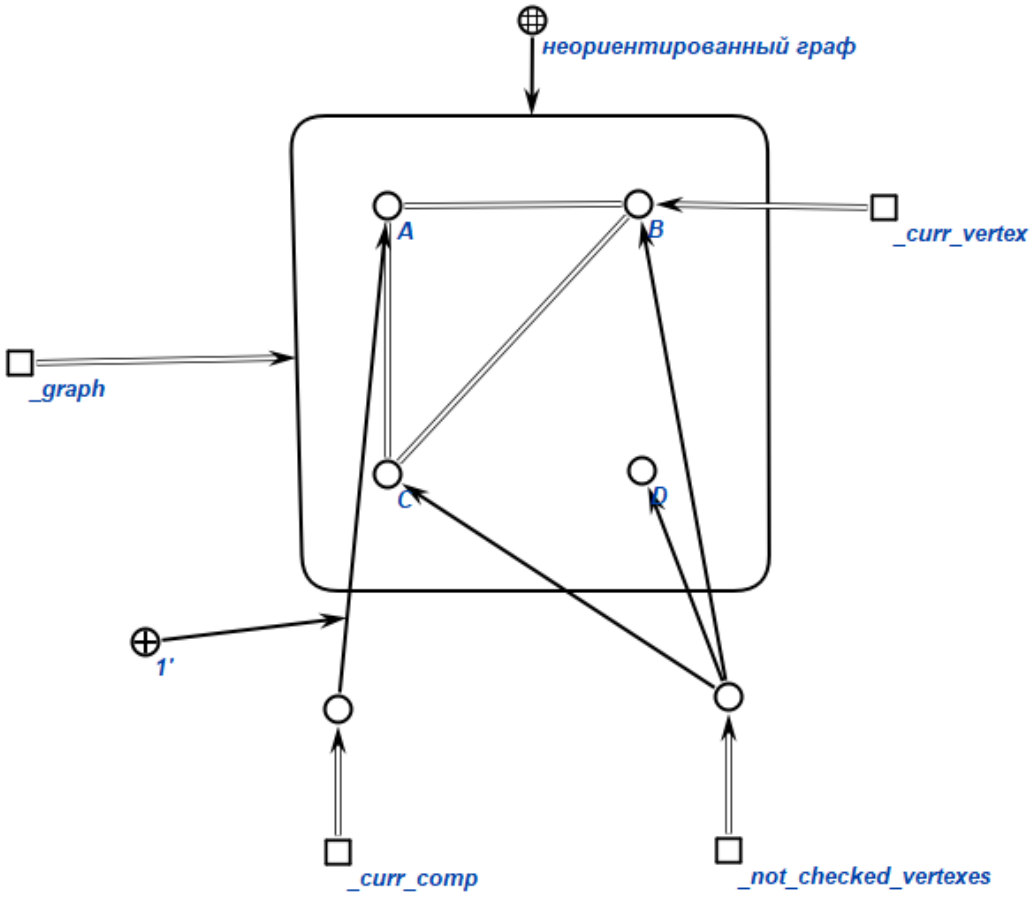
Переменные изменятся следующим образом:

- _graph получит в качестве значения sc-узел неориентированного графа;
- _curr_vertex получит в качестве значения вершину A, которая будет начальной в поиске элементов рассматриваемой компоненты связности
- _not_checked_vertexes получит в качестве значения множество непроверенных

вершин обрабатываемого графа. Каждый раз нам следует проверять, остались ли еще неиспользованные вершины, является ли следующая в рассмотрении вершина неиспользованной. Мы не можем использовать одинаковые вершины несколько раз и в нескольких компонентах. Также будем использовать это множество для начала построения новых компонент связности.

- _curr_comp получит в качестве значения множество текущей рассматриваемой компоненты связности. Будем добавлять новые элементы, относящиеся к этой компоненте, в это множество.

