Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Системотехніки

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Розробка та дослідження підсистеми планування перевезень комп'ютеризованої системи управління підприємства

Виконала:
студентка 2 курсу, групи ІТПм – 18-1

Ілюніна М.С.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки

Тип програми освітньо-професійна

Освітня програма Інформаційні технології проектування

Керівник *Рябченко І.М*
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ проф. Гребеннік І.В
(підпис) (прізвище, ініціали)

2019 р.

Атестаційна робота оформлена у відповідності до вимог діючих стандартів та методичних вказівок.

Матеріали атестаційної роботи не містять відомостей, що заборонені для опублікування у відкритих виданнях.

Попередній захист проведено.

Керівник атестаційної роботи *Рябченко І.М*

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Системотехніка

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформаційні технології проектування

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

Студентові Ілюніній Марії Сергіївні
(прізвище, ім'я, по батькові)

1. Тема роботи - Розробка та дослідження підсистеми планування перевезень комп'ютеризованої системи управління підприємства
затверджена наказом університету від 04. листопада. 2019 р. № 1627Ст
2. Термін подання студентом роботи до екзаменаційної комісії 18. 12. 2019 р.
3. Вихідні дані до роботи Перелік використаних програмних засобів:ОС Windows, середовище розробки Intellij IDEA.

4. Перелік питань, що потрібно опрацювати в роботі: 4.Вступ; 4.1. Аналіз предметної області та готових рішень; 4.2.1 Поняття автоматизованої системи управління ;4.2.2. Основні завдання транспортної оптимізації; 4.1.3 Проблеми оптимізації транспортних перевезень;4.1.4 Постановка задачі; 4.2 Огляд існуючих математичних методів оптимізації перевезень; 4.2.1 Математичний опис задачі ; 4.2.2 Оглад існуючих алгоритмів 4.2.3 Огляд досліджуваних методів; 4.2.4 Алгоритм Дейкстри; 4.2.5 Паралельний алгоритм Флойда; 4.2.6 Алгоритм A*; 4.3 Оцінка ефективності алгоритмів; 4.3.1 Критерії оцінки складності алгоритму; 4.3.2 Розробка програмного забечення 4.3.3 Огляд середовища та засобів розробки; 4.3.4 Розробка додатку; 4.2 Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) 5.1 Структура системи управління підприємством; 5.2 Залежність часу розрахунку від числа вершин графа і числа процесорів; 5.3 Графічна інтерпретація алгоритму Флойда; 5.4 Графічна інтерпретація алгоритму Дейкстри 5.5 Графічна інтерпретація алгоритму А*; 5.6_ Задання кількості вершин у додатку; 5.7 Задання ваг шляхів для кожної вершини; 5.8 Результат розрахунку найкоротшого шляху 5.9 Кількість виконаних операцій; 5.10 Результат роботи мобільного додатку; 5.11 Результат роботи алгоритму Флойда; 5.12 Результат роботи алгоритму Дейкстри; 5.13 Результат роботи алгоритму А*

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Розділи спеціальної частини	*Рябченко І.М.*		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання атестаційної роботи	01.09.2019	виконано
2	Аналіз завдання, літератури та аналогів з теми атестаційної роботи	02.09.19 – 20.09.2019	виконано
3	Вибір засобів для розробки, технічних вимог до програми	30.09.2019 – 30.10.2019	виконано
4	Структурне проектування	01.11.2019 – 10.11.2019	виконано
5	Вибір середовища розробки програми	10.11.2019 – 15.11.2019	виконано
6	Розробка програми	15.11.2019 – 30.11.2019	виконано
7	Тестування програми	30.11.2019 – 03.12.2019	виконано
8	Оформлення пояснювальної записки та програмної документації	04.12.2019 – 12.12.2019	виконано
9	Оформлення графічної частини та презентаційних матеріалів	12.12.2019 - 14.12.2019	виконано
10	Представлення атестаційної роботи ДЕК	18.12.2019	виконано

Дата видачі завдання _____ _____20__ р.

Студент _____
(підпис)

Керівник роботи _____ _____
(підпис) (посада, прізвище, ініціали)

Завідувачу кафедри
_____СТ_____
(скорочена назва кафедри)

_проф. Гребеннік І.В.
(вчене звання, П.І.Б.)

ЗАЯВА

щодо самостійності виконання атестаційної роботи та можливості її публікації в електронному архіві відкритого доступу ElAr KhNURE

Я, _____Ілюніна Марія Сергіївна_____,

студентка гр.__ІТПм-18-1_____,

здобувач вищої освіти на другому (магістерському) рівні

кафедра ____Системотехніки_____,

заявляю: моя атестаційна робота на тему Розробка та дослідження підсистеми планування перевезень комп'ютеризованої системи управління підприємства

що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

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

Дата Підпис

РЕФЕРАТ

Пояснювальна записка до атестаційної роботи містить: n сторінки, 16 рисунків, 1 таблиць, n джерел.

