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 проверки неориентированного графа на двусвязность

Выполнил Ломонос И.Н.
студент группы
021702

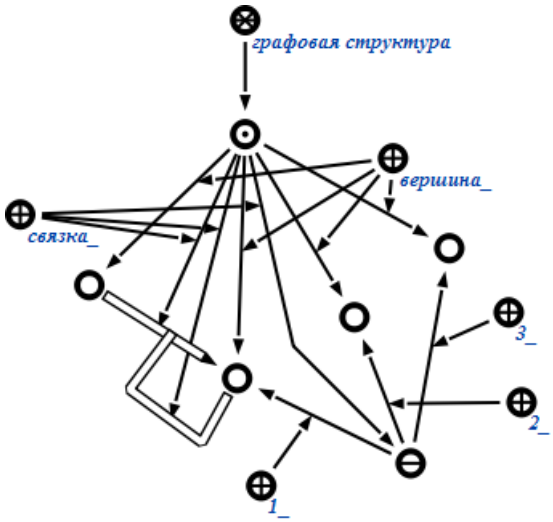
Проверил Юрков А. А.

Минск 2021

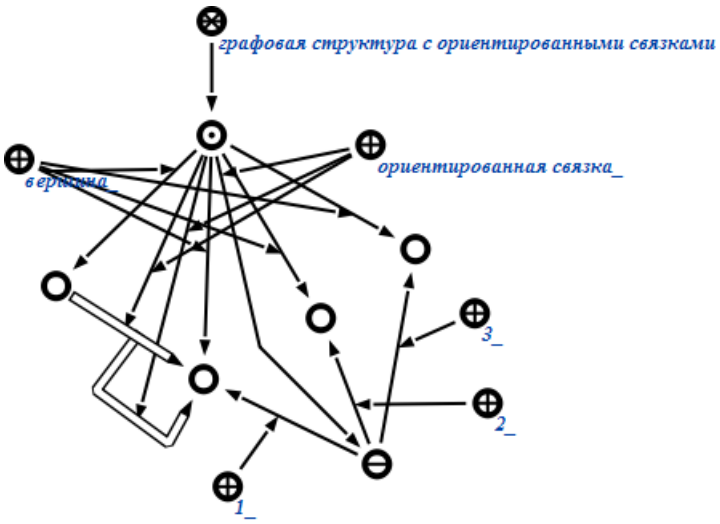
Цель: Получить навыки формализации и обработки информации с использованием семантических сетей

Задача: проверить неориентированный граф на двусвязность

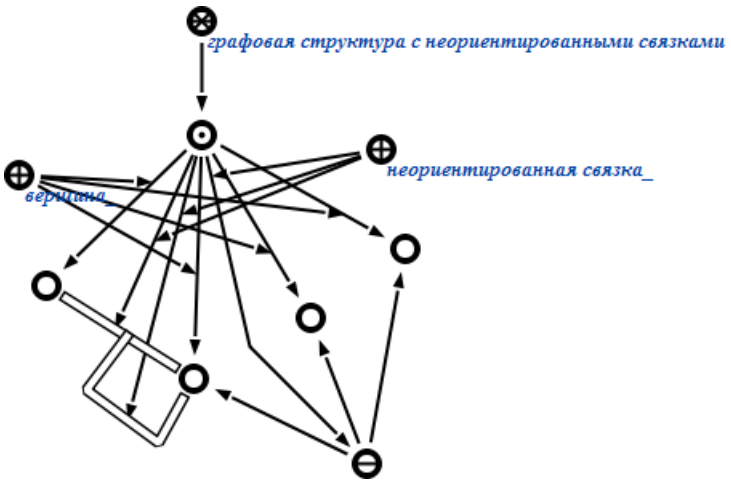
1. Графовая структура (абсолютное понятие) - это такая одноуровневая реляционная структура, объекты которой могут играть роль либо вершины, либо связки:
    a. Вершина (относительное понятие, ролевое отношение);
    b. Связка (относительное понятие, ролевое отношение).



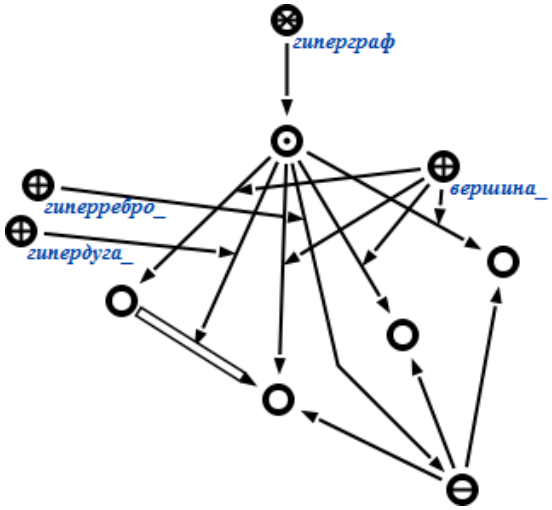
2. Графовая структура с ориентированными связками (абсолютное понятие)
    a. Ориентированная связка (относительное понятие, ролевое отношение) – связка, которая задается ориентированным множеством.



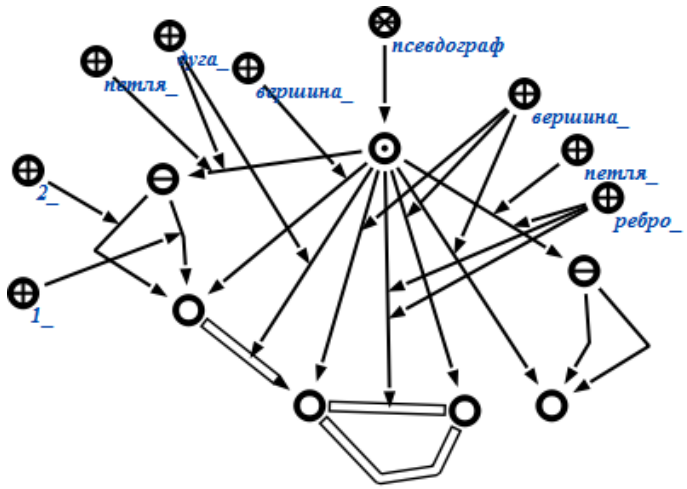
3. Графовая структура с неориентированными связками (абсолютное понятие)
    a. Неориентированная связка (относительное понятие, ролевое отношение) – связка, которая задается неориентированным множеством.



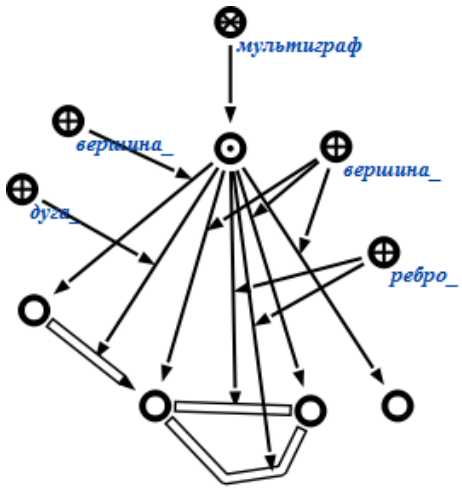
4. Гиперграф (абсолютное понятие) – это такая графовая структура, в которой связки могут связывать только вершины:
    a. Гиперсвязка (относительное понятие, ролевое отношение);
    b. Гипердуга (относительное понятие, ролевое отношение) – ориентированная гиперсвязка;
    c. Гиперребро (относительное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



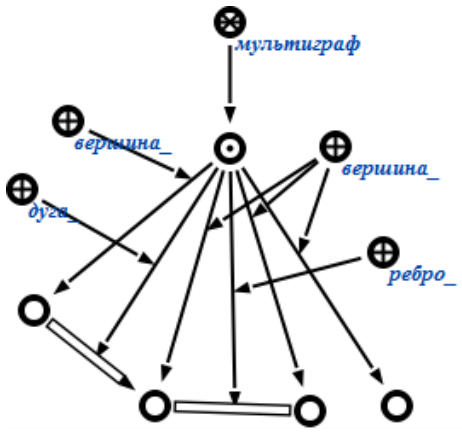
5. Псевдограф (абсолютное понятие) – это такой гиперграф, в котором все связки должны быть бинарными:
    a. Бинарная связка (относительное понятие, ролевое отношение) –гиперсвязка арности 2;
    b. Ребро (относительное понятие, ролевое отношение) –неориентированная гиперсвязка;
    c. Дуга (относительное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



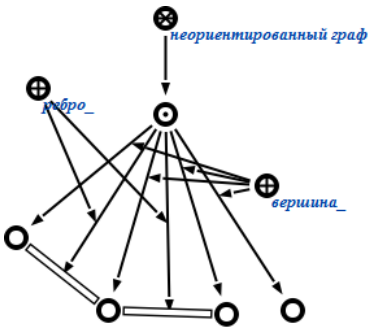
6. Мультиграф (абсолютное понятие) – это такой псевдограф, в котором не может быть петель:



