МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет радіоелектроніки
Факультет Комп'ютерних наук
Кафедра Програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

другий (магістерський)
(рівень вищої освіти)

Дослідження методів лінійного програмування для визначення короткого шляху за умови обмежених умов доступу до інтернет та GPS

Виконав:
студент 2 курсу групи ІПЗм-21-4
Валуйський В.Є.

Спеціальність: 121 – Інженерія програмного забезпечення

Тип програми: Освітньо-наукова

Керівник: доцент каф. ПІ, к.т.н., Ревенчук І.А.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри, проф. _____ З.В. Дудар
(підпис)

2023 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Програмна Інженерія

Рівень вищої освіти другий (магістерський)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва)

Тип програми Освітньо-наукова програма

Освітня програма Інженерія програмного забезпечення
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«____» _____ 20__ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Валуйському Владиславу Євгеновичу
(прізвище, ім'я, по батькові)

1. Тема роботи *«Дослідження методів лінійного програмування для визначення короткого шляху за умови обмежених умов доступу до інтернет та GPS»* затверджена наказом по університету від «___» _____ 20__ р. №
2. Термін подання студентом роботи до екзаменаційної комісії «___» ___ 20__ р.
3. Вихідні дані до роботи *В програмній системі передбачити дослідження методів позиціонування та навігації без доступу к системі Internet та GPS. Використовувати мову програмування.*
4. Зміст пояснювальної записки (перелік питань, що належить розробити) *вступ, аналіз предметної галузі, формування вимог до програмної системи, архітектура і проектування, висновки, перелік посилань, додатки.*

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапу роботи	Примітка
1	Дослідження методів позиціонування	12.04.2023	
2	Дослідження алгоритмів пошуку шляху	15.04.2023	
3	Формування вимог	18.04.2023	
4	Розробка та тестування алгоритмів	22.04.2023	
5	Оформлення пояснювальної записки	1.05.2023	
6	Підготовка доповіді та презентації	7.05.2023	
7	Перевірка роботи на плагіат	9.05.2023	
8	Проведення нормоконтролю роботи	9.05.2023	
9	Рецензування роботи	10.05.2023	
10	Занесення роботи в електронний архів	10.05.2023	
11	Попередній захист кваліфікаційної роботи	10.05.2023	
12	Допуск до захисту роботи зав. кафедри		
13	Захист кваліфікаційної роботи	18.05.2023	

Дата видачі завдання «12» _____квітня_____ 2023 р.

Студент _____
(підпис)

Керівник роботи _____ доцент каф. ПІ, к.т.н., Ревенчук І.А.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до кваліфікаційної роботи магістра, 77 стор., 10 рис., 1 табл., 11 джерел.

ЛІНІЙНЕ ПРОГРАМУВАННЯ, ТРАНСПОРТНА ЗАДАЧА, DART, НАВІГАЦІЯ, АЛГОРИТМИ ОПРАЦЮВАННЯ ГРАФІВ

Об’єкт дослідження – методи лінійного програмування у вирішенні транспортної задачі з обмеженнями пропускної спроможності, а також алгоритми апроксимації та обробки даних матричного та графового виду.

Мета дослідження – використання отриманих в ході дослідження результатів у проектуванні алгоритмів для навігації відвідувачів великих торгівельних центрів, корпоративних будівель, тощо.

Результатом роботи є виконаний аналіз доступних методів вирішення транспортної задач та проектування методів та алгоритмів для навігації людей в великих торгівельних центрах.

LINEAR PROGRAMMING, TRANSPORT TASK, DART, NAVIGATION, GRAPHS PROCESSING ALGORITHMS

Object of research are methods of the linear programming in solving the transport problem with bandwidth limitations and also algorithms of matrix and graph data type approximation and processing.

The purpose of the study is to use the results of the algorithms research obtained during the study to navigate visitors inside large shopping centers, big corporate buildings, private use etc.

The result of research work is done an analysis of available methods of solving transport task and designing methods and algoriths for navigating people in large shopping centers.

Я, Валуйський Владислав Євгенович
(прізвище, ім'я, по батькові)
студент групи __ІПЗм-21-4__ здобувач вищої освіти на другому (магістерському) рівні
кафедра програмної інженерії
(повна назва кафедри)
заявляю: моя кваліфікаційна робота на тему Дослідження методів лінійного програмування для визначення короткого шляху за умови обмежених умов доступу до інтернет та GPS,
(назва роботи)
що буде представлена до ЕК для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElArKhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ВСТУП

З кожним роком кількість автомобілів у світі зростає. Різні керуючі організації, як державні, так і інтернаціональні, намагаються регулювати кількість машин, їхній вплив на оточуюче середовище тощо. Але незмінним є той факт, що з кожним роком кількість водіїв та їх транспортних засобів зростає, як і необхідність місць, де можна залишити автомобіль на деякий час, тобто місць для паркування.

Слід також зазначити, що у деяких країнах є тенденція переходу на альтернативні транспортні засоби, такі як самокати та велосипеди. Перш за все, реальними чинниками цього переходу є історичний контекст (початкова забудова здійснювалась за довго до появи феномену автомобілів, міста не були розраховані на таку щільність населення та кількість персонального транспорту) та екологічність (перехід на транспортні засоби, які потребують меншого об'єму палива або працюють на електричній енергії).

Так, на сьогодні, у багатьох країнах світу існують проблеми з місцями для паркування. Спектр проблем на стільки різноманітний, що у різних країнах проблеми можуть бути діаметрально протилежними. Наприклад, проблема відсутності місця для майданчиків для паркування та проблема надмірно великого розміру майданчика для паркування.

Але з точки зору кінцевого користувача, навіть якщо існує достатня кількість місць для паркування, у великих паркінгах залишається проблема пошуку автомобіля. Нескінченна кількість секторів паркування, поверхів паркінгу, входів на паркінг не залишають водієві можливості швидкого пошуку свого автомобіля.

Найчастіше проблема пошуку автомобіля стає явною у найбільших торгово-розважальних центрах (англ. – shopping mall, shopping center). Площа найбільших з них сягає десятків гектарів. Виходячи з цього факту, для того щоб розмістити всіх бажаючих на території такого центру, необхідно мати великий, багаторівневий

майданчик для паркування. Вже на цьому етапі кінцевий користувач, тобто водій, що завітав до торгово-розважального центру, має прикладати значні зусилля для того, щоб припаркувати, а згодом – і відшукати, свій автомобіль. Підсилює ефект загубленості те, що ці майданчики для паркування зазвичай є багаторівневими та знаходяться у підвалах, де немає сигналу GPS.

Метою представленої роботи є система, що дозволить водіям швидко і зручно знаходити місце для паркування, не створюючи заторів, а після відвідування торгово-розважального комплексу – якнайшвидше знайти своє авто.

В рамках цієї роботи досліджується проблема найшвидшої навігації користувачів з торгового центру, до паркінгу, де вони залишили свій автомобіль. Ця проблема виражена у вигляді транспортної задачі лінійного програмування з обмеженим доступом до мережі Інтернет та GPS навігації.

Робота є актуальною, оскільки кількість великих торгово-розважальних комплексів росте з кожним роком, а для існуючих програмних додатків для розв’язання рішення цієї проблеми на даний момент не існує.

Приватні дані користувачів мають буди збережені в таємниці для забезпечення безпеки, проте використовуватись у статистичних підрахунках для можливості отримання загальних даних про відвідуваність.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Виявлення проблем та актуалізація рішень

Наразі немає єдиного програмного рішення для спрощення паркувального процесу у великого розміру паркувальних майданчиках. Існує доволі велика кількість програмних рішень, які спрощують той чи інший аспект паркування, але у рамках цієї роботи розглядається проблема, яка ще не має рішення, а саме – проблема паркування на великих паркувальних майданчиках. Зазвичай дійсно великі паркувальні майданчики знаходяться під землею, та не мають доступу до GPS[1], що є ще одним обмежувальним фактором.

Розглянемо можливі прями та непрямі аналоги даної програмної системи.

Під прямими аналогами вважаються вже існуючі засоби, що реалізовують функціонал для вирішення конкретної проблеми. На сьогоднішній день для даної програмної системи неможливо визначити прямих аналогів.

Непрямі аналоги – це існуючі засоби, що частково виконують функції іншого продукту, та можуть бути конкурентами за ресурс користувача. Першим непрямим аналогом можна виділити сервіс AParking[2]. Цей проект надає можливість користувачам знайти вільне паркувальне місце в місті Алмати, допомагає зробити пересування пішоходів більш безпечним, робить пересування міського громадського транспорту стабільнішим, а також створений задля зменшення можливий порушень ПДР.

Розглянемо, чому цей сервіс є саме непрямим аналогом. По-перше, дана система розповсюджується лише на конкретній території, що відрізняється від цільового місця розповсюдження даної програмної системи. По-друге, пошук паркувального місця робиться за наявності мережі Інтернет та системи GPS, що є повність протилежним до одного с обмежень даної програмної системи. По-третє, сфера діяльності сервісу AParking розповсюджується на наземні одноярусні паркування, що не зв’язані з цільовим місцем використання даної програмної системи. В даному сервісі аналогом можна виділити розв’язання проблеми

лінійного програмування для визначення найкоротшого шляху для користувачів, та аналізу пропускної спроможності шляхів та паркувань.

Другим непрямим аналогом можна виділити мобільний додаток Parking Helper. Цей додаток допомагає людям знаходити вільне паркувальне місце, прокладати маршрут до свого автомобіля, а також контактувати с іншими власниками авто, які блокують виїзд. Основним недоліком можна виявити обов’язковий зв'язок з мережею Інтернет та геолокацією[3], а також обмеженням реєстрацією користувачів для зв’язку між ними.

1.2 Окреслення плану дослідження

Отже, необхідно дослідити можливі способи знаходження найкоротшого шляху спроектувати систему для полегшення знаходження шляху до виходу, або паркувальний майданчик, де людина залишила свою автівку (що зазвичай знаходяться у великих торгово-розважальних комплексах, кількість яких зростає з кожним роком), без використання мережі Інтернет та системи GPS. Предметом цієї роботи є проектування функціоналу алгоритму знаходження найкоротшого шляху, базуючись на наборі багатьох змінних та факторів.

