

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  
**Найти количество компонент связности в неориентированном графе.**

Выполнил:

Д. А. Путято

Студент группы  
321701

Проверил:

В. Н. Тищенко

Минск 2024

# 1 Введение

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

**Задача:** Найти количество компонент связности в неориентированном графе.

## 2 Список понятий

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

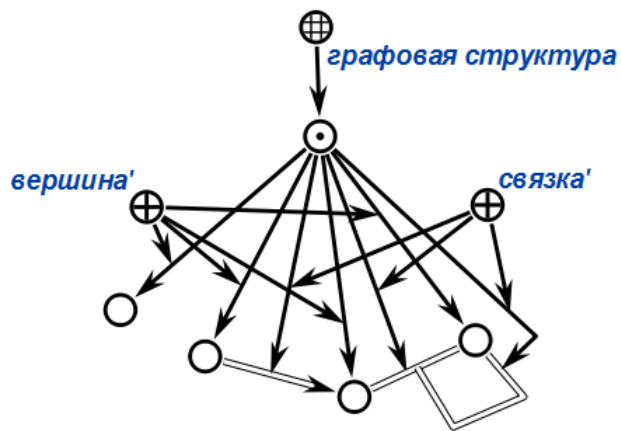


Figure 1: Абсолютное понятие графовой структуры

2. *Графовая структура с ориентированными связками* (абсолютное понятие)

– Ориентированная связка (относительное понятие, ролевое отношение) – связка, которая задается ориентированным множеством.

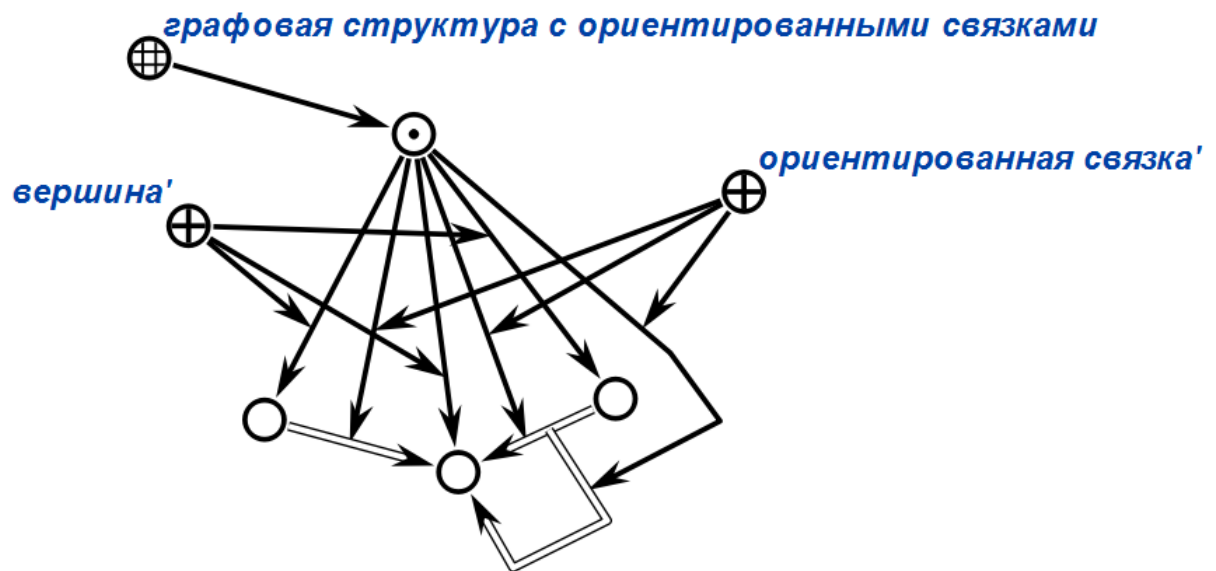


Figure 2: Графовая структура с ориентированными связками

3. *Графовая структура с неориентированными связками* (абсолютное понятие)

– Неориентированная связка (относительное понятие, ролевое отношение) – связка, которая задается неориентированным множеством.

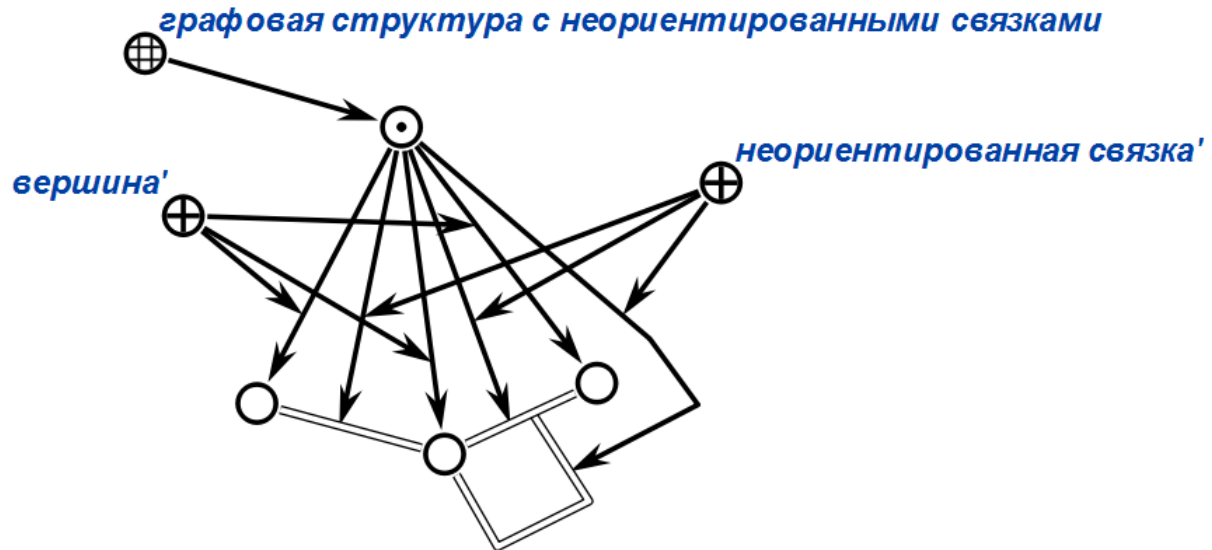


Figure 3: Графовая структура с неориентированными связками

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

- Гиперсвязка (относительное понятие, ролевое отношение);
- Гипердуга (относительное понятие, ролевое отношение) – ориентированная гипер- связь;
- Гиперребро (относительное понятие, ролевое отношение) – неориентированная гипер- связь.

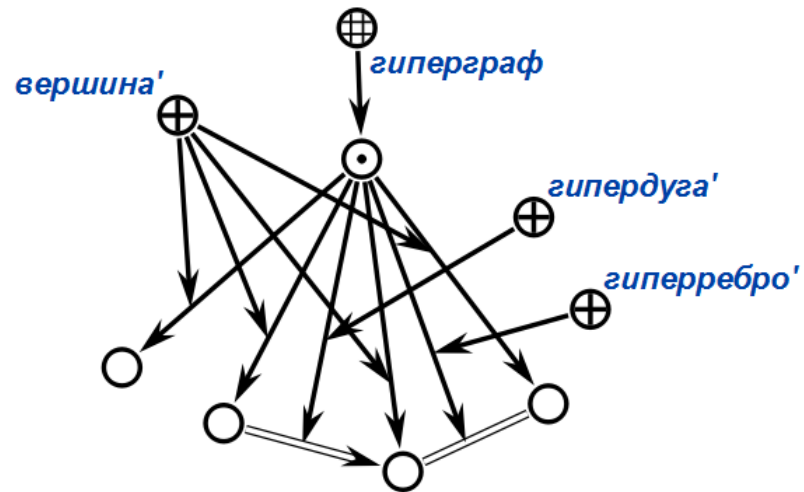


Figure 4: Гиперграф

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

- Бинарная связка (относительное понятие, ролевое отношение) – гиперсвязка арности 2;
- Ребро (относительное понятие, ролевое отношение) – неориентированная гиперсвязка;
- Дуга (относительное понятие, ролевое отношение) – ориентированная гиперсвязка;
- Петля (относительное понятие, ролевое отношение) – бинарная связка, у которой первый и второй компоненты совпадают.