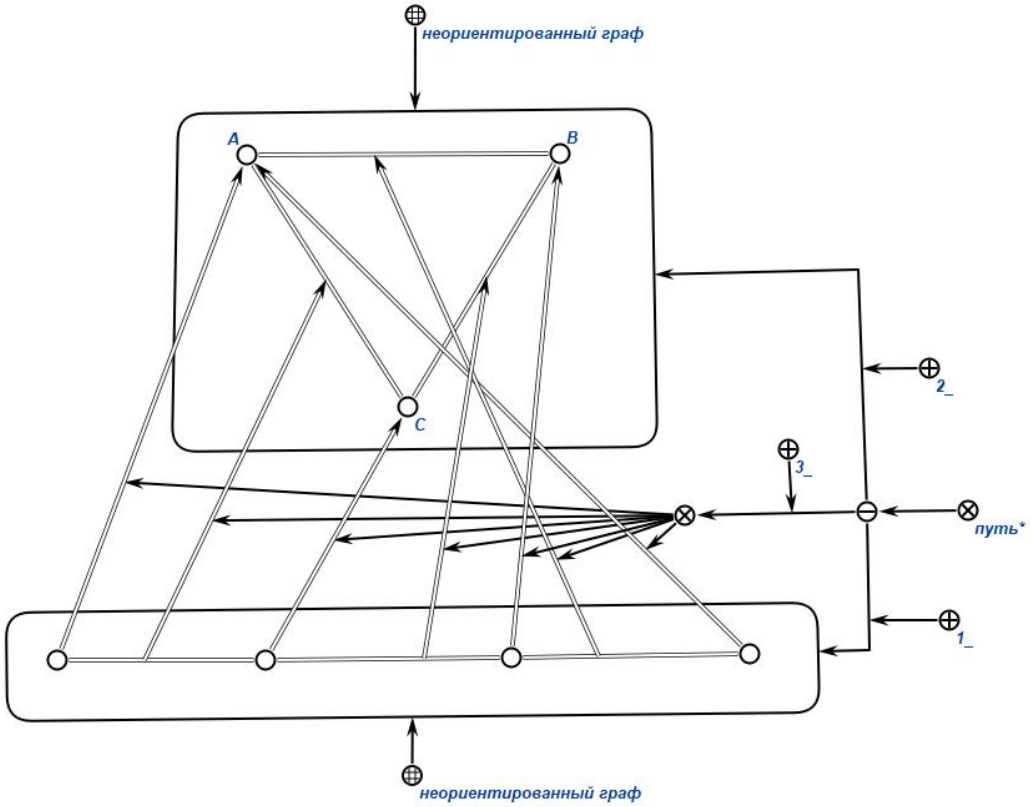
7. Граф (абсолютное понятие) – это такой мультиграф, в котором не может быть кратных связок, т.е. связок у которых первый и второй компоненты совпадают:



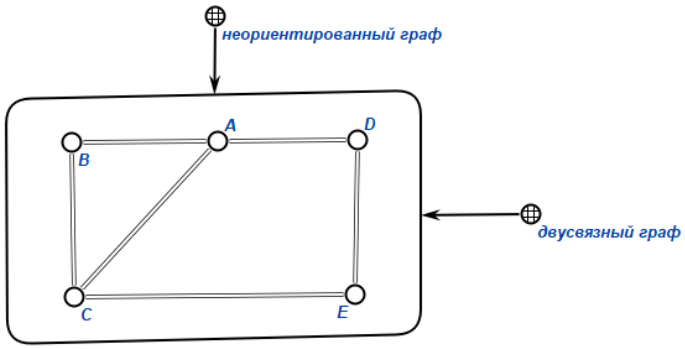
8. Неориентированный граф (абсолютное понятие) –это такой граф, в котором все связки являются ребрами:



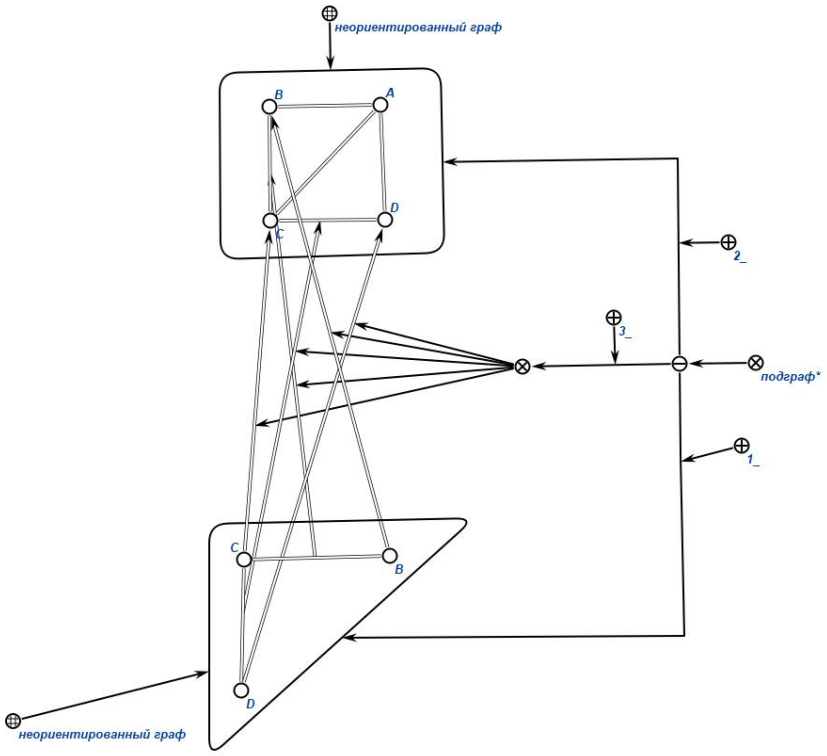
9. Связный граф (относительное понятие) – граф, у которого между любой парой вершин существует как минимум один путь:



10. Двусвязный граф (относительное понятие, бинарное отношение) – это связный граф, не теряющий связности при удалении любой вершины и всех инцидентных ей рёбер:



11. Подграф (относительное понятие) – это другой граф, образованный из подмножества вершин исходного графа вместе со всеми рёбрами, соединяющими пары вершин из этого подмножества. Для примера ниже подграф B,C,D:



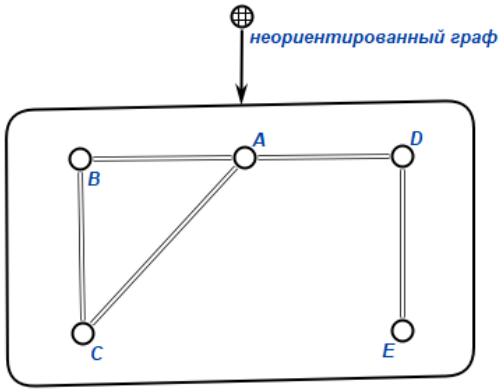
1 Тестовые примеры

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

1.1 Тест 1

Вход:

Необходимо определить, является ли входной граф двусвязным.



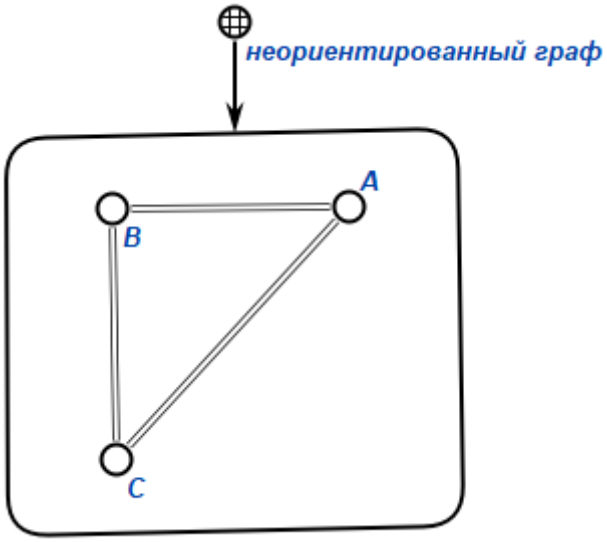
Выход:

Входной граф не двусвязный.

1.2 Тест 2

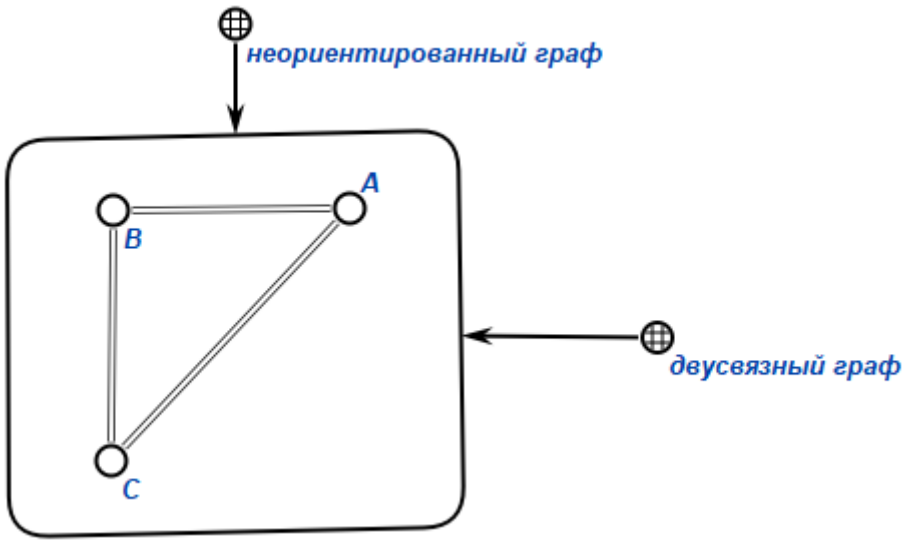
Вход:

Необходимо определить, является ли входной граф двусвязным.



Выход:

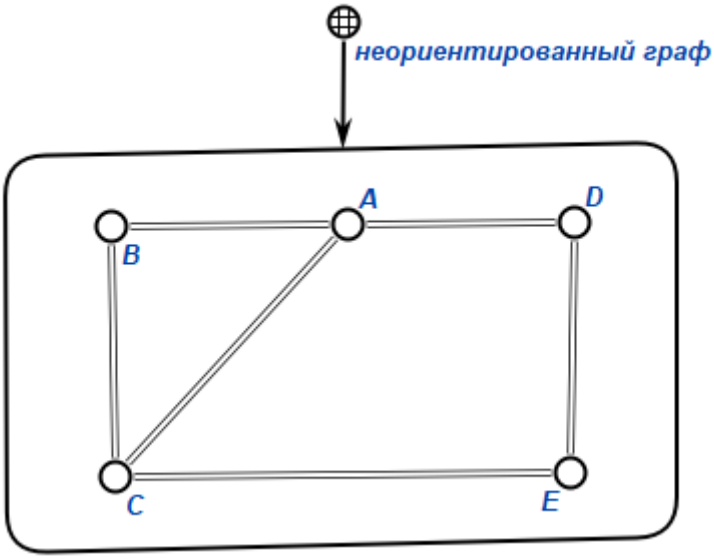
Входной граф является двусвязным.