Першим етапом дослідження повинен виступати аналіз можливостей орієнтування без доступу до мережі Інтернет та системи GPS. На даний момент найширше застосування має технологія орієнтування у просторі за допомогою WiFi мережі.

Другим етапом дослідження повинно стати апроксимація плану будівель або конкретних поверхів у математичний формат задля подальшого аналізу отриманих даних та розробки алгоритму знаходження оптимального шляху

Третій етап дослідження буде включати у себе дослідження доступних алгоритмів обробки для просторових даних для створення алгоритму знаходження короткого шляху.

Після всіх досліджень буде обрана технологія та розроблена програма, що може бути використана для вирішення проблеми навігації без доступу до мережі Інтернет та системи GPS. Вибір технології має бути проаналізований на базі плюсів та мінусів конкретної технології.

2 ДОСЛІДЖЕННЯ МЕТОДІВ WI-FI ПОЗИЦІОНУВАННЯ

2.1 Навігація за допомогою Wi-Fi мережі

У наш час мережі Wi-Fi є настільки поширеними, за допомогою них можна легко створити мережу, яка буде складатися з великої кількості пристроїв, котрі не прив’язані до конкретного місця та можуть змінювати своє положення в просторі. На даний момент практично будь-який корпоративний клієнт має Wi-Fi покриття[4], тому інфраструктуру такої бездротової мережі можна використовувати для виконання ряду певних завдань. Одним з них може бути локальне позиціонування в організації. Точність подібних систем напряму залежить від щільності точок доступу, які прив’язані до конкретних точок плану будівлі, споруди або певної території

2.1.1 Дослідження технології Wi-Fi позиціонування

Рішення Wi-Fi для внутрішнього позиціонування використовують існуючі точки доступу до Wi-Fi або спеціальні датчики, що підтримують технологію Wi-Fi для виявлення та визначення місцезнаходження пристроїв Wi-Fi, що передають дані, як приклад можна назвати смартфони та мітки відстеження всередині приміщень.

Дані про місцезнаходження, що збираються датчиками або точками доступу чи надіслані з точок доступу на клієнтські пристрої попадаюсь у різні програми визначення місцезнаходження та перетворюються на статистичні дані, що в майбутньому можуть забезпечувати чисельні випадки використання даних про місцезнаходження користувачів.

Системи позиціонування на основі Wi-Fi можуть використовувати різні методи визначення розташування пристроїв. Більшість з них покладається на

методи, які засновані на індикаторі потужності отриманого сигналу (Received Signal Strength Indicator або коротко RSSI). Проте деякі програми можуть використовувати більш просунуті методи позиціонування Wi-Fi.

2.1.2 Позиціонування Wi-Fi за допомогою точок доступу

Позиціонування Wi-Fi за допомогою точок доступу напряму залежить від існуючої інфраструктури Wi-Fi, встановленої у внутрішніх приміщеннях задля визначення місцезнаходження потрібних пристроїв.

Ця технологія дозволяє великим організаціям використовувати існуючу інфраструктуру для роботи своїх додатків із ціллю визначення місцезнаходження без необхідності додаткового встановлення нового обладнання.

Точки доступу в будівлі можуть виявляти передачі даних з навколишніх пристроїв Wi-Fi, як у локальній мережі, так і поза нею. Потім ці дані про місцезнаходження користувача надсилаються на сервер і використовуються для розрахунку положення пристрою в просторі будівлі.

2.1.3 Позиціонування Wi-Fi за допомогою датчиків

Позиціонування Wi-Fi за допомогою датчиків насамперед використовує датчики з підтримкою Wi-Fi, які розгортаються у фіксованих положеннях у внутрішньому просторі будівлі.

Ці датчики пасивно можуть виявляти і визначати місцезнаходження передачі даних зі смартфонів, тегів відстеження активів, маячків, значків персоналу, переносних пристроїв та інших Wi-Fi пристроїв.

Після цього дані про місцезнаходження, що були зібрані датчиком, надсилаються на сервер і приймаються центральною внутрішньою системою позиціонування Indoor Positioning System (IPS) або системою визначення місцезнаходження в реальному часі Real-Time Location System (RTLS).

Механізм визначення місцезнаходження аналізує дані, для того щоб визначити місцезнаходження пристрою, який ці дані передає. Ці координати можна використовувати для візуалізації місцезнаходження пристрою чи об’єкта на карті приміщень конкретного простору або використовувати задля інших цілей, що залежать від конкретної програми, яка має розпізнавати місцезнаходження.

2.2 Методи Wi-Fi Позиціонування

Найпоширеніші методи Wi-Fi позиціонування визначаюсь місцезнаходження об’єкту за допомогою показника, що називається індикатором потужності отриманого сигналу Received Signal Strength Indicator (RSSI), головним чином для мультилатерації пристрою або обчислення відбитків. Ці підходи, які були засновані на потужності сигналу, є дуже простими у застосуванні, а також економічними не можуть забезпечити високу точність, оскільки на силу сигналу може впливати навколишнє середовище.

Рішення RSSI також сприйнятливі до помилок, що можуть бути викликані рухом об’єктів у навколишньому середовищі, наприклад таких як люди.

Додавання інших менш поширених, та більш передових методів може призвести до більш точних результатів Wi-Fi позиціонування. Вони включають до себе кут прибуття Angle of Arrival (AoA) та час польоту Time of Flight (ToF)

2.2.1 Мультилатерація RSSI

У додатках на основі RSSI кілька наявних точок доступу Wi-Fi або датчиків із підтримкою Wi-Fi, що розміщенні у фіксованому положенні, будуть виявляти пристрої Wi-Fi, які передають дані, а також потужність отриманого сигналу від пристрою.

Ці дані про місцезнаходження, які були зібрані точками доступу або датчиками будуть надсилатися до центральної внутрішньої системи позиціонування Indoor Positioning System (IPS) або до системи визначення місцезнаходження в реальному часі Real-Time Location System (RTLS).

На рисунку 2.1 наведено схему мультилатерації RSSI:

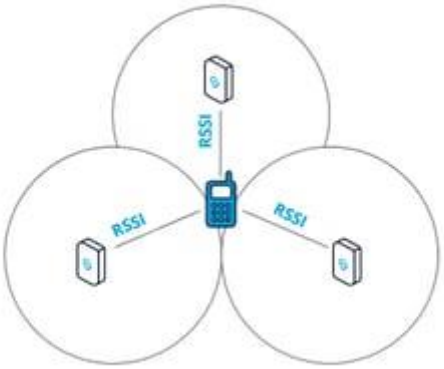

Рисунок 2.1 – Схема RSSI

Механізм визначення точного місцезнаходження аналізує дані та використовує алгоритми мультилатерації задля оцінки розташування пристроїв, які передають дані. Крім того, для визначення місцезнаходження пристрою можна використовувати потужність сигналу найближчих точок доступу щодо бездротового пристрою.

Використання методу на основі RSSI з мультилатерацією є найпростішим і найдешевшим варіантом Wi-Fi позиціонування. Проте він не може забезпечити

високого ступеню позиційної точності, оскільки піддається послабленню сигналу, поглинанню сигналу, відображенню сигналу та різним перешкодам.

2.2.2 Зняття відбитків RSSI

Зняття відбитків також є методом на основі RSSI. Wi-Fi позиціонування за допомогою відбитків передбачає насамперед використання бази даних, яка буде записувати місцезнаходження та потужність сигналу навколишніх точок доступу Wi-Fi, а також координати Wi-Fi пристрою, наприклад смартфону або мітки відстеження в неактивній фазі.

Щоб створити базу даних відбитків потрібен довгий та трудомісткий процес калібрування, який може знадобитися багаторазово виконувати. Під час активного відстеження пристрою значення RSSI порівнюється з цими відбитками у базі даних, для того щоб оцінити місцезнаходження попередньо навченого пристрою.

На рисунку 2.2 зображено схему зняття відбитків RSSI:

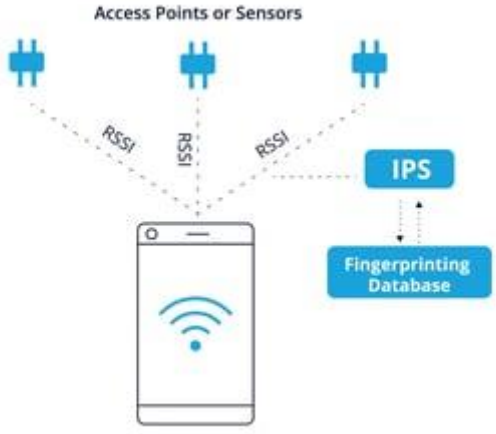

Рисунок 2.2 – Схема зняття відбитків RSSI

Подібно до визначення місцезнаходження за допомого потужності сигналу та мультилатерації, зняття відбитків не забезпечує високої точності

позиціонування, якщо система постійно не калібрується відповідно до змін навколишнього середовища. Це недорогий метод Wi-Fi позиціонування, але він вимагає постійного оновлення навчених шаблонів у базі даних.

На підходи зняття відбитків також впливають ослаблення сигналу (найбільший вплив), поглинання сигналу, відображення сигналу та випадкові перешкоди на шляху сигналу.

2.2.3 Метод часу польоту Time of Flight (ToF)

ToF – Це високоточний метод позиціонування, який використовується такими точними технологіями, як UWB. Ця передова техніка може точно вимірювати відстань між Wi-Fi пристроями за допомогою обчислення часу, який потрібен для передачі сигналу між пристроями.

