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 «Основы искусственного интеллекта»

на тему   
«Задача на поиск минимального множества вершин в графе, удаление которых позволяет сделать его деревом»

Выполнил ( Якимцов И.Д)

студент группы

521702

Проверил ( Шункевич Д.В )

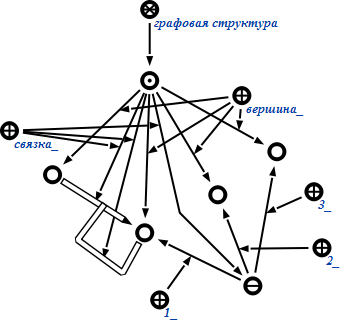
Минск 2016

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

**Задача:** Поиск минимального множества вершин в графе, удаление которых позволяет сделать его деревом

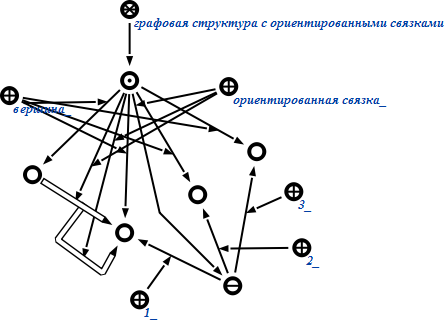
# Список понятий

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).



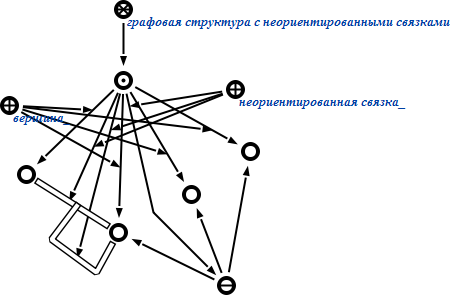
Рисунок

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



Рисунок

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



Рисунок

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

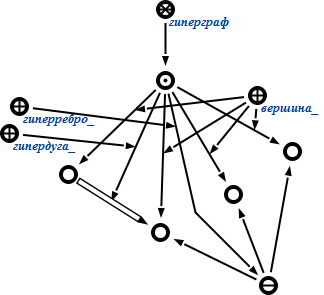


Рисунок - Гиперграф

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

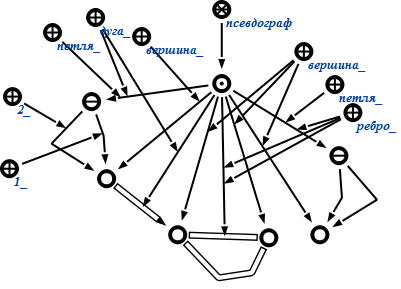


Рисунок - Псевдограф

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

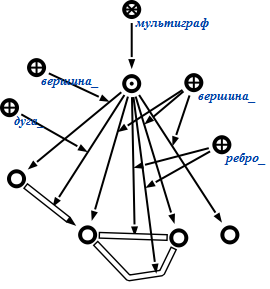


Рисунок - Мультиграф

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

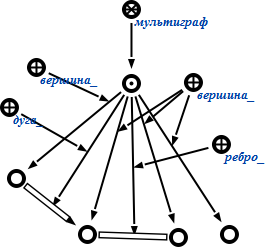


Рисунок - Граф

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

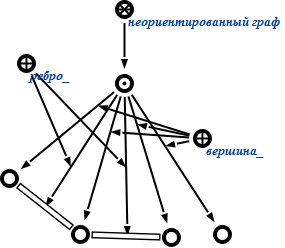


Рисунок - Неориентированный граф

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

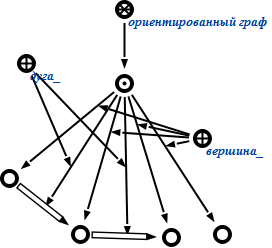


Рисунок - Ориентированный граф

1. 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, Aв гиперграфе.

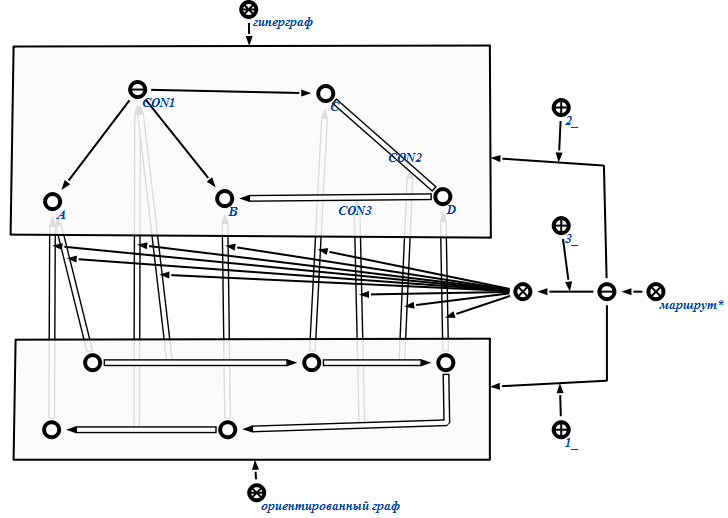


Рисунок - Маршрут

1. 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, Aв гиперграфе.

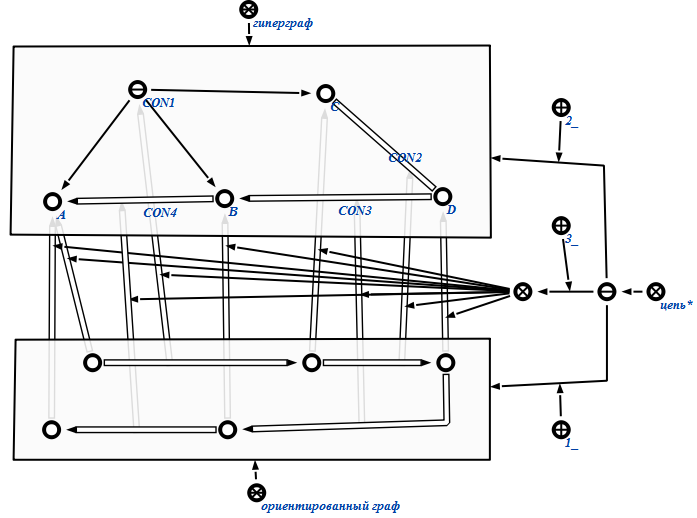


Рисунок - Цепь

12. Цикл (относительное понятие, бинарное ориентированное отношение) - это [цепь](http://pco.iis.nsk.su/grapp2/html/ReqtypetNamecep8.htm), концы которой совпадают. В примере ниже показан цикл A, CON1, C, CON2, D, CON3, B, CON4, Aв гиперграфе.

