Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет Информационных технологий и управления
Кафедра Интеллектуальных информационных технологий

К защите допустить:
Заведующий кафедрой
_____ Д. В. Шункевич

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к расчётной работе
по дисциплине «Проектирование программ в интеллектуальных системах»:

Декартово произведение графов

БГУИР РР 1–40 03 35

Студент: В. В. Крюк
Группа: 221703

Руководитель: А. С. Никифоров

Минск 2023

СОДЕРЖАНИЕ

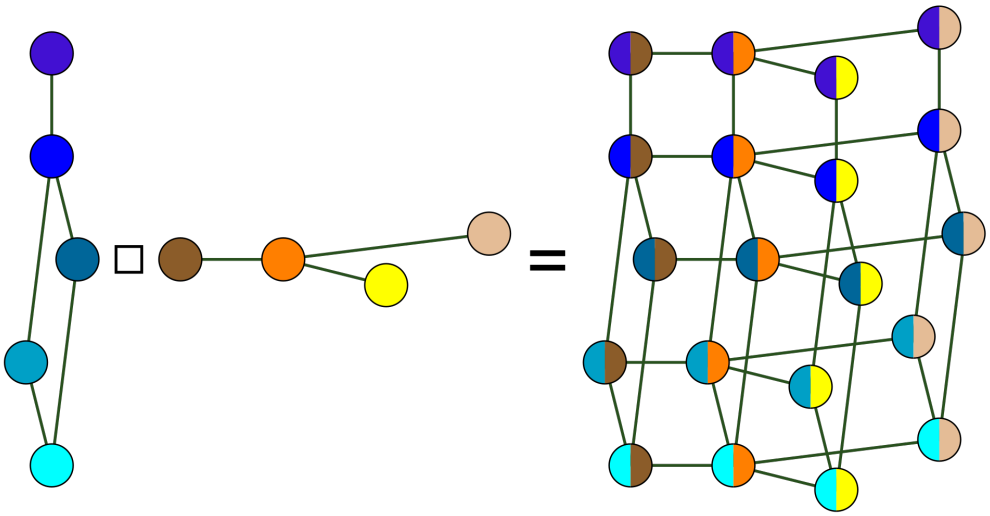
ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ

БЗ — база знаний;
SC — Semantic Code;
SCg —Semantic Code Graphical;
SCn — Semantic Code Natural;
it3 — Итератор для 2 узлов и связи между ними;
it5 — Итератор для 2 узлов и 1 узла отношения и связей между ними.

ВВЕДЕНИЕ

Цель: Создать агента для веб платформы OSTIS, который производит операцию декартова произведения двух неориентированных графов.
Задача: Нахождение декартова произведения двух неориентированных графов.

Декартово произведение[1] или прямое произведение двух графов G и H — это граф, состоящий из множества вершин и множества рёбер, соответствующий следующим условиям. Множество вершин этого графа — это декартово произведение вершин G и H. Множество рёбер состоит из рёбер, которые соответствуют следующим условиям. Любые две вершины (u,u') и (v, v') смежны в графе декартова произведения тогда и только тогда, когда либо u=v и u' смежна v' во H, либо u'=v' и u смежна v в G.



Далее в работе под графом будет подразумеваться строго неориентированный граф без взвешенных рёбер.

1 АЛГОРИТМ ДЕКАРТОВА ПРОИЗВЕДЕНИЯ

Алгоритм построения графа декартова произведения двух графов такой:

1) Создаём пустой граф, который будет являться результатом декартова произведения.

2) Создаём пустое множество вершин графа декартова произведения.

3) Производим операцию декартова произведения множеств вершин первого и второго графа. Результатом этой операции и будет множество вершин декартова произведения.

4) Создаём множество рёбер графа декартова произведения.

5) Заполняем его рёбрами по условию: Любые две вершины H(u,u') и G(v, v') смежны в графе декартова произведения тогда и только тогда, когда либо u=v и u' смежна v' во H, либо u'=v' и u смежна v в G. (H - первый граф, G - второй граф).

6) Из множества вершин и множества рёбер строим граф-результат.

7) Граф результат — является графом декартова произведения.

