НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

"На правах рукопису"
УДК_004.896____

«До захисту допущено»
Завідувач кафедри
_____ О.В. Коваль
(підпис) (ініціали, прізвище)
"___"_____2018р.

Магістерська дисертація

зі спеціальності - 121 Інженерія програмного забезпечення
за спеціалізацією - Програмне забезпечення розподілених систем
на тему: «Побудова та використання транзитивного замикання графа в онтологічно-орієнтованих навчальних системах»

Виконав: студент _6__курсу, групи_ТВ-71мп__
___Дудкін Юрій Миколайович _ _____ _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник _к.т.н., доц. Титенко С. В.____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент___ _____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ - 2018

Національний технічний університет України
“ Київський політехнічний інститут ім. Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти другий, магістерський

зі спеціальності - 121 Інженерія програмного забезпечення
за спеціалізацією - Програмне забезпечення розподілених систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
Коваль О.В. _____
(прізвище, ініціали) (підпис)
«____» _____ 2018р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ ДИСЕРТАЦІЮ СТУДЕНТУ

Дудкіну Юрію Миколайовичу
(прізвище, ім’я, по батькові)

1. Тема дисертації Побудова та використання транзитивного замикання графа в онтологічно-орієнтованих навчальних системах
науковий керівник Титенко Сергій Володимирович к. т. н., доц.
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету №___ від “___”_____20___року
2. Строк подання студентом дисертації _____
3. Об’єкт дослідження програмне забезпечення інформаційно-навчальних порталів
4. Предмет дослідження створення транзитивного замикання для графу онтологічно-орієнтованої бази знань
5. Перелік питань, які потрібно розробити
1) розглянути інформаційно-навчальні портали;
2) проаналізувати існуючі алгоритми створення транзитивного замикання;
3) розробити алгоритм транзитивного замикання;
4) розробити алгоритм оновлення транзитивного замикання;
5.) створити програмний продукт з використанням розроблених алгоритмів транзитивного замикання.

6. Орієнтований перелік ілюстративного матеріалу _____
1) Діаграма класів
2) Структура бази даних
3) Етапи роботи з програмою
4) Функції програмного забезпечення
5) Інтерфейс
7. Орієнтований перелік публікацій _____
1) Дукін Ю.М., Титенко В.В. "Використання ациклічного графа транзитивного замикання в онтологічних системах".
8. Дата видачі завдання «_29_»__вересня____ 2017 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання магістерської дисертації	Строки виконання етапів магістерської дисертації	Примітка
1	Отримання завдання	29.09.17р.	
2	Збір інформації	09.10.17р. – 25.01.18р.	
3	Аналіз вимог завдання, вибір методів і засобів розв'язання поставленої задачі	26.01.18р. – 05.09.18р.	
4	Підготовка публікацій	19.07.18р., 02.09.18р.	
5	Підготовка доповідей на конференції	15.08.18р., 20.10.18р.	
6	Підготовка дисертації	03.07.18р. – 07.12.18р.	
7	Розробка програмного продукту	10.01.18р. – 09.10.18р.	
8	Захист програмного продукту	24.10.18р.	
9	Передзахист	28.11.18р.	
10	Захист	17.12.18р.	

Студент _____ Дудкін Ю.М.
(підпис) (прізвище та ініціали)

Науковий керівник _____ Титенко С.В.
(підпис) (прізвище та ініціали

РЕФЕРАТ

Структура й обсяг дипломної роботи.

Магістерська дисертація складається зі вступу, п'яти розділів, висновку, переліку посилань з 32 найменувань, 2 додатки, і містить 32 рисунки, 22 таблиці. Повний обсяг магістерської дисертації складає 73 сторінок, з яких перелік посилань займає 3 сторінок, додатки – 6 сторінок.

Актуальність теми. Стрімкий розвиток галузей діяльності людини, що оперують великою кількістю наукової та прикладної інформації створює необхідність пошуку нових способів зберігання, представлення та обробки великих масивів даних. Одними із можливих підходів роботи з великою кількістю інформації певних предметних областей надають отологічні системи. Це зумовлює актуальність досліджень пов’язаних з роботою та знаходження способів модернізації систем такого роду.

Мета дослідження полягає у визначенні можливих способів подачі інформації користувачам, оптимізації роботи системи.

Для досягнення поставленої задачі були сформульовані наступні **завдання дослідження**, що визначили логіку дослідження та його структуру:

- проаналізувати існуючі способи подачі інформації користувачу;
- структури для збереження даних;
- знайти частини у системах, що можливо удосконалити за допомогою використання нових алгоритмів та структур даних;
- модифікувати або розробити алгоритм або алогритми для модернізації систем;
- довести корректну роботу алгоритму та можливість його реалізації у системі

− розробити програмну систему, що містить реалізацію розроблених алгоритмів.

Об’єктом дослідження алгоритми для модернізації роботи онтологічної системи.

Предметом дослідження є розробка алгоритму побудови та використання транзитивного замикання.

Методи дослідження. Розв’язання поставлених задач виконувались такими засобами:

− дослідження актуальних алгоритмів для побудови транзитивного замикання;

− перевірка існуючих рішень емпіричним та аналітичним шляхом;

− перевірка розроблемних алгоритмів емпіричним та аналітичним шляхом;

− критика знайдених рішень з ціллю виявлення слабких сторін та можливих помилок.

Практичне значення одержаних результатів роботи полягає в розробці програмної системи, що дозволить будувати та використовувати транзитивне замикання для онтологічно-орієнтованої системи через веб інтерфейс з повною точністю та прийнятною швидкістю роботи.

Ключові слова. *онтологічні системи*, *алгоритми*, *транзитивне замикання графу*, *база знань*, *оновлення транзитивного замикання*.

ABSTACT

The structure and volume of the thesis.

The master's dissertation consists of an introduction, five sections, a conclusion, a list of references from 32 titles, 2 applications, and contains 32 figures, 22 tables. The full volume of the master's dissertation is 73 pages, of which the list of links takes 3 pages, applications - 6 pages.

Topicality of the theme. The rapid development of human activities that employ a large number of scientific and applied information creates the need for new ways of storing, presenting and processing large amounts of data. One of the possible approaches to dealing with a large amount of information in certain subject areas is provided by the otolaryngeal systems. This determines the relevance of research related to work and finding ways to upgrade such systems.

The purpose and problems of research puts in the definition of possible ways to provide information to users, optimizing the system.

To achieve the task, the following were formulated the following **research objectives** that determined the logic of the research and its structure:

- analyze the existing ways of submitting information to the user;
- structures for data storage;
- to find parts in systems that can be improved by using new algorithms and data structures;
- modify or develop an algorithm or algorithms for system upgrades;
- prove the correct operation of the algorithm and the possibility of its implementation in the system;
- - develop a software system that contains the implementation of developed algorithms

The object of research algorithms for modernizing the ontological system.

The subject of research is the development of the algorithm for the construction and use of transient closure**.**

Research methods. The solution of the set tasks was carried out by the following means:

- research of actual algorithms for construction of transitive closure;
- checking existing solutions empirically and analytically;
- testing of elaborated algorithms empirically and analytically;
- a method of generating code templates based on preset using Mustache library to generate code for mobile applications based on a given database schema;
- criticism of found solutions with the aim of identifying weaknesses and possible mistakes.

The practical significance of the results obtained is to develop a software system that will allow the construction and use of a transient circuit for an ontological-oriented system through a web interface with full accuracy and acceptable work speed.

Keywords. ontological systems, algorithms, transitive graph closure, knowledge base, updating of transient closure.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

БД	База даних, база знань, транзитивне замикання (графу)
SQL	Structured query language — мова структурованих запитів
ТЗ	Транзитивне замикання
ТЗГ	Транзитивне замикання графу
СУБД	Система керування базами даних
MVC	Model-view-controller — Модель–вигляд–контролер
HTML	HyperText Markup Language — Мова розмітки гіпертекстових документів
IDE	Integrated development environment — Інтегроване середовище розробки
UI	User Interface — Інтерфейс користувача
ПК	Персональний комп'ютер
CRUD	Create read update delete (створення, зчитування, зміна і видалення)

ВСТУП

Розвиток галузей діяльності людини, що оперують великою кількістю наукової інформації створює необхідність пошуку нових способів зберігання, представлення та обробки великих масивів даних. Одними із можливих підходів роботи з великою кількістю інформації певних предметних областей надають отологічні системи.

Термін «Онтологія» прийшов з філософії і спершу означав науку про основні фундаментальні проблеми існування. Як і у філософії, немає однозначного визначення поняття онтологія. Найбільш частіше використовується визначення за Н. Грубертом «Онтологія – це специфікація концептуалізації». Під концептуалізацією слід розуміти чітке визначення системи понять, об'єктів і інших сутностей і відношень, пов'язаних один з одним[1].

Онтологічні системи дуже сильно пов'язані з такою структурою даних, як граф. Така структура даних вимагає ретельного підбору логіки для обробки, оскільки більшість нетривіальних алгоритмів для обробки графу мають велику складність. Також важливим є той факт, що зазвичай такі системи містять високий об'єм даних. Отже оптимізація роботи системи, є однією з самих пріоритетних задач при створення та підтримці такої системи. Серед багатьох можливостей оптимізації є використання транзитивного замикання графу.

Транзитивне замикання графа G (ТЗГ) – це граф G*, що складається з тих же вершин, що і граф G, тобто G* = (V, E*), множиною ребер якого є $E* = \{(u,v)$: $(u \in V)$ & $(v \in V)$ & (існує шлях від u до v у графі G) $\}$ [2]. Іншими словами це копія графу G, що також містить ребра для будь-яких вершин, між якими існує шлях у графі G.

Для створення ТЗ орієнтованого графу існує ряд алгоритмів. Проте, у випадку використання нетривіальної логіки для обчислення ваги ребер у графі транзитивного замикання можуть виникати проблеми, що викликають необхідність у детальному алалізі та підборі алгоритму, що знаходить ТЗГ.

1. ЗАДАЧА СТВОРЕННЯ ПРОГРАМНОЇ СИСТЕМИ ПОБУДОВИ ТА ВИКОРИСТАННЯ ТРАНЗИТИВНОГО ЗАМИКАННЯ

Оскільки існує потреба швидкого знаходження зв’язків графу була поставлена задача розробити програмну систему створення та використання транзититивного замикання.

Була спроба вирішинти задачу створення транзитивного замикання онтологічного графу за допомогою алгоритму Флойда-Уоршала. Але результат використання цього алгоритму був визначений як незадовільний, оскільки алгоритм має високу складність N^3і не дозволяє вирішити цю задачу за відведений час[2]. Під час додання нових звязків перерахування займало значний час для кожного нового зв’язку.

1.1 Загальні вимоги до розроблюваної системи

Розроблювана система має працювати з графом, вершинами якого є концепти або поняття. Граф є зважений, значення його ребер відображають достовірність зв'язоку між поняттями. Це значення називається фактором впевненості або коефіцієнтом впевненості і може мати дробове значення від 0 до 1. Граф має зберігатися у реляційній СУБД MySQL.

Для реалізації програмного коду серверної частини необхідно використати мову програмування PHP. Це дозволить швидко та гнучко додавати зміни у систему у разі необхідності. Також вагомим аргументом для використання саме цієї мови програмування є те, що існує ряд онтологічних систем, що працюють на мові PHP і у подальшому можливе інтегрування розроблюваної системи у дані системи.

Для розробки інтерфейсу користувача необхідно використати мову JavaScript, оскільки на даний момент це найбільш розвинена мова для WEB інтерфейсів.

У розроблюваній системі мають бути реалізовані такі функції:

- перегляд існуючих понять;
- пошук понять по префіксу слова
- перегляд дочірніх понять;
- перегляд усього графу понять з усіма зв'язками
- додавання зав'язків між поняттями;
- змінення коефіцієнту впевненості зав'язків між поняттями;
- видалення зав'язків між поняттями;
- створення транзитивного замикання на основі понять;
- оновлення транзитивного замикання після редагування зав'язків понять;
- перегляд транзитивних зав'язків певного поняття.

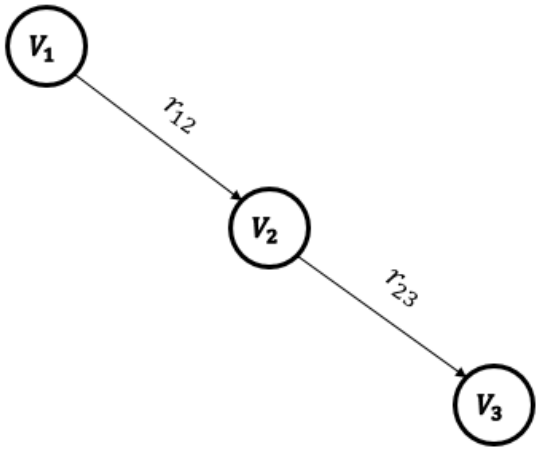
1.2 Логіка обчислення коефіцієнту впевненості транзитивних звязків

Як було указано, зв'язки понять мають коефіцієнт впевненості. Під час обчислення ТЗГ для транзитивних зв'язків також необхідно розрахувати коефіцієнт впевненості. Для цього є деякі правила обчислення коефіцієнту для шляхів.

Побудова транзитивних зв'язків між поняттями і формування транзитивного замикання графу онтології ґрунтується на правилі [3]:

$$concept\_before(c_k, c_l)\langle CF_{kl}\rangle \wedge concept\_before(c_l, c_m)\langle CF_{lm}\rangle \rightarrow$$
$$concept\_before(c_k, c_m)\langle CF_{kl} \times CF_{lm}\rangle$$

Отже, для послідновного об'єднання ребер використовується операція множення. На рисунку 1.2 можливо побачити приклад. Для обчислення відстані між вершинами V_1 та V_3, тобто знаходження r_{13}, необхідно помножити відстань від вершини V_1 до V_2 - r_{12} та аналогічну відстань від V_2 до V_3 - r_{23}. З цього можливо вивести формулу для даного випадку: $r_{13} = r_{12} \cdot r_{23}$.



s

Рисунок 1.2

У відповідності до логіки Бьюкенона для обчислення паралельних зв'язків використовується формула попарного об'єднання $CF = CF + CF_i - CF \cdot CF_i$ [3]. На рисунку 1.3 можливо побачити приклад об'єднання шляхів від V_1 до V_4 через вершини V_2 та V_3. Згідно з формули обчислення паралельних зв'язків кінцева для знаходження відстані від V_1 до V_4 буде $r_{14} = r_{124} + r_{134} - r_{124} \cdot r_{134} = r_{12} \cdot r_{24} + r_{13} \cdot r_{34} - r_{12} \cdot r_{24} \cdot r_{13} \cdot r_{34}$

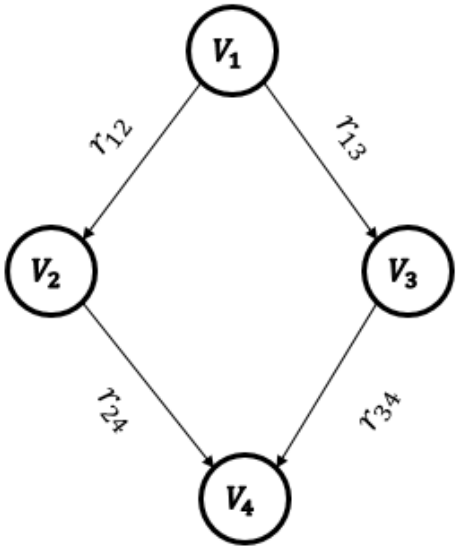


Рисунок 1.3

1.3 Алгоритм часткового оновлення транзитивного замикання

Оскільки система розрахована на додавання нових зв’язків та зміну існуючих, виникає необхідність у оптимізації пошуку транзитивного замикання після певних змін. Повністю перераховувати увесь граф транзитивного замикання не має можливості, оскільки цей процес займає значній час через наявність великої кількісті інформації. Також важливим є те, що необхідно повністю перезаповнювати реляційну таблицю, а це займає дуже багато часу і зі зростом системи час буде значно збільшуватись.

Алгоритм часткового оновлення графу має задовольняти такі вимоги:

- перерахування тільки тих зв’язків, що цього будуть змінені після операції;
- запити до бд тільки для тих зв’язків, що зазнали змін, або для нових.

Висновки до розділу 1

Отже необхідно розробити програмну систему, що дозволятиме користувачу зручно керувати зв’язками між поняттями та будувати транзитивне замикання для графу понять. Також необхідно реалізувати зручний перегляд понять та зв’язків між ними. Система також має перебудовувати існуюче транзитивне замикання графу після змін. При цьому потрібно урахувати логіку обчислення послідовних та паралельних зв’язків. Також бажаним функціональністю є візуалізація графу понять. Система має бути створення на WEB платформі за допомогою таких мов програмування як PHP та JavaScript з використанням СУБД MySQL.

2. АЛГОРИТМ ПОБУДОВИ ТРАНЗИТИВНОГО ЗАМИКАННЯ

Як зазначено в задачі необхідно знайти алгоритм для побудови транзитивного замикання графу, який містить інформацію про поняття та їх зв'язки з фактором впевненості. Для цього необхідно розглянути існуюче рішення за допомогою Флойда-Уоршала.

2.1 Алгоритм Флойда-Уоршала

Алгоритм Флойда-Уоршелла призначений для вирішення завдання пошуку всіх найкоротших шляхів на графі. Для заданого орієнтованого зваженого графа алгоритм знаходить найкоротші відстані між усіма парами вершин за час $O(V^3)$.

Алгоритм застосуємо до графів з довільними, в тому числі з негативними, вагами. Таким чином, він є більш загальним в порівнянні з алгоритмом Дейкстри, який не працює з негативною вагою ребер. Також алгоритм розпізнає наявність негативних циклів. Маємо граф $G = (V, E)$, в якому кожна вершина пронумерована від 1 до $|V|$. Сформуємо матрицю суміжності D. Ця матриця має розмір $|V| * |V|$ і кожному її елементу Дій присвоєно вагу ребра, що з'єднує вершину I з вершиною ж. Зауважимо, що в силу орієнтованості графа G матриця D може бути несиметрична.

