Міністерство освіти і науки України
Кам’янець-Подільський національний університет імені Івана Огієнка
Фізико-математичний факультет
Кафедра комп'ютерних наук

Кваліфікаційна робота бакалавра
з теми: **«ОПТИМІЗАЦІЯ ЛОГІСТИЧНИХ МАРШРУТІВ З ДОПОМОГОЮ ГРАФІВ»**

Виконав: здобувач вищої освіти
групи KN1-B21
спеціальності 122 Комп’ютерні науки
Стефура Юрій Олегович

Керівник:
Пилипюк Тетяна Михайлівна, кандидат фізико-математичних наук, доцент
Рецензент:
Громик Андрій Петрович, кандидат технічних наук, доцент

Кам’янець-Подільський – 2025 р.

АНОТАЦІЯ

У кваліфікаційній роботі розглянуто задачу оптимізації логістичних маршрутів доставки піци в межах міста Кам’янець-Подільський із використанням графової моделі дорожньої мережі. Реалізовано мовою Python програмну систему, що базується на алгоритмі Дейкстри для пошуку найкоротших шляхів у зваженому графі, побудованому за допомогою даних OpenStreetMap. Здійснено геокодування введеної користувачем адреси та розраховано три варіанти доставки: найдешевший, найшвидший та оптимальний (за функцією оцінки ціна/час). Візуалізація результатів реалізована у вигляді інтерактивної HTML-карти з маркерами та маршрутами.

Отримані результати підтверджують коректність побудови маршрутів, точність обчислення часу доставки та наочність виводу. Система демонструє гнучкість і масштабованість – її можна адаптувати для інших міст або видів логістики. Результати можуть бути використані для створення систем планування доставки в реальних умовах.

Ключові слова: оптимізація маршрутів, граф, доставка, алгоритм Дейкстри, Python, логістика, геокодування, OpenStreetMap.

ABSTRACT

The bachelor's thesis addresses the problem of optimizing logistics delivery routes in the city of Kamianets-Podilskyi using a graph-based model of the road network. A Python-based software system was developed, employing Dijkstra's algorithm to find the shortest paths in a weighted graph generated from OpenStreetMap data. The system performs geocoding of a user-specified address and calculates three delivery options: cheapest, fastest, and optimal (based on a combined price/time evaluation function). The results are visualized on an interactive HTML map with markers and route lines.

The results confirm the accuracy of route construction, reliability of time estimation, and clarity of the visual representation. The system is flexible and scalable, making it suitable for adaptation to other cities or logistics services. The developed solution can be applied in real-world delivery planning systems.

Keywords: route optimization, graph, delivery, Dijkstra’s algorithm, Python, logistics, geocoding, OpenStreetMap.

ЗМІСТ

ВСТУП

У сучасному світі, де щодня зростає попит на швидку та якісну доставку товарів, питання ефективного управління логістикою стає ключовим для багатьох компаній. Особливо це стосується сфери громадського харчування та електронної комерції, де час виконання замовлення напряму впливає на рівень задоволеності клієнтів. З кожним роком конкуренція у сфері доставки посилюється і компанії змушені шукати нові способи оптимізації своїх логістичних процесів. Одним із таких способів є використання математичних моделей, що дозволяють описати транспортну мережу у вигляді графа та застосовувати алгоритми пошуку найкоротших маршрутів.

Актуальність теми дослідження обумовлюється необхідністю впровадження сучасних комп'ютерних методів оптимізації логістичних процесів, зокрема у сфері міської доставки. Сучасні геоінформаційні сервіси надають великі обсяги просторових даних, а науково обґрунтовані алгоритми дозволяють ефективно їх використовувати для пошуку оптимальних рішень. Це відкриває широкі можливості для автоматизації задач маршрутизації, скорочення часу доставки, зменшення витрат і підвищення загальної ефективності логістичних систем.

Дослідження показують, що впровадження графових методів та сучасних технологій у процес планування маршрутів може призвести до значного скорочення часу доставки та підвищення задоволеності клієнтів. Наприклад, оптимізація маршрутів з урахуванням часових вікон доставки та пріоритетності замовлень дозволяє більш ефективно використовувати ресурси та уникати запізнень. Використання графових методів у малих містах потребує врахування специфіки місцевої інфраструктури. Наприклад, вузькі вулиці або обмеження руху можуть вимагати адаптації стандартних алгоритмів для забезпечення їхньої ефективності. Крім того, інтеграція даних про реальний стан доріг та прогнозування трафіку може покращити точність побудови маршрутів.

Об’єктом дослідження в даній кваліфікаційній роботі є логістичні процеси доставки товарів у межах міста.

Предметом дослідження – методи оптимізації маршрутів на основі графових моделей і алгоритмів пошуку найкоротших шляхів.

Метою даної роботи є розробка програмної системи для оптимізації логістичних маршрутів доставки піци в місті Кам’янець-Подільський на основі графа дорожньої мережі та алгоритмів пошуку оптимального шляху, з урахуванням таких критеріїв, як ціна, час доставки та загальна ефективність.

Для досягнення поставленої мети необхідно вирішити наступні **завдання**:

1. Провести теоретичний аналіз основ логістики та методів оптимізації маршрутів.

2. Формалізувати задачу оптимізації у вигляді зваженого графа.

3. Побудувати математичну модель на основі транспортної мережі міста.

4. Реалізувати програмне забезпечення для побудови маршрутів із використанням алгоритму Дейкстри.

5. Забезпечити введення користувацької адреси та геокодування.

6. Розрахувати три варіанти доставки: найдешевший, найшвидший і оптимальний (за функцією оцінки).

7. Реалізувати візуалізацію результатів на інтерактивній карті.

8. Перевірити коректність роботи системи за допомогою експериментальних сценаріїв.

Для реалізації поставлених завдань використовувалися такі **методи дослідження**: аналіз наукових джерел з логістики та теорії графів; методи побудови графових структур за допомогою бібліотек Python; алгоритм Дейкстри для пошуку найкоротшого шляху у зваженому графі; геокодування адрес за допомогою Geopy; створення візуалізації на основі Folium.

Практичне значення одержаних результатів полягає в можливості застосування розробленої системи для побудови маршрутів доставки в

реальних умовах. Рішення може бути адаптоване до інших міст, змінено під інші види доставки (не лише піца), а також доповнене новими параметрами, такими як врахування завантаженості трафіку, облік годин пік, підрахунок вартості доставки тощо. Система є масштабованою, зручною для користувача та може бути інтегрована у вебсервіси чи мобільні застосунки.

Апробація результатів. Результати досліджень були оприлюднені на науковій конференції за підсумками науково-дослідної роботи здобувачів вищої освіти фізико-математичного факультету Кам'янець-Подільського національного університету імені Івана Огієнка у 2024-2025 н.р., 9-10 квітня 2025 (тези доповіді) [6] та у Віснику Кам'янець-Подільського національного університету імені Івана Огієнка. Фізико-математичні науки. Випуск 17 (стаття) [7].

Структура роботи. Кваліфікаційна робота складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ОПТИМІЗАЦІЇ ЛОГІСТИЧНИХ МАРШРУТІВ

1.1 Сутність та значення логістики в сучасній економіці

Логістика – важливий компонент економічної діяльності, що забезпечує ефективне функціонування систем постачання, збуту та розподілу. Згідно з класичним визначенням, логістика – це наука про планування, реалізацію та контроль потоків матеріальних ресурсів, послуг і супутньої інформації від початкового пункту до кінцевого споживача з метою задоволення потреб за мінімально можливих витрат. У ширшому розумінні логістика охоплює комплекс заходів, спрямованих на забезпечення безперервності матеріальних та інформаційних потоків, підвищення ефективності використання ресурсів і скорочення термінів доставки.

Сучасна логістика – це міждисциплінарна галузь, що об'єднує знання в галузі економіки, менеджменту, інформатики, математики та інженерії. Це дає змогу розробляти і впроваджувати оптимальні рішення з організації поставок, транспортування, складування та інших логістичних процесів.

У сучасній ринковій економіці логістика відіграє ключову роль у забезпеченні ефективної роботи підприємств. За допомогою логістичних рішень компанії можуть оптимізувати витрати на транспортування, зберігання та обробку замовлень, що безпосередньо впливає на їхні фінансові результати.

Логістика також сприяє підвищенню якості обслуговування клієнтів завдяки скороченню термінів доставки, підвищенню надійності поставок і гнучкості реагування на зміни попиту. В умовах конкуренції швидка і надійна доставка товарів і послуг може стати вирішальним фактором у боротьбі за клієнта.

Крім того, логістика дозволяє підприємствам адаптуватися до змін у зовнішньому середовищі, включаючи коливання цін на паливо, зміни в транспортному законодавстві, міжнародну конкуренцію та інші фактори. Вона

допомагає знизити операційні витрати, підвищити прозорість бізнес-процесів і поліпшити взаємодію між усіма учасниками логістичного ланцюга.

Логістика як система складається з низки взаємопов'язаних компонентів або підсистем. До основних з них відносяться такі:

- транспортна логістика – відповідає за планування, організацію та контроль перевезень вантажів різними видами транспорту; це один з найважливіших елементів логістичної системи, оскільки на транспортування припадає значна частина загальних логістичних витрат;
- складська логістика – охоплює процеси приймання, зберігання, обробки та видачі товарів на складах; раціональна організація складування забезпечує ефективне управління запасами та мінімізацію витрат на зберігання;
- інформаційна логістика – забезпечує збір, обробку, зберігання та передачу інформації, необхідної для прийняття логістичних рішень; завдяки сучасним інформаційним системам (ERP, WMS, TMS тощо) досягається прозорість і керованість логістичних процесів;
- управління запасами – спрямоване на визначення оптимального рівня запасів для забезпечення безперебійних поставок без заморожування надлишкового капіталу в продукції;
- логістика закупівель – охоплює планування потреб у матеріалах, вибір постачальників, укладення контрактів і забезпечення своєчасної доставки ресурсів.