2 РЕАЛИЗАЦИЯ АЛГОРИТМА ДЛЯ АГЕНТА

На вход нашему агенту должны приходить 2 графа, Затем агент начинает работу согласно алгоритму, а именно создаёт структуру ответа и последовательно заносит исходные графы в неё.

2.1 Разработка

Для написания данного агента я руководствовался данной документацией[2].

Для начала я выбрал название для агента - $CartProdAgent$ (англ. $CartesianProduct$ - рус. Декартово произведение), а его системный идентификатор в экосистеме - $CardProd\_agent$. Потом я определил необходимый набор ключевых вершин: $CardProd$(агент).

Затем переходим к самому агенту. В начале мы принимаем входные графы через $OtherAddr$ в $ActionNode$ и создаём вектор адресов $Input$, в который при помощи $it3$ заносим все элементы входных графов. После создаём ещё один вектор адресов $newEdges$ уже для хранения результата отработки агента.

Чтобы декомпозировать код агента создадим функции: $createResponseConstruction$ (функция составления структуры ответа), $processing$(функция обработки входных графов в вектор структур, содержащий множество вершин графа декартова произведения), $detectEdges1$ и $detectEdges2$(функции создания рёбер графа декартова произведения, одна для перебора вершин по первому условию существования ребра, вторая по второму условию), а также функция $addResultEdges$, вызываемая из предыдущих двух функций.

Также хранение вершин графа декартова произведения реализовано в структуре $FinalNode$ с полями: адрес вершины первого графа $firstcolor$, адрес вершины второго графа $secondcolor$ и строка имя вершины $name$ графа декартова произведения.

2.1.1 $createResponseConstruction-$данная функция служит лишь для составления структуры ответа. Её аргументами являются: Данная функция служит лишь для составления структуры ответа. Её аргументами являются: $newEdges$(сама структура ответа), $graph1$ (первый исходный граф), $graph2$(второй исходный граф). В начале создаём вектор структур $newNodes$(служит для хранения множества вершин графа декартова произведения). И вызываем функции $processing$, $detectEdges1$ и $detectEdges2$.

2.1.2 $processing-$с помощью $it3$ проходимся по вершинам первого графа, для каждой его вершина, вызываем ещё один $ti3$, который проходится по второму графу, создавая множество вершин и соответственно заполняя

вектор структур вершин графа декартова произведения. Данные вершин теперь содержатся в $newNodes$.

2.1.3 $detectEdges1$ **и** $detectEdges2$—это функции создания рёбер графа декартова произведения, одна для перебора вершин по первому условию существования ребра, вторая по второму условию.

Для начала функции проходятся по всем вершинам графа декартова произведения с помощью вложенного цикла в for по j в цикл for по j, заполненного в прошлой функции, Во внутреннем цикле for с помощью $it5$ мы проверяем условие существования ребра между всеми возможными вершинами в графе декартова произведения. Мы используем первый $it5$ для проверки первой части условия — u = v и второй $it5$ для проверки второй части условия, а именно смежности u' и v'.

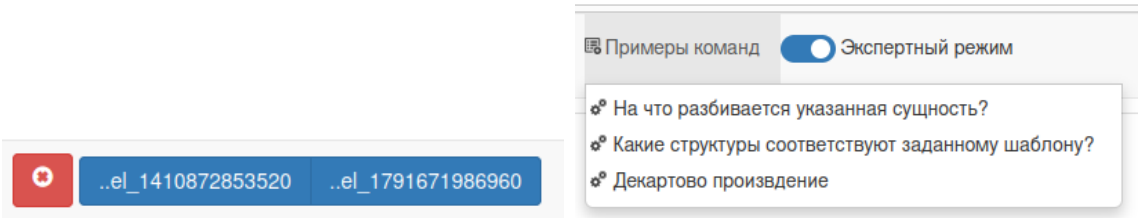
Если условия верны, то вызывается функция $addResultEdges$.