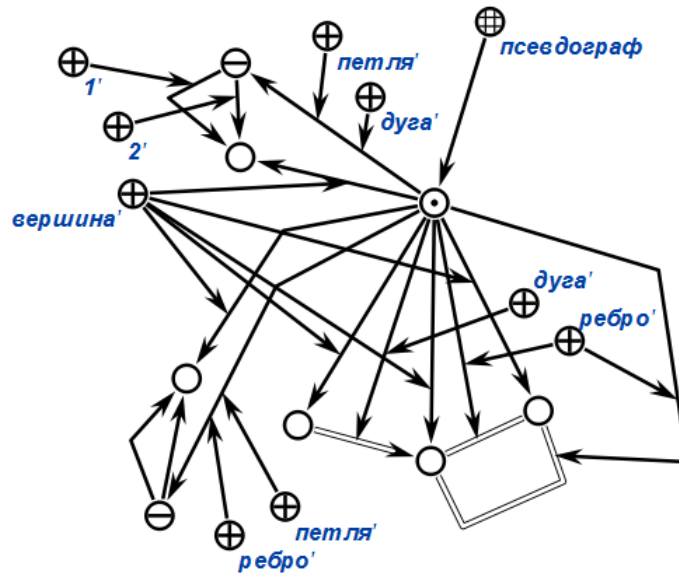


Figure 5: Псевдограф

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

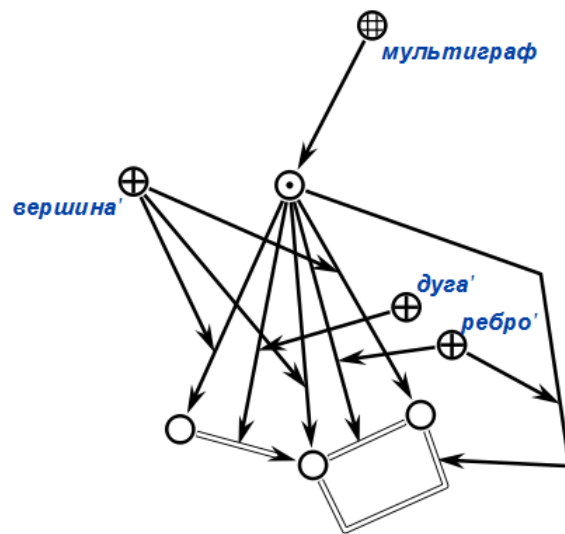


Figure 6: Мультиграф

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

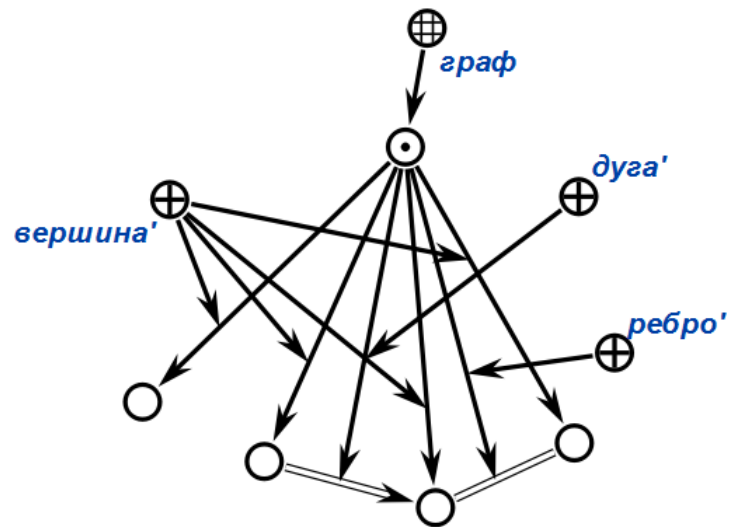


Figure 7: Граф



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

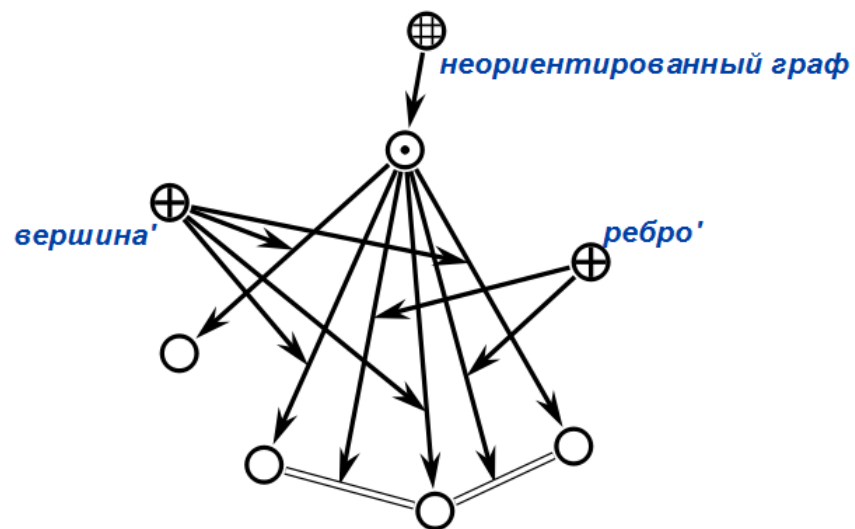


Figure 8: Неориентированный граф

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

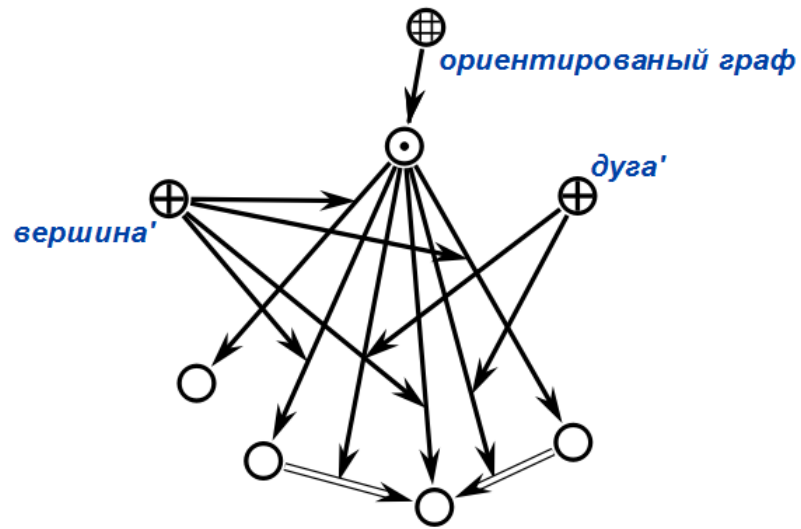


Figure 9: Ориентированный граф

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, "1", B, "2", C, "3", D, "1", A в гиперграфе. "1" – это тернарная неориентированная связка (гиперсвязка), а "2" и "3" – бинарные связки (гиперсвязки).

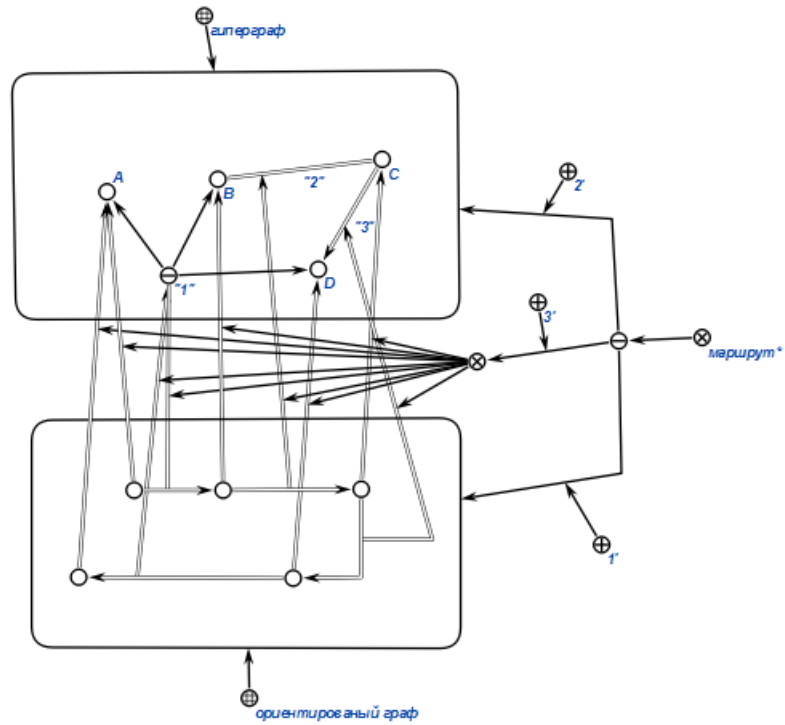


Figure 10: Маршрут

11. **Цепь** (относительное понятие, бинарное ориентированное отношение) – это маршрут, все гиперсвязки которого различны. В примере ниже показана цепь A, "1", B, "2", C, "3", D, "4", A в гиперграфе.

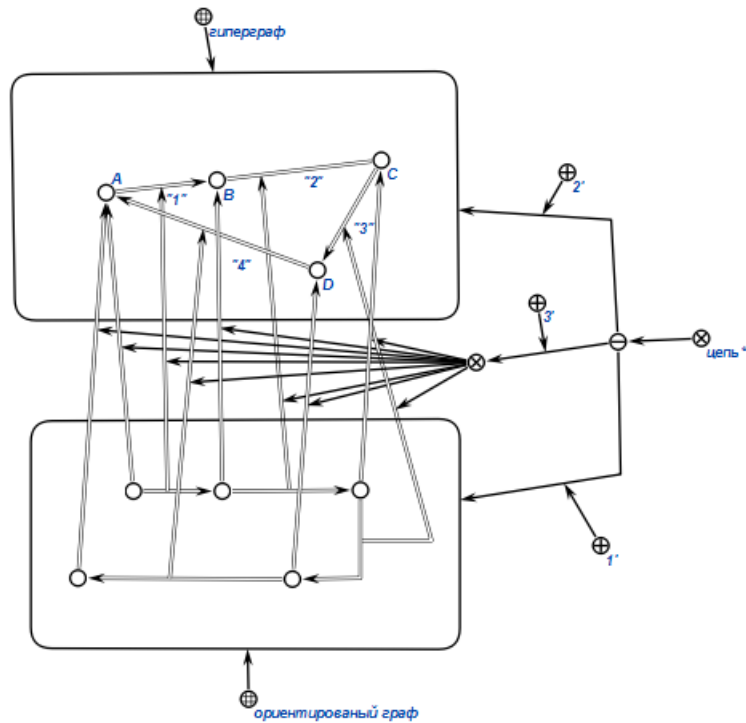


Figure 11: Цепь

12. **Простая цепь, путь** (относительное понятие, бинарное ориентированное отношение) – это цепь, в которой все вершины различны. В примере ниже показан путь A, "1", B, "2", C, "3", D в гиперграфе.

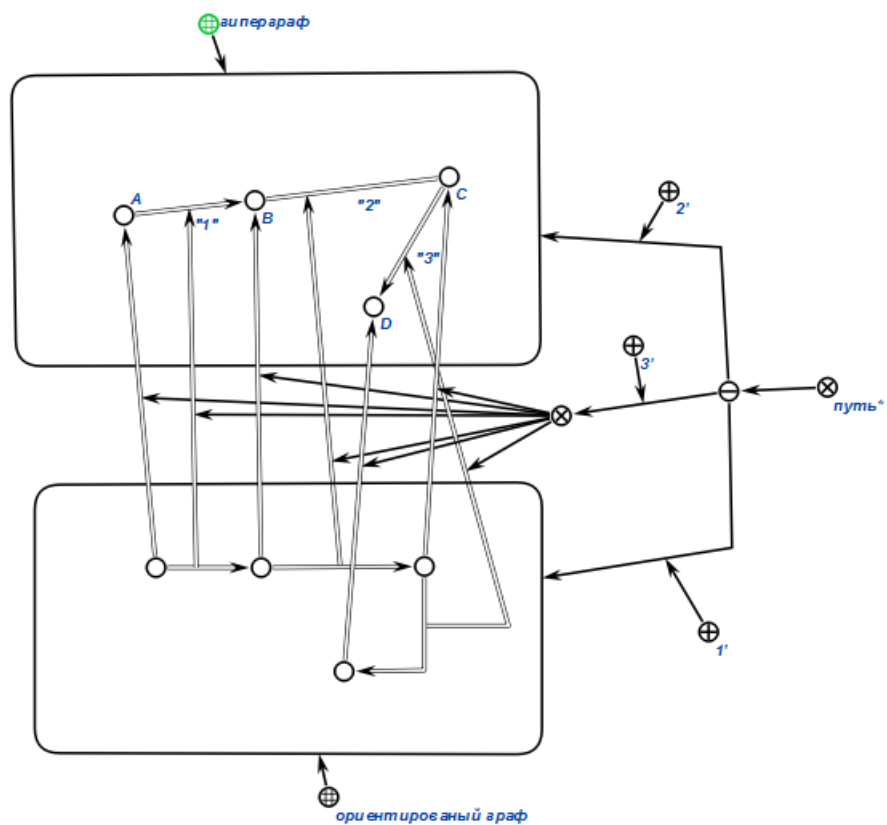


Figure 12: Путь

13. *Связный граф, компонента связности* : между любой парой вершин этого графа существует как минимум один путь.