Кожна з цих підсистем тісно пов'язана з іншими і функціонує в рамках єдиного логістичного ланцюга, від ефективності якого залежить успіх компанії на ринку.

1.2 Основи транспортної логістики

Транспортна логістика є ключовою підсистемою загальної логістики, яка відповідає за планування, реалізацію та контроль процесів транспортування матеріальних ресурсів у межах логістичного ланцюга. Вона

забезпечує ефективне переміщення товарів від постачальника до споживача шляхом оптимізації маршрутів, вибору транспортних засобів та управління витратами. Транспортна логістика охоплює не тільки фізичне переміщення товарів, а й інтеграцію з іншими логістичними функціями, такими як управління запасами, складуванням та інформаційними потоками.

Основними завданнями транспортної логістики є:

1) вибір оптимальних маршрутів транспортування: аналіз і визначення найбільш ефективних маршрутів доставки з урахуванням відстані, часу, витрат і умов транспортування;
2) вибір виду та типу транспортного засобу: оцінка характеристик вантажу та умов транспортування для визначення найбільш підходящого виду транспорту (автомобільний, залізничний, морський, повітряний) та конкретного типу транспортного засобу;
3) планування графіків перевезень: розробка графіків перевезень для забезпечення своєчасної доставки та оптимального використання транспортних ресурсів;
4) управління транспортними витратами: контроль і оптимізація витрат, пов'язаних із транспортуванням, включно з паливом, технічним обслуговуванням, оплатою праці та іншими операційними витратами;
5) забезпечення безпеки та надійності перевезень: реалізація заходів щодо мінімізації ризиків пошкодження або втрати вантажу, а також дотримання нормативних вимог і стандартів безпеки.

Оцінка ефективності транспортної логістики ґрунтується на ключових показниках: час доставки, транспортні витрати, надійність транспортування, гнучкість логістичної системи, використання транспортних ресурсів. Розглянемо їх.

Час доставки: швидкість і своєчасність транспортування, яка впливає на задоволеність клієнтів і ефективність ланцюжка поставок.

Транспортні витрати: загальні витрати, пов'язані з транспортуванням товарів, включно з прямими і непрямі витратами.

Надійність транспортування: ступінь відповідності фактичних результатів транспортування запланованим показникам, включно з частотою затримок, пошкоджень або втрат вантажу.

Гнучкість логістичної системи: здатність адаптуватися до змін попиту, ринкових умов або інших зовнішніх чинників без істотного зниження ефективності.

Використання транспортних ресурсів: ефективність використання транспортних засобів, включно з коефіцієнтом завантаження та оборотністю.

Загалом транспортна логістика спрямована на забезпечення ефективного, економічного та надійного переміщення товарів у рамках ланцюга поставок, що має вирішальне значення для досягнення стратегічних цілей підприємства.

1.3 Теоретичні основи оптимізації маршрутів

Оптимізація маршруту – це процес визначення найбільш ефективного шляху або послідовності дій для досягнення конкретної мети з урахуванням заданих критеріїв. У контексті транспортної логістики це означає планування маршрутів для перевезення вантажів таким чином, щоб мінімізувати витрати, терміни доставки та інші ресурси, забезпечуючи при цьому високу якість обслуговування клієнтів.

Основні критерії оптимізації маршруту включають:

1) мінімізацію часу доставки: скорочення часу, необхідного для транспортування вантажу від пункту відправлення до пункту призначення;
2) мінімізацію витрат: скорочення загальних витрат на транспортування, включаючи паливо, заробітну плату водіїв, амортизацію транспортних засобів тощо;

3) мінімізацію відстані: вибір найкоротшого маршруту між точками, що може бути корисно для скорочення витрат на паливо і зносу транспортних засобів;
4) комбінацію цих факторів: у багатьох випадках оптимізація вимагає балансування різних критеріїв, що може бути досягнуто за допомогою методів багатокритеріальної оптимізації.

Визначимо класичні завдання оптимізації маршрутів.

1. Завдання комівояжера: полягає у визначенні найкоротшого маршруту, який проходить через кожне місто рівно один раз і повертається до вихідної точки. Це класичне завдання комбінаторної оптимізації, яке має велике значення для планування маршрутів у транспортній логістиці.

2. Завдання про найкоротший шлях: полягає в пошуку найкоротшого шляху між двома точками в графі, де ребра мають певну вагу, що відповідає відстані або транспортним витратам. Для вирішення цього завдання широко використовуються алгоритми, такі як алгоритм Дейкстри.

3. Транспортна задача: полягає у визначенні оптимального способу доставки товарів від декількох постачальників до декількох споживачів з мінімальними витратами. Це класична задача лінійного програмування, яка використовується для планування логістичного ланцюжка.

1.4 Графова модель в логістиці

Граф – це сукупність об'єктів із зв'язками між ними. Об'єкти розглядаються як вершини, або вузли графа, а зв'язки – як дуги або ребра. Для різних галузей види графів можуть відрізнятися орієнтованістю, обмеженнями на кількість зв'язків і додатковими даними про вершини або ребра (рис. 1.1).

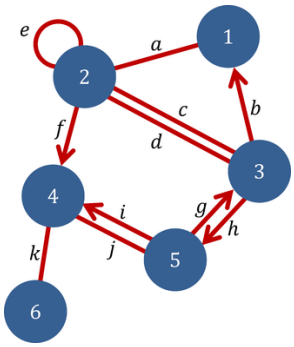


Рис. 1.1. Графічне представлення графа

У теорії графів розрізняють:

1. Спрямовані та неспрямовані графи. У спрямованих графах ребра мають напрямок, а в неспрямованих графах напрямку немає.

2. Зважені та незважені графи: у зважених графах кожному ребру присвоюється значення (вага), яке може представляти, наприклад, відстань або вартість, тоді як у незважених графах вага відсутня.

3. Зв'язні графи: граф є зв'язним, якщо між будь-якими двома його вершинами існує шлях.

У логістиці графи використовуються для моделювання транспортних мереж, де вершини представляють точки, такі як склади, магазини, пункти відправлення або пункти призначення, та ребра, які представляють маршрути між цими точками, які можуть мати різну довжину, час доставки або вартість.

Таке представлення дозволяє ефективно планувати та оптимізувати логістичні процеси з урахуванням різних факторів, таких як час, вартість і надійність маршруту.

Використання графів у логістиці має ряд переваг.

Щодо візуалізації, то графічне представлення транспортної мережі спрощує розуміння структури та взаємозв'язків між точками.

Щодо аналізу та оптимізації, то потрібно відмітити можливість застосування алгоритмів пошуку найкоротшого шляху, таких як алгоритм Дейкстри або алгоритм A*, для визначення оптимальних маршрутів.

Гнучкість – в простоті модифікації та адаптації моделі до змін у транспортній мережі або умовах експлуатації.

Також важливою перевагою є інтеграція з іншими системами: можливість об'єднання з географічними інформаційними системами (ГІС) для поліпшення планування та моніторингу логістичних операцій.

1.5 Алгоритми пошуку оптимальних маршрутів

Алгоритм Дейкстри призначений для знаходження найкоротших шляхів від початкової вершини до всіх інших вершин у графі з невід'ємними вагами ребер. Він використовує жадібний підхід, обираючи на кожному кроці вершину з найменшою відомою відстанню від початкової вершини і оновлюючи відстані до сусідніх вершин (рис. 1.2)

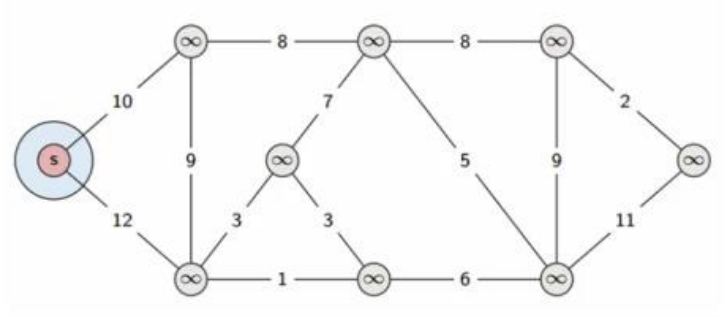


Рис.1.2. Графічне представлення алгоритму Дейкстри

Суть алгоритму Дейкстри полягає в наступному – рухатися завжди через найкоротше ребро, з найменшою вагою. Опишемо сам алгоритм Дейкстри покроково.

1) Помічаємо d(v1) = 0; d(vi) = ∞; y = v1, де d – відстань, v1 – початкова вершина.

2) Для сусідніх вершин на кожному кроці знаходимо найкоротший шлях.

3) Для наступного кроку обираємо вершину з ребром, що має найменшу вагу.

Якщо потрібно, то перераховуємо відстані. Перерахунок відстані для вершин vi , суміжних з y, виконується наступним чином d(vi) = min {d (vi), d(y) + l(y, vi)}. Тут l – вага ребра, що інцидентне обом вказаним вершинам і повторюємо, починаючи з п.2.