1.3 Тест 3

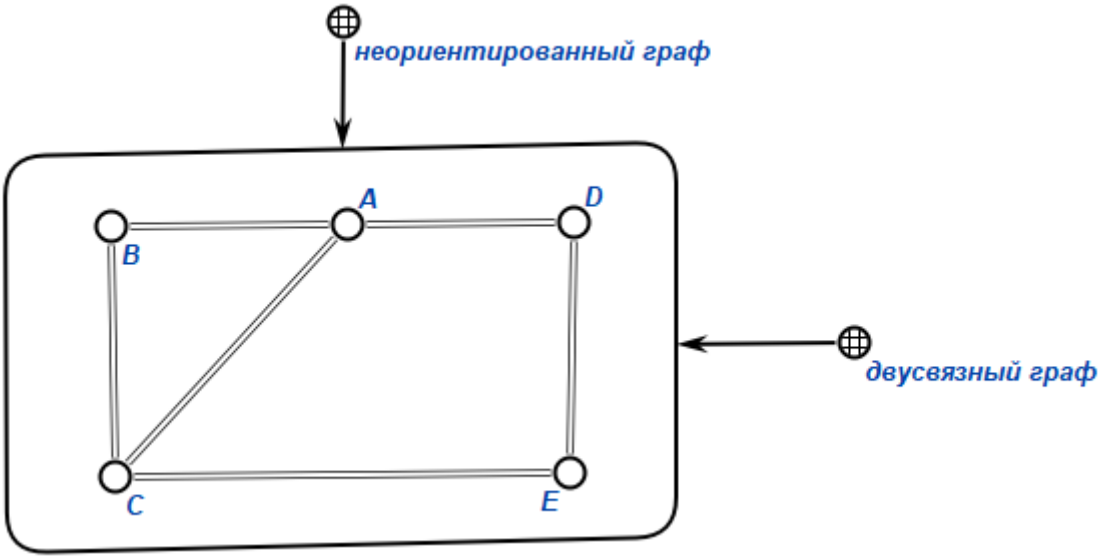
Вход:

Необходимо определить, является ли входной граф двусвязным.



Выход:

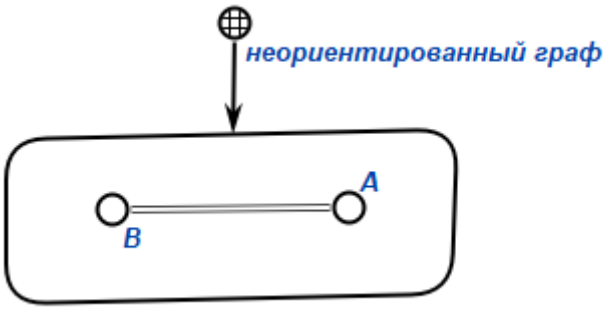
Входной граф является двусвязным.



1.4 Тест 4

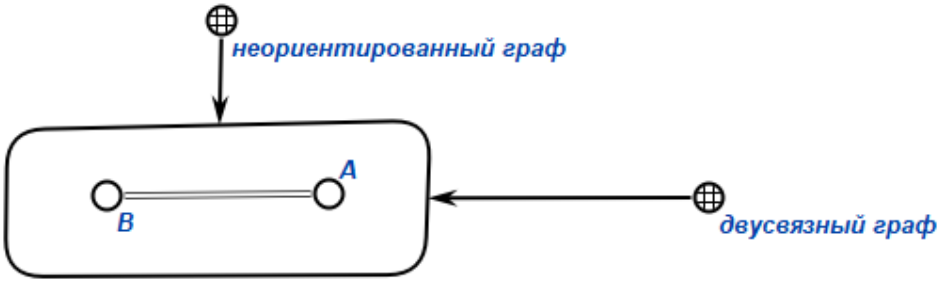
Вход:

Необходимо определить, является ли входной граф двусвязным.



Выход:

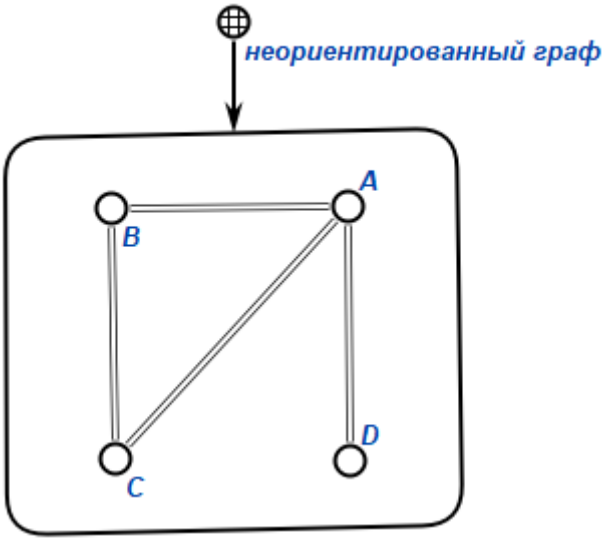
Входной граф является двусвязным.



1.5 Тест 5

Вход:

Необходимо определить, является ли входной граф двусвязным:

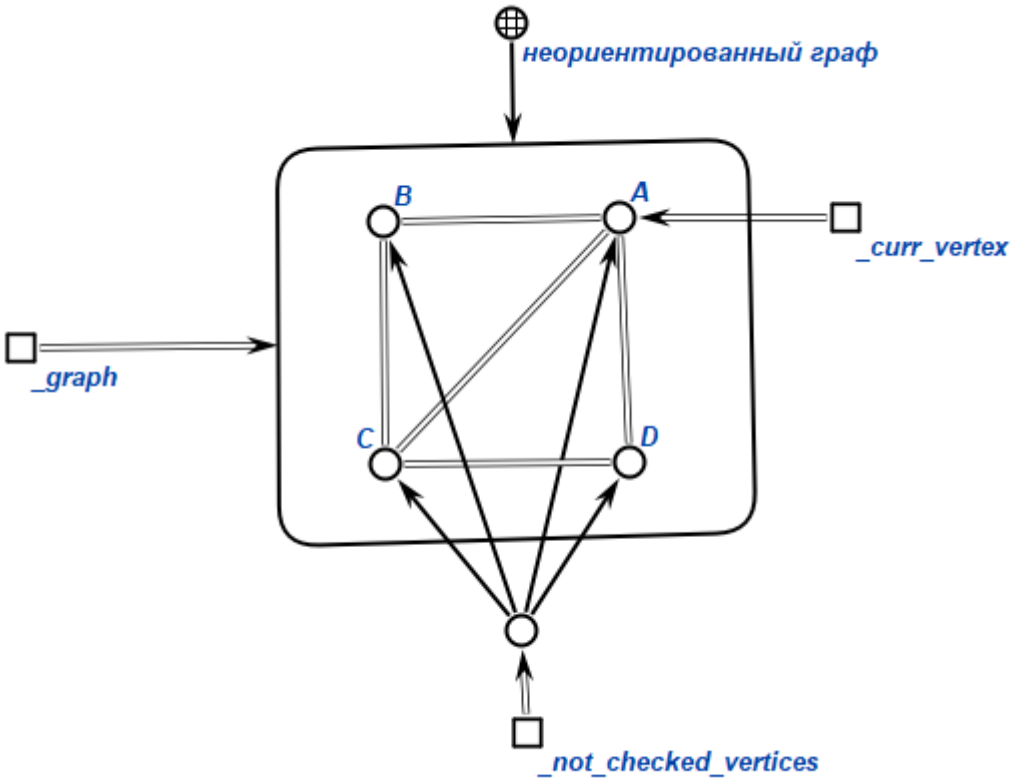


Выход:

Входной граф не двусвязный.

