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 )

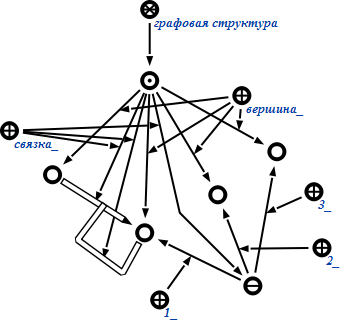
Минск 2016

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

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

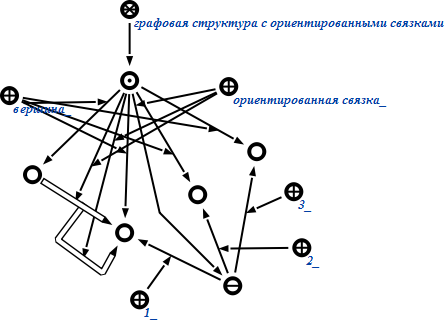
# Список понятий

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).



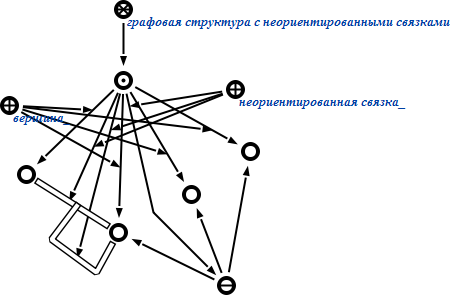
Рисунок

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



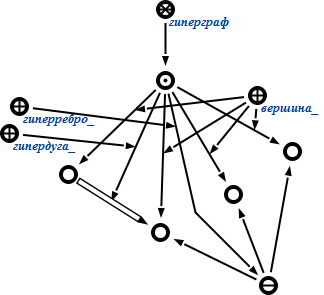
Рисунок

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



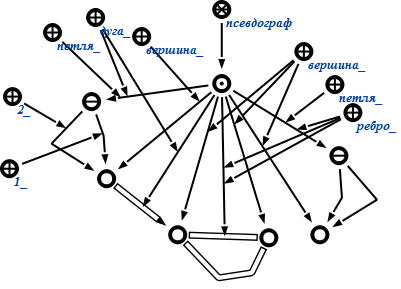
Рисунок

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.



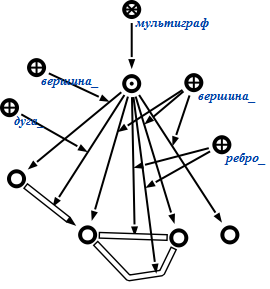
Рисунок

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.



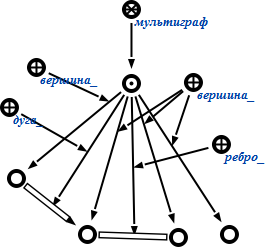
Рисунок

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



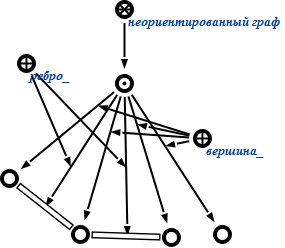
Рисунок

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



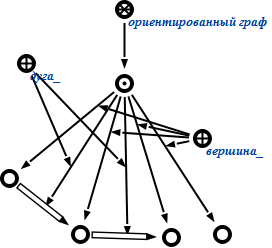
Рисунок

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



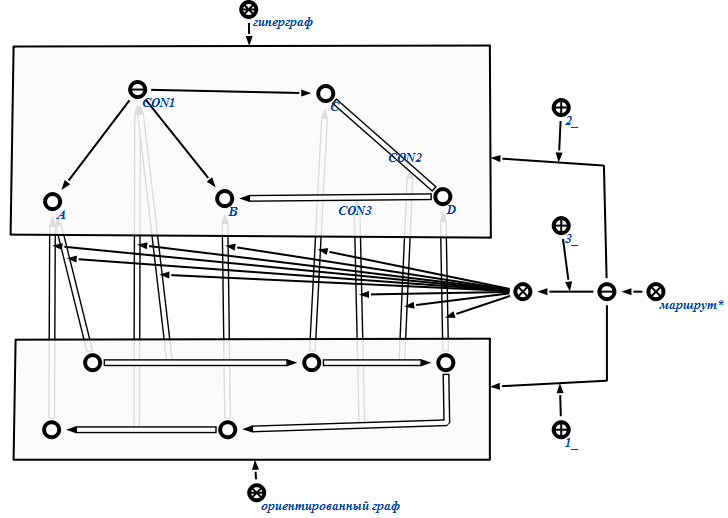
Рисунок

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



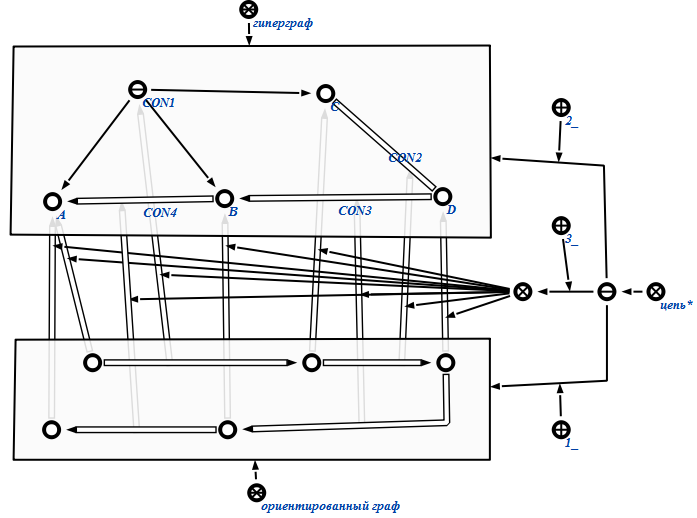
Рисунок

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.



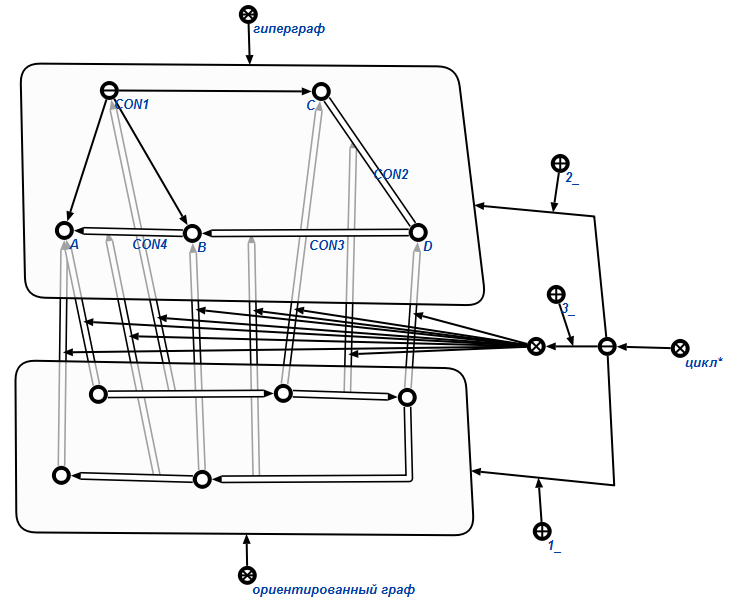
Рисунок

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.



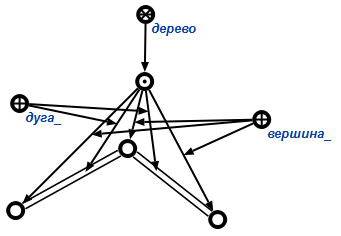
Рисунок

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

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



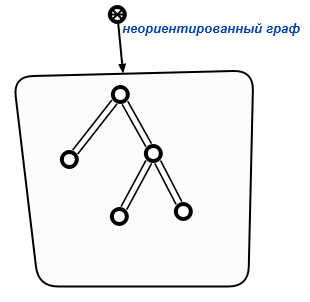
Рисунок

# Тестовые примеры

Во всех тестах графы будет приведены в сокращенной форме со скрытыми ролями элементов графа. deleted\_vertex минимальное множество вершин удаление которых позволит превратить граф в дерево.

## Тест 1

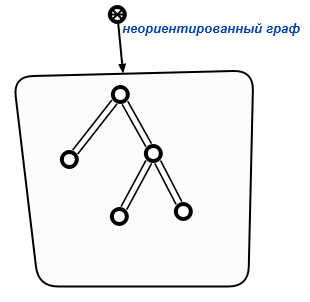
**Вход:**



Рисунок

**Выход:**

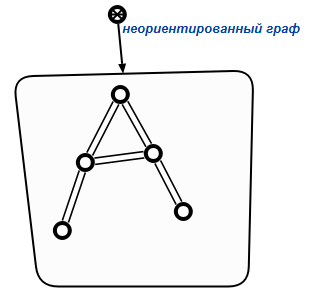
Граф не изменится так как уже является деревом:



Рисунок

## Тест 2

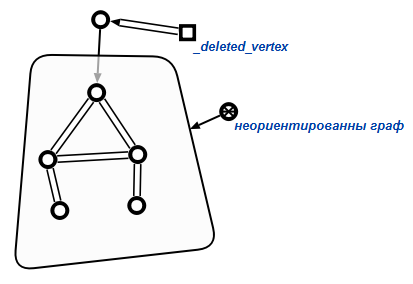
**Вход:**



Рисунок

**Выход:**

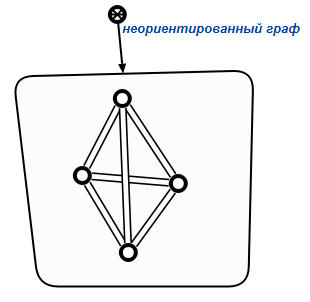
Единственная точка будет занесена в множество удаленных вершин:



Рисунок

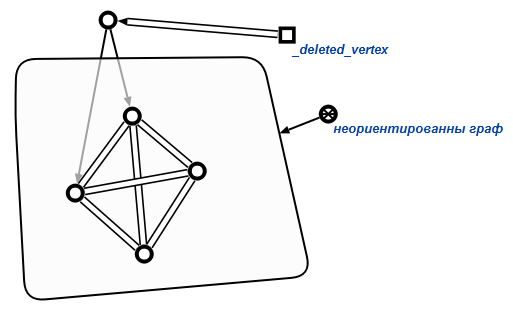
## Тест 3

**Вход:**



Рисунок

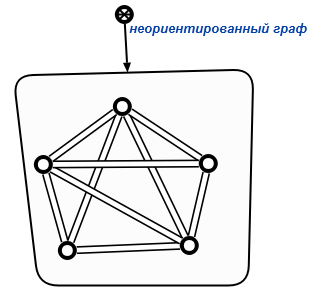
**Выход:**

****

Рисунок

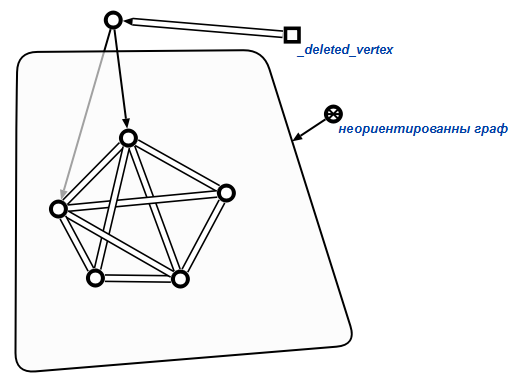
## Тест 4

**Вход:**



Рисунок

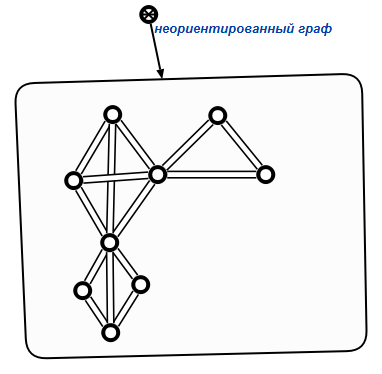
**Выход:**



Рисунок

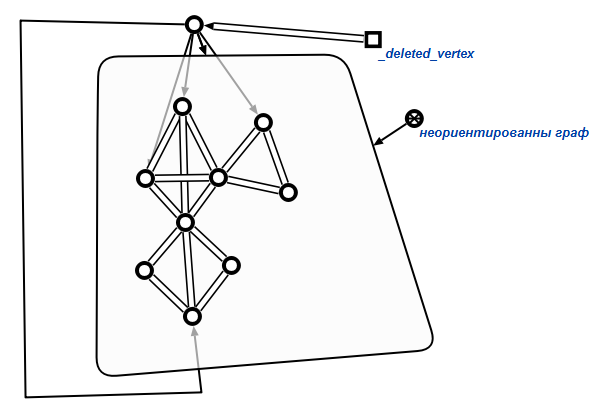
## Тест 5

**Вход:**



Рисунок

**Выход:**

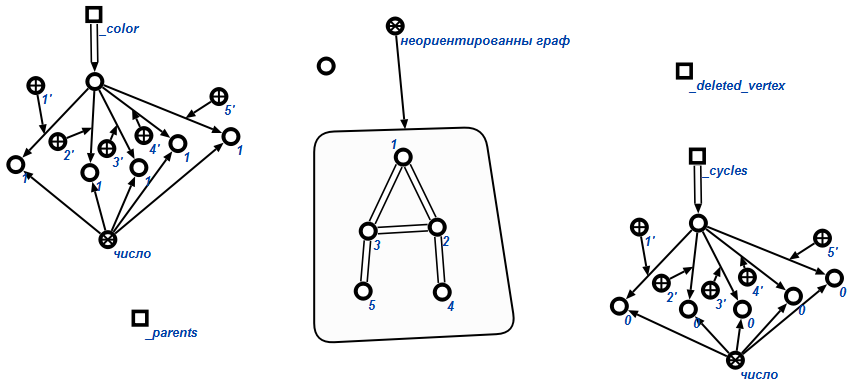


Рисунок

.

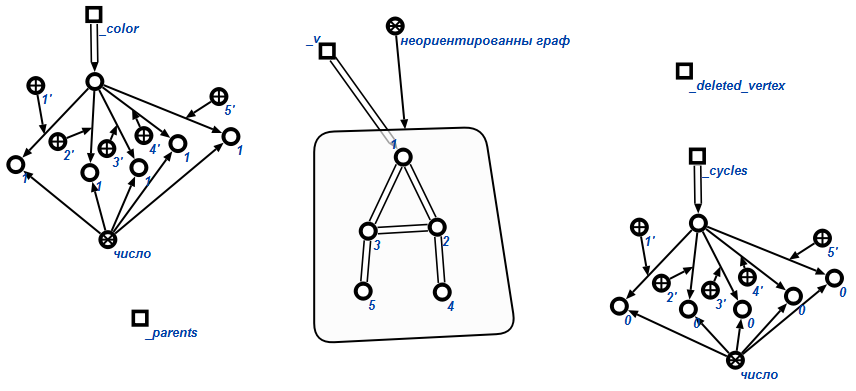
Алгоритм

1. Создаём множество color мощность которого равна количеству вершин, в котором будут храниться "цвета" вершин графа(1-вершина не посещена,2-вершина просматривается,3-вершина просмотрена).
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.

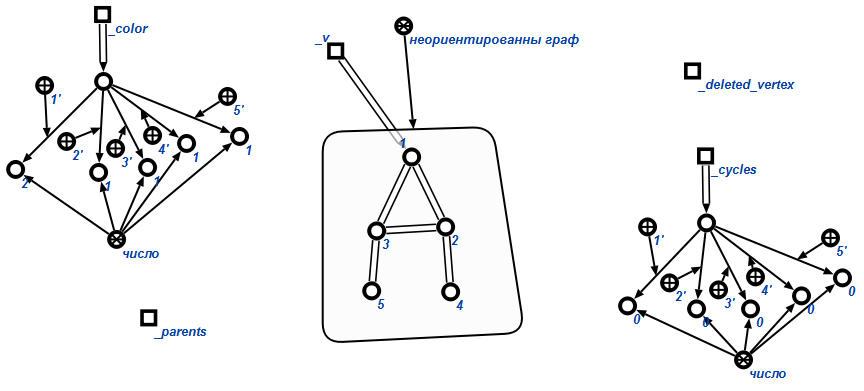


Рисунок

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

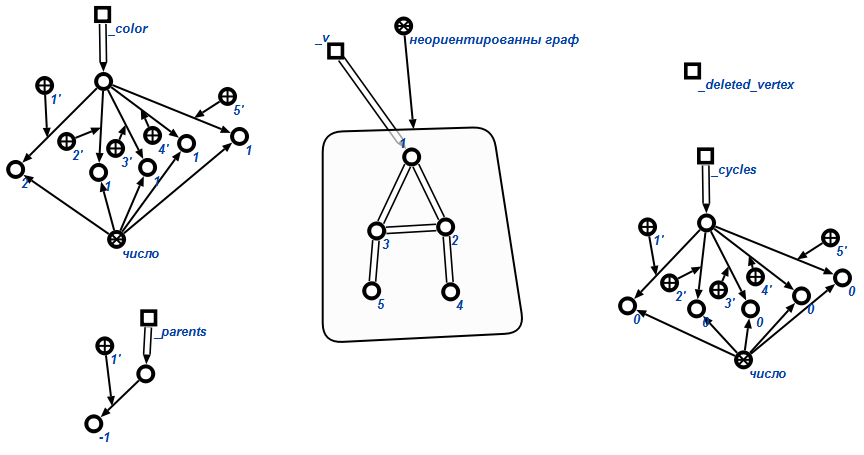


Рисунок

6. Меняем "цвет" вершины v на просматриваемый.

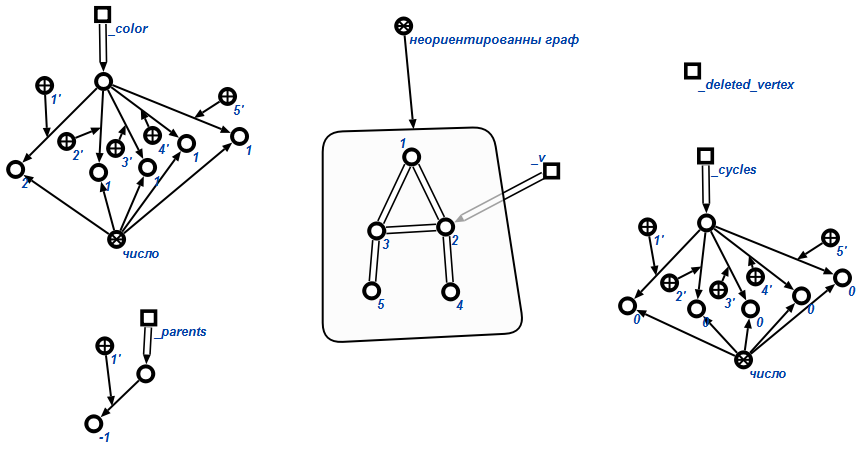
Рисунок

7. Так как вершина v - начальная, то её родителем будет число -1.



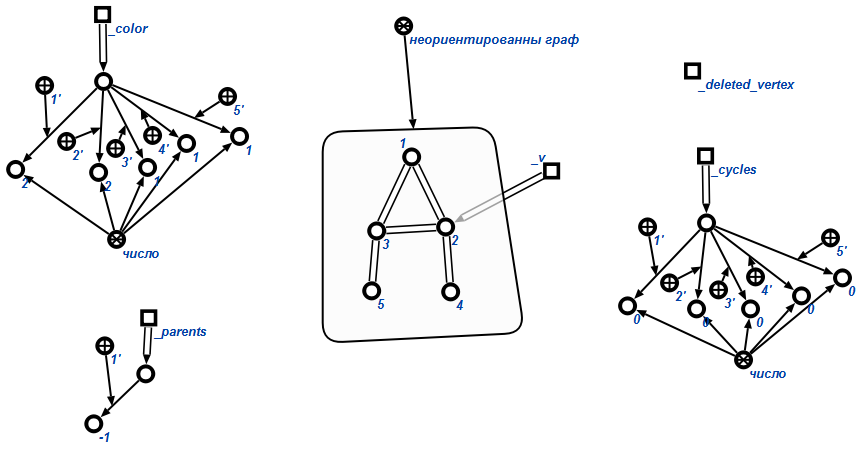
Рисунок

8.Переходим в вершину смежную v,цвет которой отмечен как не просмотренный и которая не является родителем v.



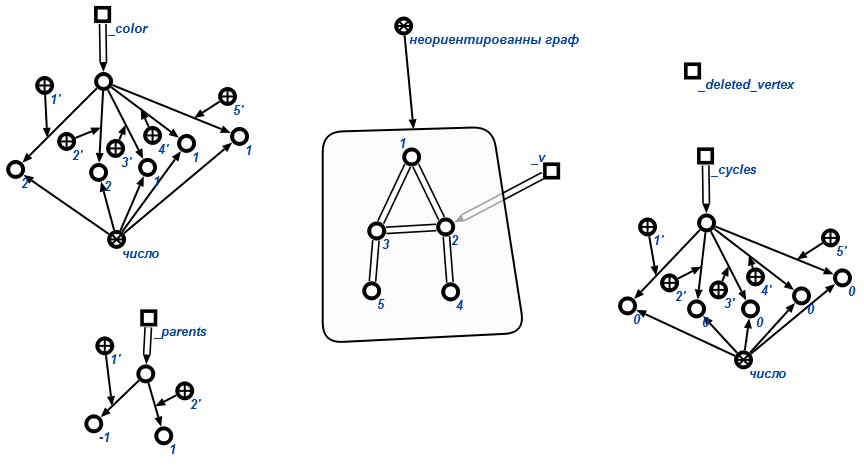
Рисунок

9.Меняем цвет вершины v на просматриваемый.



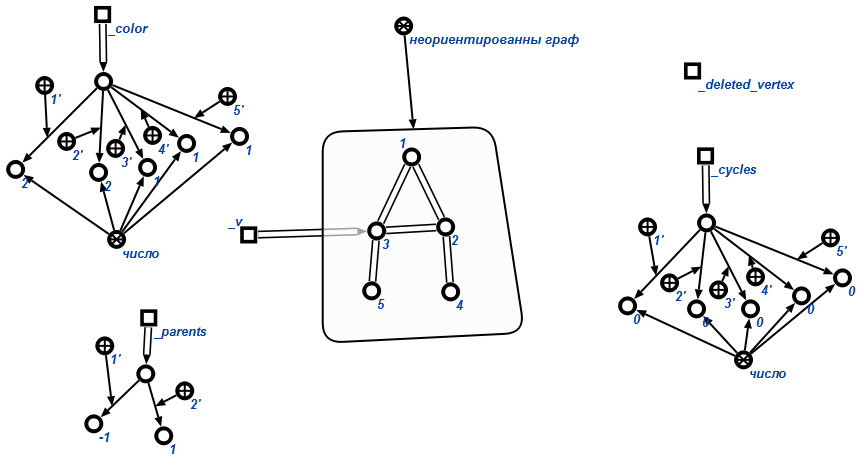
Рисунок

10. Записываем родителя вершины v.



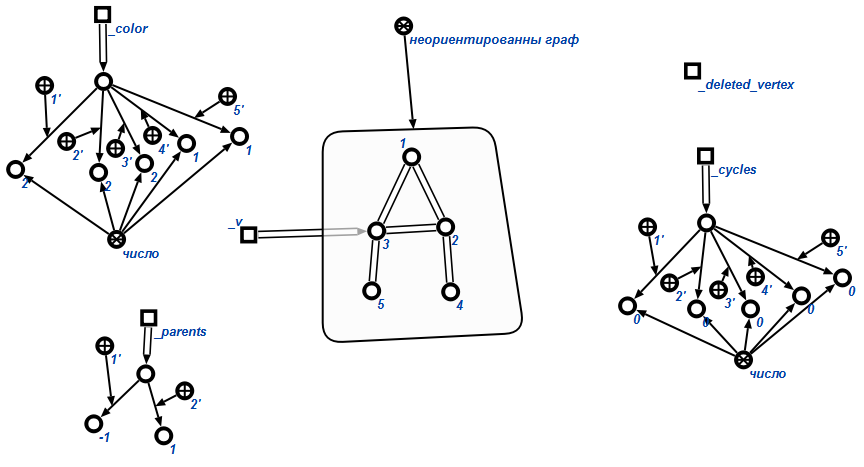
Рисунок

11. Переходим в вершину смежную v,цвет которой отмечен как не просмотренный и которая не является родителем v.