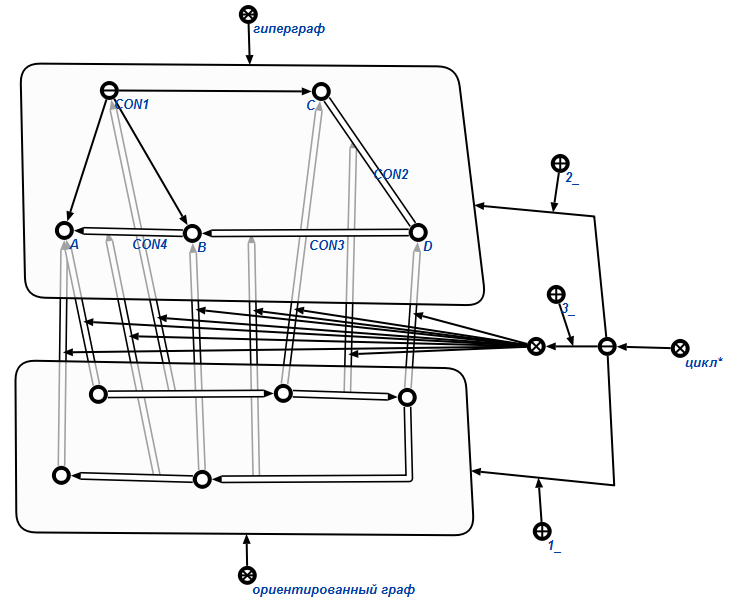


Рисунок - Цикл

13. Дерево (абсолютное понятие) - это связный граф без циклов.

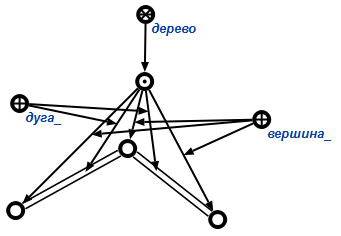


Рисунок - Дерево

# Тестовые примеры

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

## Тест 1

**Вход:**

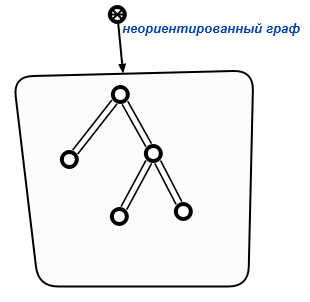


Рисунок -тест 1

**Выход:**

Множество deleted\_vertex осталось пустым так как граф является деревом:

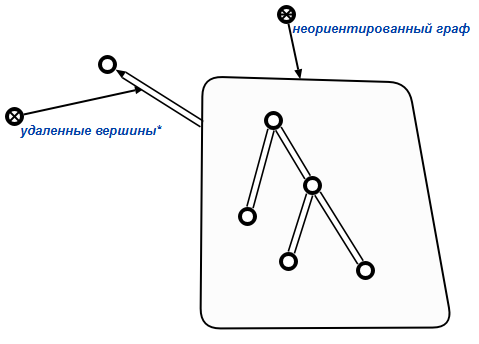


Рисунок -Результат 1 теста

## Тест 2

**Вход:**

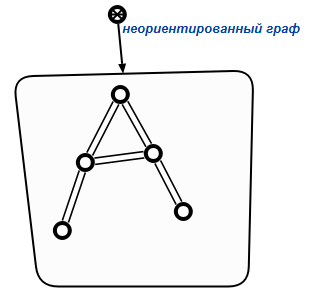


Рисунок -тест 2

**Выход:**

Единственная точка будет занесена в множество удаленных вершин:

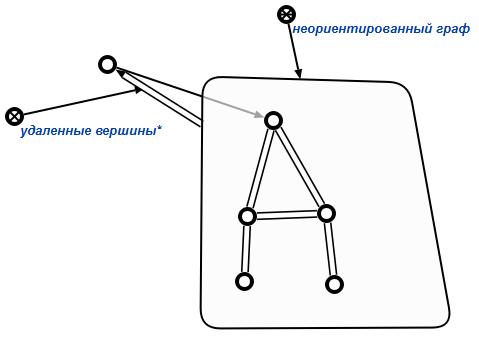


Рисунок -результат теста 2

## Тест 3

**Вход:**

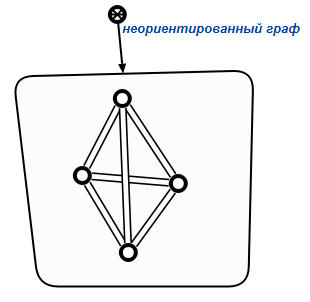


Рисунок -тест 3

**Выход:**

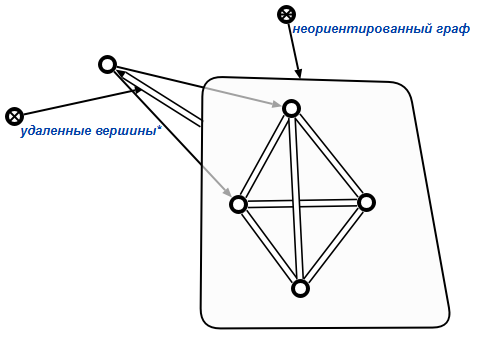


Рисунок -результат теста 3

## Тест 4

**Вход:**

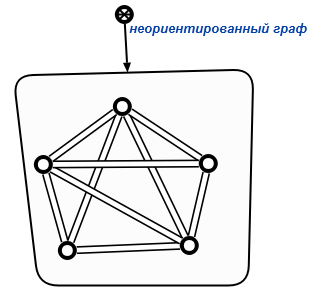


Рисунок -тест 4

**Выход:**

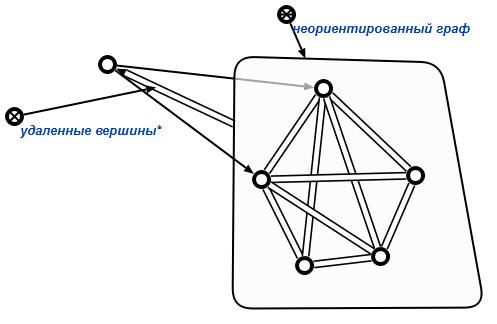


Рисунок -результат теста 4

## Тест 5

**Вход:**

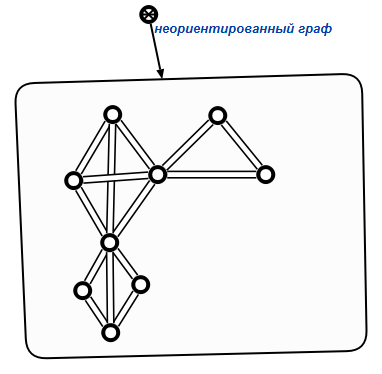


Рисунок -тест 5

**Выход:**

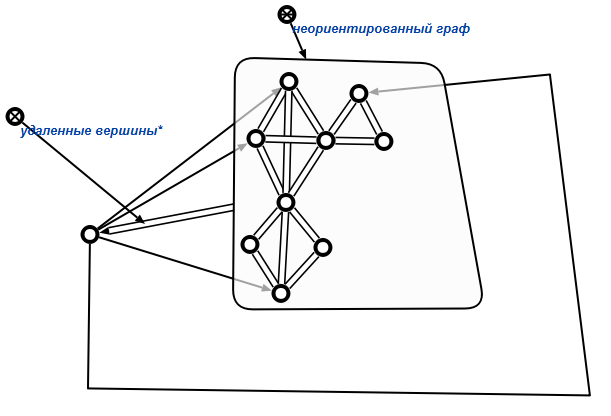


Рисунок -результат теста 5

.