По мірі виконання алгоритму дана матриця буде заповнюватись: в кожну з її комірок буде внесене значення, що визначає оптимальну довжину шляху з вершини і в вершину j Вважаємо діагональні елементи D_{ii} рівними нулю, а недіагональні елементи, що відповідають не інцидентним вершин (які не мають загального ребра), покладемо з значенням нескінченності або числом, свідомо більшим можливої відстані між ребрами. Ключова частина алгоритму складається з трьох циклів (Рисунок 2.1).

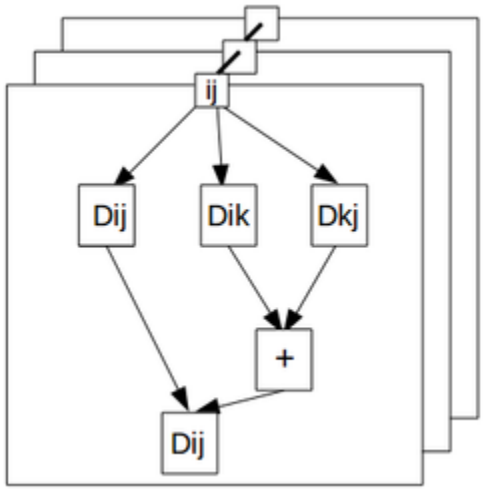


Рисунок 2.1 Алгоритм Флойда-Уоршала

Алгоритм перевіряє транзитивне замикання між вершинами i та j через вершину k. Змінна k – це лічильник зовнішнього циклу, який містить у собі цикл з лічильником i, який у свою чергу містить цикл j. Реалізація алгоритму дуже проста – 3 цикли і одна умова з командою. Проте очевидно, що для алгоритм може давати неочікуваний результат для тих випадків, коли важливий порядок обробки вершин.

Тому для паралельного обчислення коефіцієнту впевненості алгоритм не підходить. Це можливо продемонструвати на випадку, коли нумерація вершин не відповідає необхідному порядку їх обчислення (Рисунок 2.2).

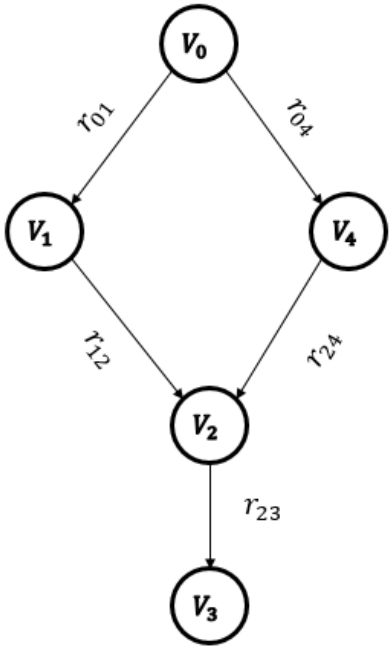


Рисунок 2.2 Приклад графу з довільним порядком вершин

Згідно з логікою паралельного та послідовного обчислення фактору впевненості очікуваним результатом коефіцієнта впевненості між вершинами V_1 та V_3 буде знайдено за формулою $r_{03} = r_{02} \cdot r_{23} = (r_{012} + r_{042} - r_{012} \cdot r_{042}) \cdot r_{23} = (r_{03} \cdot r_{12} + \cdot r_{04} \cdot r_{42} - \cdot r_{01} \cdot r_{12} \cdot r_{04} \cdot r_{42}) \cdot r_{23} = r_{03} \cdot r_{12} \cdot r_{23} + \cdot r_{04} \cdot r_{42} \cdot r_{23} - \cdot r_{01} \cdot r_{12} \cdot r_{23} \cdot r_{04} \cdot r_{42}$

Алгоритм Флойда-Уоршала першим чином знайде $r_{02} = r_{01} \cdot r_{12}$ (ітерація k=1). На ітерації k = 2 буде знайдено фактор впевненості від V_0 до V_3 через вершину V_1 як $r_{03} = r_{01} \cdot r_{12} \cdot r_{23}$. Також буде фактор впевненості від V_4 до V_3 як $r_{43} = r_{24} \cdot r_{23}$. Досі не має помилкових значень при обчисленні, не беручи уваги незакінчене значення фактору впевненості r_{03}. Проте на k=4 ітерації буде знайдено коефіцієнт між вершинами V_0 та V_3 і за формулою паралельного обчислення буде додано до існуючого значення. А отже кінцевий результат буде виглядати таким чином: $r_{03} = r_{03} \cdot r_{12} \cdot r_{23} + r_{04} \cdot r_{42} \cdot r_{23} - \cdot r_{01} \cdot r_{12} \cdot r_{23} \cdot r_{23} \cdot r_{04} \cdot r_{42}$. Одразу можливо побачити зайвий множник r_{23}, що доволі логічно, оскільки дане ребро алгоритм включає двічі, а формула паралельного обчислення коефіцієнта має відбуватися для вершини V_2 а не для вершини V_3.

2.1 Топологічне сортування

Отже, спираючись на отримані результати дослідження роботи алгоритму Флойда-Уоршала, постає очевидним використання певного порядку вершин при створенні транзитивного замикання графу. Для знаходження порядку найлогічнішим є використання топологічного сортуванння орієнтованого графу.

Топологічне сортування — впорядковування вершин безконтурного орієнтованого графа згідно з частковим порядком, визначеним ребрами цього графу на множині його вершин. Тобто у списку вершин, отриманого топологічним сортуванням кожна вершина розташована так, що кожна її дочірня вершина знаходиться далі за списком, а кожна батькіська вершина знаходиться за списком раніше. Очевидно, що для такого порядку граф має бути ациклічний, проте дані

онтологічної системи задовольняють цю потребу[7].

Для топологічного сортування використовується алгоритм Тарьяна. Основа алгоритму – це алгоритм пошуку в глибину. Пошук в глибину або обхід в глибину (англ. Depth-first search, скорочено DFS) - один з методів обходу графа. Алгоритм пошуку описується наступним чином: для кожній групі пройденої вершини необхідно знайти все не пройдені суміжні вершини і повторити пошук для них[8].

2.2 Алгоритм пошуку транзитивного замикання

Розроблений алгоритм пошуку транзитивного замикання використовує ідею топологічного сортування. Візуалізацію роботи алгоритму можливо побачити на рисунку 2.3. Алгоритм складається з трьох циклів. Цикли з лічильником і та лічильником j проходять по топологічно відсортованому списку вершин графу. При цьому цикл j стартує з вершини i + 1.

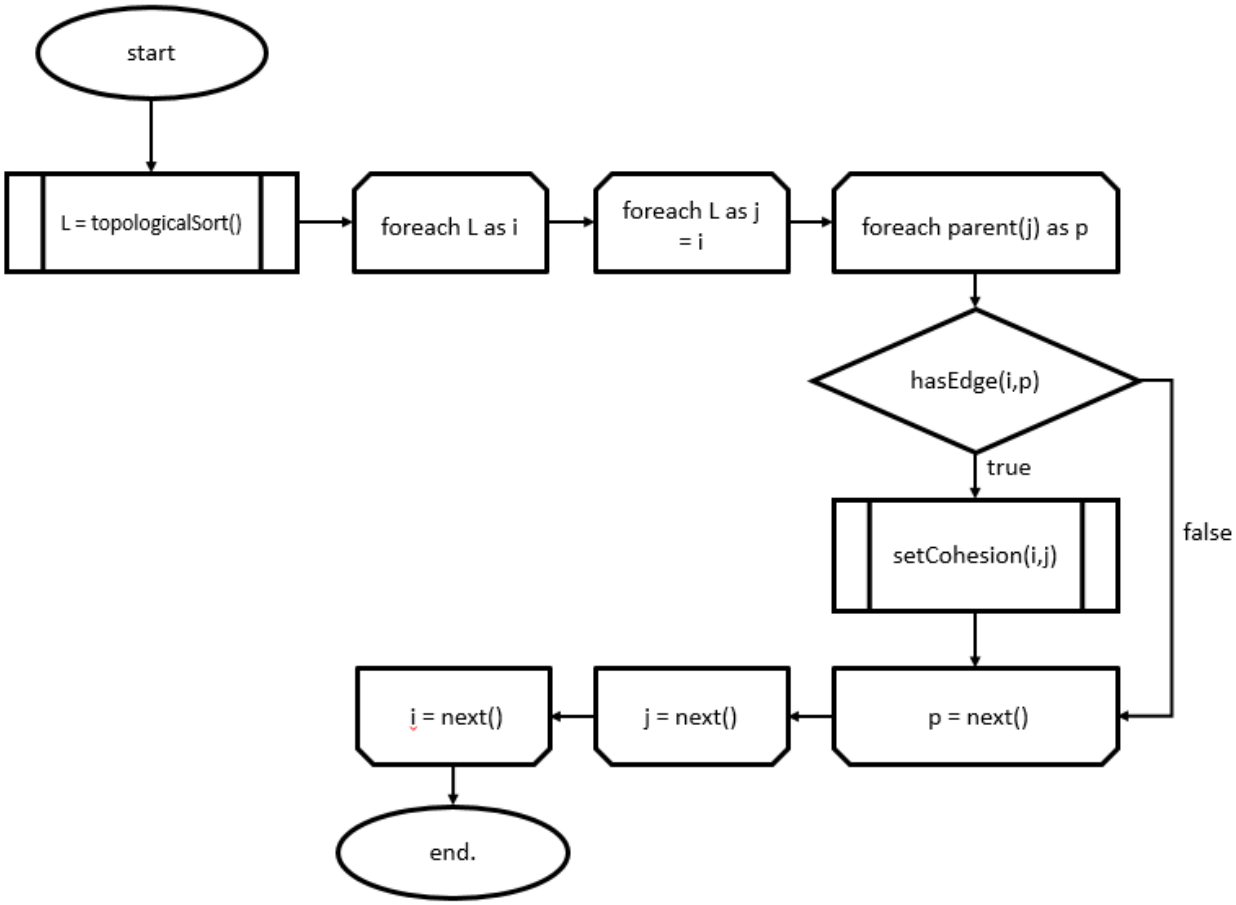


Рисунок 2.3 – Візуалізація роботи алгоритму

Третій цикл – це прохід по батьківським вершинам вершини j. Під час проходу перевіряється чи існує зв'язок між вершиною і та батьківською вершиною p вершини j.

Якщо такий зв'язок існує, тобто існує ребро між вершиною і та вершиною p викликається функція setCohesion(), що зберігає транзитивний зв'язок між вершинами і та j. Сама функція використовує властистивість послідовного множення фактору впевненості транзитивного замикання (рисунок 2.4). Для цього відбувається множення фактору впевненості від вершини і до p на фактор впевненості від p до j. Фактор впевненості і та p уже обчислений і має очікуване значення, це гарантує топологічний порядок циклів. У разі, якщо уже існує зв'язок між вершинами і та j, кінцевий коефіцієнт впевненості знаходиться за формулою паралельного множення. Тобто для випадку коли від і до j існує певне значення фактору впевненості r_{ij0}, то буде використано формулу для паралельного множення нового шляху від і до j - r_{ij1} $r_{ij} = r_{ij0} + r_{ij1} - r_{ij0}r_{ij1}$. Після отриманного значення функція записує фактор впевненості у структуру Closure, що зберігає транзитивні звязки для вершин. Після збереження зв’язку функція закінчує свою роботу.

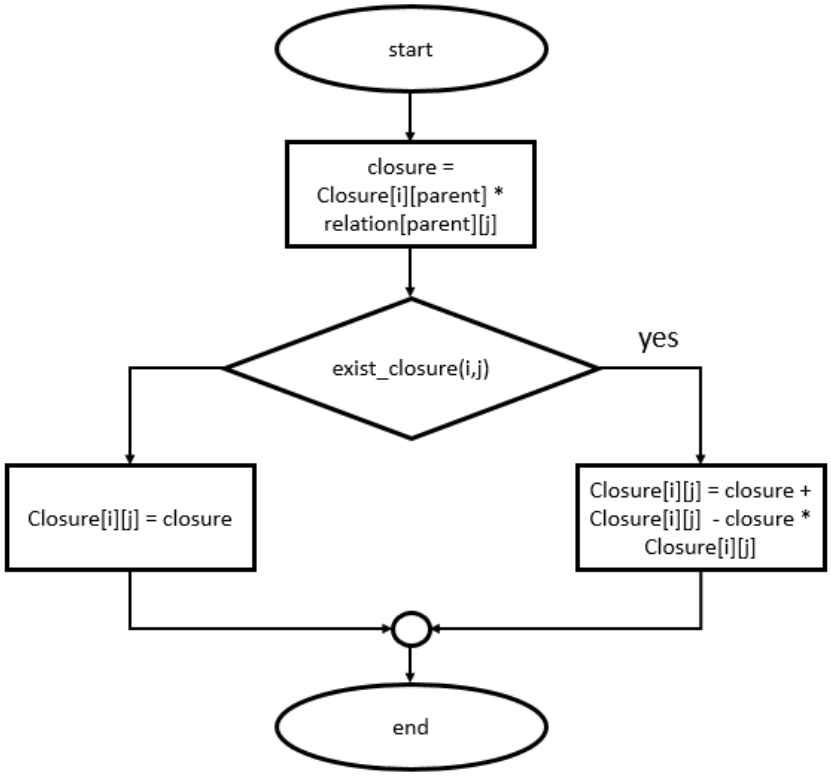


Рисунок 2.4 – Функція збереження звязку

Для кращої візуалізації роздивимось алгоритм на прикладі графу, зображеного на рисунку 2.5.

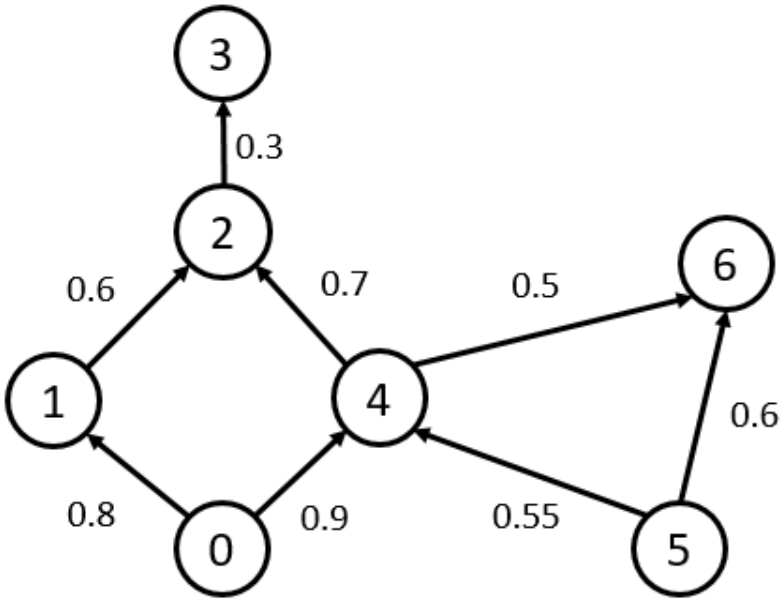


Рисунок 2.5 – Приклад орієнтованого графу

Як видно з рисунку, граф складається з 11 вершин, пронумерованих від 0 до 11. Вершини 0 та 6 є корінними, і не мають батьківських елементів. Також, згідно до вимог топологічного сортування, граф є ациклічним.

Після топологічного сортування отримаємо список зображений на рисунку

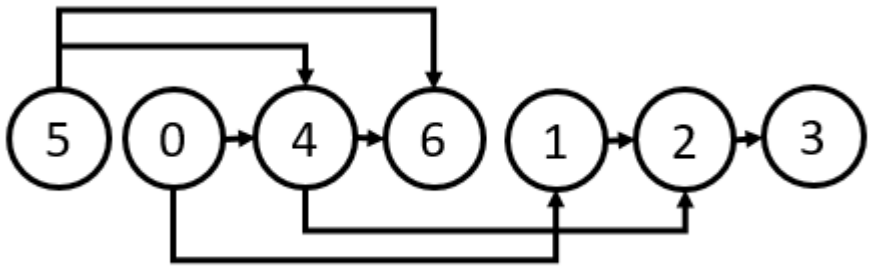


Рисунок 2.6 Топологічно відсортовані вершини.

Отриманий топологічно відсортований список вершин графу будемо використовувати під час подальших обчислень.

Роздивимось процес пошуку транзитивного замикання та фактору впевненості між вершинами 0 та 2. Перше значення буде знайдено на ітерації i = 0, j = 2, p = 1.

Тобто під час пошуку значення коефіцієнту через батьківську вершину 1 (рисунок 2.7).