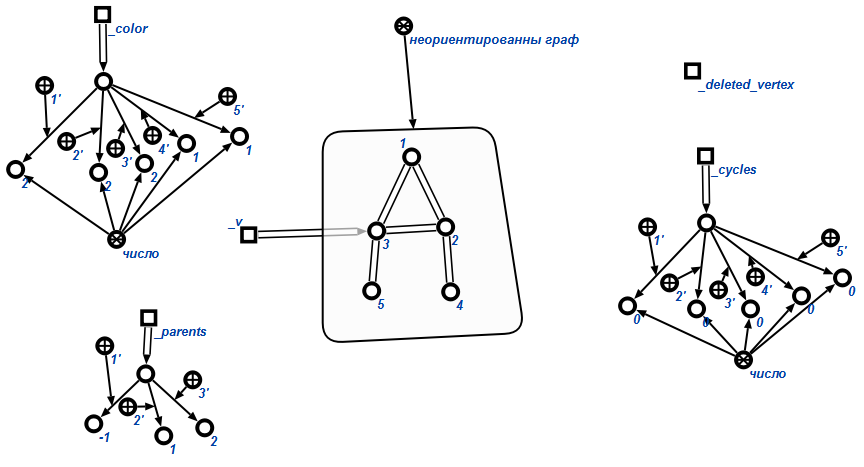
Рисунок

12. Меняем цвет вершины v на просматриваемый.



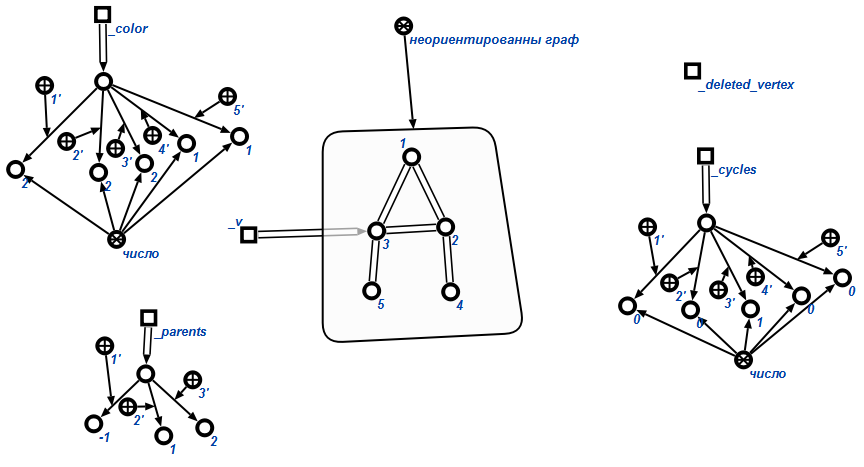
Рисунок

13. Записываем родителя вершины v.



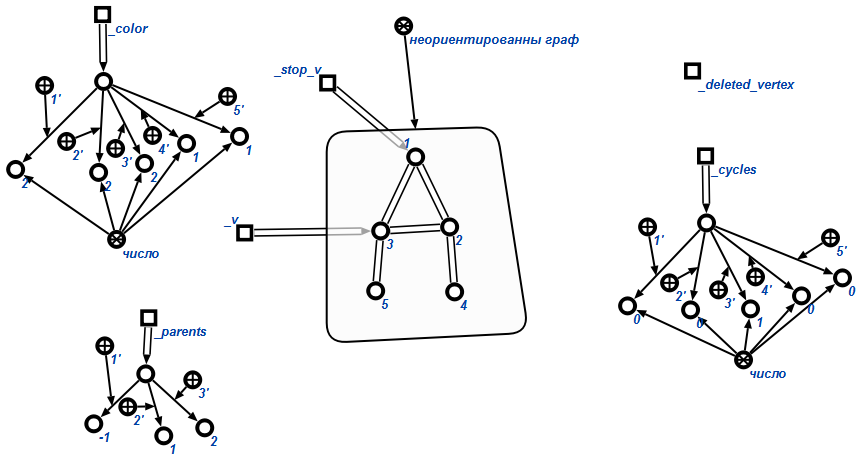
Рисунок

14.Так как цвет вершины смежной вершине v отмечен как просматриваемый и которая не является родителем v , считаем что мы нашли простой цикл. Увеличиваем приоритет на удаление вершины v.



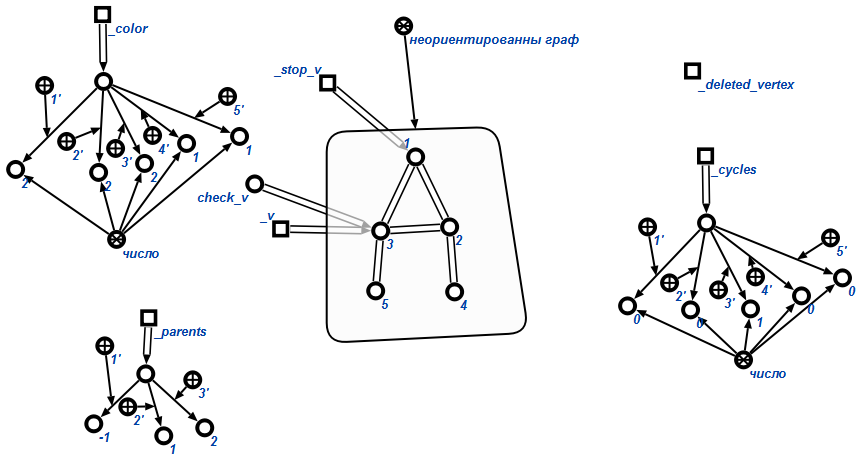
Рисунок

15. Отмечаем вершину смежную v и цвет которой отмечен как просматриваемый переменной stop\_v



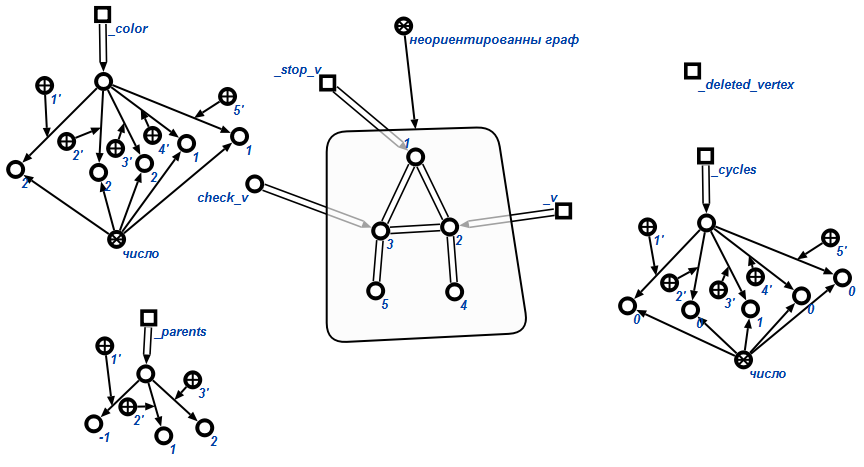
Рисунок

16.Отмечаем вершину v переменной check\_v.



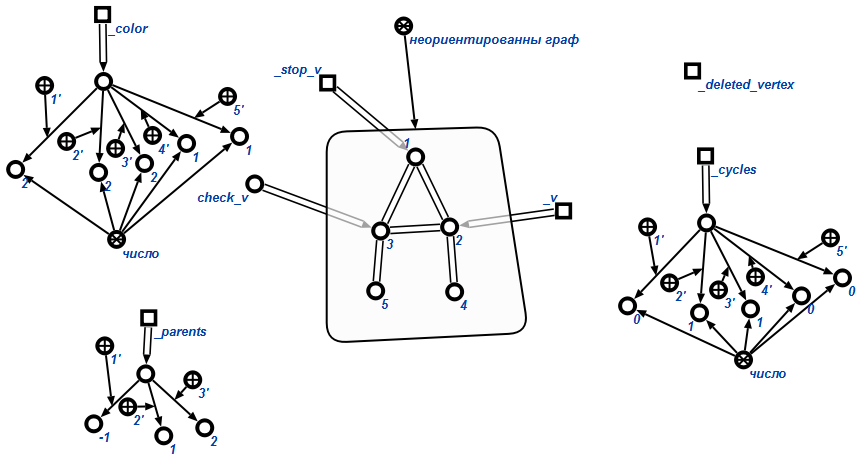
Рисунок

17. Переходим в вершину которая является родителем для вершины v.



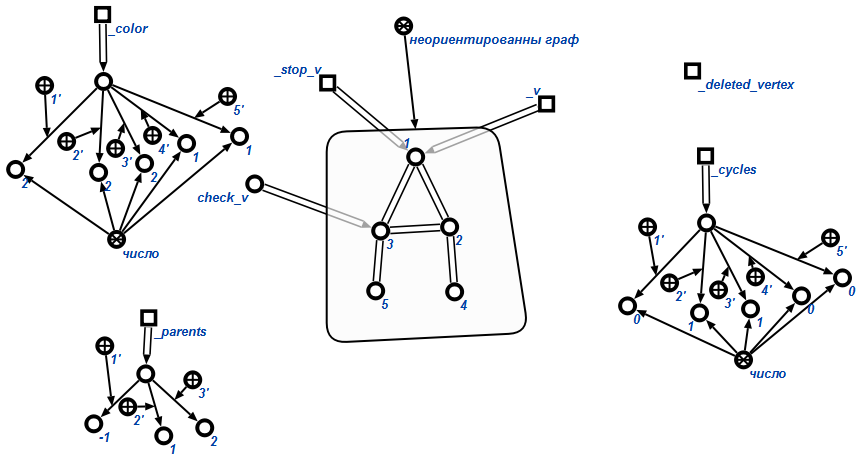
Рисунок

18. Увеличиваем приоритет на удаление вершины v.



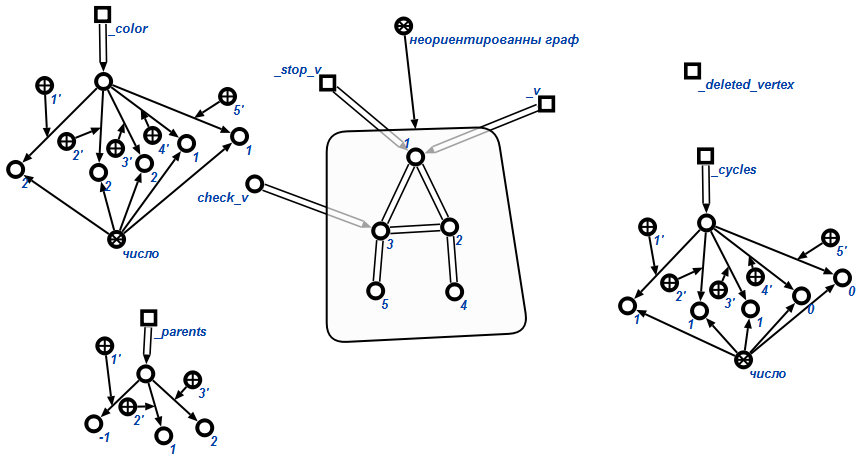
Рисунок

19. Переходим в вершину которая является родителем для вершины v.