2. Добавление в ориентированное множество текущей компоненты вершины A



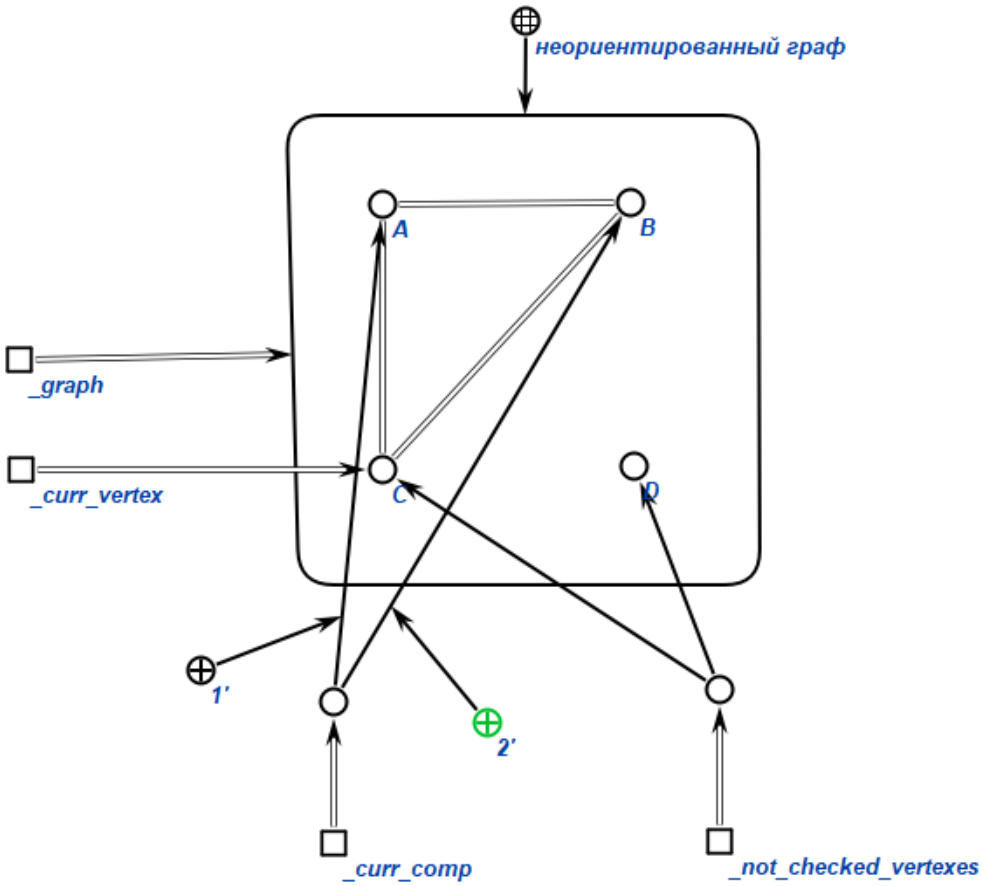
Добавим первую вершину A во множество текущей компоненты с порядковым номером 1.

Удаляем вершину A из множества неиспользованных вершин.

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

Переменной _curr_vertex задаем новое значение рассматриваемой вершины – вершину B.

3. Добавление в ориентированное множество текущей компоненты вершины B



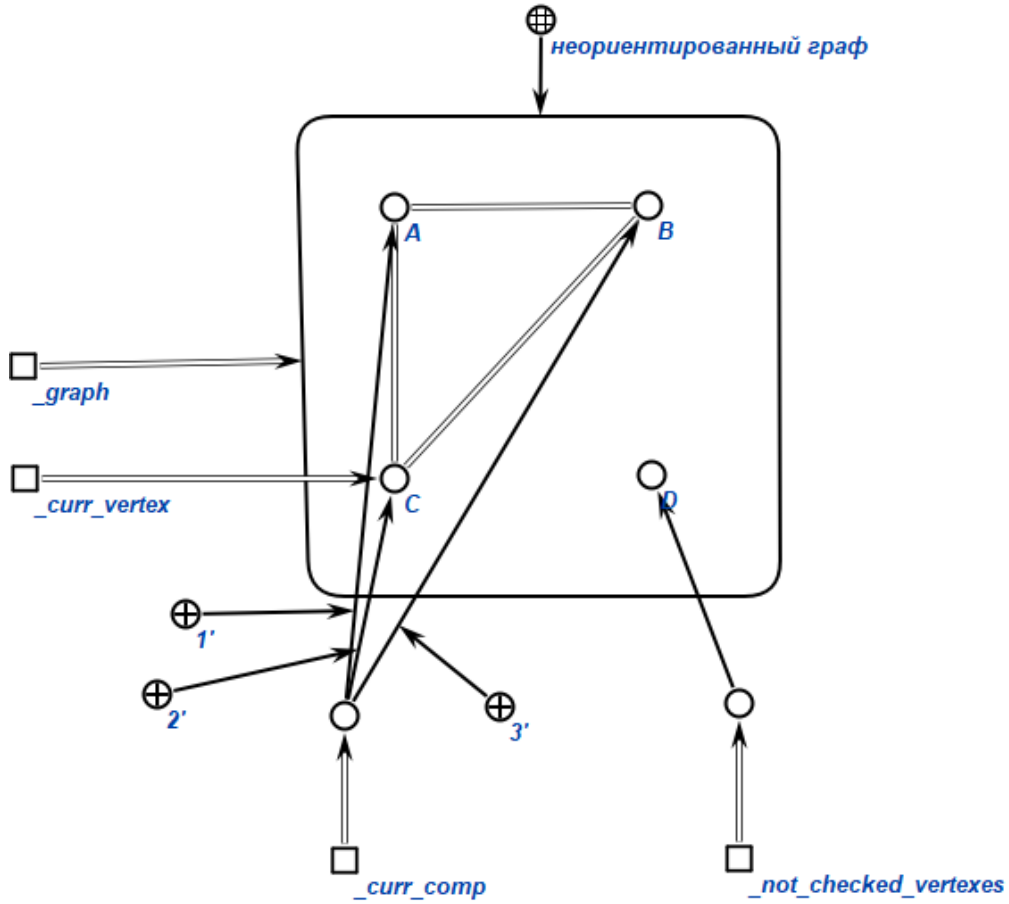
Добавим первую вершину B во множество текущей компоненты с порядковым номером 2.