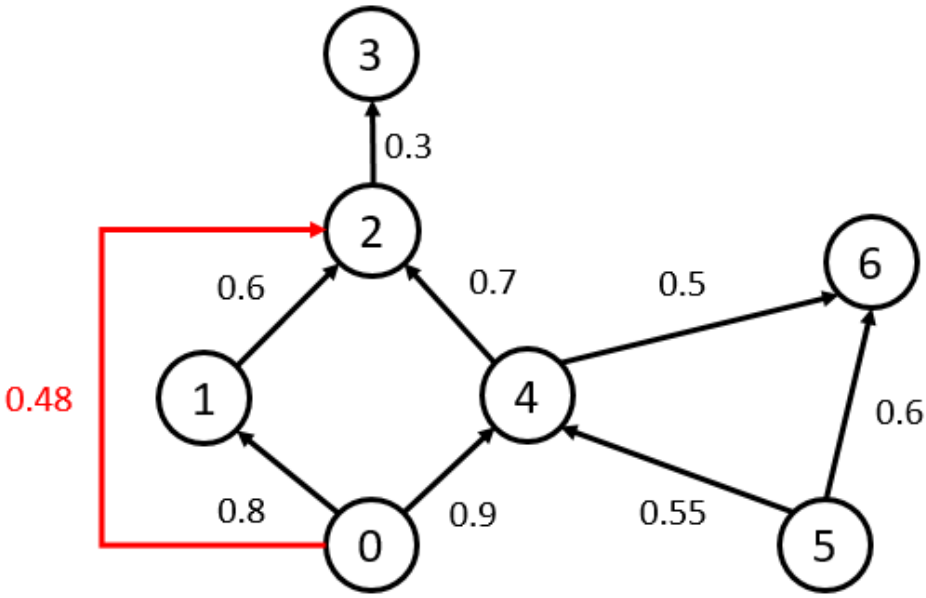


Рисунок 2.7 – Знайдено шлях через батьківську вершину 1

Під час ітерації i = 0, j = 2, p = 4 буде знайдено альтернативний маршрут через вершину 4 (рисунок 2.8).

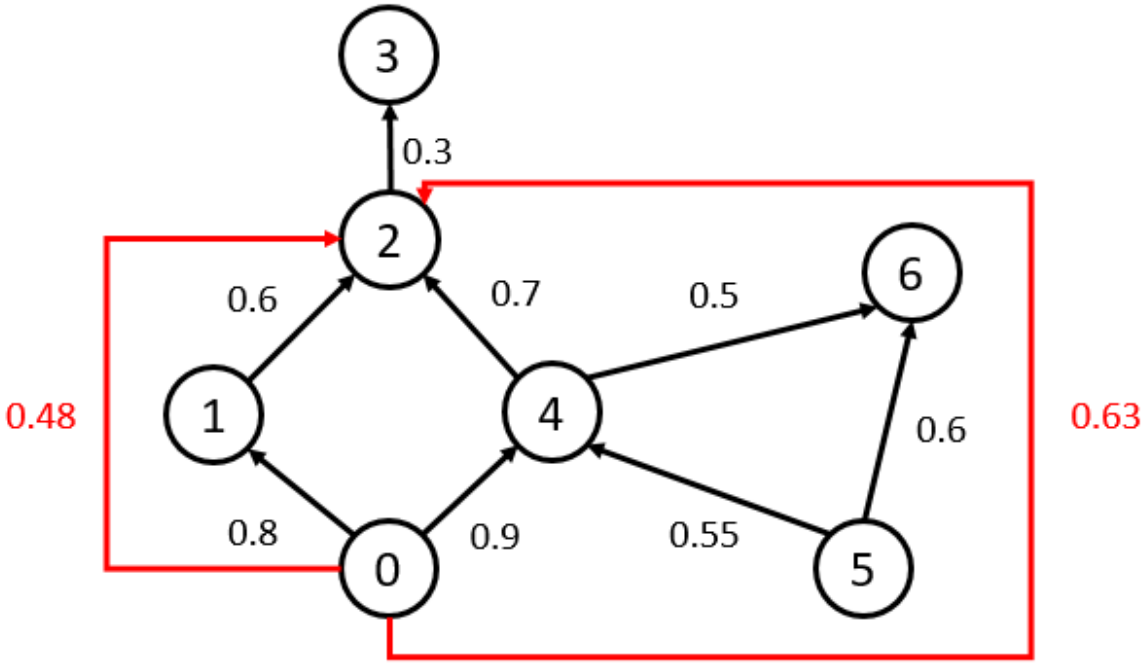


Рисунок 2.8 – Знайдено альтернативний шлях через вершину 4

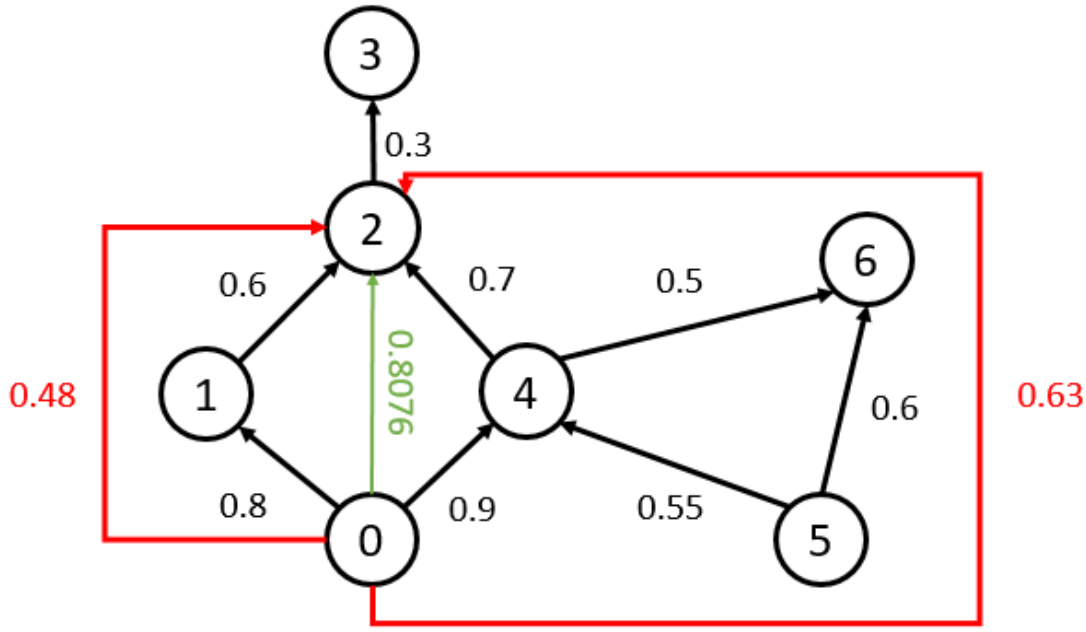


Рисунок 2.9 – Фінальне значення фактору впененості між вершинами 0 та 2

Під час збереження нового значення коефіцієнту впевненості між вершинами 0 та 2 буде виявлено старе значення, через вершину 1. Тому буде застосована формула паралельного обчислення фактору впевненості і знайдене нове значення фактору для вершин 0 та 2 – 0.8076.

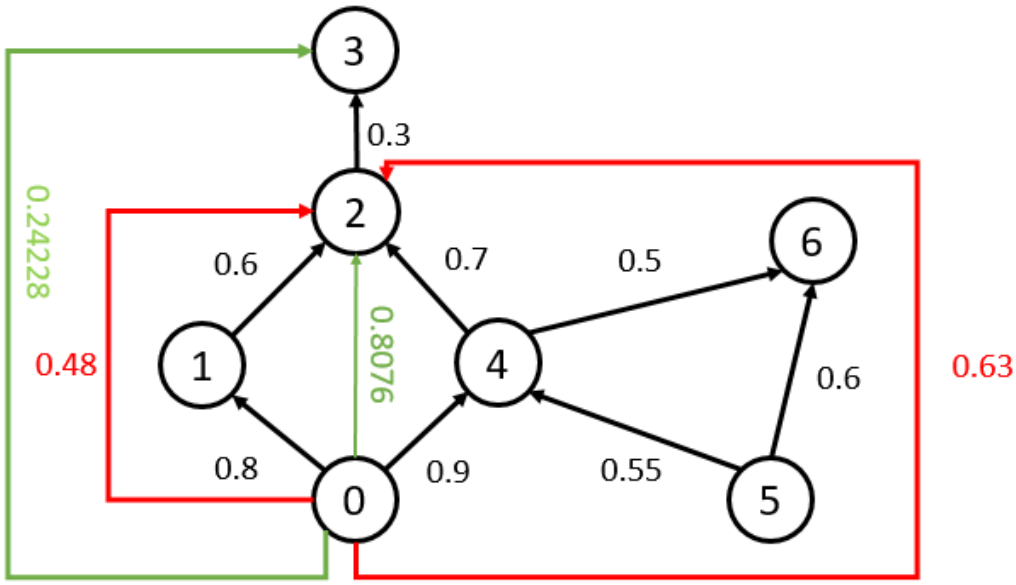


Рисунок 2.10 – Значення фактору впевненості між вершинами 0 та 3

Під час ітерації i = 0, j =3, p = 2 буде знайдено фактор впевненості між вершинами 0 та 3. Топологічний порядок вершин у циклі гарантує, що при збереженні відстані між 0 та 3 буде використовуватись корректна і обчислена відстань між 0 та 2. Отже за правилом послідовного знаходження відстані отримаємо коефіцієнт впевненості між вершинами 0 та 3 – 0.24228. Інші коефіцієнти можливо знайти аналогічним чином.

Складність алгоритму можливо легко знайти аналітичним шляхом. Процес перевірки можливості зв'язку між батьківським елементом p до елемента i та процедура встановлення зв'язку setCohesion має константну складність O(C). Цикл для усіх батьків parent(j) має складність O(p_count), де p_count – це кількість батьківських елементів вершини j. Оскільки цикл j відбувається від вершини i + 1 цикл має складність O($\frac{V}{2}$), де V – кількість вершин графу. Зовнішній цикл i очевидно має складність O(V), оскільки проходить по усім вершинам графу. Отже маємо вартість алгоритму O($\frac{V^2}{2}$ p_count). Значенням p_count можливо знехтувати, оскільки воно на порядок менше ніж значення V та не впливає на швидкість роботи алгоритму у разі додання нової вершини або зв'язку. Отже кінцева вартість алгоритму - O($\frac{V^2}{2}$).

2.3 Алгоритм оновлення транзитивного замикання

Отже, як було зазначено у завданні до магістерської роботи, необхідно було також розробити алгоритм часткового оновлення транзитивного замикання. Розроблений алгоритм працює за тим же принципом, що і алгоритм транзитивного замикання, проте перебудовує тільки ту частину графу транзитивного замикання, що необхідно змінити після редагування зв'язку між певними поняттями. Візуалізацію роботи алгоритму можливо побачити на рисунку

Позначимо вершини між якими додали зв'язок, або змінили коефіцієнт впевненості для існуючого V_1та V_2. Де вершина V_1 є батьківською, а вершина V_2 є дочірньою.

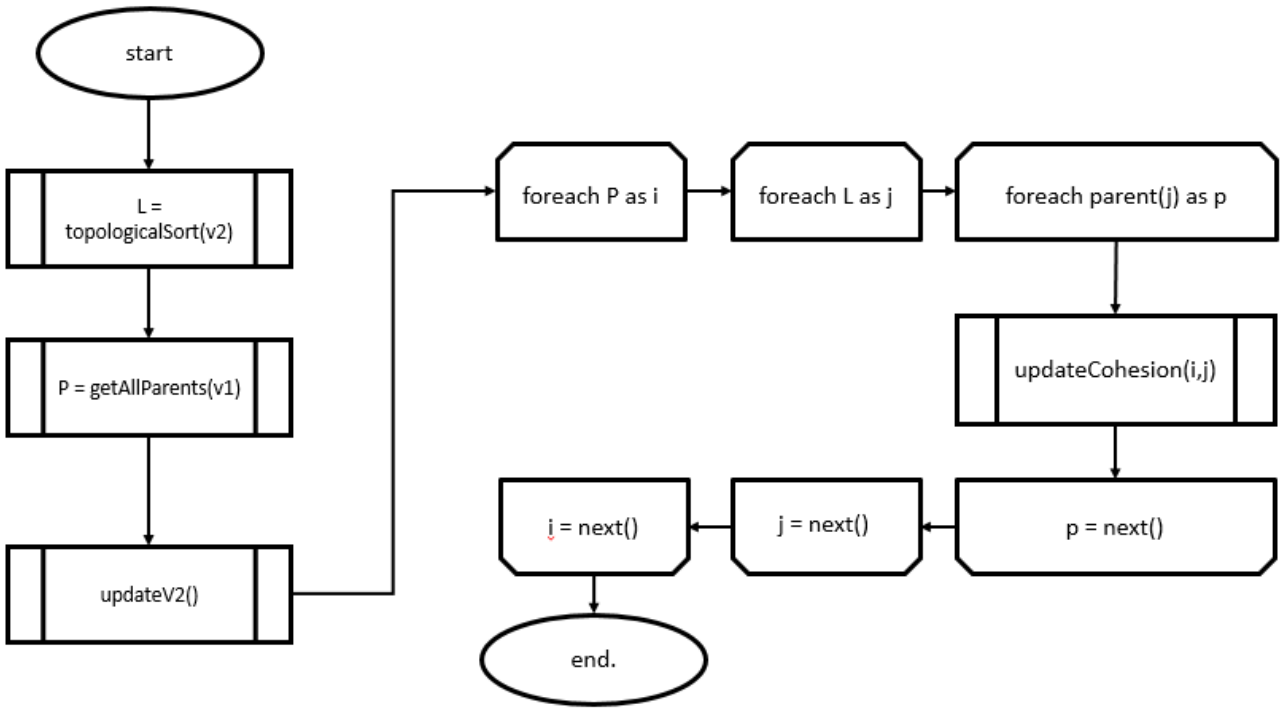


Рисунок 2.11 – Візуалізаця роботи алгоритму оновлення ТЗ

Для знаходження зв’язків, що необхідно перебудувати після операції над вершинами V_1та V_2. необхідно:

– знайти усі батьківські елементи вершини V_1та включити в цей список вершину V_1;

– знайти топологічно відсортований список вершин підграфу, корінним елементом якого є вершина V_2.

Приклад можливо побачити на рисунку 2.12, де було додано нове ребро від вершини 4 до вершини 1. Жовтим кольором відображені топологічно відсортований список вершин підграфу V_2. Синім кольором відображені усі батьківські вершини V_1.

Після знаходження списків вершин необхідно оновити значення, що з’являться після появи нового ребра, що виконує функція updateV2(). Принцип роботи функції можливо побачити на рисунку 2.13. Першим чином у оновленому транзитивному замиканні зберігається зв'язок між вершинами V_1та V_2. Після цього додається новий зязок для усіх батьківських веришин у списку P.

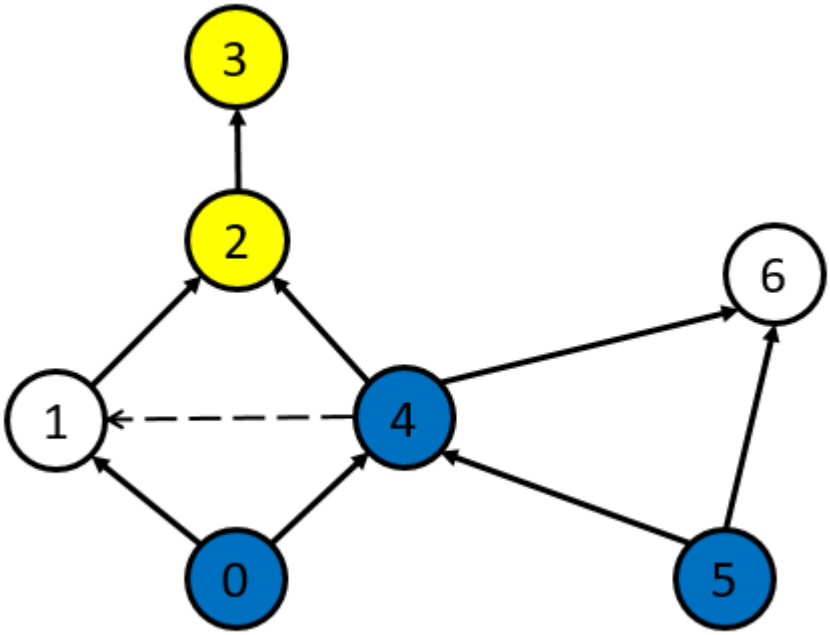


Рисунок 2.12 Знайдені вершини, зв’язки яких необхідно перебудувати

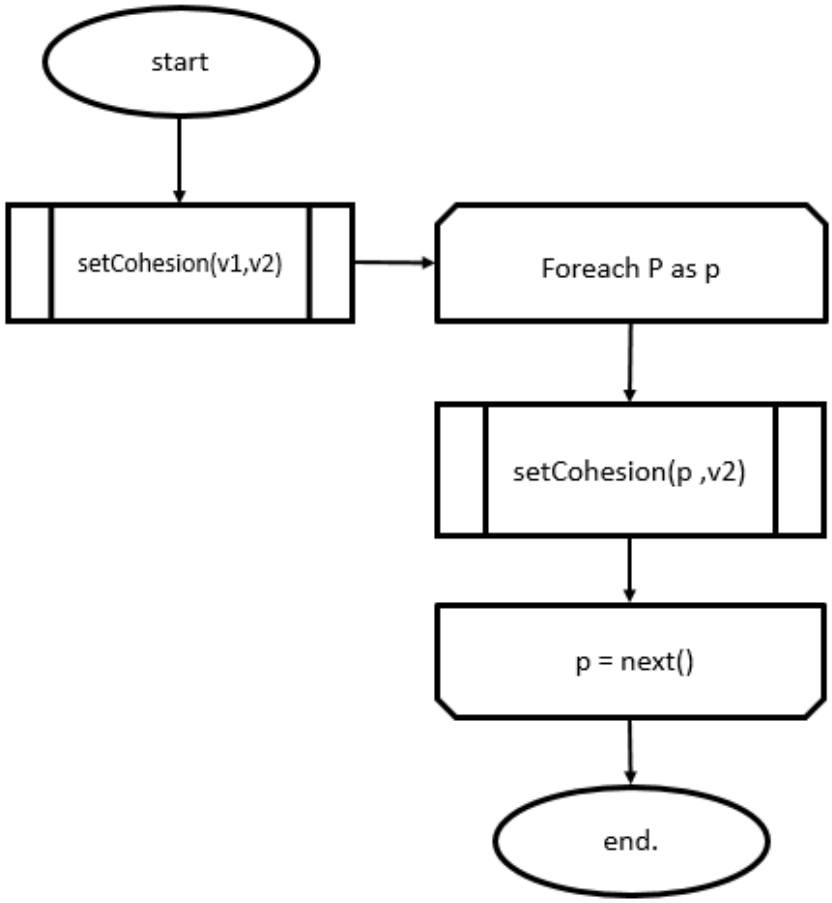


Рисунок 2.13 – Робота фукнції updateV2()

Після цього аналогічним чином викликаються цикли як у алгоритмі транзитивного замикання, проте існує певна відмінність – цикл з лічильником і проходить по знайденому списку батьківських вершин P а цикл з лічильником j проходить по топологічно відсортованому списку вершин підграфу, що утворюється з вершини V_1. Внутрішній цикл по батькам вершини j працює схожим чином, тільки у випадку знаходження звязку не додає новий зв'язок за допомогою функції setCohesion(i, j), а викликає функцію updateCohesion(i, j). Принцип роботи функції можливо побачити на рисунку.