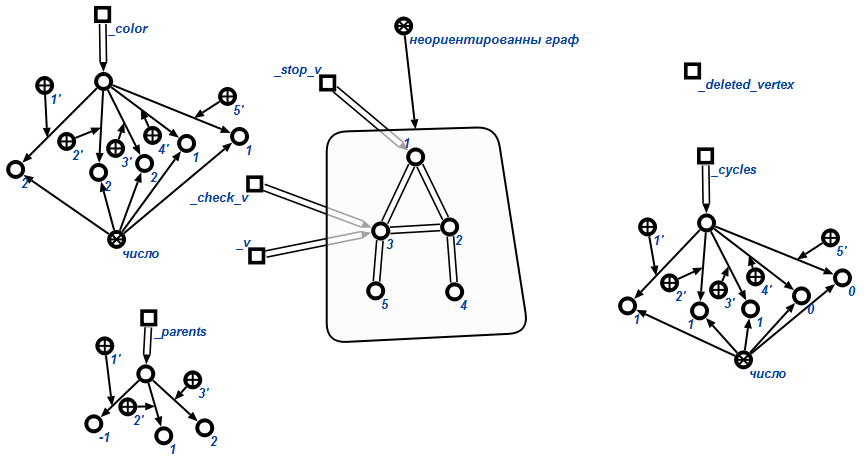
Рисунок

20. Увеличиваем приоритет на удаление вершины v.



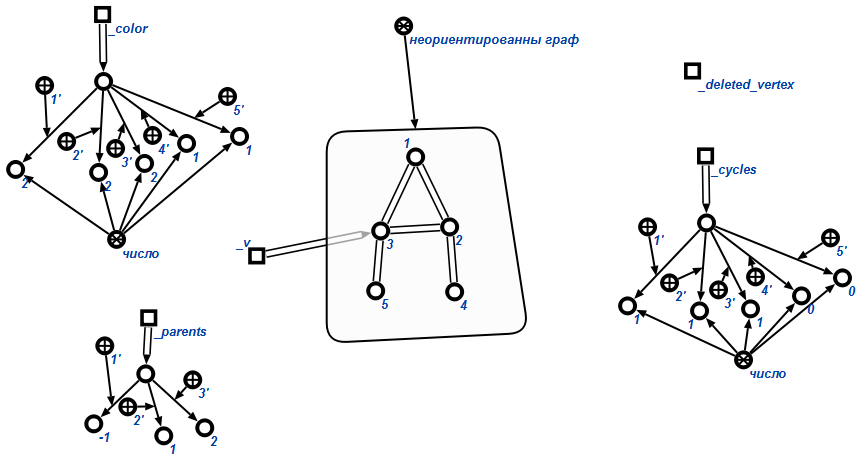
Рисунок

21. Так как переменная v совпала с вершиной stop\_v переходим в вершину check\_v.



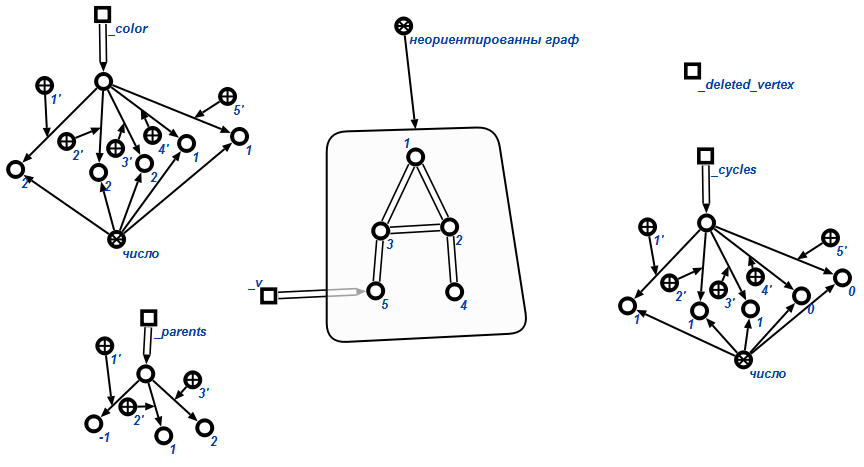
Рисунок

22.Удаляем переменные check\_v и stop\_v.



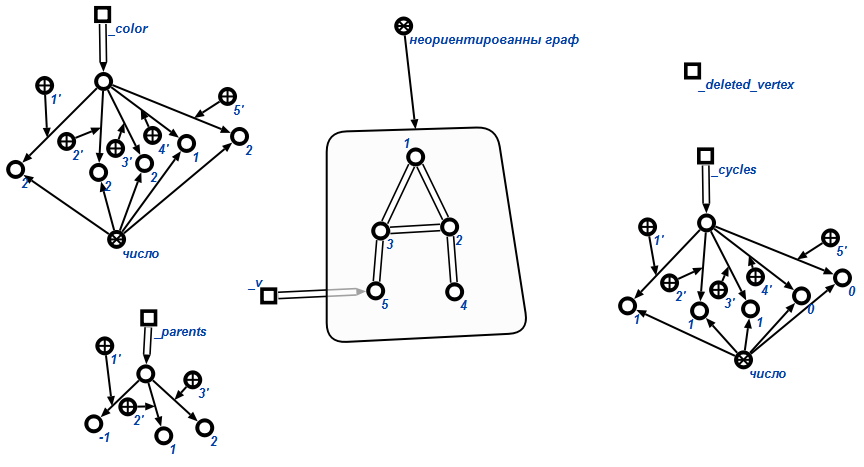
Рисунок

23.Переходим в следующую вершину смежную v цвет которой отмечен как не просмотренный и которая не является родителем v.



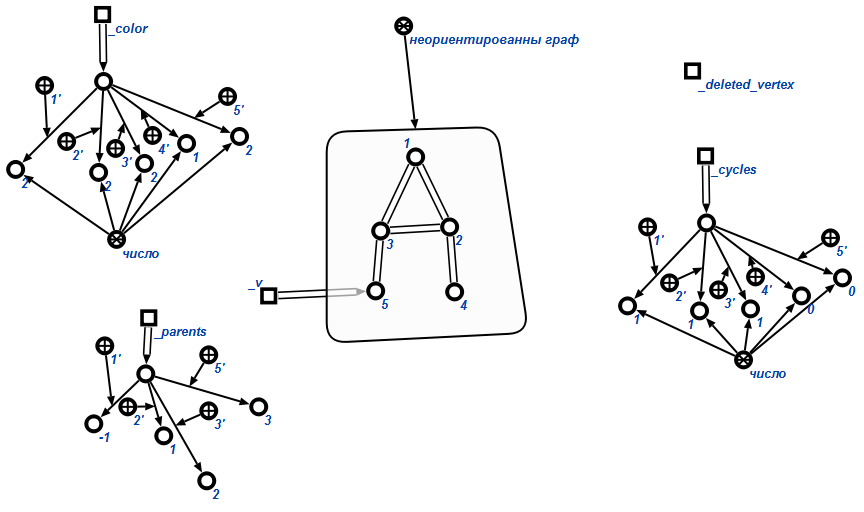
Рисунок

24. Меняем цвет вершины v на просматриваемый.



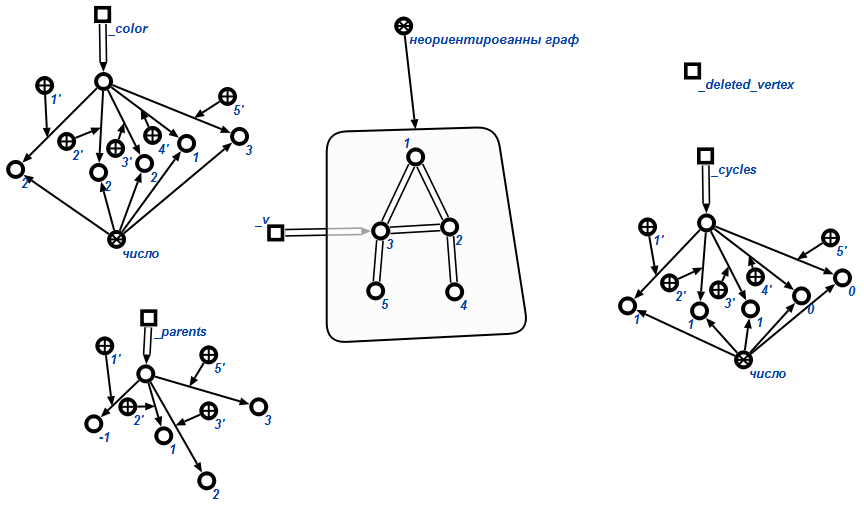
Рисунок

25. Записываем родителя вершины v.



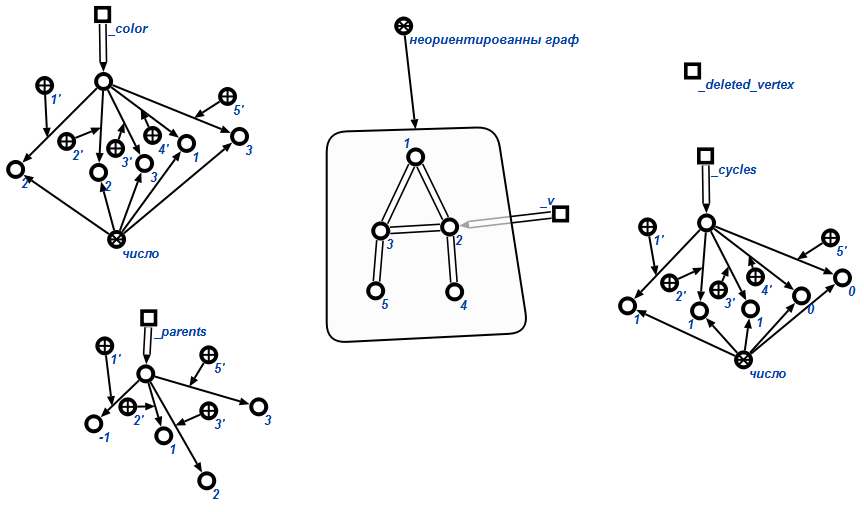
Рисунок

26.Так как для вершины v нет смежных вершин отмеченных цветом не просмотренный и которые не являлись бы родителями для данной вершины то отмечаем вершину v как просмотренную и переходим в вершину которая является родителем для вершины v.



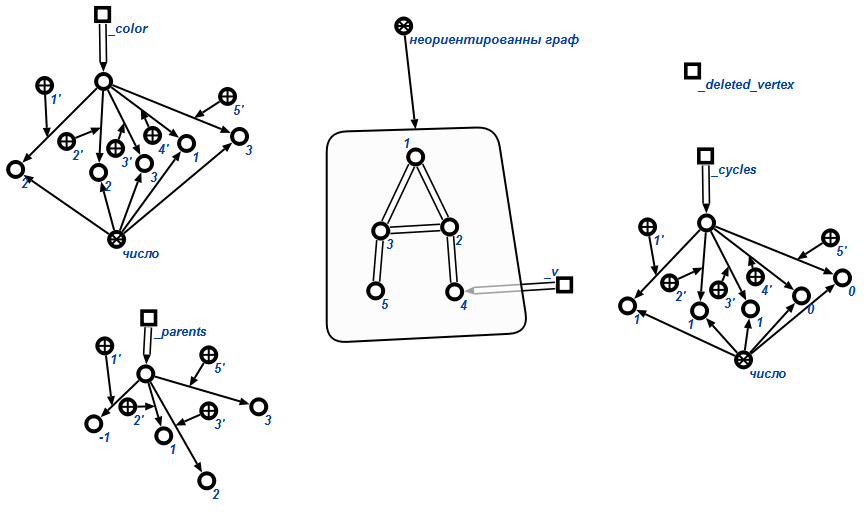
Рисунок

27. Так как для вершины v нет смежных вершин отмеченных цветом не просмотренный и которые не являлись бы родителями для данной вершины то отмечаем вершину v как просмотренную и переходим в вершину которая является родителем для вершины v.