Это было описание для функции $detectEdges1$, функция $detectEdges2$ отличается от неё тем, что воспринимает входные значения наоборот. В моей реализации нам требуется 2 функции потому, что я имею дело с неориентированным графом, поэтому из-за особенностей построения графа в kbe. Так как необходимо проверить пути из узла 1 в узел 2 и наоборот, так как мы не знаем, от чего к чему строились рёбра в исходных графах.

2.1.4 $addResultEdges$—эта функция создаёт 2 узла и методом контекста $HelperResolveSystemIdtf$ присваивает им идентификаторы: создаёт или берёт идентификатор из узлов ответа(если такие узлы уже были созданы). Заполняет структуру ответа $newEdges$.

3 ПРИМЕРЫ РАБОТЫ АГЕНТА

Для начала агента нужно вызвать. Чтобы правильно это сделать нужно закрепить два исходных графа, из которых мы хотим посчитать граф декартова произведения, используя инструмент “булавка“. После чего вызвать агента через меню “Примеры команд“и нажать на ““ .



(а) (б)

Рисунок *3.1* – Закреплённые исходные графы (а) и Агент в выборе команд (б)

3.1 Тест 1

Проверка нормальной работы агента
Рисунок 3.2

3.2 Тест 2

Проверка нормальной работы агента
Рисунок 3.3

3.3 Тест 3

Проверка нормальной работы агента
Рисунок 3.4

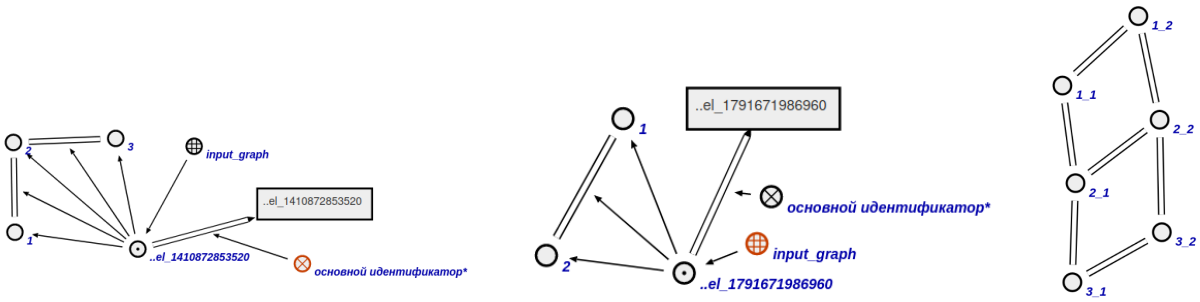
3.4 Тест 4

Проверка нормальной работы агента
Рисунок 3.5

3.5 Тест 5

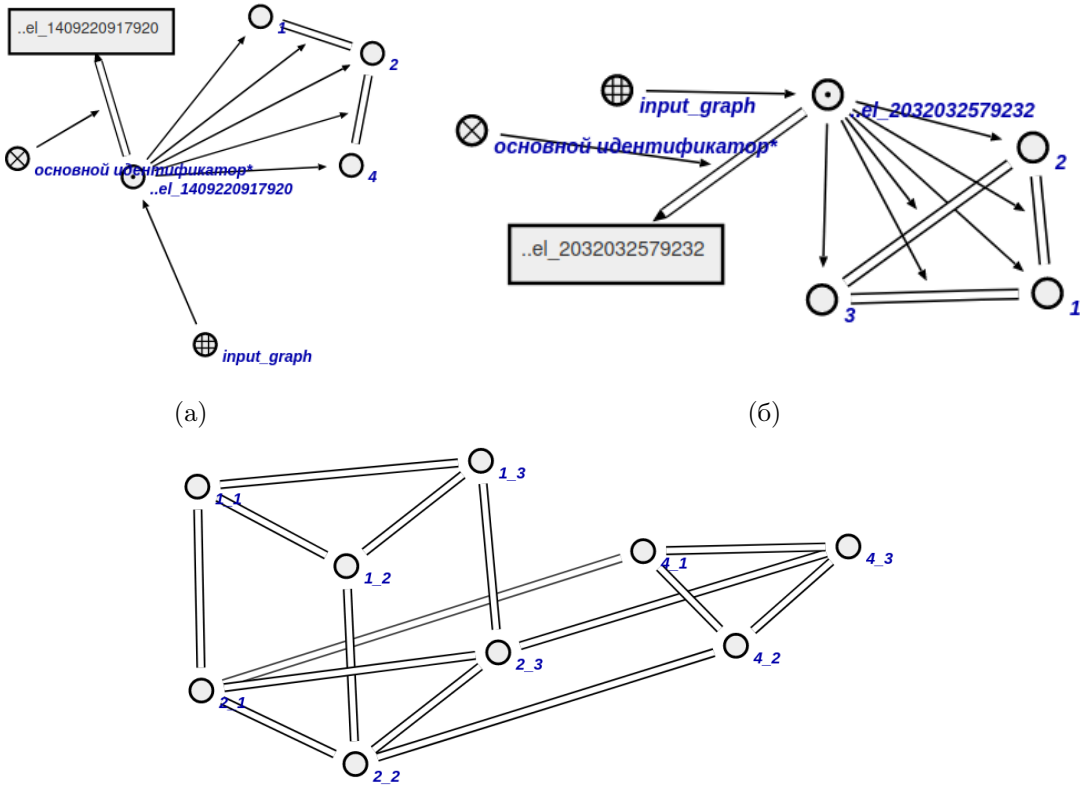
Проверка нормальной работы агента
Рисунок 3.6