На початку роботи функції іде перевірка чи існував транзитивний зв’язок через батьківську вершину p до додання нового ребра у графі. Якщо зв’язок існував, то необхідно видалити старе значення коефіцієнту впевненості за допомогою функцій removeClosure(i,j). Після цього у будь-якому випадку нове значення коефіцієнту необхідно записати за допомогою уже відомої функції setClosure(i,j).

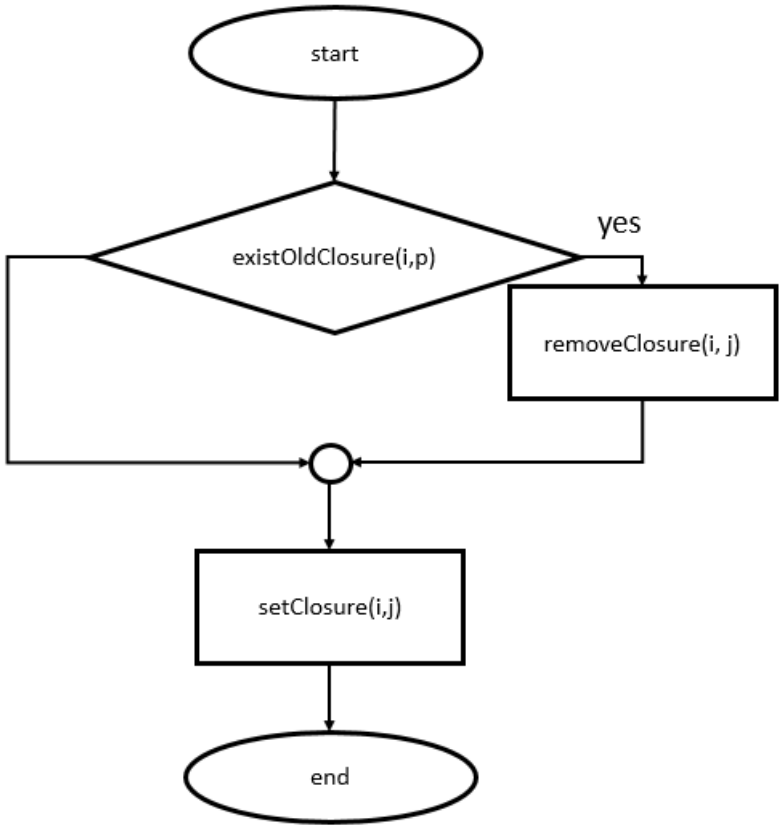


Рисунок 2.14 – Робота функції updateChild()

Для видалення старого звязку використовується формула обернена до формули паралельного знаходження коефіцієнту впевненості (рисунок 2.15).

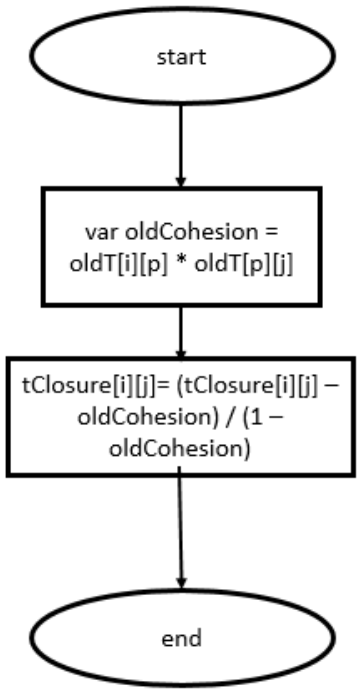


Рисунок 2.15 – Робота функції remveCohesion()

Алгоритм роботи функції для додавання нового зв’язку можливо побачити на рисунку 2.16.

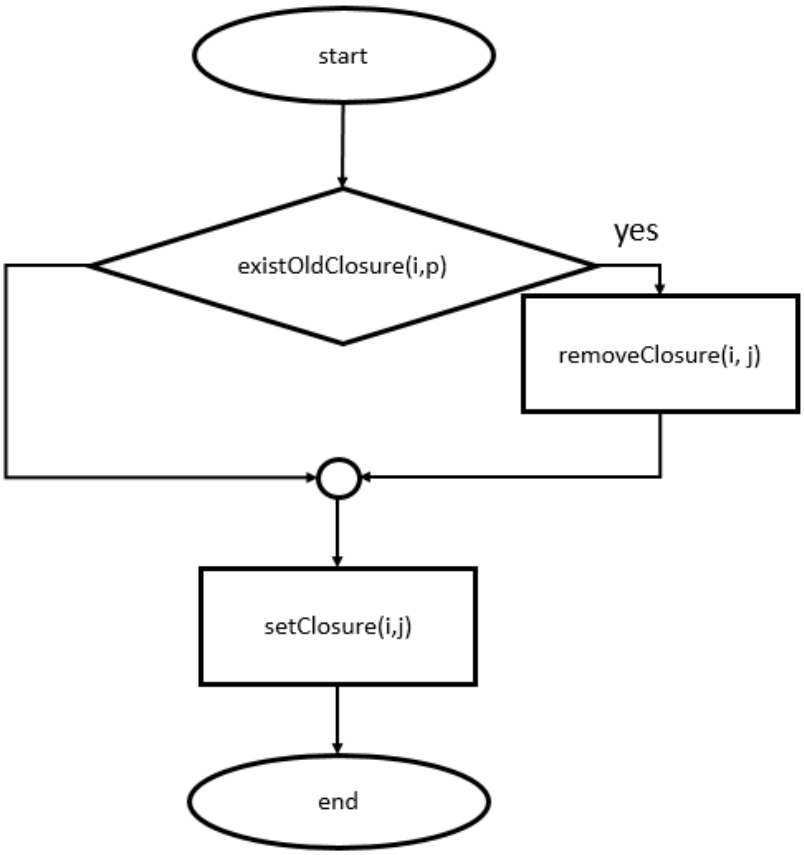


Рисунок 2.16 – Робота функції збереження звязку

Очевидно, що складність алгоритма має буде нижче ніж складність алгоритму транзитивного замикання, але мати такий же порядок, оскільки кількість циклів таж сама проте кількість ітерацій циклів зменшена. Між циклом і та j існує певна залежність. Кількість дочірніх елементів V_2 обернено пропорційна кількісті батьківських елементів V_1. А це значить, що у найгіршому випадку будемо мати половину графу дочірніх елементів та половину графу батьківських. Тобто складність зовнішнього циклу буде O($\frac{V^2}{2}$), аналогічним чином складність внутрішнього циклу j буде O($\frac{V^2}{2}$). Отже аналогічно до пошуку складності алгоритму створення транзитивного замикання можемо зробити висновок що кінцева складність буде O($\frac{V^2}{4}$). Проте на практиці алгоритм працює значно швидше.

Висновки до розділу 2

При детальному розгляді усіх нюансів побудови транзитивного замикання для онтологічної системи виникають певні проблеми. По-перше, це неправильний результат обчислення коефіцієнтів впевненості у випадках, коли вершини та зв'язки додаються у довільному порядку. Саме це можливо побачити під час роботи алгоритму Флойда-Уоршала, що було продемонстровано у даному розділі. Також важливою проблемою є висока складність алгоритмів, що вимагає значних ресурсів при роботі з великою кількістю даних для системи.

Для вирішення даних проблем та забезпечення коректної та швидкої роботи було розроблено 2 алгоритми:

- створення транзитивного замикання за час O($\frac{V^2}{2}$);
- алгоритм оновлення транзитивного замикання O($\frac{V^2}{4}$).

Основна ідея розроблених алгоритмів – використання топологічного сортування перед роботою зі списком вершин.

3. ОПИС ВИКОРИСТАНИХ ЗАСОБІВ ТА ТЕХНОЛОГІЙ

Під час реалізації системи роботи з транзитивним замиканням було використано ряд технологій. Дослідження та тестування було виконано на мові java за допомогою технології junit, оскільки це дозволяє запускати програмний код без використання сервера. Для розробки серверної частини було використано мову програмування PHP.

3.1 Мова програмування Java

Java – це сильно типізована мова програмуваннння з підтримкою об’єктно оріентованої моделі. Мова була розроблена компанією Sub Microsystems, а наразі передана компанії Oracle. Унікальність рішення під час розробки Java, є використання проміжного байт-коду що є перехідною ланкою між бінарним кодом та скриптом. Байт код виконується за домопогою Java Virtual Machine, або JVM[9].

Перевагою такого виконання коду є повна незалежність байт-коду від операційної системи, або обладнання, що дозволяє виконувати програми, написані на Java на будь-якому пристрої, для якого існує віртуальна машина Java. Іншою важливою особливістю мови є гнучка система безпеки: виконувана програма повністю контролюється за допомогою віртуальної машини. Операції, що порушуть дозволені повноваження, наприклад, несанкціонована спроба доступу до даних призводять до преривання роботи програми без затримки.

Дуже часто, вважають знижену продуктивність недоліком. Проте за час підтримки мови розробники створили ряд покрщень для збільшення швидкості роботи програм:

− технолонія компіляції байт-коду у машинний прямо під час роботи програми під назвою Just-In-Time compilation;

– використання native коду, або платформо-орієнтованого коду у стандартних бібліотеках;

– прискорення виконання коду за рахунок технологій, що вбудовані у апаратну середу.

3.2 Технологія Junit

JUnit - це технологія, що була розроблена для тестування програм, написаних з використанням технології Java. Фреймворк використовує в своїй основі TDD Test-Driven Development і входить в сімейство технологій для тестування xUnit.

Головна ідея даного підходу Test Driven Development - "спочатку тести, потім код". Це означає, що спочатку необхідно визначити, що повинно бути результатом роботи певної частини коду і написати тести, які перевіряють ідентичність результату з очікуваним, після чого вже розроблюється ця частина коду, яка у подальшому буде тестуватись. Даний підхід збільшує ефективність роботи розробника і дозволяє писати більш стабільний код. В результаті цього ми отримуємо меншу кількість часу, який буде витрачено на написання стабільно працюючого коду.

Фреймворк має такі властивості:

– відкритий вхідний код;

– простий у використанні;

– підтримує анотації для ідентифікації методів;

– підтримує затвердження для тестування отриманих результатів;

– тести можуть бути організовані в зв'язки тестів (test suites);

– має візуальну індикацію стану.

Тестовий випадок (Test Case) в юніт тестуванні - це частина коду, яка перевіряє, що інша частина коду (зокрема - метод) працює відповідно до певних вимог. Формально описаний тестовий випадок характеризується відомими вхідними даними і очікуваним висновком програми, який відомий до початку виконання тесту.

Необхідно створювати, як мінімум, два тестових випадків для кожної вимоги - позитивний і негативний[14].

3.3 СУБД MySQL

MySQL – це реляційна СУБД, що може використовуватись на безкоштовній основі. За розробку та підтримку СУБД MySQL відповідає компанія Oracle. Компанія отримимала права на систему пілся поглинання компанії Sun Microsystems. Як вже було указано субд повністю безкоштовна і поширюється за ліцензією GNU General Public License.

MySQL зачасту використовується для рішень для малого та середнього маштабу систем. Зазвичай СУБД використовується як сервер, до якого звертаються клієнти, як локальні так і віддалені. Також у дистрибутив включено бібліотеку внутрішнього сервера, що дозволяє включати MySQL у автономні застосунки.

СУБД підтримує велику кількість різних типів таблиць. Це можуть бути таблиці типу MyISAM, які підтримують повнотекстовий пошук. Також можуть будти таблиці InnoDB та EXAMPLE. Перший тип дозволяє використовувати транзакції для окремих записів у таблиці, а другий тип дозволяє створювати свої типи на основі прикладу.

Спочатку СУБД була створена для роботи з веб мовою PHP, але згодом, після знаходження підтримки шиорокої публіки у IT стала використовуватись для багатьох мов програмування, таких як:

- C/C++
- C#
- Java
- PHP
- Python
- Ruby

Для роботи з СУБД існує удобна версія десктоп клієнта, що дозволяє виконувати усі операції, що і консольний клієнт але у зручному форматі з візуалізацією функцій та результатом їх роботи.

На даний момент актуальна версія є MySQL 8.0 у якій було покращено продуктивність роботи системи та додано нові можливості для роботи.

3.4 Мова програмування PHP

PHP – це скриптова мова, що була розроблена для веб систем, що справедливо є самою популярною у даній сфері. Язик має слабу, динаминічну типізацію, але у версії 7 наявна можливість явно вказувати очікуваний тип.

Містить усі необхідні базові інструменти та механізми для розробки веб застосунків:

- автоматичне доставання параметрів для POST, GET запитів, змінних середи веб серверу;
- підтримка XForms;
- роботи з файлами;
- cookies і sessions;
- роботи з HTTP заголовками;
- підтримка веб сокетів.

PHP підтримує більшість популярних баз даних:

- MySQL;
- MySQLi;
- SQLite;
- PostgreSQL;
- Oracle;
- MS SQL;
- Sybase;
- Sybase;
- ODBC;
- mSQL;
- Apache Derby.

Під час роботи програми PHP скрипт оброблюється інтерпретатором, що передбачає 4 етапи. Інтерпретатор генерує байт код, але без створення виконуючого файлу, що дозволяє пришвидшити роботу програми і забезпечити безпеку від певних видів атак[4].

3.5 Мова програмування JavaScript

JavaScript – скриптова мова для програмування клієнтської частини веб системи. Об'еднує у собі імперативний, об'єктно-рієнтований та функціональний стилі, тобто є мультипарадигмовою мовою. Зазвичай виконується у браузері, що дозволяє створити інтерактивні веб сторінки.

Об'єкто-орієнтований стиль забезпечується за рахунок прототивного програмування. Це передбачає відстутність класів, як таких і наявність об'єктів що задають базову структуру для усіх об'єктів, що є дочірними.

JavaScript є С подібною мовою, проте існують деякі відмінності від мови С:

- функції є об'єктами першого класу
- автоматичне приведення типів
- анонімні функції
- інтроспекція
- автоматичне вивільнення пам'яті.

Оскільки застосунки, написані на мові JavaScript мають виконуватись на стороні кліента існує ряд вимог до безпеки мови. Це зумовлює відсутність таких звичних речей як:

- доступ до файлової системи;
- управління потоками ввода-виводу;
- типи для бінарних даних;
- інтерфейси до веб-серверів;
- інтерфейси до баз даних;
- система керування пакетами.

Для доступа до властивостей сторінки використовується об'єктна модель браузера. Це – арі браузера, що яляє собою посередником між розробленим JavaScript кодом та моделью документу. На даний момент не існує єдиного стандарту для створення об'єктної браузера.

Об'єктна модель документа – це інтерфейс, що дозволяє працювати з сторінкою як з об'єктом, а саме забезпечує CRUD операції для роботи з вузлами сторінки. [20]

4. ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ КЕРУВАННЯ ТРАНЗИТИВНИМ ЗАМИКАННЯМ

Програмна система для роботи з транзитивними звязками створена за допомогою мови програмування PHP. Кліентська частина реалізована за допомогою технологій HTML, CSS та Javascript. Для збереження даних було використано СУБД MySQL. Для тестування та аналізу роботи алгоритмів було використано проект на мові програмування Java.

4.1 Проект для аналізу та тестування алгоритмів

Проект для тестування складаєься з 5 класів (рисунок 4.1) та 5 тестів, що покривають дані класи. Тести створенні як для алгоритмів, які розроблялися на протязі виконання магістерської роботи, так і для компонентів що використовувались у алгоритмах. Це гарантує корректність роботи і достовірність пройдених тестів.

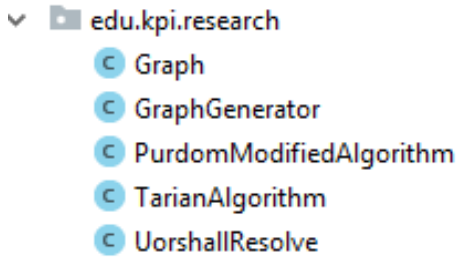


Рисунок 4.1 – Класи проекту для тестування алгоритмів

Класс Graph – це класс, що зберігає інформаці про граф та містить деякі операції над даними графа, що будуть необхідні під час роботи розроблюваних алгоритмів.

Класс GraphGenerator містить логіку по створенню випадкових даних. Такий підхід збільшує вірогідність знаходження помилки у випадку, якщо вона існує.

Класс UorhallResolve містить логіку створення транзитивного замикання ща алгоритмом Флойда-Уоршала. У процесі аналізу було доведено неефективність

даного алгоритму.

Класс TarianAlgorithm містить логіку топологічного сортування, результати якого використовуються у розроблених алгоритмах.

Класс PudromModifiedAlgorithm містить логіку розроблюваних алгоритмів, тобто алгоритму транзитивного замикання та алгоритм оновлення транзитивного замикання.

Тести-класи, що наявні у системі тестування алгоримів можливо побачити на рисунку 4.2.