На рисунку 2.3 зображено схему методу часу польоту:

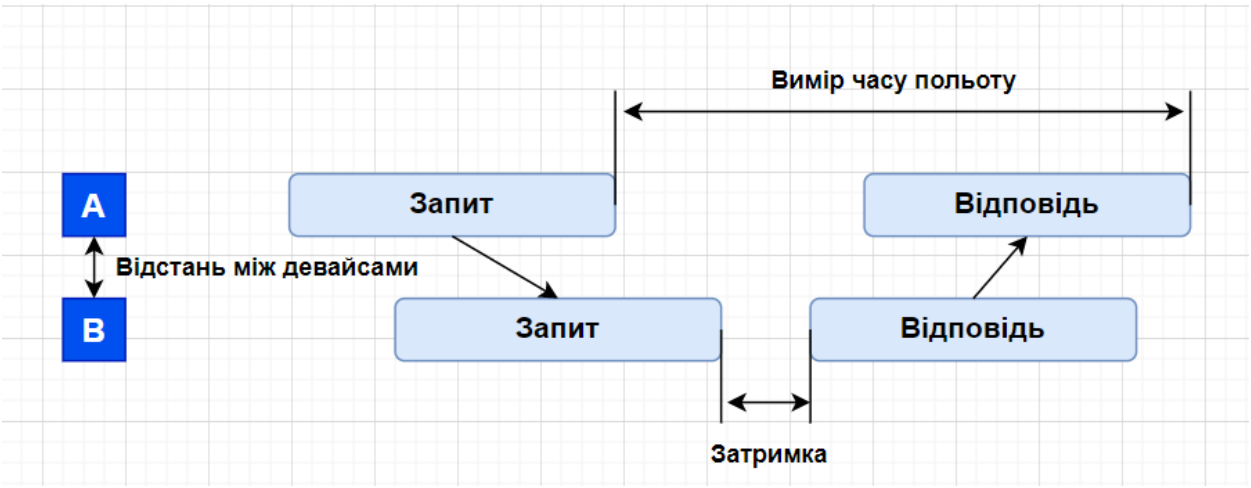

Рисунок 2.3 – Схема методу часу польоту

ToF можна використовувати для визначення точного розташування пристрою Wi-Fi за допомогою кількох датчиків або точок доступу. Для цього

потрібне цільне розгортання точок доступу або датчиків для виявлення Wi-Fi пристроїв, таких як смартфон або тег відстеження.

Щоб працювати належним чином, датчики або точки доступу Wi-Fi мають бути точно синхронізовані з одним головним годинником. Сигнали Wi-Fi пристрою будуть отримані точками доступу або датчиками в діапазоні зв'язку та з позначкою часу.

Потім усі дані з міткою часу надсилаються до центрального ISP або RTLS. Система локації буде аналізувати дані кожного якоря та різницю в часі прибуття до кожного якоря, а після використовуватиме мультилатерацію задля точного обчислення координат мітки.

Хоча ToF і забезпечує точніше Wi-Fi позиціонування, проте розгортання цього підходу пов'язане з більшим рівнем складності та може бути нерентабельним у сценаріях, коли висока точність не потрібна. Wi-Fi Round-Trip-Time (Wi-Fi RTT) – це новий метод визначення місцезнаходження пристроїв Wi-Fi, який використовує ToF. Зазначений у стандарті IEEE 802.11-2016, Wi-Fi RTT дозволяє пристроям вимірювати відстань між пристроями за допомогою часу проходження сигналу між пристроями.

Цей розрахунок базується на шляху, що був пройдений сигналом і визначається за допомогою швидкості передачі електромагнітної хвилі та швидкості світла. Це можна використовувати для визначення діапазону між двома пристроями або для позиціонування в приміщенні за допомогою кількох точок доступу або датчиків із мультилатерацією.

2.2.4 Метод куту прибуття Angle of Arrival (AoA)

Метод куту прибуття – це передовий метод, який може забезпечити Wi-Fi позиціонування з підвищеною точністю порівняно з більш традиційними методами, такими як зняття відбитків та RSSI.

Це стає можливим завдяки Wi-Fi-інтерфейсу Multiple Input Multiple Output (MIMO). Щоб мати можливість визначити напрямок, мобільний пристрій, такий як тег або маяк, з однією антеною передає дані на фіксований датчик Wi-Fi із системою з кількох антен.

Фазовий зсув кількох антен у результаті прийому сигналу вимірюється та обчислюється для визначення кута передавального мобільного пристрою та створення зони визначеності об’єкта, який потрібно знайти.

Однією з переваг методу кута прибуття є те, що він зменшує кількість необхідних опорних точок. Замість мінімум трьох датчиків, які необхідні для будь-якого мультилатераційного методу, вам потрібні лише два для однозначного визначення положення.

Додаткові опорні точки підвищують точність і надійність обчислених позицій. Хоча визначення позиціонування в приміщення за допомогою методу кута прибуття є більш точним, ніж інші методи до потужності сигналу, рішення, які використовують цю техніку, тільки-но виходять на ринок.

2.3 Точність Wi-Fi позиціонування

Так наскільки ж точним може бути позиціонування за допомогою Wi-Fi? Wi-Fi 5 і нижче зазвичай менш точні, ніж інші радіочастотні технології, такі як Ultra Wideband Beacons (UWB) та Bluetooth Low Energy (BLE), і досягають точності визначення Wi-Fi місцезнаходження як правило, менше 10 метрів (за оптимальних умов і розгортання).

Поточний стандарт технології Wi-Fi 6 обіцяє точність у метровому діапазоні, проте оскільки це нова технологія, це ще не було продемонстровано та

протестовано в польових умовах. Минуле оновлення стандарту 802.11az, яке відбулося у березні 2021 року, також називається наступним поколінням позиціонування Next Generation Positioning (NGP), обіцяє покращену точність визначення Wi-Fi місцезнаходження.

При використанні лиже точок доступу точність Wi-Fi позиціонування значно нижча, однак датчики з підтримкою Wi-Fi можна додати для посилення відстеження традиційних точок доступу або навіть як окремі датчика для точнішого позиціонування в приміженні.

Різні методи Wi-Fi позиціонування також можуть забезпечувати різний ступінь точності. Традиційні підходи, як-от мультилатерація RSSI та зняття відбитків, забезпечують точність, яка є значно нижчею за більш просунуті методи такі як метод куту прибуття, метод часу польоту та Wi-Fi RTT.

2.4 Діапазон Wi-Fi позиціонування

Діапазон Wi-Fi позиціонування може відрізнятися в залежності від таких факторів, як використання точок доступу Wi-Fi чи датчиків або характеру внутрішнього простору. Wi-Fi, що працює на частоті 2,4 ГГц, як правило, працює на відстані до 100 метрів (за оптимальних умов і розгортання) і може використовуватися навіть поза приміщеннями, де є відповідна інфраструктура.

Однак Wi-Fi, що працює на частоті 5 ГГц, зменшує діапазон завдяки вищій частоти, працюючи приблизно на 50% від досяжного діапазону 2,4 ГГц. Крім того, на частоті 5 ГГц слід очікувати більш високого послаблення сигналу, якщо сигнали мають проходити через перешкоди навколишнього середовища, такі як двері, стіни або металізовані вікна.

2.5 Відмінності технології Wi-Fi від інших

Технологія Wi-Fi, як і інші радіочастотні стандарти, пропонує унікальні характеристики та переваги, які можуть зробити його прийнятим варіантом залежно від індивідуальних потреб, бюджету проекту, об’єкта та конкретних випадків використання на основі місцезнаходження.

Найбільш важливими відмінностями між Wi-Fi та іншими технологіями є його здатність використовувати наявну інфраструктуру Wi-Fi та гнучкість для використання в багатьох програмах, які використовують місцезнаходження. Wi-Fi є в пристроях майже у всіх приміщеннях, використовується багатьма системами відстеження місцезнаходження, а також його можна розширити для позиціонування всередині приміщень у цілому наборі галузей і випадків використання.

2.5.1 Різниця між Wi-Fi та BLE

Wi-Fi та BLE є двома найпоширенішими радіочастотними технологіями, присутніми в нашому повсякденному житті та в приміщеннях. Вони мають багато подібних характеристик зокрема роботу в діапазоні частот 2,4 ГГЦ, великі екосистеми, а також відоме використання для позиціонування всередині приміщень.

BLE та Wi-Fi в основному використовують RRSI для визначення місцезнаходження людей, пристроїв та активів, однак відомо, що BLE досягає вищого ступеня точності визначення місцезнаходження. BLE потребує значно менше енергії, що забезпечує більш гнучкі варіанти апаратного забезпечення програм.

Однак багато організацій мають існуючу інфраструктуру Wi-Fi, яку можна використовувати для позиціонування всередині приміщень, тоді як впровадження BLE, ймовірно, вимагає інтеграції нових маячків та датчиків тощо. Wi-Fi може обмінюватися даними на великих відстанях і з високою швидкістю передачі даних в обох областях, де BLE набагато обмежені.

2.5.2 Різниця між Wi-Fi та UMB

Широка присутність Wi-Fi у наших пристроях і в приміщеннях зробила її ключовою радіочастотною технологією для розміщення всередині приміщень із досить низьким бар'єром для входу на ринок. У розширених сценаріях на основі визначення місцезнаходження Wi-Fi може бути обмежено через його низьку точність. UMB перевершує ці більш просунуті програми, де потрібен високий ступінь точності. В таблиці 2.1 наведене порівняння різних технологій з Wi-Fi.

Таблиця 2.1 – Порівняння технологій радіо сигналів

	Wi-Fi	**BLE**	**UWB**
Точність розташування	< 10м	< 5м	~40см
Діапазон	0-50м (до 500м)	0-25м (до 100м)	0-50м (до 200м)
Затримка	3-5с	3-5с	< 1 мс
Споживання енергії	Середнє	Дуже мале	Мале
Ціна	Мала (якщо є пристрій)	Середня	Середня
Частота	2,4 ГГц, 5 ГГц	2,4 ГГц	3,1 – 10,6 ГГц
Швидкість	До 1 Гб/с	До 2 Мб/с	До 2 Мб/с

Точність Wi-Fi набагато нижча, ніж UMB, оскільки він зазвичай вимірює місцезнаходження не за відстанню, а радше за силою сигналу, як і Bluetooth. Також більша ймовірність виникнення якихось перешкод сигналу, до яких UWB має дуже велику та сильну резистентність.

UWB також потребує менше енергії, що дозволяє використовувати більш корисні та доступні інструменти, такі як мітки відстеження активів, які можуть живитися від довговічних батарейок типу «таблетка».

Незважаючи на певні недоліки Wi-Fi, широкий спектр пристроїв із підтримкою Wi-Fi та можливість використовувати наявну інфраструктуру, таку як точки доступу, роблять її важливою технологією визначення місця розташування в приміщенні, особливо коли не потрібен високий ступінь точності.

2.6 Варіанти використання Wi-Fi

Повсюдна поширеність Wi-Fi і здатність легко активуватися робить його ефективним варіантом для великої кількості випадків використання внутрішнього позиціонування. Нижче наведено кілька варіантів використання та додатків, у яких використовується Wi-Fi у приміщенні.