Об’єктом дослідження є процеси перевезення вантажів автомобільним транспортом
Предметом дослідження є ефективні методи пошуку найкоротшрого шляху на графі та їх програмна реалізація

Метою даної роботи є підвщення ефективності процесів перевезення вантажів за рахунок використання ефективних методів пошуку найкоротшрого шляху на графі та відповідного програмного забезпечення.
В рамках даної роботи було проведено дослідження ефективності методів пошуку найкоротшого шляху на графі. Була досліджена ефективність алгоритмів Дейкстри, Флойда-Уоршолла та алгоритму A*, в залежності від зміни складності шляху. Створене відповідне по та проведені обчислювальні експерименти та вироблена рекомедація з вибору методів.

ВАНТАЖНІ ПЕРЕВЕЗЕННЯ, ГРАФ, НАЙКОРОТШИЙ ШЛЯХ, МЕТОД ФЛОЙДА, МЕТОД ДЕЙКСТРИ, ОПТИМІЗАЦІЯ, JAVA.

ABSTRACT

An explanatory note: n pages, 16 figures, 1 table, n applications..

About the project ϵ process transporting the plumbing in automobile transport

The subject of the effective methodology is to make the shortest hat on the graph of software. Realization

By the way I have given robots ϵ partial efficiency of the process of transportation for the wicker, the most efficient methods of making a short hat for the graph of a program-specific security.

In the framework of this robotic bullet, the previous efficiency methods were carried out in order to make the shortest hat on the graph. Bula location is the efficiency of Dijkstri, Floyd-Warshall algorithm and A * algorithm, in the deposits of the form of folding hat. The author of the experiment was performed by calculating the experiment and testing the method with the vibration method.

VANTAGE MOVED, GRAPH, SHORTEST PATH, FLOYD ALGORITHM, DIJKSTRA'S ALGORITHM, A* ALGORITHM OPTIMIZATION, JAVA.

ЗМІСТ

ВСТУП

Актуальність роботи обумовлена тим, що у порівнянні з багатьма успішними транспортними підприємствами розвинених країн, транспортні підприємства нашої країни значно програють, що в умовах ринку з його жорсткою конкуренцією дає значні збитки. Планування і правильна організація транспортної діяльності підприємств є найважливішим умовою їх виживання. У процесі розвитку, а також у міру зміни економічних умов всі підприємства стикаються з необхідністю вдосконалення своїх управлінських структур. При цьому підприємства переслідують дві основні мети: підвищити ефективність використання їх транспортних ресурсів і та підвищити кількість виручки. Досягненню цих цілей сприяє впровадження оптимізації та автоматизації логістичного управління підприємства, що обумовлює актуальність обраної теми роботи.

При вантажних перевезеннях автомобільний транспорт бере участь практично у всіх взаємозв'язках виробників і споживачів продукції виробничого призначення і товарів народного споживання. На сучасному етапі російські підприємства переглядають існуючі системи управління, впроваджують нові інформаційні системи управління, проводять реорганізацію бізнесу на основі сучасних методів реінжинірингу. Сформована на підприємствах ситуація обумовлює необхідність формування нових методичних основ і розробки практичних рекомендацій з побудови систем управління процесами, як одного з найважливіших умов розвитку вітчизняних підприємств і системоутворюючих чинників підвищення ефективності виробничої і комерційної діяльності.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ГОТОВИХ РІШЕНЬ

Об’єктом дослідження даної дослідницької роботи є процеси перевезення вантажів автомобільним транспортом. Цей напрям діяльності притаманний здебільшого транспортним підприємствам, які займаються транспортуванням вантажів та доставкою необхідних товарів споживачам.

Але такі підприємства здебільшого не мають необхідного оснащення для ефективного управління та планування. Всі процеси врегульовуються вручну, без жодної автоматизації, або вона часткова та не приносить необідних переваг. Тому найбільш вигідним рішенням є впровадження загальної автоматизованої системи управління та планування процесів на піжприємстві, що дозволить комплексно вирішити більшість проблем. Оскільки дослідження буде проходити в рамках такої системи та буде її частиною, необхідно детально розлянути всі аспекти автоматизованих систем управління.

1.1 Поняття автоматизованої системи управління

Автоматизована система управління підприємством - комплекс програм, що підвищує ефективність управління на всіх рівнях шляхом оптимального планування, доступу до самої релевантної інформації, контролю основних бізнес-процесів. Розвиток автоматизованих систем управління забезпечує точність прогнозування і прийняття рішень. [1]

Автоматизовані системи управління перевезеннями формують заявки перевізнику, комерційні пропозиції, накладні, надають вибір заявок по клієнтах,

прикріплення до заявки супутніх документів. За допомогою автоматизованої системи керування вантажними перевезеннями виробляється консолідація вантажів, облік повного фрахту, аналіз зводу по транспортним засобам. В автоматизованій системі управління вантажний станції відстежується рух вантажів, ведеться облік зберігання і перевезення, розраховується вартість надання додаткових послуг.