Удаляем вершину B из множества неиспользованных вершин.

По множеству неиспользованных вершин проверяем, с какими вершинами имеет связи вершина B. Это вершина C. Рассмотрим следующей вершину C.

Переменной _curr_vertex задаем новое значение рассматриваемой вершины – вершину C.

4. Добавление во множество текущей компоненты вершины B



Добавим первую вершину C во множество текущей компоненты с порядковым номером 3.

Удаляем вершину C из множества неиспользованных вершин.

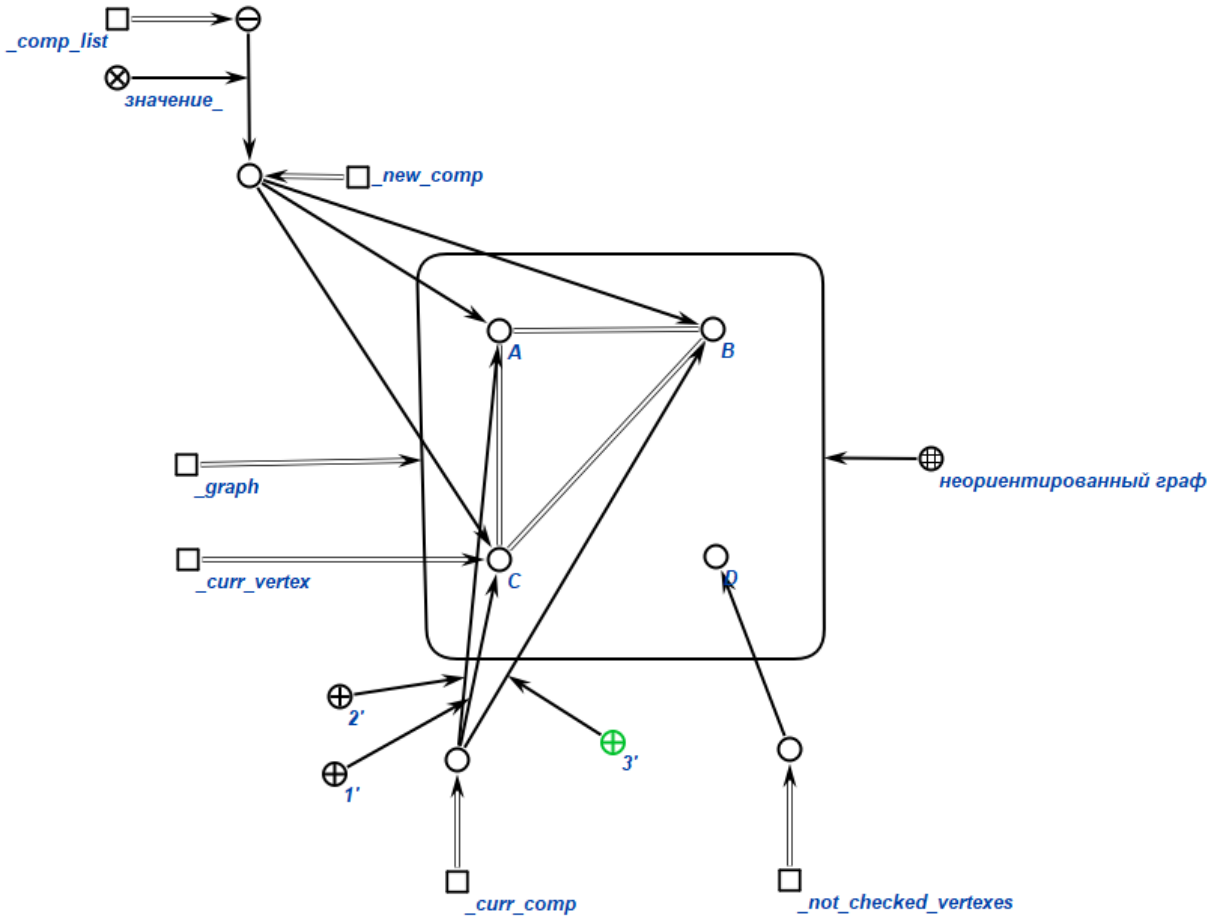
По множеству неиспользованных вершин проверяем, с какими вершинами имеет связи вершина C. Вершина C не имеет связи ни с одной вершиной из множества неиспользованных вершин.