2 Описание алгоритма

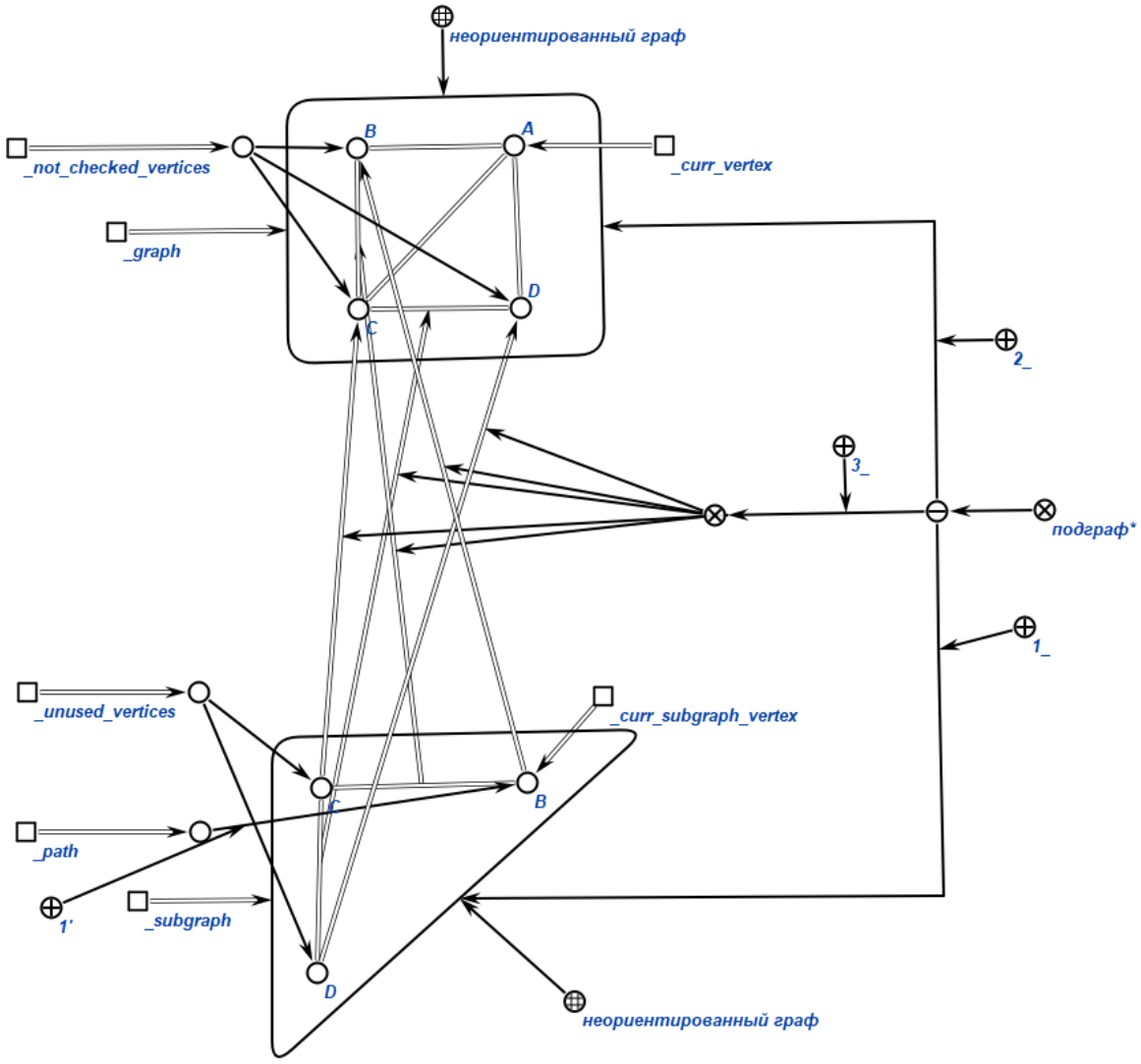
1. Задание входного графа, неориентированного множества неиспользованных вершин



Переменные изменятся следующим образом:

- _graph получит в качестве значения sc-узел неориентированного графа;
- _not_checked_vertices получит в качестве значения множество непроверенных вершин обрабатываемого графа. Для каждой вершины будет проводиться проверка, является ли подграф, полученный удалением данной вершины, связным. Если для какой-либо вершины проверка будет провалена, граф не является двусвязным по определению;
- _curr_vertex получит в качестве значения начальную вершину для начала проверки.

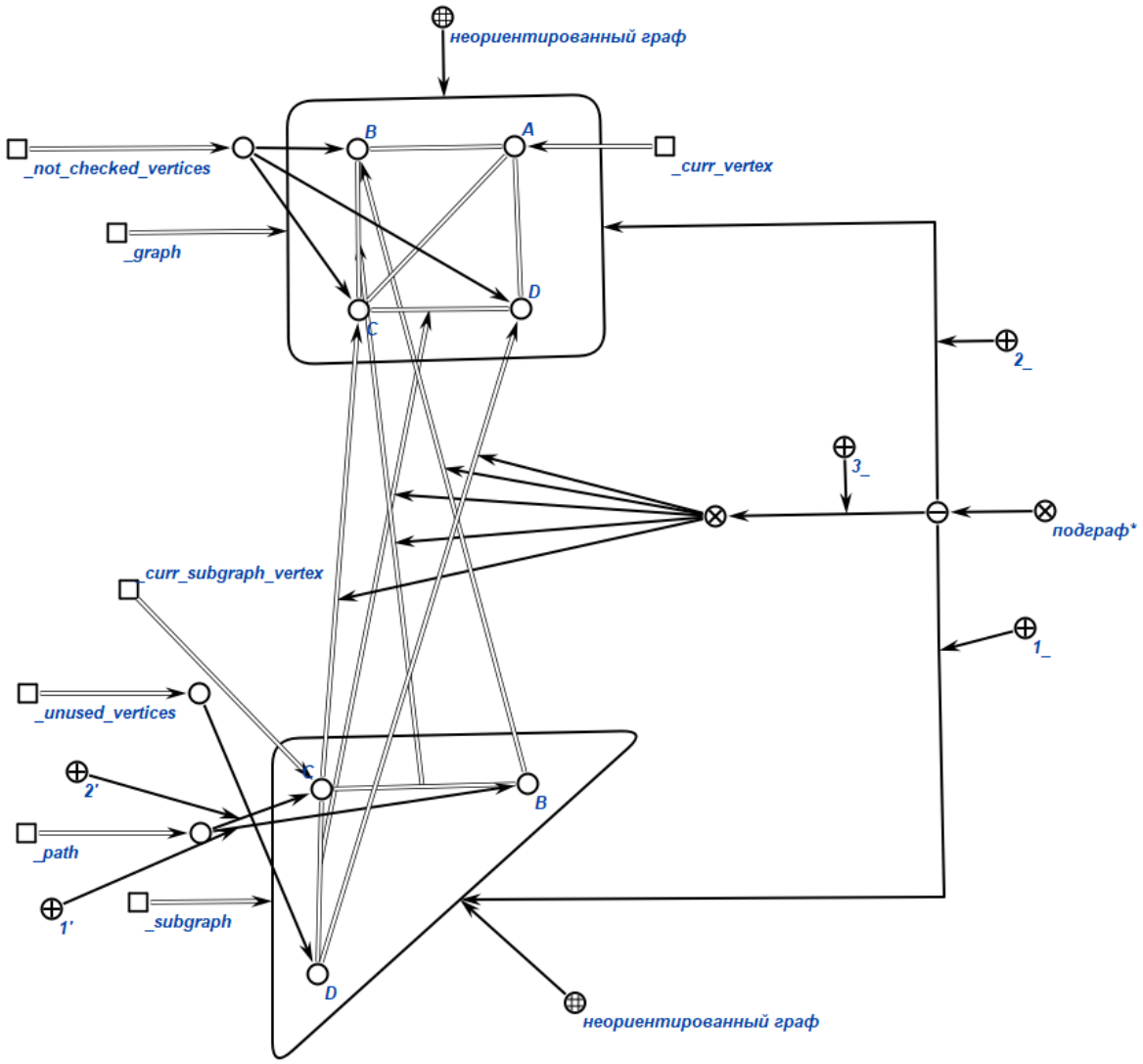
2. Создание связки отношения подграфа* и создание первого подграфа удалением вершины A



Начинаем проверку для вершины A. Построим подграф исходного графа, удалив вершину A и все инцидентные ей ребра. Полученный подграф зададим переменной _subgraph. Подграф нужно проверить для связность. Для этого будем искать путь, содержащий все вершины подграфа. Зададим множество _unused_vertices, чтобы проверять, все ли вершины подграфа были использованы. Ориентированное множество _path зададим для хранения текущего строящегося пути.

Начнем путь с вершины B, запишем B в _path с порядковым номером 1. Удалим ее из множества неиспользованных вершин и зададим в переменную _curr_subgraph_vertex. Из множества неиспользованных вершин вершина B имеет связь с вершиной C.

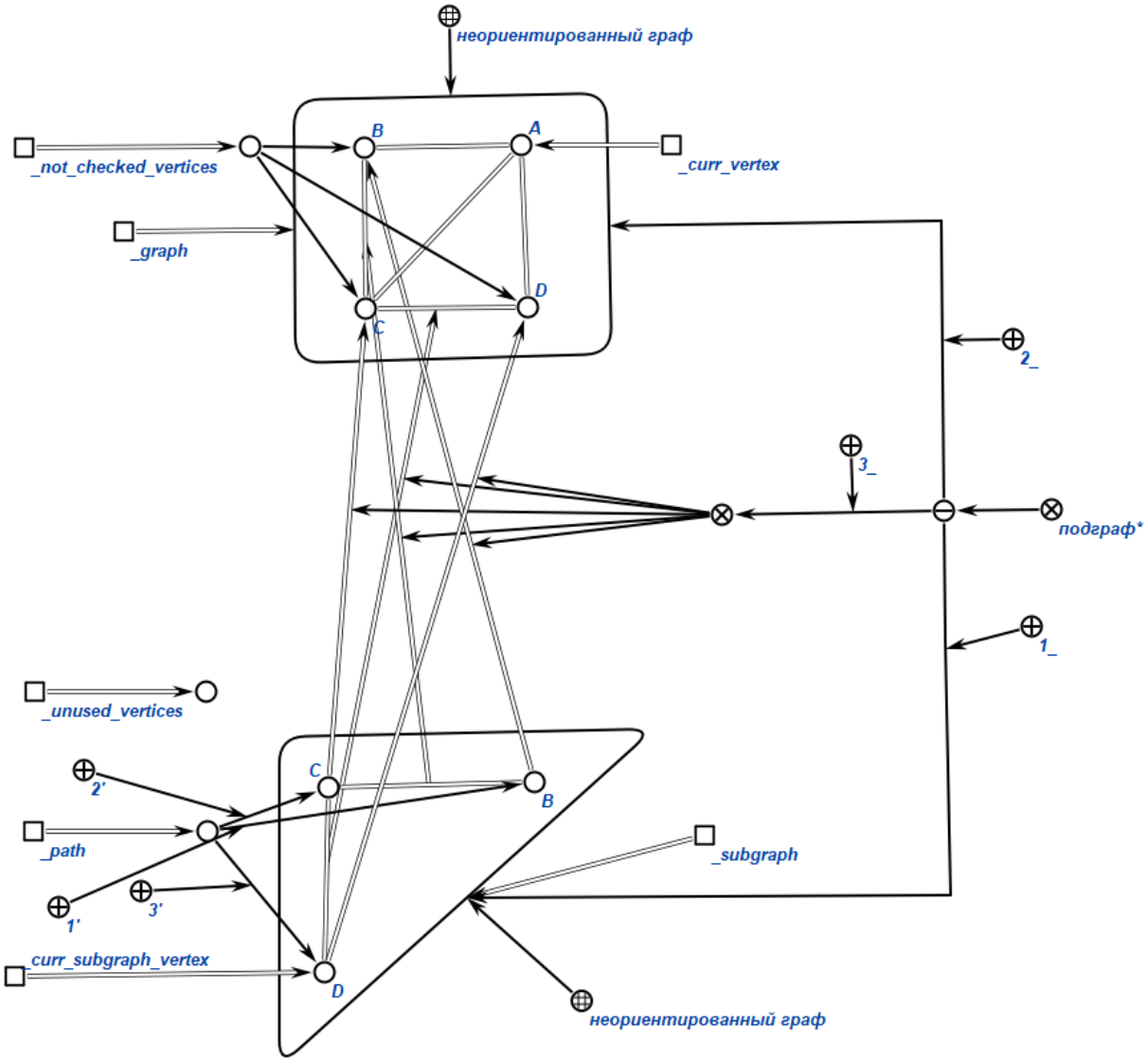
3. Проверка на связность подграфа. Добавление вершины C в ориентированное множество пути