Перевага алгоритму Дейкстри в тому, що він простий у реалізації; гарантує знаходження оптимального рішення. Але є певні недоліки в застосуванні даного алгоритму. Через високу обчислювальну складність алгоритм Дейкстри є неефективним для великих графів з великою кількістю вершин. Також варто відмітити, що алгоритм Дейкстри застосовують для розв’язання задачі найкоротшого шляху в транспортних мережах з відомою кількістю точок [3, с. 186].

Обмеження:

– не працює з графами, що містять ребра з від'ємними вагами;
– може бути менш ефективним на великих графах без оптимізацій.

Класичним прикладом застосування алгоритму Дейкстри є пошук найкоротшого шляху в GPS-навігаційних системах.

Алгоритм A* є евристичним методом [3, с. 191] пошуку найкоротшого шляху, який використовує функцію оцінки для пришвидшення пошуку. Він комбінує переваги алгоритму Дейкстри та жадібного пошуку, обираючи шляхи, які обіцяють бути найбільш ефективними (рис. 1.3).

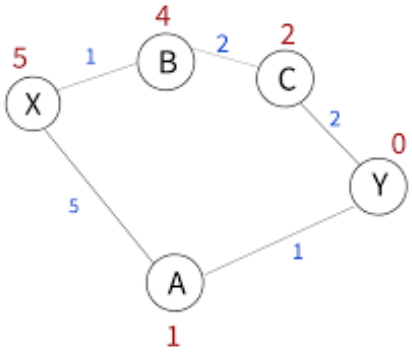


Рис.1.3 Графічне представлення алгоритму A*

Алгоритм ділить вершини на три класи.

1. Невідомі вершини: ці вершини ще не були знайдені. Ще не відомий шлях до них. На початку роботи алгоритму всі вершини, окрім початкової, належать до класу невідомих.

2. Відомі вершини (OpenList): вже відомий (можливо не оптимальний) шлях до цих вершин. Всі відомі вершини разом зі значеннями f зберігаються в списку. З цього списку вибираються, в першу чергу, найперспективніші вершини. Конкретна реалізація цього списку має суттєвий вплив на швидкодію алгоритму, і зазвичай має вигляд черги з пріоритетом (наприклад, бінарна купа). На початку роботи алгоритму до відомих вершин належить лише початкова вершина.

3. Повністю досліджені вершини (ClosedList). До цих вершин вже відомий найкоротший шлях. Повністю досліджені вершини додаються до так званого замкненого списку, аби запобігти багаторазовому дослідженню вже досліджених вершин. Список повністю досліджених вершин на початку роботи алгоритму порожній.

Кожна відома або повністю досліджена вершина має вказівник на попередні вершини. Завдяки цьому вказівникові, можна пройти шляхом від цієї до початкової вершини.

Коли вершину x буде повністю досліджено (або розкрито, релаксовано), суміжні з нею вершини додаються до списку відомих вершин, а сама вершина додається в список повністю досліджених. Вказівники на попередню вершину встановлюються на x. Суміжні вершини, які вже є в списку повністю досліджених вершин, до списку відомих не додаються, а зворотні вказівники не змінюються. Суміжні вершини, які вже є в списку відомих, лише оновлюються (значення f та вказівник на попередню вершину), якщо знайдений до них шлях коротший за вже відомий.

Алгоритм зупиняється, коли кінцева вершина потрапляє до списку повністю досліджених вершин. Знайдений шлях відтворюється за допомогою вказівників на попередню вершину. Якщо список відомих вершин порожній, то розв'язку задачі не існує й алгоритм припиняє пошук.

Відтворений за зворотніми вказівниками знайдений шлях починається з кінцевої вершини та прямує до початкової. Аби одразу отримати шлях в

правильному напрямі, з початкової вершини до кінцевої, в умовах задачі треба переставити місцями початок та кінець. Якщо шукати шлях починаючи з кінцевої вершини, відтворений список починатиметься з початкової вершини й прямуватиме до кінцевої.

Алгоритм пошуку A* знаходить оптимальний шлях між двома вершинами в графі. В залежності від функції вартості, яка задає кожному ребру його «вагу», оптимальність може означати найкоротший, найшвидший або навіть найпростіший шлях. Теоретично, алгоритм може розв'язувати всі задачі, які можна представити у вигляді задачі пошуку оптимального шляху на графі. Алгоритм A* використовується як для планування шляхів, так і в комп'ютерних іграх. Для планування шляхів, як евристична функція використовується лінійна відстань до цілі, оскільки згідно з нерівністю трикутника вона дає оптимальні оцінки. Також алгоритм A* використовується в іграх, в яких необхідно досягти наперед заданий стан, наприклад, в задачі про вісім ферзів, або в «п'ятнашках». Евристикою може слугувати, наприклад, кількість невірно пересунутих камінців. Порівняльну характеристику алгоритмів подано в таблиці 1.1.

Таблиця 1.1

Порівняння алгоритмів

Характеристика	Алгоритм Дейкстри	Алгоритм A*
Тип алгоритму	Жадібний	Евристичний
Вимоги до ваг	Невід'ємні	Евристичний
Гарантія оптимальності	Так	Так (за умови допустимої евристики)
Призначення	Загальний пошук	Спеціалізований пошук
Швидкість	Повільніший	Швидший

Вибір алгоритму залежить від конкретної задачі. Алгоритм Дейкстри є універсальним і підходить для загальних випадків, тоді як алгоритм A* є більш ефективним при наявності хорошої евристики та специфічних вимог до пошуку.

РОЗДІЛ 2. АНАЛІЗ ТА ПОСТАНОВКА ЗАДАЧІ ОПТИМІЗАЦІЇ ЛОГІСТИЧНИХ МАРШРУТІВ

2.1 Аналіз предметної області

У сучасних умовах стрімкого розвитку електронної комерції та сервісів доставки питання ефективної організації логістичних маршрутів набуває особливої актуальності. Однією з важливих складових будь-якої логістичної системи є побудова оптимального маршруту доставки товару або послуги до кінцевого споживача. Неефективна побудова маршрутів призводить до збільшення витрат, зниження швидкості обслуговування клієнтів, зростання навантаження на транспортну інфраструктуру та зниження прибутковості підприємств.

У даній роботі предметною областю є доставка піци в межах міста Кам'янець-Подільський. На прикладі умовної системи доставки, що включає кілька піцерій з різними цінами, координатами та умовами доставки, моделюється процес вибору найкращого маршруту доставки замовлення до клієнта.

Головна мета – забезпечити вибір оптимального варіанту доставки серед кількох закладів, враховуючи такі критерії:

- ціну піци (у грн);
- час доставки (у хвилинах).

У цьому випадку оптимальним вважається варіант, що забезпечує найнижчу вартість, найменший час або найкращий компроміс між цими показниками. Для вирішення задачі використано графовий підхід, при якому транспортна мережа міста подається у вигляді графа, а доставка з піцерій до клієнта – у вигляді маршруту в цьому графі. Граф Кам’янця-Подільського представлено на рис. 2.1.

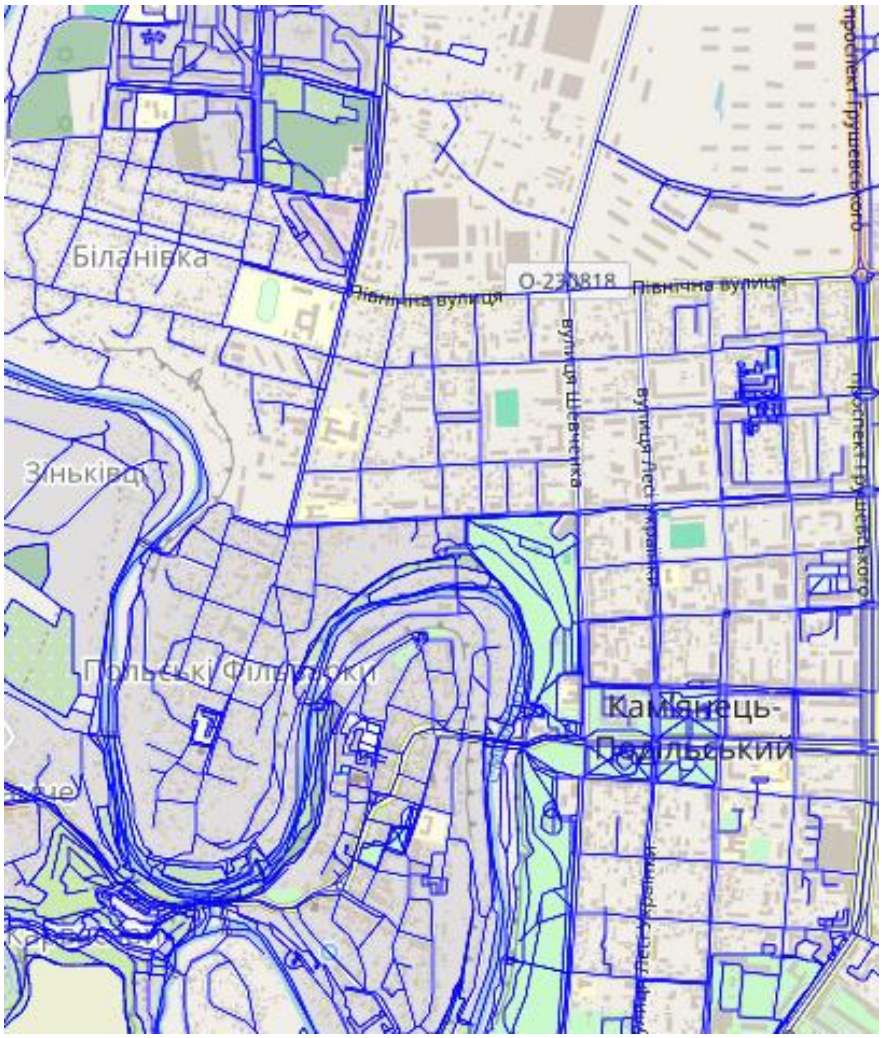


Рис. 2.1. Граф Кам’янця-Подільського

Ключовими елементами логістичної системи виступають:

- вершини – піцерії та точка доставки;
- ребра – вулична мережа між піцеріями та клієнтом;
- ваги – час проходження маршруту або вартість (як атрибути ребер).

