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 «Проектирование программного обеспечения в интеллектуальных системах»
на тему
«Задача поиска подграфов в неориентированном графе, изоморфных графу-образцу»

Выполнил Липский Р. В.
студент группы
121701

Проверил Бутрин. С. В.

Минск 2022

Цель: получить навыки формализации и обработки информации с использованием семантических сетей
Задача: поиск подграфов в неориентированном графе, изоморфных графу-образцу.

Список понятий

1. Граф (абсолютное понятие) - совокупность непустого множества вершин и наборов пар вершин (связей между вершинами).

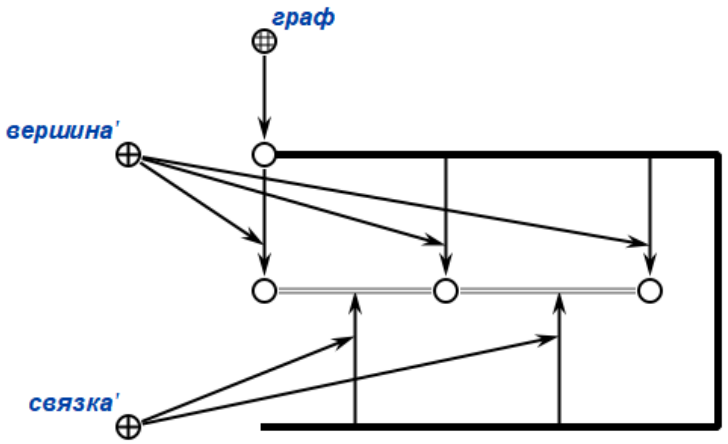

Рисунок 1.1. - Граф

2. Неориентированный граф (абсолютное понятие) – граф, в котором все связки - ребра.

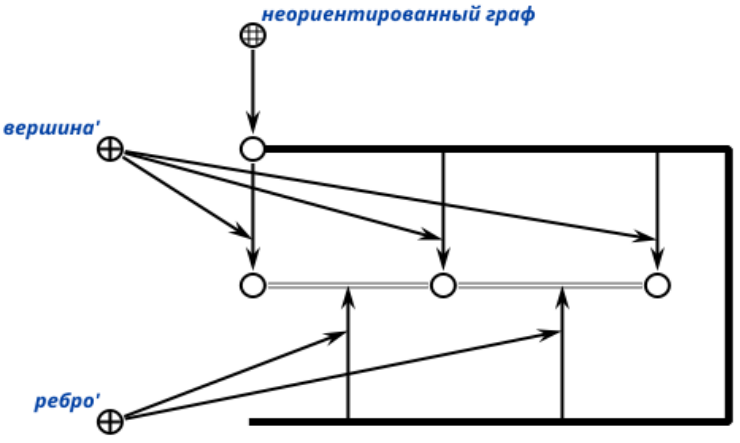

Рисунок 1.2. - Неориентированный граф

3. Подграф (абсолютное понятие) — граф, образованный из подмножества вершин графа вместе со всеми рёбрами, соединяющими пары вершин из этого подмножества.

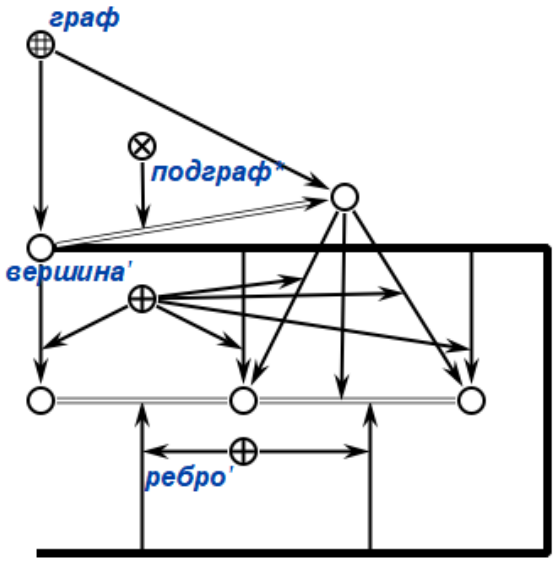

Рисунок 1.3. - Подграф

4. Изоморфизм графов G = <V_G, E_G> и H = <V_H, E_H> (абсолютное понятие) — биекция между множествами вершин графов $f: V_G \to V_H$, такая, что любые две вершины $u и v$ графа G смежны тогда и только тогда, когда вершины $f(u) и f(v)$ смежны в графе H.