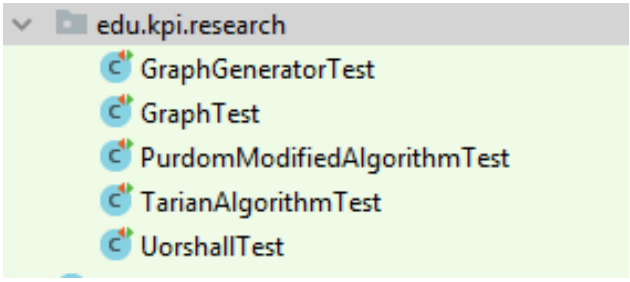


Рисунок 4.2 – Тест-класи проекту для тестування алгоритмів

Як можливо побачити тести покривають усі класи. Більш того, для забезпеченння максимальної коректності досліджень – тести покривають 100% коду. Тільки такий підхід може гарантувати повну відсутність помилок у коді.

Самим важливим тестом класом є PudromModifiedAlgorithmTest. Саме він тестує логіку розроблюваних алгоритмів. Клас містить 4 тести:

- bughananShouldWork – перевіряє логіку роботи паралельного та послідовного множення під час створення ТЗ;
- createTransitiveClosure – перевіряє логіку роботи алгоритму створення ТЗ;
- updateTransitiveClosure_shouldUpdateV2 – перевіряє логіку роботи алгоритму оновлення транзитивного замикання під час обробки дочірньої вершини V_2;
- updateTransitiveClosure – перевіряє логіку роботи алгоритму оновлення транзитивного замикання.

Результат роботи тестів можливо побачити на рисунку 4.3.

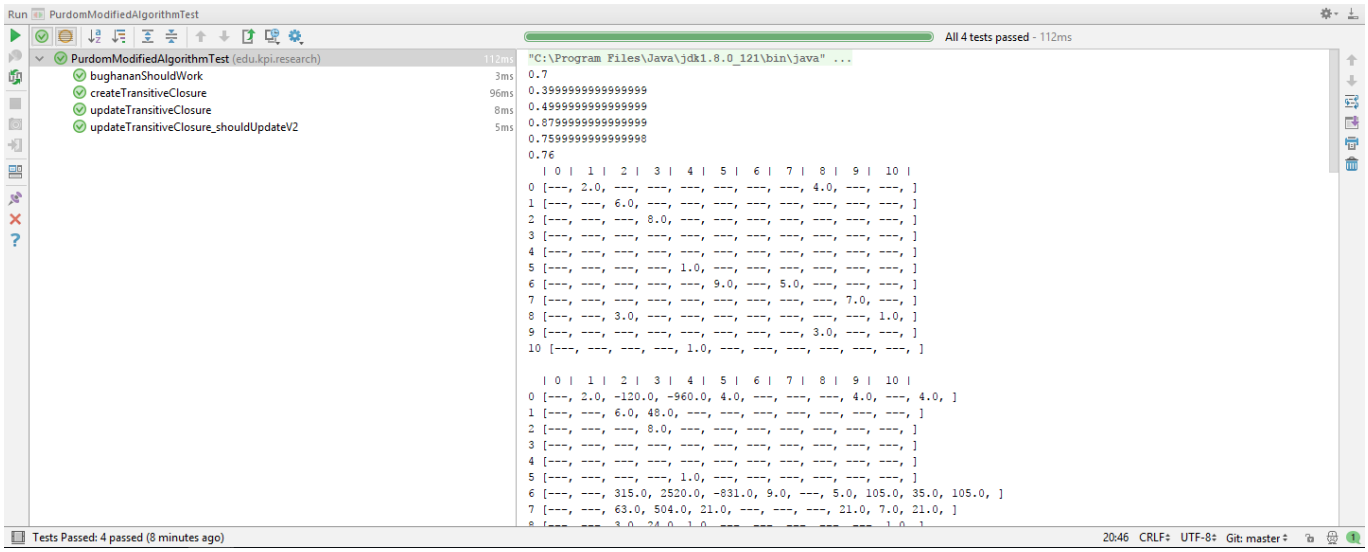


Рисунок 4.3 – Результат роботи тестів

4.2 Структура проекту системи побудови та використання ТЗ

Проект складається з логічно розділених каталогів, що містять файли для використання для різних цілей (рисунок 4.4) . Проект містить такі каталоги:

- classes;
- configuration
- script;
- style;
- корінний каталог.

Корінний каталог містить тільки ті файли, що містять логіку виводу створення html документів. Тобто backend код що виводить інформацію користувачу. Деякі файли є самостійними обробниками сторінок, інші включаються у інші файли.

Каталог classes містить php класи, що відповідає за роботу з моделю. Це зв'язок з базою даних, алгоритми, структури даних, що необхідні для роботи з даними алгоритмами.

Каталог script містить файли у яких зберігається javascript дані, або дані, що необхідно включити у javascript код.

Каталог style містить css файли що відповідні за налаштування CSS стилю для виводу інформації користувачу.

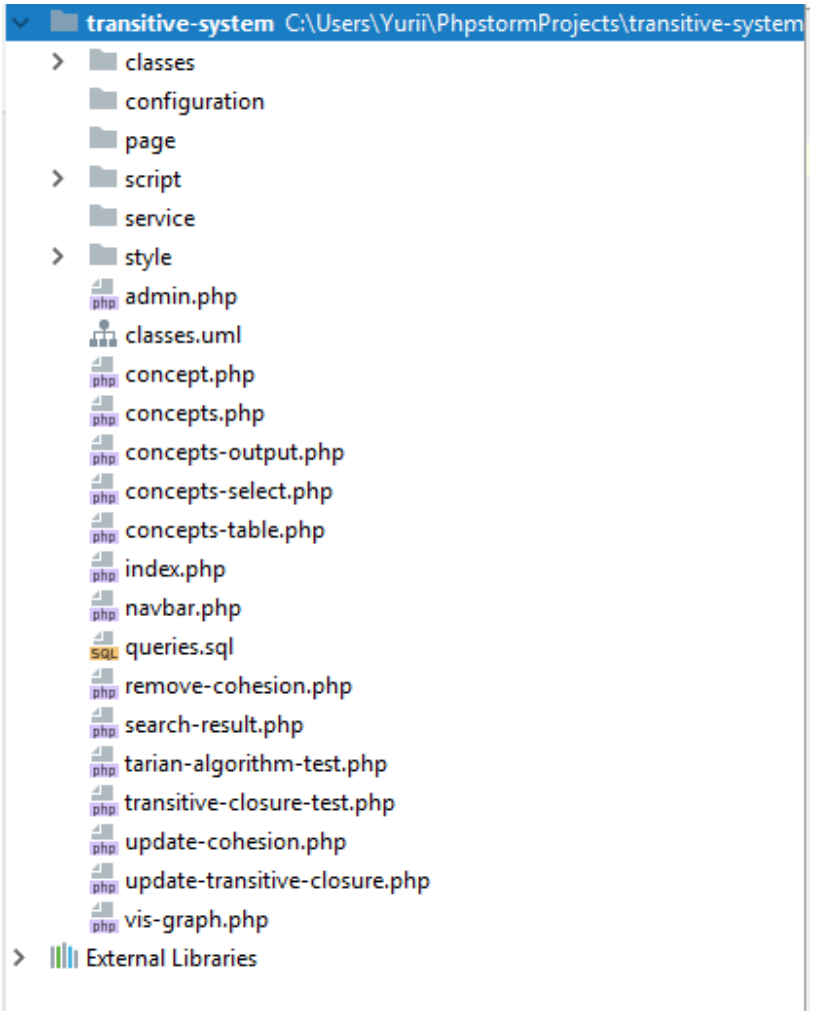


Рисунок 4.4 – Структура проекту

4.2 Структура бази даних

База даних для роботи системи містить 3 таблиці (рисунок 4.5). Таблиця Concept (Концепт) мітить дані про поняття в онтологічній системі. Вона містить два поля: id – первиний ключ таблиці та name – назва поняття. Для забезпечення звязку багато до багатьох використовуються 2 таблиці – ctoc (Звязок) та cuptoc (Транзитивний звязок). Ці таблиці містять поля:

- id – первиний ключ;
- from_id (ідБатька) – зовнішній ключ батька концепту;
- to_id (ідДочірнього) – зовнішній ключ дочірнього концепту;

- CF (коефіцієнтВпевненості) – коефіцієнт впевненості звязку.

Різниця між таблицями – це те, що Транзитивний Зв'язок містит також транзитивні звязки між поняттями.

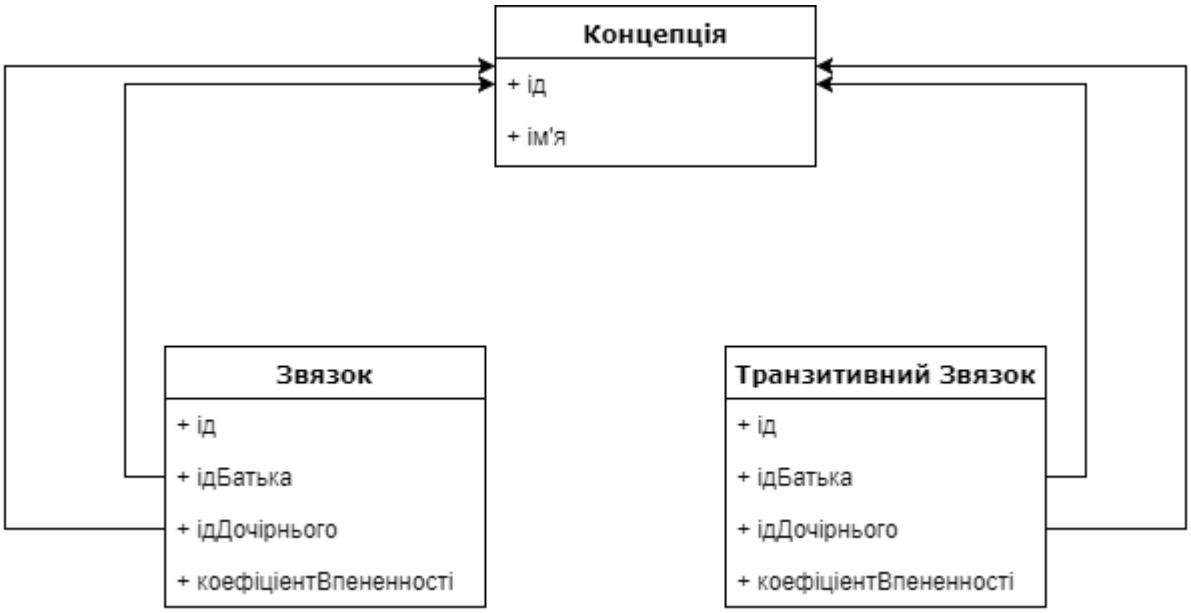


Рисунок 4.5 –Діаграма БД

4.2 Структура класів

Діаграму усіх класів системи можливо побачити на рисунку 4.6. Найважливішими класами у системі є ConceptService та TransitiveClosure. Класи Database, RawModel, SQL, Recordset відповідають за зв'язок системи з базою даних.

Клас TransitiveClosure (рисунок 4.7) містить логіку створення транзитивного замикання для понять. Для роботи з графом понять використовується клас-структура Graph. Основними методами, що використовуються у інших класах є createTransitiveClosure, що створює транзитивне замикання, для заданого графу понять та updateTransitiveClosure, що оновлює транзитивне замикання для заданого графу, старого транзитивного замикання, та вершин V_1 та V_2. Інші методи викликаються у середині самого класу при роботі алгоритмів.

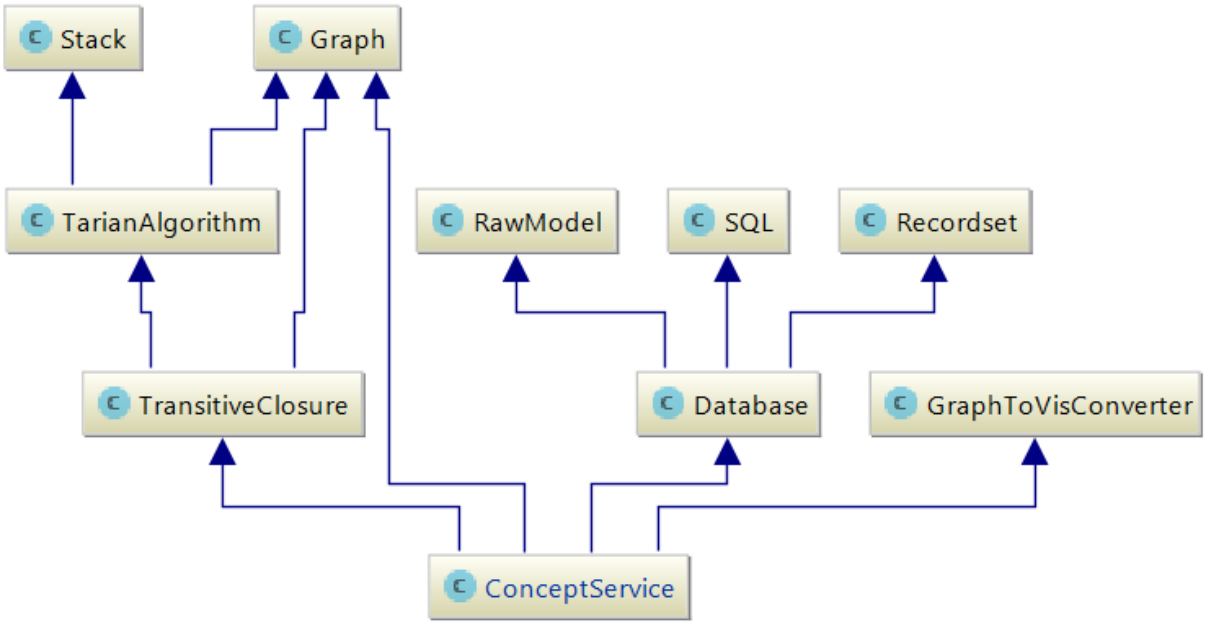


Рисунок 4.6 –Діаграма класів

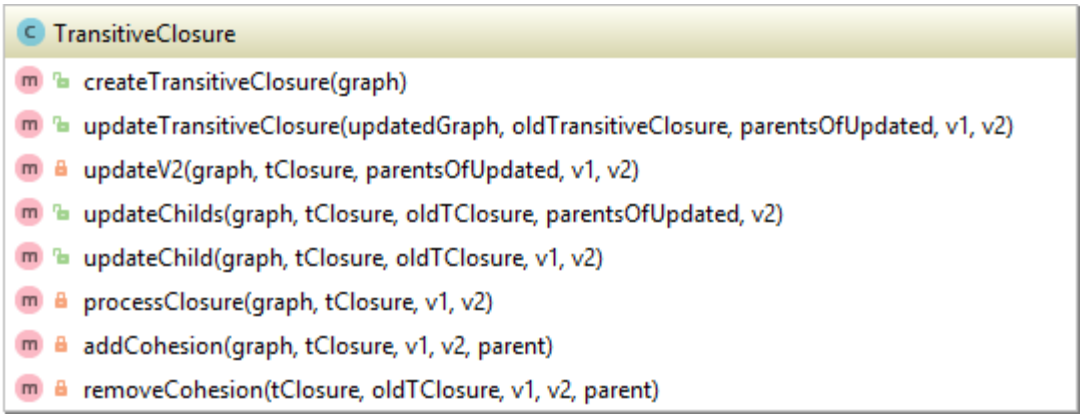


Рисунок 4.7 – Клас TransitiveClosure

Класс ConceptService використовується для виклику усієї бізнес логіки системи. Він містить CRUD методи для роботи з поняттями та транзитивним замиканням. Уся логіка запитів до бази даних також міститься у методах даного сервісу. Також сервіс містить метод для створення даних для візулізації графу на стороні клієнту. Для цього використовується клас-структура даних GraphToVisConverter, що містить логіку перетворення об’єктів типу Graph у формат, з яким зможе працювати бібліотека Vis для візуалізації графів.

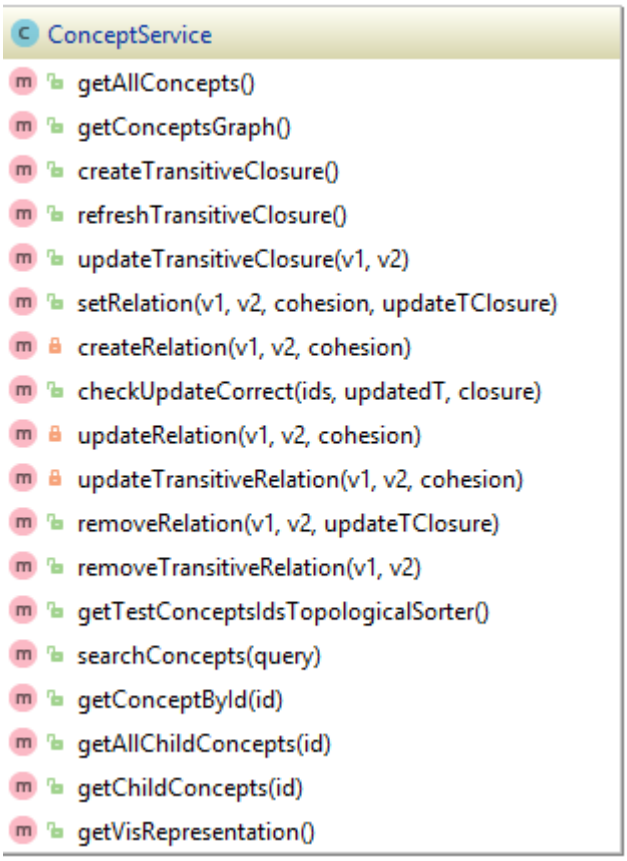


Рисунок 4.8 – Клас ConeptService

Висновки до розділу 4

Розроблений продукт відповідає вимогам, зазначеним у завданні. Система містить реалізацію розроблених алгоритмів, алгоритму побудови ТЗ, алгоритм оновлення ТЗ. Система дає можливість роботи з транзитивним замиканням. При цьому система спроектована згідно з сучасними вимогами до якісного по, тобто з використанням:

- паттернів SOLID/GRASP;
- архітектурним розподіленням системи компонентів;
- Unit тестів;
- прийомами ООП для повторного використання коду.