Задача моделюється як багатокритеріальна оптимізація, що враховує різні аспекти доставки, зокрема мінімізацію часу, вартості та їх комбінацій.

Таким чином, у рамках цієї предметної області постає практична задача вибору оптимального маршруту доставки замовлення в межах міста з кількох потенційних точок відправлення. Розв’язання цієї задачі є основою подальшого формування математичної моделі та реалізації програмного модуля, що автоматично обирає найкращий варіант доставки.

2.2 Формалізація задачі оптимізації

Для ефективного вирішення задачі оптимізації логістичних маршрутів із використанням графів необхідно провести формалізацію – представити логістичну систему у вигляді математичної моделі, придатної для обчислень.

Опишемо математичну модель.

У рамках даної роботи транспортна мережа міста Кам’янець-Подільський подається у вигляді зваженого графа:

$$G = (V,E) \tag{2.1}$$

де:

V – множина вершин, що відповідають географічним точкам (вузли дорожньої мережі, піцерії, адреса клієнта);

E – множина орієнтованих ребер (дороги між точками), кожне з яких має вагу, що відповідає часу проходження ділянки, враховуючи довжину дороги, тип покриття, затори, світлофори тощо.

Постановка задачі: користувач вводить адресу доставки, а система виконує такі кроки:

1. Побудова графа дорожньої мережі міста на основі OpenStreetMap (через бібліотеку OSMnx).

2. Обчислення найкоротших маршрутів за часом від кожної піцерії до введеної адреси (за допомогою алгоритму Дейкстри).

3. Розрахунок загального часу доставки з урахуванням:

- часу руху;
- середньої затримки на світлофорах;

4. Формування трьох результатів:

- найдешевший варіант – мінімальна ціна;
- найшвидший варіант – мінімальний час;
- Оптимальний варіант – найменше значення функції оцінки (score).

Для визначення «оптимального» варіанту використовується зважена нормалізована функція:

$$p['score'] = 0.5 * price\_norm + 0.5 * time\_norm \quad (2.2)$$

де: 0.5 – вагові коефіцієнти.

Нормалізація дозволяє звести показники до однакового масштабу (від 0 до 1) для об’єктивного порівняння.

Цей підхід дозволяє гнучко змінювати пріоритети залежно від побажань користувача (наприклад, зробити акцент на часі або ціні).

Вхідні дані:

1. Географічні координати піцерій.

2. Ціна страви для кожної піцерії.

3. Адреса доставки (вводиться вручну).

5. Дорожня мережа міста (отримана через OSMnx).

Вихідні дані:

1. Побудовані маршрути доставки.

2. Таблиця з цінами, часом доставки та значенням функції оцінки.

3. Інтерактивна карта, яка візуалізує:

- маршрути;
- маркери з назвами піцерій;
- легенду категорій.

Таким чином, задача оптимізації логістичних маршрутів формалізується як задача пошуку найкоротших шляхів у зваженому графі. Це дозволяє застосовувати ефективні алгоритми пошуку, забезпечуючи точне й швидке знаходження оптимального варіанту доставки з урахуванням як часу, так і ціни.

2.3 Вибір методу та обґрунтування

Для розв’язання задачі оптимізації логістичних маршрутів на основі графової моделі було обрано підхід, що передбачає використання алгоритмів пошуку найкоротшого шляху в зваженому графі. Серед різних відомих

методів найбільш придатними для поставленої задачі є алгоритми Дейкстри та алгоритм A*.

У даному проекті алгоритм Дейкстри ефективно використовується для обчислення найшвидшого маршруту між піцерією та адресою клієнта, де вага ребер відповідає часу проходження ділянки дороги.

У контексті проєкту алгоритм Дейкстри застосовується для:

- визначення найшвидшого маршруту з піцерії до клієнта;
- роботи з графом, у якому кожне ребро має вагу у вигляді часу (розрахованого за типом дороги, відстанню та додатковими затримками).

Перевагою алгоритма Дейкстри є простота реалізації (реалізований через бібліотеку NetworkX) та гарантія знаходження оптимального рішення. Ідеально підходить для міських транспортних мереж із додатними вагами.

A* має потенційну перевагу при роботі з великими або розгалуженими графами, оскільки може зменшити кількість переглянутих вузлів.

Обґрунтуємо вибір.

У рамках даного проєкту реалізовано алгоритм Дейкстри як основний інструмент для побудови маршрутів. Водночас алгоритм A* також був розглянутий на теоретичному рівні як альтернатива для масштабування.

Алгоритм Дейкстри обрано з огляду на достатню точність, повну підтримку у бібліотеці NetworkX, адекватну продуктивність для задач у межах одного міста.

Алгоритм A* може бути інтегрований у майбутньому для підвищення продуктивності при переході на регіональний рівень (міжміські перевезення), роботі з великими графами або складними топологіями.

Вибір алгоритму Дейкстри забезпечив надійність і точність у побудові маршрутів, а розгляд A* відкриває можливість майбутнього вдосконалення системи. Обидва методи демонструють високу ефективність при правильному формулюванні задачі в межах графової моделі.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНІ РЕЗУЛЬТАТИ

3.1 Реалізація моделі

Для практичного вирішення задачі оптимізації логістичних маршрутів у межах міста Кам'янець-Подільський було розроблено повноцінну програмну модель мовою програмування Python. Модель реалізована як консольна програма з автоматичною побудовою маршрутів і візуалізацією на інтерактивній карті.

Використані інструменти та бібліотеки:

1. OSMnx – завантаження та обробка графа дорожньої мережі з OpenStreetMap;

2. NetworkX – робота з графами, реалізація алгоритму Дейкстри;

3. Geopy – геокодування введеної адреси у координати (широта/довгота);

4. Folium – створення HTML-карти з маршрутом;

5. webbrowser – відкриття карти у браузері;

6. time, os – для кешування та оновлення графа через певний час.

Імпорт бібліотек подано на рис. 3.1.

```
import osmnx as ox
import networkx as nx
import folium
from geopy.geocoders import Nominatim
import webbrowser
import time
import os
```

Рис. 3.1. Імпорт бібліотек

Опишемо архітектуру моделі.

Побудова графа дорожньої мережі здійснюється за допомогою OSMnx, з подальшим збереженням у форматі .graphml, вершини це точки на перехрестях або об'єктах, ребра – дороги з вагою у вигляді приблизного часу проходження.

Користувач вводить адресу доставки вручну через консоль (рис. 3.2).

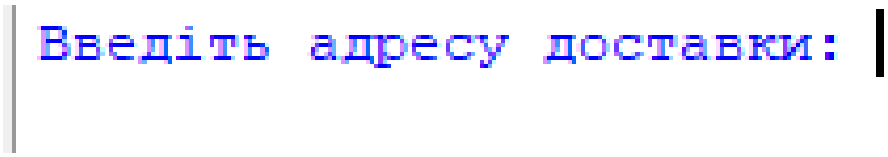


Рис. 3.2. Консоль скрипта

У програмі зберігається список піцерій (рис. 3.3) з фіксованими координатами та цінами.

```
pizza_places = [
    {
        "name": "Pronto Pizza & Sushi",
        "address": "вулиця Соборна, 29, Кам'янець-Подільський",
        "price": 180,
        "coords": (48.67929,26.58970)
    },
    {
        "name": "Dominic's pizza",
        "address": "вулиця Привокзальна, 31б, Кам'янець-Подільський",
        "price": 150,
        "coords": (48.67808,26.59807)
    },
    {
        "name": "Drova: Family Pizza",
        "address": "вулиця Павла Скоропадського, 19, Кам'янець-Подільський",
        "price": 200,
        "coords": (48.68276,26.56949)
    },
    {
        "name": "PizzaBIX",
        "address": "вулиця Соборна, 18, Кам'янець-Подільський",
        "price": 160,
        "coords": (48.67876,26.59124)
    },
    {
```

Рис. 3.3. Список частини піцерій

Для коректності координати піцерій визначались вручну через Google Maps. Якщо введено некоректну або неіснуючу адресу, програма дозволяє ввести її повторно без завершення роботи.

Пошук маршрутів здійснюється таким чином, що для кожної піцерії виконується пошук найкоротшого шляху до точки доставки за алгоритмом Дейкстри. Обчислення часу включає: час на основі швидкості та додаткові затримки (затори/умовні світлофори) за типом дороги.

Для оцінювання варіантів доставки кожен маршрут аналізується за трьома критеріями:

– найдешевше – піцерія з найнижчою ціною;

– найшвидше – піцерія з найкоротшим розрахунковим часом;

– оптимальне – вибір на основі функції (рис. 3.4)

```
p['score'] = 0.5 * price_norm + 0.5 * time_norm
```

Рис. 3.4. Формула для обрахунку оптимального маршруту

Це дозволяє враховувати баланс між ціною та часом доставки.

