Київський національний торговельно-економічний університет
Кафедра інженерії програмного забезпечення та кібербезпеки

ВИПУСКНИЙ КВАЛІФІКАЦІЙНИЙ ПРОЕКТ

на тему:

«Розробка інформаційної системи відслідкування потенційних клієнтів фірми»

Студентки 2м курсу, 6 групи, спеціальності 121 «Інженерія програмного забезпечення» спеціалізації «Інженерія програмного забезпечення» _____ підпис студента — Гаврилюк Яни Миколаївни

Науковий керівник кандидат економічних наук, доцент кафедри інженерії програмного забезпечення та кібербезпеки _____ підпис керівника — Криворучко Олена Володимирівна

Гарант освітньої програми доктор технічних наук, професор кафедри інженерії програмного забезпечення та кібербезпеки _____ підпис гаранта — Криворучко Олена Володимирівна

КИЇВ - 2020

Факультет інформаційних технологій

Кафедра інженерії програмного забезпечення та кібербезпеки

Освітній ступінь магістр

Спеціальність 121 «Інженерія програмного забезпечення»

Затверджую
Зав. кафедри інженерії програмного забезпечення та кібербезпеки
Криворучко О. В.
8 листопада 2019 р.

Завдання
на випускний кваліфікаційний проект студентові

Гаврилюк Яна Миколаївна
(прізвище, ім'я, по батькові)

1. Тема випускного кваліфікаційного проекту «Розробка інформаційної системи відслідкування потенційних клієнтів»

Затверджена наказом ректора від "13" грудня 2019 р. № 4304

2. Строк здачі студентом закінченої проекту 01 грудня 2020р.

3. Цільова установка та вихідні дані до проекту

Мета проекту комбінування та автоматизація методів пошуку оптимальних рішень відслідкування потенційних покупців, які базуються на використанні засобів теорії графів

Об'єкт дослідження задачі пошуку оптимальних рішень

Предмет дослідження застосування засобів теорії графів у процесі розробки програмного продукту пошуку оптимальних рішень відслідкування потенційних покупців

4. Консультанти проекту із зазначенням розділів, які консультують:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв

5. Зміст випускного кваліфікаційного проекту (перелік питань за кожним розділом)

ВСТУП

Розділ 1 ДОСЛІДЖЕННЯ МЕТОДІВ ПРОГРАМУВАННЯ ТА ТЕОРІЇ ГРАФІВ ПОШУКУ ОПТИМАЛЬНИХ РІШЕНЬ

1.1. Методи програмування, визначення їх переваг та недоліків

1.2. Алгоритми та теорія графів

1.3. Дослідження проблем які виникають в задачах візуалізації графів

1.4. Висновки до розділу 1

Розділ 2 АЛГОРИТМИ ВІЗУАЛІЗАЦІЇ РІЗНОМАНІТНИХ ГРАФІВ

2.1. Дослідження та опис алгоритмів візуалізації графів

2.2. Алгоритм «Розміщення під дією сили»

2.3.Алгоритми пошуку найкоротших маршрутів у графі

2.4. Алгоритм Дейкстри.

*2.5. Алгоритм пошуку A**

2.6. Алгоритм Флойда Уоршела

2.7. Алгоритм Джонсона

2.8. Алгоритм Contraction hierarchies

2.9. Висновки до розділу 2

Розділ 3 ФУНКЦІОНУВАННЯ ТОРГОВЕЛЬНОЇ МЕРЕЖІ ЗА ДОПОМОГОЮ ТЕОРІЇ ГРАФІВ

3.1. Математичний апарат постановки задачі.

3.2. Оптимальне розміщення складського приміщення в торговельній мережі «Nash Kraj» у місті Луцьк

3.3. Розрахунки пошуку медіани за допомогою алгоритму Флойда.

3.4. Програмний продукт розрахунку оптимального розміщення елементів торговельної мережі

3.5. Висновки до розділу 3

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ТЕХНІЧНЕ ЗАВДАННЯ

6. Календарний план виконання проекту

№ пор.	Назва етапів випускного кваліфікаційного проекту	Строк виконання етапів проекту	
		за планом	фактично
1	2	3	4
1.	*Вибір теми випускної кваліфікаційної роботи*	*20.09.2019*	*20.09.2019*
2.	*Розробка та затвердження завдання на проект магістра*	*08.11.2019*	*08.11.2019*
3.	*Вступ та перелік літературних джерел*	*24.01.2020*	*24.01.2020*
4.	*Наукова стаття*	*01.09.2020*	*01.09.2020*
5.	*Технічне завдання*	*28.02.2020*	*28.02.2020*
6.	*Розділ 1. Дослідження методів програмування та теорії графів*	*25.06.2020*	*25.06.2020*
7.	*Розділ 2. Алгоритми візуалізації різноманітних графів*	*07.09.2020*	*07.09.2020*
8.	*Розділ 3. Функціонування торговельної мережі за допомогою теорії графів*	*19.10.2020*	*19.10.2020*
9.	*Програма та методика тестування*	*21.10.2020*	*21.10.2020*
10.	*Керівництво користувача*	*23.10.2020*	*23.10.2020*
11.	*Висновки та пропозиції*	*30.10.2020*	*30.10.2020*
12.	*Здача випускного кваліфікаційного проекту на кафедру (перша перевірка)*	*05.11.2020*	*05.11.2020*
13.	*Підготовка автореферату та презентації доповіді*	*05.11.2020*	*05.11.2020*
14.	*Попередній захист випускного кваліфікаційного проекту*	*25.11.2020-27.11.2020*	*25.11.2020-27.11.2020*
15.	*Зовнішнє рецензування випускної кваліфікаційної роботи*	*01.12.2020*	*01.12.2020*
16.	*Підготовка до публічного захисту випускної кваліфікаційної роботи*	*10.12.2020-11.12.2020*	*08.12.2020*

7. Дата видачі завдання «8» листопада 2019 р.

8. Науковий керівник випускного кваліфікаційного проекту Криворучко О. В.
(прізвище, ініціали, підпис)

9. Гарант освітньої програми Криворучко О. В.
(прізвище, ініціали, підпис)

10. Завдання прийняв до виконання студент Гаврилюк Я. М.
(прізвище, ініціали, підпис)

11. Відгук керівника випускного кваліфікаційного проекту

Науковий керівник випускного кваліфікаційного проекту

(підпис, дата)

Відмітка про попередній захист Криворучко О. В.

(ПІБ, підпис, дата)

12. Висновок про випускний кваліфікаційний проект

Випускний кваліфікаційний проект студента Гаврилюк Я. М.

(прізвище, ініціали)

може бути допущена до захисту екзаменаційній комісії.

Гарант освітньої програми Криворучко О. В.

(прізвище, ініціали, підпис)

Завідувач кафедри Криворучко О. В.

(підпис, прізвище, ініціали)

«_____»_____20 ___ р.

АНОТАЦІЯ

Відповідно до мети дослідження робот просвічена комбінуванню та автоматизації методів пошуку оптимальних рішень, які базуються на використанні засобів теорії графів, для розробка інформаційної системи відслідкування потенційних клієнтів фірми.

В результаті дослідження запропоновано вибір оптимального розміщення складського приміщення та визначення найкоротших маршрутів транспортування товару в межах торговельної мережі підприємства торгівлі «Nash Kraj» у місті Луцьк для задоволення попиту покупців на той чи інший товар із застосуванням теорії графів. Розглянуто визначення найменшої домінуючої множини графа, мінімальних відстаней та маршрутів між об'єктами (алгоритм Флойда); медіани графа. Програмний продукт вирішено стандартним підходом – через відображення графів у матрицях та за допомогою візуалізації C ++ Anti-GrainGeometry (AGG).

Ключові слова: *теорія графів, алгоритм Флойда, медіани графа, Фреймворк SwiftPlot, Swift, C ++.*

ABSTRACT

In accordance with the purpose of the study, the robot is dedicated to combining and automating methods of finding optimal solutions based on the use of graph theory to develop an information system for tracking potential customers of the firm.

As a result of the research, the choice of the optimal location of the warehouse and the definition of the shortest routes of transportation of goods within the trade network of the trade enterprise "Nash Kraj" in Lutsk to meet customer demand for a product using graph theory. The definition of the smallest dominant set of a graph, minimum distances and routes between objects is considered (Floyd's algorithm); the medians of the graph. The software product is solved by a standard approach - by displaying graphs in matrices and by visualizing C ++ Anti-GrainGeometry (AGG).

Keywords: *graph theory, Floyd's algorithm, graph medians, SwiftPlot framework, Swift, C ++.*

ЗМІСТ

ВСТУП

У сучасних умовах господарювання важливим є внесок кожної сфери економічної діяльності в розвиток національної економіки. Поряд з виробничими галузями не меншу роль відіграє торгівля, завдяки якій підтримується збалансованість виробництва і споживання, формується суттєва частка валової доданої вартості в Україні, забезпечується робочими місцями економічно активне населення.

Проте на споживчому ринку пропозиція частіше перевищує попит. Торговельні об'єкти ведуть запеклу боротьбу за клієнтів, намагаючись зацікавити покупця будь-яким можливим способом. Виникають нові форми супермаркетів, покликані задовольнити найвибагливіших відвідувачів. Звичайним торговим павільйонам стає все складніше виживати в таких умовах. Підприємці, які мають власну мережу магазинів, усе більше зіштовхуються з проблемою раціонального розміщення торговельних точок, складських приміщень, що є одним з вагомих факторів ведення конкурентної боротьби. Знайти розв'язок задач оптимального розміщення торгових об'єктів можна за допомогою мереж або графів.

Огляд останніх джерел досліджень і публікацій. За останні роки з'явилася велика кількість наукових робіт, у яких використовуються можливості теорії графів не тільки в математиці й традиційних додатках – хімії та електротехніці, а й у соціології, лінгвістиці, економіці та генетиці. Наочність теоретико-графових структур дозволяє зробити доступними для широкого кола науковців складні прикладні задачі. Відомості щодо інструментів можна знайти в працях А. А. Зикова, Н. Кристофідеса, В. Н. Салія та А. М. Богомолова, Р. Уілсона, Р. Дістеля.

Постановка завдання. Завдання роботи полягає у вирішенні питання застосування теорії графів, її методів та інструментів для оптимізації розміщення торгівельної мережі для збільшення потенційних покупців, тобто для зменшення витрат на транспортні шляхи до мережі магазинів.

Мета роботи - комбінування та автоматизація методів пошуку оптимальних рішень, які базуються на використанні засобів теорії графів.

Під ***об'єктом дослідження випускного кваліфікаційного проекту*** розуміємо задачі пошуку оптимальних рішень.

Предметом дослідження випускного кваліфікаційного проекту постають можливості застосування засобів теорії графів у процесі розробки програмного продукту пошуку оптимальних рішень.

Основними ***задачами є*** розробка комбінованого методу пошуку оптимальних рішень з використанням теорії графів та його програмну реалізацію розв'язання оптимізаційних задач.

Методи, які застосовано для розв'язання задач: основи теорії графів, системний підхід для аналізу інформації, метод параметричної оптимізації для оптимізації алгоритму, структурно-функціональний метод для здійснення постановки задачі.

Проблема пошуку оптимального рішення постає базовою у процесі розв'язання оптимізаційних задач у різних галузях людської діяльності: економіці, логіці, техніці, медицині, мережевих технологіях тощо. Розв'язок таких задач часто можна формалізувати до опису математичної моделі критеріїв оптимальності засобами теорії графів та математичної статистики [1,2] і розглядати питання оптимізації як алгоритмізований пошук найкоротших шляхів між вершинами графа з урахуванням фізичного значення його вершин та дуг. Тому актуальною є розробка програмного продукту розв'язання оптимізаційних задач.

РОЗДІЛ 1.
ДОСЛІДЖЕННЯ МЕТОДІВ ПРОГРАМУВАННЯ ТА ТЕОРІЇ ГРАФІВ ПОШУКУ ОПТИМАЛЬНИХ РІШЕНЬ

Термін “граф” вперше з’явився в науковій літературі в 1936 році в роботах угорського математика Д. Кьоніга, хоча елементи теорії графів були відомі та широко використовувались ще у XVIII столітті, зокрема в роботах Л. Ейлера.

Незалежно від уподобань, з графами, мабуть, зустрічався кожний. Розуміючи під графом певну множину точок, деякі з яких з’єднані лініями, цей незвичний математичний об’єкт можна без перебільшення застосовувати в будь-якій науковій чи практичній галузі.

Можливим застосуванням графів в економіці можуть бути:

- задачі розподілу ресурсів, які виникають при певному наборі операцій (робіт), що необхідно виконати за обмежених ресурсів;
- задачі управління запасами, що полягають у знаходженні оптимальних значень рівня запасів і розміру замовлення для забезпечення безперебійного виробничого процесу;
- задачі мережного планування і управління, в яких розглядаються співвідношення між термінами закінчення великого комплексу операцій і моментами початку всіх операцій комплексу. Потрібно знайти мінімальні тривалості комплексу операцій, оптимальні співвідношення вартості і термінів виконання;
- мережні задачі, що полягають у оптимізації процесу обслуговування на мережах чи самої структури мережі;

- задачі дослідження конфліктних ситуацій, які полягають у виборі оптимальних стратегій поведінки учасників конфлікту;
- задачі масового обслуговування систем з чергами заявок;
- задачі ремонту і заміни устаткування, присвячені зносу і старінню устаткування з необхідністю його заміни з часом;
- задачі складання розкладів полягають у визначенні оптимальної черговості виконання операцій на різних видах устаткування чи при певному способі надання послуг;
- задачі планування і розміщення, пов'язані з визначенням оптимального числа і місця розміщення нових об'єктів з урахуванням їх взаємодії з наявними об'єктами і між собою

З метою вирішення багатьох задач, пов'язаних зі складними виробничими структурами, останні поділяються на сукупність дрібніших, але все-таки досить великих елементів для того, щоб отримана формалізована структура по суті "припинила" б бути великою системою, тобто описувалася б досить обмеженою кількістю параметрів. Такою структурою вже можна оперативно керувати і розв'язувати для неї оптимізаційні задачі, як це й робиться на практиці. Такі розподілені системи називають мережними, а сукупність методів, що застосовують до їх аналізу – методами (чи алгоритмами) оптимізації на мережах чи графах.

Граф є поширеним методом для відображення зв'язків між певними даними. Причому він має широке застосування не тільки як інструмент вирішення задач, а й як основний засіб для побудови математичних та інформаційних моделей.

Можна навести декілька прикладів, де візуалізація графових структур є дуже корисною:

- ієрархії класів і пакетів вихідного коду будь-якої програми;

- візуалізація соціального графа (той же Twitter або Facebook);
- зображення графа цитувань (дослідження зв’язності статей, хто на кого посилаються).

1.1. Методи програмування, визначення їх переваг та недоліків

І снує два основних підходи до написання програмного коду: текстове та візуальне програмування.

Під текстовим програмуванням будемо розуміти спосіб написання при якому код вводиться з клавіатури та зберігається у вигляді текстового файлу. Як правило, коли люди говорять про текстові мови програмування, то вони посилаються на такі мови програмування, як Python, C#, Java і JavaScript. Для написання програмного коду можна використовувати будь-який текстовий редактор. Проте, для підвищення ефективності, зазвичай, використовується інтегроване середовище розробки (IDE) програмного забезпечення. Підвищення продуктивні досягається завдяки візуально простим інтерфейсам в IDE та сильно зв’язаним компонентам, таким як текстовий редактор, компілятор тощо. Як правило, інтегроване середовище являє собою один програмний комплекс, в якому ведеться уся розробка. Він, зазвичай, має багато модулів для написання, редагування, компілювання, налагодження та розгортання програмного забезпечення. IDE дає можливість абстрагуватися від виконання другорядних задач, що дозволяє розробнику значно зменшити кількість витраченого часу на типові технічні дії (наприклад, виклик компілятора, запуск засобів для налагодження коду) та дає можливість зосередитися на виконанні бізнес-задач та алгоритмічних задач. Окрім цього, інтегроване середовище розробки дає можливість зробити аналіз написаного коду і миттєво сповістити розробника про синтаксичні помилки у коді. Це дозволяє значно зменшити час на розробку програм. Завдяки вищенаведеним можливостям збільшується продуктивність розробника.