**Алгоритм**

**Задаём исходный граф**

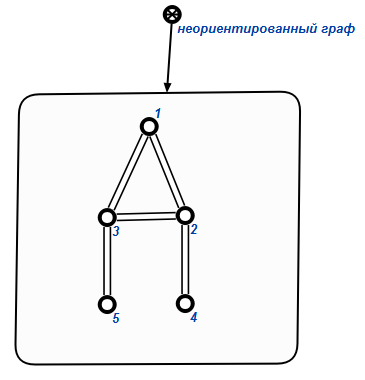
****

Рисунок -ввод данных

Алгоритм

1. Создаем множество in\_ process просматриваемых вершин.
2. Создаём множество parents, в котором будут храниться родители каждой вершины графа.(для стартовой вершины родитель равен -1)
3. Создаём множество cycles мощность которого равна количеству вершин, в котором будет хранится приоритет на удаление вершин графа(изначально у всех вершин приоритет равен 0).
4. Создаём пустое множество deleted\_vertex, в котором будет храниться список удаленных вершин.
5. Создаем множество watched для просмотренных вершин.

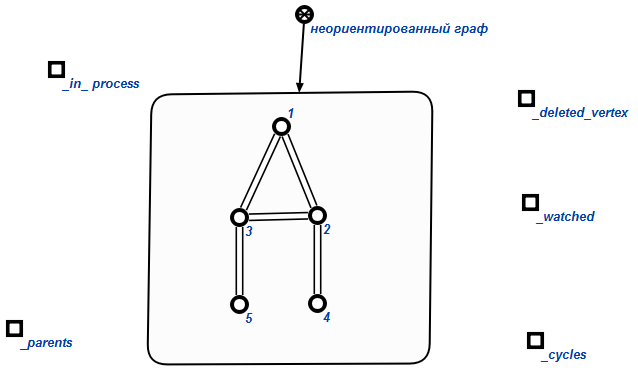


Рисунок -Начальное состояние

1. Обозначаем вершину, в которой мы находимся, переменной v (выбираем вершину).

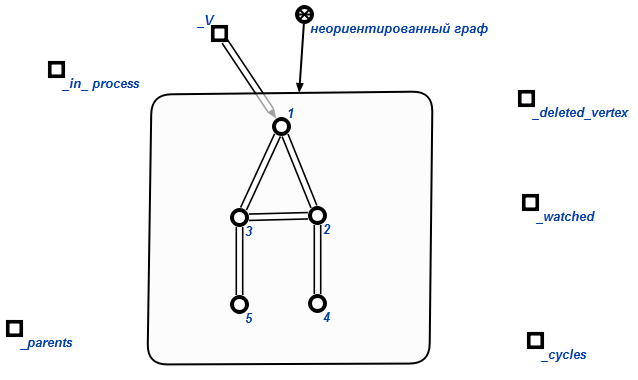


Рисунок -Выбираем вершину

7.Добавляем вершину V в множество просматриваемы вершин.

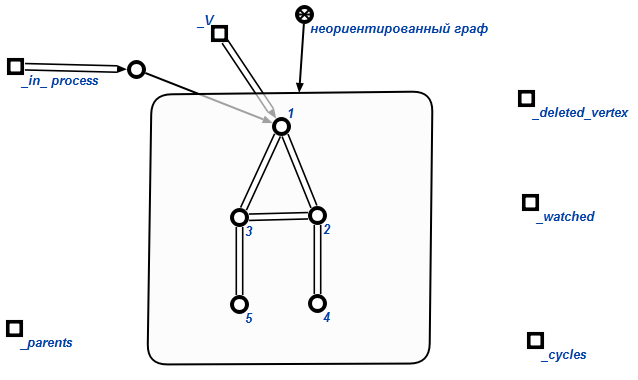
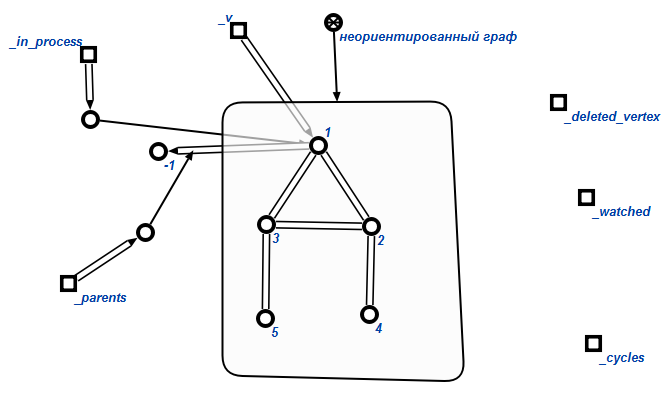


Рисунок -Добавляем вершину в множество просматриваемых

8.Добавляем родителя вершины v(так как данная вершина первая то её родитель -1).



9.next\_v - вершина, смежная вершине v, которая в данный момент не просматривается и не является родителем вершины v.

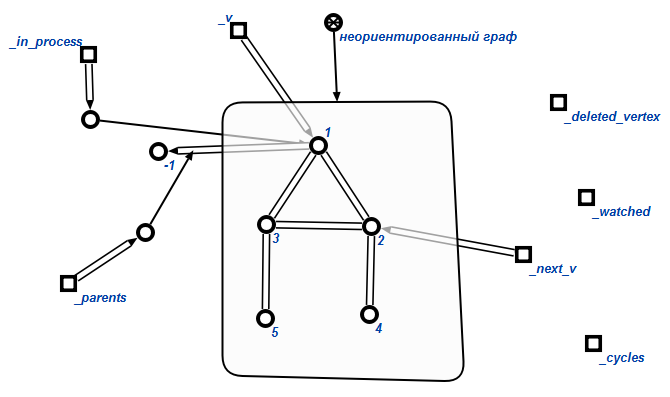


Рисунок -Находим смежную вершину

10.Переходим в вершину next\_v.

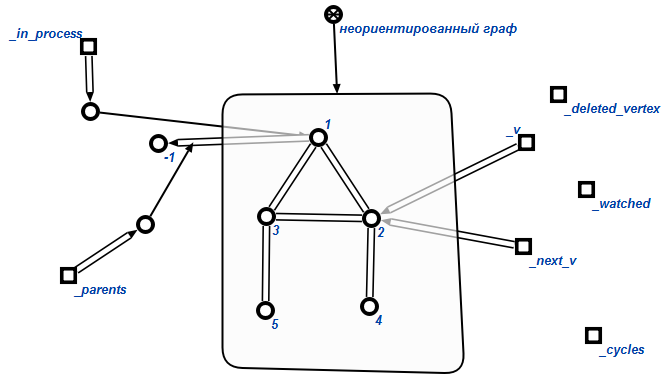


Рисунок -переходим в смежную вершину

11. Добавляем вершину v в множество просматриваемы вершин.