Возвращаемся к рассмотрению вершины B, стоящей предыдущей в ориентированном множестве. Она не имеет связи ни с одной вершиной из множества неиспользованных вершин.

Возвращаемся к рассмотрению вершины A, стоящей предыдущей в ориентированном множестве. Она не имеет связи ни с одной вершиной из множества неиспользованных вершин. Вершина C, с которой вершина A имела связь, про что указано в шаге 2, уже добавлена во множество текущей компоненты.

Построение текущей компоненты связности закончено.

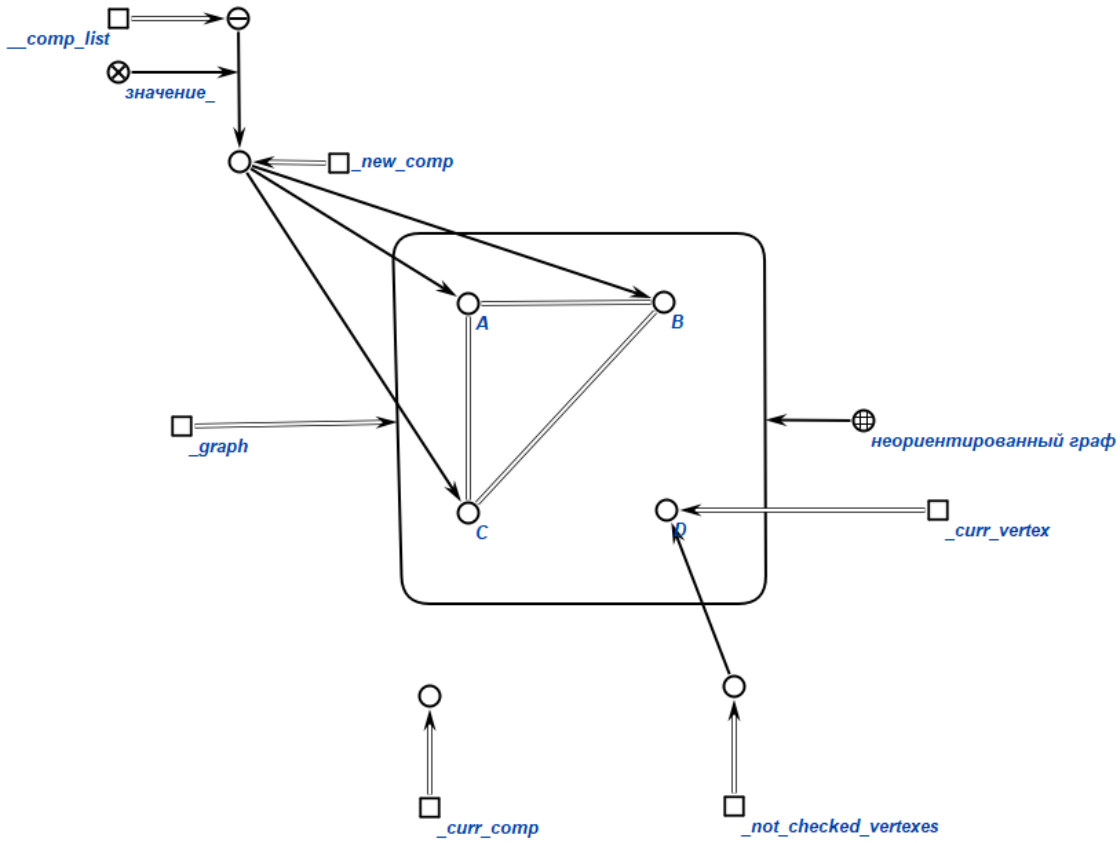
5. Создание списка компонент связности, добавление первого элемента в списке