Візуальне програмування – спосіб написання програми для електронно-обчислювальної машини методом графічного маніпулювання програмними елементами замість написання тексту [1]. Візуальне програмування часто представляють як наступний етап розвитку текстових мов програмування. Наочним прикладом може служити Microsoft Visual Studio, де графічні об’єкти редагуються і водночас змінюється та показується відповідний програмний код. У зв'язку з розвитком мобільних пристроїв (смартфонів, планшетів) програмісти стали більше цікавитися візуальним програмуванням. Візуальне програмування в основному використовується для створення програм з графічним інтерфейсом для операційних систем з графічним інтерфейсом користувача.

Необхідно розрізняти наступні терміни:

- графічна мова програмування – мова, призначена для написання програми для комп'ютера або обчислювального пристрою, в якому замість текстового опису алгоритму роботи використовується графічний;
- візуальні засоби розробки – засоби проектування, в якій найбільш поширені блоки програмного коду представлені у вигляді графічних об'єктів. Застосовуються в основному для створення прикладних програм і розробки графічного інтерфейсу користувача (GUI).

Візуальні мови програмування можуть бути додатково класифіковані на: мови, що базуються на іконках; мови, що базуються на формах [1]. Візуальне середовище розробки забезпечує графічні елементи, які можуть інтерактивно управлятися користувачами відповідно до певної специфічної просторової граматики для побудови програми.

Загальна мета візуальних мов програмування – зробити програмування більш доступним для новачків та підтримувати програмістів на трьох різних рівнях [2]:

– Синтаксис. Візуальні мови програмування використовують іконки/блоки, форми та діаграми, які намагаються зменшити або навіть повною мірою уникнути можливість виникнення синтаксичних помилок.

– Семантика. Візуальні мови програмування можуть забезпечити деякі механізми для розкриття значення програмних примітивів. Вони можуть містити допоміжні функції, які забезпечують функції документації.

– Прагматика. Візуальні мови програмування мають можливість показати стан програми у конкретних ситуаціях. Це дозволяє користувачам розміщувати артефакти, створені за допомогою візуальної мови програмування, у певний стан, щоб вивчити, як програма реагує на цей стан. Приклади: в AgentSheets або AgentCubes користувачі можуть встановлювати гру чи симуляцію в певний стан, щоб побачити, як реагує програма. Користувач, що володіє мовою програмування Thymio, може помістити робота в певний стан, щоб побачити, як він буде реагувати, тобто, які датчики активуються.

Графічна мова програмування, як правило, використовує функцію перетягування, а не введення. Вона може містити іконки або текстові мітки на блоках і елементах. Часто використовуються такі GUI елементи, як діалоги та спадні меню вибору (drop-down menu).

Зазвичай, у візуальному програмуванні блок-схема алгоритмів виконання відображується у вигляді графа, тому необхідно навести термінологію із теорії графів.

1.2. Алгоритми та теорія графів

У математиці теорія графів полягає у вивченні графів, які є математичними структурами, що використовуються для моделювання попарних зв'язків між об'єктами. Граф у цьому контексті складається з

вершин(вузлів або точок), які з'єднані ребрами (дугами або лініями). Графи є одним з основних об'єктів дослідження в дискретній математиці.

У найбільш загальному розумінні граф – це упорядкована пара $G = (V, E)$, що складається з множини V вершин та множину E ребер. Причому множина ребер E є декартовим добутком $V \times V$ множини V. Це означає що кожне ребро з'єднує рівно дві вершини.

Теорія графів – розділ дискретної математики, що вивчає властивості графів. В загальному сенсі граф представляється як множина вершин (вузлів), з'єднаних ребрами $G = (V, E)$, де V – підмножина будь-якої множини, а E – підмножина $V \times V$.

Якщо на кожному ребрі обрати певний напрямок, то такий граф називається *орієнтованим*, а його ребра з напрямками — *дугами.* Для орієнтованого графа зберігаються всі поняття та характеристики, що відносяться до відповідного неорієнтованого.

Маршрутом між вершинами x_a та x_b у графі називають скінченну послідовність ребер та інцидентних їм вершин, що складають неперервну криву з кінцями x_a, x_b. Число ребер у маршруті називається його *довжиною* (включаючи внесок повторюваних ребер).

Маршрут називається *ланцюгом,* якщо всі його ребра (дуги) різні, і *простим ланцюгом,* якщо всі його вершини (крім кінців) різні.

Циклом у графі називається ланцюг, в якому перша та остання вершина збігається з початковою [3].

1.3. Дослідження проблем які виникають в задачах візуалізації графів

У системах візуального програмування блок-схема алгоритмів виконання відображується у вигляді орієнтованого направленого графа.

Причому, розмір даного графа може бути дуже суттєвим. Зрозуміло, що для ефективної взаємодії з даним графом необхідно щоб він задовольняв основним візуально-естетичним параметрам [3]:

- *накладання*. Кількість накладань (схрещувань) в графі це число з'єднань, які візуально перетинаються. Існує домовленість що планарний (граф, який може бути зображений на площині без перетину ребер) граф рисується без накладань. Такий граф ще називають плоским.
- *область зображення*. Область зображення – розмір найменшого обмежувального прямокутника, який будуються по відношенні до найменшої відстані між двома вершинами графа. Зазвичай, чим меншою є ця область, тим кращим є зображення графа, тому що це дозволяє показати більше вершин графа.
- *загальна/максимальна довжина ребер*. Декілька, часто вживаних параметрів якості зображення відносяться до довжини відрізків зображення. Зазвичай, бажано мінімізувати загальну довжину ребер графа. Це, у свою чергу, означає що контекстно-близькі вершини будуть знаходитися поряд (що також є важливим параметром оцінки якості візуалізації).
- *загальна кількість згинів*. Для простоти отриманого зображення необхідно щоб з’єднання між вершинами графа являли собою якомога більш прості геометричні примітиви. Це робиться з метою спрощення сприйняття відрізків зображення оком людини. Зазвичай, складність з’єднання визначається кількістю згинів. Багато алгоритмів візуалізації графів орієнтовані на мінімізацію згинів в цілому чи мінімізацію згинів для кожного з’єднання.

У системах візуального програмування кожна вершина може містити декілька входів-виходів. Відповідно, для зменшення складності отримуваного

зображення графа необхідно реалізувати механізм візуалізації графів, оптимізуючи вищенаведені параметри.

1.4. Висновки до розділу 1

Проведено порівняльний аналіз текстового та візуального програмування, наведено базову термінологію теорії графів. Було досліджено проблеми, які виникають при візуалізації графів. На основі цих проблем було визначено мету магістерської дисертації та здійснено постановку задач. Також було проведено аналіз існуючих алгоритмів візуалізації графів. Поставлено задачі, основною метою яких є створення алгоритму візуалізації з'єднань графа, який буде виконувати поставлену мету та задовольнятиме критеріям оптимальності.

Для аналізу інформації був використаний системний підхід. Мається на увазі, що об’єкт дослідження, бібліотека візуалізації графів/діаграм обчислень, розглядається як комплекс взаємопов’язаних компонентів. Системний підхід виявляє взаємозв’язки та закономірності в об’єкті з метою їх більш ефективного використання. Також необхідно зазначити, що системний підхід не є засобом для вирішення деяких задач, а є методом для постановки задач. Основним принципом системного підходу є цілісність, яка дає змогу аналізувати об’єкт як одне ціле та водночас, як підсистему для вищих рівнів. Наступним вагомим принципом є структуризація, яка дає можливість аналізувати елементи системи і їх взаємозв’язки в рамках конкретної організаційної структури.

РОЗДІЛ 2.
АЛГОРИТМИ ВІЗУАЛІЗАЦІЇ РІЗНОМАНІТНИХ ГРАФІВ

2.1. Дослідження та опис алгоритмів візуалізації графів

Перед початком проектування та розробки певного алгоритму візуалізації графа необхідно провести аналіз вже існуючих рішень. Для кожного методу визначити недоліки та переваги, щоб мати змогу розробити власний алгоритм.

Існує багато алгоритмів для візуалізації графів. Розглянемо найпоширеніші з них, виділимо їх переваги та недоліки.

Алгоритм випадкового розміщення. Одним із найпростіших алгоритмів у візуалізації графів є алгоритм Випадкового (Random) розміщення. Суть даного методу полягає у тому що вершини розташовуються у довільному положенні та порядку. Вибирається прямокутна область потрібної величини. Далі у цій області випадковим чином розміщують всі вершини та ребра графа. Розподіл вершин у даній області є рівномірним.

Зрозуміло, що даний алгоритм дуже простий у реалізації та може візуалізувати граф із довільною кількістю вершин. Проте, даний метод не задовольняє параметрам естетичності та практичності.

Приклад візуалізації за допомогою алгоритму випадкового розміщення зображений на Рис. 2.1.

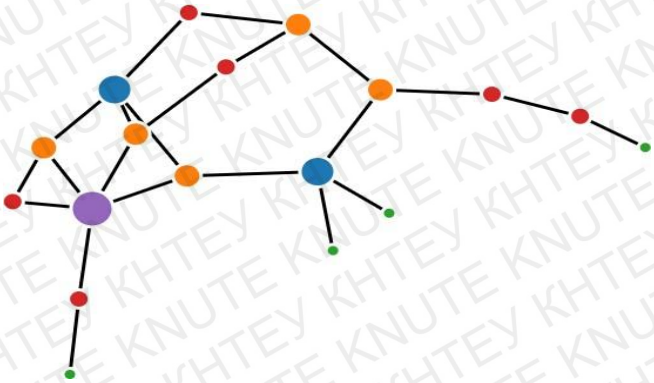

Рис. 2.1 Візуалізація з допомогою алгоритму випадкового розміщення

Алгоритм кругового розміщення. Алгоритм кругового розміщення – алгоритм, в якому вершини графа розмішуються на колі. Зазвичай, відстані між вершинами графа є однаковими. Потрібно відмітити, що вказаний алгоритм практично не залежить від кількості вершин та ребер графа, а також дає хороший результат. Однією із переваг даного алгоритму є його нейтралітет. Це досягається тим що усі вершини розташовуються на однакових відстанях один від одного та від центру круга. Тобто, жодній вершині не надається привілейована позиція, так як зазвичай користувач сприймає центрально розташовані вузли як важливіші. Вказана властивість кругового розміщення є корисною для деяких сфер, зокрема, вона активно використовується у біоінформатиці та аналізі соціальних мереж.

Ребра графа можуть бути зображені у вигляді хорд кола (Рис. 2.2) чи дуг (Рис. 2.3). Зображення з допомогою є хорд використовується лише при малій кількості вершин графа, адже із збільшенням вершин графа інформативність та зрозумілість отриманого зображення різко зменшується.

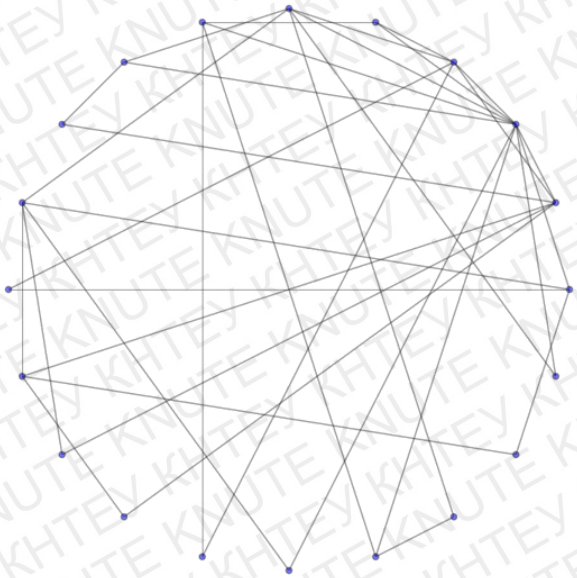

Рис. 2.2. Зображення графа алгоритмом кругового розміщення з хордами

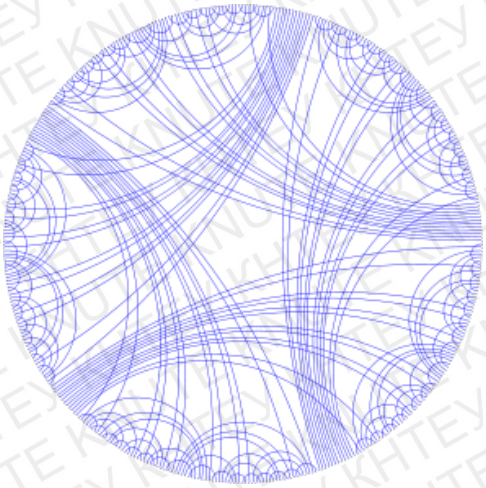

Рис. 2.3 Зображення графа алгоритмом кругового розміщення з дугами

Попри всі свої переваги, круговий алгоритм візуалізації графа є незастосовним для масштабних і складний графів, адже він на надає інформацію про структуру графа.

Алгоритм дугового розміщення. Дугова діаграма – це стиль подання графа, в якому вершини розташовані вздовж прямої на евклідовій площині, а

ребра утворюються у вигляді півкіл на одній з двох плоских поверхонь [6]. Слід відмітити, що у деяких випадках відрізки прямої також використовуються для представлення ребер графа, якщо сусідні вершини графа містять зв'язок між собою (Рис. 2.4).

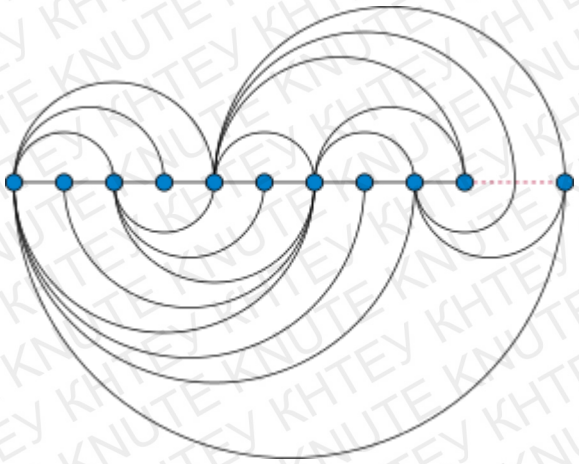

Рис. 2.4. Зображення у вигляді дугової діаграми

Для зручності зображення орієнтованих графів існує домовленість напрямок зліва направо представляти у вигляді дуги над прямою, а напрямок справа наліво – під прямою. Ця домовленість називається орієнтацією по годинниковій стрілці і була розроблена як частина іншого стилю візуалізації графа, до складу якої входили Фекке, Ванг, Данг та Аріс. Пізніше, Преторіус і ван Вейк застосували це правило до дугових діаграм.

Дугові діаграми дають можливість відобразити багатомірні дані, які асоціюються з вершинами графа у простій формі. Проте, в даного методу представлення є один суттєвий недолік, сформульований авторами Хір, Босток и Огівєтскі. Вони писали, що дугові діаграми: «не можуть передати повну структуру графа так саме ефективно, як це робить двовимірне представлення» [7].

2.2. Алгоритм «Розміщення під дією сили»

«Розміщення під дією сили» (Force Directed Placement) – клас алгоритмів, спрямованих на візуалізацію графів із хаотично розміщеними вузлами, так щоб отримане в результаті зображення задовольняло параметрам естетичності (симетричності та мінімальності кількості перетинів ребер). Їх метою є розташування вершин графа в двовимірному чи тривимірному просторі так, щоб всі ребра були більш-менш однакової довжини та щоб кількість накладань була наскільки малою, наскільки це можливо(Рис. 1.5). Це досягається присвоєнням вершинам та ребрам певної сили (зазвичай використовується закон Гука), яка відповідає відстані між вершинами. Мається на увазі, що чим більша відстань між вершинами, тим менша буде величина присвоєної сили і навпаки. На наступному етапі відбувається імітації руху ребер та вузлів. Ці рухи відбуваються в напрямку зменшення їхньої енергії [8].