Відстеження співробітників і персоналу – організації можуть використовувати точки доступу Wi-Fi, мітки персоналу та датчики для створення видимості місцезнаходження співробітників і персоналу для роботи безлічі додатків, що визначають місцезнаходження:

а) виявлення бездротових пристроїв – об’єкти, які піклуються про безпеку, включно з конфіденційними урядовими та корпоративними будівлями, можуть використовувати датчики для виявлення Wi-Fi та інших пристроїв, що передають радіочастоти, у своєму внутрішньому просторі;

б) оптимізація робочого місця - підвищує операційну ефективність і продуктивність за допомогою недорогих тегів, таких як значки

співробітників, наклейки для відстеження тощо, щоб створити видимість використання простору та розташування співробітників у просторі;

в) безпека працівників - створює безпечніші внутрішні простори та швидко виявляє місцезнаходження та сповіщає працівників у разі надзвичайних ситуацій або евакуації;

г) готовність до робочого місця - підтримує протоколи та нормативні вимоги, які допомагають запобігти та пом’якшити поширення хвороби у просторі за допомогою інструментів, які дозволяють відстежувати контакти на основі пристрою, обізнаність щодо дотримання фізичного дистанціювання, ефективне керівництво санітарією тощо.

Відстеження активів - Wi-Fi є ефективним радіочастотним стандартом для відстеження фізичних активів. Організації в різних галузях можуть використовувати технологію Wi-Fi для відстеження поточного місцезнаходження та стану ключових активів і обладнання:

а) корпоративні простори - покращує продуктивність і розподіл ресурсів, створюючи чітку картину ресурсів, активів і обладнання у великих корпоративних будівлях і приміщеннях;

б) охорона здоров’я - додає можливості відстеження активів, щоб допомогти швидко знаходити та відстежувати місцезнаходження ключового обладнання, наприклад вентиляторів та інвалідних візків;

в) розумне виробництво - створює видимість розташування та переміщення обладнання, машин і ресурсів;

г) управління складом - включає відстеження активів, щоб знайти обладнання, інструменти та інвентар на великих підприємствах.

Служби визначення місцезнаходження - завдяки інтеграції внутрішнього позиціонування Wi-Fi організації можуть створювати розумні будівлі, які використовують місцезнаходження для полегшення різноманітних взаємодій та обміну повідомленнями, серед інших можливостей.

а) безпосередній обмін повідомленням - створює привабливі умови для клієнтів, визначаючи пункти призначення поблизу користувачів і активів і використовує ці дані для безпосереднього залучення відвідувачів гіперлокальним вмістом, таким як купони, маркетингові кампанії, найближчі об’єкти інтересу тощо;

б) геозонування - створює віртуальні географічні межі навколо різних зон внутрішнього простору, які запускають певну дію, коли користувачі входять, виходять або перебувають у певних зонах;

в) надсилання геоданих - дозволяє користувачам ділитися своїм поточним місцем розташування або визначати місцезнаходження інших, зокрема членів родини, друзів або колег, у великих будівлях.

Навігація в приміщенні - робить простір миттєво знайомим і зручним для дослідження за допомогою внутрішньої навігації та визначення шляху за допомогою позиціонування Wi-Fi.

а) пошук маршруту з синьою точкою - вмикає покрокову навігацію та пошук шляху за допомогою блакитної точки, яка показує користувачам, де саме вони знаходяться в приміщенні. Створює надзвичайно точні умови роботи в приміщеннях для об’єктів у багатьох галузях промисловості, включаючи корпоративні підприємства, охорону здоров’я, роздрібну торгівлю, готельний бізнес, транспорт тощо.

Бізнес-аналітика - Wi-Fi можна використовувати для отримання цінних даних про місцезнаходження, які можна перетворити на ефективну організаційну інформацію.

а) аналітика та бізнес-аналітика - створіть видимість того, як відвідувачі взаємодіють із внутрішніми приміщеннями та пересуваються ними, щоб ви могли приймати розумніші та більш обґрунтовані бізнес-рішення.

3 АПРОКСИМАЦІЯ ДАНИХ

3.1 Перенос плану будівлі до математичного виду

Кожна будівля має свій унікальний план побудови і він може відрізнятися навіть у будівель, що були збудовані за одним й тим же проектом. Ця проблема більш видна у великих торгівельних центрах, в яких торгівельні павільйони можуть бути перебудовані у короткий час згідно з потребами продавців.

Приклад плану будівлі, що буде розглядатися для дослідження апроксимації наведено на рисунку 3.1:

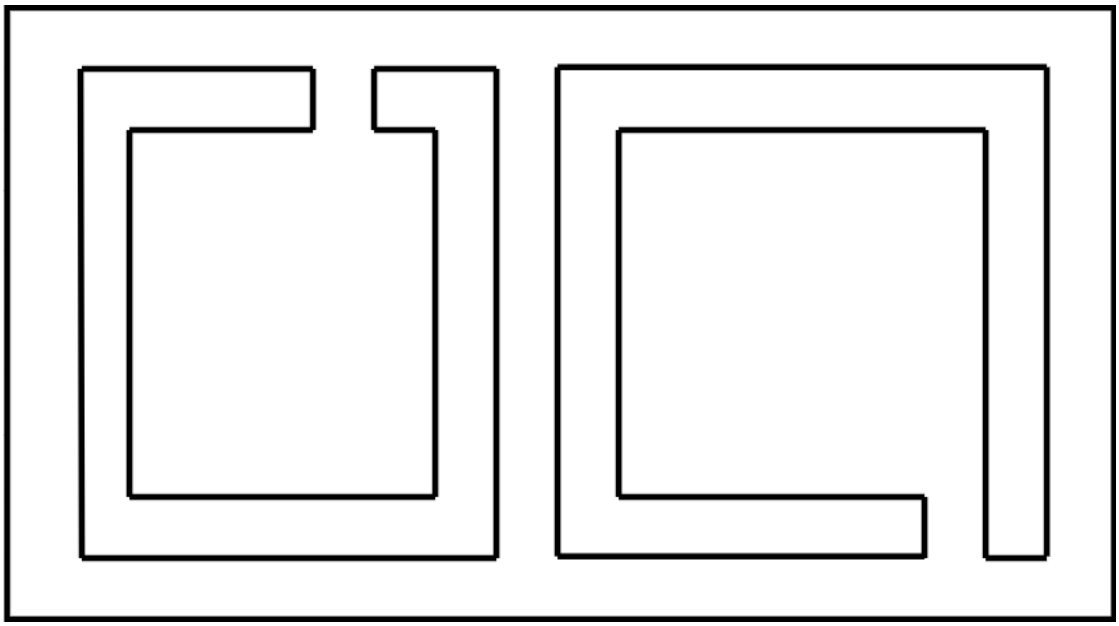

Рисунок 3.1 - План розглядаємої будівлі

У плані, що був використаний для прикладу внутрішні стіни позначені окремим об’ємним чином для майбутнього переводу у математичний вид[5].

Зробимо наступні перетворення поточного плану для спрощення процесу переводу його у математичний вигляд:

а) пофарбуємо стіни та місця, де не можна пройти у червоний колір;

б) місця виходу з кімнат або торгівельних точок пофарбуємо у зелений колір;

в) місця всередині кімнат або торгівельних точок пофарбуємо у жовтий колір;

г) коридори та проходи навколо кімнат пофарбуємо у синій колір.

На рисунку 3.2 наведено отриманий результат після визначення необхідних об’єктів з плану будівлі:

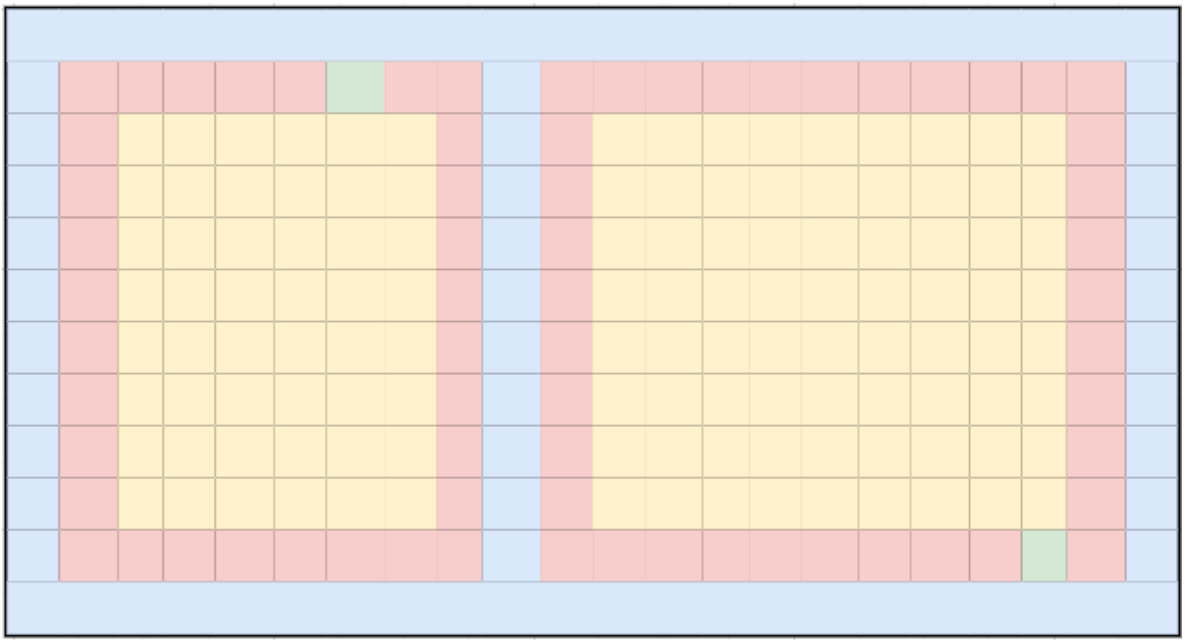

Рисунок 3.2 - перетворений план будівлі

В результаті цих перетворень ми отримали план будівлі, що поділений на окремі однакові квадрати. Кожен з цих квадратів має свій колір и може отримати числове значення для подальшої математичної ідентифікації.