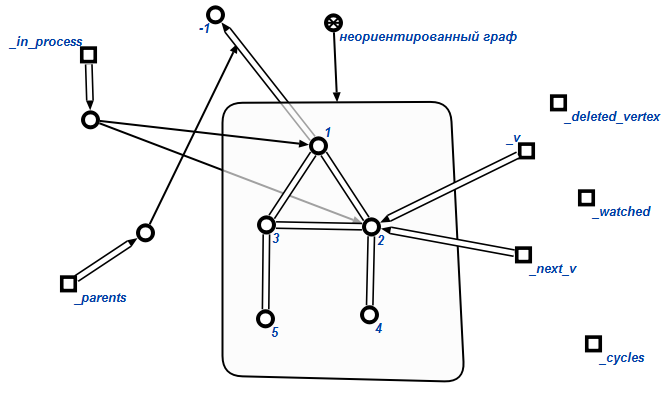


Рисунок -Добавляем вершину в множество просматриваемых

12. Добавляем родителя вершины v.

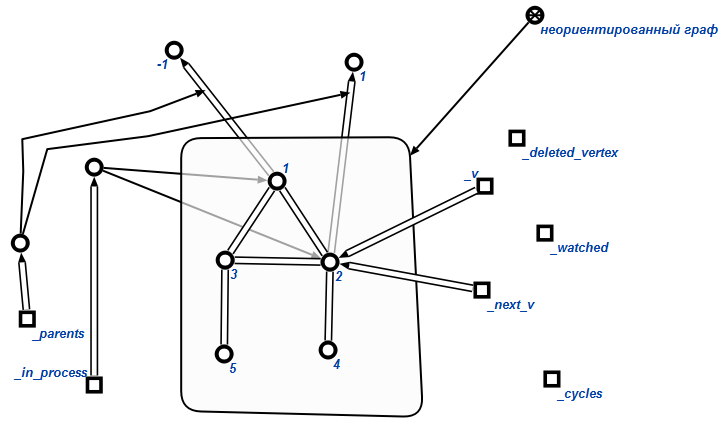


Рисунок -Добавляем родителя вершины

13. next\_v - вершина, смежная вершине v,которая в данный момент не просматривается и не является родителем вершины v.

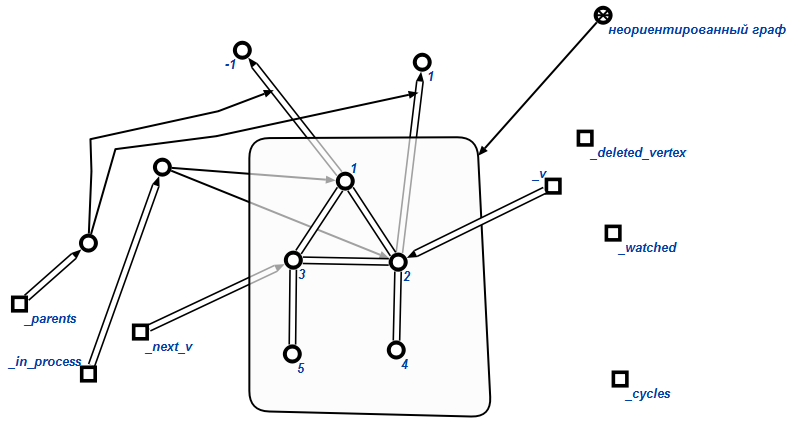


Рисунок -Находим смежную вершину

14.Переходим в вершину next\_v.

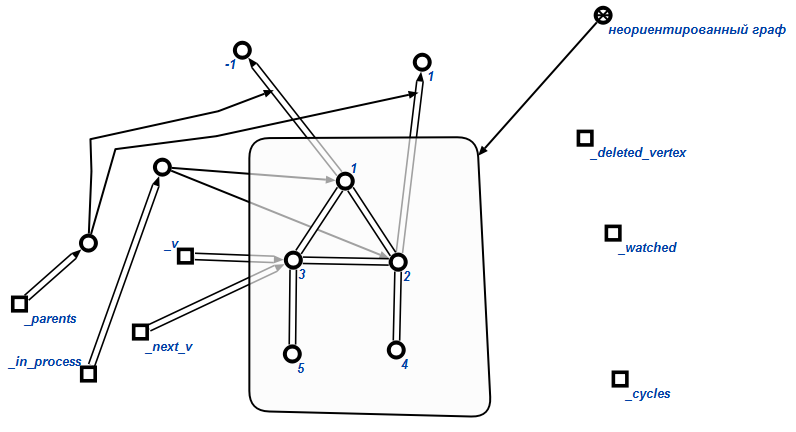


Рисунок -Переходим в смежную вершину

15. Добавляем родителя вершины v и заносим вершину v в множество просматриваемых вершин.

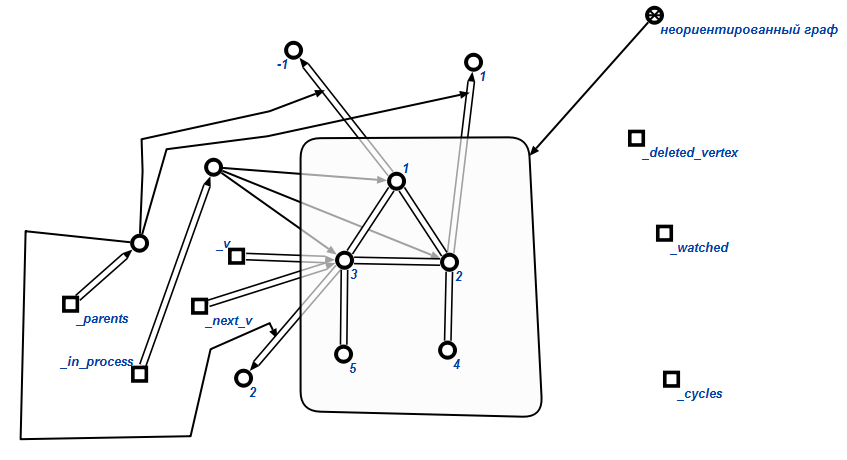


Рисунок -заносим вершину в множество просматриваемых

16. next\_v - вершина, смежная вершине v,которая в данный момент не просматривается .