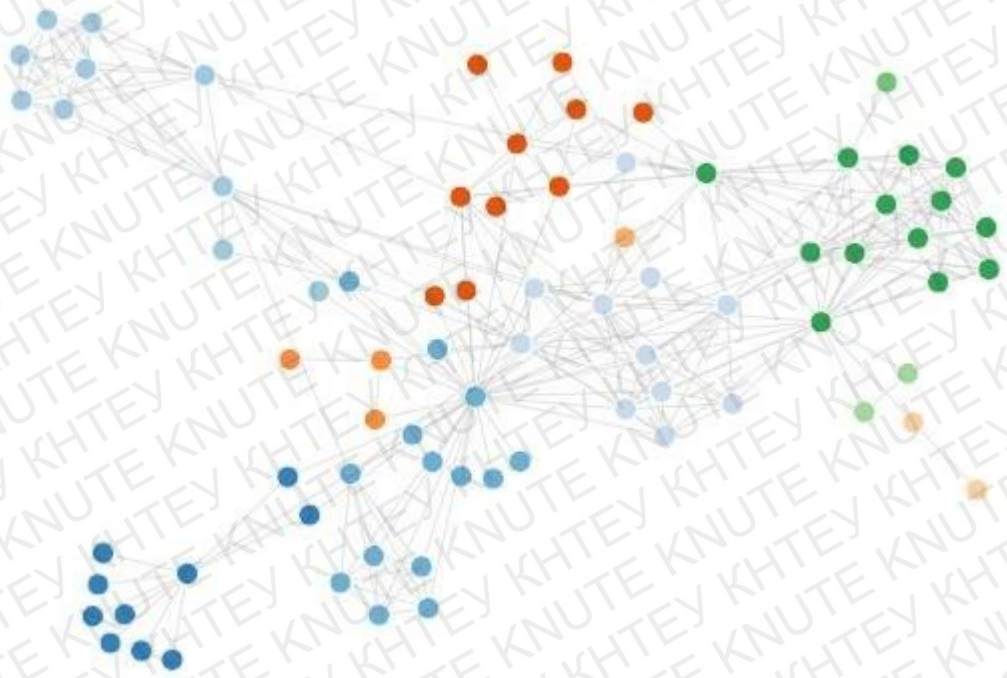

Рис. 2.5. Візуалізація алгоритмом «Розміщення під дією сили»

Зображення графа може бути складною проблемою, проте алгоритми класу «Розміщення під дією сили», будучи фізичним моделюванням, зазвичай не вимагають спеціальних знань про теорію графів, таких як планарність.

Алгоритм Сугіями. Алгоритм Сугіями призначений перш за все для оптимізації візуальних- естетичних критеріїв, як зменшення довжин ребер та мінімізація перетинів ребер.

Вказаний алгоритм складається з трьох основних етапів (Рис. 2.6):

– розподіл вершин за рівнями так, щоб напрямок дуг був незмінним (зверху вниз для низхідних, знизу вверх для висхідних зображень) і довжина кожного ребра була мінімальною.

– мінімізація кількості перетинів ребер за допомогою зміни порядку вершин на кожному рівні.

– вибір координат вершин на кожному рівні для мінімізації числа згинів ребер [10].

Для вирішення завдань кожного з зазначених етапів існує ряд евристик. Таким чином, конкретна реалізація є «конструктором» з різних методів, що вирішують завдання різних етапів.

Завдяки інтуїтивній зрозумілості та застосовності до широкого класу графів, порівневий підхід користується високою популярністю серед розробників систем візуалізації графів.

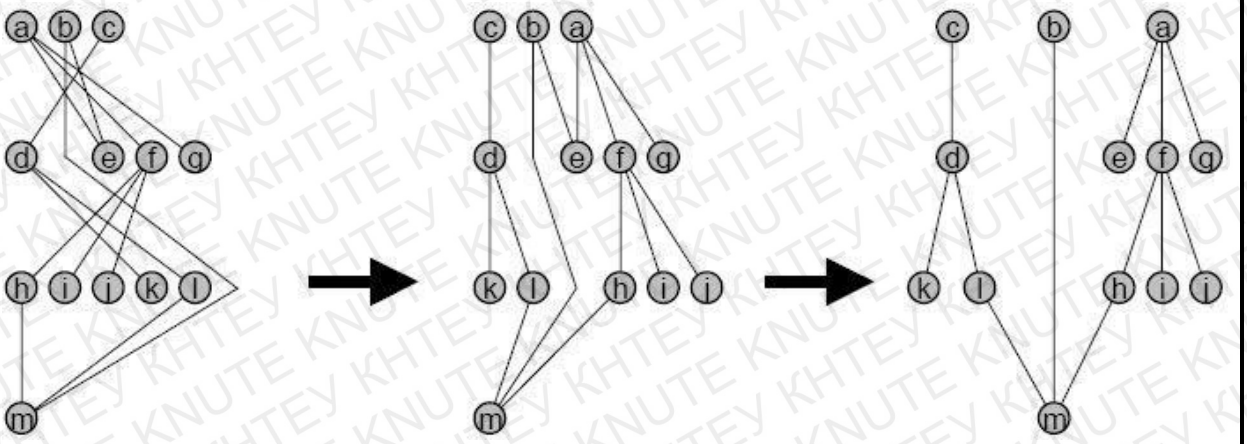

Рис. 2.6. Ілюстрація алгоритму Сугіями

При всіх своїх перевагах у даного методу є декілька недоліків:

– алгоритм працює тільки для направлених графів;
– граф не повинен містити цикли.

Розглянувши декілька алгоритмів візуалізації графів, визначимо параметри, які важливі при виборі оптимального алгоритму.

Алгоритм має задовольняти основним візуально-естетичним параметрам. Так, необхідно мінімізувати число з'єднань, які візуально накладаються. Це дозволить простіше слідувати від однієї вершини графа до наступної, що в свою чергу підвищить зрозумілість зображуваного графа.

Другим бажаним параметром є мінімізація довжини ребер отриманого графа. Це, також, також означає що пов'язані вершини будуть розміщені одна біля одної, що дозволить зрозуміти структуру графа із зображення.

Для забезпечення простоти зображення графа необхідно щоб з'єднання між вершинами мали просту геометричну форму. Зазвичай, складність такого з'єднання визначається кількістю згинів.

Останнім параметром є швидкість виконання алгоритму. Для ефективної взаємодії з користувачем необхідно щоб алгоритм був застосовним для складних і масштабних графів (більше 500 вузлів та 1000 з'єднань).

Відповідність алгоритмів даним вимогам можна знайти у табл. 2.1.

Таблиця 2.1.

Порівняння алгоритмів візуалізації графів

Алгоритм **Параметр**	**Випадкове розміщення**	**Кругове розміщення**	**Дугова діаграма**	**Алгоритм «Розміщення під дією сили»**	**Алгоритм Сугіями**
Мінімізація кількості накладань	–	+	+	+	+
Мінімізація довжини ребер	–	–	+	–	+
Мінімізація кількості згинів	–	–	–	–	–
Застосовний для масштабних і складних графів	+	–	–	–	+

Як бачимо, серед розглянутих алгоритмів немає такого, який задовольнив би усі вимоги.

2.3. Алгоритми пошуку найкоротших маршрутів у графі

Задача про мінімізацію довжин ребер графа є надзвичайно актуальною для системи візуального програмування. Адже, як згадувалося раніше, мінімізація довжин з'єднань графа є корисним для спрощення та покращення естетичних показників зображення. Це дозволить розташувати сильно пов'язані вершини одна біля одної.

Задача про найкоротший шлях - завдання пошуку найкоротшого шляху між двома вершинами графа, в якій мінімізується сума ваг ребер, що складають шлях. Задача про знаходження найкоротшого шляху є однією із найбільш фундаментальних і типових задач теорії графів. Інакше вона ще називається задачею про мінімальний шлях. Практичне значення даної задачі визначається тим, що вона входить до складу інших більш складних задач та є моделлю для розробки ефективних методів вирішення задач теорії графів.

Значимість розв'язання даної задачі визначається її різними практичними застосуваннями. Наприклад, в GPS-навігаторах здійснюється пошук найкоротшого шляху між двома перехрестями. Вершинами виступають перехрестя, а дороги – ребрами, які лежать між ними. Якщо сума довжин доріг між перехрестями мінімальна, тоді знайдений шлях найкоротший.

Так, наприклад, у економічних за своїм змістом задачах довжину ребра можуть відігравати такі величини, як час, вартість, обсяг витрачених ресурсів. Важливо, що при русі вздовж шляху ці характеристики додаються, відповідно, як і самі довжини при розрахунку загальної довжини. Це дає змогу мінімізувати вищевказані показники.

Рішення завдання про найкоротший шлях може бути проведено двома способами:

- на основі алгоритму повного перебору усіх можливих шляхів;
- методом динамічного програмування на графі.

Метод вирішення задачі про найкоротший шлях на основі алгоритму повного перебору шляхів, що з'єднують задані початкову і кінцеву вершини, є простим і очевидним. Спочатку будуються всі можливі шляхи між початковою і кінцевою точками. Обчислюються довжини усіх отриманих шляхів і вибирається мінімальна з них. Вибраний шлях (один або кілька) і буде розв'язком задачі. При цьому в сам алгоритм перебору шляхів можна внести

незначні поправки, що дозволяють відкинути деякі явно невигідні варіанти. Наприклад, не потрібно продовжувати обчислення довжини шляху, якщо його довжина вже перевищує мінімальну довжину знайдених раніше шляхів. Зрозуміло, що основним недоліком даного алгоритму є його низька швидкодія та трудомісткість для складних графів.

Якщо проблема може бути вирішена оптимально, розбиваючи її на під проблеми, а потім рекурсивно шукаючи оптимальні рішення підзадач, то вона, як кажуть, має оптимальну структуру.

2.4. Алгоритм Дейкстри

Одним із прикладів використання принципів динамічного програмування є алгоритм Дейкстри для пошуку найкоротшого шляху. Алгоритм знаходить найкоротший шлях між вказаною вершиною графа та усіма іншими вершинами графа. Також, можна знайти відстань між деякою початковою і кінцевою вершиною. Для цього виконання алгоритму починається із вказаної точки і продовжується до моменту, поки не буде досягнута очікувана кінцева точка. Наприклад, якщо вершини графа будуть представлені містами, а ребра – дистанцію між цими містами, то алгоритм Дейкстри може бути використаний для знаходження мінімальної відстані між деяким початковим містом і всіма іншими. Завдяки наведеній властивості, даний алгоритм широко використовується в мережевих протоколах маршрутизації (таких як IS-IS і OSPF) [11].

Наведемо опис послідовності виконання алгоритму Дейкстри:

– Усі вершини графа позначаються як ”невідвідані”. Алгоритм виконується для кожної точки лише один раз і для уникнення повторного виконання вводиться термін ”відвідана” вершина (Рис. 2.7).

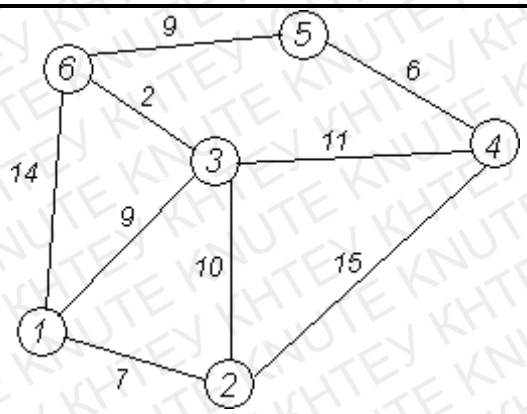

Рис. 2.7. Усі вершини 'невідвідані'

– Кожній вершині графа присвоюється проміжне значення довжини відрізка. Початковій вершині присвоюється значення 0, усім іншим – безкінечність (Рис. 2.8). Початкова вершина позначається як поточна вершина.

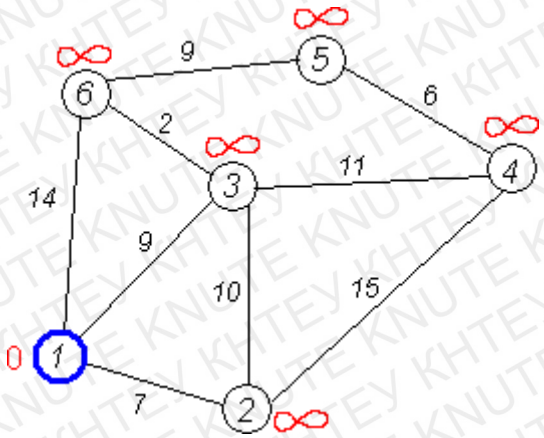

Рис. 2.8. Початкова ініціалізація значень

– Знаходять сусідні вершини для поточної вершини і обчислюється відстань від поточної вершини до усіх сусідніх. Якщо отримане розрахункове значення довжини менше за значення мітки сусіда, то мітка сусіда замінюється на отримане значення довжини (Рис. 2.9). Наприклад, якщо поточна вершина А позначена відстанню 6, а ребро, що з'єднує його з сусідом В, має довжину 2, тоді відстань до В через А буде дорівнювати 6 + 2 = 8. Якщо вершина В була позначена міткою, більшою за 8, то значення мітки замінюється на 8. В іншому випадку значення мітки не змінюється.

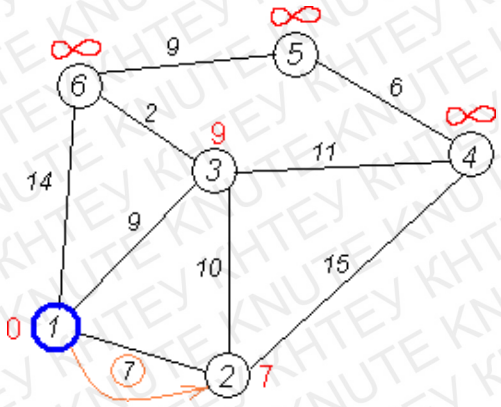

Рис. 2.9. Визначення вершини з мінімальною вагою

– Після закінчення розглядання усіх сусідів поточної вершини, поточна вершина позначається ”відвіданою” (Рис. 2.10). ”Відвідана” вершина більше не буде перевірятися у наступних кроках.

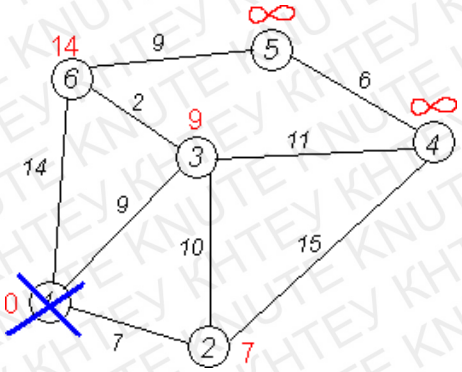

Рис. 2.10. Позначення ‘відвіданої’ вершини

– Алгоритм повторюється до тих пір поки усі вершини не будуть ”відвіданими” (Рис. 2.11)

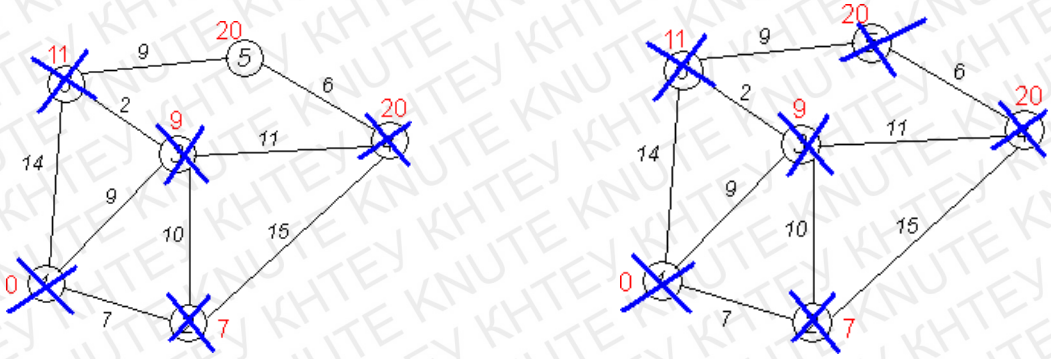

Рис. 2.11. Закінчення виконання алгоритму Дейкстри

При обчисленні маршруту не обов’язково чекати поки кінцева вершина буде позначена ”відвіданою”: алгоритм може бути завершеним у випадку коли

кінцева вершина має найменшу розраховану дистанції серед усіх ”невідвіданих” вершин. У деякий випадках це може значно скоротити час виконання алгоритму.