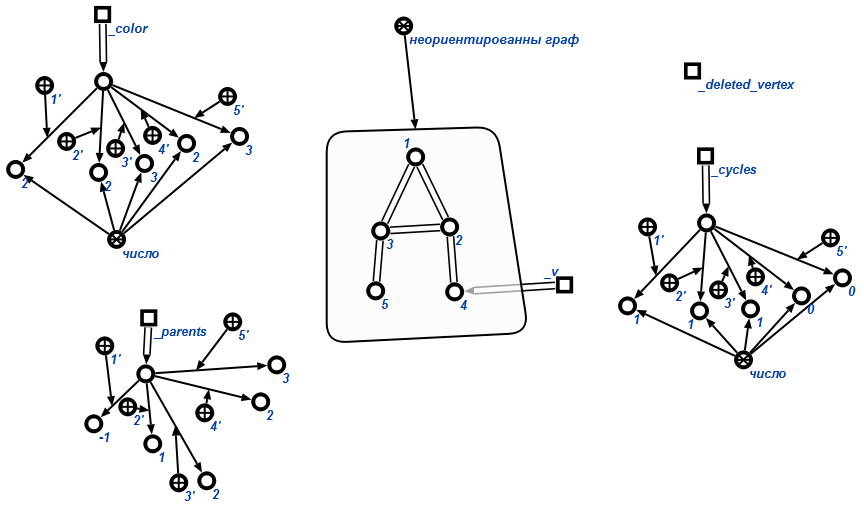
Рисунок

28. Переходим в вершину смежную v,цвет которой отмечен как не просмотренный и которая не является родителем v.



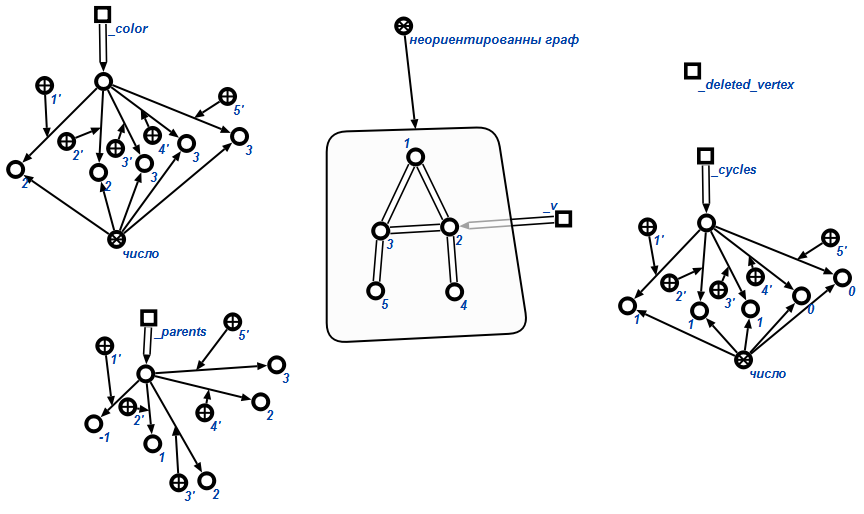
Рисунок

29.Отмечаем цвет вершины v как просматриваемый и добавляем родителя v.



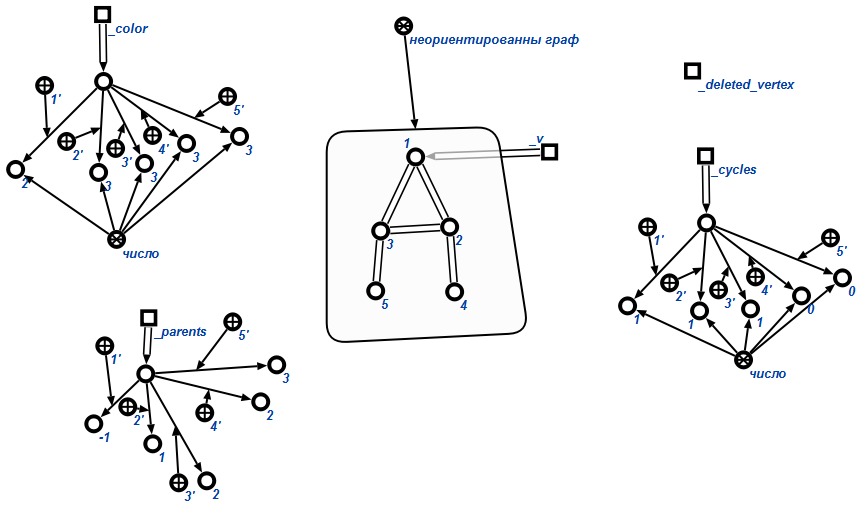
Рисунок

30. Так как для вершины v нет смежных вершин отмеченных цветом не просмотренный и которые не являлись бы родителями для данной вершины то отмечаем вершину v как просмотренную и переходим в вершину которая является родителем для вершины v.



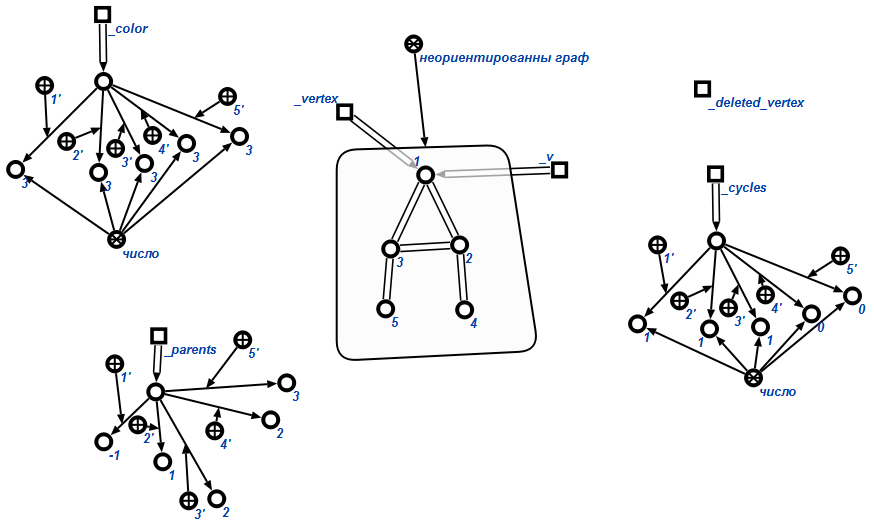
Рисунок

31. Так как для вершины v нет смежных вершин отмеченных цветом не просмотренный и которые не являлись бы родителями для данной вершины то отмечаем вершину v как просмотренную и переходим в вершину которая является родителем для вершины v.



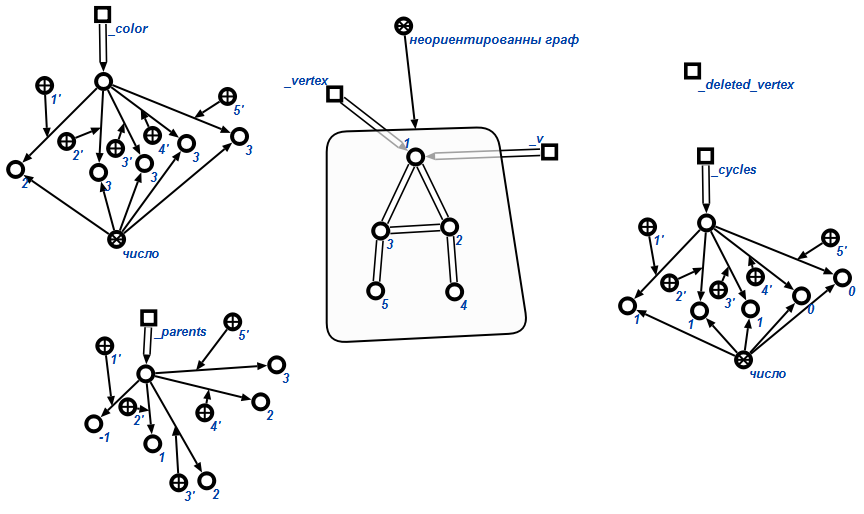
Рисунок

31. Так как для вершины v нет смежных вершин отмеченных цветом не просмотренный и которые не являлись бы родителями для данной вершин, и родитель вершины v равен -1 считаем что мы просмотрели все вершины графа. Отмечаем вершину v как просмотренную.



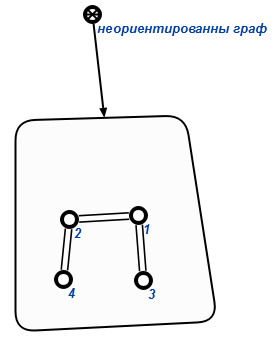
Рисунок

32.Далее выбираем вершину для удаления в зависимости от её приоритета на удаления(в случае равенства приоритета нескольких вершин берем любую из них).Отмечаем её переменной vertex.



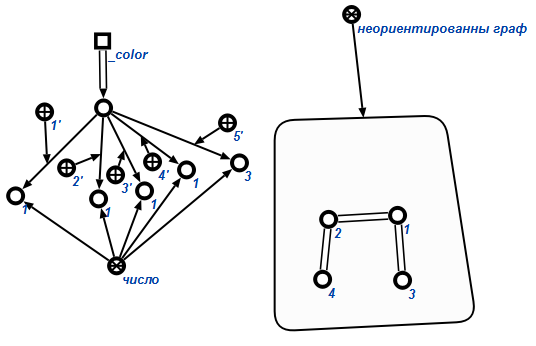
Рисунок

33. Проверяем граф на связанность. создаём временный граф без вершины vertex.



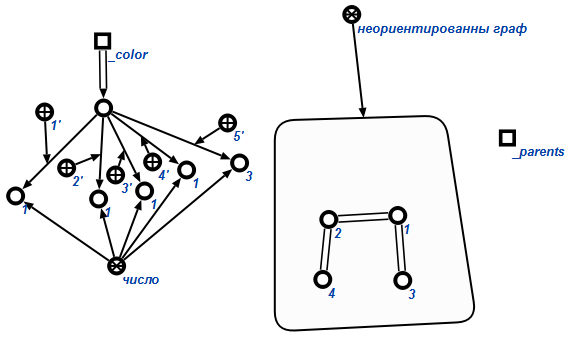
Рисунок

34. Отмечаем все вершины временного графа как не просмотренные.



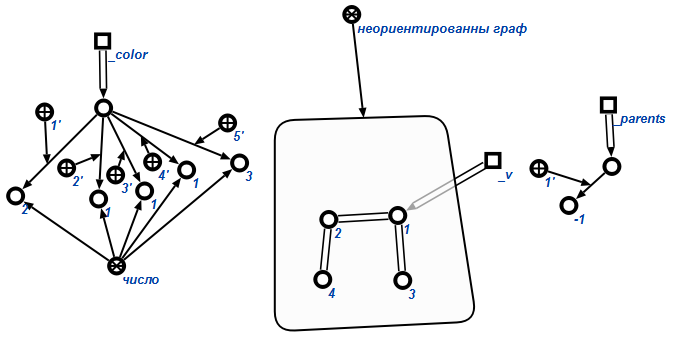
Рисунок

35. Удаляем все элементы множества perents.



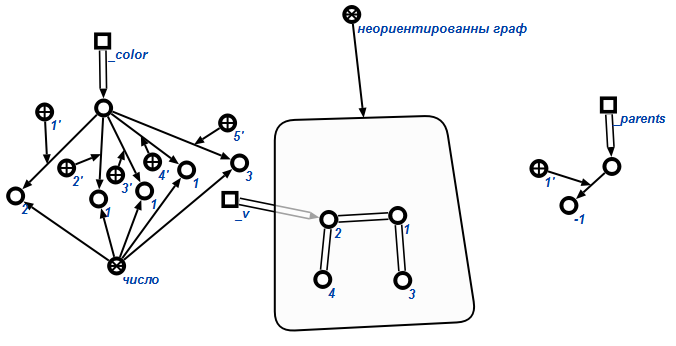
Рисунок

36. Обозначаем вершину, в которой мы находимся, переменной v (выбираем вершину) и обозначаем её цвет как просматриваемый так как v первая вершина её родитель равен -1.



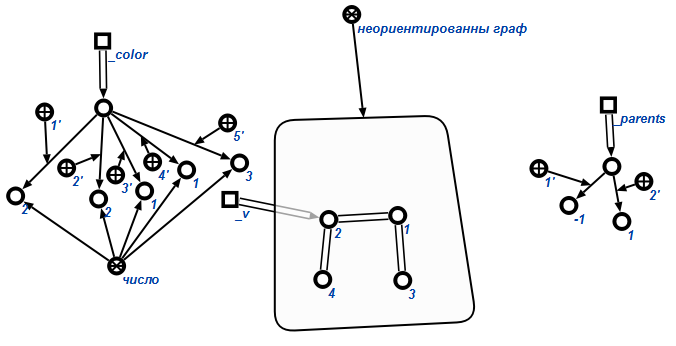
Рисунок

37. Переходим в вершину смежную v,цвет которой отмечен как не просмотренный и которая не является родителем v.