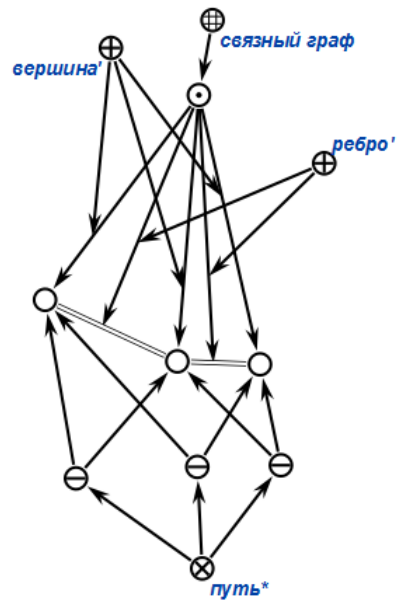


Figure 13: Связный граф

### 3 Тестовые примеры

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

#### 3.1 Тест 1

**Вход:**

Необходимо найти количество компонент связности неориентированного графа.

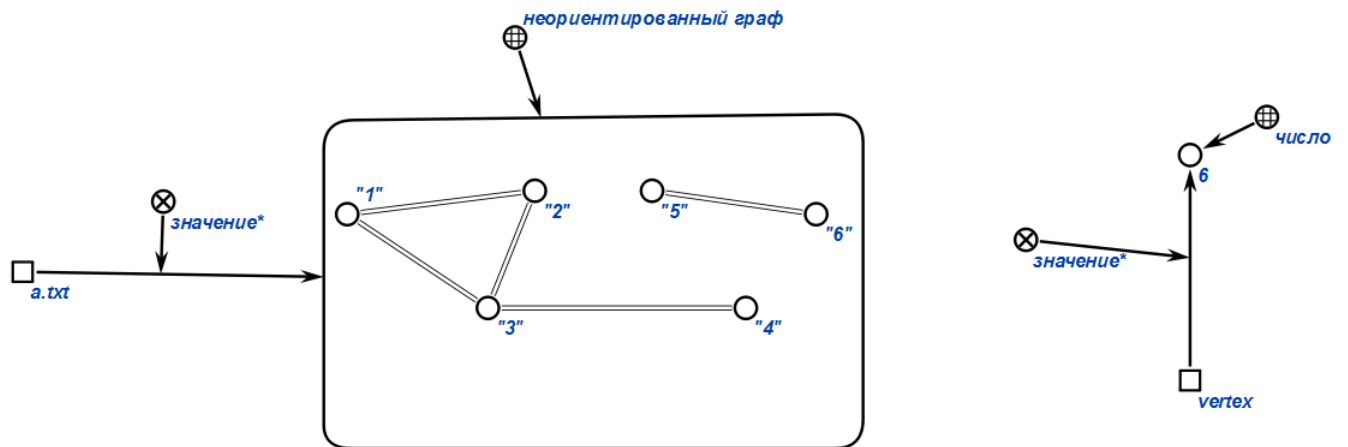


Figure 14: Вход теста 1

**Выход:** Будет выведено число два, так как у данного графа две компоненты связности.

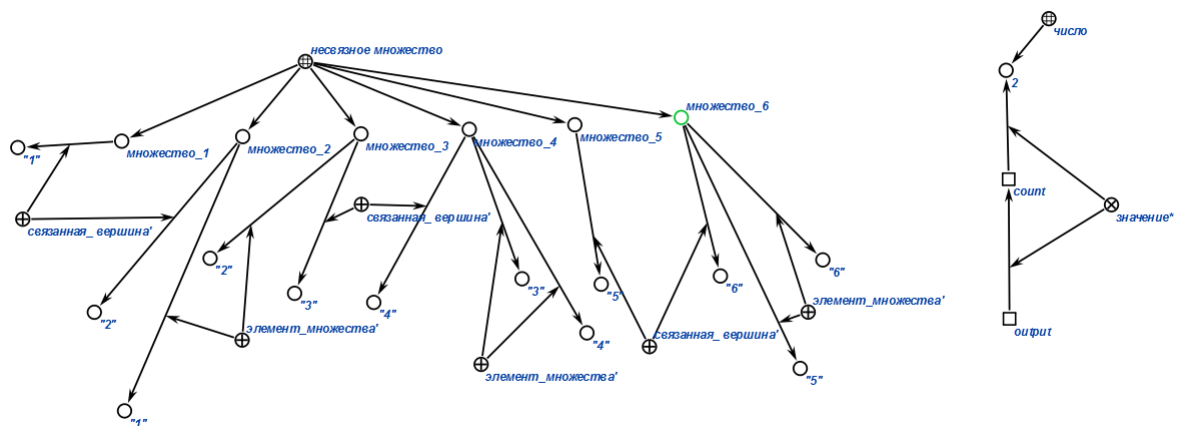


Figure 15: Выход теста 1

## 3.2 Тест 2

**Вход:** Необходимо найти количество компонент связности неориентированного графа.

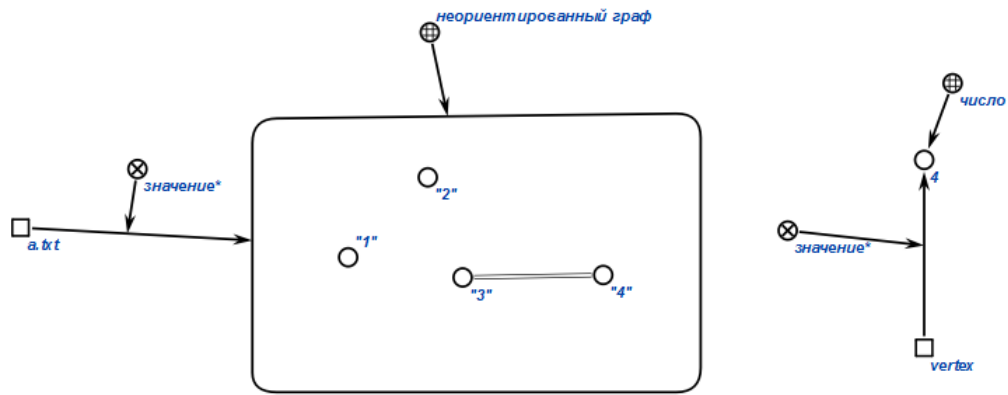


Figure 16: Вход теста 2

**Выход:** Будет выведено число три, так как у данного графа три компоненты связности.

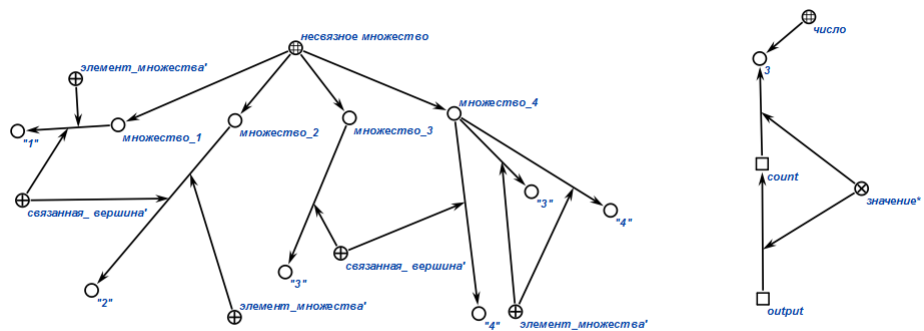


Figure 17: Выход теста 2



### 3.3 Тест 3

**Вход:** Необходимо найти количество компонент связности неориентированного графа. **Выход:** Будет

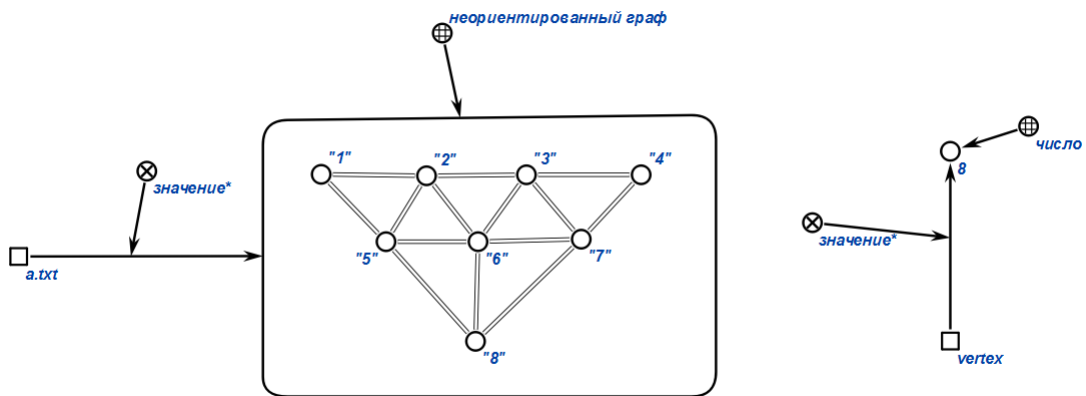


Figure 18: Вход теста 3

выведено число один, так как у данного графа одна компонента связности.