Для візуалізації результатів:

1. Генерується інтерактивна HTML-мапа (pizza_map.html) (рис. 3.5);

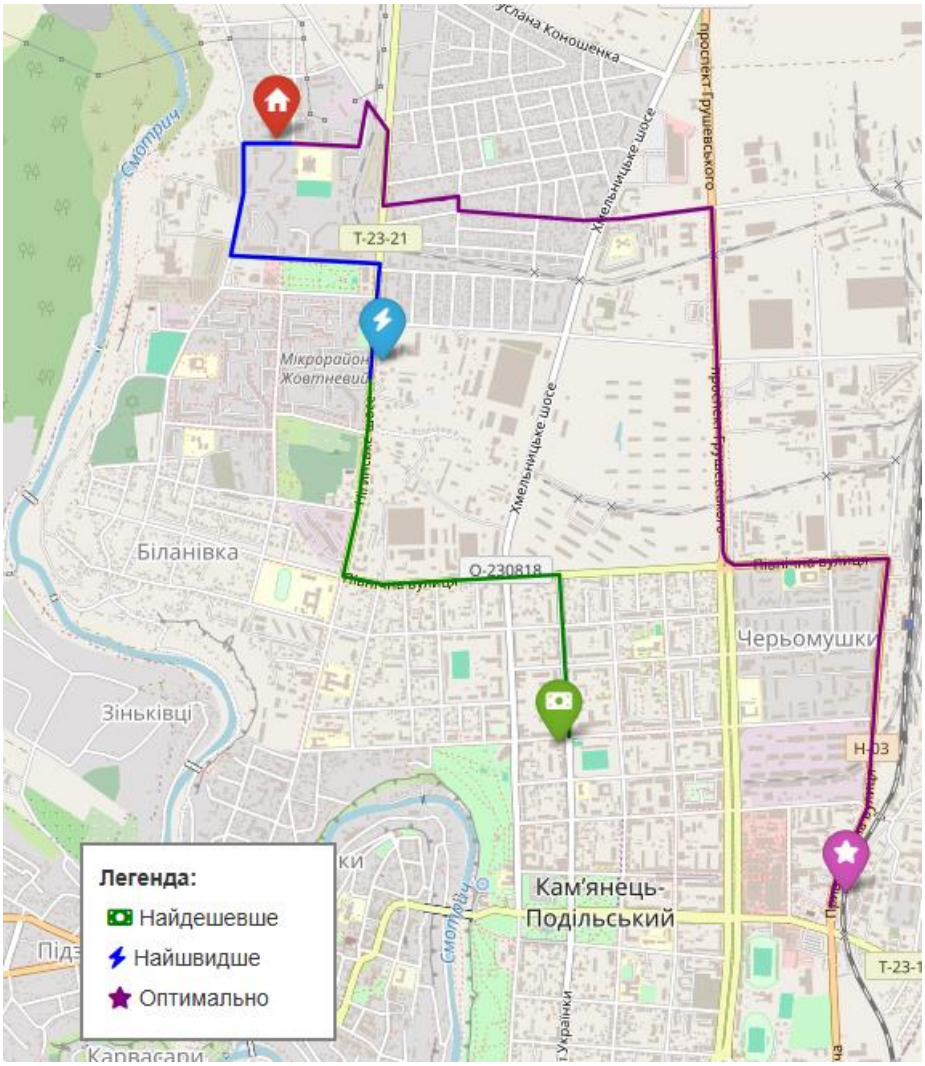


Рис. 3.5. Інтерактивна мапа

2. На карту додаються:

- точка доставки (червоний будинок);
- 3 піцерії (зелений, синій, фіолетовий маркер);
- полілінії маршрутів від піцерій до користувача;
- спливаючі підказки (час, ціна, назва піцерії) (рис. 3.6);
- легенда з поясненнями кольорів та значків.

Рис. 3.6. Спливаючі підказки

Серед удосконалень, реалізованих у фінальній версії Python-скрипта:

- повторне введення адреси, якщо геокодування неуспішне (рис. 3.7);
- оновлення графа раз на 30 днів – з метою збереження актуальності дорожньої мережі;
- фільтрація маршрутів, у яких немає координат (щоб уникнути помилок PolyLine);
- захист від падіння коду через try-except при пошуку маршруту;
- вилучення непотрібних імпортів, змінних та надлишкових обчислень.

```
Введіть адресу доставки: вулиця Огієка, 61
⚠ Адресу не знайдено. Спробуйте ще раз.

Введіть адресу доставки:
```

Рис. 3.7. Помилкове введення адреси

3.2 Експериментальні дослідження

Для перевірки працездатності реалізованої моделі було проведено серію експериментів у межах міста Кам’янець-Подільський. Метою досліджень було перевірити, наскільки стабільно, точно та логічно працює система побудови маршрутів доставки піци з кількох закладів до введеної користувачем адреси.

Мета досліджень:

1. Перевірити стабільність побудови маршрутів на реальній дорожній мережі.

2. Оцінити коректність обчислення часу доставки.

3. Перевірити, чи система об'єктивно вибирає найкращі варіанти доставки.

4. Виявити потенційні проблеми точності геоданих та логіки візуалізації.

Проведені етапи експерименту:

1. Геокодування введеної адреси користувачем через Geopy з використанням сервісу Nominatim.

2. Завантаження графа дорожньої мережі Кам’янця-Подільського з OpenStreetMap.

3. Пошук маршруту від кожної з 17 піцерій до точки доставки за алгоритмом Дейкстри;

4. Обчислення загального часу доставки, який включає:

- час руху маршрутом;
- затримки на світлофорах.

5. Оцінювання піцерій за ціною, часом та оптимальним (баланс часу й ціни).

6. Генерація інтерактивної HTML-мапи з візуалізацією маршрутів та маркерів для трьох найкращих варіантів.

Опишемо виявлені труднощі та їх усунення.

1. Геокодування піцерій.

Проблема: при автоматичному геокодуванні через Geopy координати поверталися для центру вулиці, а не фактичного розташування піцерії.

Рішення: для кожної піцерії вручну визначено точні координати через Google Maps і внесено їх безпосередньо до коду. Це суттєво підвищило точність побудови маршрутів.

2. Відсутність координат у графі.

Проблема: у деяких маршрутах окремі вершини графа не мали координат *x* і *y*, що призводило до відсутності лінії на карті.

Рішення: у коді реалізовано перевірку наявності координат перед додаванням точок до маршруту. Якщо координат немає – точка не додається, що забезпечує стабільність PolyLine.

Визначимо отримані результати експерименту.

Побудова маршрутів працює стабільно, навіть для адрес з віддаленими координатами.

Система точно визначає:

- піцерію з найменшою вартістю;
- піцерію з найшвидшим часом доставки;
- оптимальний варіант за нормалізованою функцією оцінки.

Карта будується автоматично, відкривається в браузері і містить:

- маркери піцерій;
- маршрути від піцерій до користувача;
- кольорову категоризацію (зелений, синій, фіолетовий);
- спливаючі підказки з ціною, часом, категорією;
- легенду з поясненнями.

Отже, результати досліджень підтверджують ефективність та надійність реалізованої моделі. Всі основні задачі – побудова маршруту, обчислення часу, оцінка та візуалізація виконуються коректно, що дозволяє використовувати систему як основу для реального логістичного рішення.

3.3 Аналіз результатів

У результаті проведених експериментів була підтверджена працездатність розробленої моделі та її ефективність з точки зору точності побудови маршрутів, логіки обчислень і зручності використання для кінцевого користувача. Алгоритм Дейкстри успішно знаходить найкоротший маршрут за критерієм часу відповідно до логіки дорожнього графа. Розрахунок часу доставки враховує не лише довжину маршруту, а й базову затримку на типі дороги та умовні затримки на світлофорах.

Візуалізація маршрутів працює стабільно навіть при обробці складних ділянок міської мережі. Система класифікує піцерії за трьома логічними критеріями:

1) найдешевше – обирається піцерія з мінімальною вартістю замовлення;
2) найшвидше – визначається найкоротший час доставки з урахуванням усіх факторів;
3) оптимальне – варіант із найнижчим комбінованим балом (score), що враховує і ціну, і час. Цей підхід дозволяє користувачу приймати рішення на основі його власних пріоритетів.

Генерується повноцінна HTML-мапа, яка відкривається автоматично в браузері. На карті відображаються:

- маркер адреси доставки (червоний);
- три піцерії з відповідним кольоровим позначенням;
- лінії маршрутів (PolyLine) від піцерій до клієнта;
- спливаючі підказки з деталями (назва, час, ціна, категорія);
- легенда, що пояснює кольори та значки.

Визначимо переваги реалізованої системи.

1. Гнучкість: легко додати нові піцерії або змінити критерії оцінки.

2. Масштабованість: система здатна адаптуватися до будь-якого міста без зміни логіки коду.

3. Інтуїтивність: користувач взаємодіє лише через одне поле вводу – адресу доставки.

4. Автоматизація: усі обчислення, вибір варіантів і вивід результатів виконуються без потреби ручного втручання.

Отже, результати показали, що реалізована система успішно вирішує поставлену задачу оптимізації логістичних маршрутів на основі графової моделі, демонструючи високу надійність, точність і практичну цінність.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи розроблено та реалізовано програмну систему для оптимізації логістичних маршрутів доставки піци в межах міста Кам’янець-Подільський із використанням графової моделі дорожньої мережі та алгоритмічних методів пошуку найкоротших шляхів. У результаті виконаного дослідження досягнуті усі поставлені цілі та виконані заплановані завдання.

На першому етапі проведено теоретичне дослідження предметної області, розглянуто поняття логістики та транспортної логістики, а також основи оптимізації маршрутів. Особливу увагу приділено графовому підходу до моделювання транспортних мереж, який дозволяє ефективно застосовувати математичні алгоритми для вирішення задач оптимізації. Розглянуто основні алгоритми пошуку найкоротшого шляху – алгоритм Дейкстри та A*, проведено їх теоретичне порівняння, описано переваги та недоліки кожного.