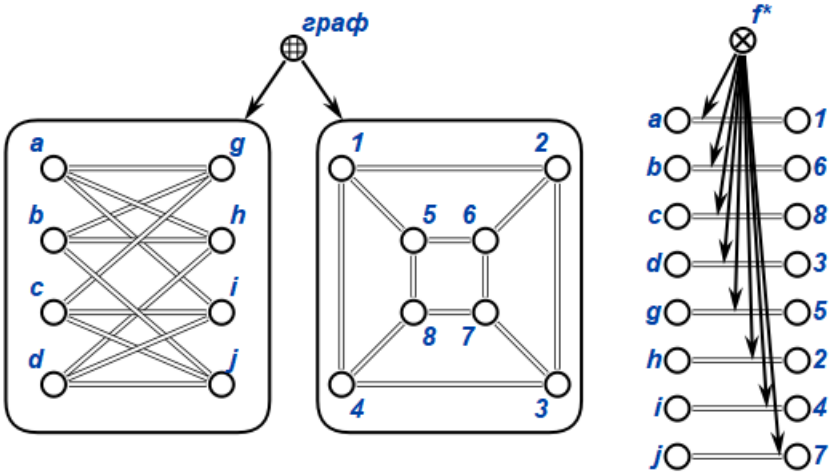

Рисунок 1.4. – Изоморфизм графов

Алгоритм

1. Обозначим изначальный граф как $G_1 = \langle V_1, E_1 \rangle$, а граф-образец как $G_2 = \langle V_2, E_2 \rangle$
2. Создадим некоторую биекцию f между V_1 и V_2, все биекции должны проверяться только один раз.
3. Проверим, правда ли, что для $\forall \langle f(x), f(y) \rangle \in E_2$, где $x и y \in V_1$, $\exists \langle x, y \rangle \in E_1$. (обозначим это соответствие как x*)
4. Если данное условие выполняется, созданная биекция — один ответов.
5. Если ещё существует непроверенные биекции, перейдём к пункту 2.
6. Если все биекции были проверены, алгоритм завершается.

Использованные технологии и библиотеки

- Java 17 (https://ru.wikipedia.org/wiki/Java)
- Java WebSocket (https://github.com/TooTallNate/Java-WebSocket)
- SC WebSocket API (http://ostis-dev.github.io/sc-machine/http/websocket/)
- Google Guava (https://github.com/google/guava)
- JESC (https://github.com/ungaf/jesc)

Реализация

Регистрация агента:

```
public static void main(String[] args) {
    var client = new ScClient("localhost", 8090);
    var agentRegistry = new ScAgentRegistry(client);
    var context = new ScContextCommon(new ScApi(client));

    var questionNode = context.findBySystemIdentifier("v3_qfis").get();
    var eventId = context.createEvent(ScEventType.ADD_OUTGOING_EDGE, questionNode);

    agentRegistry.registerAgent(new IsomorphicSubgraphSearchAgent(Set.of(eventId), client));
    while (true) { }
}
```