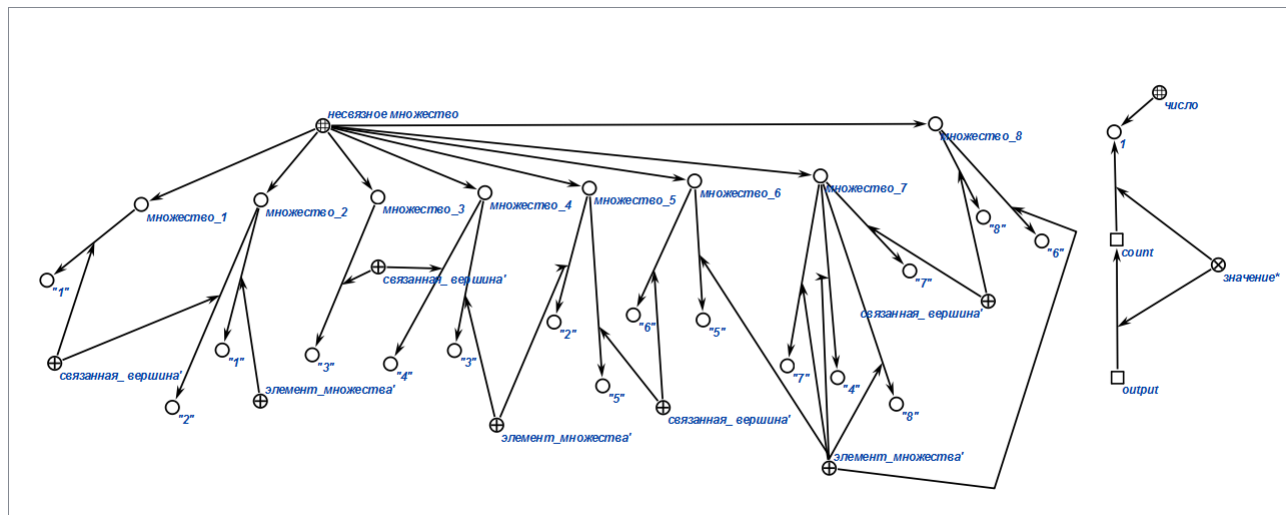


Figure 19: Выход теста 3

### 3.4 Тест 4

**Вход:** Необходимо найти количество компонент связности неориентированного графа. **Выход:** Будет

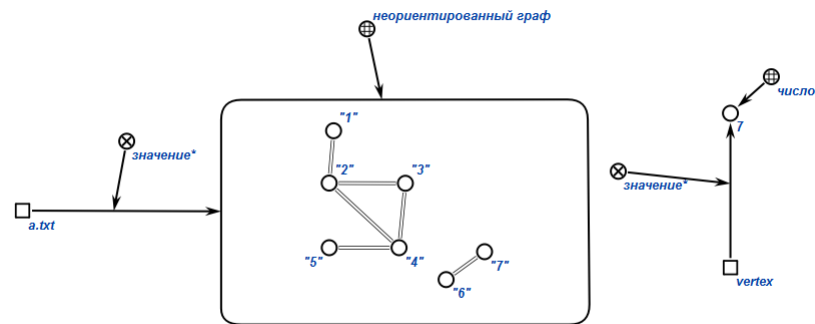


Figure 20: Вход теста 4

выведено число два, так как у данного графа две компоненты связности.

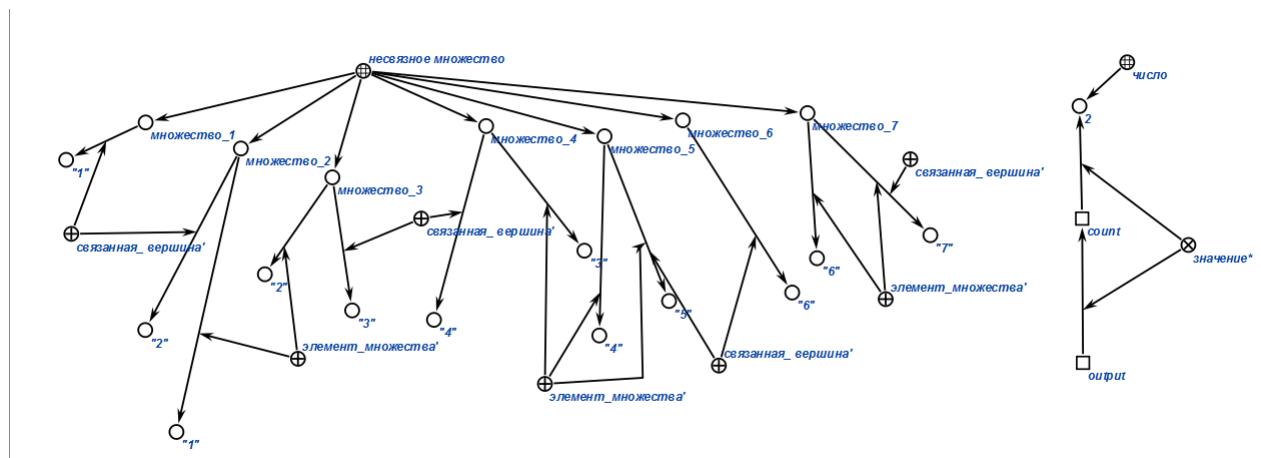


Figure 21: Выход теста 4

## 4 Пример работы алгоритма в семантической памяти

### 1. Задание входного графа

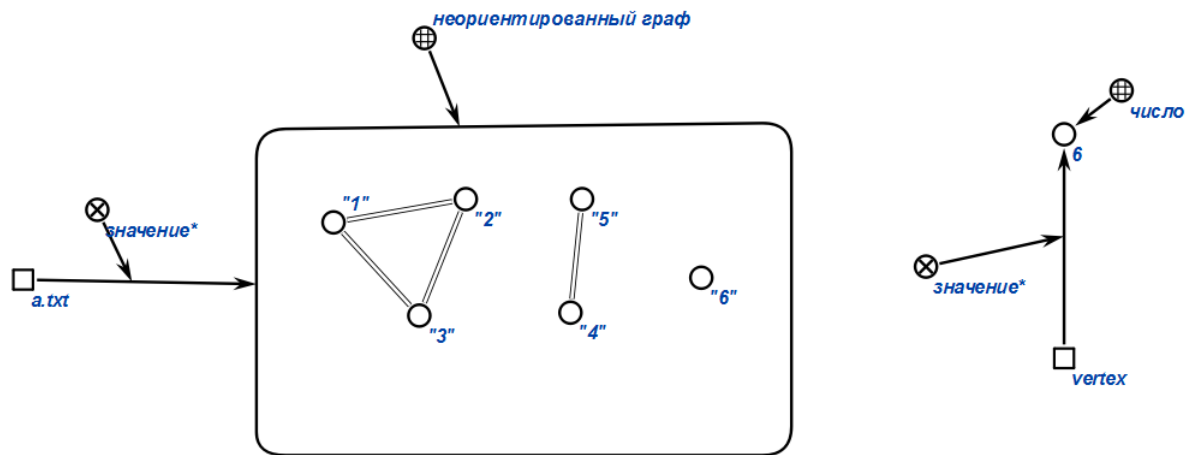


Figure 22: Шаг 1

a.txt получит в качестве значения sc-контур неориентированного графа. Переменная vertex получит в качестве значения количество вершин графа.

## 2. Создание несвязных множеств

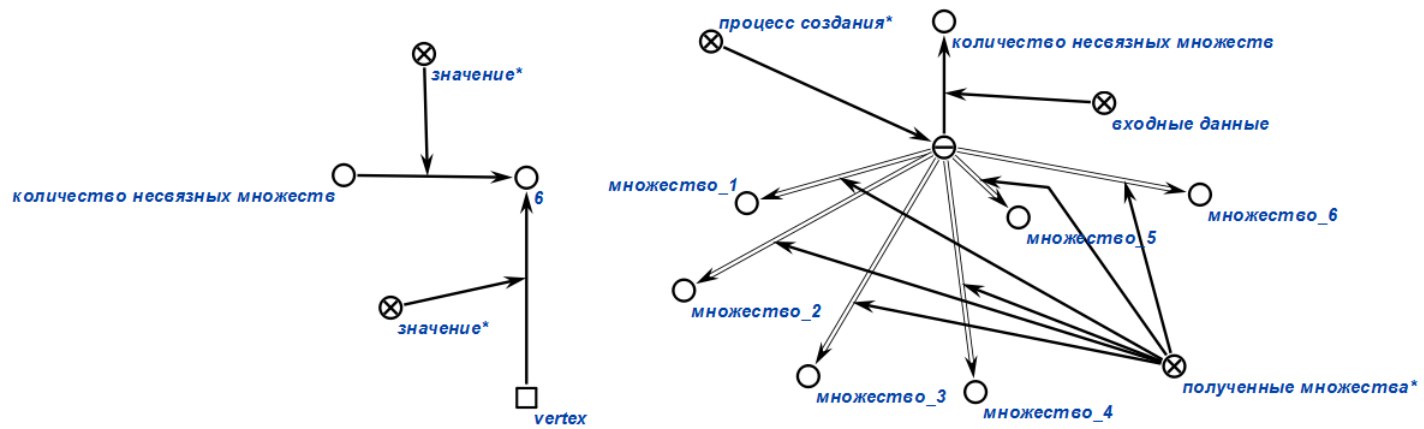


Figure 23: Шаг 2

Sc-узел 'количество несвязных множеств' получит в качестве значения значение переменной vertex. Далее создадим столько несвязных множеств, сколько хранится в качестве значения Sc-узла 'количество несвязных множеств'.

### 3. Связывание вершин с множествами

Свяжем каждую вершину с соответствующим множеством отношением 'связная вершина'. Далее сделаем каждую вершину элементом соответствующего множества. Получим следующие множества:

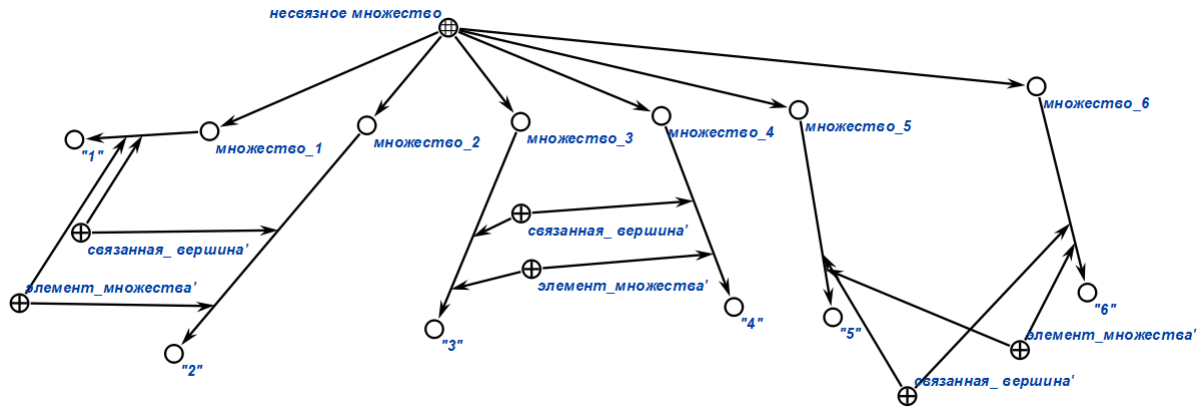


Figure 24: Шаг 3

#### 4. Обработка каждой пары вершин, связанных ребром

4.1.1 Обработываем пару вершин "1" и "2" Для каждой вершины найдем ее представителя (вершину, с которой связано множество, элементом которого является наша обрабатываемая вершина) и запишем их значения в переменные 'представитель1' и 'представитель2' соответственно.