2.5. Алгоритм пошуку A*

Пошук A* – в інформатиці та математиці, алгоритм пошуку по першому найкращому збігу на графі, який знаходить найкоротший маршрут від однієї вершини (початкової) до іншої (кінцевої).

Порядок обходу вершин визначається евристичною функцією «відстань + вартість» (зазвичай позначається як f (x)). Ця функція – сума двох інших: функції вартості досягнення розглянутої вершини (x) з початкової вершини (зазвичай позначається як g (x) і може бути як евристичної, так і не), а також функції евристичної оцінки відстані від поточної вершини до кінцевої (позначається як h (x)) [12].

Функція h (x) повинна бути допустимою евристичною оцінкою, тобто не повинна переоцінювати відстані до цільової вершини. Наприклад, для задачі маршрутизації h (x) може представляти собою відстань до цілі по прямій лінії, так як це фізично найменша можлива відстань між двома точками.

Цей алгоритм був вперше описаний в 1968 році Пітером Хартом, Нілсом Нільсоном і Бертрамом Рафаелем. По суті це було розширення алгоритму Дейкстри. Новий алгоритм досяг більш високої продуктивності (критерієм часу) за допомогою евристики. В їх роботі він згадується як

«алгоритм А». Але так як він обчислює найкращий маршрут для заданої евристики, то він був названий А*.

А * покроково переглядає всі маршрути, що ведуть від початкової вершини в кінцеву, поки не знайде мінімальний. Як і всі евристичні алгоритми пошуку, він переглядає спочатку ті маршрути, які «здаються» ведуть до мети. Від жадібного алгоритму, який теж є алгоритмом пошуку по першому найкращому збігу, його відрізняє те, що при виборі вершини він враховує, крім іншого, весь пройдений до неї шлях. Складова g (x) - це вартість шляху від початкової вершини, а не від попередньої, як в жадібному алгоритмі.

На початку роботи проглядаються вузли, суміжні з початковим; вибирається той з них, який має мінімальне значення f (x), після чого цей вузол розкривається. На кожному етапі алгоритм оперує з безліччю шляхів з початкової точки до всіх ще не розкритих (листових) вершин графа - безліччю приватних рішень, - яке розміщується в черзі з пріоритетом. Пріоритет шляху визначається за значенням f (x) = g (x) + h (x). Алгоритм продовжує свою роботу до тих пір, поки значення f (x) цільової вершини не виявиться меншим, ніж будь- яке значення в черзі, або поки все дерево не буде переглянуте. З безлічі рішень вибирається рішення з найменшою вартістю.

Чим менше евристика h(x), тим більший пріоритет має вершина, тому для реалізації черги можна використовувати сортувальні дерева.

Безліч переглянутих вершин зберігається в closed, а які потребують розгляду шляху - в черзі з пріоритетом open. Пріоритет шляху обчислюється за допомогою функції f (x) всередині реалізації черги з пріоритетом.

Безліч вже пройдених вершин можна опустити (при цьому алгоритм перетвориться в пошук по дереву рішень), якщо відомо, що рішення існує або якщо successors вміє відсікати цикли.

Алгоритм A* допустимий і обходить при цьому мінімальну кількість вершин, завдяки тому, що він працює з «оптимістичною» оцінкою шляху через вершину. Оптимістичною в тому сенсі, що, якщо він піде через цю вершину, у алгоритму «є шанс», що реальна вартість результату буде дорівнює цій оцінці, але ніяк не менше. Але, оскільки A* є поінформованим алгоритмом, така рівність може бути цілком можливою.

Коли A* завершує пошук, він, згідно з визначенням, знайшов шлях, справжня вартість якого менше, ніж оцінка вартості будь-якого шляху через будь-який відкритий вузол. Але оскільки ці оцінки є оптимістичними, відповідні вузли можна без сумнівів відкинути. Інакше кажучи, A* ніколи не пропустить можливості мінімізувати довжину шляху і тому є допустимим.

Припустимо тепер, що якийсь алгоритм B повернув в якості результату шлях, довжина якого більше оцінки вартості шляху через деяку вершину. Напідставі евристичної інформації, для алгоритму B не можна виключити можливість, що цей шлях мав і меншу реальну довжину, ніж результат. Відповідно, поки алгоритм B переглянув менше вершин, ніж A*, він не буде допустимим. Отже, A* проходить найменшу кількість вершин графа серед допустимих алгоритмів, що використовують таку ж точну (або менш точну) евристику.

2.6. Алгоритм Флойда Уоршела

Алгоритм Флойда-Уоршела(**Воршелла)** - динамічний алгоритм для знаходження найкоротших відстаней між усіма вершинами зваженого орієнтованого графа. Розроблено в 1962 році Робертом Флойдом і Стівеном Уоршелом, хоча в 1959 році Бернард Рой (Bernard Roy) опублікував практично такий же алгоритм, але це залишилося непоміченим.

Нехай вершини графа $G = (V,\ E),\ |V| = n$ пронумеровані від 1 до n і введено позначення d_{ij}^k для довжини найкоротшого шляху від i до j, який окрім самих вершин $i,\ j$ проходить тільки через вершини $1 \ldots k$. Очевидно, що d_{ij}^0 - довжина (вага) ребра $(i,\ j)$, якщо воно існує (в іншому разі його довжина може бути позначена як ∞)

Існує два варіанти значення $d_{ij}^k,\ k \in (1,\ \ldots,\ n)$:

1. Найкоротший шлях між $i,\ j$ не проходить через вершину k, тоді $d_{ij}^k = d_{ij}^{k-1}$
2. Існує більш короткий шлях між $i,\ j$, що проходить через k, тоді він спочатку йде від i до k, а потім від k до j. У цьому випадку, очевидно, $d_{ij}^k = d_{ik}^{k-1} + d_{kj}^{k-1}$

Таким чином, для знаходження значення функції досить вибрати мінімум з двох позначених значень.

Тоді рекурентна формула для d_{ij}^k має вигляд:

d_{ij}^0 — довжина ребра $(i,\ j)$

$d_{ij}^k = \min(d_{ij}^{k-1},\ d_{ik}^{k-1} + d_{kj}^{k-1})$

Алгоритм Флойда-Воршелла послідовно обчислює всі значення d_{ij}^k, $\forall i,\ j$ для k від 1 до n. Отримані значення d_{ij}^n є довжинами найкоротших шляхів між вершинами $i,\ j$.

Основним завданням **може бути** програмна реалізація алгоритму пошуку найкоротшого шляху між двома будь-якими вершинами графа. Програма повинна працювати так, щоб користувач вводив кількість вершин і довжини ребер графа, а після обробки цих даних на екран виводився найкоротший шлях між двома заданими вершинами. Необхідно передбачити різні результати пошуку, щоб програма не видавала помилок і працювала правильно.

Алгоритми пошуку найкоротших шляхів широко застосовуються і розвиваються в наш час. Кожен з нас завжди шукає найкоротший шлях для вирішення певної задачі. Правильне використання алгоритмів дає змогу знаходити найкоротший шлях з найменшими затратами часу та ресурсів.

2.7. Алгоритм Джонсона

Алгоритм Джонсона дозволяє знайти найкоротші шляхи між усіма парами вершин зваженого орієнтованого графа. Даний алгоритм працює, якщо в графі містяться ребра з позитивною або негативною вагою, але відсутні цикли з негативною вагою.

Даний граф $G = (V, E)$ з ваговою функцією ω: $E \rightarrow R$. Якщо ваги ребер ω у графі невід'ємні, можна знайти найкоротші шляхи між усіма парами вершин, запустивши алгоритм Дейкстри по одному разу для кожної вершини. Якщо в графі містяться ребра з негативною вагою, але відсутні цикли з негативною вагою, можна обчислити нову множину ребер з невід'ємними вагами, що дозволяє скористатися попереднім методом. Нова множина, що складається з ваг ребер $\hat{\omega}$повинна задовольняти наступні вимоги:

- для усіх ребер (u, v) нова вага $\hat{\omega}(u, v) > 0$;
- для усіх пар вершин $u, v \in V$ шлях P є найкоротшим шляхом з вершини u до вершини v з використанням вагової функції ω тоді і тільки тоді, коли P — також найкоротший шлях з вершини u до вершини v з ваговою функцією $\hat{\omega}$[10].

Алгоритм Лі. Алгоритм хвильового трасування (хвильовий алгоритм, алгоритм Лі) [9]

- алгоритм пошуку шляху, алгоритм пошуку найкоротшого шляху на планарному графі. Належить до алгоритмів, заснованих на методах пошуку в ширину. В основному використовується при комп'ютерному трасуванні (розводці) друкованих плат, з'єднувальних провідників на поверхні мікросхем.

Інше застосування хвильового алгоритму - пошук найкоротшої відстані на карті в комп’ютерних стратегічних іграх.

Хвильовий алгоритм в контексті пошуку шляху в лабіринті був запропонований Е. Ф. Муром. Лі незалежно відкрив цей же алгоритм при формалізації алгоритмів трасування друкованих плат в 1961 році.

Алгоритм працює на дискретному робочому полі (ДРП) , що представляє собою обмежену замкнутою лінією фігуру, не обов'язково прямокутну, розбиту на прямокутні осередки, в окремому випадку - квадратні. Безліч всіх осередків ДРП розбивається на підмножини: «прохідні» (вільні), тобто при пошуку шляху їх можна проходити, «непрохідні» (перешкоди),шлях через цей осередок заборонений, стартовий осередок (джерело) і фінішний (приймач). Призначення стартового і фінішного осередків умовно, достатньо вказівки пари осередків, між якими потрібно знайти найкоротший шлях.

Алгоритм призначений для пошуку найкоротшого шляху від стартового осередку до кінцевого осередку, якщо це можливо, або, за відсутності шляху видати повідомлення про непрохідність.

Робота алгоритму включає в себе три етапи: ініціалізацію, поширення хвилі і відновлення шляху.

Під час ініціалізації будується образ множини осередків оброблюваного поля, кожному осередку приписуються атрибути прохідності / непрохідності, запам'ятовуються стартовий і фінішний осередки.

Далі, від стартового осередку породжується крок до сусіднього осередку, при цьому перевіряється, чи прохідний він, і чи не належить раніше відміченому у дорозі осередку.

Сусідні осередки прийнято класифікувати двояко: в сенсі околиці Мура і околиці фон Неймана, які відрізняються тим, що в околиці фон Неймана

сусідніми осередками вважаються тільки 4 осередки по вертикалі і горизонталі, в околиці Мура – усі 8 осередків, включаючи діагональні.

При виконанні умов прохідності і неналежності її до раніше позначених в дорозі осередків, в атрибут осередку записується число, рівне кількості кроків від стартового осередку, від стартового осередку на першому кроці це буде 1. Кожен осередок, мічений числом кроків від стартового осередку стає стартовим і з нього породжуються чергові кроки в сусідні осередки. Очевидно, що при такому переборі буде знайдено шлях від початкового осередку до кінцевого, або черговий крок з будь-якого породженого в дорозі осередку буде неможливий.

Відновлення найкоротшого шляху відбувається в зворотному напрямку: при виборі осередку від фінішного осередку до стартового на кожному кроці вибирається осередок, що має атрибут відстані від стартового на одиницю менше поточного осередку. Очевидно, що таким чином знаходиться найкоротший шлях між парою заданих осередків [13].

2.8. Алгоритм Contraction hierarchies

Алгоритм призначений для швидкого пошуку маршруту між двома вершинами графа. Є алгоритмом переобробки графа. Тобто завдяки цьому алгоритму ми отримаємо оптимізований граф для пошуку маршруту. Після чого можна буде використати будь-який з відомих алгоритмів пошуку маршруту.

Суть алгоритму полягає в наступному: на кожному кроці ми обираємо одну вершину і для всіх сусідніх вершин додаємо ребра скорочення, де це можливо. На рис. 2.12 зображено приклад додавання ребра скорочення.

Як можемо побачити, найкоротший маршрут від A до E лежить через вершину C. Тому між ребрами A і E додається ребро скорочення AE.

Аналогічно з вершинами A і B. В подальшому при пошуку маршруту вершина C ігноруватиметься.

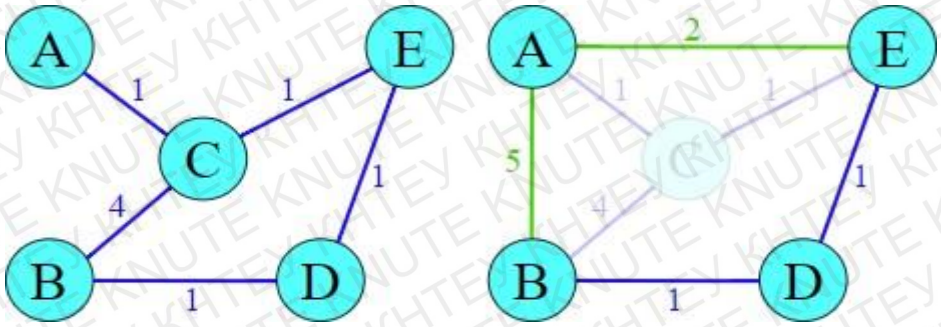

Рис. 2.12 Додавання ребер скорочення

Принципове значення для тривалості роботи алгоритму має порядок розгляду вершин. Результат від цього не буде залежати, але в залежності від вибору порядку обробки тривалість роботи може відрізнятися в рази.

Передобробка може займати значний час. Але для графа реальних доріг, який може включати мільйони вершин і ребер вона дає великий виграш у швидкості пошуку маршрутів [15].

2.9. Висновки до розділу 2

Розглянуто та системно упорядковано описані основні задачі теорії графів пов'язані з пошуком шляху. Спочатку була розглянута задача пошуку найкоротшого шляху графі, наведена формальна постановка. Після були описані алгоритми, які найчастіше використовуються при вирішенні завдань пошуку найкоротшого шляху в графі. Зокрема: алгоритм Дейкстри, алгоритм Белмана - Форда, алгоритм A*, алгоритм Флойда-Уоршела, алгоритм Джонсона і алгоритм Лі. Починаючи від винайдення алгоритму Дейкстри в 1959 року і до сьогодні було винайдено багато алгоритмів на графах, що використовуються в багатьох сферах життя. Очевидно, що сучасні алгоритми пошуку шляхів у графах, що використовуються навігаторами є добре оптимізованими і винайти щось краще вкрай важко. Але базуючись на вже

відомих алгоритмах можна будувати нові інтелектуальні системи, що стануть у нагоді багатьом користувачам.

Використано структурно-функціональний метод – підхід при поясненні та описі системи, який орієнтується на виявлення структури системи. Під структурою системи розуміється комплекс взаємозв'язків між елементами системи та їх ролі. Також необхідно зазначити, що структурно-функціональний метод є різновидом системного підходу.

Даний підхід вимагає вивчення структури об'єкта , аналіз елементів досліджуваного об'єкта і їх функціональних характеристик, дослідження зміни елементів та їх функцій, розгляд розвитку об'єкта в цілому.