Після розбиття плану на окремі квадрати і їх розфарбовуванні пронумеруємо кожен з них у наступному вигляді:

а) «0» для квадратів через які не можна пройти;

б) «1» для усіх квадратів, які знаходяться всередині кімнат або торгівельних точок;

в) «2» для усіх виходів з кімнат або торгівельних точок;

г) «3» для усіх квадратів, які можуть бути пройдені;

д) «4» для усіх квадратів, які слугують виходами з будівлі. Це може бути як вихід, так і сходи або ліфт;

Результат нумерації квадратів для наведеного плану будівлі можна побачити на рисунку 3.3:

4	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
3	0	0	0	0	0	**2**	0	0	3	0	0	0	0	0	0	0	0	0	0	0	3
3	0	1	1	1	1	1	1	0	3	0	1	1	1	1	1	1	1	1	1	0	3
3	0	1	1	1	1	1	1	0	3	0	1	1	1	1	1	1	1	1	1	0	3
3	0	1	1	1	1	1	1	0	3	0	1	1	1	1	1	1	1	1	1	0	3
3	0	1	1	1	1	1	1	0	3	0	1	1	1	1	1	1	1	1	1	0	3
3	0	1	1	1	1	1	1	0	3	0	1	1	1	1	1	1	1	1	1	0	3
3	0	1	1	1	1	1	1	0	3	0	1	1	1	1	1	1	1	1	1	0	3
3	0	1	1	1	1	1	1	0	3	0	1	1	1	1	1	1	1	1	1	0	3
3	0	1	1	1	1	1	1	0	3	0	1	1	1	1	1	1	1	1	1	0	3
3	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	**2**	0	3
3	3	3	3	3	3	3	**4**	3	3	3	3	3	3	3	3	3	3	3	3	3	3

Рисунок 3.3 – Нумерація квадратів плану будівлі

Після розбиття плану будівлі на окремі квадрати та нумерації цих квадратів задля точної ідентифікації місцезнаходження конкретного квадрату на карті можна представити план будівлі як матрицю[6].

На рисунку 3.4 наведена матриця, отримана в результаті апроксимації плану будівлі:

$$
\begin{pmatrix}
4 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\
3 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 \\
3 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 3 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 3 \\
3 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 3 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 3 \\
3 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 3 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 3 \\
3 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 3 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 3 \\
3 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 3 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 3 \\
3 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 3 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 3 \\
3 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 3 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 3 \\
3 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 3 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 3 \\
3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 3 \\
3 & 3 & 3 & 3 & 3 & 3 & 3 & 4 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3
\end{pmatrix}
$$

Рисунок 3.4 – атриця апроксимації плану будівлі

Ця матриця і буде представляти з себе математичну модель плану будівлі, яка потрібна для знаходження оптимального шляху[7].

4 ПОШУК НАЙКОРОТШОГО ШЛЯХУ

4.1 Аналіз існуючих алгоритмів

Для знаходження найкоротшого шляху з однієї точки матриці до іншої скористаймося графом та методами пошуку шляху між вершинами графів.

На даний момент основними методами пошуку найкоротшого шляху у графі є метод пошуку в ширину Breadth-first search (BFS) та пошук у глибину Depth-first search (DFS). Розглянемо та дослідимо обидва ці алгоритми пошуку всередині графів.

Також після опрацювання графів та отримання найкоротших маршрутів з однієї вершини графа до іншої можна використовувати транспортну задачу лінійного програмування для знаходження оптимального шляху.

4.2 Пошук в ширину (BFS)

Сам пошук працює шляхом послідовного огляду окремих рівнів графу починаючи з вихідної вершини. Для цього потрібно розглянути всі ребра графу, що виходять з конкретної вершини. Якщо чергова вершина, котру опрацював алгоритм є цільовою – то пошук завершується, в іншому випадку вершина додається до черги. Після того, як будуть перевірені усі ребра, що виходять з вершини, з черги прибирається наступна вершина и весь процес повторюється[8].

З цього опису можна виділити наступні кроки:

Крок 1. Зазначити вершину графу з якої починати пошук і додати її до пустої черги.

Крок 2. Дістати з початку черги вершину графу і вважати її обробленою, якщо ця вершина є цільовою – завершити пошук. В іншому випадку у кінець черги треба додати усі вершини, що з'єднані ребрами с цільовою вершиною, не були опрацьовані та не знаходяться у черзі в даний момент.

Крок 3. Якщо черга стала порожня – це означає, що усі вершини графу були оброблені, а цільова вершина не може бути досягнута.

Крок 4. Інакше треба повернутись до п.2 та повторити процес.

На рисунку 4.1 наведено процес алгоритму роботу пошуку в ширину для графа:

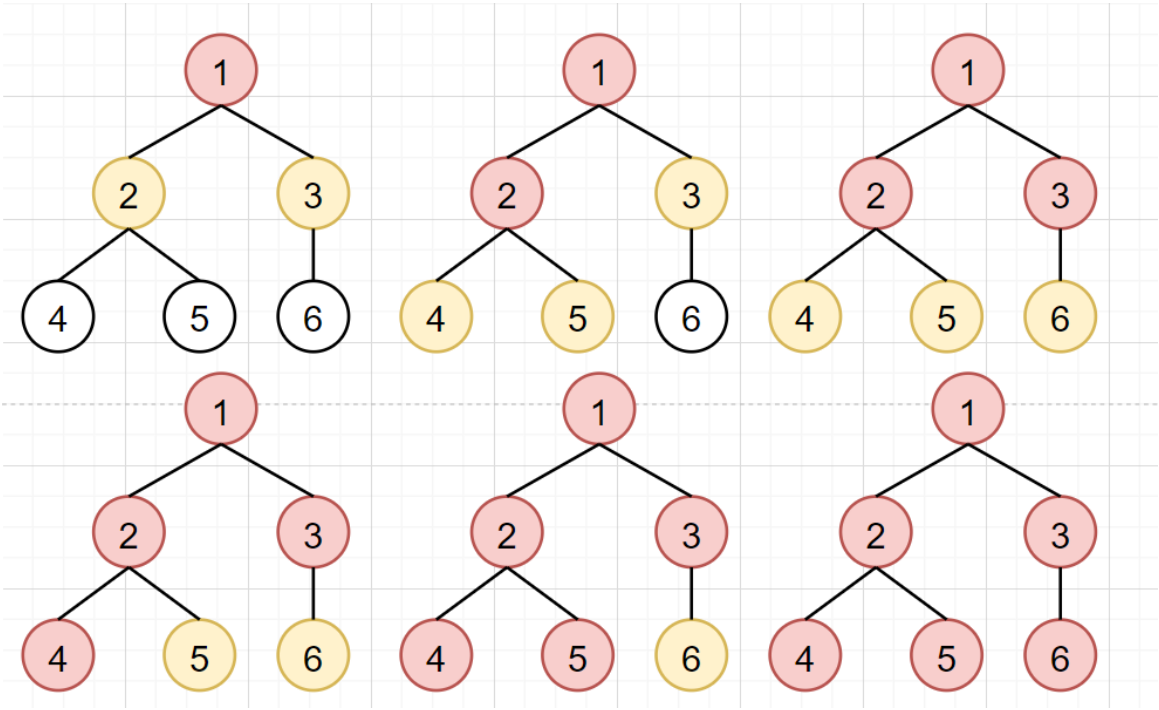

Рисунок 4.1 – Процес пошуку в ширину

Таким чином видно, що пошук в ширину працює так би мовити по рівням графу. На рисунку 4.1 червоним кольором позначені опрацьовані вершини графа, жовтим кольором позначені вершини, що додані до черги опрацювання, та білим кольором позначені вершини, що не були ще знайдені.

4.3 Пошук в глибину (DFS)

Аналогом алгоритму пошуку в ширину є пошук в глибину. Головною стратегією цього алгоритму є обхід графу «в глибину» на скільки це можливо[8].

Цей алгоритм описується рекурсією і має наступні кроки:

Крок 1. Зазначаємо вершину графу з якої починаємо пошук.

Крок 2. Рекурсивно розглядаємо усі ребра, що ведуть від цієї вершини

Крок 3. Якщо ребро веде до вершини, що не була опрацьована раніше – запускаємо алгоритм для цієї вершини, а після повертаємось до інших ребер початкової вершини

Крок 4. Повернення до попередньої вершини відбувається у тому випадку, коли в поточній вершині не залишилось неопрацьованих ребер

Крок 5. Якщо, після завершення алгоритму не всі вершини були розглянуті потрібно запустити пошук для одної з нерозглянутих вершин.

На рисунку 4.2 наведено процес роботи алгоритму пошуку в глибину для графа:

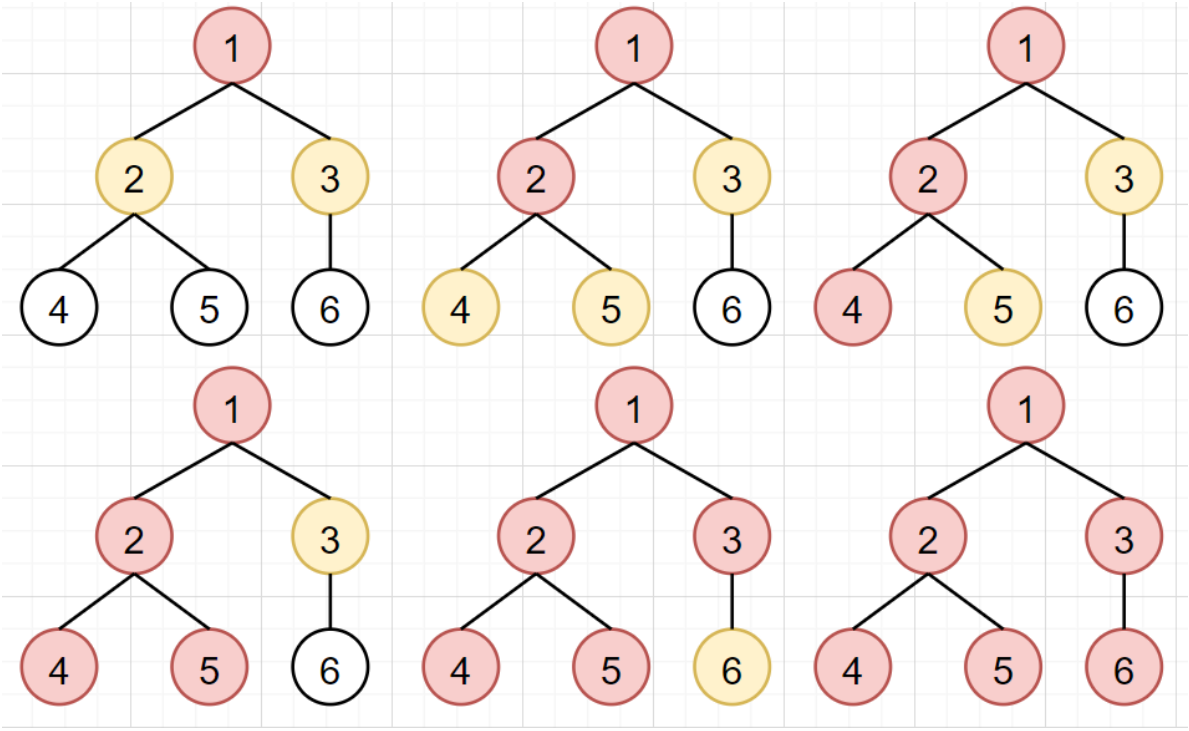

Рисунок 4.2 – Процес пошуку в глибину

Таким чином видно, що пошук в шлибину працює так би мовити в глибину графу. На рисунку 4.1 червоним кольором позначені опрацьовані вершини графа, жовтим кольором позначені вершини, що додані до черги опрацювання, та білим кольором позначені вершини, що не були ще знайдені.