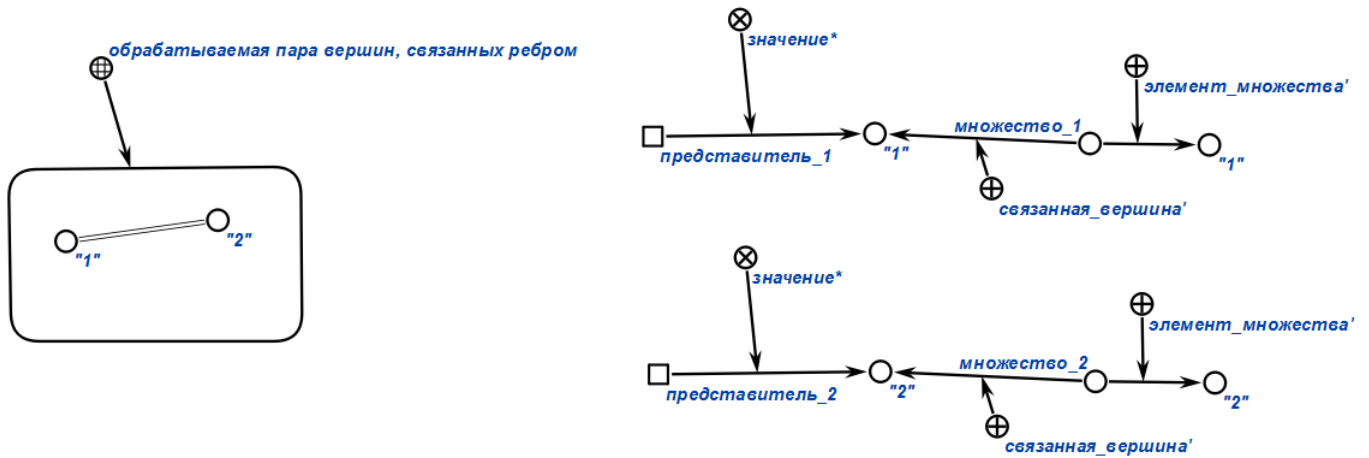


Figure 25: Шаг 4.1.1

4.1.2 Обрабатываем пару "i" и "j". Далее создаем переменные i и j и присваиваем им значения переменных 'представитель1' и 'представитель2' соответственно.

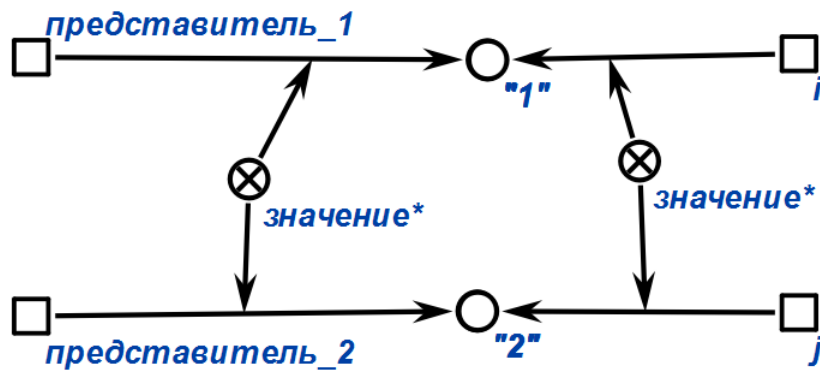


Figure 26: Шаг 4.1.2

Сравниваем значения  $i$  и  $j$  и при значении  $\text{flag} = \text{false}$  присваиваем переменной  $i$  значение переменной  $j$ .

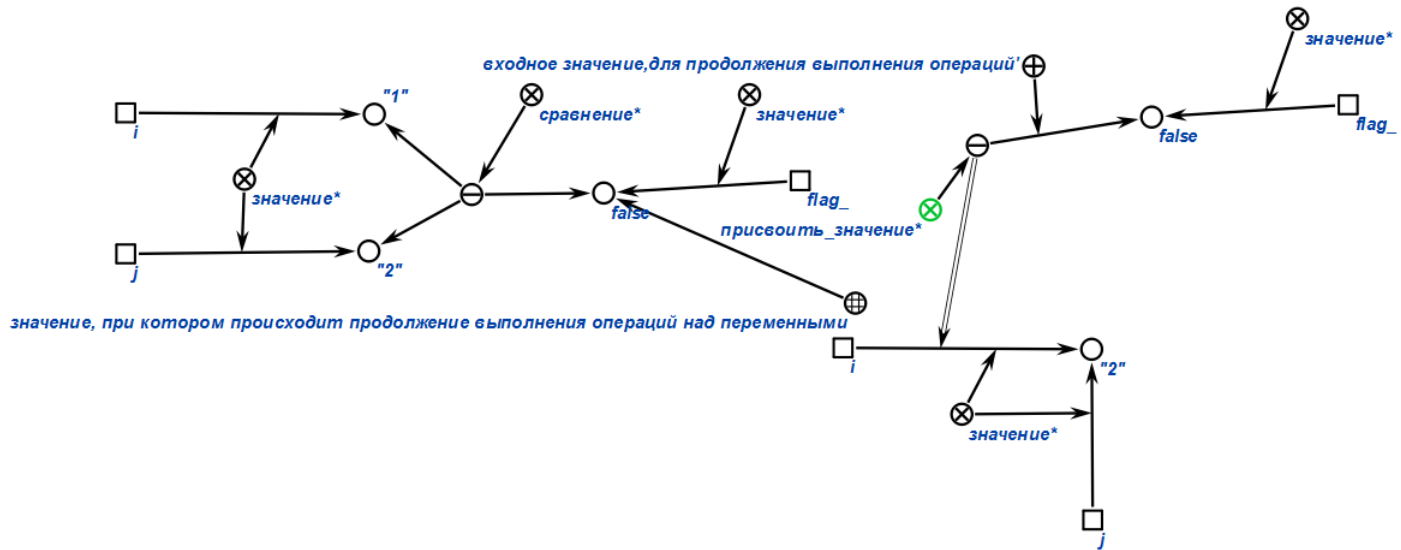


Figure 27: Шаг 4.1.2



4.1.3 Перенос элемента из одного множества в другое Перенесем вершину, которая находится в переменной представитель1 в множество, которое связано с вершиной, хранящейся в переменной i.

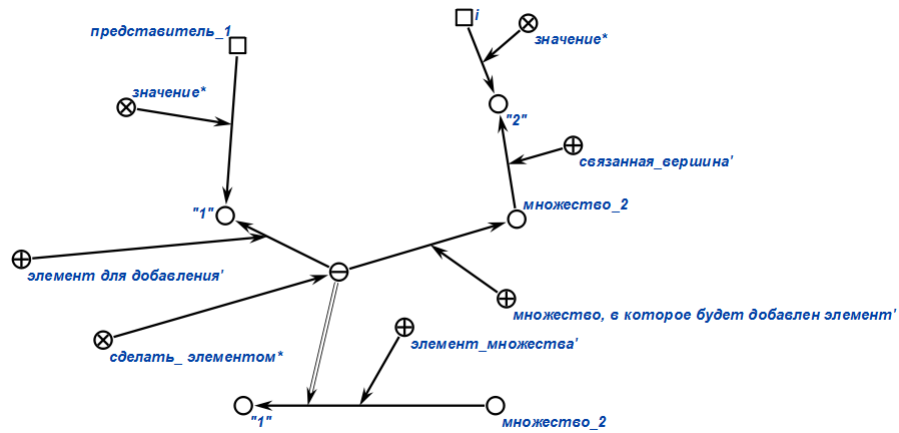


Figure 28: Шаг 4.1.3

4.2.1 Обработываем пару вершин "1" и "3". Для каждой вершины найдем ее представителя (вершину, с которой связано множество, элементом которого является наша обрабатываемая вершина) и запишем их значения в переменные 'представитель1' и 'представитель2' соответственно.

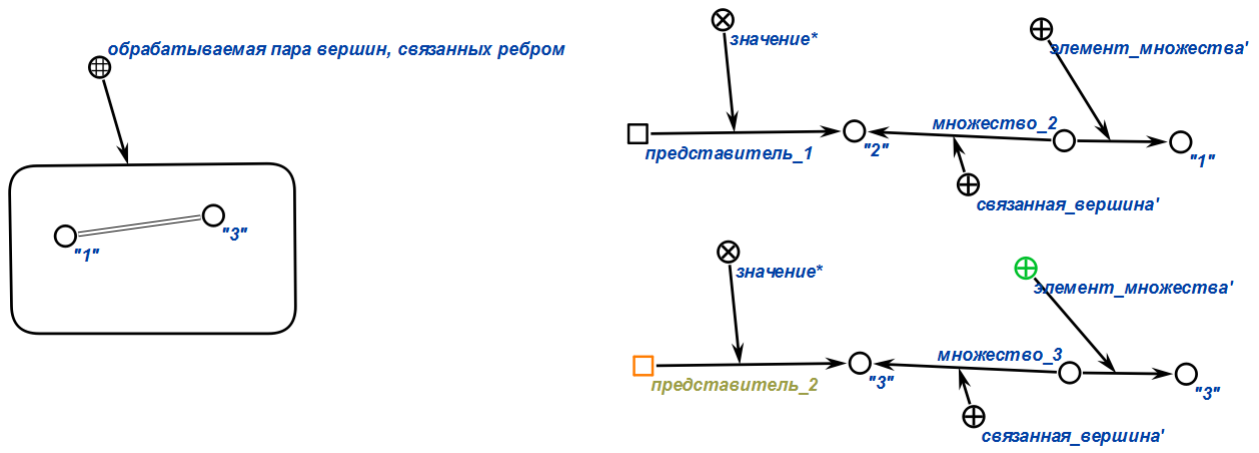


Figure 29: Шаг 4.2.1

4.2.2 Обработываем пару "i" и "j". Далее создаем переменные i и j и присваиваем им значения переменных 'представитель1' и 'представитель2' соответственно.

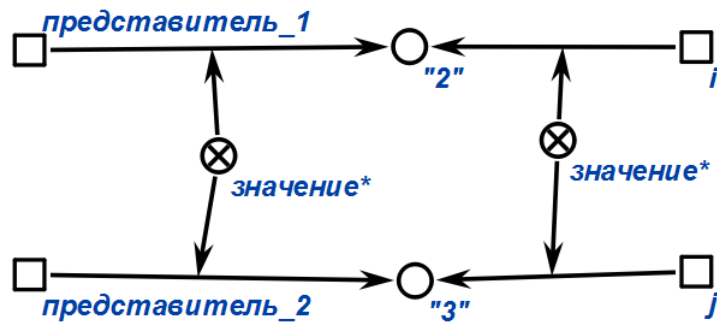


Figure 30: Шаг 4.2.2

Сравниваем значения  $i$  и  $j$  и при значении  $\text{flag} = \text{false}$  присваиваем переменной  $i$  значение переменной  $j$ .