Метод агента, реагирующий на событие

```
@Override
public void onTrigger(ScEvent event) {
    var problemNode = event.getPayload().get(2);
    var targetEdges = getEdgeSet(relTarget);
    var patternEdges = getEdgeSet(relPattern);

    var pattern = getGraphParentNode(problemNode, patternEdges);
    var target = getGraphParentNode(problemNode, targetEdges);

    var patternNodeSet = getGraphNodeSet(pattern);
    var targetNodeSet = getGraphNodeSet(target);

    var patternNodeList = new ArrayList<>(patternNodeSet);

    var isomorphisms = new HashSet<Map<Long, Long>>();

    for (List<Long> lst : Collections2.permutations(targetNodeSet)) {

        var patternToTarget = new HashMap<Long, Long>();
        var targetToPattern = new HashMap<Long, Long>();

        for (int i = 0; i < patternNodeList.size(); i++) {
            patternToTarget.put(patternNodeList.get(i), lst.get(i));
            targetToPattern.put(lst.get(i), patternNodeList.get(i));
        }

        var fits = true;
        checking: for (Long p1 : patternNodeList) {
            for (Long p2 : patternNodeList) {
                var edgeExistsInPattern = areAdjacent(p1, p2);
                var edgeExistsInTarget = areAdjacent(patternToTarget.get(p1), patternToTarget.get(p2));
                if (edgeExistsInPattern != edgeExistsInTarget) {
                    fits = false;
                    break checking;
                }
            }
        }

        if (fits) {
            isomorphisms.add(patternToTarget);
        }
    }

    System.out.printf("Target node set: %s%n", targetNodeSet);
    System.out.printf("Pattern node set: %s%n", patternNodeSet);
    System.out.printf("Isomorphisms: %s%n", isomorphisms);
}
```

Метод areAdjacent() - проверяет наличие ребра между узлами

```
boolean areAdjacent(Long nodeAddr1, Long nodeAddr2) {
    for (Triplet<Long> t : context.iterator3(
            ScReference.addr(nodeAddr1),
            ScReference.type(ScType.EDGE_U_COMMON_VAR),
            ScReference.addr(nodeAddr2)
    )) {
        return true;
    }

    for (Triplet<Long> t : context.iterator3(
            ScReference.addr(nodeAddr2),
            ScReference.type(ScType.EDGE_U_COMMON_VAR),
            ScReference.addr(nodeAddr1)
    )) {
        return true;
    }
    return false;
}
```

Метод getGraphNodeSet() - получить все элементы-узлы множества

```
Set<Long> getGraphNodeSet(Long graphParentNode) {
    var nodeSet = new HashSet<Long>();
    for (var t : context.iterator3(
            ScReference.addr(graphParentNode),
            ScReference.type(ScType.EDGE_ACCESS_VAR_POS_PERM),
            ScReference.type(ScType.NODE_VAR)
    )) {
        nodeSet.add(t.getThird());
    }
    return nodeSet;
}
```

Метод getGraphParentNode() - получить элемент, выполняющий определенную роль во множестве

```
Long getGraphParentNode(Long problemNode, Collection<Long> edges) {
    Long patternGraph;
    for (var t : context.iterator3(
            ScReference.addr(problemNode),
            ScReference.type(ScType.EDGE_ACCESS_VAR_POS_PERM),
            ScReference.type(ScType.NODE_VAR)
    )) {
        if (edges.contains((t.getSecond()))) {
            return t.getThird();
        }
    }

    throw new RuntimeException("Graph parent node not found.");
}
```

Метод getEdgeSet() - получить элементы-ребра множества

```
Set<Long> getEdgeSet(Long parentNode) {
    var edges = new HashSet<Long>();
    context.iterator3(
            ScReference.addr(parentNode),
            ScReference.type(ScType.EDGE_ACCESS_VAR_POS_PERM),
            ScReference.type(ScType.EDGE_ACCESS_VAR_POS_PERM)
    ).forEach(t -> edges.add(t.getThird()));
    return edges;
}
```

Конструктор агента

```
protected IsomorphicSubgraphSearchAgent(Set<Long> triggerEventIds, ScClient client) {
    super(triggerEventIds, client);

    questionNode = context.findBySystemIdentifier("qfis").get();
    relTarget = context.findBySystemIdentifier("rrel_qfis_target").get();
    relPattern = context.findBySystemIdentifier("rrel_qfis_pattern").get();
}
```

Тестовые примеры

Во всех тестах графы будут приведены в сокращенной форме со скрытыми ролями элементов графа и будет требоваться найти все подграфы, изоморфные графу образцу, в неориентированном графе.