Для оптимізації алгоритму доцільно використати метод параметричної оптимізації. Параметрична оптимізація – процес знаходження значень параметрів, при яких досягається максимальна ефективність виконання процедури або забезпечуються найкращі характеристики пристрою або системи з точки зору встановленого критерію. За кількістю параметрів, які оптимізуються, задача оптимізації може бути однопараметричною чи багатопараметричною, за кількістю критеріїв – однокритеріальною чи багатокритеріальною. Виходячи із поставлених задач до магістерської роботи та сформульованих критеріїв оптимальності, можна зробити висновок, що задача є багатопараметричною та багатокритеріальною

Для досягнення поставленої мети варто застосувати алгоритми визначення:

- найменшої домінуючої множини графа;
- мінімальних відстаней та маршрутів між об'єктами (алгоритм Флойда);
- медіани граф

РОЗДІЛ 3.
ФУНКЦІОНУВАННЯ ТОРГОВЕЛЬНОЇ МЕРЕЖІ ЗА ДОПОМОГОЮ ТЕОРІЇ ГРАФІВ

Підприємці, які мають власну мережу магазинів, усе більше зіштовхуються з проблемою раціонального розміщення торговельних точок, складських приміщень, що є одним з вагомих факторів ведення конкурентної боротьби. Знайти розв'язок задач оптимального розміщення торгових об'єктів можна за допомогою мереж або графів.

Завдання вирішення проблеми полягає у вирішенні питання застосування теорії графів, її методів та інструментів для оптимізації розміщення торгівельної мережі для зменшення витрат на транспортування товарів., які користуються попитом.

Отже, для реалізації мети роботи - вибір оптимального розміщення складського приміщення та визначення найкоротших маршрутів транспортування товару в межах торговельної мережі підприємства торгівлі «Nash Kraj» у місті Луцьк для задоволення попиту покупців на той чи інший товар застосуємо теорію графів.

Рис. 3.1. Мережа підприємства торговельної мережі «Nash Kraj» у місті Луцьк.

Для досягнення оптимального розміщення складського приміщення та визначення найкоротших маршрутів транспортування товару в межах торговельної мережі торгівлі «Nash Kraj» у місті Луцьк для задоволення попиту покупців на той чи інший товар застосовано алгоритми визначення:

1) найменшої домінуючої множини графа,

2) мінімальних відстаней та маршрутів між об'єктами (алгоритм Флойда);

3) медіани графа.

3.1. Математичний апарат постановки задачі

Малі та середні фірми, що обмежують збут своєї продукції одним або декількома довколишніми районами, мають, як правило, один склад. Територіальне розміщення складів та їх кількість визначаються потужністю матеріальних потоків і їх раціональною організацією, попитом на ринку збуту, розмірами районів збуту й концентрацією в ньому споживачів,

відносним розташуванням постачальників й покупців, особливостями комунікаційних зв'язків і т.д. Слід мати на увазі, що завдання розміщення та формування складської мережі – оптимізаційне, оскільки, з одного боку, будівництво нових і купівля діючих складів та їх експлуатація пов'язані зі значними капіталовкладеннями, а з іншого – треба забезпечити (разом з підвищенням рівня обслуговування споживачів) скорочення витрат за рахунок максимального наближення складів до клієнтів.

У загальному вигляді доцільно зобразити цю ситуацію у вигляді графа, який складається з точок (вершин), що відображають основні елементи ситуації, та лінії (ребра), які з'єднують пари цих вершин та відображають зв'язки між ними.

Граф, або неорієнтований граф, G − це впорядкована пара $G = (V, S)$, для якої виконуються такі умови:

V − множина вершин або вузлів,

S − множина пар (у випадку неорієнтованого графа − невпорядкованих) вершин, які називають ребрами.

Визначення найменшої домінуючої множини графа (найменшої зовнішньої стійкості графа) дозволяє знайти об'єкти, які можуть бути використаними як складське приміщення [1].

Найменшою домінуючою множиною графа називається множина $R \subseteq V$, яка містить мінімальну кількість елементів і задовольняє умову:

$$(\forall \nu_J \in V - R)(\exists \nu_j \in R)(s_{ij} = 1) \qquad (3.1)$$

Граф $G = (V, S)$, надається множиною вершин $V = \{V_1, V_2, \ldots, V_n\}$ і матрицею суміжності $S = [S_{ij}]$ порядку n, де:

$$s_{ij} = \begin{cases} l, \nu_{ij} \text{ т } \nu_j \text{ - з'єднані ребром;} \\ 0 \text{ - у іншому випадку.} \end{cases} \quad (3.2)$$

Цілочислові змінні – x_{ij}, $i = \overline{I,n}$, $j = \overline{I,n}$, *де*

$$x_{ij} = \begin{cases} 1, \text{вершина } \nu_i \text{ входить у шукану множину;} \\ 0 \text{ - у іншому випадку} \end{cases} \quad (3.3)$$

Цільова функція має вигляд:

$$\sum_{i=I}^{n} x_i \to \min \quad (3.4)$$

1. Кожна вершина графа, що не входить у домінуючу множину, повинна бути видимою із його вершини хоча б один раз:

$$\sum_{i=I}^{n} S_{iR} x_{ik} \geq 1; \; k = \overline{I,n} \quad (3.5)$$

2. Область допустимих значень:

$$0 \leq x_{ij} \leq 1, \; i = \overline{I,n} \quad j = \overline{I,n} \quad (3.6)$$

3.2. Оптимальне розміщення складського приміщення в торговельній мережі «Nash Kraj» у місті Луцьк

Для вибору оптимального розміщення складського приміщення в межах уже існуючої торговельної мережі (рис. 3.2.), подамо її у вигляді графа.

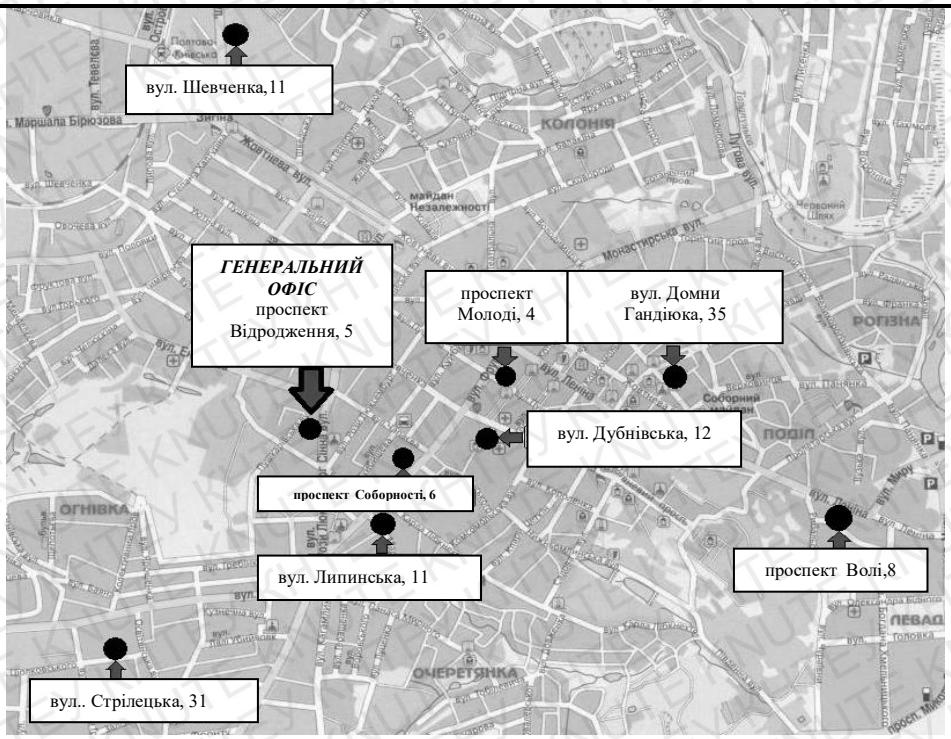

Рис. 3.2 Об'єкти торговельної мережі «Nash Kraj» у місті Луцьк

Відповідно з'єднавши об'єкти торговельної мережі, отримаємо неорієнтований граф. Слід зазначити, що кожна пара об'єктів не є з'єднаними виключно однією вулицею, щоб дістатися того чи іншого пункту необхідно проїхати відповідний маршрут, що і з'єднує вершини графа. Для маршрутів було обрано мінімальні відстані між точками.

Отже, вершинами графа є підприємства-об'єкти мережі торгівлі «Nash Kraj» у місті Луцьк, а ребрами − маршрути (рис. 3.3)

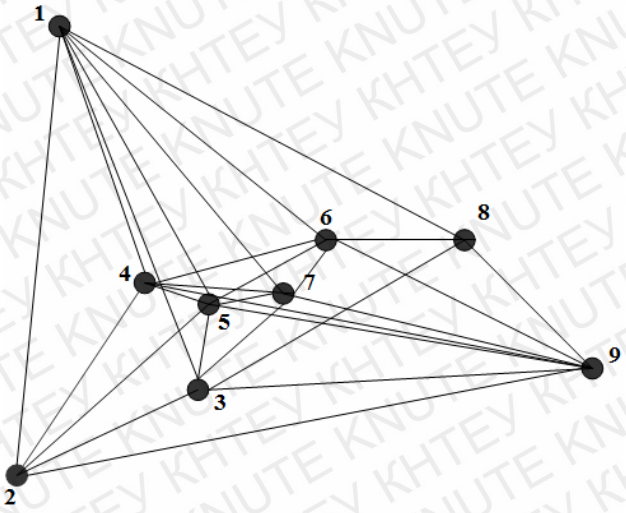

Рис. 3.3. Граф торговельної мережі «Nash Kraj» у місті Луцьк

Зображений граф (рис. 3.3) є повним, тому що всі 9 точок попарно з'єднані між собою, та мають прямий доступ одна до одної.

За результатами застосування алгоритму 1 (визначення найменшої домінуючої множини графа), об'єкт, розміщений у точці 1 (вул. Шевченка, 11), є найменшою домінуючою множиною та може контролювати райони функціонування інших точок. У цій точці можна розмістити складське приміщення (рис. 3.4).

	A	B	C	D	E	F	G	H	I	J	K	L	M
1		вул. Шевченка,11	вул. Стрілецька, 31	вул. Липинська, 11	пров. Відродження, 5	просп. Соборності, 6	просп. Молоді, 4	вул. Дубнівська, 12	вул. Домни Гондіюка, 35	просп. Волі, 8		Змінні Xi	
2	вул. Шевченка,11	1	1	1	1	1	1	1	1	1		1	
3	вул. Стрілецька, 31	1	1	1	1	1	1	1	1	1		0	
4	вул. Липинська, 11	1	1	1	1	1	1	1	1	1		0	
5	пров. Відродження, 5	1	1	1	1	1	1	1	1	1		0	
6	просп. Соборності, 6	1	1	1	1	1	1	1	1	1		0	
7	просп. Молоді, 4	1	1	1	1	1	1	1	1	1		0	
8	вул. Дубнівська, 12	1	1	1	1	1	1	1	1	1		0	
9	вул. Домни Гондіюка, 35	1	1	1	1	1	1	1	1	1		0	
10	просп. Волі, 8	1	1	1	1	1	1	1	1	1		0	
11		загальне бачення графа										ЦФ	
12		1	1	1	1	1	1	1	1	1		1	

Рис. 3.4. Знаходження найменшої домінуючої множини точок графа

Для підтвердження чи спростування результатів проведено додаткове дослідження.

Оскільки це торговельна мережа, то вагомим етапом для поліпшення її функціонування є розрахунок відстаней маршрутів, саме вони дозволять повністю дослідити існуючу мережу та виробити подальші рекомендації щодо оптимального розміщення торгових об'єктів. Цю вимогу можна задовольнити за допомогою обчислення мінімальних відстаней та маршрутів за алгоритмом Флойда [2].

У початковій матриці ваг $c_{ij}=0$ для всіх $i=1,2,3, \ldots, n$ і $c_{ij}=\infty$, якщо у графа відсутня дуга (x_i, x_j).

Крок 1. Присвоюємо початкові значення $k=0$.

Крок 2. $k=k+1$.

Крок 3. Для усіх $i \neq k$, таких, що $c_{ij} \neq \infty$, і для всіх $j \neq k$, таких, що $c_{ij} \neq \infty$, вводимо операцію:

$$c_{ij} = \min \left[c_{ij} \left(c_{ij} + c_{kj} \right) \right] \qquad (3.7)$$

Крок 4. Якщо $c_{ii} < 0$, то в графі G існує цикл від'ємної ваги, що містить вершину x_i і розв'язку немає. *Закінчення.*

Якщо $c_{ii} \geq 0$ і $k = n$, то отримано розв'язок. Матриця $[c_{ij}]$ дає довжину всіх найкоротших шляхів. *Закінчення.*

Якщо $c_{ii} \geq 0$, але $k < n$, то повернутися до кроку 2 [2].

Найкоротші ж шляхи можна знайти за заданими довжинами за допомогою рекурсивної процедури. При цьому, як до повернення до вагової матриці C, зберігається та оновлюється друга $(n \times n)$ матриця $\Theta = [\Theta_{ij}]$. Елемент Θ_{ij} вказує вершину, що є попередньою вершині x_j у найкоротшому шляху від x_i до x_j.

Відповідно до формули (3.7) на кроці 3 алгоритм оновлення матриці має такий вигляд:

$$\Theta_{ij} = \begin{cases} \Theta_{k,j}, \text{ ящ } \left(c_{kj} + c_{kj}\right) < c_{ij} \quad \text{що визначається у} \\ \text{квадратних дужках формули } (3.7) \\ \text{не змінюється, гккщ } c_{ij} \le \left(c_{ik} + c_{kj}\right) \end{cases} \qquad (3.8)$$

У кінці алгоритму 2 (визначення мінімальних відстаней та маршрутів між об'єктами), найкоротші шляхи отримано безпосередньо із завершальної матриці *Θ*. Цей метод можна використовувати як до невід'ємних, так і до будь-яких вагових матриць.

Застосувавши значення відстаней між об'єктами торговельної мережі підприємства торгівлі «Nash Kraj» у місті Луцьк та сформувавши матрицю маршрутів (рис. 3.5), визначимо перевірку всіх елементів матриці за допомогою трикутного оператора на можливість їх поліпшення.

	1	2	3	4	5	6	7	8	9
1) вул. Шевченка,11	0	5	5	3	3	3	3	4	5
2) вул. Стрілецька, 31	5	0	3	3	3	3	3	4	5
3) вул. Липинська, 11	5	3	0	2	2	2	2	2	3
4) проп. Відродження, 5	3	3	2	0	1	2	2	3	4
5) просп. Соборності, 6	3	3	2	1	0	1	1	2	3
6) просп. Молоді, 4	3	3	2	2	1	0	1	1	2
7) вул. Дубнівська, 12	3	3	2	2	1	1	0	2	2
8) вул. Домни Гондіюка, 35	4	4	2	3	2	1	2	0	2
9) просп. Волі, 8	5	5	3	4	3	2	2	2	0

k=0

	1	2	3	4	5	6	7	8	9
1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9

Рис. 3.5. Початкові дані для знаходження центра графа

Матриця маршрутів залишається незмінною, тому що проміжні пункти відсутні та точки з'єднані. Перевірка елементів матриці є ітераційним процесом. Виконуючи останню ітерацію, визначимо дані для знаходження центра графа − вершини, максимальна відстань між якою і будь-якою іншою вершиною є найменшою з усіх можливих (рис. 3.6).