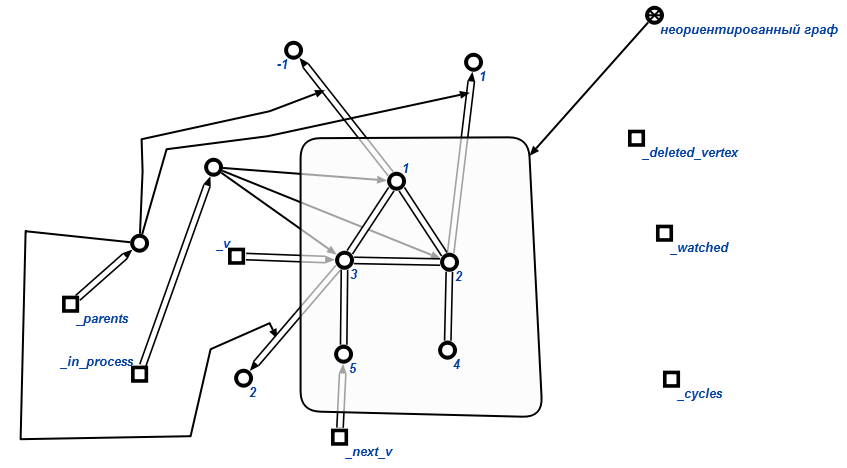


Рисунок -Находим смежную вершину

17.Stop\_v вершина смежная v которая в данный момент просматривается и не является родителем вершины v.

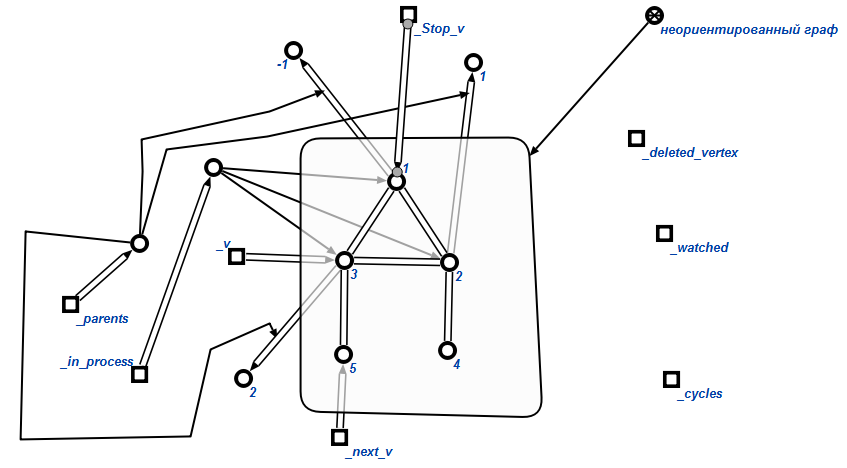


Рисунок -отмечаем просматриваемую смежную вершину

18.Увеличиваем приоритет на удаление вершины v.

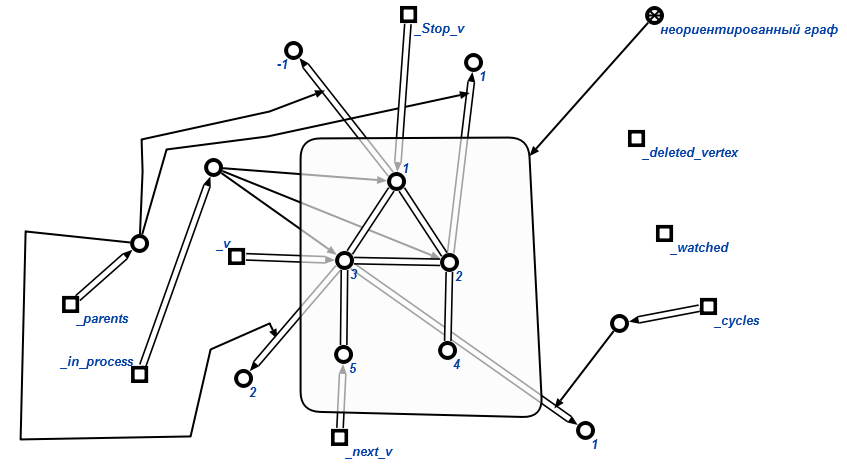


Рисунок -меняем приоритет на удаление вершины

19.Переходим в родителя вершины v.

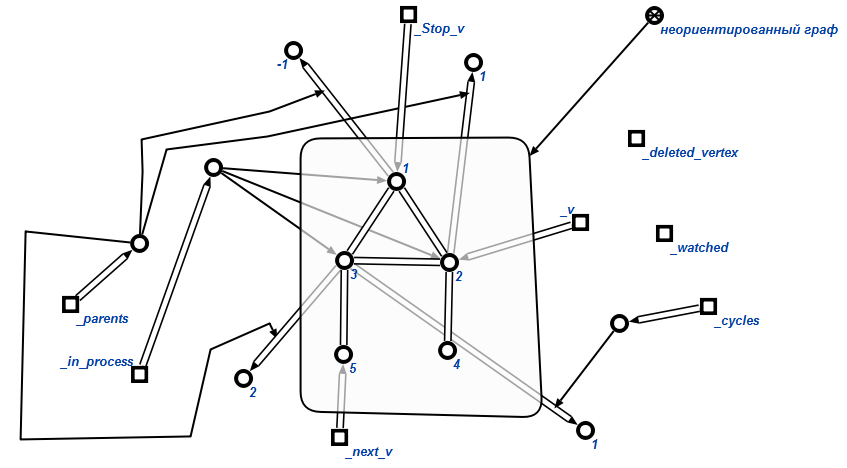


Рисунок -переходим в родителя вершины

20.Повторяем пункты 18-19 пока не дойдем до вершины Stop\_v.

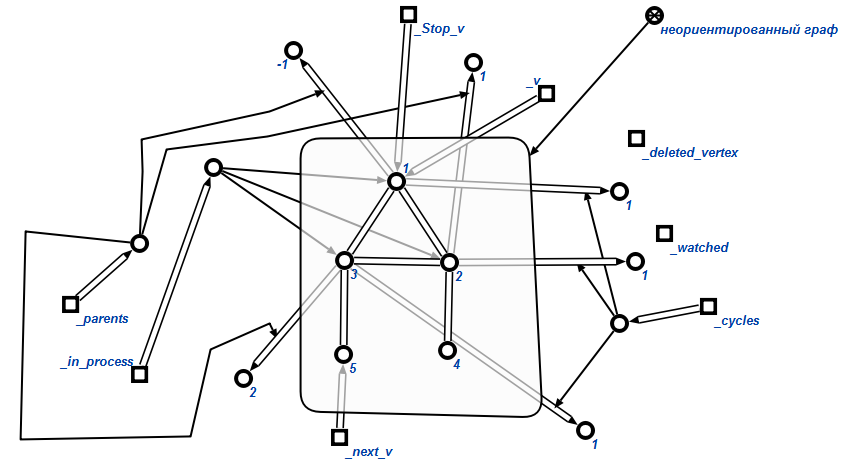


Рисунок -Меняем приоритет на удаления вершин

21.Переходим в вершину next\_v.