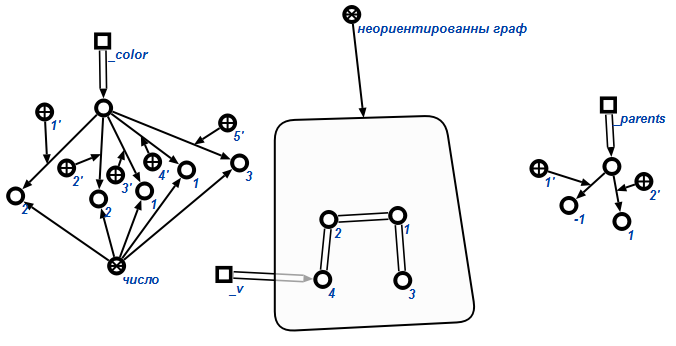
Рисунок

39.Меняем цвет вершины v на просматриваемый и добавляем родителя вершины v.



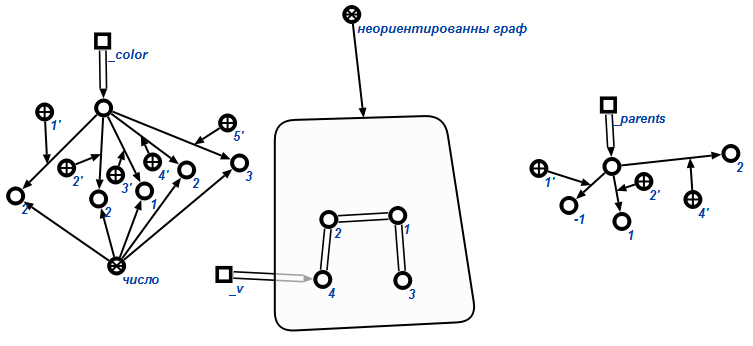
Рисунок

40. Переходим в вершину смежную v,цвет которой отмечен как не просмотренный и которая не является родителем v.



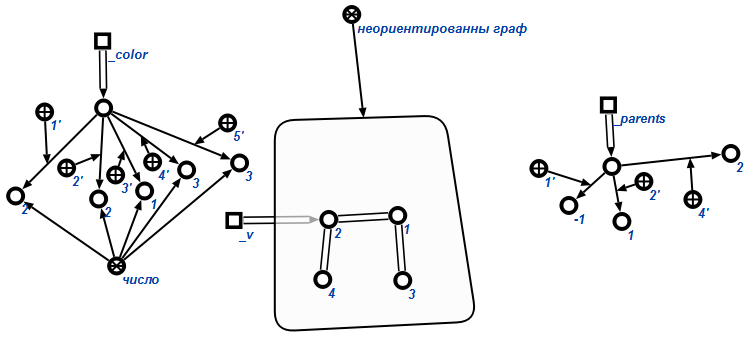
Рисунок

41. Меняем цвет вершины v на просматриваемый и добавляем родителя вершины v.



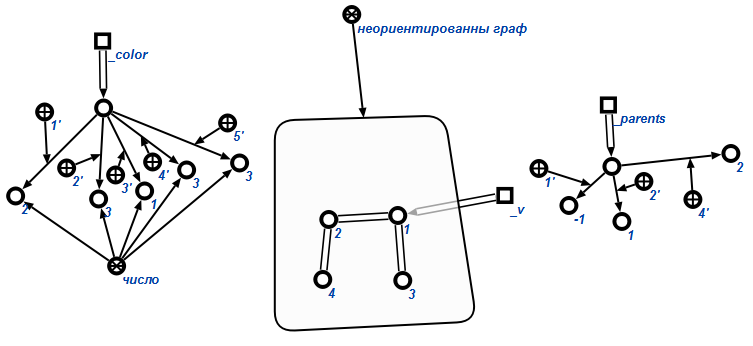
Рисунок

42. Так как для вершины v нет смежных вершин отмеченных цветом не просмотренный и которые не являлись бы родителями для данной вершины то отмечаем вершину v как просмотренную и переходим в вершину которая является родителем для вершины v.



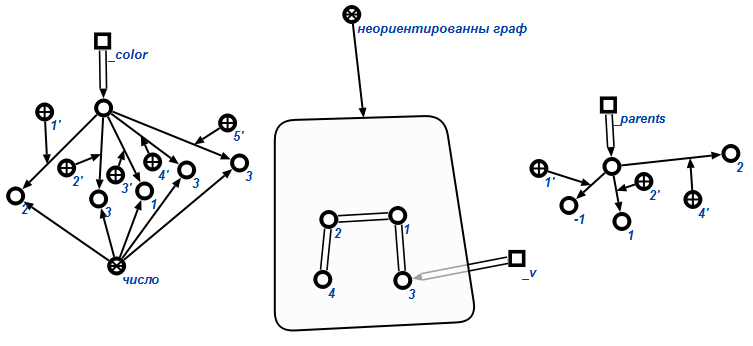
Рисунок

43. Так как для вершины v нет смежных вершин отмеченных цветом не просмотренный и которые не являлись бы родителями для данной вершины то отмечаем вершину v как просмотренную и переходим в вершину которая является родителем для вершины v.



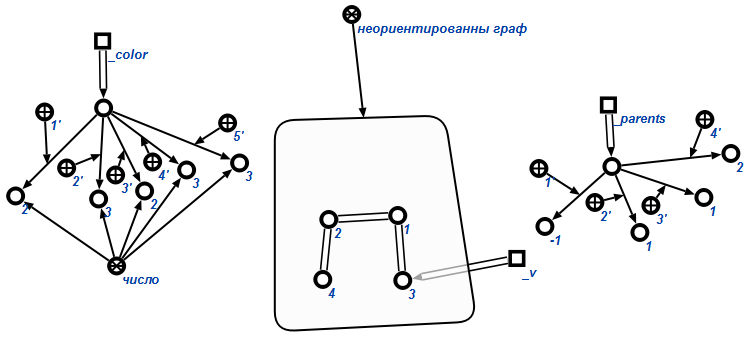
Рисунок

44. Переходим в вершину смежную v,цвет которой отмечен как не просмотренный и которая не является родителем v.



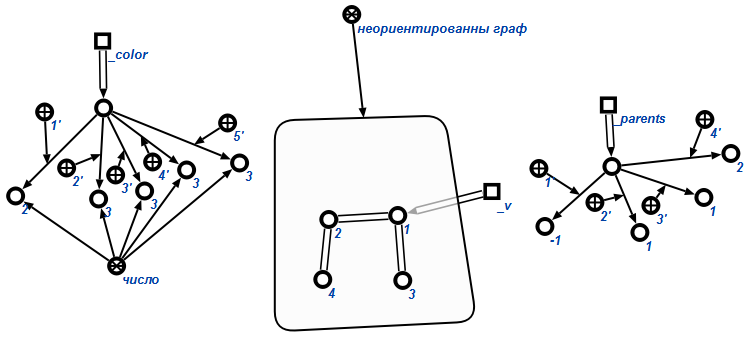
Рисунок

45. Меняем цвет вершины v на просматриваемый и добавляем родителя вершины v.



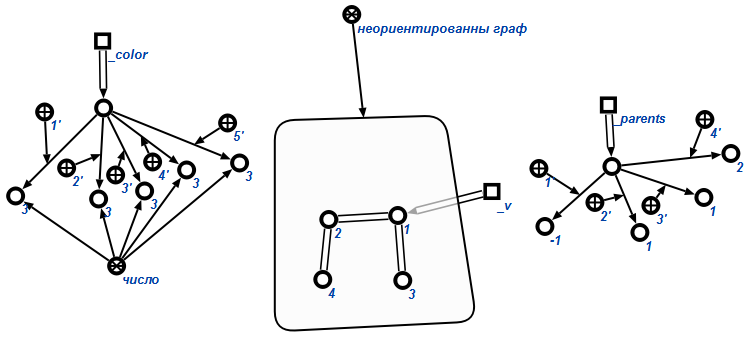
Рисунок

46. Так как для вершины v нет смежных вершин отмеченных цветом не просмотренный и которые не являлись бы родителями для данной вершины то отмечаем вершину v как просмотренную и переходим в вершину которая является родителем для вершины v.



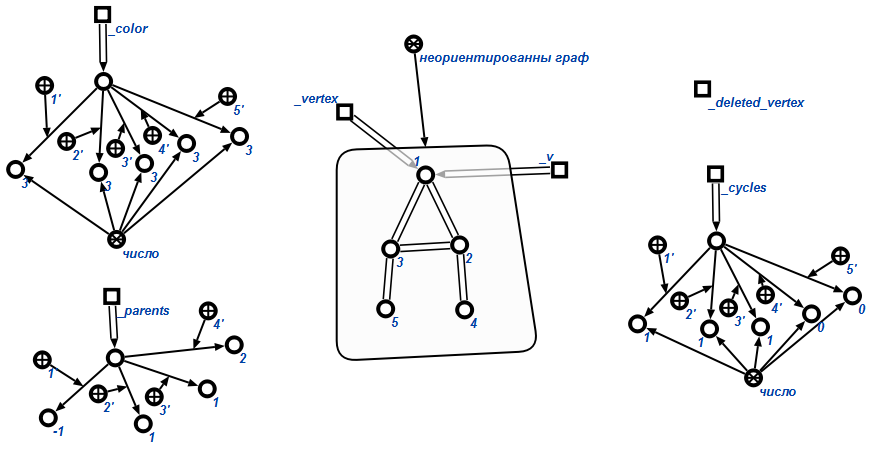
Рисунок

47 . Так как для вершины v нет вершин отмеченных цветом не просмотренный и которые не являлись бы родителями для данной вершин, и родитель вершины v равен -1 считаем что мы просмотрели все вершины графа. Отмечаем вершину v как просмотренную.



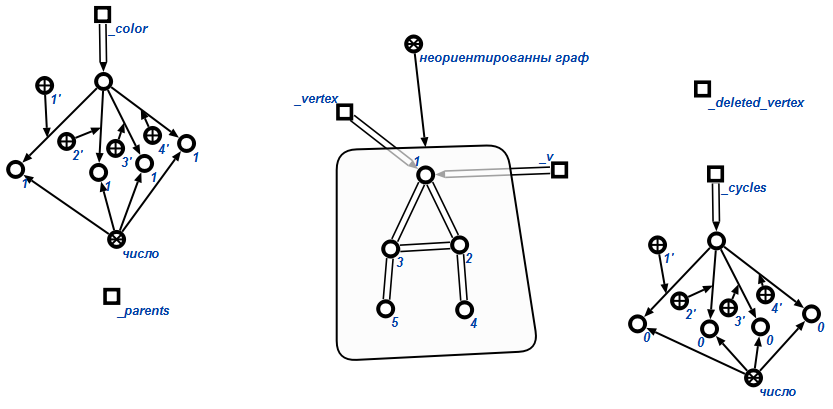
Рисунок

48. Так как все вершины временного графа отмечены как просмотренные считаем что граф остается связанным после удаления вершины vertex.Удаляем временный граф.



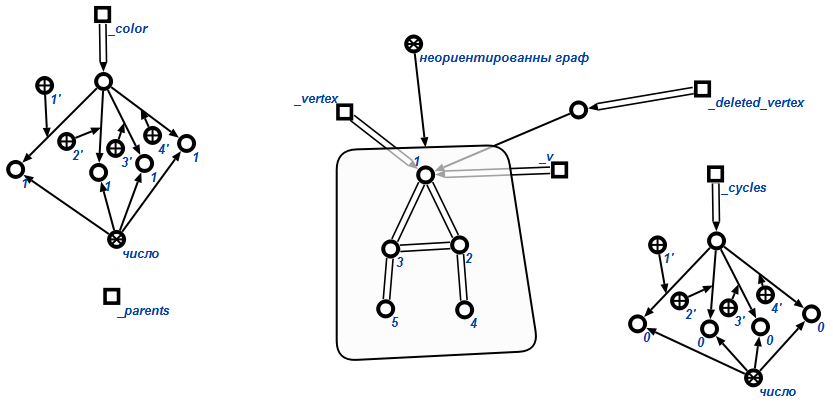
Рисунок

49. Удаляем все элементы множества parents. Удаляем 1 элемент множества color. Удаляем 1 элемент множества cycles. Отмечаем цвет всех вершин как не просмотренный. Ставим приоритет удаления всех вершин на 0.