На основі теоретичних знань сформульовано математичну модель у вигляді зваженого графа, де вершини представляють точки на мапі (піцерії та адреса користувача), а ребра – відрізки дороги з певною вагою (час проходження). Це дозволило формалізувати логістичну задачу доставки як задачу пошуку найкоротших шляхів у графі з вагами. Залежно від критерію оптимізації (ціна, час або їх комбінація) запропоновано різні стратегії вибору найкращого маршруту.

У другій частині роботи проведено детальний аналіз поставленої задачі, а також визначено вимоги до побудови моделі. Обґрунтовано вибір методу – алгоритм Дейкстри як ефективного для задач маршрутизації з фіксованими невід’ємними вагами. Для комбінованої оцінки варіантів доставки розроблено функцію оцінки на основі нормалізованих значень ціни та часу, що дозволяє об’єктивно оцінювати баланс між швидкістю доставки та вартістю.

На третьому етапі реалізовано програмну модель мовою Python з використанням бібліотек OSMnx, NetworkX, Geopy, Folium, що забезпечили

завантаження дорожнього графа з OpenStreetMap, геокодування адрес, обчислення маршрутів і побудову інтерактивної карти. Користувач вводить адресу доставки, а система автоматично визначає три найкращі варіанти: найдешевший, найшвидший і оптимальний (за критерієм ціна/час), після чого результати виводяться на карту з відповідними кольорами, піктограмами та спливаючими підказками.

У ході експериментального дослідження система успішно обробила дані по 17 піцеріях у Кам’янці-Подільському. Виявлено технічні проблеми, пов’язані з неточностями у геокодуванні адрес, які були вирішені шляхом ручного збору координат через Google Maps. Також реалізована перевірка даних перед візуалізацією, що дозволило уникнути помилок при побудові маршрутів. Результати експериментів підтвердили, що система працює коректно, стабільно, будує реалістичні маршрути, обчислює час доставки та адекватно ранжує варіанти доставки.

Основними перевагами реалізованої моделі є наочна візуалізація маршрутів на карті, висока точність завдяки ручному введенню координат піцерій, автоматичний підбір оптимального варіанту доставки, можливість масштабування на інші міста або типи логістичних задач, простота у використанні для кінцевого користувача – достатньо лише ввести адресу.

Таким чином, у кваліфікаційній роботі успішно реалізовано задачу оптимізації логістичних маршрутів з використанням графів. Система може бути корисною як приклад побудови логістичного сервісу доставки на основі відкритих даних та алгоритмів маршрутизації. У майбутньому її можна доповнити новими функціями, такими як врахування «пробок» через API, введення годин пік, підрахунок витрат на паливо тощо.

Отже, розроблена модель підтвердила свою ефективність і практичну придатність для використання у реальних умовах. Робота повністю відповідає поставленій меті, має значну прикладну цінність і може бути використана в подальших дослідженнях та розробках у сфері логістики й транспортного моделювання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Жердієв Д. О. Оптимізація маршрутів руху транспортних засобів з використанням методу Ant-логістики у ТОВ «АВТО ТРЕЙД 2021» (м. Кропивницький) [Магістерська робота]. ЦНТУ, м. Кропивницький.
2. Загурський О. М., Загурська С. М. Моделювання маршрутів транспортно-логістичної системи постачань швидкопсувних харчових продуктів. International Science Journal of Engineering & Agriculture, 2022. 1(3). С. 216–228.
3. Креневич А.П. Алгоритми і структури даних. Підручник. Київ: ВПЦ «Київський Університет», 2021. 200 с.
4. Мінакова С., Грігорі О. Сучасні методи оптимізації логістичних процесів. Підприємництво та управління розвитком соціально-економічних систем, №2, 2023. С. 107-127.
5. Помаз А. С. Оптимізація транспортних маршрутів в логістиці за допомогою динамічного програмування. XXIV Міжнародна конференція здобувачів вищої освіти і молодих учених «Політ. Сучасні проблеми науки», К.: НАУ, 3–4 квітня 2024. С. 128–129.
6. Стефура Ю. Графові методи для мінімізації часу доставки у малих містах. Збірник матеріалів наукової конференції за підсумками науково-дослідної роботи здобувачів вищої освіти фізико-математичного факультету Кам’янець-Подільського національного університету імені Івана Огієнка у 2024-2025 н.р., 9-10 квітня 2025 року [Електронний ресурс]. Кам’янець-Подільський : Кам’янець-Подільський національний університет імені Івана Огієнка, фізико-математичний факультет, 2025. С. 10-13. URL: http://elar.kpnu.edu.ua/xmlui/handle/123456789/8094.
7. Стефура Ю. Методи оптимізації логістичних маршрутів за допомогою графів. Вісник Кам’янець-Подільського національного університету імені Івана Огієнка. Фізико-математичні науки. Випуск 17. Кам’янець-Подільський : Кам’янець-Подільський національний університет імені

Івана Огієнка, 2024. С. 158-163. URL: https://fizmat.kpnu.edu.ua/wp-content/uploads/2024/12/visnyk-17-2024-17-12-2024.pdf

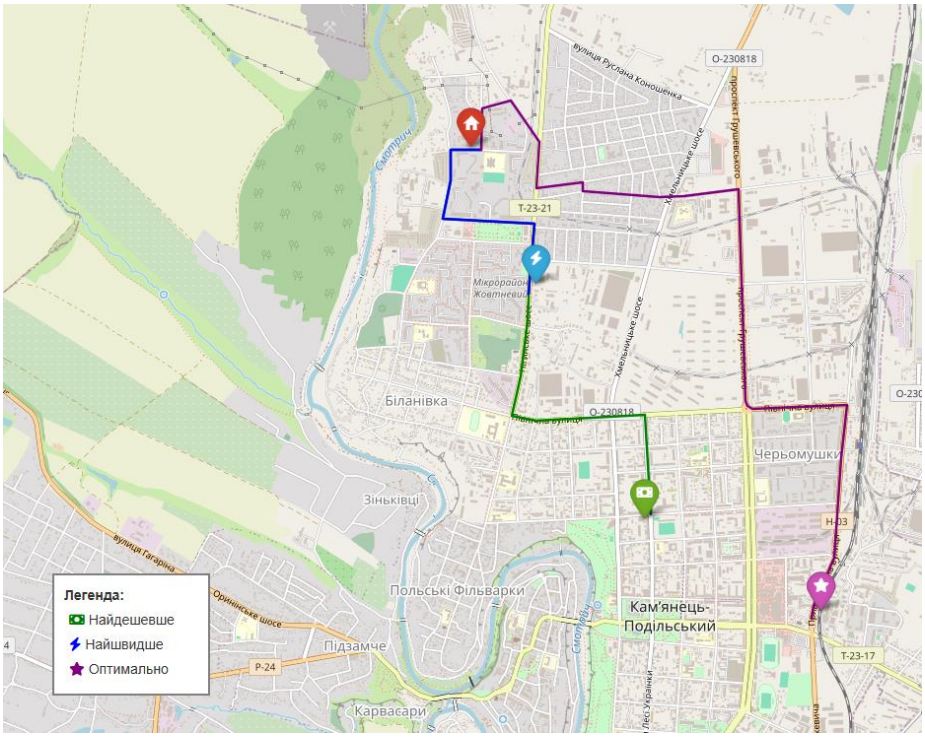
8. Bilski P., Markowski T. (2021). Optimization of autonomous agent routes in logistics warehouse. *International Journal of Electronics and Telecommunication*.
9. Boyles S. D., Lownes N. E., Unnikrishnan A**.** (2025). *Transportation Network Analysis, Volume I: Static and Dynamic Traffic Assignment*. ArXiv.
10. Lusiani A., Sartika E., Binarto A., Habinuddin E., Azis I. (2021). Determination of the Fastest Path on Logistics Distribution by Using Dijkstra Algorithm. *Advances in Engineering Research*, ISSAT.
11. Wang R., Lu Z., Jin Y., Liang C. (2022). Application of A* algorithm in intelligent vehicle path planning. *Mathematical Models in Engineering*, Vol. 8, 82–90.