	1	2	3	4	5	6	7	8	9		max		1	2	3	4	5	6	7	8	9
1) вул. Шевченка,11	0	5	5	3	3	3	3	4	5		5		1	1	1	1	1	1	1	1	1
2) вул. Стрілецька, 31	5	0	3	3	3	3	3	4	5		5		2	2	2	2	2	2	2	2	2
3) вул. Липинська, 11	5	3	0	2	2	2	2	2	3		5		3	3	3	3	3	3	3	3	3
4) проп. Відродження, 5	3	3	2	0	1	2	2	3	4		4		4	4	4	4	4	4	4	4	4
5) просп. Соборності, 6	3	3	2	1	0	1	1	2	3	k=9	3		5	5	5	5	5	5	5	5	5
6) просп. Молоді, 4	3	3	2	2	1	0	1	1	2		3		6	6	6	6	6	6	6	6	6
7) вул. Дубнівська, 12	3	3	2	2	1	1	0	2	2		3		7	7	7	7	7	7	7	7	7
8) вул. Домни Гондюка, 35	4	4	2	3	2	1	2	0	2		4		8	8	8	8	8	8	8	8	8
9) просп. Волі, 8	5	5	3	4	3	2	2	2	0		5		9	9	9	9	9	9	9	9	9

Рис. 3.6. Обчислення найменших відстаней графа

За результатами розрахунків центром графа можуть бути три торговельні об'єкти, що розміщені по просп. Соборності, 6, просп. Молоді, 4 та вул. Дубнівська, 12 тому що вони знаходяться на мінімальній відстані від інших об'єктів.

У випадку розміщення складу потрібно звернути увагу на географічне розміщення торговельних точок. Об'єкти по просп. Соборності, 6 та вул. Дубнівська, 12 розміщено в середовищі скупчень торговельних підприємств, тому орендна плата буде вищою, ніж на просп. Молоді, 4. Також треба врахувати доступність та зручність доставок до складських приміщень.

Оскільки мережа торговельних точок не є досить великою, то для її функціонування достатньо буде 1-го складу.

Для подальшого дослідження застосовано алгоритм 3 (знаходження медіани графа). Медіана графа – така вершина *x*, сумарна відстань від якої до всіх інших вершин графа мінімальна. Сумарна відстань від вершини до всіх інших вершин – *CBB(i)* визначається співвідношенням $CBB(i) = \Sigma c_{i,j}$ – сумарна відстань від вершини *i* до всіх *j* .

$$CBB(x) = min\ \{CBB(i)\}. \qquad (3.\ 9)$$

3.3. Розрахунки пошуку медіани за допомогою алгоритму Флойда.

Зауважимо, що сума значень елементів *i*-го рядка матриці C^N відстаней вершин – вершина дорівнює сумі відстаней від вершини до всіх інших

вершин графа, тобто дорівнює *CBB(i)*. Отже, медіана відповідає будь-якому рядку матриці C^N, для якої сума значень елементів мінімальна. Елементи матриці C^N можуть бути розраховані за допомогою алгоритму Флойда.

Алгоритм пошуку медіани:

Крок 1. Знаходимо C_0 – матрицю, елементами якої є $c_{i,j}$ – довжини найкоротших дуг.

Крок 2. Шукаємо C^N – матрицю довжин найкоротших шляхів між усіма вершинами графа за Флойдом.

Крок 3. Визначаємо *CBB (i).*

Крок 4. З усіх *CBB (i)* вибираємо найменше значення, відповідна вершина і буде медіаною.[3]

За алгоритмом Флойда попередньо було розраховано (рис. 3.6) мінімальні відстані між усіма точками графа, тому для знаходження медіани потрібно знайти лише сумарні відстані по кожному об'єкту та обрати з них мінімальний показник, що відповідно і є медіаною графа (рис. 3.7).

	1	2	3	4	5	6	7	8	9		CBB
1) вул. Шевченка,11	0	5	5	3	3	3	3	4	5		31
2) вул. Стрілецька, 31	5	0	3	3	3	3	3	4	5		29
3) вул. Липинська, 11	5	3	0	2	2	2	2	2	3		21
4) проп. Відродження, 5	3	3	2	0	1	2	2	3	4		20
5) просп. Соборності, 6	3	3	2	1	0	1	1	2	3	k=9	16
6) просп. Молоді, 4	3	3	2	2	1	0	1	1	2		15
7) вул. Дубнівська, 12	3	3	2	2	1	1	0	2	2		16
8) вул. Домни Гондюка, 35	4	4	2	3	2	1	2	0	2		20
9) просп. Волі, 8	5	5	3	4	3	2	2	2	0		26

	1	2	3	4	5	6	7	8	9
1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9

Рис. 3.7. Медіана рафа

Медіаною графа є така вершина x_i, для якої $CBB(x) = min\{CBB(i)\}$. Для торгової мережі приватного підприємства об'єкт за адресою просп. Молоді, 4 має мінімальне значення *CBB (6)= 15* (рис. 3.8).

Таким чином, об'єкт по просп. Молоді, 4 має перевагу над попередньо розглянутими торговельними точками, тому що він має оптимальне розміщення стосовно торговельних об'єктів мережі.

Розрахунки показали, що за трьома алгоритмами: метод найменшої домінуючої множини графа, мінімальних відстаней і маршрутів між об'єктами та медіани графа.

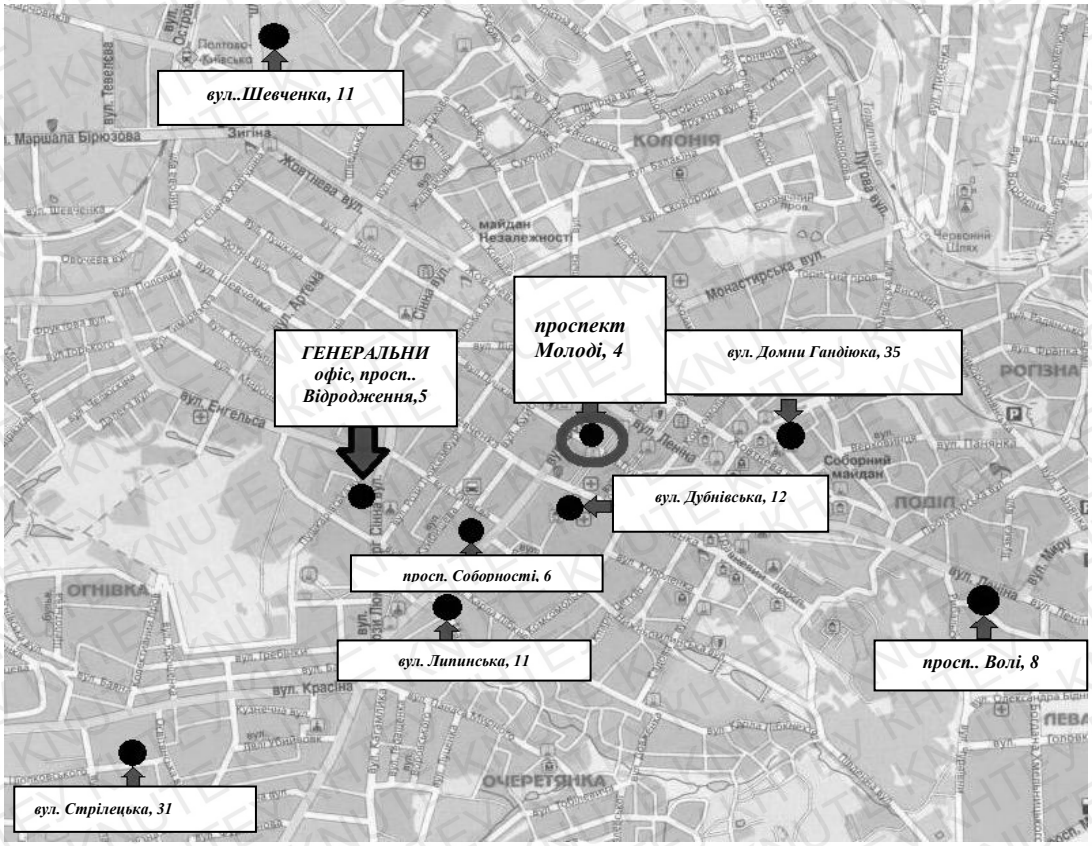

Рис. 3.8. Медіана графа безпосередньо на карті торговельної мережі «Nash Kraj» у місті Луцьк

Рекомендовано для розміщення складського приміщення використовувати торговий об'єкт за адресою: просп. Молоді, 4.

3.4. Програмний продукт «Інформаційна система оптимізації доставки продукції» торговельної мережі «Nash Kraj»

Широке коло оптимізаційних задач, дані яких однозначно подаються за допомогою масиву вершин та дуг графа, що їх сполучають, можна розв'язувати за визначеними алгоритмами теорії графів з допомогою програмних продуктів пошуку оптимальних рішень. Програмний продукт дозволяє автоматизовувати процес пошуку найкоротших шляхів між усіма

вершинами графа за допомогою алгоритма Флойда; пошук шляху комівояжера (базується на використанні удосконаленого алгоритму Флойда) та передбачає можливість динамічної зміни структури графа у визначеному часовому просторі за рахунок впровадження алгоритмів побудови оптимізованого за довжиною укладання дерева. Комбінований метод пошуку оптимальних рішень дозволяє однозначно алгоритмізувати процеси пошуку найкоротших шляхів між визначеною кількістю вершин графа.. Крім того, додатково може закладатися початкова умова визначення шляху комівояжера, тобто обов'язковим є проходження шуканого оптимального шляху через усі вершини графа. У реальних умовах виконання задачі часто відбуваються динамічні зміни самої структури графа, що, звісно, може привести до зміни очікуваних результатів. Наявність динамічних змін структури графа обумовлює проведенням додаткових перевірок отриманих розв'язків на предмет їх оптимальності за умов впровадження уставлених режимів.

Програмний продукт «Інформаційна система оптимізації доставки продукції» торговельної мережі «Nash Kraj» пошуку оптимального рішення передбачає реалізацію множини підходів до отримання вхідних даних, вибір алгоритмів оптимізації графа, виведення результатів пошуку оптимального рішення (рис. 3.9).

Блок введення даних дозволяє користувачеві отримувати вхідні дані одним із трьох способів:

- введення набору репрезентативних даних (дані можна ввести за допомогою двохвимірної матриці, яка математично забезпечує опис графа); - графічне введення даних (дозволяє задати вершини графа і довжини дуг за допомогою графічного інтерфейсу програми та передбачає використання графічних засобів автоматизованої ідентифікації математичної моделі графа);

- введення даних з файлу (забезпечує можливість конвертування і обробки даних, що зберігаються у файлі з визначеним форматом даних).

Блок вибору алгоритму розв'язання оптимізаційної задачі передбачає:

- пошук найкоротшого шляху між двома вершинами графа;
- пошук найкоротших шляхів між усіма вершинами графа;
- пошук шляху комівояжера.

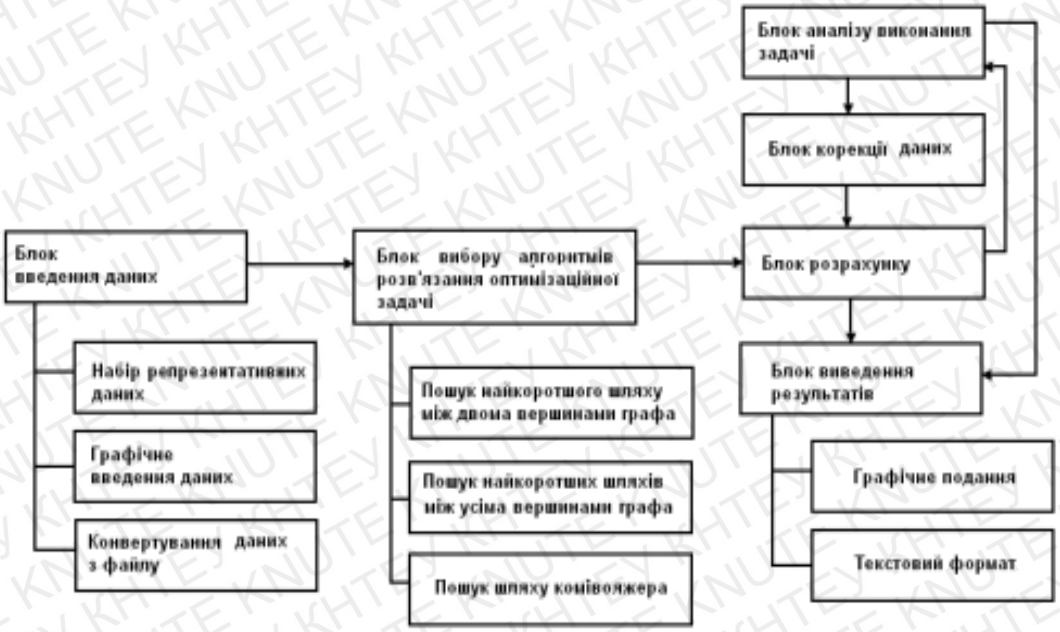

Рис. 3.9. Модель програмного продукту пошуку оптимальних рішень

Блок розрахунку виконує алгоритмізовану обробку даних у процесі пошуку оптимальних рішень. Блоку аналізу виконання задачі аналізує стан задачі на предмет виконання усіх вхідних вимог. Блок корекції даних забезпечує можливість динамічної зміни структури графа шляхом корегування вхідних даних. Блок виведення результатів забезпечує подання інформації у зручному форматі.

Для реалізації задачі «Інформаційної системи оптимізації доставки продукції» торговельної мережі «Nash Kraj» в програмний продукт вирішено використовувати стандартний підхід відображення графів у матрицях. Нижче

наведено приклад роботи програмного продуту який вирішує поставлене завдання.

Фреймворк SwiftPlot - це міжплатформна бібліотека, яка дозволяє вбудовувати графіки в Swift. Існуючі схеми побудови графіків Swift (наприклад, CorePlot) працюють лише на iOS або Mac. Ідея SwiftPlot полягає у створенні міжплатформенної бібліотеки, яка працює на iOS, Mac, Linux та Windows.

На даний момент SwiftPlot використовує три бекенди візуалізації для створення графіків:

1. Бібліотека візуалізації C ++ Anti-GrainGeometry (AGG)
2. Простий візуалізатор SVG
3. ВізуалізаторCoreGraphics з підтримкою macOS, iOS, watchOS і tvOS

Для кодування графів як зображень у форматі PNG він використовує бібліотеку lodepng. SwiftPlot також можна використовувати в блокнотах Jupyter, з підтримкою взаємодії Python для GoogleColab.Приклади, що демонструють усі функції, включені до сховища під каталогом Tests / SwiftPlotTests.