Запишем C в _path с порядковым номером 2. Удалим ее из множества неиспользованных вершин и зададим в переменную _curr_subgraph_vertex.

Из множества неиспользованных вершин вершина C имеет связь с вершиной D.

4. Проверка на связность подграфа. Добавление вершины D в ориентированное множество пути

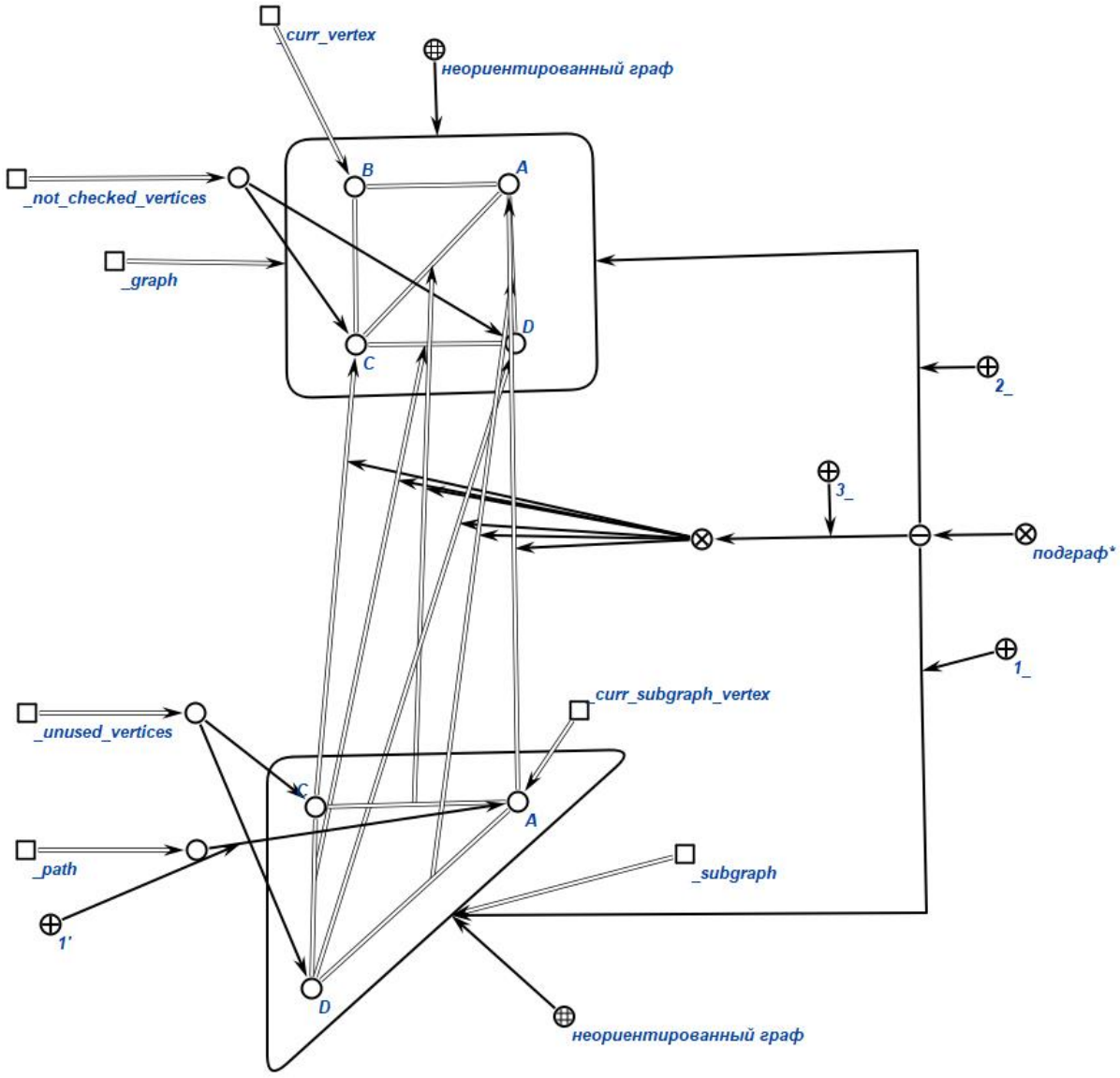


Запишем D в _path с порядковым номером 3. Удалим ее из множества неиспользованных вершин и зададим в переменную _curr_subgraph_vertex.

Множество неиспользованных вершин пусто. Можем сделать вывод, что подграф, полученный удалением вершины A и инцидентных ей ребер, получился связным.

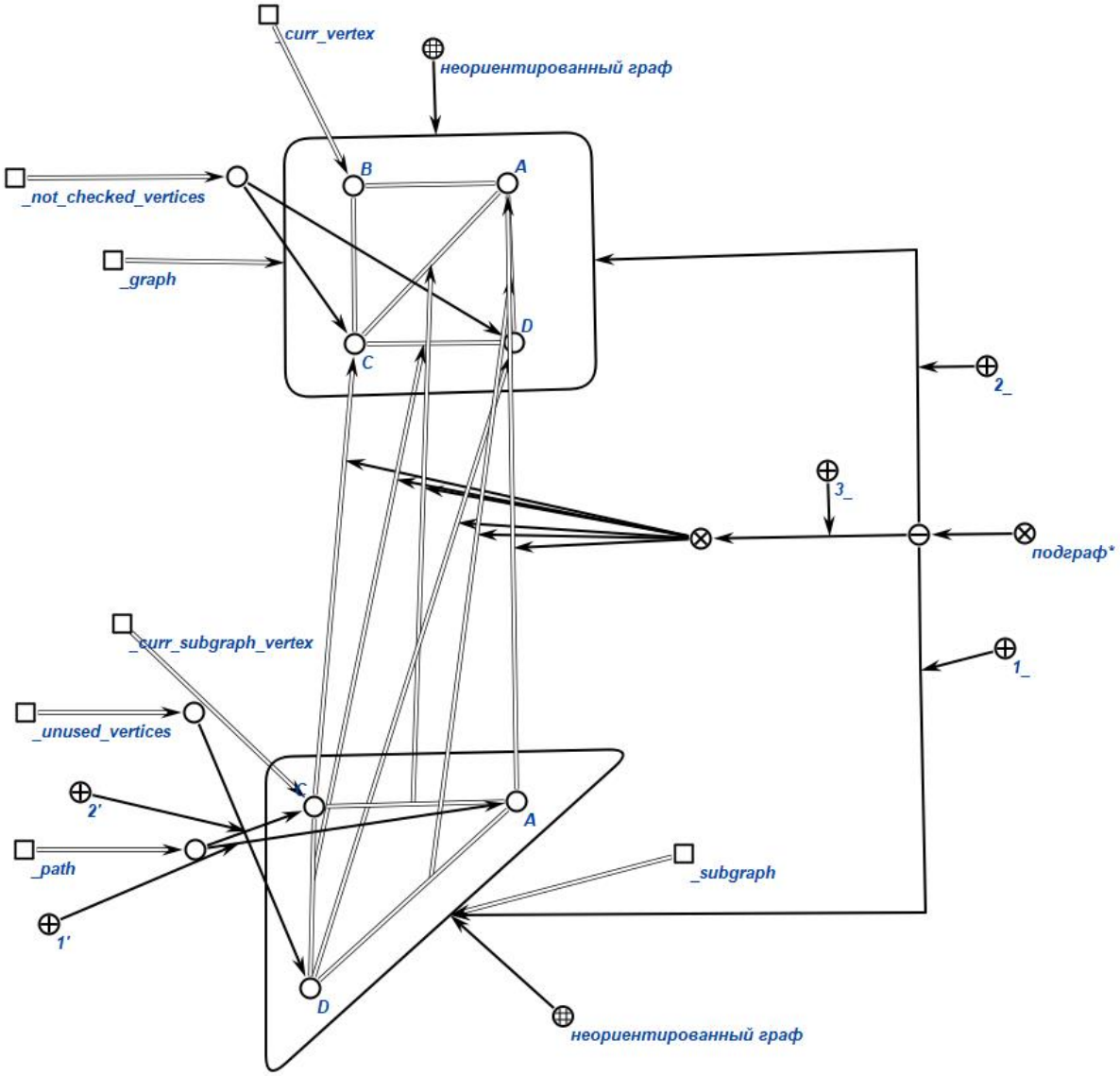
Продолжаем проверку для следующей вершины из множества _not_checked_vertices.