3.6 Тесты рисунки



(а) (б) (в)

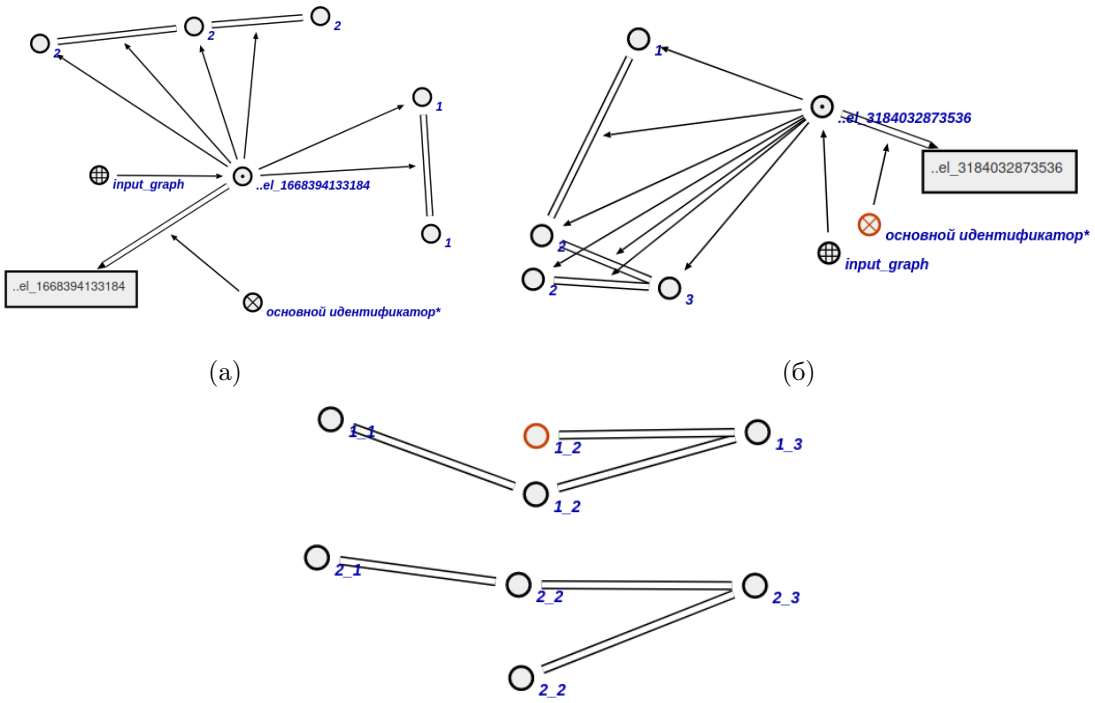
Рисунок *3.2* – Первый исходный граф (а), Второй исходный граф (б) и Результат отработки агента (в)