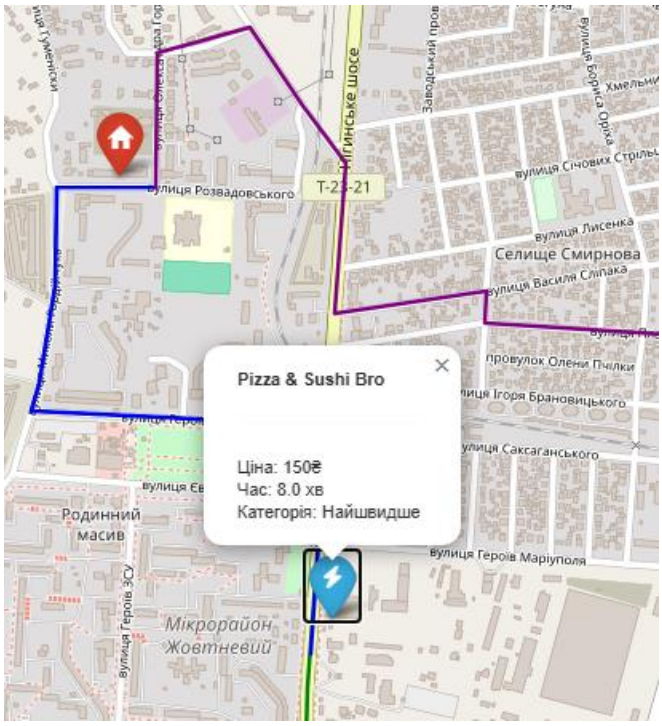
ДОДАТКИ

Додаток А

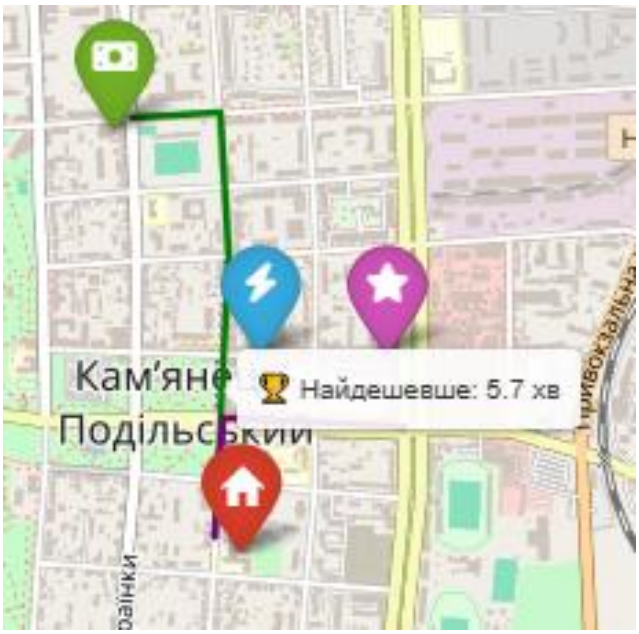
Результати роботи скрипта

```
Введіть адресу доставки: вулиця Розвадовского, 1
Створення нової дорожньої мережі...

========================================
🏆 Найдешевше: ШоТаке Pizza&Burger - 140₴ (16.2 хв)
🚀 Найшвидше: Pizza & Sushi Bro - 150₴ (8.0 хв)
☆ Оптимально: Dominic's pizza - 150₴ (20.3 хв)
========================================
```

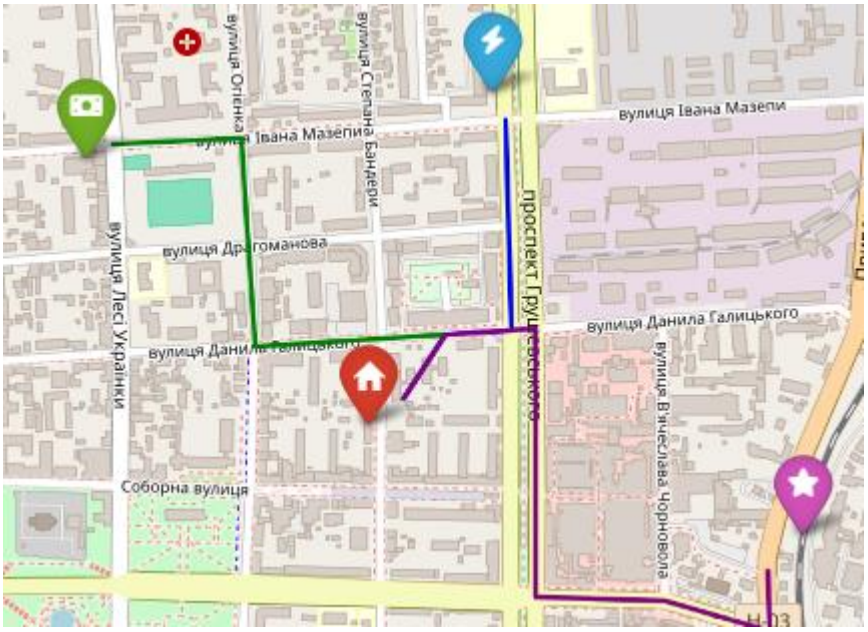




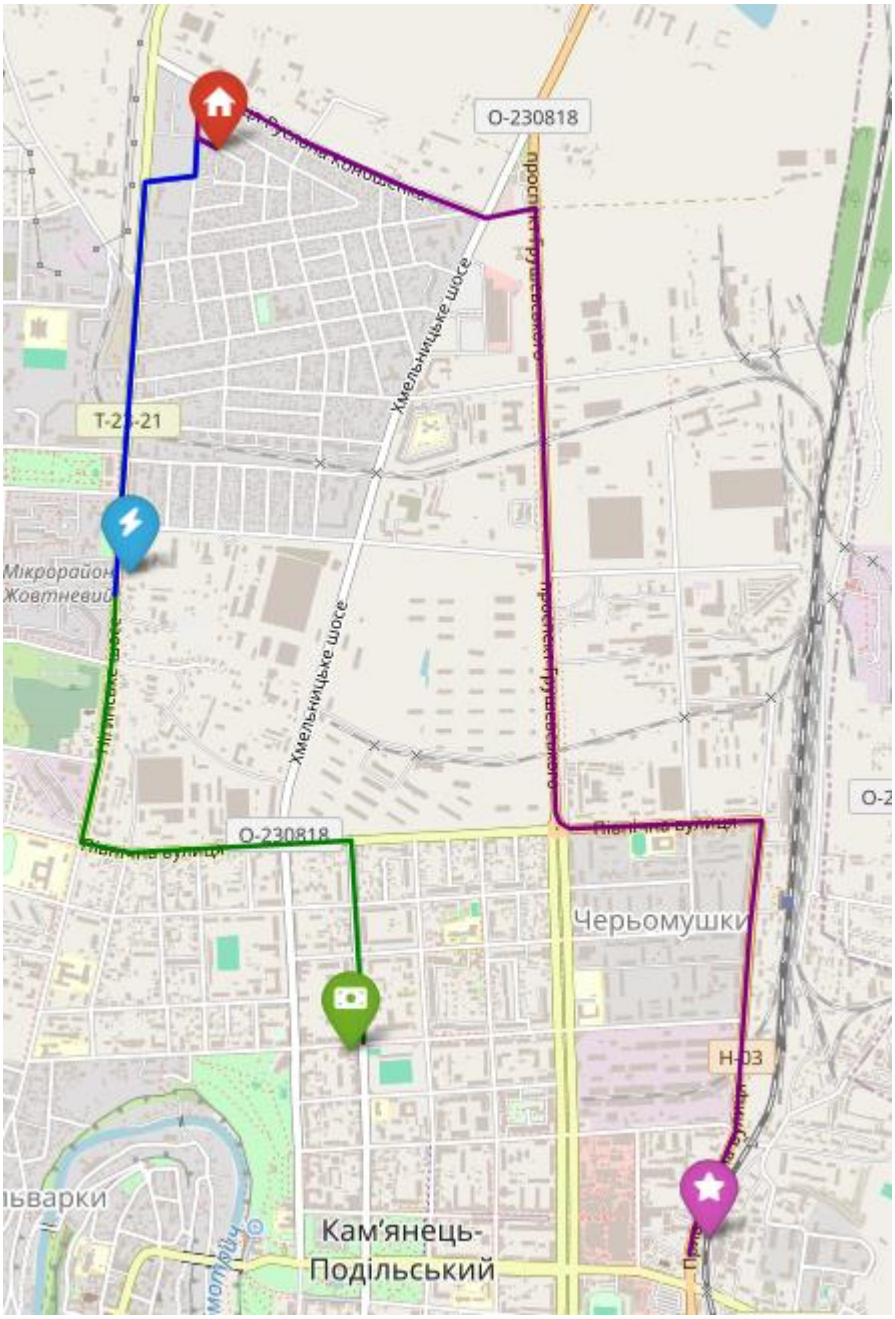

```
Введіть адресу доставки: вулиця Огієнка, 61

========================================
🏆 Найдешевше: ШоТаке Pizza&Burger - 140₴ (5.7 хв)
🚀 Найшвидше: Тераса - 200₴ (1.9 хв)
☆ Оптимально: PizzaBIX - 160₴ (3.1 хв)
========================================
```



```
Введіть адресу доставки: вулиця Степана Бандери, 46, Кам'янець-Подільський

========================================
🏆 Найдешевше: ШоТаке Pizza&Burger – 140₴ (6.1 хв)
🚀 Найшвидше: Скайпицца – 175₴ (3.2 хв)
☆ Оптимально: Dominic’s pizza – 150₴ (5.5 хв)
========================================
```



```
Введіть адресу доставки: вулиця Богуна, 29, Кам'янець-Подільський

========================================
🏆 Найдешевше: ШоТаке Pizza&Burger – 140₴ (16.4 хв)
🚀 Найшвидше: Pizza & Sushi Bro – 150₴ (8.2 хв)
☆ Оптимально: Dominic's pizza – 150₴ (16.5 хв)
========================================
```