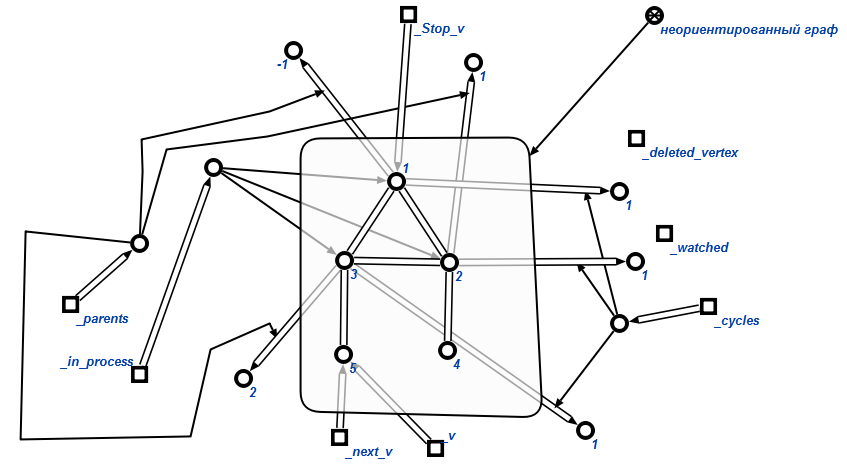


Рисунок -переходим в вершину на которой остановились

22.Заносим вершину в множество просматриваемых.

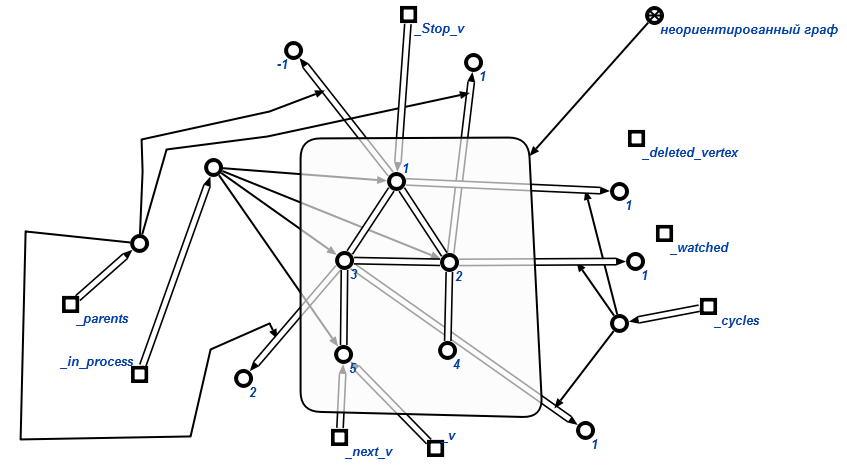


Рисунок -заносим вершину в множество просматриваемых

23.Так как для вершины v нет не просматриваемых смежных вершин которые не являлись бы родителями вершины v, удаляем вершину v множества просматриваемых и заносим в множество просмотренных.

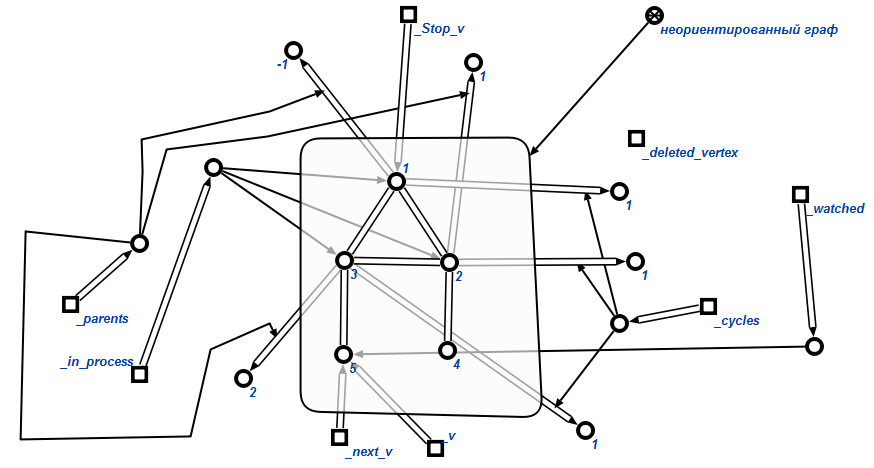


Рисунок -Заносим вершину в множество просмотренных

24.Заносим родителя вершины v.

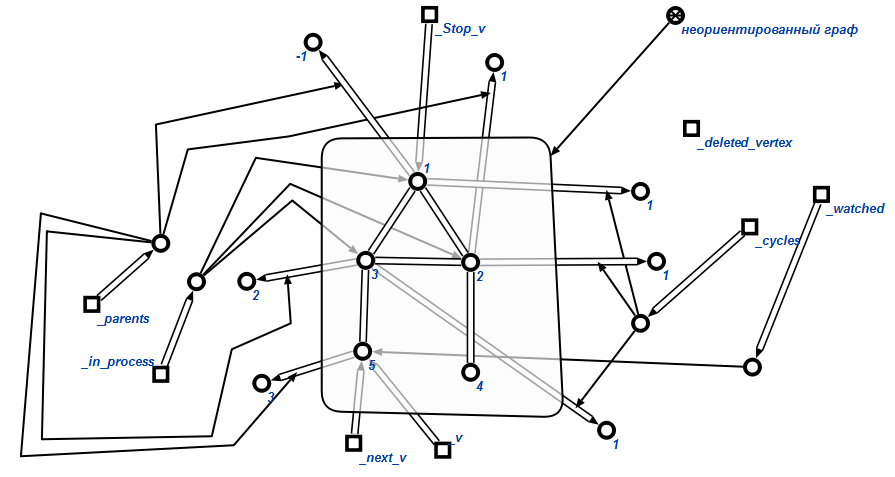


Рисунок -Добавляем родителя вершины

25.Переходим в родителя вершины v.

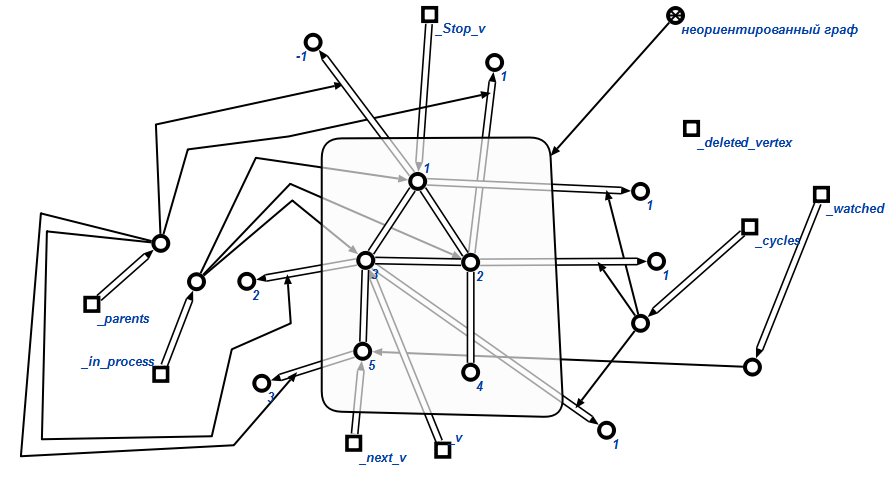


Рисунок -Переходим в родителя вершины

26.Повторяем пункты аналогичные предыдущим пока все вершины не будут занесены в множество просмотренных вершин.