(а) (б)

(в)

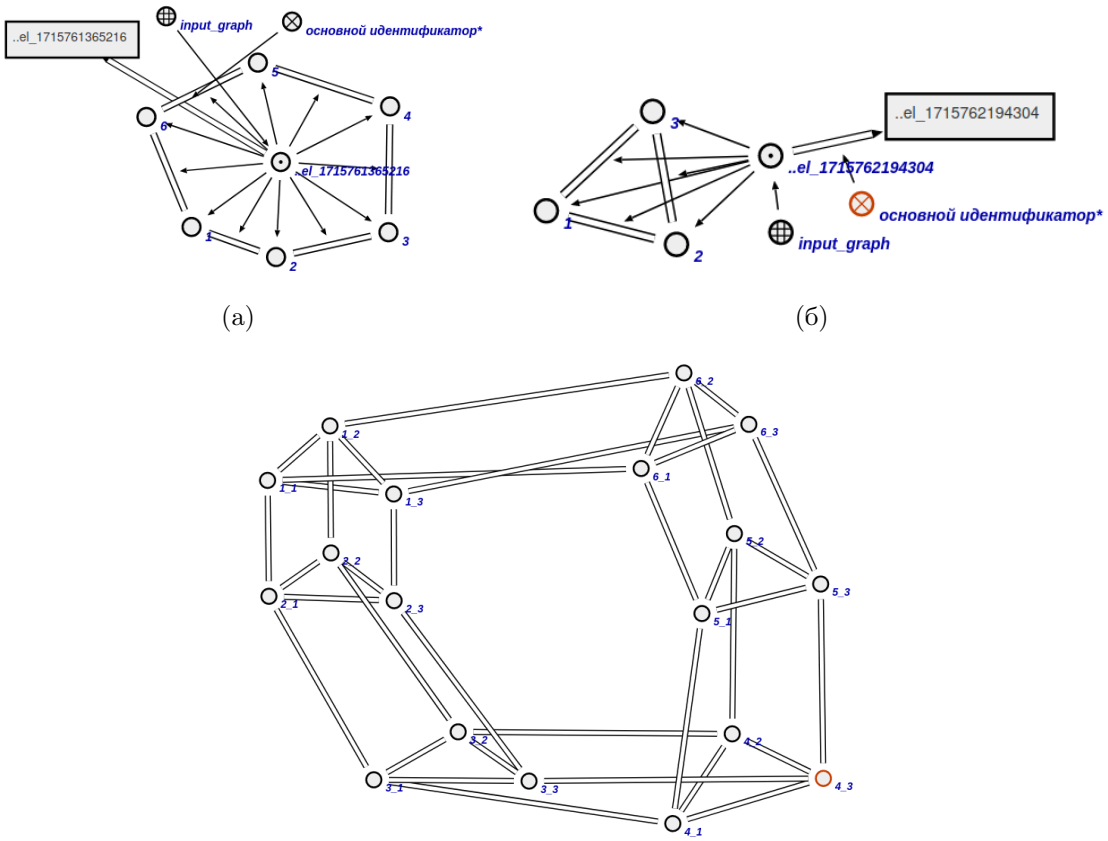
Рисунок *3.3* – Первый исходный граф (а), Второй исходный граф (б) и Результат отработки агента (в)



(а) (б)

(в)