Після знаходження маршрутів за допомогою алгоритмів пошуку по графам будемо використовувати транспортну задачу лінійного програмування задля знаходження самого дешевого шляху.

4.4 Транспортна задача лінійного програмування

Транспортна задача – математична задача лінійного програмування професійного виду. Її можна перебрати як звичайне повітряне перевезення вантажів з пункту відправлення до пункту споживання з урахуванням витрат на перевезення.

Транспортна задача з рахунку складності обчислень входить до класу складності P. Коли обсяг об’ємних пропозицій не дорівнює загальному обсягу обсягів споживання товарів у пункти споживання, то така транспортна задача називається незбалансованою або відкритою[9].

Розглянемо транспортну задачу у матричній постановці та її властивості. Дана задача зводиться до визначення такого плану перевезень певного продукту з пунктів його виробництва до пунктів споживання $|x_{i,j}|_{mxn}$ який мінімізує цільову функцію зображену на рисунку 4.3:

$$f(x) = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{i,j} x_{i,j}$$

Рисунок 4.3 – Цільова функція транспортної задачі

Дана цільова функція повинна виконуватись на множині допустимих планів, зображених на рисунку 4.4:

$$D = \{ x \in R^{mn} \mid \sum_{j=1}^{n} x_{i,j} = a_i, \qquad i \in \mathbf{1} : m;$$

Рисунок 4.4 – Множина допустимих планів

Транспортна задача є представником класу задач лінійного програмування і тому має всі якості лінійних оптимізаційних завдань, але одночасно вона має і ряд додаткових корисних властивостей, які дозволили розробити спеціальні методи її вирішення.

Якщо привести умови транспортного завдання до канонічної форми задачі лінійного програмування, то матриця завдання матиме розмірність *(m+n)mn*. Матриці систем рівнянь в обмеженнях мають ранги, рівні відповідно *m* та *n*. Однак, якщо, з одного боку, підсумувати рівняння по *m*, а з іншого - рівняння по *n*, то отримаємо те саме значення. З цього випливає, що одне із рівнянь у системі є лінійною комбінацією інших. Таким чином, ранг матриці транспортного завдання

дорівнює *m+n-1*, і її невироджений базисний план повинен містити *m+n-1* ненульових компонентів.

Процес вирішення транспортної задачі зручно оформляти як послідовності таблиць, структура яких представлена у таблиці 4.1:

$C_{1,1}$ $X_{1,1}$	$C_{1,2}$ $X_{1,2}$	$C_{1,n}$ $X_{1,n}$	A_1
$C_{2,1}$ $X_{2,1}$	$C_{2,2}$ $X_{2,2}$	$C_{2,n}$ $X_{2,n}$	A_2
$C_{m,1}$ $X_{m,1}$	$C_{m,2}$ $X_{m,2}$	$C_{m,n}$ $X_{m,n}$	A_m
B_1	B_2	B_n	

Рисунок 4.5 - Структура транспортної задачі

Рядки транспортної таблиці відповідають пунктам виробництва (в останній клітині кожного рядка вказаний обсяг запасу продукту a_i), а стовпці - пунктам споживання (остання клітина кожного стовпця містить значення потреби b_j). Всі клітини таблиці (крім тих, які розташовані в нижньому рядку і правому стовпці) містять інформацію про перевезення з *i*-го пункту в *j*-й: у лівому верхньому куті знаходиться ціна перевезення одиниці продукту, а в правому нижньому - значення обсягу вантажу, що перевозиться для даних пунктів. Клітини, які містять нульові перевезення ($x_{i,j} = 0$), називають вільними, а ненульові - зайнятими ($x_{i,j} > 0$).

4.5 Побудова допустимого базисного плану в транспортній задачі

За аналогією з іншими завданнями лінійного програмування рішення транспортної задачі починається з побудови допустимого базисного плану.

Найбільш простий спосіб його знаходження ґрунтується на так званому методі північно-західного кута. Суть методу полягає в послідовному розподілі всіх запасів, що є в першому, другому і т. д. пунктах виробництва, за першим, другим і т. д. пунктами споживання. Кожен крок розподілу зводиться до спроби повного вичерпання запасів у черговому пункті виробництва або спроби повного задоволення потреб у черговому пункті споживання.

На кожному кроці q величини поточних нерозподілених запасів позначаються $a_i^{(q)}$, а поточних незадоволених потреб $b_j^{(q)}$. Побудова допустимого початкового плану, згідно з методом північно-західного кута, починається з лівого верхнього кута транспортної таблиці, причому вважаємо $a^{i(0)}= a_i$, $b_j^{(0)}= b_j$.

Для чергової клітини, розташованої в рядку i та стовпці j, розглядаються значення нерозподіленого запасу в і-му пункті виробництва та незадоволеної потреби j-му пункті споживання, з них вибирається мінімальне і призначається як обсяг перевезення між цими пунктами: $x_{i,j}=min\ \{a_i^{(q)}, b_j^{(q)}\}$. Після цього значення нерозподіленого запасу та незадоволеної потреби у відповідних пунктах зменшуються на наступну величину:

$a_i^{(q+1)}= a_i^{(q)} - x_{i,j}$, $b_j^{(q+1)}= b_j^{(q)} - x_{i,j}$

Очевидно, що на кожному кроці виконується хоча б одне з рівнянь*:* $a_i^{(q+1)}= 0$ або $b_j^{(q+1)}= 0$. Якщо справедливо перше, це означає, що весь запас i-го пункту виробництва вичерпаний і необхідно перейти до розподілу запасу в пункті виробництва $i+1$, тобто переміститися до наступної клітини вниз стовпцем. Якщо ж $b_j^{(q+1)} = 0$, значить, повністю задоволена потреба для j-го пункту, після чого слідує перехід на клітинку, розташовану праворуч по рядку. Знову обрана клітина стає поточною, й у неї повторюються всі перелічені операції.

Ґрунтуючись на умові балансу запасів та потреб, неважко довести, що за кінцеву кількість кроків ми отримаємо допустимий план. У силу тієї ж умови кількість кроків алгоритму не може бути більшою, ніж m+n-1, тому завжди залишаться вільними (нульовими) $mn-(m+n-1)$ клітин. Отже, отриманий план є базовим. Ймовірно, що у певному проміжному кроці поточний нерозподілений

запас виявляється рівним поточної незадоволеної потреби ($a_i^{(q)}=b_j^{(q)}$). У цьому випадку перехід до наступної клітини відбувається в діагональному напрямку (одночасно змінюються поточні пункти виробництва та споживання), а це означає «втрату» однієї ненульової компоненти у плані або, іншими словами, виродженість побудованого плану.

Особливістю допустимого плану, побудованого методом північно-західного кута, і те, що цільова функція у ньому приймає значення, зазвичай, далеке від оптимального. Це тому, що з його побудові не враховуються значення $c_{i,j}$. У зв'язку з цим практично для отримання вихідного плану використовується інший спосіб — метод мінімального елемента, у якому під час розподілу обсягів перевезень насамперед займаються клітини з найменшими цінами.

4.6 Незбалансована транспортна задача

Якщо сума одиниць товару постачальників не дорівнює сумі одиниць товару споживачів, то задача є незбалансованим (відкритим), в іншому випадку задача вважається збалансованою (закритою).

Якщо задача незбалансована, то додаємо новий пункт перевезень (фіктивних перевезень) постачальника чи споживача, залежно від надлишку попиту чи пропозиції відповідно. Кількість одиниць товару нового пункту визначається покриттям надлишку попиту чи пропозиції. Цей пункт не повинен брати участь у загальній вартості плану перевезень, тому вартість перевезень в/з цього пункту повинна дорівнювати нулю.

4.7 Алгоритм методу потенціалів для транспортної задачі

Алгоритм починається з вибору деякого допустимого базового плану (початковий план перевезень, складений, наприклад, методом північно-західного кута). Якщо цей план не вироджений, він містить $m+n-1$ ненульових базисних клітин, і з нього можна визначити потенціали u_i і v_j, щоб кожної базисної клітини (тобто тієї, у якій $x_{i,j} > 0$) виконувалася наступна умова:

$v_j - u_i = c_{i,j}$, *якщо* $x_{i,j} > 0$

Змінні u_i називають потенціалами пунктів виробництва, а v_j - потенціалами пунктів споживання. Для цього складіть систему для заповнених клітин плану перевезень: $v_j - u_i = c_{i,j}$; де $c_{i,j}$ - вартість перевезення з пункту i до пункту j. Оскільки система містить $m+n-1$ рівняння та $m+n$ невідомих, то один із потенціалів можна задати довільно (наприклад, прирівняти v_1 або u_1 до нуля). Після цього решта невідомих v_j і u_i - визначаються однозначно.