На этом этапе программа создает первую компоненту связности из списка компонент. Первая компонента содержит все элементы из множества _curr_comp, где, как указано ранее, хранится завершенная компонента связности. Переменная _new_comp получает в качестве значения созданную компоненту, и в будущем будет всегда указывать на вновь созданную компоненту.

Переменная _comp_list указывает на список полученных компонент.

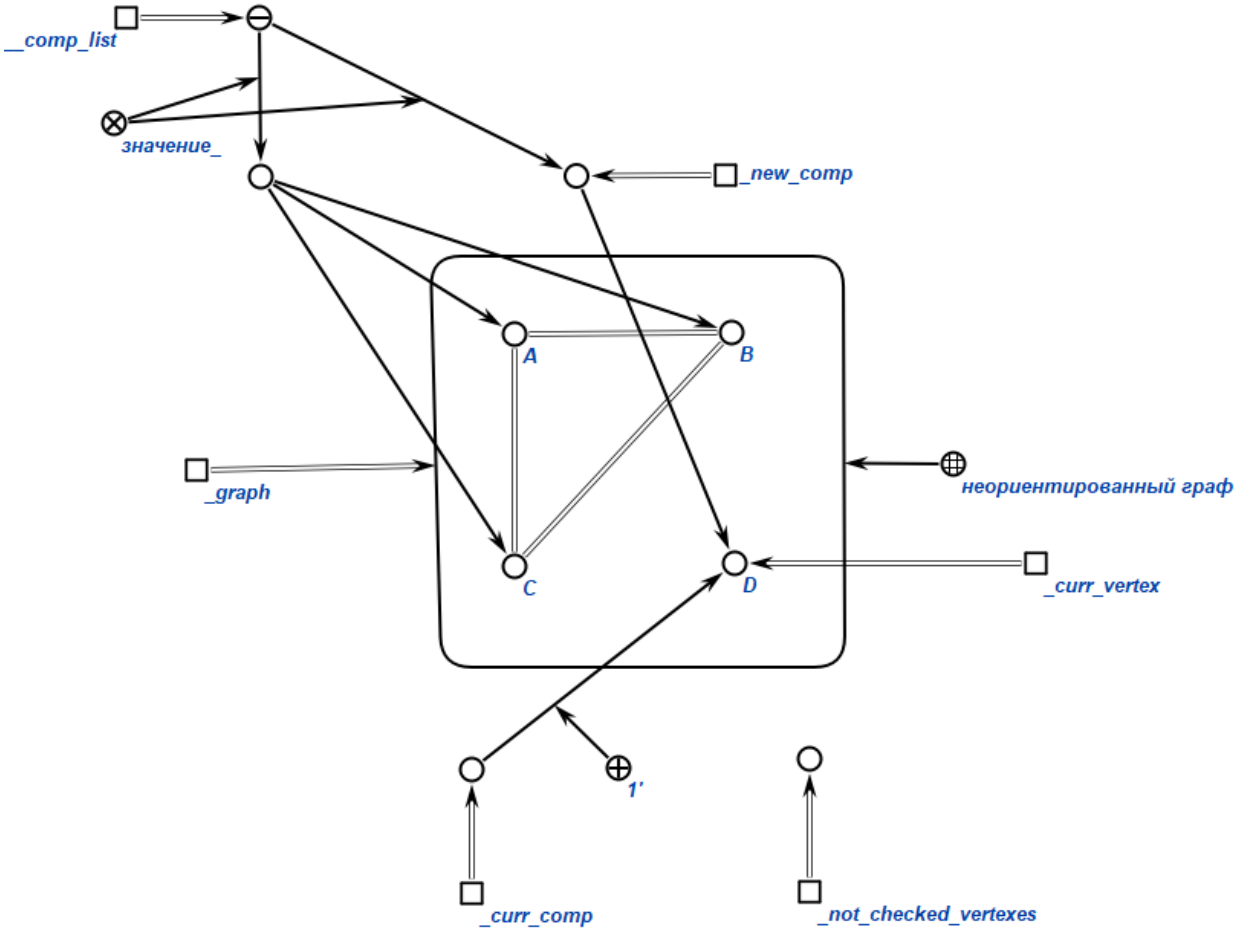
6. Начало рассмотрения новой компоненты связности



Переменной _curr_vertex задаем новое значение рассматриваемой вершины из множества неиспользованных вершин – вершину D.