Для підключення бібліотеки потрібно додати бібліотеку до залежностей ваших проектів у файлі Package.swift, як показано нижче залежності:

[

.package (url: "https://github.com/swiftpilit/swiftplot.git", з: "2.0.0")),

],

Додаються ці рядки до першої комірки:

%install-swiftpm-flags -Xcc -isystem/usr/include/freetype2 -Xswiftc -lfreetype

%install '.package(url: "https://github.com/IBM-Swift/BlueCryptor.git", from: "1.0.28")' Cryptor

```
%install '.package(url:
"https://github.com/KarthikRIyer/swiftplot", from: "2.0.0")'
SwiftPlotAGGRenderer
```

Для того, щоб відобразити згенерований сюжет у блокноті, додають цей рядок у нову комірку:

```
%include "EnableJupyterDisplay.swift"
```

Якщо необхідно відобразити згенерований граф у середовищі GoogleColab, додаються ці рядки в нову комірку:

```
import Python
%include "EnableIPythonDisplay.swift"
funcdisplay(base64EncodedPNG: String) {
 let displayImage = Python.import("IPython.display")
 let codecs = Python.import("codecs")
 let imageData = codecs.decode(Python.bytes(base64EncodedPNG,
encoding: "utf8"), encoding: "base64")
displayImage.Image(data: imageData, format: "png").display()
```

Щоб використовувати бібліотеку у своєму пакеті, необхідно ключити її як залежність від цілиу файл Package.swift.

```
import SwiftPlot
import AGGRenderer
let x:[Float] = [10,100,263,489]
let y:[Float] = [10,120,500,800]
var agg_renderer: AGGRenderer = AGGRenderer()
var lineGraph = LineGraph<Float,Float>(enablePrimaryAxisGrid: true)
lineGraph.addSeries(x, y, label: "Plot 1", color: .lightBlue)
lineGraph.plotTitle.title = "SINGLE SERIES"
lineGraph.plotLabel.xLabel = "X-AXIS"
```

```
lineGraph.plotLabel.yLabel = "Y-AXIS"
lineGraph.plotLineThickness = 3.0
lineGraph.drawGraphAndOutput(fileName: filePath+"agg/"+fileName,
renderer: agg_renderer)
```

Для графу вирішення оптимізаційної задачі, яка наведена в попереньому підрозділі, потрібно виконати наступний код:

```
import SwiftPlot
    import AGGRenderer
    import SVGRenderer

    let x1:[Float] = [0,100,263,489]
    let y1:[Float] = [0,320,310,170]
    let x2:[Float] = [0,50,113,250]
    let y2:[Float] = [0,20,100,170]

    var agg_renderer: AGGRenderer = AGGRenderer()
    var lineGraph = LineGraph<Float,Float>(enablePrimaryAxisGrid: true)
    lineGraph.addSeries(x1, y1, label: "Plot 1", color: .lightBlue)
    lineGraph.addSeries(x2, y2, label: "Plot 2", color: .orange)
    lineGraph.plotTitle.title = "MULTIPLE SERIES"
    lineGraph.plotLabel.xLabel = "X-AXIS"
    lineGraph.plotlabel.yLabel = "Y-AXIS"
    lineGraph.plotLineThickness = 3.0
    lineGraph.drawGraphAndOutput(fileName: filePath+"agg/"+fileName,
renderer: agg_renderer)
```

Для розрахунку графа в програмному додатку відкриємо головну сторінку додатку:

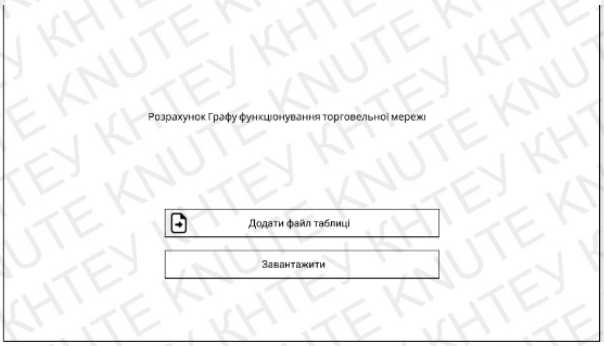

Рис. 3.10. Головне вікно додатку.

Розрахунки проходять згідно заданим параметрам та відповідно до заданих алгоритмів розрахунку які були описані вище.

Після задання даних та розрахунку результату граф як на рисунку (додати точне число) який накладається на карту для отримання точного розуміння та відображення реальної картини розрахунку.

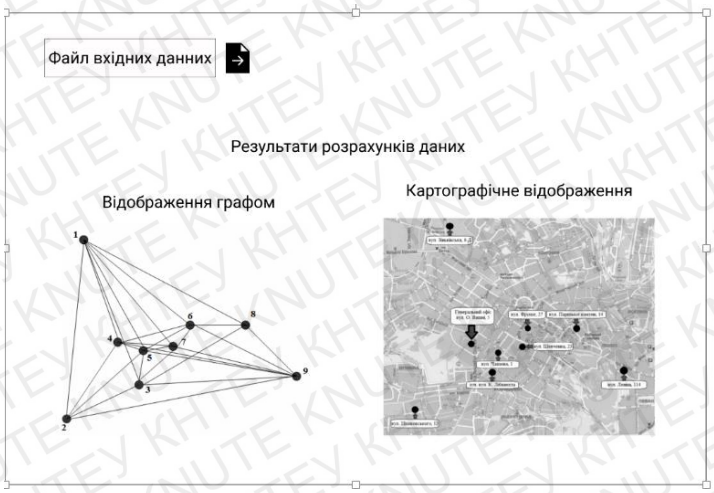

Рис. 3.11. Відображення результатів розрахунку в додатку

Тим самим ми можемо зробити висновок що для оптимізації графу торгової мережі потрібно велика кількість вхідної інформації яка подається у вигляді матриці для більш точного розрахунку даних.

Граф, що відображений по точках карти:

```
import SwiftPlot
import AGGRenderer

let x:[Float] = [10,100,263,489]
let y:[Float] = [10,120,500,800]

var agg_renderer: AGGRenderer = AGGRenderer()
var subplot = SubPlot(layout: .horizontal)

var lineGraph1 = LineGraph<Float,Float>(enablePrimaryAxisGrid:
true)
lineGraph1.addSeries(x, y, label: "Plot 1", color: .lightBlue)
lineGraph1.plotTitle.title = "PLOT 1"
lineGraph1.plotLabel.xLabel = "X-AXIS"
lineGraph1.plotLabel.yLabel = "Y-AXIS"
lineGraph1.plotLineThickness = 3.0

var lineGraph2 = LineGraph<Float,Float>(enablePrimaryAxisGrid:
true)
lineGraph2.addSeries(x, y, label: "Plot 2", color: .orange)
lineGraph2.plotTitle.title = "PLOT 2"
lineGraph2.plotLabel.xLabel = "X-AXIS"
lineGraph2.plotLabel.yLabel = "Y-AXIS"
lineGraph2.plotLineThickness = 3.0
```

subplot.plots = [lineGraph1, lineGraph2]

subPlot.drawGraphAndOutput(fileName: "subPlotsHorizontallyStacked", renderer: agg_renderer)

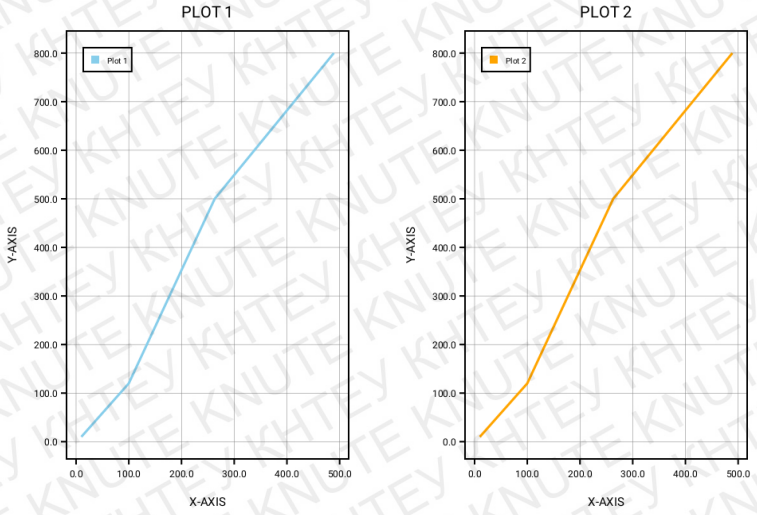

Рис. 3.12. Приклад двох графів розташованих горизонтально.

Використання вторинної осі в лінійному графіку

importSwiftPlot

importAGGRenderer

let x:[Float] = [10,100,263,489]

let y:[Float] = [10,120,500,800]

let x1:[Float] = [100,200,361,672]

let y1:[Float] = [150,250,628,800]

varagg_renderer: AGGRenderer = AGGRenderer()

varlineGraph = LineGraph<Float,Float>()

lineGraph.addSeries(x1, y1, label: "Plot 1", color: .lightBlue, axisType: .primaryAxis)

lineGraph.addSeries(x, y, label: "Plot 2", color: .orange, axisType: .secondaryAxis)

lineGraph.plotTitle.title = "SECONDARY AXIS"

lineGraph.plotLabel.xLabel = "X-AXIS"

lineGraph.plotLabel.yLabel = "Y-AXIS"

lineGraph.plotLineThickness = 3.0

lineGraph.drawGraphAndOutput(fileName: filePath+"agg/"+fileName, renderer: agg_renderer)

Серії, побудовані на вторинній осі, намальовані пунктиром.Ви можете відображати графіки в блокноті Jupyter, використовуючи лише AGGRenderer. Для цього створіть графіки, як показано у наведених вище прикладах, і замість того, щоб використовувати функцію drawGraphAndOutput від LineGraph, використовуйте функцію drawGraph, потім отримайте закодоване зображення base64 від AGGRenderer і передайте його функції відображення, як showm нижче:

lineGraph.drawGraph(renderer: agg_renderer)

display(base64EncodedPNG: agg_renderer.base64Png())

3.5. Висновки до розділу 3

Вирішення задач оптимізації за допомогою теорії графів передбачає, що має бути з’єднання розбитих на ортогональних відрізків, які повинні розташовуватись таким чином щоб уникати перетину з вузлами графа на його візуальному відображенні. Крім цього, необхідно реалізувати бібліотеку візуалізації графів, щоб наглядно на прикладах показати виконання поставлених задач та задоволення критеріїв оптимальності.

Перевага таких алгоритмів полягає в тому, що вони можуть використовуватися для будь-яких мереж: лікарень, пожежних, поліцейських відділків, аптек та магазинів різного профілю.

Визначення графа є настільки загальним, що цим терміном можна описувати безліч подій та об’єктів повсякденного життя. Високий рівень абстракції й узагальнення дозволяє використовувати типові алгоритми теорії графів для вирішення зовнішньо несхожих задач у транспортних і комп’ютерних мережах, будівельному проектуванні, молекулярному моделюванні тощо.

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

В ході виконання випускного кваліфікаційного проекту проведено порівняльний аналіз текстового та візуального програмування, наведено базову термінологію теорії графів. Також досліджено та проведено аналіз існуючих алгоритмів візуалізації графів. Було розглянуто особливості кожного алгоритму, наведено їх переваги та недоліки. Досліджено проблеми, які виникають при візуалізації графів та існуючі підходи до візуалізації графів. На основі отриманих знань обґрунтовано актуальність оптимізації швидкості знаходження найкоротших шляхів між вузлами направленого графа.

Слід зазначити, що розташування вершин і ребер графа не є простою задачею і може вимагати значної кількості витраченого часу в ручному режимі уже для графа із порядком 10 (порядок графа – кількість вершин). Саме тому візуалізація графів і алгоритмів на графах є актуальним завданням. Одними з основних критеріїв ефективності алгоритму побудови зображення графа є:

- час візуалізації графа з великою кількістю вузлів і з'єднань (більше 500 вузлів і 1000 з'єднань) для ефективної взаємодії з користувачем;
- мінімальна кількість накладань ребер графа та мінімальна довжина з'єднань між вершинами графа.

На сьогодні існує безліч алгоритмів та бібліотек для візуалізації графа, проте дані рішення не відповідають вищевказаним критеріям.

У зв’язку з цим запропоновано алгоритм візуалізації з'єднань графа який оптимізує візуалізації графів/діаграм обчислень.

При розробці нового алгоритму необхідно визначити критерії оптимальності, тобто деякі показники, при яких можна сказати що алгоритм працює правильно і задовольняє потреби. Критерій оптимальності – характерний показник розв’язку задачі, по значенню якого оцінюється оптимальність знайденого розв’язку (максимальна наближеність до поставлених вимог). Правильний вибір критерії оптимальності відіграє суттєву роль в виборі оптимального вирішення поставлених задач

Вирішено задача щодо оптимізації розміщення елементів торговельної мережі «Nash Kraj» у місті Луцьк за допомогою теорії графів, і тим самим доведено ефективність запропонованого методу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Введение в .NET [Электронный ресурс]. – Режим доступа: ttp://mindberg.blogspot.com/2006/08/net.html [Введение в .NET - интернет блог] – Назв. с экрана..
2. Спекторський І.Я. Дискретна математика / І. Я. Спекторський. – 2004. – 220 с.
3. Алгоритмы на графах – Интернет университет информационных технологий [Электронный ресурс]. – Режим доступа: http://www.intuit.ru/department/algorithms /algocombi/16/1.html
4. Thomas H. Cormen. CliffordStein "22.2", IntroductiontoAlgorithms / Thomas H. Cormen, E. Charles Leiserson, RonaldL. Rivest. – 2001 (англ.). – 531 с.
5. Ананий В. Глава 9. Жадные методы: Алгоритм Дейкстры / В. Ананий // Алгоритмы: введение в разработку и анализ. – Introduction to The Design and Analysis of Aigorithms. – М.: "Вильямс", 2006. – С. 189 – 195.
6. Ананий В. Алгоритмы: введение в разработку и анализ / В. Ананий. – М.: Вильямс, 2006. – 548 с.
7. Пауэрс Л. Microsoft Visual Studio 2008 Unleashed by Lars Powers and Mike Snell. / Л. Пауэрс, М. Снелл — С.Пб.: "БХВ-Петербург", 2008. – 1200 с.

8. Выбор между визуальным и текстовым программированием для детей [Електронний ресурс]: веб-ресурс habr. – Режим доступу https://habr.com/post/400495/
9. Graph [Електронний ресурс]: веб-сторінка онлайн довідника Вікіпедія. – Режим доступу https://en.wikipedia.org/wiki/Graph(discrete_mathematics)
10. Directed acyclic graph [Електронний ресурс]: веб-сторінка онлайн довідника Вікіпедія. – Режим доступу https://en.wikipedia.org/wiki/Directed_acyclic_graph
11. Circle graph [Електронний ресурс]: веб-сторінка онлайн довідника Вікіпедія. – Режим доступу https://en.wikipedia.org/wiki/Circle_graph
12. Arc diagram [Електронний ресурс]: веб-сторінка онлайн довідника Вікіпедія. – Режим доступу https://en.wikipedia.org/wiki/Arc_diagram
13. Heer Jeffrey. A tour through the visualization zoo / Heer Jeffrey; Bostock Michael; Ogievetsky Vadim – Communications of the ACM, 2010. – С. 59-67.
14. Force-directed graph drawing [Електронний ресурс]: веб-сторінка онлайн довідника Вікіпедія. – Режим доступу https://en.wikipedia.org/wiki/Forcedirected_graph_drawing
15. Візуалізація графів [Електронний ресурс]: веб-ресурс science-community. – Режим доступу https://www.science-community.org/en/node/5582
16. Построение изображений ациклических ориентированных графов [Електронний ресурс]: веб-ресурс rain.ifmo.ru. – Режим доступу http://rain.ifmo.ru/cat/view.php/theory/graph-coloring-layout/dagplanarization-2007

17. Dijkstra's algorithm [Електронний ресурс]: веб-сторінка онлайн довідника Вікіпедія. – Режим доступу https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm 82 12.A* search algorithm [Електронний ресурс]: веб-сторінка онлайн довідника Вікіпедія. – Режим доступу https://en.wikipedia.org/wiki/A*_search_algorithm

ТЕХНІЧНЕ ЗАВДАННЯ

Однією з найголовніших запорук відповідності вимогам та високої якості розробленого програмного рішення є грамотно спроектоване до початку розробки технічне завдання. Отже, далі розкриємо невід'ємні положення технічного завдання до розробки Android-додатку для імітації тачпаду ПК.

1. Загальні відомості

Назва проекту: MobileMouse.

Планові строки розробки: серпень 2020 – вересень 2020.

Потенційні користувачі: користувачі стаціонарних ПК з встановленою ОС Windows.

Призначення програмного рішення: надати можливість користувачу ПК проводити процесс комбінування та автоматизації методів пошуку оптимальних рішень, які базуються на використанні засобів теорії графів

Мета створення програмного рішення: опанування сучасних мов програмуванняф та їх закріплення на практиці.

2. Функціональні вимоги до додатку

Пошук оптимальних рішень на основі теорії графів

3. Вимоги до програмного забезпечення

Розробка серверної частини програмного рішення здійснюватиметься із використанням мови програмування Python, у середовищі розробки SwiftPlot.

					КНТЕУ 121 06-04.МР			
Зм.	*Аркуш*	*№ докум.*	*Підпис*	*Дата*				
Зав. каф.		Криворучко О.В.		28.02.20	Розробка інформаційної ситеми відслідкування потенційних клієнтів фірми	*Стадія*	*Аркуш*	*Аркушів*
Керівник		Криворучко О.В.		28.02.20		*ТЗ*	*60*	*61*
Гарант		Криворучко О.В.		28.02.20		Факультет інформаційних технологій 2м курс, 6 група		
Розробив		Гаврилюк Я.М.		28.02.20	*Технічне завдання*			

Розробка мобільного додатку (клієнтської частини програмного рішення) забезпечуватиметься наступним:

- мова програмування Java;
- мова розмітки XML;
- інструментарій автоматизації збірки Gradle;
- середовище розробки Android Studio.