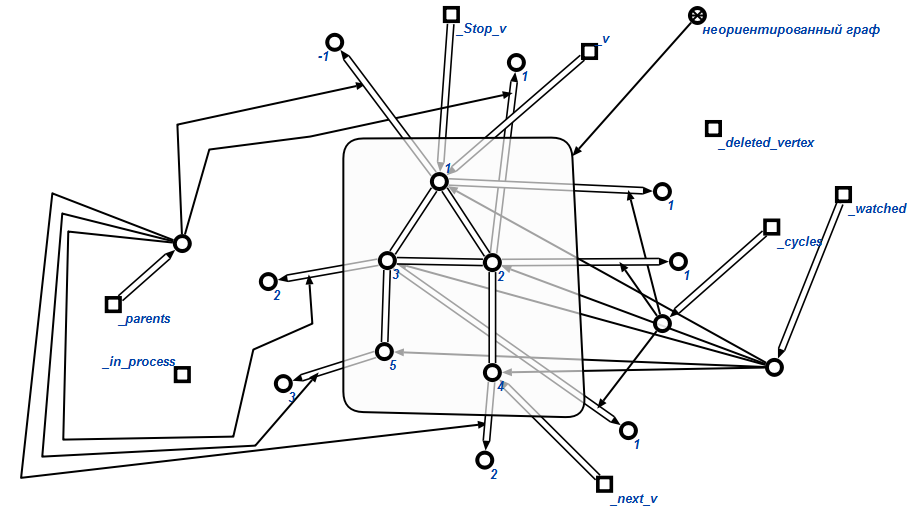


Рисунок -Просматриваем все вершины

28.Удаляем из множество просмотренных вершин все вершины кроме вершины с наибольшим приоритетом на удаление(в случае равенства приоритетов берем любую из вершин с равными приоритетами).

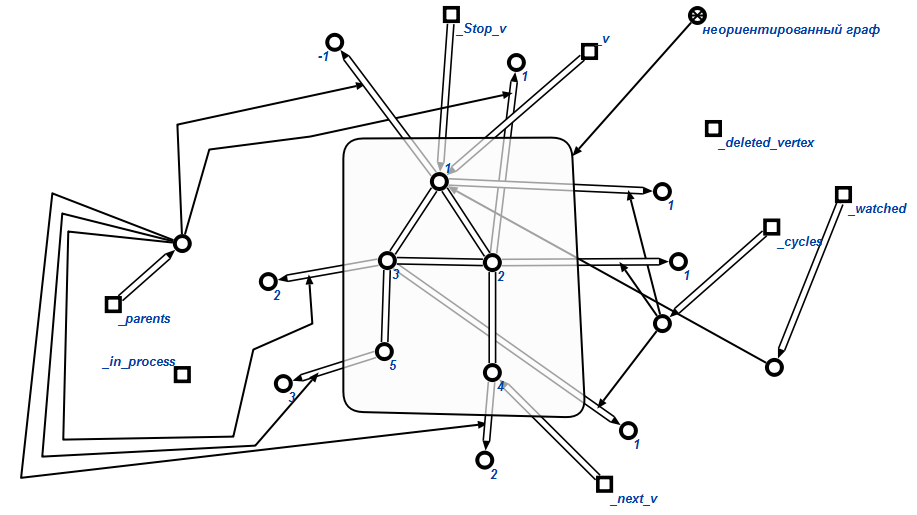


Рисунок -Удаляем часть вершин из множества просматриваемых

29.Удаляем все вершины из множества parents.

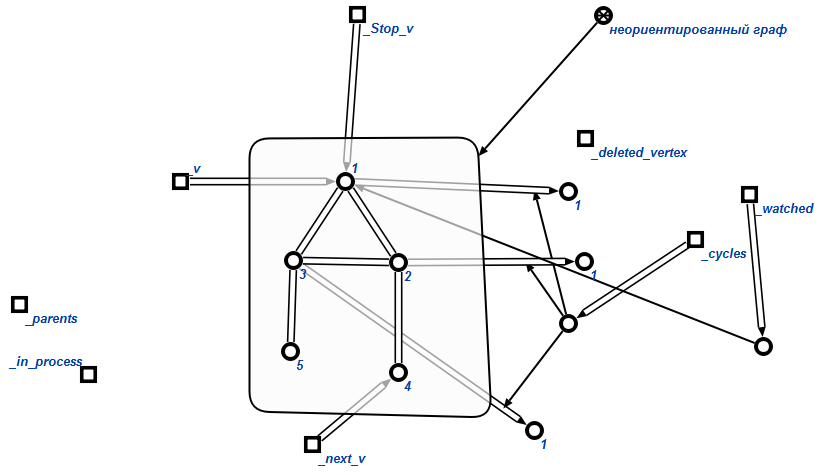


Рисунок -очищаем множество

30.Отмечае вершину v переменной del.

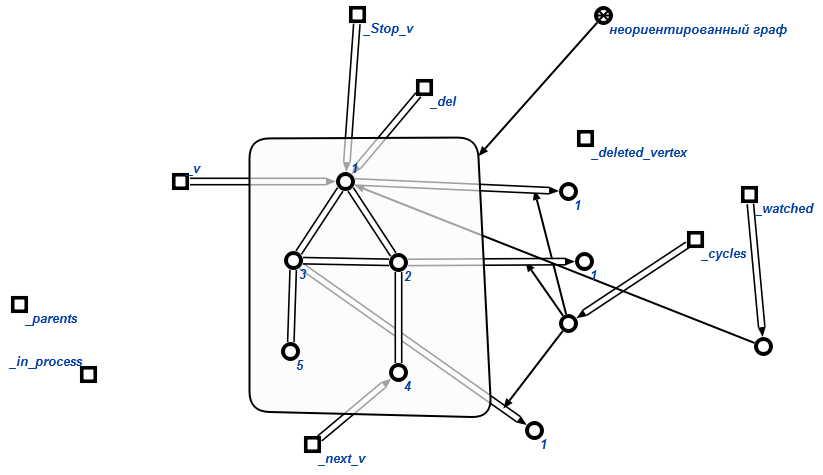


Рисунок -отмечаем вершину

30. Проверяем граф на связанность. Повторяем пункты аналогичные пунктам 6-14 начиная от любой не просмотренной вершины.