Множество текущей рассматриваемой компоненты очищаем, так как начинаем построение новой компоненты.

7. Добавление во множество текущей компоненты вершины D



Добавим первую вершину D во множество текущей компоненты с порядковым номером 1.

Удаляем вершину D из множества неиспользованных вершин.

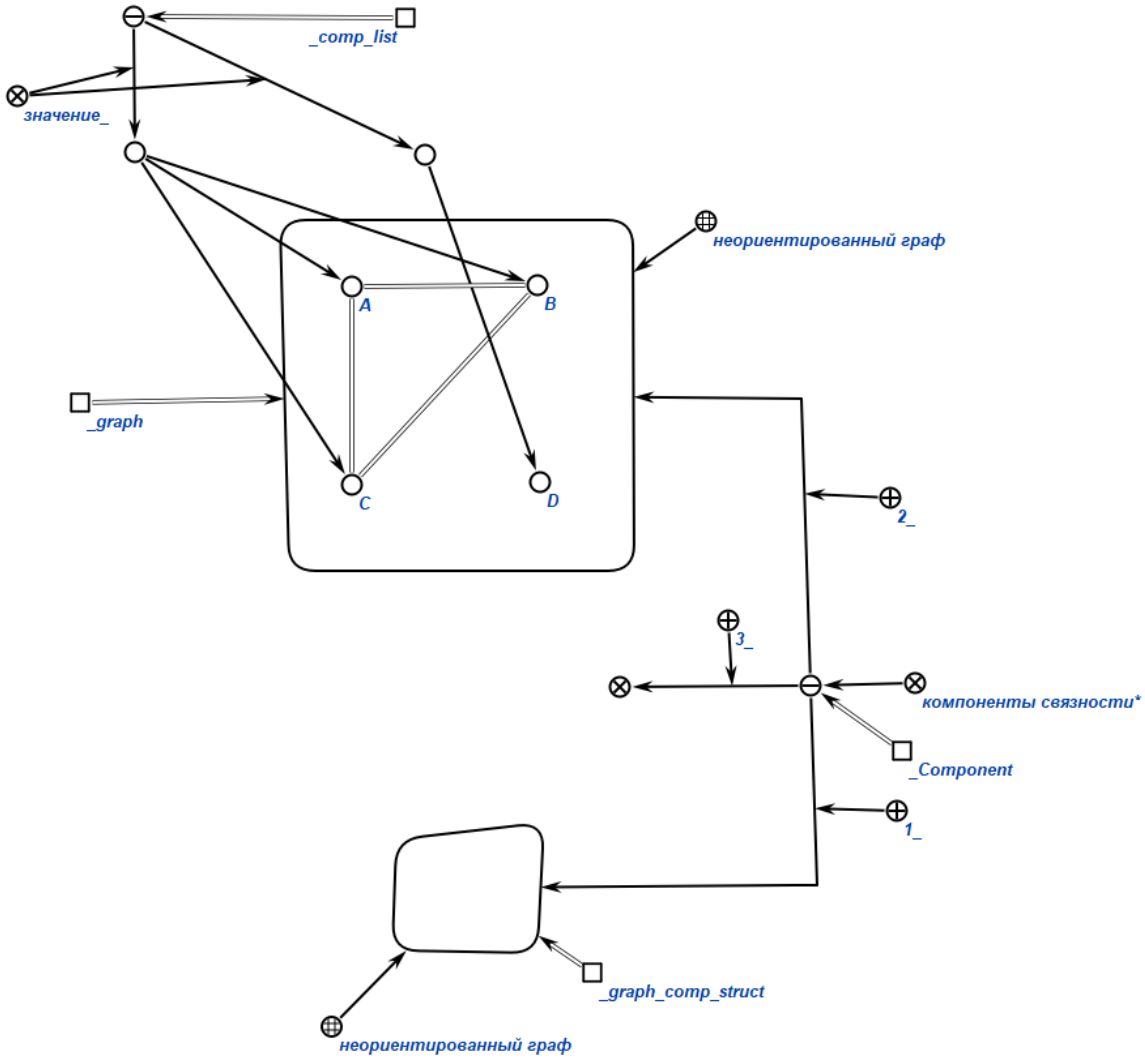
По множеству неиспользованных вершин проверяем, с какими вершинами имеет связи вершина D. Вершина D не имеет связи ни с одной вершиной из множества неиспользованных вершин. Множество неиспользованных вершин пусто, следовательно, построение текущей компоненты завершено, а начать построение новой компоненты не будет являться возможным.