Тест 1

Вход:

В неориентированном графе необходимо найти все подграфы, изомофорные графу-образцу.

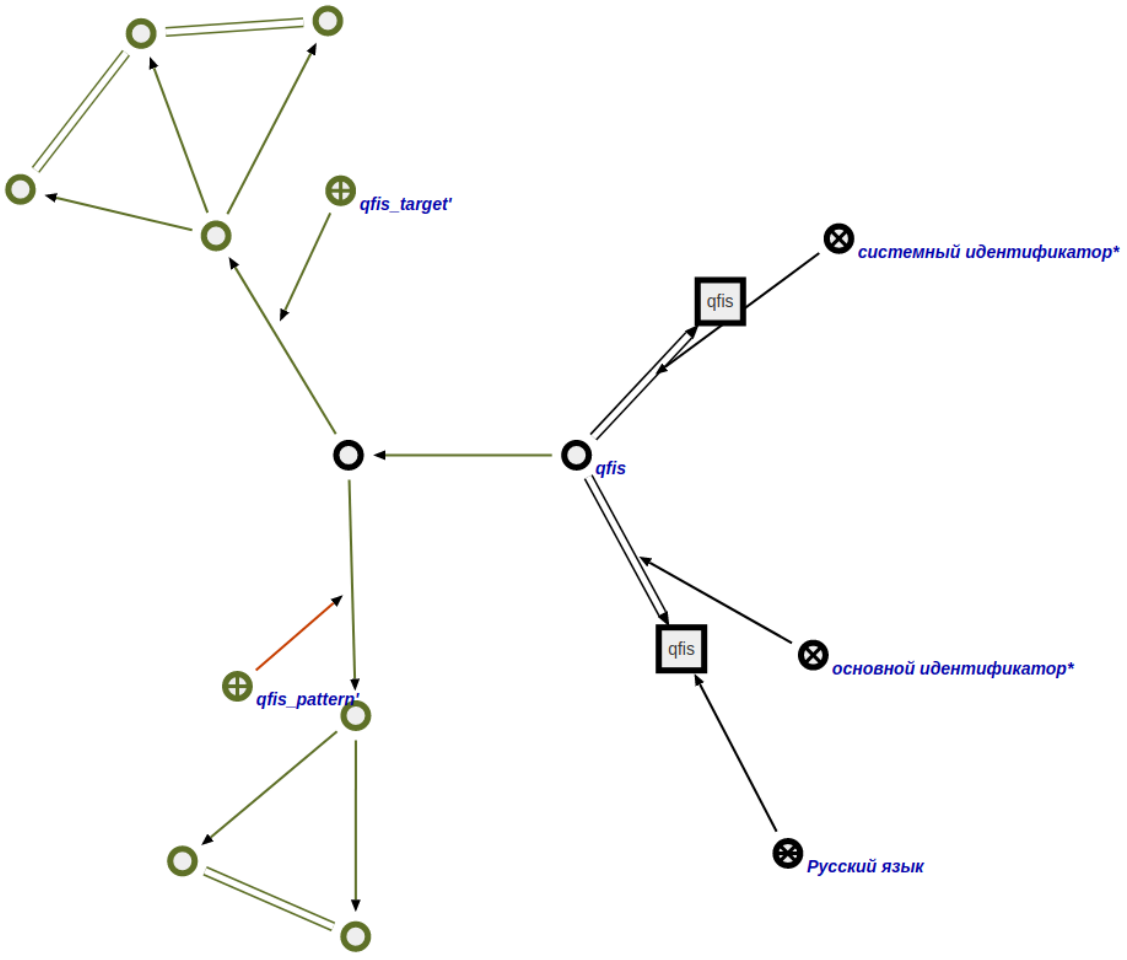

Рисунок 2.1.1. - Вход теста

Выход:

Найдено 4 подграфа, изоморфных графу-образцу: [{393829=393925, 393797=393893}, {393829=393861, 393797=393893}, {393829=393893, 393797=393861}, {393829=393893, 393797=393925}]

Тест 2

Вход:

В неориентированном графе необходимо найти все подграфы, изомофорные графу-образцу.

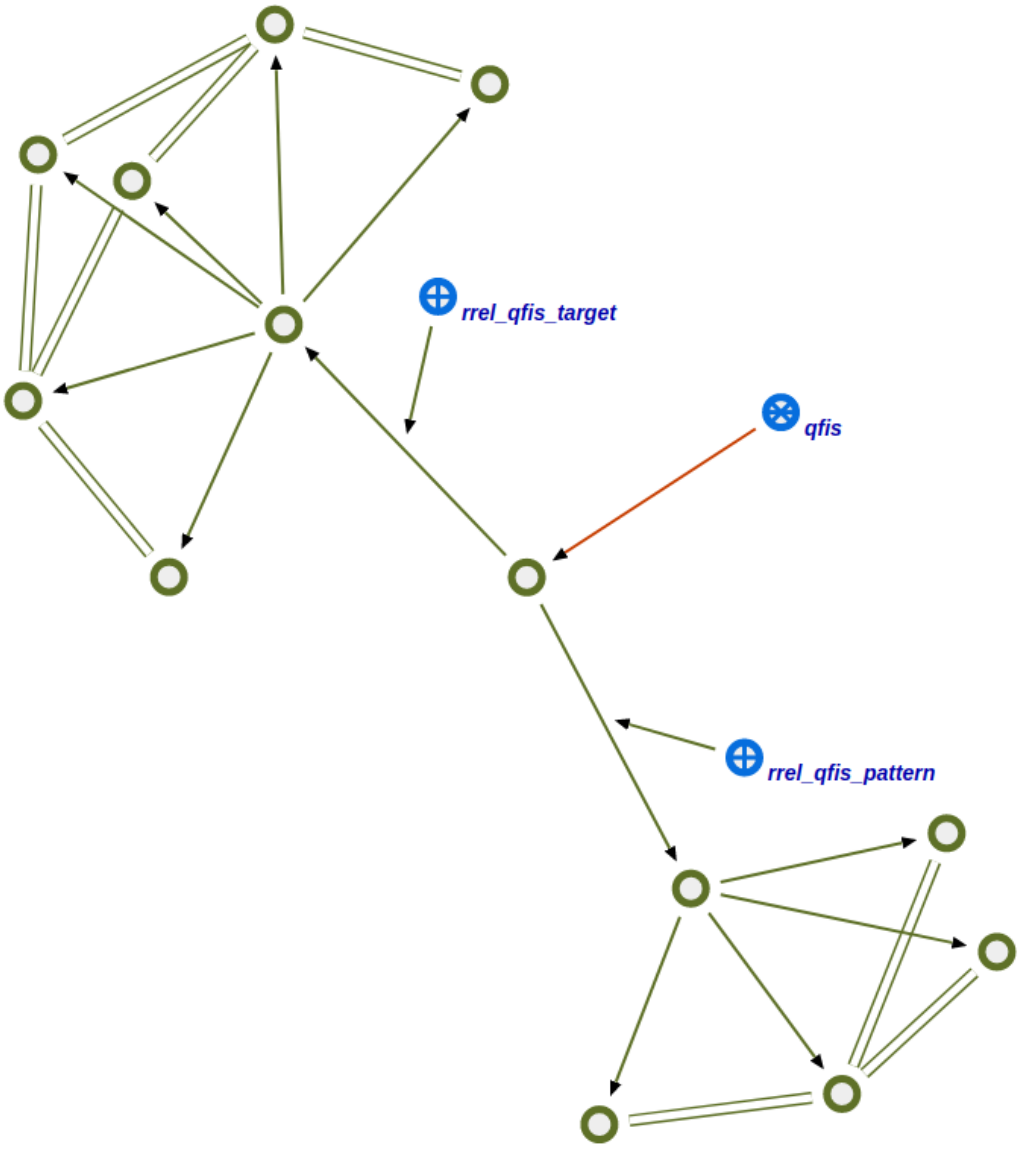

Рисунок 2.2.1. – Вход теста

Выход:

Найдены подграфы изоморфные графу-образцу:

[{393591=393431, 393623=393399, 393559=393367, 393527=393335},
{393591=393495, 393623=393399, 393559=393463, 393527=393431},
{393591=393399, 393623=393495, 393559=393463, 393527=393431},
{393591=393495, 393623=393431, 393559=393463, 393527=393399},