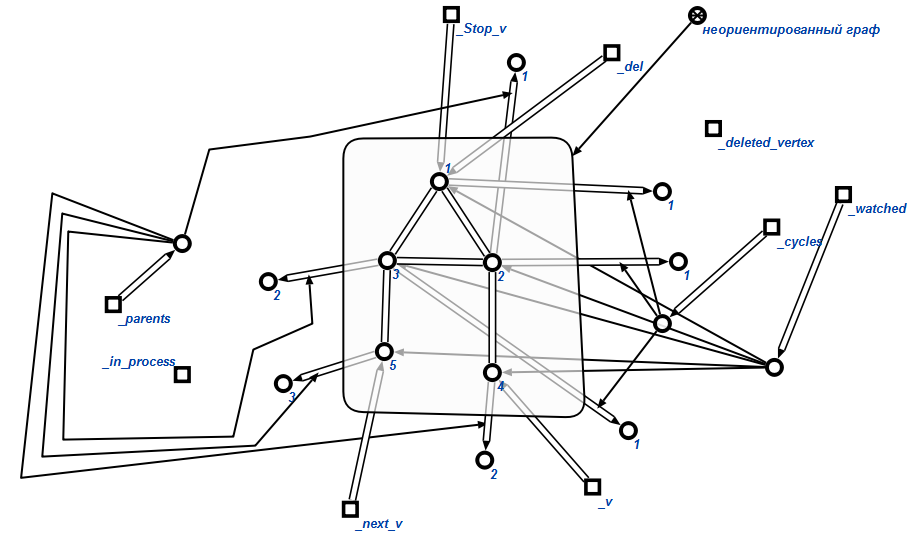


Рисунок -проверяем граф на связанность

31.Если все вершины графа занесены в множество просмотренных вершин то считаем что граф связан. Заносим вершину v в множество deleted\_vertex.

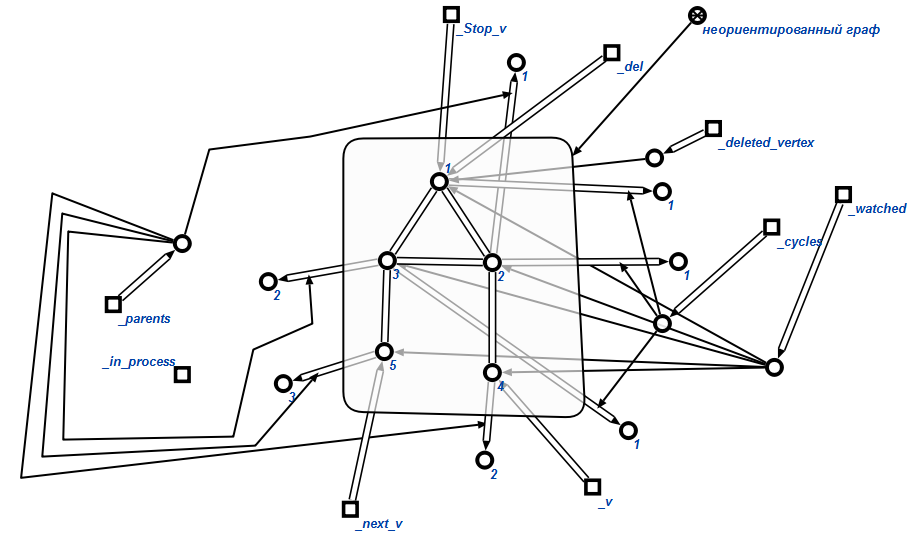


Рисунок -заносим вершину в множество

32.Удаляем все элементы множества parents. Удаляем все элементы множества cycles. Удаляем все вершины не принадлежащие множеству deleted\_vertex из множества просмотренных.

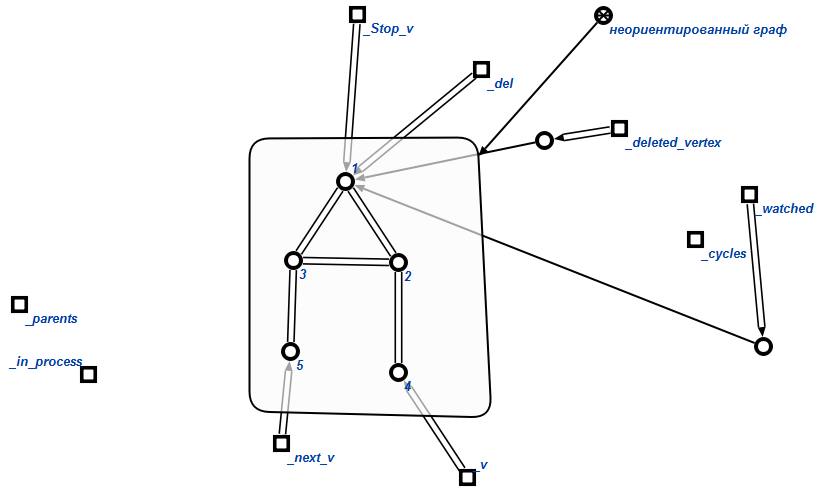


Рисунок -очищаем множество

33.Повторяем пункты аналогичные предыдущим. Если приоритет на удаления всех вершин равен 0 то считаем что граф превратится в дерево после удаления вершин из множества deleted\_vertex.

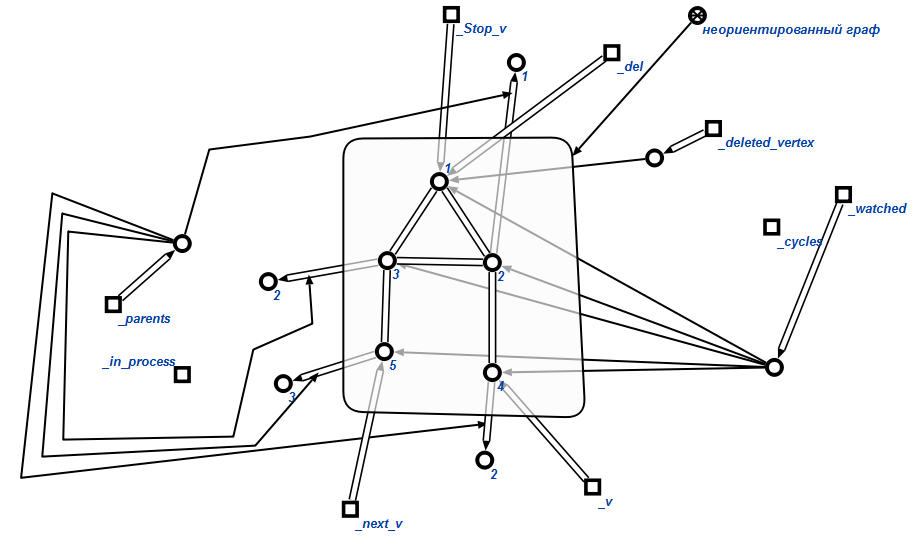


Рисунок -Повторяем предыдущие пункты

34. Удаляем все переменные и все множества кроме множества вершин удаление которых сделает граф деревом.

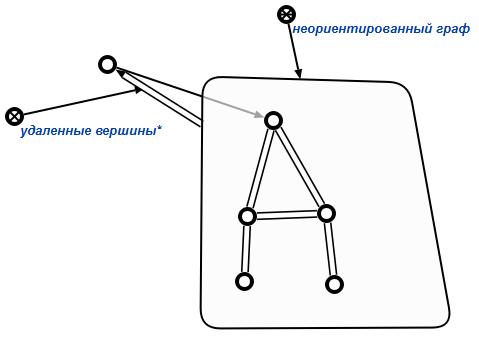


Рисунок -ответ

# 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.