Формируем новую компоненту связности из множества текущей компоненты. Она состоит из единственной вершины – D.

Переменная _new_comp – созданную компоненту, а список компонент получает новое множество.

Удаляем множество неиспользованных вершин, множество текущей компоненты связности, переменной, хранящей новую созданную компоненту связности в списке компонент связности.

8. Создание связки отношения компоненты связности*



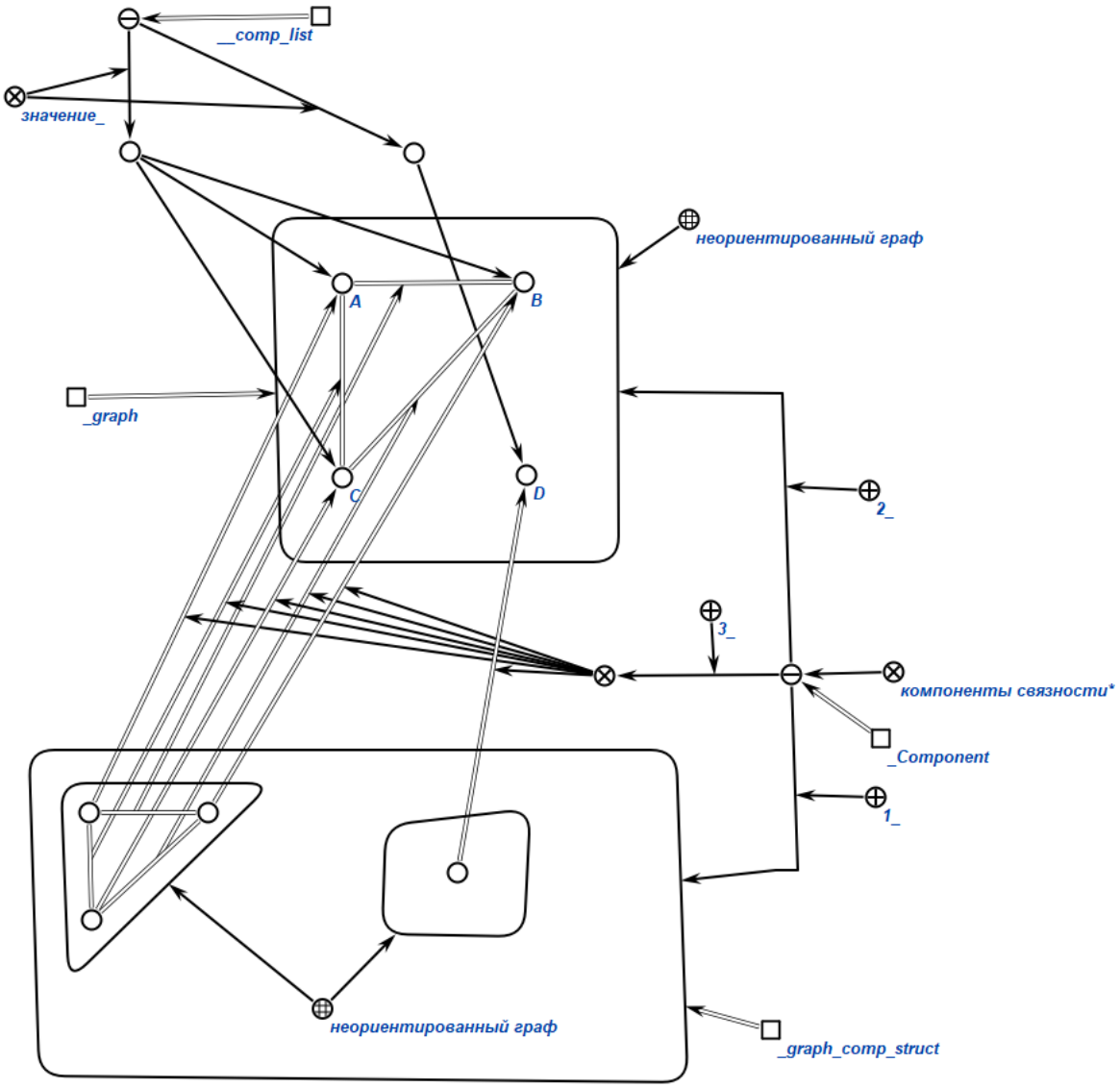
Можем получить ответ, так как множество неиспользованных вершин осталось пустым и было удалено, а мы получили два множества, хранящих две завершенные компоненты связности.