{393591=393431, 393623=393399, 393559=393463, 393527=393495},
{393591=393431, 393623=393495, 393559=393463, 393527=393399},
{393591=393335, 393623=393431, 393559=393367, 393527=393399},
{393591=393431, 393623=393335, 393559=393367, 393527=393399},
{393591=393399, 393623=393335, 393559=393367, 393527=393431},
{393591=393335, 393623=393399, 393559=393367, 393527=393431},
{393591=393399, 393623=393431, 393559=393367, 393527=393335},
{393591=393399, 393623=393431, 393559=393463, 393527=393495}]

Тест 3

Вход:

В неориентированном графе необходимо найти все подграфы, изомофорные графу-образцу.

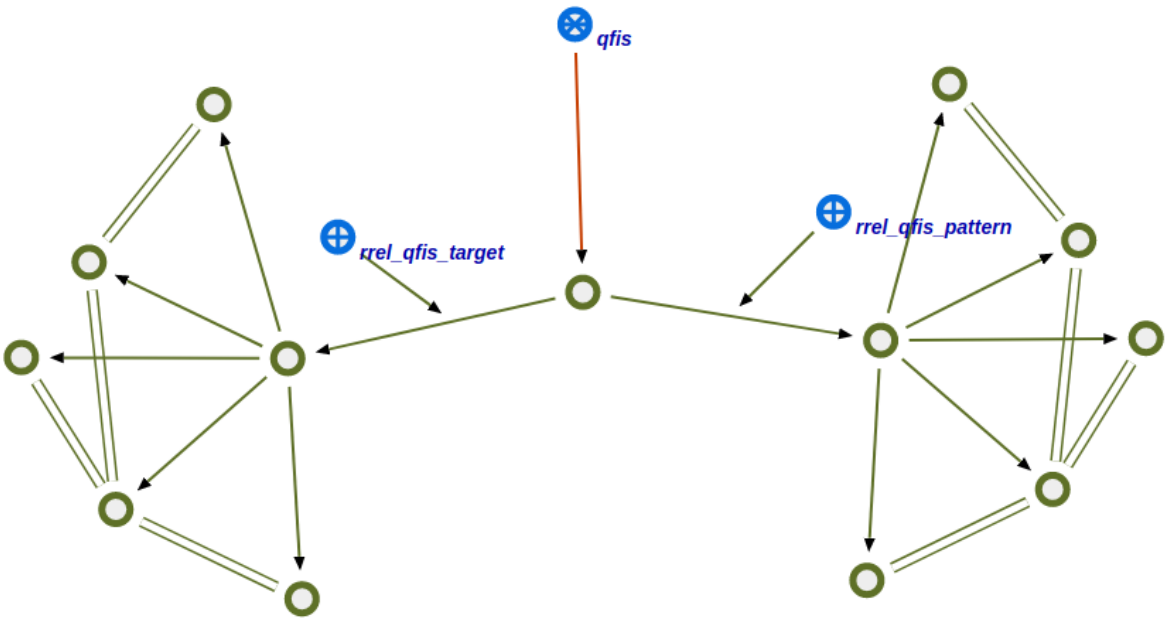

Рисунок 2.3.1. - Вход теста

Выход:

Найдено 2 подграфа, изоморфных графу-образцу: [{395991=395799, 395959=395767, 395927=395735, 395895=395703, 395863=395671}, {395991=395767, 395959=395799, 395927=395735, 395895=395703, 395863=395671}]

Тест 4

Вход:

В неориентированном графе необходимо найти все подграфы, изомофорные графу-образцу.

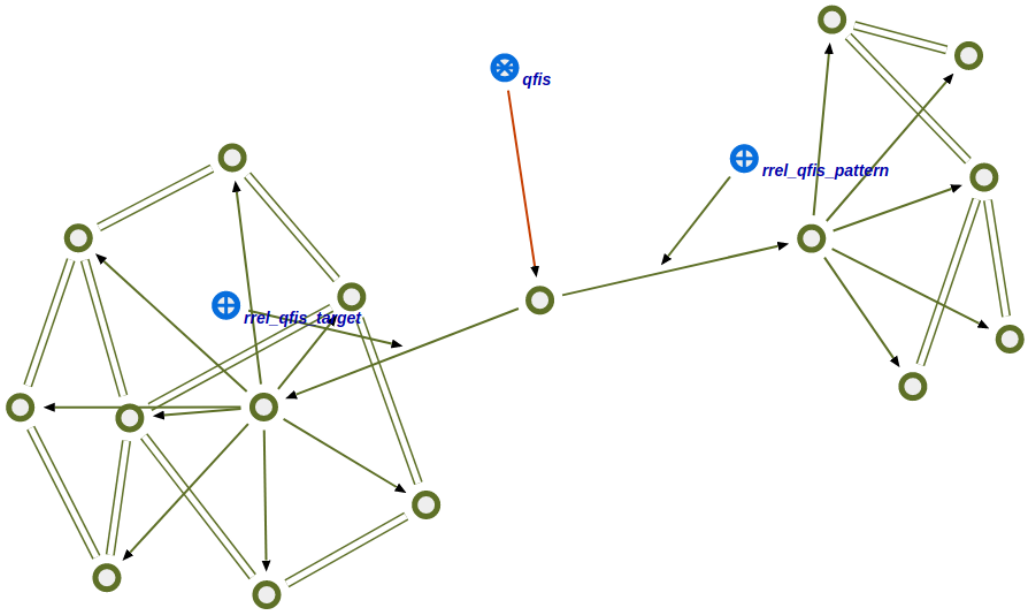

Рисунок 2.4.1. - Вход теста

Выход:

Найдено 16 подграфов изоморфных графу-образцу:

[{398359=398167, 398295=398103, 398327=398007, 398231=398039, 398263=398135}, {398359=398007, 398295=398103, 398327=398167, 398231=398039, 398263=398135}, {398359=398039, 398295=398103, 398327=398199, 398231=398167, 398263=398071}, {398359=398199, 398295=398103, 398327=398039, 398231=398167, 398263=398071}, {398359=398199, 398295=398103, 398327=398167, 398231=398007, 398263=397975}, {398359=398167, 398295=398103, 398327=398199, 398231=398007, 398263=397975}, {398359=398135, 398295=398039, 398327=397975, 398231=398103, 398263=398167}, {398359=398039, 398295=398103, 398327=398199, 398231=398007, 398263=398071}, {398359=397975, 398295=398039, 398327=398135, 398231=398103, 398263=398167}, {398359=398199, 398295=398103, 398327=398039, 398231=398007, 398263=398071}, {398359=398199, 398295=398103, 398327=398167, 398231=398039, 398263=397975}, {398359=398167, 398295=398103, 398327=398199, 398231=398039, 398263=397975},

{398359=398071, 398295=398007, 398327=397975, 398231=398103, 398263=398199}, {398359=398167, 398295=398103, 398327=398007, 398231=398199, 398263=398135}, {398359=397975, 398295=398007, 398327=398071, 398231=398103, 398263=398199}, {398359=398007, 398295=398103, 398327=398167, 398231=398199, 398263=398135}]

Тест 5

Вход:

В неориентированном графе необходимо найти все подграфы, изомофорные графу-образцу.