5. Проверка на связность подграфа. Добавление вершины A в ориентированное множество пути



Начинаем проверку для вершины B. Построим подграф исходного графа, удалив вершину C и все инцидентные ей ребра. Полученный подграф зададим переменной _subgraph. Начнем путь с вершины A, запишем A в _path с порядковым номером 1. Удалим ее из множества неиспользованных вершин и зададим в переменную _curr_subgraph_vertex. Из множества неиспользованных вершин вершина A имеет связь с вершинами C и D.

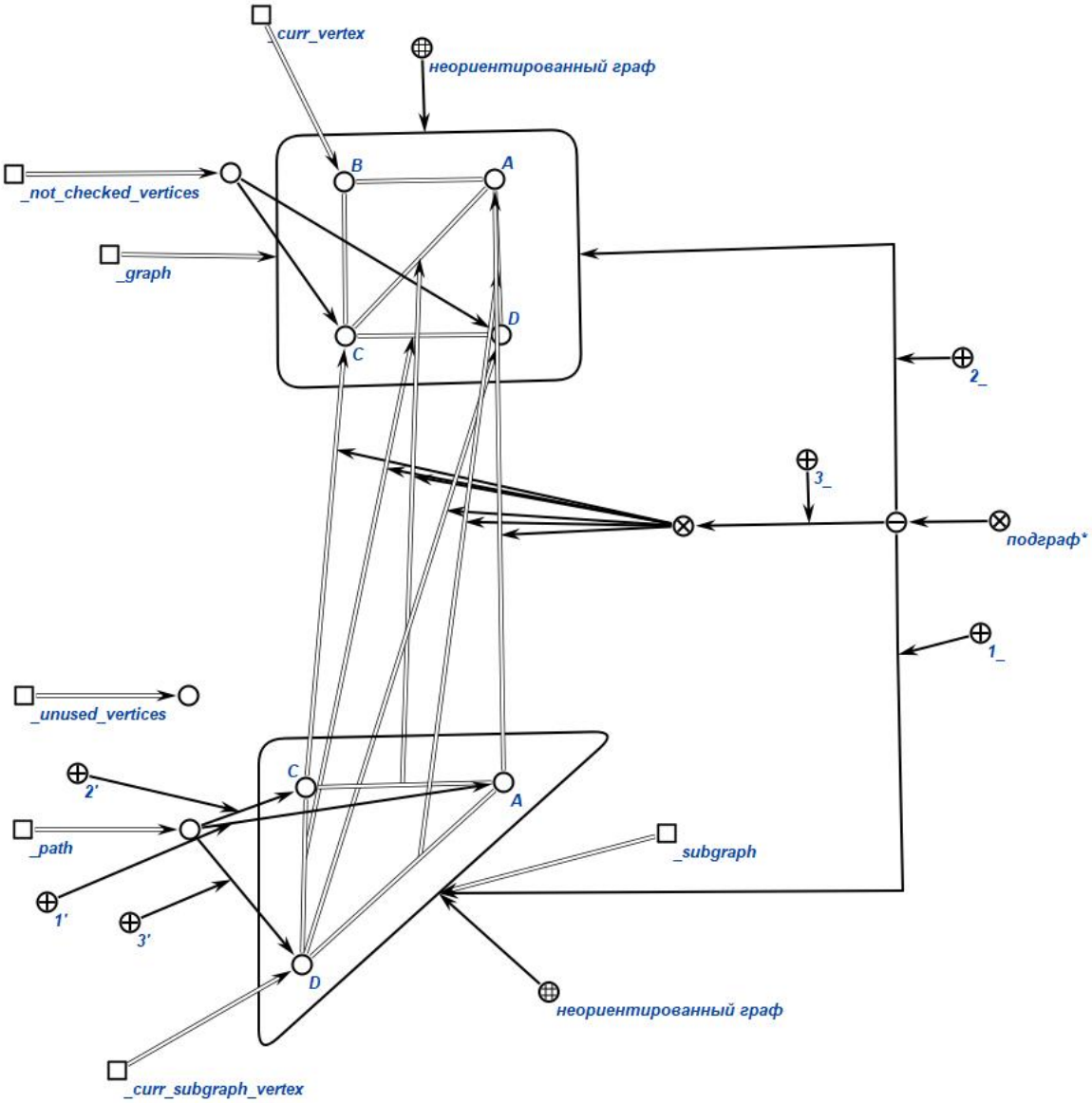
6. Проверка на связность подграфа. Добавление вершины C в ориентированное множество пути



Запишем C в _path с порядковым номером 2. Удалим ее из множества неиспользованных вершин и зададим в переменную _curr_subgraph_vertex.

Из множества неиспользованных вершин вершина C имеет связь с вершиной D.

7. Проверка на связность подграфа. Добавление вершины D в ориентированное множество пути

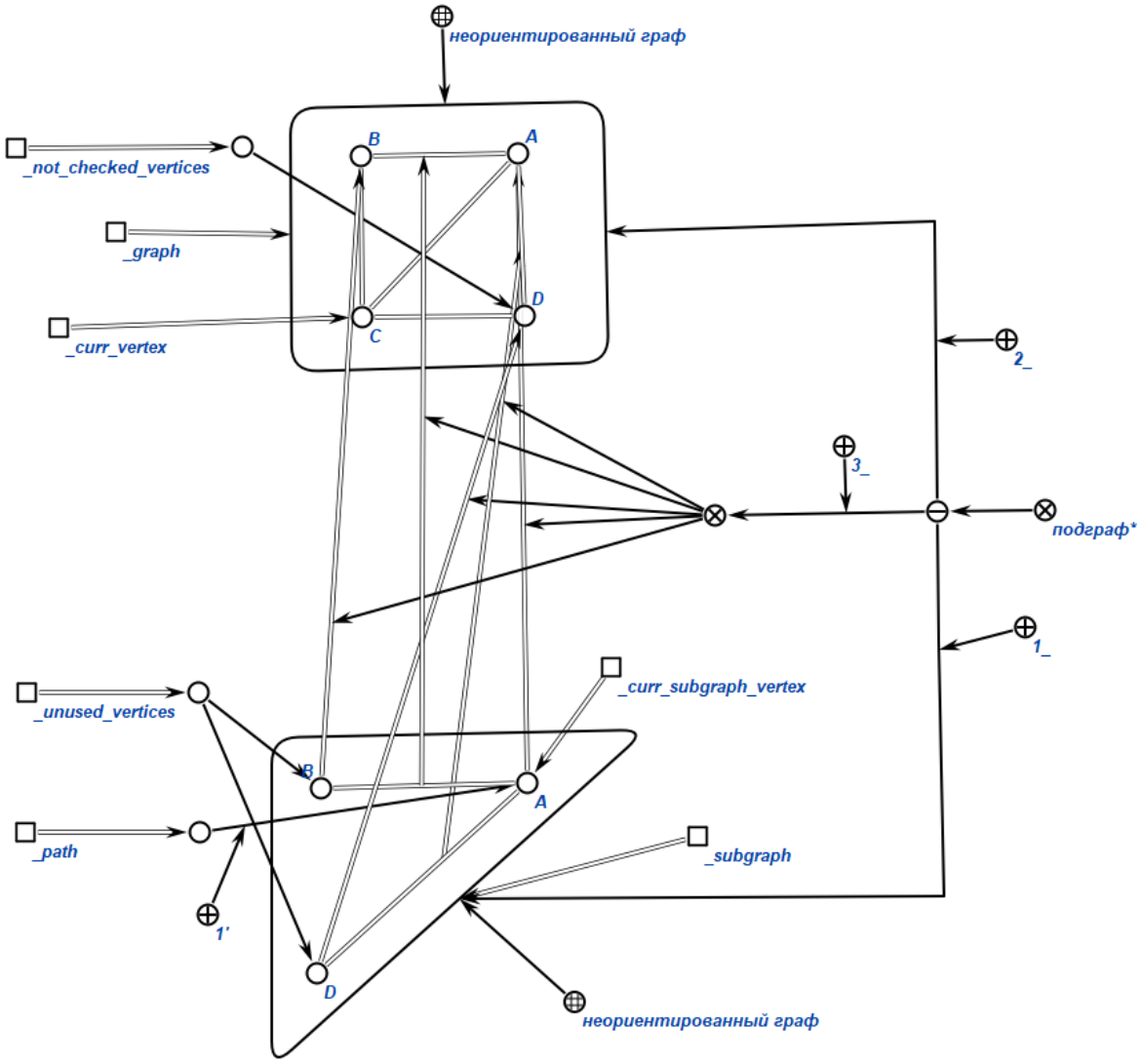


Запишем D в _path с порядковым номером 3. Удалим ее из множества неиспользованных вершин и зададим в переменную _curr_subgraph_vertex.

Множество неиспользованных вершин пусто. Можем сделать вывод, что подграф, полученный удалением вершины B и инцидентных ей ребер, получился связным.