Також важливим є наявність окремого проекту для тестування розроблених алгоритмів.

5. ІНСТРУКЦІЯ РОБОТИ ПРОГРАМИ

Як вже було указано система має веб інтерфейс для роботи с користувачем (рисунок 5.1). Загалом інтефейс можливо поділити на такі частини:

- перегляд списку понять;
- перегляд поняття та звязків для нього;
- сторінка редагування звязків;
- сторінка перегляду візуалізованого графу.

Будь-яка сторінка системи включає у себе меню, що знаходиться зверху. Сторінка перегляду списку понять включає у себе усі поняття, або результат пошуку (рисунок 5.2).

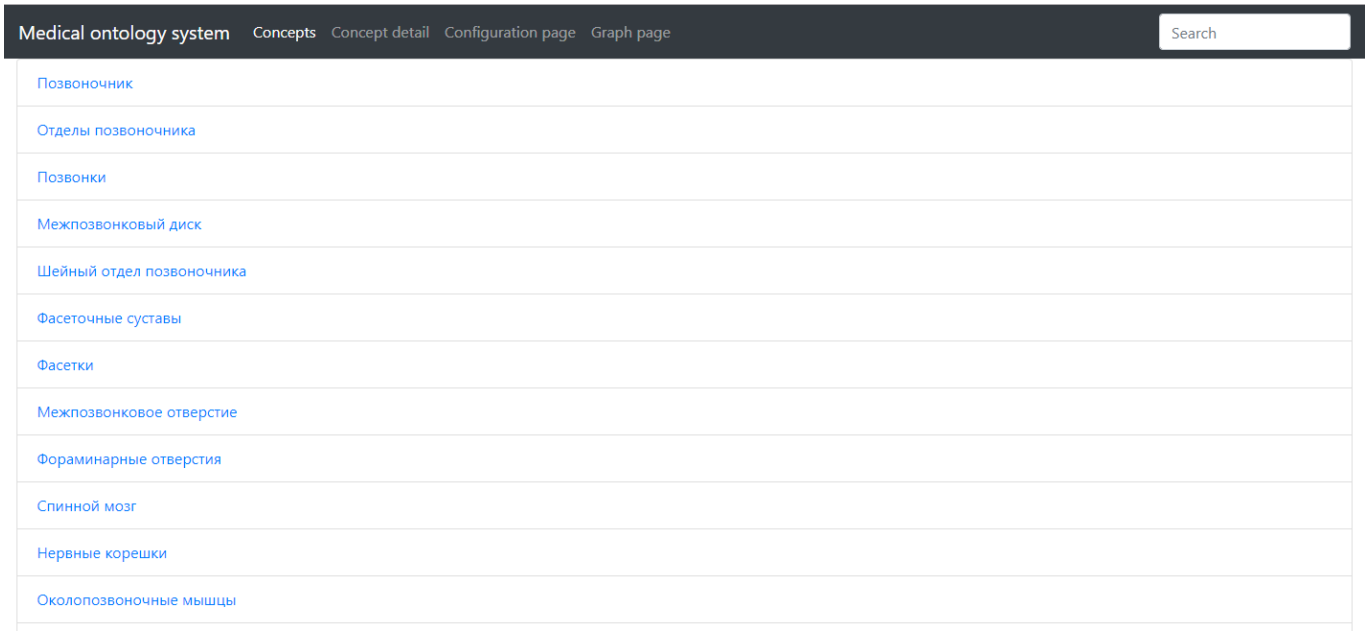


Рисунок 5.1 – Перегляд списку понять

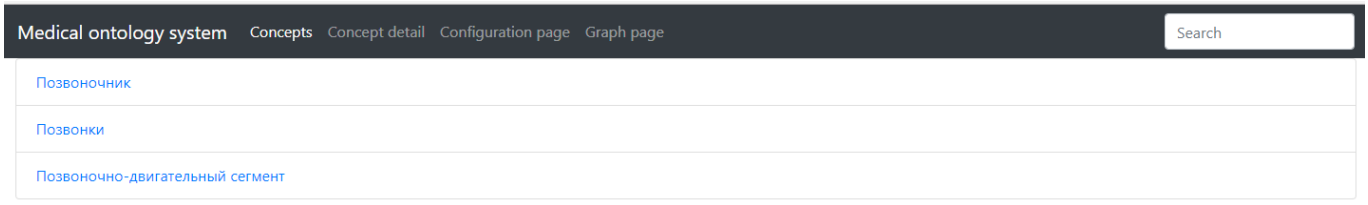


Рисунок 5.2 – Сторінка результатів пошкуку

При натиснені на поняття зі списку відбувається перехід на сторінку поняття. Сторінка містить дві таблиці:

- зв’язки, що задані явно (рисунок 5.3);
- транзитивні з’вязки, що створені за допомогою алгоритма(рисунок 5.4).

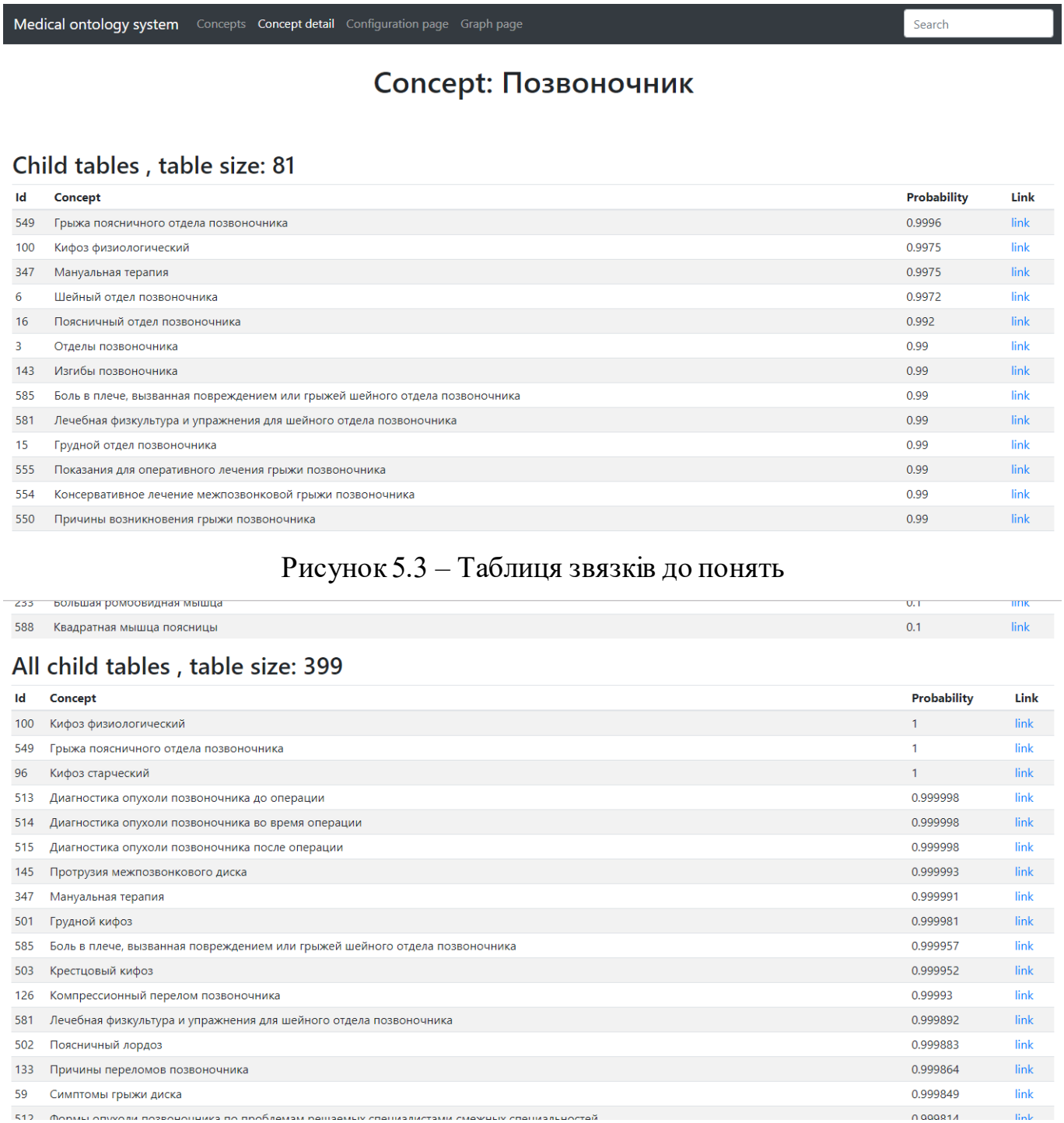


Рисунок 5.3 – Таблиця звязків до понять

Рисунок 5.4 – Таблиця транзитивних звязків до понять

При натиснені на посилання link відбувається перехід на сторінку відповідного поняття, що повністю ідентична до розглянутої раніше (рисунок 5.5).

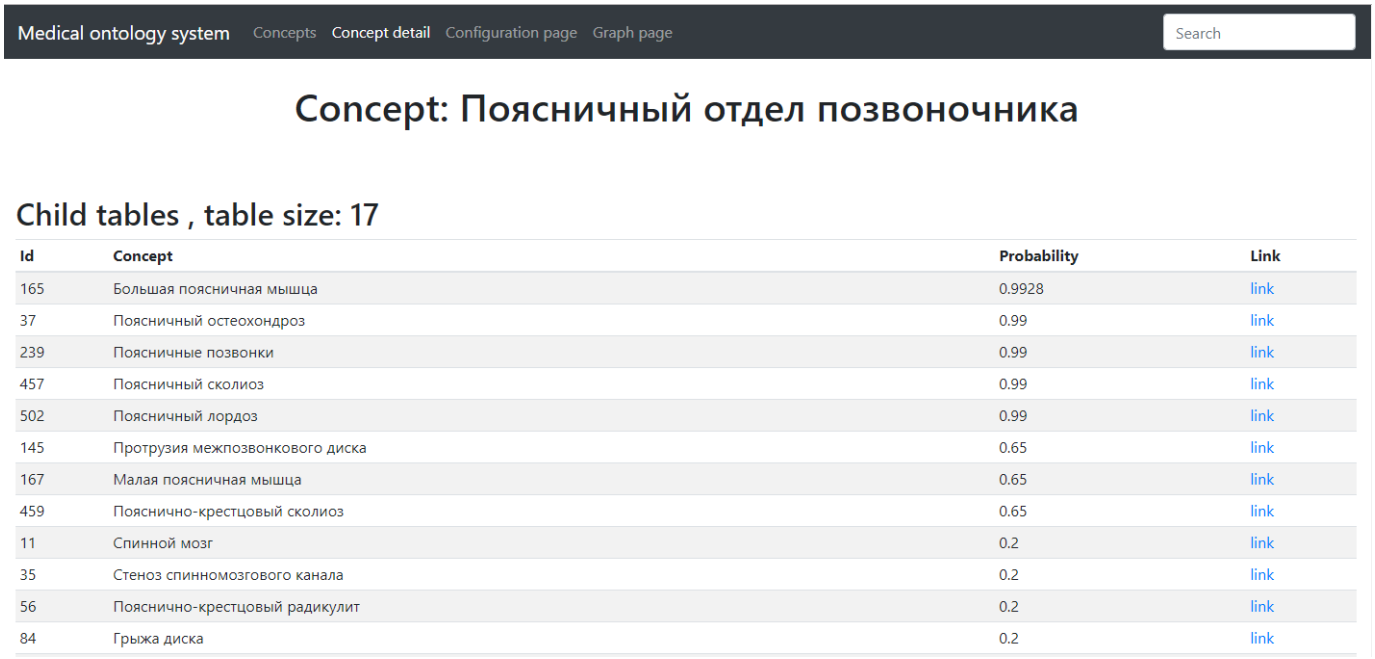


Рисунок 5.5 – Результат переходу до поняття

Сторінка редагування понять містить такі елементи керування:

- поля вибору понять та поле значення фактору впевненості;
- чекбокс автоматичного оновлення
- кнопки збереження та виделення поняття;
- кнопка ручного оновлення транзитивного замикання (рисунок 5.6).

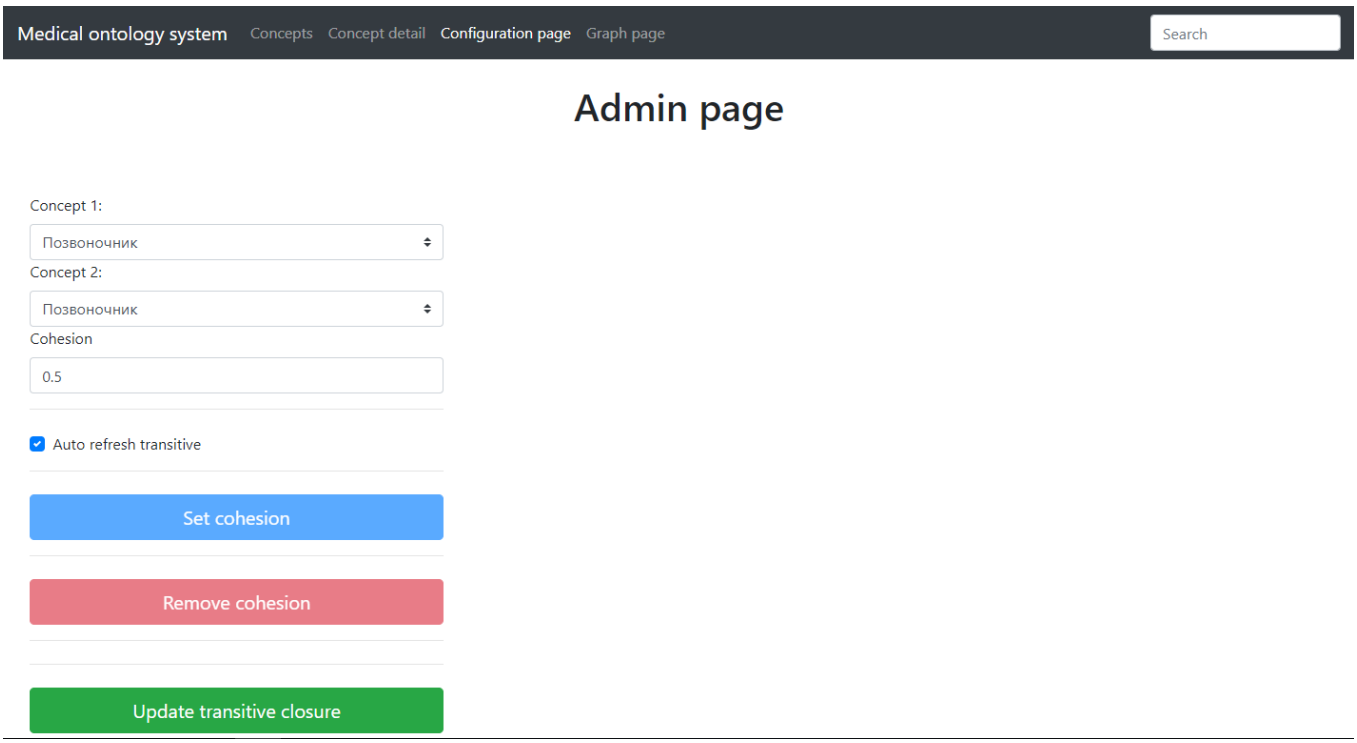


Рисунок 5.6 – Сторінка корегування звязків

Для створення нового звязку необхідно вибрати поняття у полі вибору батьківського поняття та у полі вибору дочірньої і встановити значення фактору впевненості. За замовчуванням значення будет 0.5. Для автоматичного оновлення транзитивного замикання після редагування звязків необхідно вімкнути чекбокс.

Для редагування звязку неохідно виконати аналогічні кроки. Після вибору неообхідних понять у поле фактору впевненості буде підгружено існуюче значення. Кнопка збереження звязку між поняттями змінить назву на Update (рисунок 5.8). Видалення виконується аналогічним чином.