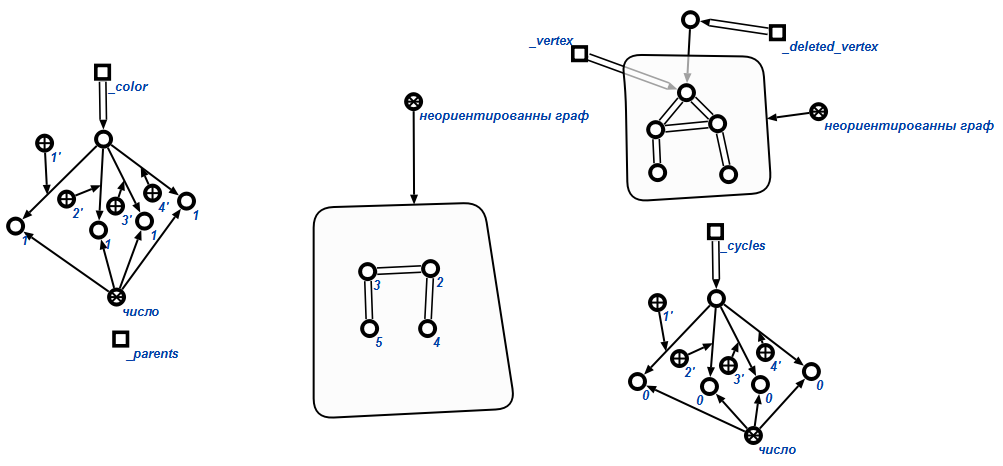
Рисунок

50. Заносим вершину vertex в множество deleted\_vertex.



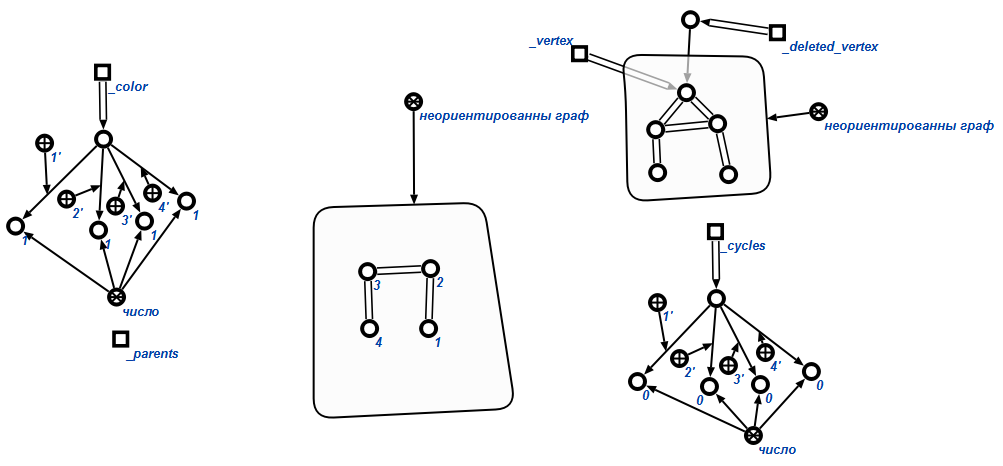
Рисунок

51.Создаем граф равный исходному. Удаляем вершину vertex из исходного графа и отмечаем её в скопированном графе.



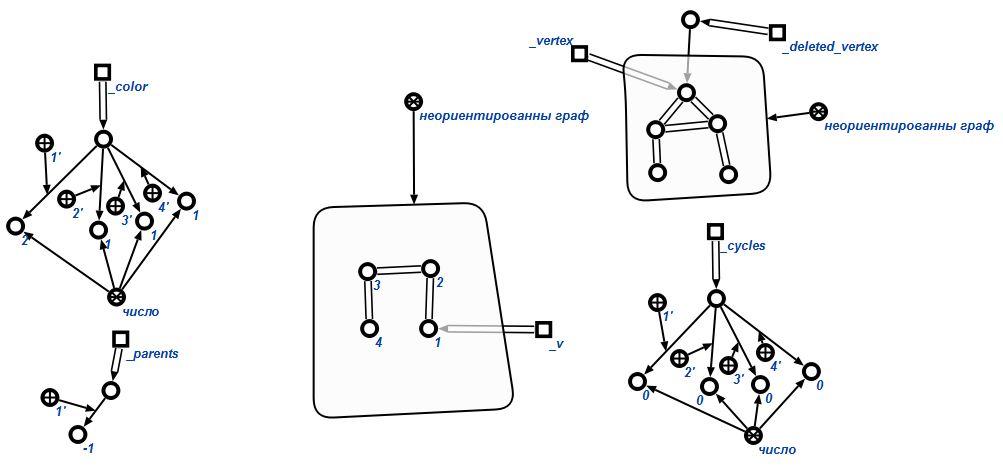
Рисунок

52. Заново нумеруем вершины графа.



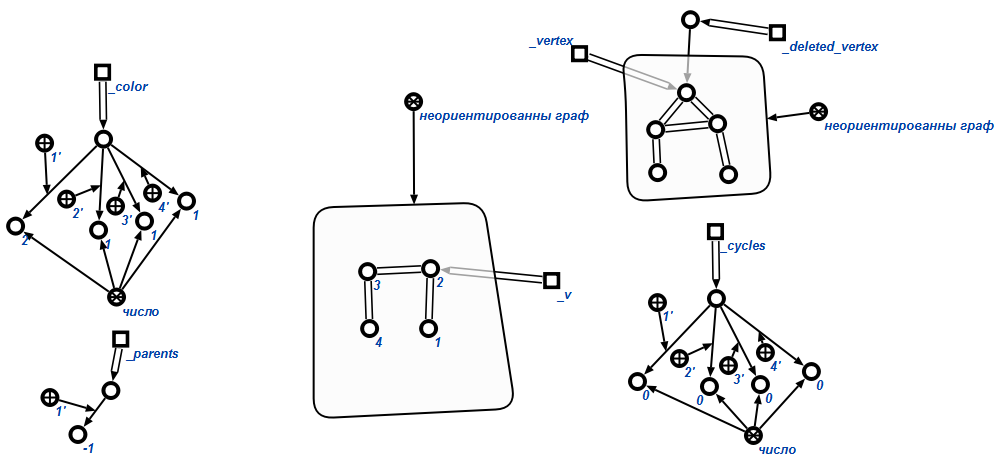
Рисунок

53. Обозначаем вершину, в которой мы находимся, переменной v (выбираем вершину) и обозначаем её цвет как просматриваемый так как v первая вершина её родитель равен -1.



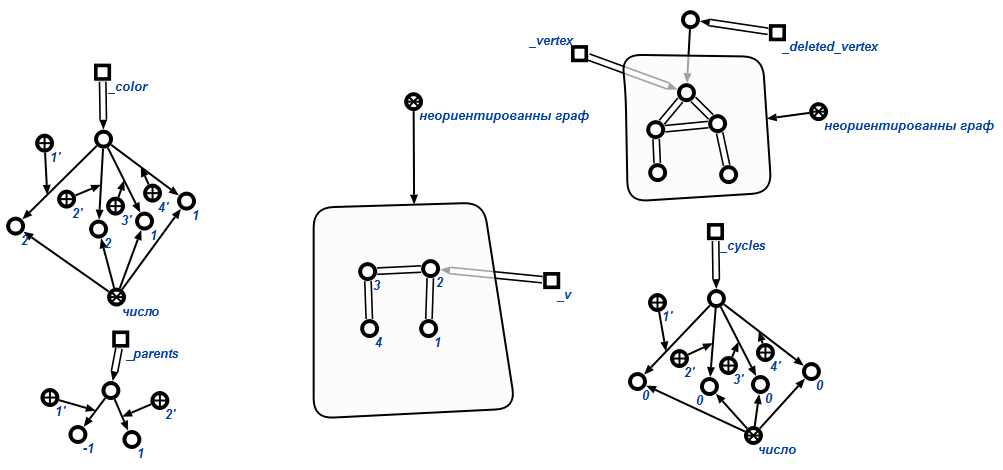
Рисунок

54. Переходим в вершину смежную v,цвет которой отмечен как не просмотренный и которая не является родителем v.



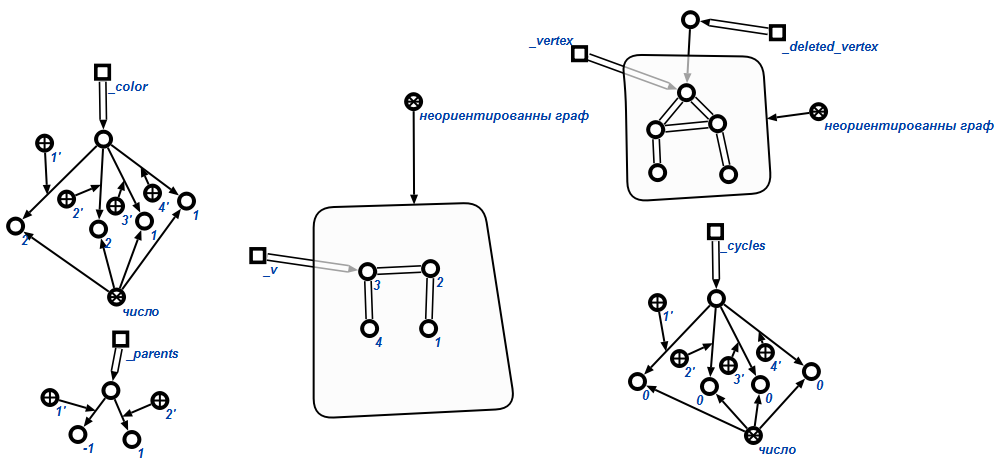
Рисунок

55. Меняем цвет вершины v на просматриваемый и добавляем родителя вершины v.



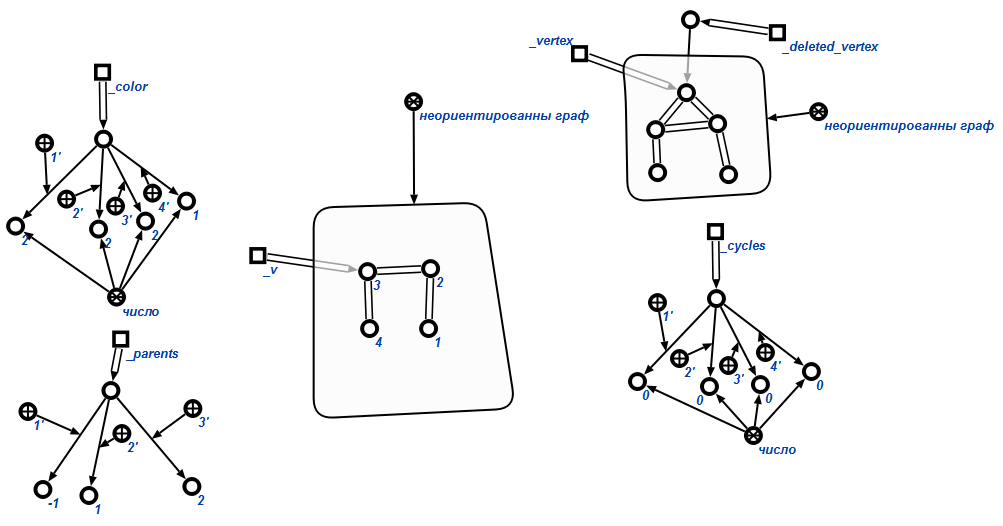
Рисунок

56. Переходим в вершину смежную v,цвет которой отмечен как не просмотренный и которая не является родителем v.