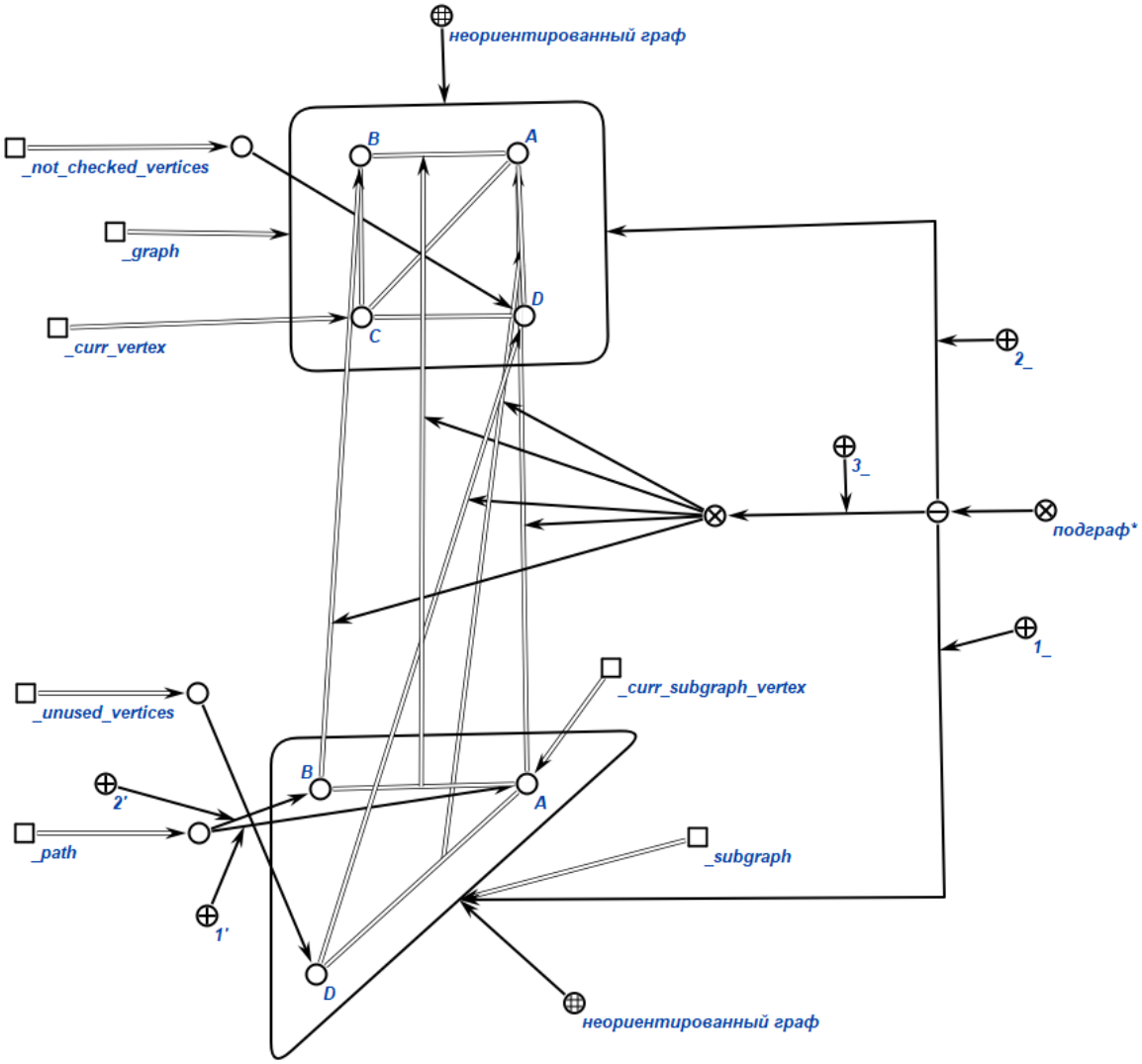
Продолжаем проверку для следующей вершины из множества _not_checked_vertices.

8. Проверка на связность подграфа. Добавление вершины A в ориентированное множество пути



Начинаем проверку для вершины C. Построим подграф исходного графа, удалив вершину C и все инцидентные ей ребра. Полученный подграф зададим переменной _subgraph. Начнем путь с вершины A, запишем A в _path с порядковым номером 1. Удалим ее из множества неиспользованных вершин и зададим в переменную _curr_subgraph_vertex. Из множества неиспользованных вершин вершина A имеет связь с вершинами B и D.

9. Проверка на связность подграфа. Добавление вершины B в ориентированное множество пути



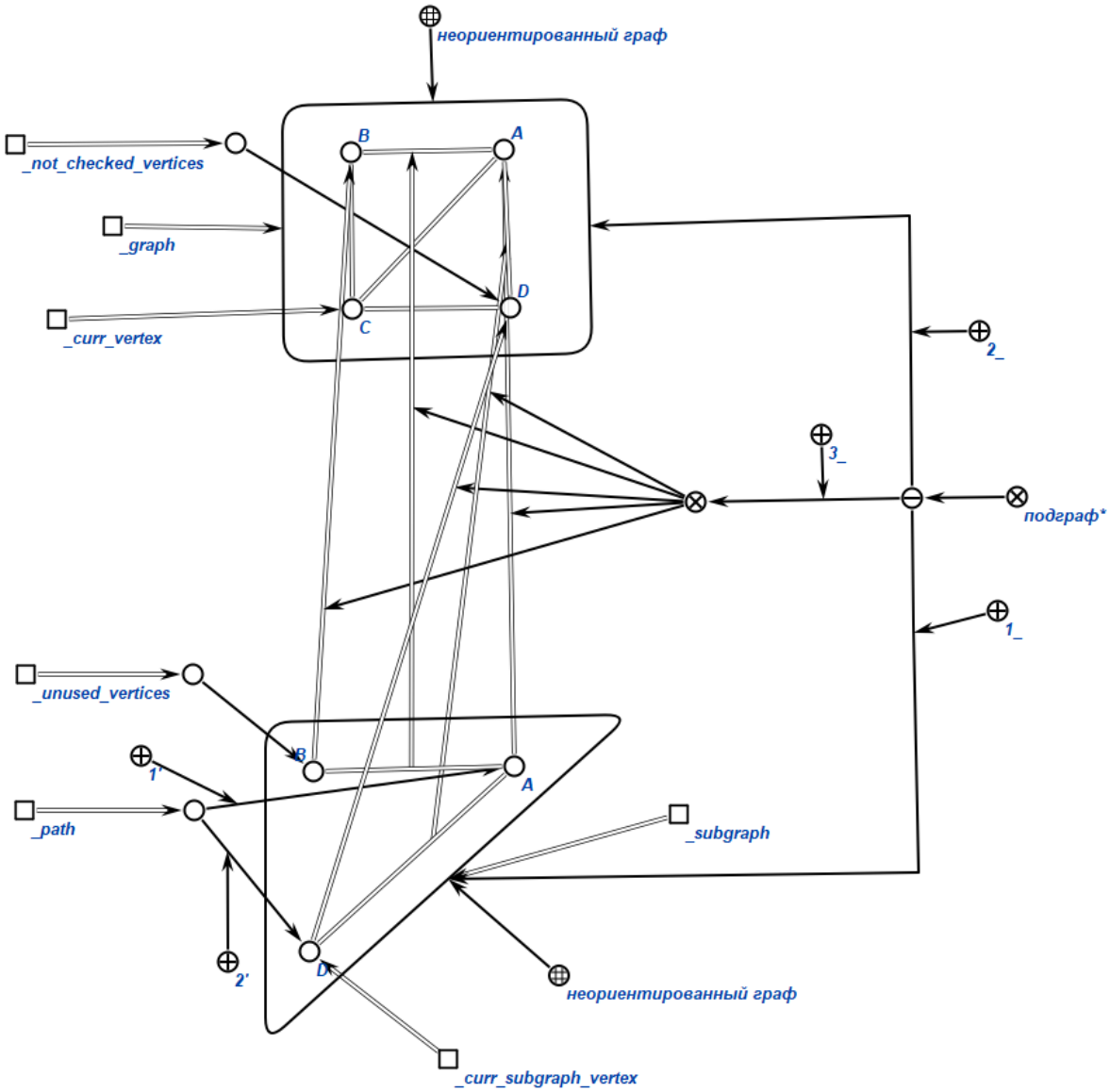
Запишем B в _path с порядковым номером 2. Удалим ее из множества неиспользованных вершин и зададим в переменную _curr_subgraph_vertex.

Из множества неиспользованных вершин вершина B не имеет связей ни с одной.

Возвращаемся к предыдущей вершине. Вершину B удаляем из ориентированного множества _path, добавляем во множество неиспользованных вершин. Вершину A зададим в переменную _curr_subgraph_vertex.

Вершина A имеет еще одну связь с вершиной D.

10. Проверка на связность подграфа. Добавление вершины D в ориентированное множество пути