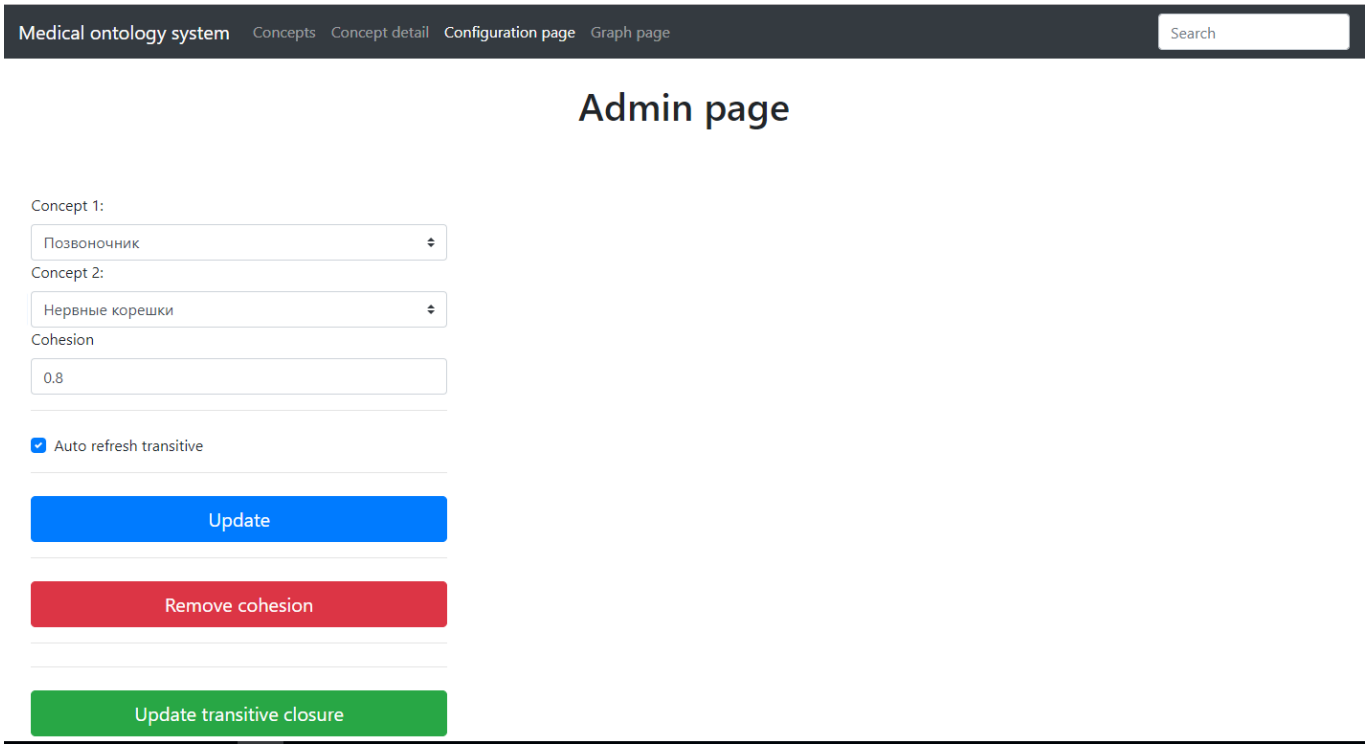


Рисунок 5.8 – Редагування існуючого звязку

Сторінка перегляду візуалізованого графу містить тільки компонент візуалізації. Компонент візуалізації виводить усі вершини графу та звязки між ними. Звязки виводяться тільки прямі, це зумовлено тим, що транзитивних звязків занато багато для виводу користувачу. На ребрах можливо побачити вивід інформації про коефіцієнт впевненості для окремого звязку.

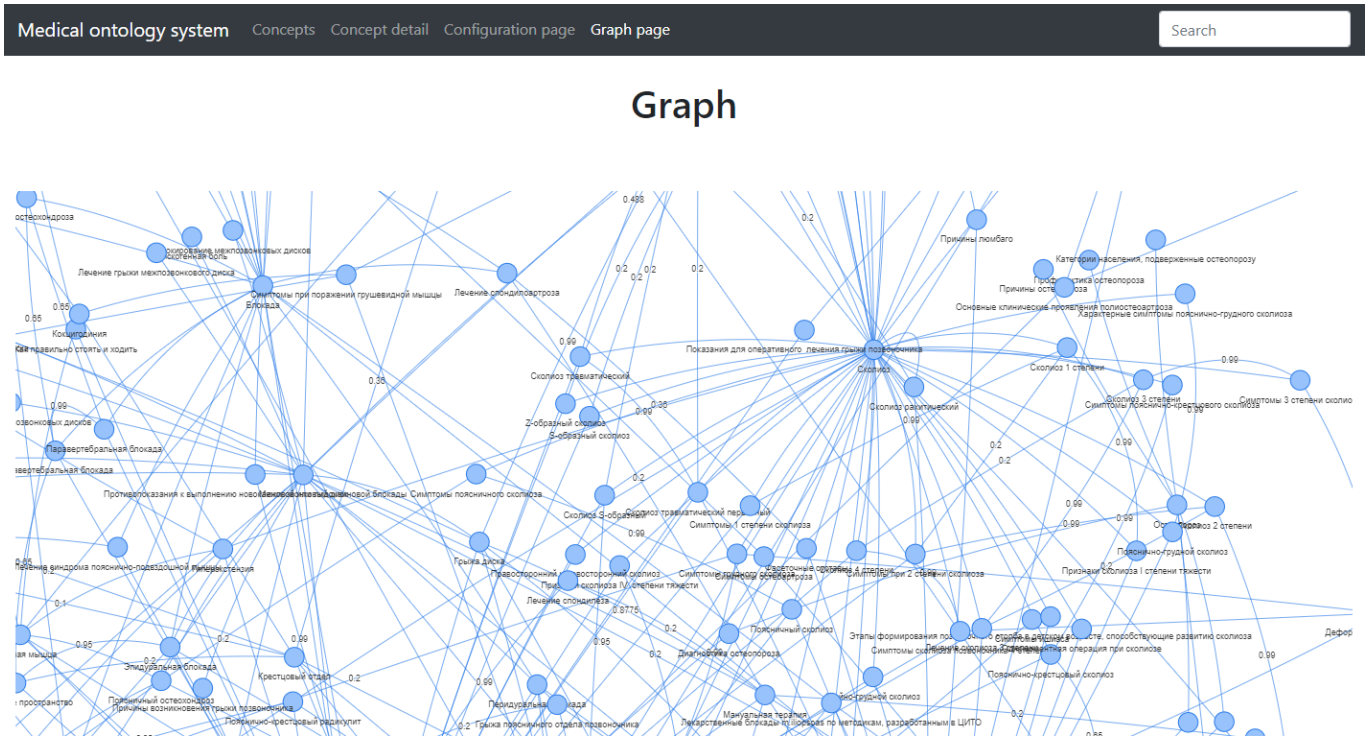


Рисунок 5.9 – Візуалізація графу

Висновки до розділу 5

Розроблений програмний інтерфейс задовольняє усім поставленим вимогам:

- перегляд існуючих понять;
- пошук понять по префіксу слова
- перегляд дочірніх понять;
- перегляд усього графу понять з усіма звязками
- додавання звязків між поняттями;
- змізнення коефіцієнту впевненності звязків між поняттями;
- видалення звязків між поняттями.

Система дозволяє користувачу виконувати усі операції швидко і зручно, має приємний графічний інтерфейс.

6. СТАРТАП ПРОЕКТ

Розділ має на меті проведення маркетингового аналізу стартап проекту задля визначення принципової можливості його ринкового впровадження та можливих напрямів реалізації цього впровадження. Проведення маркетингового аналізу передбачає виконання нижченаведених кроків.

6.1 Опис ідеї проекту

В межах підпункту слід проаналізувати та подати у вигляді таблиць:

- зміст ідеї (що пропонується);
- можливі напрямки застосування;
- основні вигоди, що може отримати користувач товару;
- чим відрізняється від існуючих аналогів та замінників.

Перші три пункти можливо побачити на вигляді таблиці 6.1. Вони дають цілісне уявлення про зміст ідеї та можливі базові потенційні ринки.

Таблиця 6.1. Опис ідеї стартап-проекту

Зміст ідеї	**Напрямки застосування**	**Вигоди для користувача**
Система створення та корегування транзитивного замикання для онтологчни	1. Оптимізація рекурсивних запитів	1. Економія на ресурсах сервера
	2. Пропонування контенту користувачу	2. Дозволяє збільшити час перебування користувача на порталі.
	3. Знаходження дидактичного порядку між великими масивами контенту	3. Дозволяє залучити більше користувачів на веб портал.

Далі виконаємо аналіз потенційних техніко-економічних переваг ідеї (чим відрізняється від існуючих аналогів та замінників). Порівняно із пропозиціями конкурентів передбачає:

− визначення переліку техніко-економічних властивостей та характеристик ідеї;

− визначення попереднього кола конкурентів (проектів-конкурентів) або товарів-замінників чи товарів-аналогів, що вже існують на ринку, та проводиться збір інформації щодо значень техніко-економічних показників для ідеї власного проекту та проектів-конкурентів відповідно до визначеного вище переліку;

− проводиться порівняльний аналіз показників: для власної ідеї визначаються показники, що мають а) гірші значення (W, слабкі); б) аналогічні (N, нейтральні) значення; в) кращі значення (S, сильні) (таблиця 6.2). [32]

Таблиця 6.2. Визначення сильних, слабких та нейтральних характеристик

No п/п		**(потенційні) товари/концепції конкурентів**		
		Мій проект	**WEB сервіси для роботи з графами**	**Бібліотеки для роботи з графами**
1	W слабка сторона	Складна інтеграція	Низька продуктивність	Необхідно додатково реалізувати роботу програму
2		Розроблено під спеціальні дані	Відсутня підтримка додаткової логіки при обчислені	Необхідність проектування симтеми базуючись на струкутрі бібліотеки
3	N нейтральна сторона	Легка у супроводі	Відносно низька вартість послуг.	Безкоштовне використання або за відносно низьку вартість.
4	S сильна сторона	Висока продуктивність	Відсутня необхідність релізовувати значну частину системи	

5		Підтримує додаткову логіку при обчислені ТЗ		

6.2 Технологічний аудит ідеї проекту

В межах даного підрозділу проведемо аудит технології, за допомогою якої можна реалізувати ідею проекту. Визначення технологічної здійсненності ідеї проекту передбачає аналіз таких складових (таблиця 6.3):

- за якою технологією буде виготовлено товар згідно ідеї проекту;
- чи існують такі технології, чи їх потрібно розробити/доробити;
- чи доступні такі технології авторам проекту.

Таблиця 6.3. Технологічна здійсненність ідеї проекту

No п/п	**Ідея проекту**	**Технології її реалізації**	**Наявність технологій**	**Доступність технологій**
1	Модуль створення ТЗ	Php	Наявна	Доступна безкоштовно
2	Веб-інтерфейс користувача	Javascript	Наявна	Доступна безкоштовно
3	Аналіз та тестування алгоритмів	Java	Наявна	Доступна безкоштовно
4	Збереження результатів пошуку ТЗ	MySQL	Наявна	Доступна безкоштовно
5	Візуалізація роботи алгоритму	JavaScript	Відсутня	Відсутня
Висновок: проект реалізувати можливо. Обрана технологія реалізації ідеї проекту: Веб-інтерфейс користувача, Модуль створення ТЗ				

За результатами аналізу таблиці можливо зробити висновок щодо можливості технологічної реалізації проекту: проект можливо розробити. Основні технології та ідеї для реалізації проекту - Веб-інтерфейс користувача, Модуль створення ТЗ.

6.3 Аналіз ринкових можливостей запуску стартап-проекту

Визначемо ринкові можливості, які зможемо використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту. Це дозволить спланувати напрями розвитку проекту із урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів-конкурентів.

Спочатку проведемо аналіз попиту: наявність попиту, обсяг, динаміка розвитку ринку (таблиця 6.4).

Таблиця 6.4. Попередня характеристика потенційного ринку стартап-проекту

No п/п	**Показники стану ринку (найменування)**	**Характеристика**
1	Кількість головних гравців, од	3
2	Загальний обсяг продаж, грн/ум.од	240 грн
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Немає
5	Специфічні вимоги до стандартизації та сертифікації	Немає
6	Середня норма рентабельності в галузі (або по ринку), %	50 %

Середня норма рентабельності в галузі порівнюється із банківським відсотком на вкладення. За умови, що останній є вищим, можливо, має сенс вкласти кошти в інший проект. Оскільки норма рентабельності значно вище банківського відсотку можливо зробити висновок, що ринок є привабливим для входження за попереднім оцінюванням.

Надалі визначимо потенційних груп клієнтів, їх характеристики, та сформуємо орієнтовний перелік вимог до товару для кожної групи (таблиця 6.5).

Таблиця 6.5. Характеристика потенційних клієнтів стартап-проекту

No п/п	**Потреба, що формує ринок**	**Цільова аудиторія (цільові сегменти ринку)**	**Відмінності у поведінці різних потенційних цільових груп клієнтів**	**Вимоги споживачів до товару**
1	Система створення та корегування транзитивного замикання ациклічного графу для онтологічної системи	Інформаційні системи, портали для навчання, наукові бази знань.	компанії заключають довготривалі договори, а стартапери віддають перевагу пробному терміну	стабільність роботи Невисока ціна Наявність пробного періоду Наявність документації

Після визначення потенційних груп клієнтів проведемо аналіз ринкового середовища: необхідно скласти таблиці факторів, що сприяють ринковому впровадженню проекту, та факторів, що йому перешкоджають (таблиці 5.6-5.7).

Таблиця 5.6. Фактори загроз

– **No п/п**	– **Фактор**	– **Зміст загрози**	– **Можлива реакція компанії**
– 1	– Власний формат зберігання	– При необхідності потрібно розробка сервіс преведення до визначенного формату	– Додавання можливості автоматизованого експорту в різні типи сховищ, розробка додаткового ПЗ
– 2	– Обмеженість функцій	– Інструмент обмежений наявними функціями і не має деяких функцій, які мають конкуренти	– Додавання нових функцій за потреби

Таблиця 6.7. Фактори можливостей

No п/п	**Фактор**	**Зміст можливості**	**Можлива реакція компанії**
1	Популярність систем дистанційного навчнання	Системи дистанційного навчання набирають популярність с кожного року	Вихід на ринок дистанційних систем
2	Використання субд MYSQL	SQL бази даних використовуються у великій кількості систем	Використання MySQL
3	Відсутність повноцінних альтернатив	Існуючі альтернативи не надають можливості для корректного знаходження результатів з урахуванням особливостей логіки онтологічної системи	Розширена підтримка існуючих платформ

Надалі проведемо аналіз пропозиції: визначаються загальні риси конкуренції на ринку. Аналіз пропозиції необхідно виконати аналізуючи існуючі види конкуренцій.

Пропозиції повинні відповідати на питання “Як просувати продукт”.

Аналіз пропозицій зображено на таблиці 5.8.

Таблиця 6.8. Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	**В чому проявляється дана характеристика**	**Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)**
1. Вказати тип конкуренції - монополія/олігополія/ монополістична/чиста	чиста	Прямі договори з стартапами, презентація продукту на виставках

2. За рівнем конкурентної боротьби - локальний/національний /...	національний	Публікація статей на міжнародних сайтах
3. За галузевою ознакою - міжгалузева/ внутрішньогалузева	внутрішньогалузева	Розвивати напрямки, нерозвинуті конкурентами
4. Конкуренція за видами товарів: - товарно-родова - товарно-видова - між бажаннями	товарно-видова	Розповідати про свої переваги перед конкурентом у цій галузі
5. За характером конкурентних переваг - цінова / нецінова	нецінова	Надання функцій, які не надають конкуренти
6. За інтенсивністю - марочна/не марочна	марочна	Надання функцій, які не надають конкуренти

Після аналізу конкуренції проводиться більш детальний аналіз умов конкуренції в галузі (таблиця 5.9). [30]

Таблиця 6.9. Аналіз конкуренції в галузі за М. Портером

	– **Прямі конкуренти в галузі**	– **Потенційні конкуренти**	– **Постачальники**	– **Клієнти**	– **Товари-замінники**
– **Складові аналізу**	– Система керування транзитивним замиканням	– Web сервіси для роботи з графами	– Постальчальник остається у вигодному положенні	– Студенти, Науковці, Люди що мають інтерес до певних галузей	– Відсутні

– Висновки:	– Прямі конкуренти ведуть сильну боротьбу	– Є можливості виходу на ринок, оскільки існуючі рішення не надають потрібних переваг	– Постачальники підлаштовуються під ринок –	– Клієнти диктують вимоги згідно з умовами експлуатації –	– Обмеження для роботи на ринку через товари замінники

На основі аналізу конкуренції, проведеного в п. 3.5 (таблиця 6.9), а також із урахуванням характеристик ідеї проекту (таблиця 6.2), вимог споживачів до товару (таблиця 6.5) та факторів маркетингового середовища (таблиця 6.6-6.7) визначимо та обґрунтовуємо перелік факторів конкурентоспроможності. Аналіз оформлюється за таблицею 10. [30]

Таблиця 6.10. Обґрунтування факторів конкурентоспроможності

No п/п	**Фактор конкурентоспроможності**	**Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)**
1	Швидка робота системи	Існуючі конкуренти не можуть запропонувати достойний рівень продуктивності системи за певний час
2	Урахування специфічних потреб при створенні транзитивного замикання	Існуючі конкуренти не передбачають особливості логіки для онтологічних систем

За визначеними факторами конкурентоспроможності (таблиця 6.10) проведемо аналіз сильних та слабких сторін стартап-проекту (таблиця 6.11). [30]

Таблиця 6.11. Порівняльний аналіз сильних та слабких сторін

No п/п	**Фактор конкурентоспроможності**	**Бали 1-20**	**Рейтинг товарів-конкурентів у порівнянні з Database Generator (даним продуктом)**

			-3	-2	-1	0	1	2	3
1	Швидка робота системи	20	+						
2	Урахування специфічних потреб при створенні транзитивного замикання	10			+				

Фінальним етапом ринкового аналізу можливостей впровадження проекту є складання SWOT-аналізу (матриці аналізу сильних (Strength) та слабких (Weak) сторін, загроз (Troubles) та можливостей (Opportunities) (таблиця 6.12) на основі виділених ринкових загроз та можливостей, та сильних і слабких сторін (таблиця 6.11).

Перелік ринкових загроз та ринкових можливостей складається на основі аналізу факторів загроз та факторів можливостей маркетингового середовища. Ринкові загрози та ринкові можливості є наслідками (прогнозованими результатами) впливу факторів, і, на відміну від них, ще не є реалізованими на ринку та мають певну ймовірність здійснення.