Ця задача має певний критерій оптимальності. Для того, щоб допустимий план транспортного завдання $x_{i,j}$ був оптимальним, необхідно і достатньо, щоб знайшлися такі потенціали u_i, v_j, для яких виконуються наступні умови:

$v_j - u_i = c_{i,j}$, *якщо* $x_{i,j} > 0$,
$v_j - u_i \leq c_{i,j}$, *якщо* $x_{i,j} = 0$

Обчисливши коефіцієнти зміни вартості $dc_{i,j}$ для незаповнених клітин плану отримуємо наступне рівняння:

$dc_{i,j} = v_j - u_i - c_{i,j}$

Важливо звернути увагу на те що, якщо всі $dc_{i,j}$ виявилися негативними, то отриманий план оптимальний. Якщо є хоча б один позитивний елемент $dc_{i,j}$, далі провідною (опорною) клітиною буде клітина $[i,j]$ (при $dc_{i,j} > 0$).

Щоб знайти новий план перевезень необхідно скласти цикл перерахунку.

Цикл перерахунку являє собою замкнуту ламану лінію, що складається з горизонтальних та вертикальних ліній, кінці яких лежать у заповнених клітинах. Ламана починається і закінчується в опорній клітці. Вузол в опорній клітці вважається позитивним, наступний - негативний і так далі чергуючись. Береться мінімальне за абсолютною величиною значення негативних клітинах. У всіх

негативних клітинах це значення забирається, у позитивних додається. В результаті цих операцій і можна отримати новий план перевезень.

Цикл повторюється до того часу, поки все $dc_{i,j}$ стануть негативними, тобто поки не буде отримано оптимальний план.

5. ФОРМУВАННЯ ВИМОГ

5.1 Загальні вимоги до системи

Основна вимога до програмної системи, що проектується, це забезпечення кінцевого користувача маршрутом, що буде найефективнішим з можливих, до виходу з торгівельного центру або до авто на паркувальному майданчику. Серед факторів, які будуть впливати на розрахунок найефективнішого шляху можна виділити:

а) відстань до паркувального місця;

б) завантаженість шляху до паркувального місця;

в) наявність ліфтів та сходів;

г) завантаженість ліфтів;

д) наявність багатьох виходів до паркувального майданчику.

Проаналізувавши різні можливі рішення цієї задачі в результаті було обрано транспортну задачу з обмеженою пропускною спроможністю.

Транспортна задача є інструментом вирішення різноманітних проблем для великої кількості предметних галузей. Найбільше застосування зазначена задача знайшла себе у економічному секторі, де використовується за своїм прямим призначенням – транспортування продукту з місця їх дислокації до місця їх споживання з мінімальними витратами.

5.2 Функціональні вимоги

Програмна система має виконувати наступні функціональні вимоги:

а) розрахунок завантаженості певного відрізку шляху в торгівельному центрі;

б) розрахунок ефективного шляху від певного місця в торгівельному центрі до виходу з нього чи паркувального місця з його машиною;

в) збереження даних щодо схеми торгівельного центру (плани поверхів, торгівельні зали, сходи, ліфти), статистики завантаженості певних проходів, сходів ліфтів, поточну завантаженість шляху.

5.3 Нефункціональні вимоги

Не зважаючи на невеликий обсяг функціональних вимог, система має досить великий список нефункціональних вимог. Це пов’язано з тим, що насамперед ця система має оперувати даними у своїй системі координат та не використовувати дані мережі Інтернет та GPS.

До нефункціональних вимог буде достатньо віднести:

а) використання предмет-орієнтованої мови розмітки для опису торгівельного центру та шляхів у ньому;

б) можливість масштабування схеми торгівельного центру;

в) можливість додання різних типів факторів та перешкод до схеми торгівельного центру;

г) можливість роботи без доступу до мережі інтернет в умовах локальної мережі;

д) добре задокументовану кодову базу, для полегшення підтримки програмної системи;

е) потенціальна можливість інтеграції з декількома пов’язаними будівлями одночасно;

є) використання отриманих в результаті дослідження методів розв’язку транспортної задачі з обмеженою пропускною спроміжністю.

6 ВИБІР ТЕХНОЛОГІЇ РОЗРОБКИ

6.1 Історія мови програмування Dart та її можливості

Почнемо з того, що Dart – це компільована високорівнева мова програмування з об'єктно-орієнтованою парадигмою. Свій синтаксис Dart успадковує у C та JavaScript. Найчастіше Dart застосовується при розробці мобільних додатків та високонавантажених сайтів, проте його можливості можуть бути значно ширшими.

Перші згадки про нову мову від компанії Google з'явилися в 2011 році. До листопада 2013 року вийшла стабільна версія платформи, призначена для загального доступу. Метою розробників Dart було створення мови програмування, схожого на JavaScript, але позбавленого його "фундаментальних недоліків". Синтаксис мав бути знаомим практикуючим розробникам, щоб на перенавчання не потребувалося багато часу. Dart також спочатку замислювався як інструмент, що однаково підходить для створення ПЗ під різні платформи - від браузерів до мобільних ОС.

2018 року розробники випустили версію Dart 2.0. У ній було введено сильну типізацію і додано новий компілятор. На даний момент оновлення системи виходять приблизно раз на 2-3 місяці.

6.2 Переваги технології Dart

Dart має достатньо багато переваг, розглянемо основні з них:

а) поєднання JIT та AOT компіляцій. У всіх мовах, що компілюються, для того щоб пристрій міг виконати прописані алгоритми, програмний код перекладається в машинний. Провести таку операцію можна двома способами. Just-in-time компіляція передбачає трансляцію коду "на лету".

Це дає певні переваги безпосередньо під час розробки - тестування та налагодження відбуваються швидше, редагування вносяться оперативніше. Щоправда, при використанні готового додатку на практиці воно завантажуватиметься надто довго і, можливо, почне “лагати”. Ahead-of-time компілятор спочатку обробляє код, і потім починається саме виконання програми. Це сповільнює розробку, але на пристрої користувача додаток буде більш "швидким". Dart успішно застосовує обидві концепції, поєднуючи їх плюси та надаючи максимальну продуктивність;

б) проміжна компіляція**.** Код на Dart легко транслюється в код JavaScript, після чого його спокійно можна запустити в браузері. За рахунок цього програма стає, за фактом, крос платформною;

в) використання опціональних типів. В технології Dart не обов'язково прописувати тип змінної безпосередньо відразу при її оголошенні. Компілятор може визначити його самостійно;

г) поліпшений збирач сміття (Garbage collector GC). Коли об'єкти в інтерфейсі програми постійно змінюються, для згладжування анімації важливо оперативно розподіляти пам'ять пристрою і вчасно видаляти сміття. Тому в Dart передбачено просунутий інструмент налагодження, що діє за принципом поколінь об'єктів. З ним система, в якій усі задіяні віджети, по суті, перезбираються заново для кожного кадру, працює без лагів;

д) асинхронні дії. В однопотокових мовах, до яких належить Dart, великий ризик підвисання програми через перекриття потоку надто важким завданням. Щоб уникнути такої помилки, у платформі передбачено механізм під назвою Event Loop. Він дозволяє виконувати дії асинхронно, тимчасово відкладаючи блокуючі потік операції;

е) відкритий код платформи. Спільнота активно бере участь у розвитку та вдосконаленні мови. Відкритий код сприяє цьому. Крім того, команда

розробників завжди залишається на зв'язку з користувачами та враховує їхню думку при формуванні оновлень;

ж) стабільна та плавна робота додатків. Програмне забезпечення на Dart без лагів працює при частоті оновлень монітора 60 FPS. Згладжування анімації та робота без підвисань - це те, за що розробники та замовники люблять продукти на Flutter, головного фреймворку, що використовує Dart;

з) легкість освоєння. Синтаксис мови буде зрозумілий будь-якому розробнику, знайомому з C++, JavaScript чи Python. Всі конструкції, що використовуються, діють передбачувано, відповідно до загальноприйнятих правил ООП.

6.3 Недоліки технології Dart

Хоча технології Dart має багато переваг, не можна забувати про деякі недоліки, що можуть бути виділені конкретно для цієї технології. Вони складаються з:

а) низька популярність. На жаль, вакансій на Dart все ще небагато: в Україні їх всього кілька сотен, і в основному вони включають розробку на Flutter фреймворку. Але, можливо, найближчими роками мова стане популярнішою, і розробники що нею володіють будуть потрібні в більшій кількості;

б) невелике ком’юніті. Ця проблема - наслідок попередньої: розробників на Dart мало, тому спільнота у мови невелика. Людей у ньому мало (порівняно з ти же JavaScript), і з боку ком’юніті мова розвивається не так активно, як могла би;

в) нездатність повністю замінити JS. Dart все ще не може стати аналогом JavaScript і навряд чи зможе найближчими роками. Він не виконується в

популярних браузерах, а фреймворків та бібліотек для нього поки що замало, щоб покрити всі можливості екосистеми JS. Проте незважаючи на це створення бібліотек для Dart хоч і займає певний час, та зробити це дуже легко.

7 ОПИС ПРОГРАМНИХ РІШЕНЬ

7.1 Апроксимація матриці до графу

Після виконання перетворень для переведення плану будівлі до виду матриці потрібно розробити алгоритм, що забезпечить апроксимацію матриці до графу.

Такий підхід забезпечить можливість використання алгоритмів пошуку по графу для знаходження найкоротшого шляху.

Спочатку створимо клас, що буде реалізовувати базову функціональність самого графа. Код даного класу наведено нижче:

```
class Graph<T> {
  final Map<T, List<T>> _adjacencyList = {};

    Graph();

    void insertEdge(T node1, T node2) {
    _adjacencyList.putIfAbsent(node1, () => []).add(node2);
  }
}
```

З коду, наведеного вище, видно що граф має мапу, що складається з вершин, та списку сусідніх вершин.

Також граф має можливість додавати вершину, що має інформацію про сусідні вершини.

Наступним кроком потрібно зробити алгоритм, що буде додавати вершини до графу, на базі матриці. Матриця в мові Dart може бути представлена двовимірним списком, тобто потрібно зробити алгоритм, що буде переводити дані з `List<List<T>>` до інстансу типу `Graph<T>`[10].