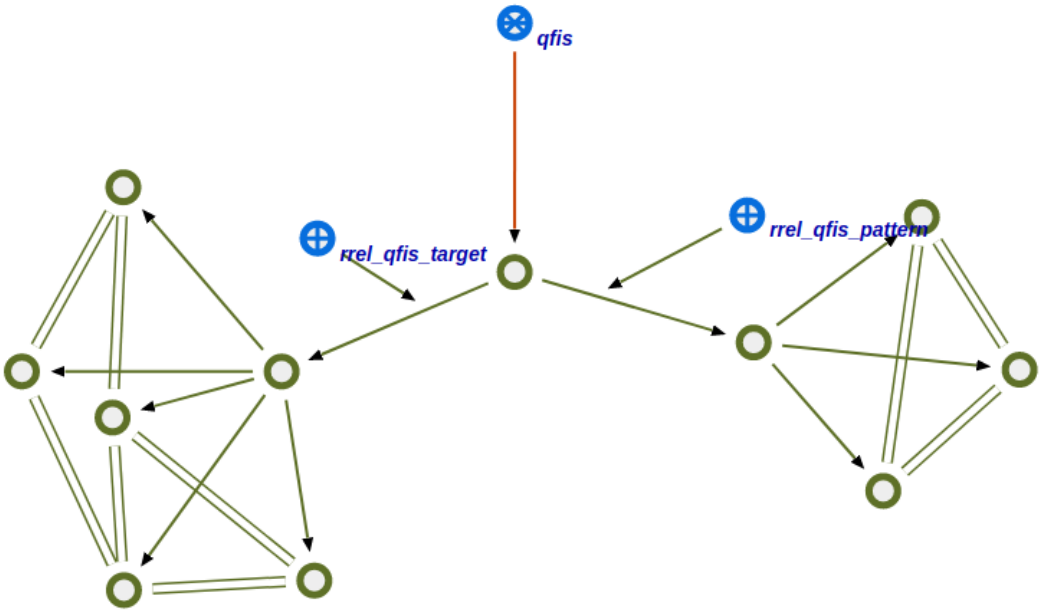

Рисунок 2.5.1. - Вход теста

Выход:

Найдено 6 подграфов изоморфных графу-образцу: [{393536=393408, 393472=393344, 393504=393376}, {393536=393408, 393472=393376, 393504=393344}, {393536=393376, 393472=393408, 393504=393344}, {393536=393376, 393472=393344, 393504=393408}, {393536=393344, 393472=393376, 393504=393408}, {393536=393344, 393472=393408, 393504=393376}]

Вывод

Мы получили навыки формализации и обработки информации с использованием семантических сетей, углубились в теорию графов, в частности, в изоморфизм графов. Разработали и проверили работоспособность алгоритма по поиску изоморфных подграфов в графе.

Список литературы

1. База знаний по теории графов OSTIS [Электронный ресурс]. – Режим доступа: http://ostisgraphstheo.sourceforge.net/index.php/. – Дата доступа: 06.04.2022.
2. Харрари, Ф. Теория графов / Ф. Харрари. – Москва : Едиториал УРСС, 2003.
3. Пономаренко, И. Н. Проблема изоморфизма графов: Алгоритмические аспекты (записки к лекциям) / И. Н. Пономаренко. – Санкт-Петербург : Санкт-Петербургское отделение Математического института им. В. А. Стеклова, 2010.