Таблиця 6.12. SWOT-аналіз стартап-проекту

Сильні сторони: Висока продуктивність системи Урахування особливостей логіки	**Слабкі сторони:** Прив'язаність до певної мови та СУБД. Інтеграція з системою вимагає певного часу та ресурсів
Можливості: Популярність систем дистанційного навчнання Використання субд MYSQL Відсутність повноцінних альтернатив	**Загрози:** Власний формат зберігання Обмеженість функцій

Перелік ринкових загроз та ринкових можливостей складається на основі аналізу факторів загроз та факторів можливостей маркетингового середовища. Ринкові загрози та ринкові можливості є наслідками (прогнозованими результатами) впливу факторів, і, на відміну від них, ще не є реалізованими на ринку та мають певну

ймовірність здійснення. [30]

Таблиця 6.13. Альтернативи ринкового впровадження стартап-проекту

No п/п	**Альтернатива (орієнтовний комплекс заходів) ринкової поведінки**	**Ймовірність отримання ресурсів**	**Строки реалізації**
1	Орієнтація поточної моделі на ринок стартаперів	30 %	30 год
2	Орієнтація поточної моделі на ринок державних установ	20 %	120 год
3	Орієнтація поточної моделі на ринок ентерпрайз	10 %	150 год
4	Переорієнтація на генерацію серверної частини	65 %	100 год
5	Переорієнтація на веб-розробку	30 %	70 год
6	Переорієнтація на перенесення БД з одних платформ на інші	34 %	178 год

Альтернатива, де отримання ресурсів є більш простим та ймовірним – №4 "Переорієнтація на генерацію серверної частини", що становить 65 відсотків. Це значення перевищує інші альтернативи.

Альтернатива, де строки реалізації є більш стислими – №1 " Орієнтація поточної моделі на ринок стартаперів". Терміни реалізації в цьому разі становлять лише 30 годин.

6.4 Розроблення ринкової стратегії проекту

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів (таблиця 6.14).

За результатами аналізу потенційних груп споживачів (сегментів) оберемо цільові групи, для яких буде запропоновано товар, та визначимо стратегію охоплення ринку.

Таблиця 6.14. Вибір цільових груп потенційних споживачів

No п/п	**Опис профілю цільової групи потенційних клієнтів**	**Готовність споживачів сприйняти продукт**	**Орієнтовний попит в межах цільової групи (сегменту)**	**Інтенсивність конкуренції в сегменті**	**Простота входу у сегмент**
1	Стартапери	Готові	Високий	Висока	Просто
2	Державні установи	Потребують недовгих переговорів	Середній	Середня	Складно
3	Ентерпрайз	Потребують довгих переговорів	Низький	Низька	Дуже складно
Які цільові групи обрано: стартапери					

Для роботи в обраних сегментах ринку необхідно сформувати базову стратегію розвитку (таблиця 6.15).

Таблиця 6.15. Визначення базової стратегії розвитку

No п/п	**Обрана альтернатива розвитку проекту**	**Стратегія охоплення ринку**	**Ключові конкурентоспроможні позиції відповідно до обраної альтернативи**	**Базова стратегія розвитку***

Таблиця 6.15. (продовження)

1	Орієнтація поточної моделі на ринок стартаперів	Стратегія концентрованого маркетингу	Стартапери потребують швидкості розробки, яку надає підтримка декількох платформ даним продуктом	Стратегія спеціалізації (спирається на диференціацію)

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів.

Перелік ринкових загроз та ринкових можливостей складається на основі аналізу факторів загроз та факторів можливостей маркетингового середовища. Після визначення потенційних груп клієнтів проводиться аналіз ринкового середовища: складаються таблиці факторів, що сприяють ринковому впровадженню проекту.

Наступним кроком є вибір стратегії конкурентної поведінки (таблиця 6.16).

Таблиця 6.16. Визначення базової стратегії конкурентної поведінки

No п/п	**Чи є проект «першопрохідцем» на ринку?**	**Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?**	**Чи буде компанія копіювати основні характеристики товару конкурента, і які?**	**Стратегія конкурентної поведінки***
1	Ні	Обидві стратегії	Ні	Стратегія заняття конкурентної ніші

З обраних сегментів до постачальника (стартап-компанії) та до продукту розробляється стратегія позиціонування (таблиця 6.17). що полягає у формуванні ринкової позиції, за яким споживачі мають ідентифікувати торгівельну марку/проект. [30]

Таблиця 6.17. Визначення стратегії позиціонування

No п/п	**Вимоги до товару цільової аудиторії**	**Базова стратегія розвитку**	**Ключові конкуренто-спроможні позиції власного стартап-проекту**	**Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)**
1	Стабільність роботи Невисока ціна Наявність пробного періоду Наявність документації Підтримка необхідних платформ Генерація оптимізованого коду	Стратегія спеціалізації (спирається на диференціацію)	Стартапери потребують швидкості розробки, яку надає підтримка декількох платформ даним продуктом	пришвидшення розробки ПЗ підтримка декількох платформ графічний редактор схеми БД

6.5 Розроблення маркетингової програми стартап-проекту

Для цього у таблиці 6.18 потрібно підсумувати результати попереднього аналізу конкурентоспроможності товару. [**Ошибка! Источник ссылки не найден.**]

Таблиця 6.18. Визначення ключових переваг концепції потенційного товару

No п/п	**Потреба**	**Вигода, яку пропонує товар**	**Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)**
1	Пришвидшення роботи алгоритму ТЗ	Висока продуктивність	Більшість конкуренти працюють дуже повільно
2	Використання нетривіальної логіки обчислення ТЗ	Ураховує логіку	Конкуренти не ураховують кастомну логіку.

Надалі розробимо трирівнева маркетингова модель товару. Для цього уточнимо ідею продукту та/або послуги, його фізичні складові, особливості процесу його надання (таблиця 6.19).

Таблиця 6.19. Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td>I. Товар за задумом</td><td colspan="3">Система керування транзитивним замиканням</td></tr>
<tr><td rowspan="6">II. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх /Тл/Е/Ор</td></tr>
<tr><td>можливість оптимізації витрат часу</td><td>М</td><td>Тл</td></tr>
<tr><td>можливість оптимізації витрат коштів</td><td>М</td><td>Вр</td></tr>
<tr><td>відповідність актуальним технологіям</td><td>Нм</td><td>Тх</td></tr>
<tr><td colspan="3">Відповідає вимогам ДСТУ ISO/IEC 25030:2015 Програмна інженерія. Вимоги щодо якості та оцінювання програмного продукту (SQuaRE). Вимоги щодо якості</td></tr>
<tr><td colspan="3">Пакування: проект idea</td></tr>
<tr><td colspan="3">Марка: Transitive System</td></tr>
<tr><td>III. Товар із підкрі- пленням</td><td colspan="3">Потенційний користувач може ознайомитись з поточним товаром з наукових конференцій та публічних виступів, а також наукових вісників на яких була представлена інформація про даний продукт</td></tr>
<tr><td colspan="4">За рахунок чого потенційний товар буде захищено від копіювання: Назва і контент захищені ліцензією MIT; захист інтелектуальної власності</td></tr>
</table>

М/Нм – монотонні або немонотонні;

Вр/Тх/Тл/Е/Ор – вартісні, технічні, технологічні, ергономічні або органолептичні (останній – для продуктів харчування)

Після формування маркетингової моделі товару слід особливо відмітити – чим саме проект буде захищено від копіювання. Захист може бути організовано за рахунок захисту ідеї товару (захист інтелектуальної власності), або ноу-хау, чи комплексне поєднання властивостей і характеристик, закладене на другому та третьому рівнях товару. [32]

Наступним кроком є визначення цінових меж, якими необхідно керуватись при встановленні ціни на потенційний товар (таблиця 6.20).

Таблиця 6.20. Визначення меж встановлення ціни

No п/п	**Рівень цін на товари-замінники**	**Рівень цін на товари-аналоги**	**Рівень доходів цільової групи споживачів**	**Верхня та нижня межі встановлення ціни на товар/послугу**
1	39…400 грн	300…600 грн	20000…98000 грн	40…150 грн

Наступним кроком є визначення оптимальної системи збуту, в межах якого приймається рішення (таблиця 6.21):

- проводити збут власними силами або залучати сторонніх посередників (власна або залучена система збуту);
- вибір та обґрунтування оптимальної глибини каналу збуту;
- вибір та обґрунтування виду посередників.

Таблиця 6.21. Формування системи збуту

No п/п	**Специфіка закупівельної поведінки цільових кліє-нтів**	**Функції збуту, які має виконувати постачальник товару**	**Глибина каналу збуту**	**Оптимальна система збуту**
1	Клієнт повинен надаватися в режимах “тріал” та “повний”.	Легість в встановленні, легкість в сплаті послуг	Веб сайт. Платформа для продажі voda.org.	Проводити збут силами платформи для продажі voda.org

Останньою складової маркетингової програми є розроблення концепції маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування, визначену специфіку поведінки клієнтів (таблиця 6.22).

Таблиця 6.22. Концепція маркетингових комунікацій

No п/п	**Специфіка поведінки цільових клієнтів**	**Канали комунікацій, якими користуються цільові клієнти**	**Ключові позиції, обрані для позиціонування**	**Завдання рекламного повідомлення**	**Концепція рекламного звернення**
1	Купляють програми через платформу voda.org	Web платформи	Пришвидшення роботи системи	Довести, що програмний продукт краще ніж конкуренти	Бистродія – це наше все.

Висновки до розділу 6

Розроблений програмний продукт конкурентоспроможний, оскільки має ряд переваг над існуючими продуктами. Серед них є значно краща продуктивність роботи системи та коректні результати для логіки, що необхідна для роботи онтологічної системи. Серед недоліків можливо виділити складну інтеграцію та певні вимоги до вхідних даних. Але переваги значно більші ніж недоліки. Отже продукт може стати комерційно успішним.

ВИСНОВКИ

Згідно з завданням необхідно було розробити програмну систему, що дозволятиме користувачу зручно керувати зв’язками між поняттями та будувати транзитивне замикання для графу понять. Також необхідно було реалізувати зручний інтерфейс перегляду та редагування понять та зв’язків між ними. Система має будувати транзитивне замикання для графу понять, також перебудовувати існуюче транзитивне замикання графу після змін. При цьому потрібно було урахувати логіку обчислення послідовних та паралельних звязків.

При детальному розгляді роботи існуючого алгоритму на базі алгоритму Флойда-Уоршала виникли проблеми:

- неправильний результат обчислення коефіцієнтів впевненості у випадках, коли вершини та зв’язки додаються у довільному порядку;
- висока складність алгоритму, що вимагає значних ресурсів при роботі з великою кількістю даних для системи.

Саме тому для вирішення даних проблем та забезпечення коректної та швидкої роботи було розроблено два алгоритми, спираються на використання топологічного сортування перед роботою зі списком врешин. Це алгоритм створення транзитивного замикання що має складність $O(\frac{V^2}{2})$ та алгоритм оновлення транзитивного замикання, що має складність $O(\frac{V^2}{4})$. Для аналізу та тестування алгоритмів був використаний окремий проект, реалізований на мові програмування Java та технології Junit.

Для реалізації серверної частини системи побудови та використання ТЗ було використано мову програмування PHP та технологію XDEbuger. Для розробки клієнтської частини були використані мови HTMP/CSS та JavaScript. Візуалізація графу відбувається за допомогою бібліотеки VISGraph.

Система відповідає вимогам, зазначеним у завданні. Вона дає можливість роботи з транзитивним замиканням. При цьому система спроектована згідно з

паттернами SOLID/GRASP, архітектурним розподіленням системи компонентів, написанням Unit тестів, ООП для повторного використання коду.

Розроблений програмний інтерфейс задовольняє усім поставленим вимогам: Система дозволяє користувачу виконувати усі операції швидко і зручно, маючи при цьому приємний інтерфейс, що також збільшує задоволеність користувачів продуктом.

Розроблений програмний продукт конкурентоспроможний і може стати комерційно учпішним. Це зумовлює велика кількість переваг перед конкурентами та незначна кількість невеликих недолікив у порівнянні з перевагами.

Під час роботи було здобуто пройдено усі етапи розробки програмного забезпечення, що містить у собі певну наукову новизну. Це етапи побудови математичної моделі та проектування, розробки та тестування ПО.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Онтологии как системы хранения знаний [Електронний ресурс]. – Режим доступу: http://www.ict.edu.ru/ft/005706/68352e2-st08.pdf
2. Транзитивное замыкание графа. Алгоритм Уоршалла (Warshall) [Електронний ресурс]. – Режим доступу: http://www.niss.gov.ua/articles/252.
3. Tytenko, S. (2010). Construction of didactic ontology based on the analysis of the elements of the conceptual-thesaurus model. Naukovi Visti NTUU KPI, 1(69), pp.82-87. (in Ukrainian).
4. D. Novak. Concept mapping: A useful tool for science education. Journal of Research in Science Teaching, 27:937-949, 199.
5. M. Guzdial, J. Rick, and C. Kehoe. Beyond adoption to invention: Teacher-created collaborative activities in higher education. Journal of the Learning Sciences, 2002.
6. Coursera. (2018). Coursera - Online Courses & Credentials by Top Educators[online] Available at: https://www.coursera.org/
7. Седжвик, Р. Алгоритмы на Java / Р. Седжвик, К. Уэйн; перевод с англ. – Москва : Альпина паблишер, 2018. – 848 с
8. Алгоритм_Тарьяна_поиска_LCA_за_O(1)_в_оффлайн [Електронний ресурс]. – Режим доступу: http://e-maxx.ru/algo/lca_linear_offline
9. Эккель, Б. Философия Java/ Б. Эккель, К. Джонсон — СПб.: «Питер», 2018. — 1168 с.
10. GoConqr. (2018). GoConqr - Changing the way you learn. [online] Available at: https://www.goconqr.com/
11. Титенко С. В. Побудова дидактичної онтології на основі аналізу елементів понятійно-тезисної моделі/ С. В. Титенко // *Наукові вісті НТУУ "КПІ". – 2010. – № 1(69). – С. 82-87.*
12. EdEra. (2018). EdEra – interactive online education. [online] Available at: https://www.ed-era.com/

13. Erkunt, H. (2004). Developing Systematic Quality E-learning Content. In L. Cantoni & C. McLoughlin (Eds.), Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2004 (pp. 50-55). Chesapeake, VA: AACE.
14. JUnit 5 User Guide [Електронний ресурс]. – Режим доступу: https://junit.org/junit5/docs/current/user-guide/
15. Ellen Francine Barbosa and Jose Carlos Maldonado Computer Science Department, ICMC/USP - Sao Carlos P.O. Box 668 13560-970 - Sao Carlos, SP, Brazil.
16. PHP: HyperText Preprocessor [Електронний ресурс]. – Режим доступу: http://php.net.
17. Web-Технології та Web-дизайн: Скриптова мова PHP [Електронний ресурс]. – Режим доступу: https://sites.google.com/site/da21svietlova/skriptova-mova-php.
18. Udacity. (2018). Udacity – Free Online Classes and Nanodegrees. [online] Available at: https://www.udacity.com/
19. Barbosa, E.F., Maldonado, .T.C, 2006, in International Federation for Information Processing, Volume 210, Education for the 2P' Century-Impact of ICT and Digital Resources, eds. D. Kumar, and Turner J., (Boston: Springer), pp. 17-26.
20. Современный учебник Javascript [Електронний ресурс]. – Режим доступу: https://learn.javascript.ru/
21. Borges V.A., Barbosa, E.F., Using Ontologies for Modeling Educational Content. Instituto de Ciências Matemáticas e de Computação Universidade de São Paulo - USP São Carlos-SP, Brasil.
22. Титенко С.В., Гагарін О.О. Семантична модель знань для цілей організації контролю знань у навчальній системі. // Сборник трудов международной конференции «Интеллектуальный анализ информации-2006». – Київ: Просвіта, 2006. – С.298-307.
23. Титенко С. В. Програмне забезпечення онтологічно-орієнтованої системи керування інформаційно-навчальним Web-контентом : дис. канд. техн. наук : 01.05.03 / Титенко Сергій Володимирович – Київ, 2011. – 120 с.
24. Tytenko, S. (2009). Generation of test tasks in the system of distance learning based

on the model of formalization of didactic text. Naukovi Visti NTUU KPI, 1(63), pp.47-57. (in Ukrainian).

25. Hollingsworth, M. (2016). Building a better eTextbook. Bulletin of the IEEE Technical Committee on Learning Technology, 18(2/3), pp.14-17.
26. Бадд Т. Объектно-ориентированное программирование в действии./ Бадд Т. — СПб.: «Питер», 1997. — 464 с.
27. Php Storm: The Lightning smart IDE for Php programming by JetBrains [Електронний ресурс]. – Режим доступу: https://www.jetbrains.com/phpstorm.
28. рих Гамма. Приемы объектно-ориентированного проектирования. Паттерны проектирования / Эрих Гамма, Ричард Хелм, Ральф Джонсон, Джон Влиссидес. — «Питер», Addison-Wesley, 2009. — С. 366
29. Петруненков А. Оценка коммерческой привлекательности проекта [Електронний ресурс].–Режим доступу: http://www.techbusiness.ru/tb/archiv/number2/page01.htm
30. Тиль, П. От нуля к единице : как создать стартап, который изменит будущее / П. Тиль, Б. Мастерс; перевод с англ. – Москва : Альпина паблишер, 2015. – 188 с.
31. Харниш, В. Правила прибыльных стартапов : как расти и зарабатывать деньги / В. Харниш ; пер. с англ. В. Хозинского. – Москва : Манн, Иванов и Фербер, 2012. – 279 с.
32. Квашнин А. Как продвигать проекты коммерциализации технологий: серия методических материалов «Практические руководства для центров коммерциализации технологий» / М. Катешова, А. Квашнин, под рук. П. Линдхольма, проект EuropeAid «Наука и коммерциализация технологий», 2006. – 52 с.
33. Харниш, В. Правила прибыльных стартапов : как расти и зарабатывать деньги / В. Харниш ; пер. с англ. В. Хозинского. – Москва : Манн, Иванов и Фербер, 2012. – 279 с.

ДОДАТОК А

Побудова та використання транзитивного замикання графа в онтологічно-орієнтованих навчальних системах

Апробації

УКР.НТУУ “КПІ”.ТВ1184_17М

Аркушів 4

2017

ДОДАТОК Б

Побудова та використання транзитивного замикання графа в онтологічно-орієнтованих навчальних системах

Акт впровадження

УКР.НТУУ “КПІ”.ТВ1184_17М

Аркушів 2

2017