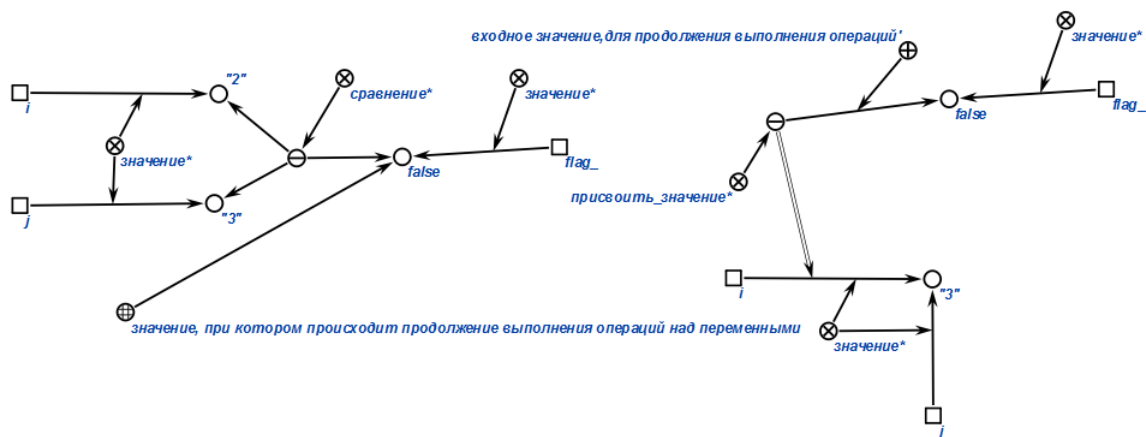


Figure 31: Шаг 4.2.2

4.2.3 Перенос элемента из одного множества в другое Перенесем вершину, которая находится в переменной представитель1 в множество, которое связано с вершиной, хранящейся в переменной i.

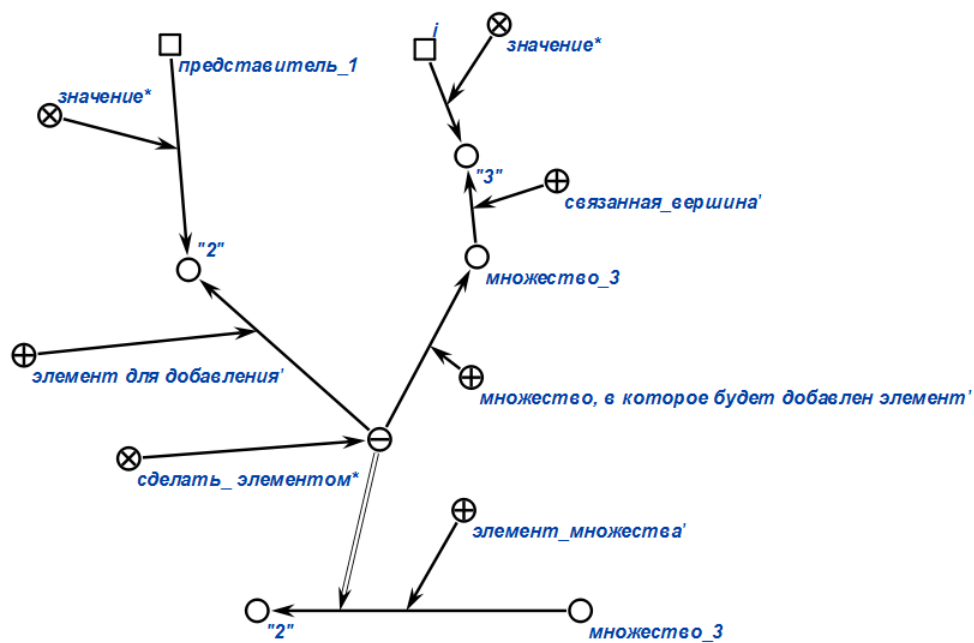


Figure 32: Шаг 4.2.3

4.3.1 Обработываем пару вершин "2" и "3". Для каждой вершины найдем ее представителя (вершину, с которой связано множество, элементом которого является наша обрабатываемая вершина) и запишем их значения в переменные 'представитель1' и 'представитель2' соответственно.

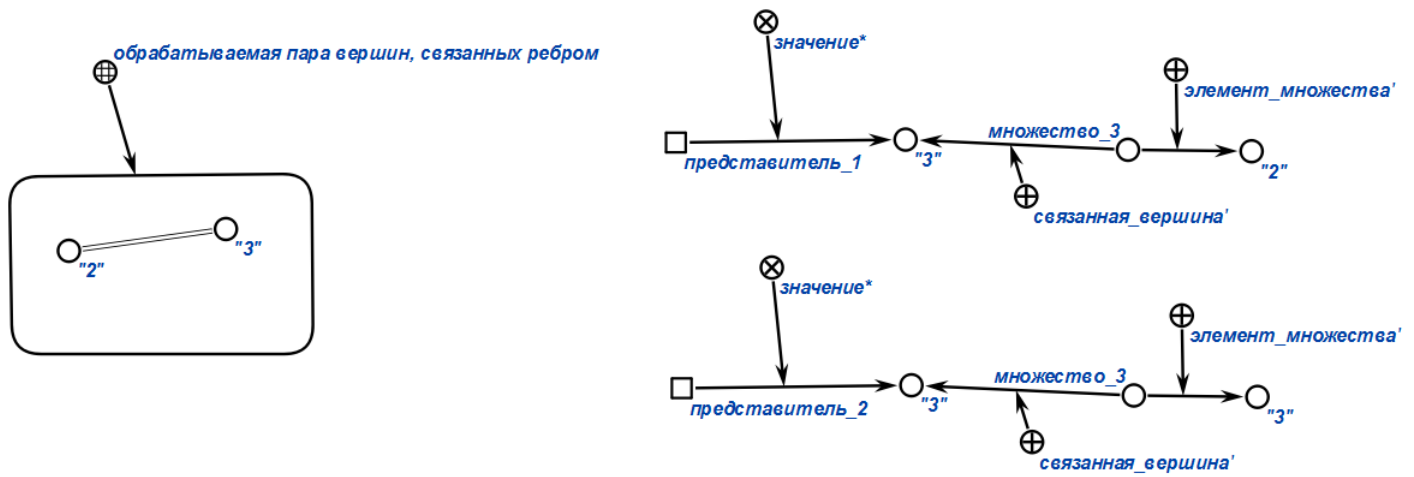


Figure 33: Шаг 4.3.1

4.3.2 Обработываем пару "i"и "j" Далее создаем переменные i и j и присваеваем им значения переменных 'представитель1'и 'представитель2' соответственно.

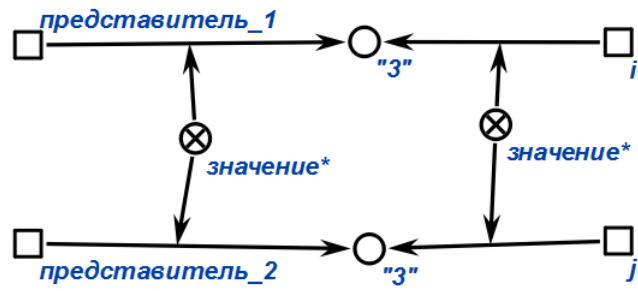


Figure 34: Шаг 4.3.2

Сравниваем значения  $i$  и  $j$  и при значении  $flag = true$  переходим к обработке следующей пары вершин. .

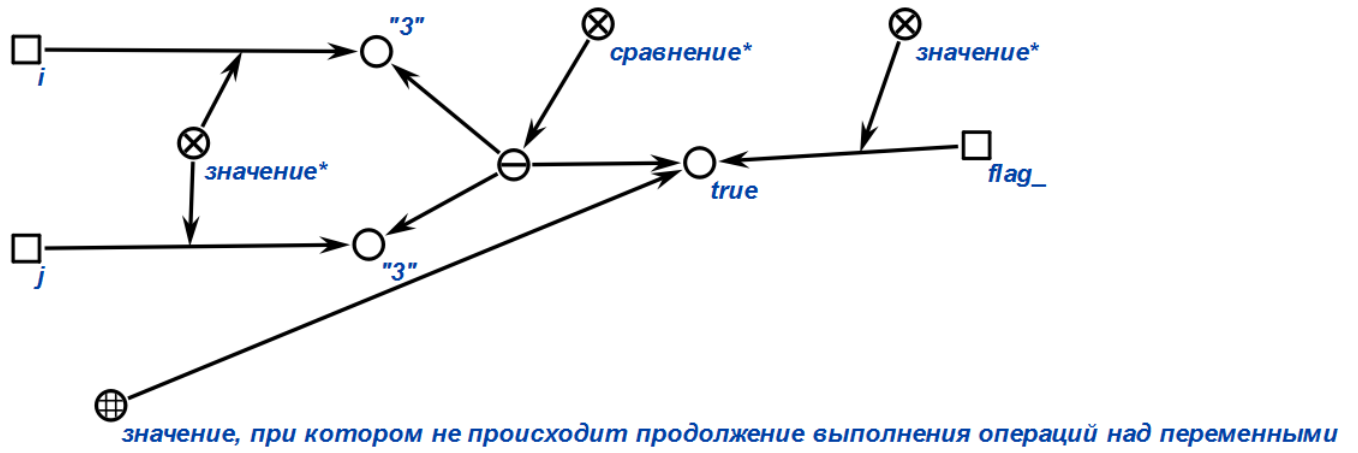


Figure 35: Шаг 4.3.2



4.4.1 Обработываем пару вершин "4" и "5". Для каждой вершины найдем ее представителя (вершину, с которой связано множество, элементом которого является наша обрабатываемая вершина) и запишем их значения в переменные 'представитель1' и 'представитель2' соответственно.

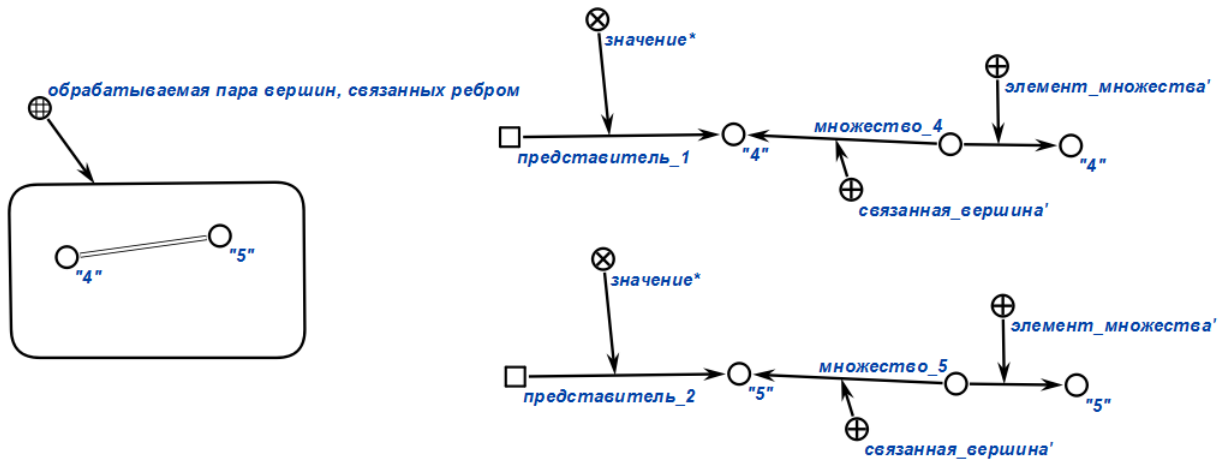


Figure 36: Шаг 4.4.1

4.4.2 Обработываем пару "i" и "j". Далее создаем переменные i и j и присваеваем им значения переменных 'представитель1' и 'представитель2' соответственно.