Код, що виконує дане перетворення даних наведено нижче

```
extension GraphExtension on Graph<int> {
  void fillWithMatrixData(List<List<int>> matrix) {
    int graphVertexNumber = 1;
    // Cycle to check all matrix rows
```

```
    for (int i = 0; i < matrix.length; i++) {
```

Продовження лістингу коду:

```
      // Cycle to check all columns elements in the row
      for (int j = 0; j < matrix[i].length; j++) {
        final currElement = matrix[i][j];
        if (currElement != 0) {
          if (matrix.isSafePosition(i, j + 1)) {
            insertEdge(graphVertexNumber, graphVertexNumber + 1);
          }
          if (matrix.isSafePosition(i, j - 1)) {
            insertEdge(graphVertexNumber, graphVertexNumber - 1);
          }
          if (matrix.isSafePosition(i + 1, j)) {
            int vertexCount = 0;
            final currRow = matrix[i];
            for (int index = currRow.length - 1; index > j; index--)
{
              if (currRow[index] != 0) {
                vertexCount++;
              }
            }
            final nextRow = matrix[i + 1];
            for (int index = 0; index <= j; index++) {
              if (nextRow[index] != 0) {
                vertexCount++;
              }
            }
            insertEdge(graphVertexNumber, graphVertexNumber +
vertexCount);
          }
          if (matrix.isSafePosition(i - 1, j)) {
            int vertexCount = 0;
            final currRow = matrix[i];
            for (int index = 0; index < j; index++) {
              if (currRow[index] != 0) {
                vertexCount++;
              }
            }
            final prevRow = matrix[i - 1];
            for (int index = prevRow.length - 1; index >= j; index--
) {
              if (prevRow[index] != 0) {
                vertexCount++;
              }
            }
            insertEdge(graphVertexNumber, graphVertexNumber -
vertexCount);
          }
```

```
      graphVertexNumber++;
    }
```

7.2 Реалізація алгоритму BFS

Після того, як в нас є можливість переводити матрицю у граф, ми можемо реалізувати алгоритм обходу графу. Почнемо з реалізації BFS.

Реалізації алгоритму BFS є найчастіше нерекурсивними. Вони використовують структуру даних що називається чергою.

Нижче наведено реалізацію алгоритму BFS на мові Dart з використанням класу `Graph<T>`, що був створений раніше.

```
void bfs(T startNode) {
    final Set<T> visited = <T>{};
    final queue = Queue<T>();
    visited.add(startNode);
    queue.add(startNode);

    while (queue.isNotEmpty) {
      var currentNode = queue.removeFirst();
      print("Visited: $currentNode");
      try {
        if (_adjacencyList[currentNode] == null) {
          throw EmptyException('Empty List..');
        }
        for (T neighbor in _adjacencyList[currentNode]!) {
          if (!visited.contains(neighbor)) {
            visited.add(neighbor);
            queue.add(neighbor);
          }
        }
      } on EmptyException catch (e) {
        print(e);
```

```
    } catch (e) {
      print(e);
    }
```

7.3 Реалізація алгоритму DFS

На відміну від реалізації алгоритму BFS, реалізації DFS є найчастіше рекурсивними. Вони використовують структуру даних Set для того, щоб зберігати значення вершин графу, які вже відвідали.

Нижче наведено реалізацію алгоритму DFS на мові Dart з використанням класу `Graph<T>`, що був створений раніше.

```
void dfs(T startNode, Set<T> visited) {
    try {
      visited.add(startNode);
      print("Visited Node: $startNode");
      if (_adjacencyList[startNode] == null) {
        throw EmptyException('No Elements');
      }
      for (T neighbour in _adjacencyList[startNode]!) {
        if (!visited.contains(neighbour)) {
          dfs(neighbour, visited);
        }
      }
    } on EmptyException catch (e) {
      print(e);
    } catch (e) {
      print(e);
    }
  }
```

В результаті роботи було отримано два алгоритми обходу графу, що можуть використовуватися для побудови маршруту. В наступних пунктах буде проведено тестування швидкодії обох алгоритмів для різних розмірів графу.

7.4 Реалізація алгоритмів транспортної задачі

Після того, як були розроблені алгоритми пошуку найкоротших маршрутів для знаходження оптимального шляху можна використовувати транспортну задачу лінійного програмування.

Двома основними методами розв’язання транспортної задачі є метод північного кута та метод найменшої вартості.

Реалізуємо обидва методи використовуючи базовий клас та спеціальну модель вводу, наведених нижче:

```
abstract class TransportationProblemMethod {
  String get name;

  List<List<int>>
calculateResultMatrix(TransportationProblemInput input);
}

class TransportationProblemInput {
  final List<int> initialSupplies;
  final List<int> initialDemands;
  final List<List<int>> initialCostMatrix;

  TransportationProblemInput({
    required this.initialSupplies,
    required this.initialDemands,
    required this.initialCostMatrix,
```

```
    });
}
```

Повна реалізація алгоритмів північного куту та методу найменшої вартості наведені в додатках Б та В відповідно.

8 ТЕСТУВАННЯ ШВИДКОСТІ РОБОТИ АЛГОРИТМІВ

8.1 Порівняння швидкості роботи алгоритму BFS та DFS

Перевіримо швидкість роботи алгоритму BFS на різних розмірах матриці. Будемо використовувати наступні розміри 3x3, 5x5, 10x10, 20x20, 100x100, 1000x1000. Кожна матриця буде заповнена числами відмінними від 0 задля максимальної загрузки алгоритму[11].

Для перевірки створимо таймер як раз перед виконанням алгоритму і потім вирахуємо різницю у часі між початком та кінцем виконання алгоритму.

Для приведених вище розмірів матриці було отримано наступні результати роботи алгоритму BFS, що вимірюються в мілісекундах:

а) 3x3 – 4 мс;

б) 5x5 – 4 мс;

в) 10x10 – 6 мс;

г) 20x20 – 8 мс;

д) 100x100 – 21 мс;

е) 1000x1000 – 505 мс.

Для приведених вище розмірів матриці було отримано наступні результати роботи алгоритму DFS, що вимірюються в мілісекундах:

а) 3x3 – 2 мс;

б) 5x5 – 3 мс;

в) 10x10 – 3мс;

г) 20x20 – 4 мс;

д) 100x100 – Stack overflow;

е) 1000x1000 – Stack overflow.

З отриманих даних видно, що алгоритм DFS через свою рекурсивну реалізацію не може коректно працювати на великому об’ємі даних.

Проте алгоритм DFS має значно більшу швидкість роботи на відміну від BFS. В результаті цього дослідження можна сказати, що в рамках цієї роботи буде

використовуватись алгоритм BFS оскільки він підтримує більший об’єм даних, хоча і опрацьовує їх за більший час.

ВИСНОВКИ

В роботі було розглянуто основні алгоритми, що можуть використовуватися для знаходження оптимального шляху для користувача.

В ході аналізу предметної галузі було розглянуто проблематику даного рішення, та його вплив на ринок схожих додатків, проте жоден з них не надає повного рішення проблеми, яка описана в даному проекті, були проаналізовані існуючі аналоги, що реалізують функціонал спрощення процесу орієнтування в торгівельних центрах, та аналоги, що реалізують систему навігації у макромасштабах.

Було наведено приклад апроксимації плану будівлі до математичного виду, на базі якого можна перетворювати план будівлі до матричного виду. Цей підхід використовувався за допомогою декількох операцій, що потрібно робити вручну і на базі цього створювати вихідні дані.

В майбутньому на базі цього дослідження можна розглядати нейронні мережі, які будуть отримувати на вхід зображення плану будівлі і користуючись поновленими алгоритмами переводити його у вид матриці, що може бути використана для орієнтування в будівлі.

Найбільш перспективним методом позиціонування наразі є Wi-Fi позиціонування. Хоча воно і не є найбільш точним на сьогоднішній день, проте воно має найбільше перспектив через свою поширеність та невелику ціну, якщо в будівлі вже встановлена мережа Wi-Fi задля інших цілей.

Алгоритми пошуку шляху по графу показали, що алгоритм пошуку в глибину (DFS), що найчастіше задля прискорення виконання має рекурсивний вид на даному етапі не є оптимальним засобом опрацювання великої кількості даних, хоча він і має велику швидкість порівняно з пошуком в ширину.

Ці результати можна використовувати в подальших дослідженнях прискорення роботи обох алгоритмів, оптимізації споживання оперативної та системної пам’яті.

Транспортна задача лінійного програмування є досить сильним методом розробки оптимальності перевезень та переміщень користувача. За допомогою неї та створенню додаткових факторів, що впливають на ціну перевезення або пересування можна визначати оптимальні шляхи для користувачів.

Дослідження у даній галузі можна продовжувати з ціллю оптимізації, знаходження нових впливових факторів та покращення оцінки «ціни» вже існуючих факторів. Також на базі результатів переміщень, можна збирати статистику і базуючись на цьому оновлювати внутрішнє наповнення та плани будівель та приміщень для оптимальнішого використання простору та орієнтації у ньому.

ПЕРЕЛІК ПОСИЛАНЬ

1. Google Мапи [Електронний ресурс] – Режим доступу до ресурсу: https://www.google.com.ua/maps.

2. AParking [Електронний ресурс] – Режим доступу до ресурсу: https://aparking.kz/ru.

3. OpenStreetMap [Електронний ресурс] – Режим доступу до ресурсу: https://www.openstreetmap.org/

4. Mapsted – Wi-Fi positioning systems [Електронний ресурс] – Режим доступу до ресурсу: https://mapsted.com/blog/wifi-positioning-system-explained/

5. Ревенчук І.А. Агарков Є.С. Моделювання доповненої реальності на основі маркерів. Біоніка інтеллекту.-№.-1(96)2021.-С.90-95

6. Sus, B., Tmienova, N., Revenchuk, I., Bauzha, O., Stirenko, S. Gamification approach to the creation of virtual laboratory works and educational courses.- CEUR Workshop Proceedings, 2020, 2711, P. 68–78.

7. Sus, B., Tmienova, N., Revenchuk, I., Vialkova, V. Development of virtual laboratory works for technical and computer sciences.- Communications in Computer and Information Science, 2019, 1078 CCIS, P. 383–394.

8.Tim Roughgarden Algorithms Illuminated: Omnibus Edition 1st Edition, 2022, 690 p.

9. Robert Dorfman Linear programming and economic analysis, 2003, Dover Publication Inc. 525 p.

10. <IAMSUMAN> [Електронний ресурс] – Режим доступу до ресурсу: https://iamsuman.com/breadth-first-search-bfs-depth-first-search-dfs-graph-traversal-using-generic-class-with-null-safety-dart

11. Gayathri Mohan Full Stack Testing: A Practical Guide for Delivering High Quality Software, 2021, 385 p.