Додаток Б

Код мовою Python

```
import osmnx as ox
import networkx as nx
import folium
from geopy.geocoders import Nominatim
import webbrowser
import time
import os
# Конфігурація
GRAPH_FILE = "kamianets.graphml"
SPEED_PROFILE = {
    'motorway': 60, 'trunk': 50, 'primary': 40,
    'secondary': 35, 'tertiary': 30, 'residential': 25,
    'unclassified': 20, 'service': 15
}
DELAYS = {
    'motorway': 0.0, 'trunk': 0.0, 'primary': 0.1,
    'secondary': 0.2, 'tertiary': 0.3, 'residential': 0.4
}
# Список усіх піцерій
pizza_places = [
    {
        "name": "Pronto Pizza & Sushi",
        "address": "вулиця Соборна, 29, Кам'янець-Подільський",
        "price": 180,
        "coords": (48.67929,26.58970)
    },
    {
```

```
    "name": "Dominic’s pizza",
    "address": "вулиця Привокзальна, 31б, Кам'янець-Подільський",
    "price": 150,
    "coords": (48.67808,26.59807)
},
{
    "name": "Drova: Family Pizza",
    "address": "вулиця Павла Скоропадського, 19, Кам'янець-Подільський",
    "price": 200,
    "coords": (48.68276,26.56949)
},
{
    "name": "PizzaBIX",
    "address": "вулиця Соборна, 18, Кам'янець-Подільський",
    "price": 160,
    "coords": (48.67876,26.59124)
},
{
    "name": "Піцерія Даніель",
    "address": "площа Польський ринок, 4Г, Кам'янець-Подільський",
    "price": 170,
    "coords": (48.67630,26.57398)
},
{
    "name": "ШоТаке Pizza&Burger",
    "address": "вулиця Івана Мазепи, Кам'янець-Подільський",
    "price": 140,
    "coords": (48.68382,26.58311)
},
```

```
{
   "name": "Скайпицца",
   "address": "проспект Грушевського, 32а, Кам'янець-Подільський",
   "price": 175,
   "coords": (48.68456,26.59160)
},
{
   "name": "Kvadrat Sushi&Pizza",
   "address": "вулиця Соборна, 5, Кам'янець-Подільський",
   "price": 190,
   "coords": (48.67930,26.58413)
},
{
   "name": "BAZZAR",
   "address": "вулиця Князів Коріатовичів, 14а/1, Кам'янець-Подільський",
   "price": 160,
   "coords": (48.67741,26.59319)
},
{
   "name": "Pizza & Sushi Bro",
   "address": "Нігинське шосе, 41/2, Кам'янець-Подільський",
   "price": 150,
   "coords": (48.69712,26.57350)
},
{
   "name": "Space Food",
   "address": "вулиця Данила Галицького, 20, Кам'янець-Подільський",
   "price": 185,
   "coords": (48.68062,26.58804)
```

```
},
{
   "name": "Green Pizza",
   "address": "вулиця Князів Коріатовичів, 25, Кам'янець-Подільський",
   "price": 165,
   "coords": (48.67789,26.60292)
},
{
   "name": "Fayna Pitsa",
   "address": "вулиця Першотравнева, 7б, Кам'янець-Подільський",
   "price": 160,
   "coords": (48.68006,26.59569)
},
{
   "name": "Sushizoom",
   "address": "вулиця Шевченка, 15, Кам'янець-Подільський",
   "price": 170,
   "coords": (48.68371,26.58129)
},
{
   "name": "Тераса",
   "address": "вулиця Соборна, 4, Кам'янець-Подільський",
   "price": 200,
   "coords": (48.67895,26.58750)
},
{
   "name": "Піцерія Сінема",
   "address": "проспект Грушевського, 15, Кам'янець-Подільський",
   "price": 165,
```

```
        "coords": (48.68659,26.59345)
    },
    {
        "name": "Паштетъ",
        "address": "проспект Грушевського, 31, Кам'янець-Подільський",
        "price": 155,
        "coords": (48.67339,26.59263)
    }
]

class GeoService:
    def __init__(self):
        self.geolocator = Nominatim(user_agent="pizza_delivery_v7")
        self.cache = {}

    def geocode(self, address):
        if address in self.cache:
            return self.cache[address]

        try:
            time.sleep(1)
            location = self.geolocator.geocode(address, timeout=15)
            if location:
                coords = (location.latitude, location.longitude)
                self.cache[address] = coords
                return coords
        except Exception as e:
            print(f"Помилка геокодування: {str(e)}")
        return None
```

```
def create_network():
    if os.path.exists(GRAPH_FILE):
        file_age = time.time() - os.path.getmtime(GRAPH_FILE)
        if file_age > 30 * 24 * 3600:  # 30 days
            print("Оновлення дорожньої мережі...")
            os.remove(GRAPH_FILE)

    try:
        G = ox.load_graphml(GRAPH_FILE)
        for u, v, data in G.edges(data=True):
            data['time'] = float(data.get('time', 0))
        return G
    except FileNotFoundError:
        print("Створення нової дорожньої мережі...")
        G = ox.graph_from_place("Kamianets-Podilskyi, Ukraine",
network_type='drive_service')

        for u, v, data in G.edges(data=True):
            road_type = data.get('highway', 'residential')
            if isinstance(road_type, list):
                road_type = road_type[0]

            speed = SPEED_PROFILE.get(road_type, 20)
            base_delay = DELAYS.get(road_type, 1.0)
            length = float(data.get('length', 0))

            time_min = (length / 1000) / (speed / 60) + base_delay
            data['time'] = round(time_min, 1)
```

```
        ox.save_graphml(G, GRAPH_FILE)
        return G

def get_route(graph, origin, destination):
    try:
        orig_node = ox.distance.nearest_nodes(graph, origin[1], origin[0])
        dest_node = ox.distance.nearest_nodes(graph, destination[1], destination[0])
        return nx.shortest_path(graph, orig_node, dest_node, weight='time')
    except Exception as e:
        print(f"Помилка маршруту: {str(e)}")
        return None

def calculate_route_time(graph, path):
    total = 0.0

    for u, v in zip(path[:-1], path[1:]):
        try:
            edge_data = graph[u][v]
            first_edge = next(iter(edge_data.values()))
            total += float(first_edge.get('time', 0))
        except (KeyError, StopIteration):
            return 9999.9



    return round(max(total, 0.1), 1)

def main():
    geo = GeoService()
```

```
delivery_coords = None
while not delivery_coords:
    delivery_address = input("Введіть адресу доставки: ")
    delivery_coords = geo.geocode(delivery_address)
    if not delivery_coords:
        print("⚠ Адресу не знайдено. Спробуйте ще раз.\n")



G = create_network()
results = []

for place in pizza_places:
    path = get_route(G, place['coords'], delivery_coords)
    if path and len(path) > 1:
        time_min = calculate_route_time(G, path)
        route_coords = []
        for n in path:
            node = G.nodes[n]
            if 'x' in node and 'y' in node:
                route_coords.append((node['y'], node['x']))
            else:
                print(f"⚠ Вершина {n} не має координат.")
        results.append({**place, 'time': time_min, 'route': route_coords})

if not results:
    print("Доставка неможлива: жоден ресторан не обслуговує цю адресу")
    return

try:
```

```
        max_price = max(p['price'] for p in results)
        min_price = min(p['price'] for p in results)
        max_time = max(p['time'] for p in results)
    except ValueError:
        print("Недостатньо даних для розрахунків")
        return

    for p in results:
        price_norm = (p['price'] - min_price) / (max_price - min_price) if (max_price -
min_price) != 0 else 0
        time_norm = p['time'] / max_time if max_time != 0 else 0
        p['score'] = 0.5 * price_norm + 0.5 * time_norm

    candidates = results.copy()
    categories = []

    # Найдешевша
    cheapest = min(candidates, key=lambda x: x['price'])
    categories.append(('🏆 Найдешевше', cheapest, 'green', 'money-bill'))
    candidates = [p for p in candidates if p['name'] != cheapest['name']]

    # Найшвидша
    if candidates:
        fastest = min(candidates, key=lambda x: x['time'])
        categories.append(('🚀 Найшвидше', fastest, 'blue', 'bolt'))
        candidates = [p for p in candidates if p['name'] != fastest['name']]

    # Оптимальна
```

```
    best_overall = min(results, key=lambda x: x['score']) if not candidates else min(candidates, key=lambda x: x['score'])
    categories.append(('⭐ Оптимально', best_overall, 'purple', 'star'))

    # Вивід результатів
    print("\n" + "="*40)
    for label, place, _, _ in categories:
        print(f"{label}: {place['name']} - {place['price']}₴ ({place['time']} хв)")
    print("="*40)

    # Візуалізація
    m = folium.Map(location=delivery_coords, zoom_start=14)
    folium.Marker(delivery_coords, icon=folium.Icon(color='red', icon='home')).add_to(m)

    for idx, (label, place, color, icon) in enumerate(categories):
        offset = 0.0002 * idx
        marker_coords = (place['coords'][0] + offset, place['coords'][1] + offset)

        folium.Marker(
            marker_coords,
            popup=folium.Popup(
                f"<b>{place['name']}</b><hr>"
                f"Ціна: {place['price']}₴<br>"
                f"Час: {place['time']} хв<br>"
                f"Категорія: {label.split()[1]}",
                max_width=250
            ),
            icon=folium.Icon(color=color, icon=icon, prefix='fa')
```

```
        ).add_to(m)

        folium.PolyLine(
            place['route'],
            color=color,
            weight=2.5,
            tooltip=f"{label}: {place['time']} хв"
        ).add_to(m)

    # Легенда
    legend_html = '''
    <div style="position: fixed; bottom: 50px; left: 50px;
                width: 160px; background: white; padding: 10px;
                border: 2px solid grey; z-index: 9999; font-size: 14px;">
        <b>Легенда:</b>
        <p style="margin:5px;"><i class="fa fa-money-bill" style="color:green"></i> Найдешевше</p>
        <p style="margin:5px;"><i class="fa fa-bolt" style="color:blue"></i> Найшвидше</p>
        <p style="margin:5px;"><i class="fa fa-star" style="color:purple"></i> Оптимально</p>
    </div>
    '''
    m.get_root().html.add_child(folium.Element(legend_html))
        m.save("pizza_map.html")
    webbrowser.open("pizza_map.html")
if __name__ == "__main__":
    main()
```