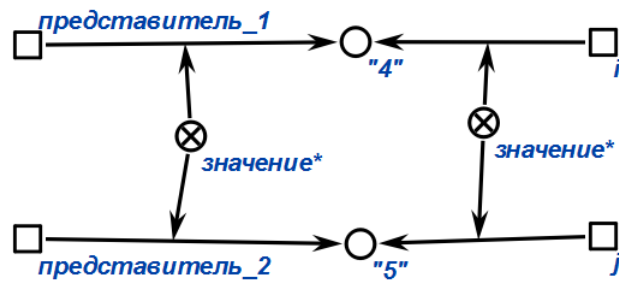


Figure 37: Шаг 4.4.2

Сравниваем значения `i` и `j` и при назначении `flag = false` присваиваем переменной `i` значение переменной `j`.

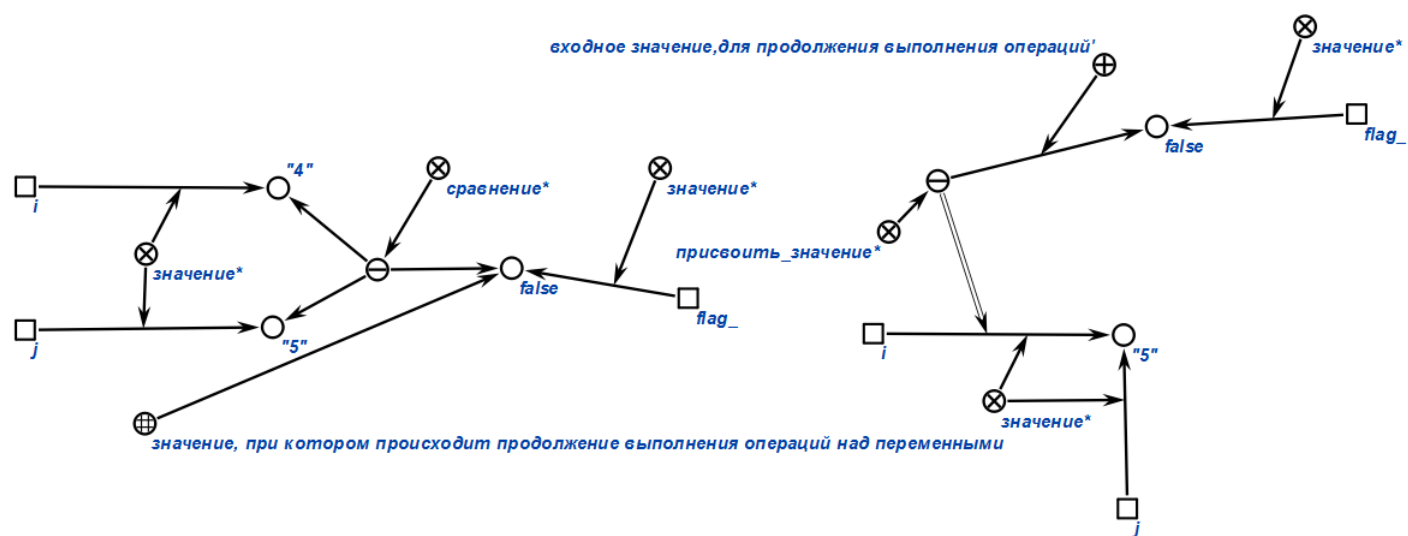


Figure 38: Mar 4.4.2

4.4.3 Перенос элемента из одного множества в другое Перенесем вершину, которая находится в переменной представитель1 в множество, которое связано с вершиной, хранящейся в переменной i.

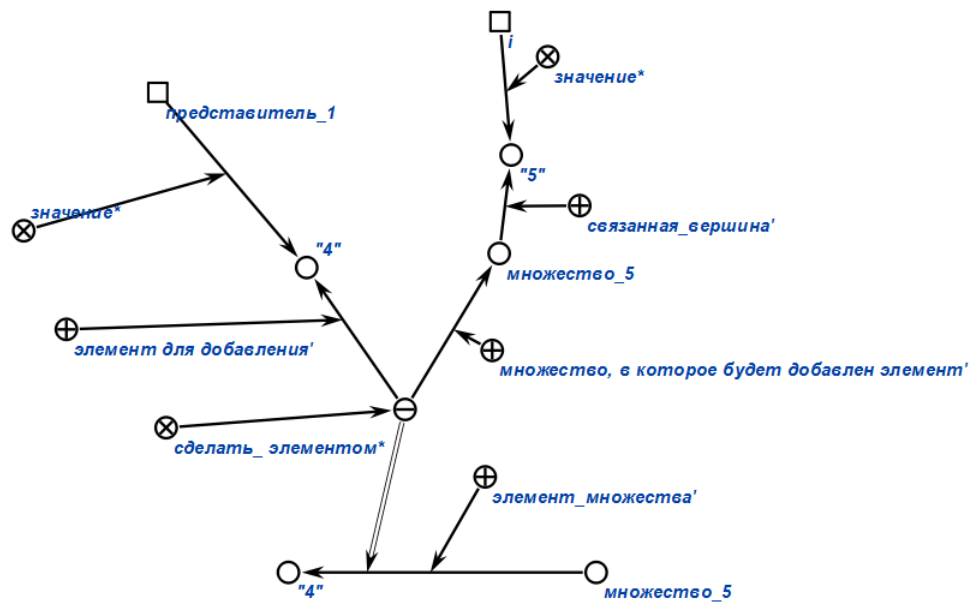


Figure 39: Шаг 4.4.3

## 5. Определение количества компонент связности

После обработки всех пар вершин, связанных между собой ребром, получим следующую картину несвязных множеств.

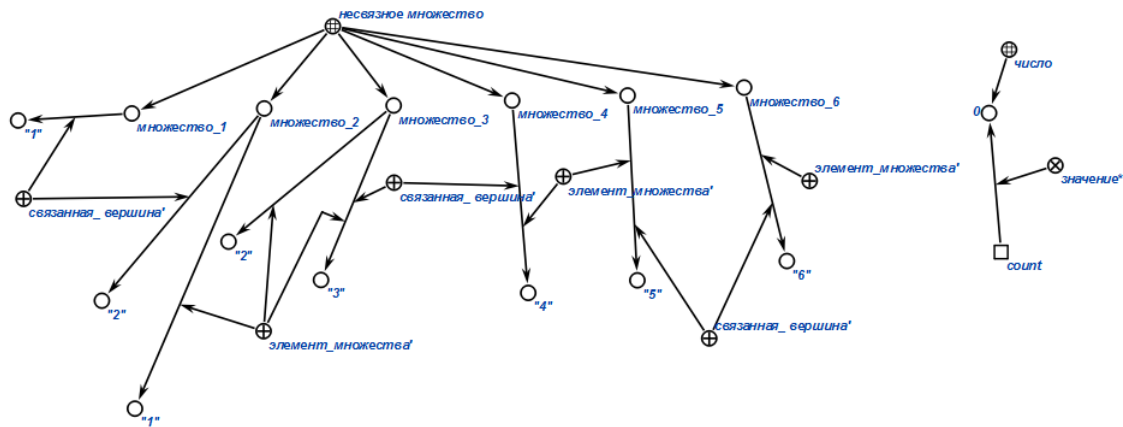


Figure 40: Шаг 5

Пройдемся по каждой вершине и определим, равна ли она своему представителю. В случае равенства присваиваем переменной flag1 значение true, тем самым создаем необходимое условие для увеличения переменной count. Далее будет приведен пример прохода по всем вершинам и увеличение переменной count.