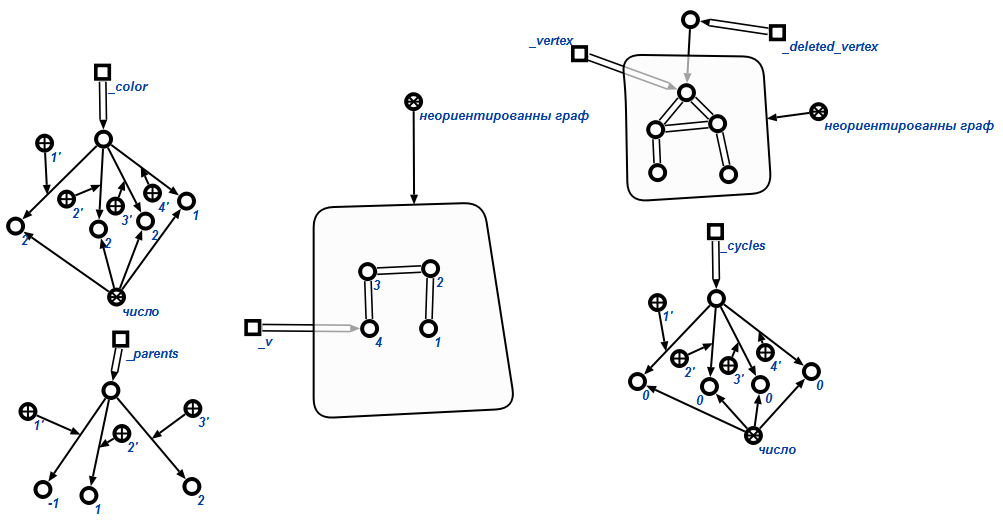
Рисунок

57. Меняем цвет вершины v на просматриваемый и добавляем родителя вершины v.



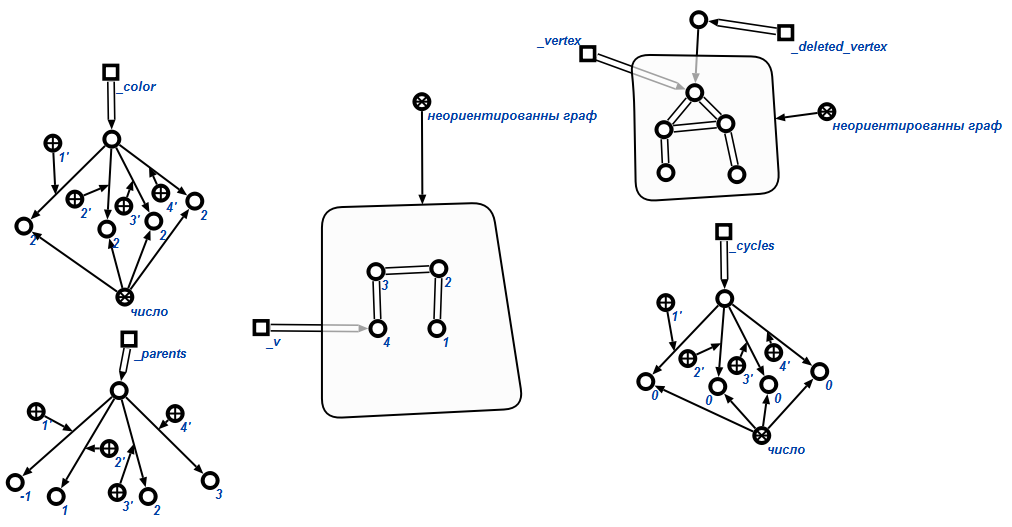
Рисунок

58. Переходим в вершину смежную v,цвет которой отмечен как не просмотренный и которая не является родителем v.



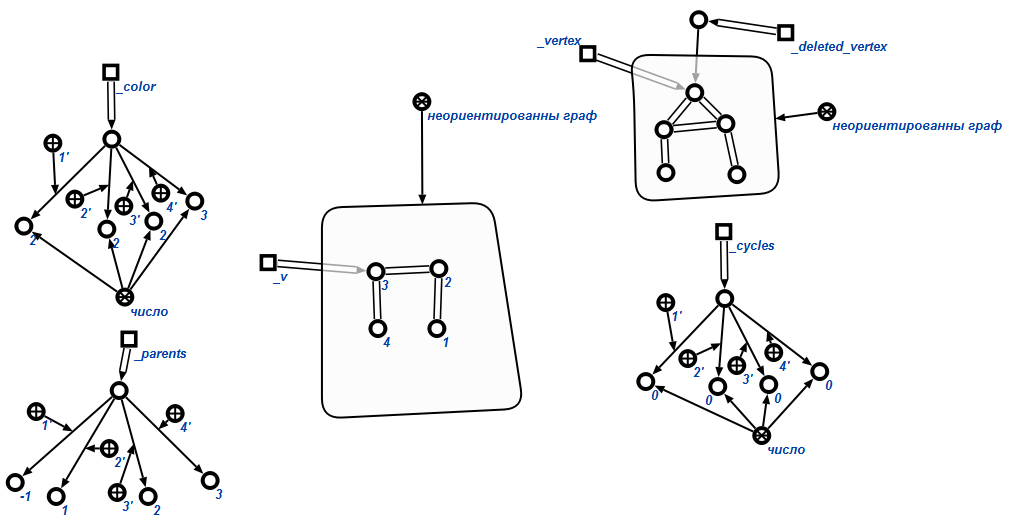
Рисунок

59. Меняем цвет вершины v на просматриваемый и добавляем родителя вершины v.



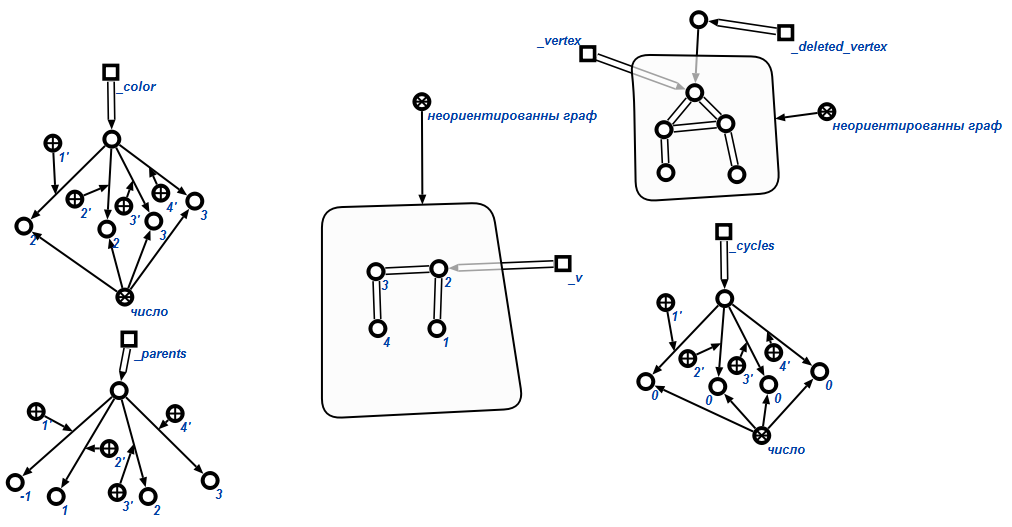
Рисунок

60. Так как для вершины v нет смежных вершин отмеченных цветом не просмотренный и которые не являлись бы родителями для данной вершины то отмечаем вершину v как просмотренную и переходим в вершину которая является родителем для вершины v.



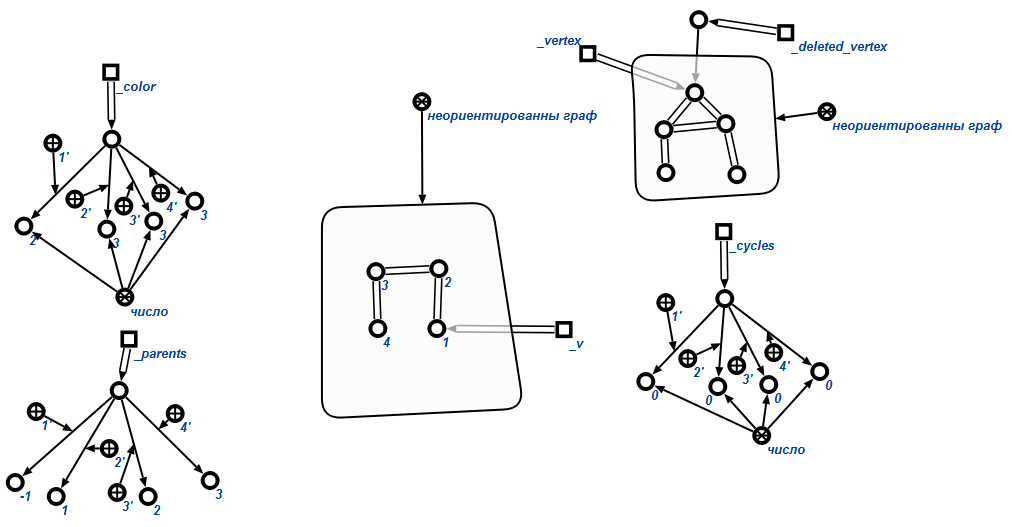
Рисунок

61. Так как для вершины v нет смежных вершин отмеченных цветом не просмотренный и которые не являлись бы родителями для данной вершины то отмечаем вершину v как просмотренную и переходим в вершину которая является родителем для вершины v.



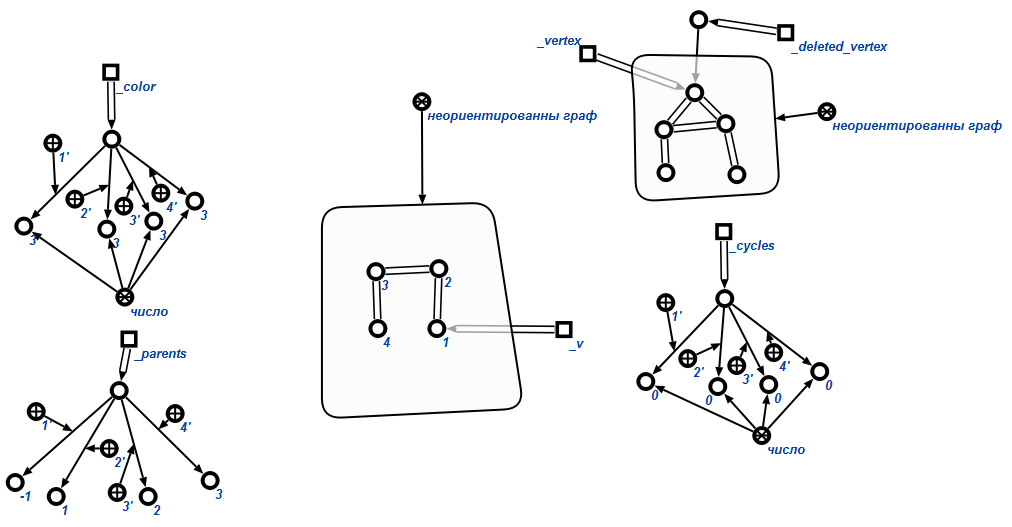
Рисунок

62. Так как для вершины v нет смежных вершин отмеченных цветом не просмотренный и которые не являлись бы родителями для данной вершины то отмечаем вершину v как просмотренную и переходим в вершину которая является родителем для вершины v.



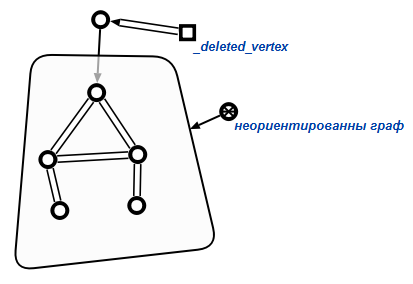
Рисунок

63. Так как для вершины v нет вершин отмеченных цветом не просмотренный и которые не являлись бы родителями для данной вершин, и родитель вершины v равен -1 считаем что мы просмотрели все вершины графа. Отмечаем вершину v как просмотренную.



Рисунок

64.Так как приоритет на удаления всех вершин равен 0, считаем что граф превратился в дерево. Удаляем все множества корме deleted\_vertex. Удаляем все переменные. Удаляем исходный граф.



Рисунок

# 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.