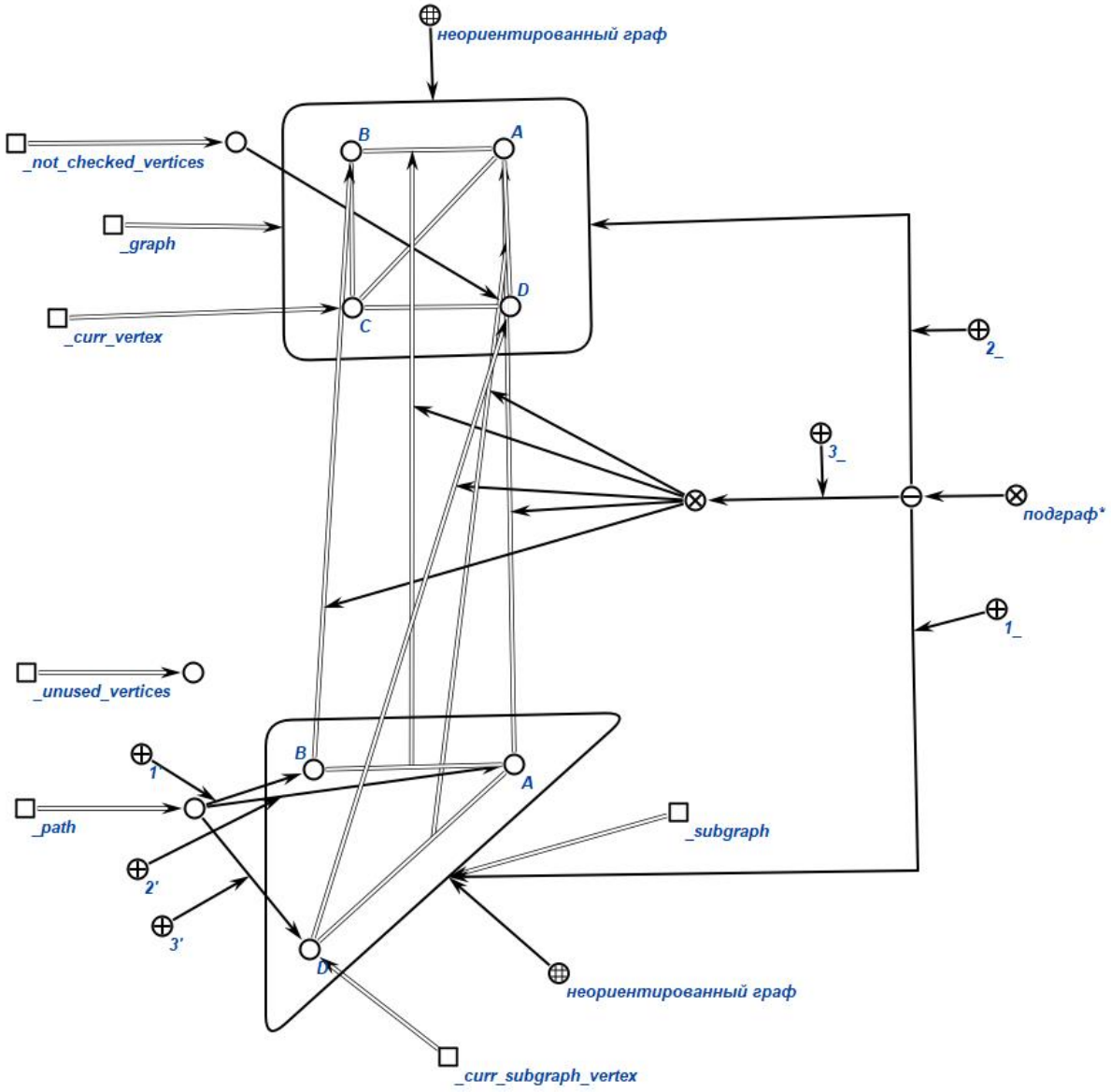
Запишем D в _path с порядковым номером 2. Удалим ее из множества неиспользованных вершин и зададим в переменную _curr_subgraph_vertex.

Из множества неиспользованных вершин вершина D не имеет связей ни с одной.

Возвращаемся к предыдущей вершине. Вершину D удаляем из ориентированного множества _path, добавляем во множество неиспользованных вершин. Вершину A зададим в переменную _curr_subgraph_vertex.

Продолжаем проверку, пробуем построить путь из вершины B.

11. Проверка на связность подграфа. Добавление вершины A в ориентированное множество пути



Запишем A в _path с порядковым номером 2. Удалим ее из множества неиспользованных вершин и зададим в переменную _curr_subgraph_vertex.

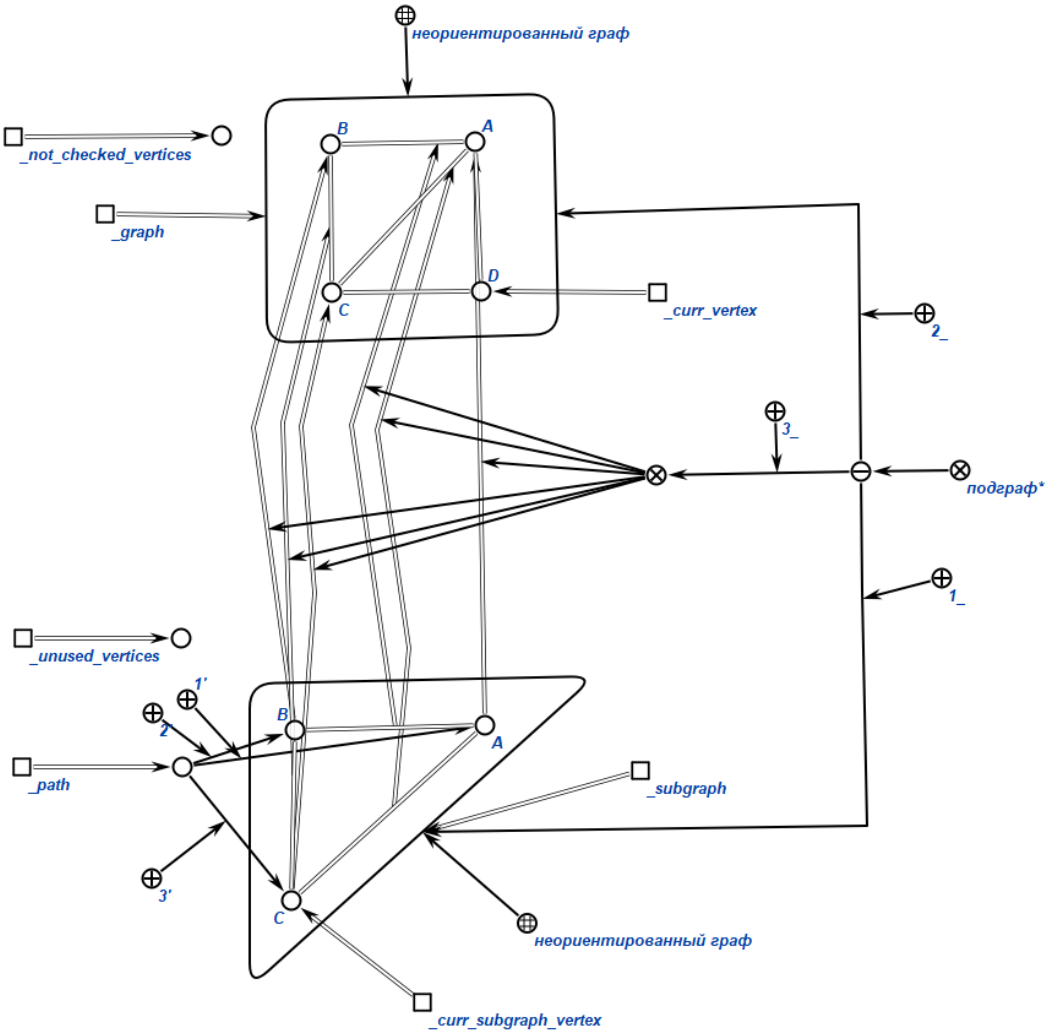
Из множества неиспользованных вершин вершина A имеет связь с вершиной D.

Запишем D в _path с порядковым номером 3. Удалим ее из множества неиспользованных вершин и зададим в переменную _curr_subgraph_vertex.

Множество неиспользованных вершин пусто. Можем сделать вывод, что подграф, полученный удалением вершины C и инцидентных ей ребер, получился связным.

Продолжаем проверку для следующей вершины из множества _not_checked_vertices.

12. Проверка на связность подграфа. Проверка для вершины D.



Начинаем проверку для вершины D. Построим подграф исходного графа, удалив вершину C и все инцидентные ей ребра. Полученный подграф зададим переменной _subgraph.

Начнем путь с вершины A, запишем A в _path с порядковым номером 1. Удалим ее из множества неиспользованных вершин и зададим в переменную _curr_subgraph_vertex.

Из множества неиспользованных вершин вершина A имеет связь с вершинами B и C.

Запишем B в _path с порядковым номером 2. Удалим ее из множества неиспользованных вершин и зададим в переменную _curr_subgraph_vertex.

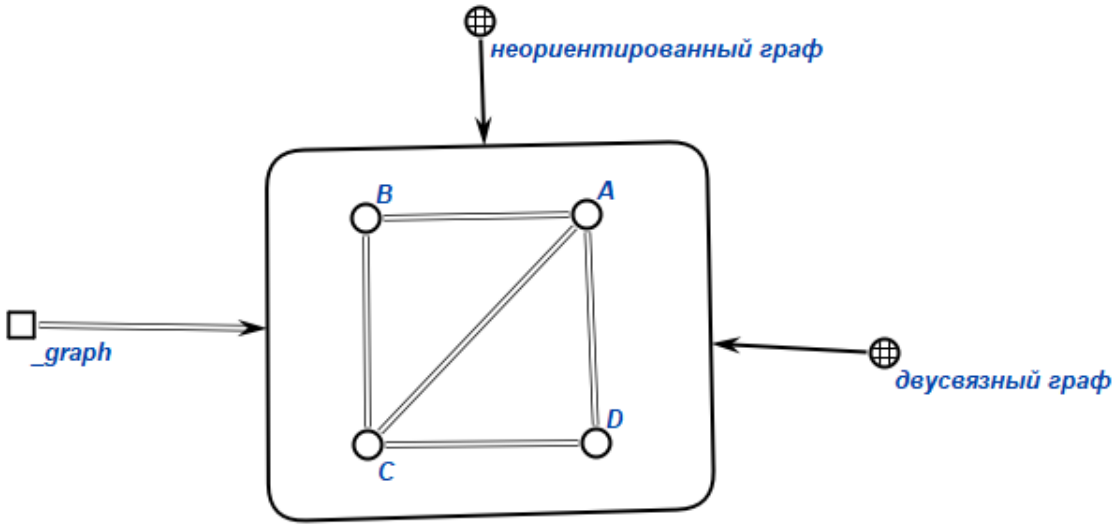
Из множества неиспользованных вершин вершина B имеет связь с вершиной C.

Запишем C в _path с порядковым номером 3. Удалим ее из множества неиспользованных вершин и зададим в переменную _curr_subgraph_vertex.

Множество неиспользованных вершин пусто. Можем сделать вывод, что подграф, полученный удалением вершины D и инцидентных ей ребер, получился связным.

Множество непроверенных вершин пусто, делаем вывод, что граф двусвязный. Удаляем связку подграф*, удаляем все лишние множества.

13. Проверка на связность подграфа. Проверка для вершины D.



3 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.