Начинаем генерацию компонент связности.

Создадим связку отношения компоненты связности* и установим ее в качестве значения переменной _Component.

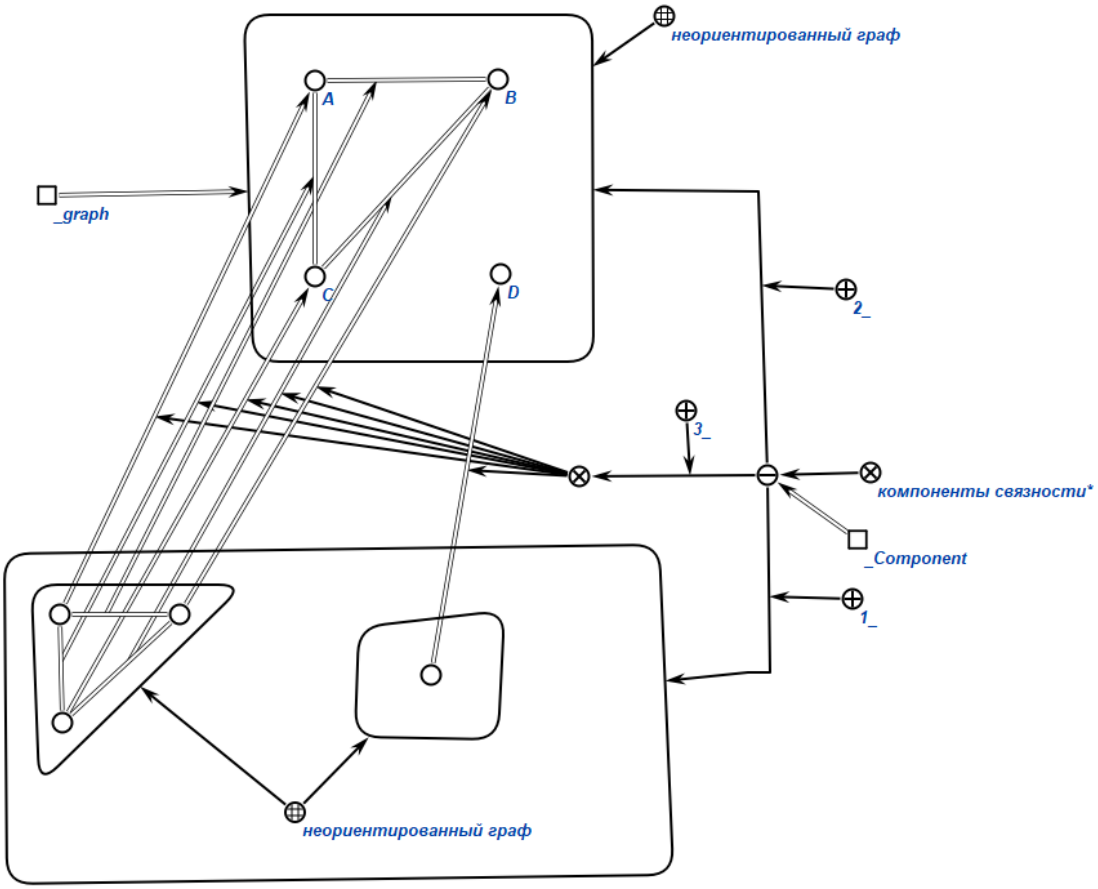
Переменная _graph_comp_struct получит в качестве значения неориентированный граф.

9. Добавление элементов компоненты связности и создание посещения элементов



Добавим в структуру компонент связности все компоненты, хранящихся в списке, на которое указывает _comp_list_head. Одна компонента связности состоит из единственного элемента, вторая состоит из трех элементов. Добавляем связующие ребра.

10. Удаление списка компонент связности. 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.