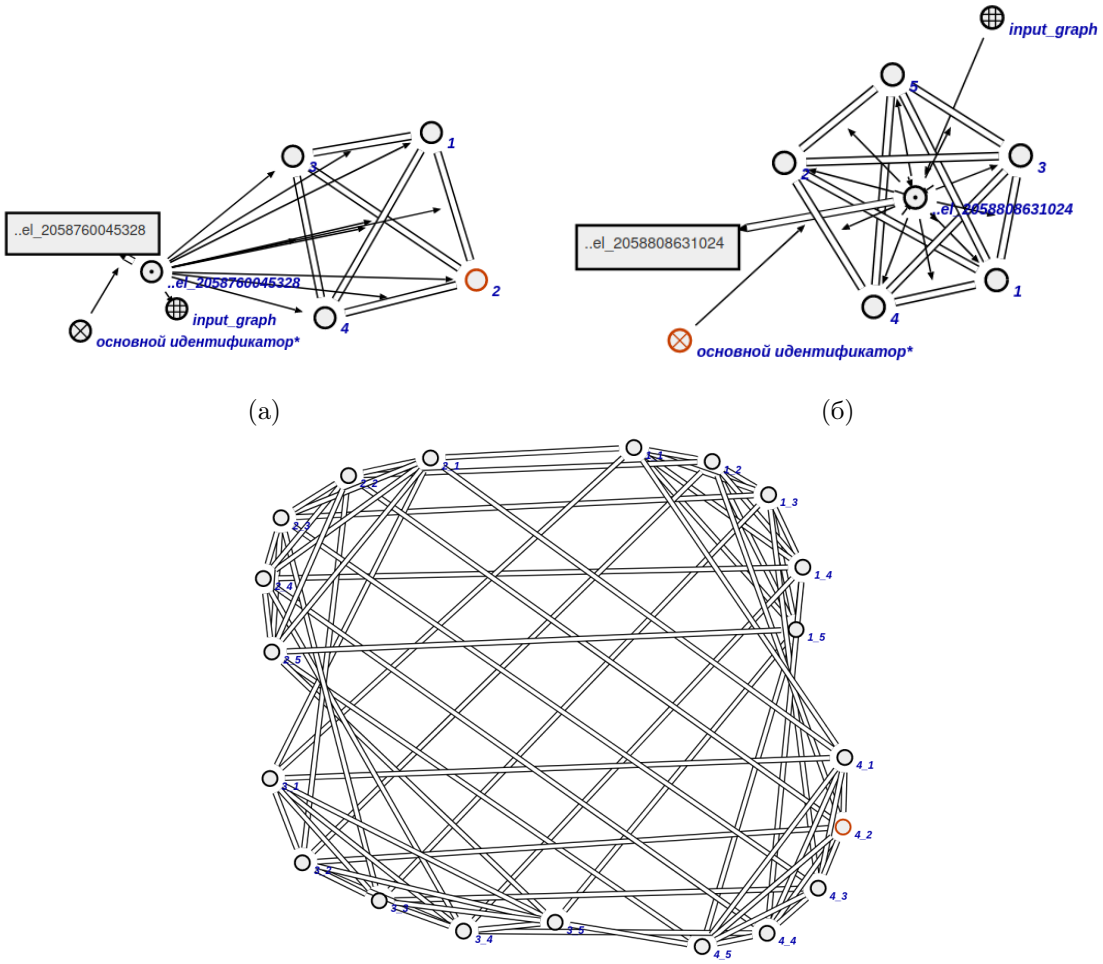
Рисунок *3.4* – Первый исходный граф (а), Второй исходный граф (б) и Результат отработки агента (в)



(а) (б)

(в)

Рисунок *3.5* – Первый исходный граф (а), Второй исходный граф (б) и Результат отработки агента (в)



(а) (б)

(в)

Рисунок *3.6* – Первый исходный граф (а), Второй исходный граф (б) и Результат отработки агента (в)

ЗАКЛЮЧЕНИЕ

Выполняя данную расчётную работу я изучил то, как работают агенты в экосистеме, и также написал своего агента, выполняющего операцию Декартова произведения графов. Повторил и улучшил свои знания теории графов и реализации алгоритмов на графах разной сложности. Изучил и применил различные методы взаимодействия с экосистемой. Усовершенствовал свои навыки проектирования программных средств на языке программирования C++.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Декартово произведение графов. ru.wikipedia.org/wiki/P§CЃCSPёP«Px_P»CЃP«PчPчPўPxPyPxP,,PчPx_PyCЃPeChP«Pў.

[2] Документация программного варианта реализации платформы интерпретации компьютерных систем. https://github.com/ostis-ai/ostis-web-platform/blob/develop/docs/.

[3] Документация программного варианта реализации платформы интерпретации компьютерных систем. https://github.com/Mediano4e/bsuir_guides/tree/main/example_app_agent.