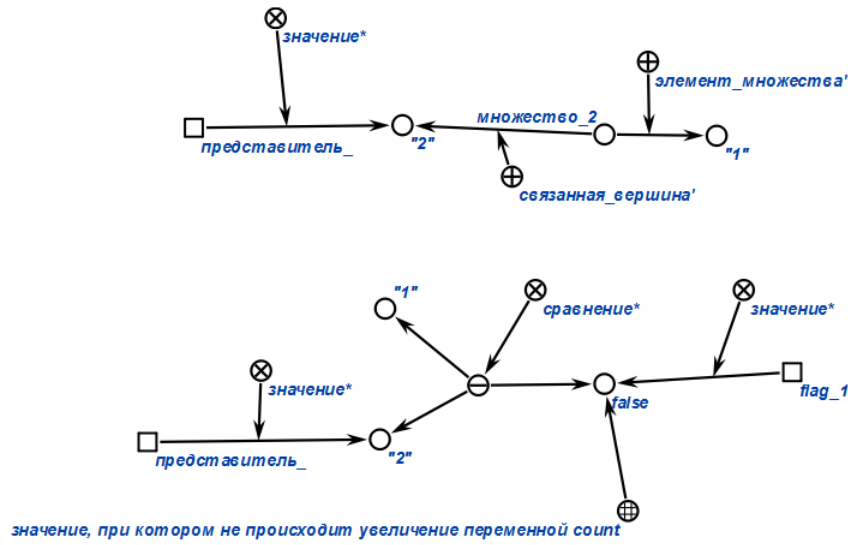


Figure 41: Шаг 5.1

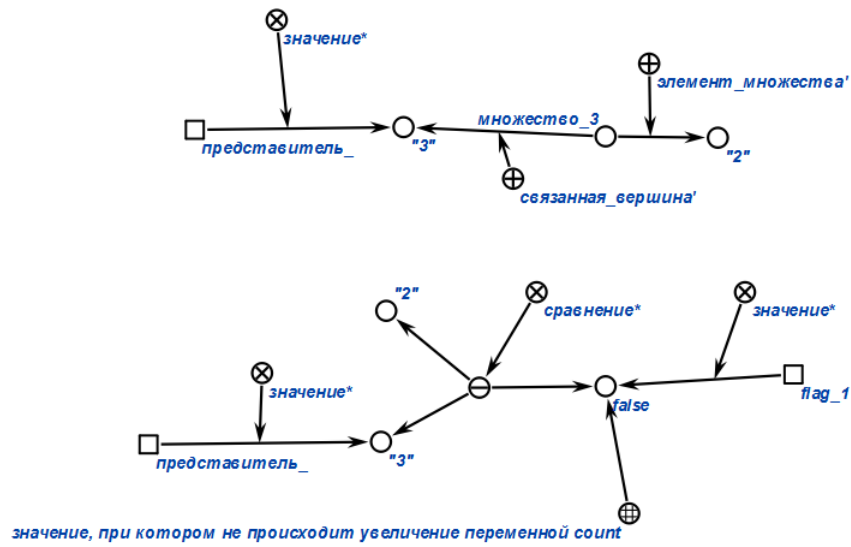


Figure 42: Шаг 5.2

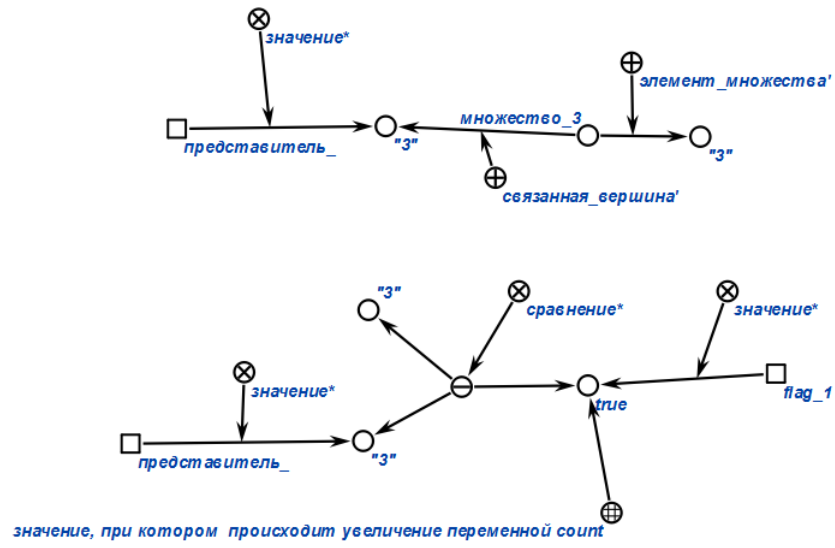
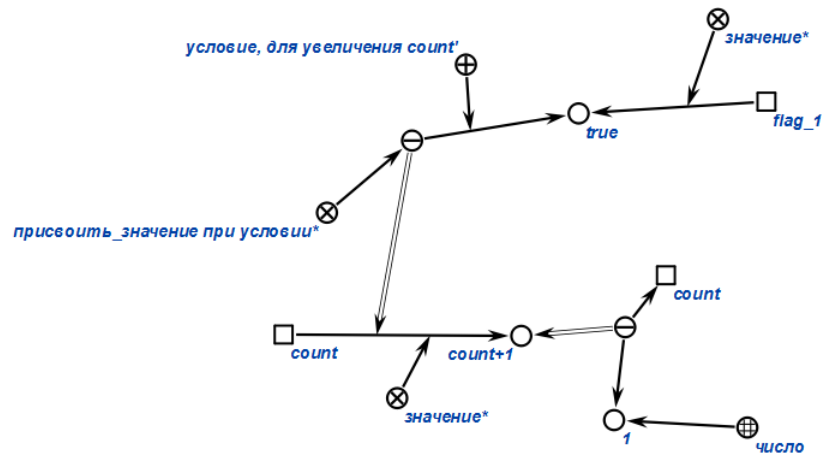


Figure 43: Шаг 5.3





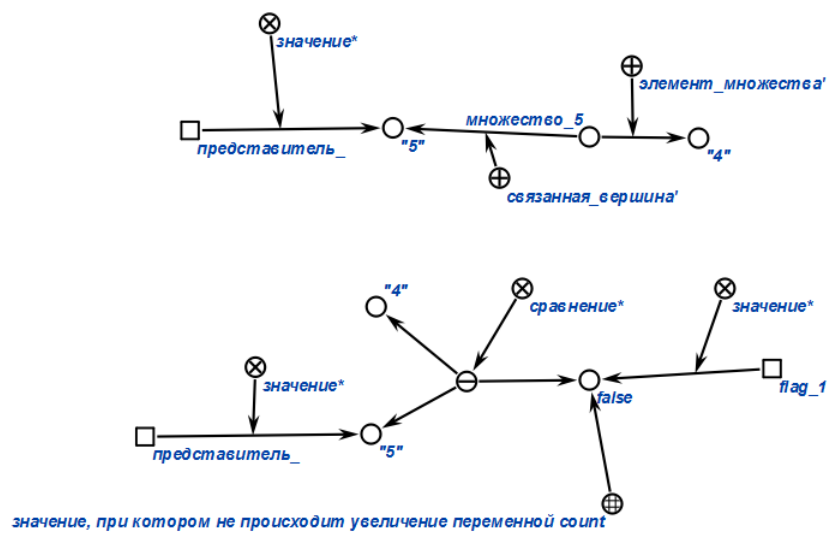


Figure 45: Шар 5.5

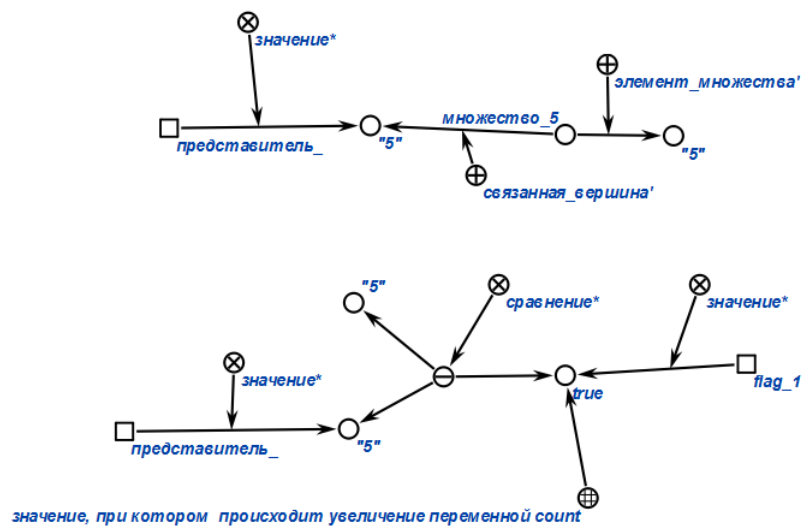


Figure 46: Шаг 5.6

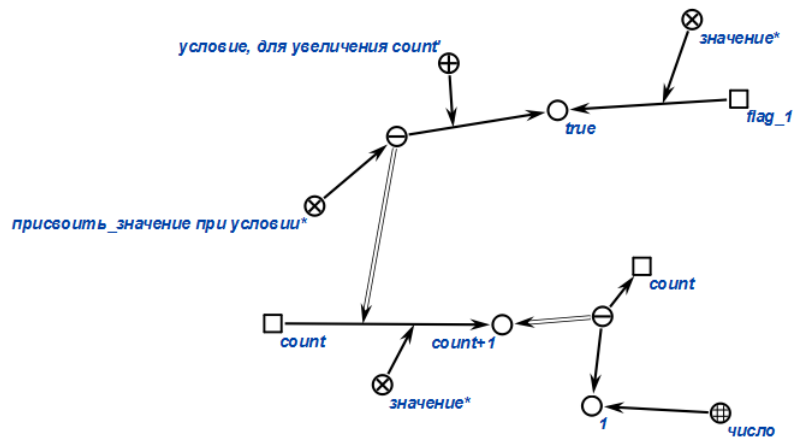


Figure 47: Шаг 5.7

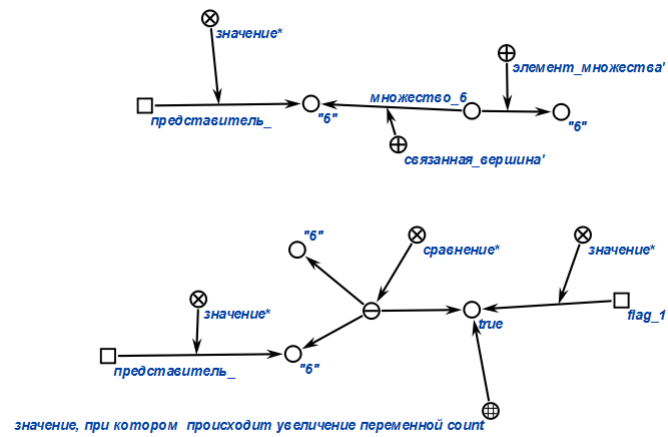


Figure 48: Шаг 5.8

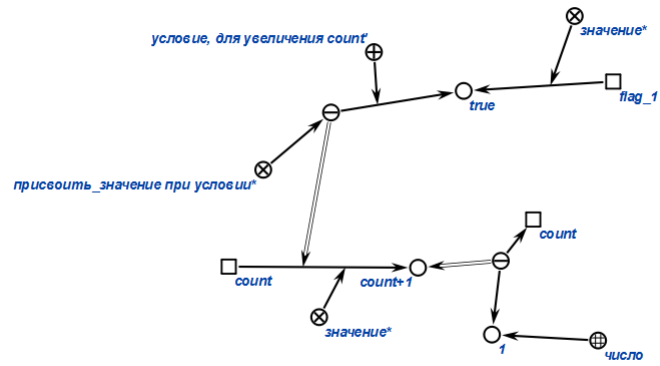


Figure 49: Шаг 5.9

6. Вывод ответа: Выводим количество компонент связности, находящихся в переменной `output`, присвоив ей перед этим значение переменной `count`.

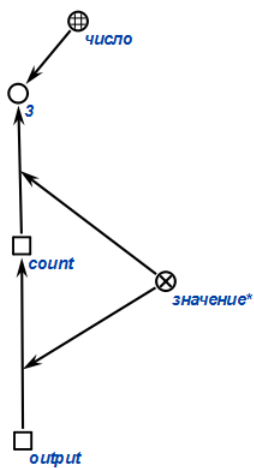


Figure 50: Шаг 6

## 5 Заключение

Получила навыки формализации и обработки информации с использованием семантических сетей на примере работы алгоритма конкретной задачи.



## 6 Используемая литература

### References

- [1] Емеличев В. А., Мельников О. И., Сарванов В. И., Тышкевич Р. И. Лекции по теории графов. М.: Наука, 1990. 384с. (Изд.2, испр. М.: УРСС, 2009. 392 с.)
- [2] Оре О. Теория графов. – 2-е изд.. – М.: Наука, 1980. – С. 336.
- [3] Лазуркин Д.А. "Руководство к выполнению расчетной работы по курсам ОИИ и ППВИС" - 19.02.2013 г. - 13с.