4. Вимоги до технічного забезпечення

Найважливішою вимогою до задіяних пристроїв є належність до однієї локальної мережі.

Серверна частина для ПК має виконуватися за наступних мінімальних конфігурацій:

- Microsoft Windows 10/8/7.
- RAM: 1 Гб і більше.
- JDK 1.6 і вище.
- Передвстановлене середовище виконання Python-сценаріїв із можливістю одночасного перегляду консолі під час виконання.

Клієнтська частина (мобільний додаток) має виконуватися на смартфоні із наступними конфігураціями:

- Android 10/9/8/7/6/5;
- RAM: 2 Гб і більш

Київський національний торговельно-економічний університет
Кафедра інженерії програмного забезпечення та кібербезпеки

РЕФЕРАТ

ВИПУСКНИЙ КВАЛІФІКАЦІЙНИЙ ПРОЕКТ

на тему:
«Розробка інформаційної системи від слідкування потенційних клієнтів фірми»

Студента 2м курсу, 6 групи, спеціальності 121 «Інженерія програмного забезпечення» спеціалізації «Інженерія програмного забезпечення» _____ підпис студента — Гаврилюк Яни Миколаївни

Науковий керівник доктор технічних наук, професор, завідувач кафедри інженерії програмного забезпечення та кібербезпеки _____ Підпис керівника — Криворучко Олена Володимирівна

Київ -2020

ВСТУП

У сучасних умовах господарювання важливим є внесок кожної сфери економічної діяльності в розвиток національної економіки. Поряд з виробничими галузями не меншу роль відіграє торгівля, завдяки якій підтримується збалансованість виробництва і споживання, формується суттєва частка валової доданої вартості в Україні, забезпечується робочими місцями економічно активне населення.

Проте на споживчому ринку пропозиція частіше перевищує попит. Торговельні об'єкти ведуть запеклу боротьбу за клієнтів, намагаючись зацікавити покупця будь-яким можливим способом. Виникають нові форми супермаркетів, покликані задовольнити найвибагливіших відвідувачів. Звичайним торговим павільйонам стає все складніше виживати в таких умовах. Підприємці, які мають власну мережу магазинів, усе більше зіштовхуються з проблемою раціонального розміщення торговельних точок, складських приміщень, що є одним з вагомих факторів ведення конкурентної боротьби. Знайти розв'язок задач оптимального розміщення торгових об'єктів можна за допомогою мереж або графів.

Постановка завдання. Завдання роботи полягає у вирішенні питання застосування теорії графів, її методів та інструментів для оптимізації розміщення торгівельної мережі для збільшення потенційних покупців, тобто для зменшення витрат на транспортні шляхи до мережі магазинів.

Мета роботи - комбінування та автоматизація методів пошуку оптимальних рішень, які базуються на використанні засобів теорії графів.

Під *об'єктом дослідження випускного кваліфікаційного проекту* розуміємо задачі пошуку оптимальних рішень.

Предметом дослідження випускного кваліфікаційного проекту постають можливості застосування засобів теорії графів у процесі розробки програмного продукту пошуку оптимальних рішень.

Основними *задачами є* розробка комбінованого методу пошуку оптимальних рішень з використанням теорії графів та його програмну реалізацію розв'язання оптимізаційних задач.

Методи, які застосовано для розв'язання задач: основи теорії графів, системний підхід для аналізу інформації, метод параметричної оптимізації для оптимізації алгоритму, структурно-функціональний метод для здійснення постановки задачі.

РОЗДІЛ 1.
ДОСЛІДЖЕННЯ МЕТОДІВ ПРОГРАМУВАННЯ ТА ТЕОРІЇ ГРАФІВ ПОШУКУ ОПТИМАЛЬНИХ РІШЕНЬ

Термін "граф" вперше з'явився в науковій літературі в 1936 році в роботах угорського математика Д. Кьоніга, хоча елементи теорії графів були відомі та широко використовувались ще у XVIII столітті, зокрема в роботах Л. Ейлера.

Незалежно від уподобань, з графами, мабуть, зустрічався кожний. Розуміючи під графом певну множину точок, деякі з яких з'єднані лініями, цей незвичний математичний об'єкт можна без перебільшення застосовувати в будь-якій науковій чи практичній галузі.

Провівши порівняльний аналіз текстового та візуального програмування, наведено базову термінологію теорії графів. Досліджено проблеми, які виникають при візуалізації графів. На основі цих проблем визначено мету кваліфікаційного випускного проекту та здійснено постановку задач. В результаті поставлено задачі, основною метою яких є створення алгоритму візуалізації з'єднань графа, який буде виконувати поставлену мету та задовольнятиме критеріям оптимальності.

РОЗДІЛ 2.
АЛГОРИТМИ ВІЗУАЛІЗАЦІЇ РІЗНОМАНІТНИХ ГРАФІВ

Існує багато алгоритмів для візуалізації графів. Розглянемо найпоширеніші з них, виділимо їх переваги та недоліки.

Для оптимізації алгоритму доцільно використати метод параметричної оптимізації. Параметрична оптимізація – процес знаходження значень параметрів, при яких досягається максимальна ефективність виконання процедури або забезпечуються найкращі характеристики пристрою або системи з точки зору встановленого критерію. За кількістю параметрів, які оптимізуються, задача оптимізації може бути однопараметричною чи багатопараметричною, за кількістю критеріїв – однокритеріальною чи багатокритеріальною. Виходячи із поставлених задач до випускного кваліфікаційного проекту та сформульованих критеріїв оптимальності, можна зробити висновок, що задача є багато параметричною та багатокритеріальною.

РОЗДІЛ 3.
ФУНКЦІОНУВАННЯ ТОРГОВЕЛЬНОЇ МЕРЕЖІ ЗА ДОПОМОГОЮ ТЕОРІЇ ГРАФІВ

Завдання вирішення проблеми полягає у вирішенні питання застосування теорії графів, її методів та інструментів для оптимізації розміщення торгівельної мережі для зменшення витрат на транспортування товарів, які користуються попитом.

Для досягнення оптимального розміщення складського приміщення та визначення найкоротших маршрутів транспортування товару в межах торговельної мережі торгівлі «NashKraj» у місті Луцьк для задоволення попиту покупців на той чи інший товар застосовано алгоритми визначення:

1) найменшої домінуючої множини графа,

2) мінімальних відстаней та маршрутів між об’єктами (алгоритм Флойда);

3) медіани графа.

Рис.3.1. Мережа підприємства торговельної мережі «NashKraj» умісті Луцьк.

У загальному вигляді доцільно зобразити цю ситуацію у вигляді графа, який складається з точок (вершин), що відображають основні елементи ситуації, та лінії (ребра), які з’єднують пари цих вершин та відображають зв’язки між ними.

Визначення найменшої домінуючої множини графа (найменшої зовнішньої стійкості графа) дозволяє знайти об’єкти, які можуть бути використаними як складське приміщення [1].

Найменшою домінуючою множиною графа називається множина $R \subseteq V$, яка містить мінімальну кількість елементів і задовольняє умову:

$$(\forall \nu_J \in V - R)(\exists \nu_j \in R)(s_{ij} = 1) \qquad (3.1)$$

Граф $G=(V,S)$, надається множиною вершин $V=\{V_1, V_2, \ldots, V_n\}$ і матрицею суміжності $S=[S_{ij}]$ порядку n, де:

$$s_{ij} = \begin{cases} l, \nu_{ij} \text{ т } \nu_j \text{ - з'єднані ребром;} \\ 0 \text{ - у іншому випадку.} \end{cases} \qquad (3.2)$$

Цілочислові змінні – x_{ij}, $i = \overline{I,n}$, $j = \overline{I,n}$, *де*

$$x_{ij} = \begin{cases} 1, \text{вершина } \nu_i \text{ входить у шукану множину;} \\ 0 \text{ - у іншому випадку} \end{cases} \qquad (3.3)$$

Цільова функція має вигляд:

$$\sum_{i=I}^{n} x_i \to \min \qquad (3.4)$$

1. Кожна вершина графа, що не входить у домінуючу множину, повинна бути видимою із його вершини хоча б один раз:

$$\sum_{i=I}^{n} S_{iR} x_{ik} \geq 1; \; k = \overline{I,n} \qquad (3.5)$$

2. Область допустимих значень:

Вершинами графа є підприємства - об’єкти мережі торгівлі «NashKraj» у місті Луцьк, а ребрами − маршрути (рис.3.2)

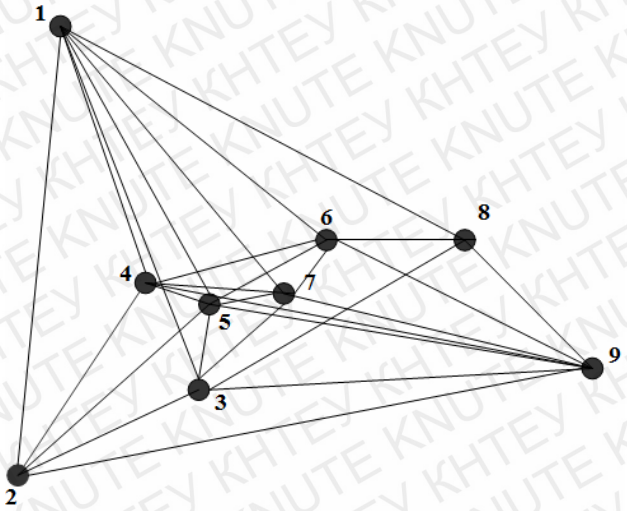

Рис. 3.2. Граф торговельної мережі «NashKraj» у місті Луцьк

Оскільки це торговельна мережа, то вагомим етапом для поліпшення її функціонування є розрахунок відстаней маршрутів, саме вони дозволять повністю дослідити існуючу мережу та виробити подальші рекомендації щодо оптимального розміщення торгових об’єктів. Цю вимогу можна задовольнити за допомогою обчислення мінімальних відстаней та маршрутів за алгоритмом Флойда.

Розрахунки показали, що за трьома алгоритмами: метод найменшої домінуючої множини графа, мінімальних відстаней і маршрутів між об’єктами та медіани графа рекомендовано для розміщення складського приміщення використовувати торговий об’єкт за однією і тією ж адресою: просп. Молоді, 4.

Для реалізації задачі «Інформаційної системи оптимізації доставки продукції» торговельної мережі «NashKraj» в програмний продукт вирішено використовувати стандартний підхід відображення графів у матрицях.

Для кодування графів як зображень у форматі PNG він використовує бібліотеку lodepng. SwiftPlot також можна використовувати в блокнотах Jupyter, з підтримкою взаємодії Python для GoogleColab. Приклади, що

демонструють усі функції, включені до сховища під каталогом Tests / SwiftPlotTests.

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

В ході виконання випускного кваліфікаційного проекту проведено порівняльний аналіз текстового та візуального програмування, наведено базову термінологію теорії графів. Також досліджено та проведено аналіз існуючих алгоритмів візуалізації графів. Було розглянуто особливості кожного алгоритму, наведено їх переваги та недоліки. Досліджено проблеми, які виникають при візуалізації графів та існуючі підходи до візуалізації графів. На основі отриманих знань обґрунтовано актуальність оптимізації швидкості знаходження найкоротших шляхів між вузлами направленого графа.

Слід зазначити, що розташування вершин і ребер графа не є простою задачею і може вимагати значної кількості витраченого часу в ручному режимі уже для графа із порядком10 (порядок графа–кількість вершин). Саме тому візуалізація графів і алгоритмів на графах є актуальнимзавданням. Одними з основних критеріїв ефективності алгоритму побудови зображення графає:

- час візуалізації графа з великою кількістю вузлів і з’єднань (більше 500 вузлів і 1000 з’єднань) для ефективної взаємодії зкористувачем;
- мінімальна кількість накладань ребер графа та мінімальна довжина з’єднань між вершинами графа.

На сьогодні існує безліч алгоритмів та бібліотек для візуалізації графа, проте дані рішення не відповідають вищевказаним критеріям.

У зв’язку з цим запропоновано алгоритм візуалізації з'єднань графа який оптимізує візуалізації графів/діаграм обчислень.

При розробці нового алгоритму необхідно визначити критерії оптимальності, тобто деякі показники, при яких можна сказати що алгоритм працює правильно і задовольняє потреби. Критерій оптимальності – характерний показник розв’язку задачі, по значенню якого оцінюється оптимальність знайденого розв’язку (максимальна наближеність до поставлених вимог). Правильний вибір критерії оптимальності відіграє суттєву роль в виборі оптимального вирішення поставлених задач

Вирішено задача щодо оптимізації розміщення елементів торговельної мережі «NashKraj» у місті Луцьк за допомогою теорії графів, і тим самим доведено ефективність запропонованого методу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Введение в .NET [Электронный ресурс]. – Режим доступа: ttp://mindberg.blogspot.com/2006/08/net.html [Введение в .NET - интернет блог] – Назв. с экрана..
2. Спекторський І.Я. Дискретна математика / І.Я. Спекторський. – 2004. – 220 с.
3. Алгоритмы на графах – Интернетуниверситетинформационныхтехнологий [Электронный ресурс]. – Режим доступа: http://www.intuit.ru/department/algorithms /algocombi/16/1.html
4. Thomas H. Cormen. CliffordStein "22.2", IntroductiontoAlgorithms / Thomas H. Cormen, E. CharlesLeiserson, RonaldL. Rivest. – 2001 (англ.). – 531 с.
5. Ананий В. Глава 9. Жадные методы: Алгоритм Дейкстры / В. Ананий // Алгоритмы: введение в разработку и анализ. –

IntroductiontoTheDesignandAnalysisofAigorithms. — M.: "Вильямс", 2006. — С. 189—195.

6. Ананий В. Алгоритмы: введение в разработку и анализ / В. Ананий. — М.: Вильямс, 2006. – 548 с.
7. Пауэрс Л. Microsoft VisualStudio 2008 Unleashedby Lars Powersand MikeSnell. / Л. Пауэрс, М. Снелл — С.Пб.: "БХВ-Петербург", 2008. – 1200 с
8. Visualprogramminglanguage [Електронний ресурс]: веб-сторінка онлайн довідника Вікіпедія. – Режим доступу https://en.wikipedia.org/wiki/ Visual_programming_language.
9. Выбормеждувизуальным и текстовымпрограммированием для детей [Електронний ресурс]: веб-ресурс habr. – Режим доступу https://habr.com/post/400495/
10. Graph [Електронний ресурс]: веб-сторінка онлайн довідника Вікіпедія. – Режим доступу https://en.wikipedia.org/wiki/Graph(discrete_mathematics)
11. Directedacyclicgraph [Електронний ресурс]: веб-сторінка онлайн довідника Вікіпедія. – Режим доступу https://en.wikipedia.org/wiki/Directed_acyclic_graph
12. Circlegraph [Електронний ресурс]: веб-сторінка онлайн довідника Вікіпедія. – Режим доступу https://en.wikipedia.org/wiki/Circle_graph
13. Arcdiagram [Електронний ресурс]: веб-сторінка онлайн довідника Вікіпедія. – Режим доступу https://en.wikipedia.org/wiki/Arc_diagram
14. HeerJeffrey. A tourthroughthevisualizationzoo / HeerJeffrey; BostockMichael; OgievetskyVadim – Communicationsofthe ACM, 2010. – С. 59-67.
15. Force-directedgraphdrawing [Електронний ресурс]: веб-сторінка онлайн довідника Вікіпедія. – Режим доступу

https://en.wikipedia.org/wiki/Forcedirected_graph_drawing

16. Візуалізація графів [Електронний ресурс]: веб-ресурс science-community. – Режим доступу https://www.science-community.org/en/node/5582