Впровадження автоматизованих систем керування завдяки делегуванню різних прав доступу надасть керівництву контроль над будь-якими діями інших користувачів в програмі, комплекс управлінської звітності, аналіз основних бізнес-процесів. Рядові ж співробітники отримають доступ тільки до необхідної інформації і модулів по роботі, наприклад, з автоматизованими системами управління автомобільними перевезеннями. [2]

Як відомо, процес виготовлення продукції на підприємстві різного типу супроводжується переміщенням великої кількості різноманітних вантажів: сировини, матеріалів напівфабрикатів, палива, відходів. Протягом виробничого циклу всі ці вантажі піддаються численним переміщенням і вантажно-розвантажувальних операцій, багаторазово збільшує обсяг транспортних робіт. На кожну технологічну операцію припадає декілька транспортних операцій. Це обумовлює великі витрати на транспортні роботи. Транспорт є не тільки засобом переміщення вантажів, а й знаряддям праці, організуючим роботу підприємства в заданому ритмі графіку.

Велике значення транспорту і в своєчасному забезпеченні підприємства матеріальними ресурсами, а також реалізації готової продукції. [3]

Таким чином, транспортне господарство має вирішувати такі завдання: своєчасне забезпечення виробництва всіма видами транспортних засобів і послуг; раціональна організація експлуатації транспортних засобів і підйомних механізмів при мінімальних витратах на транспортування; розвиток технічної бази і механізація всіх трудомістких транспортних процесів.

1.2. Основні завдання транспортної оптимізації

Основне завдання - зниження витрат на перевезення без втрати їх якості. Розглянемо основні типи витрат, характерних для процесу перевезення сировини, готового товару або персоналу:

-Витрати на завантаження / розвантаження, а також доставку товару в торговельну точку.

- Витрати на експлуатацію, поточний і капітальний ремонт ТЗ.

- Паливні витрати.

- Оплата праці водіїв, механіків, експедиторів.

- Сплата податків, зборів, митних зборів.

- Витрати на проїзд по платних автодорогах.

Оптимізація транспортної роботи необхідна при неконтрольованому зростанні описаних витрат. Процес оптимізації транспортних витрат на підприємстві починається з аналізу поточної логістичної стратегії і збору рекомендацій по її корекції.

Аналізу піддаються такі аспекти транспортної системи підприємства:

- спосіб переміщення вантажів;

- вибір типу транспортного засобу та його конкретної моделі;

- підбір компанії-перевізника та інших логістичних посередників;

- схема розташування складських терміналів компанії. [4]

Корекція поточної логістичної стратегії дозволить виробити ефективний методологічний апарат для оптимізації роботи транспорту. Для ефективних змін

логістичний відділ повинен виробити «дорожню карту» і узгодити її з керівництвом і фінансовим підрозділом.[2]

Пріоритетні завдання, які повинні бути вирішені в ході оптимізації процесів транспорту:

1. максимальна автоматизація трудомісткої роботи;

2. своєчасне оновлення і капітальне обслуговування парку транспортних засобів;

3. впровадження автоматизованих логістичних систем, здатних надавати зведені дані про всі перевезеннях за необхідний період. І також система повинна надавати деталізацію по кожній окремій поїздці.

4. в рамках «оздоровлення» логістичного напряму підприємства потрібно дотримуватися таких заходів, які відзначаються на безпеці вантажу або пасажирів, дотриманні часу прибуття-відбуття, збільшенні часу простою на етапі завантаження / розвантаження. [3]

Як оптимізувати транспортну роботу підприємства. Існує три основних напрямки, нововведення в яких забезпечать зниження витрат на перевезення вантажів і пасажирів:

Вибір оптимальних транспортних засобів:

· Експлуатаційні характеристики транспортного засобу формують рівень витрат на транспорт. Важливо, щоб в парку компанії були негабаритні моделі для перевезень в межах населеного пункту;

· Підбір оптимально розташованих розвантажувально-навантажувальних пунктів: склади потрібно аналізувати не тільки за рівнем зручності під'їзду / навантаження, але і за ступенем віддаленості постачальників;

· Аналіз доцільності володіння власним парком транспортних засобів: це актуально для суб'єктів малого бізнесу. Логістичний підрозділ має зіставити витрати на володіння власним парком з розцінками на послуги транспортних фірм. [5]

Після аналізу пріоритетних напрямків логістики підприємства потрібно ініціювати розробку плану оптимізації транспортних витрат на підприємстві. Цей план має, як правило, загальну конфігурацію незалежно від типу підприємства:

- Постановка завдання перед групою менеджерів логістичного відділу;

- Координація міжвідомчої роботи, видання директив і регламентів, що регулюють взаємодію між підрозділами;

- Впровадження єдиних показників ефективності роботи для відділів, пов'язаних з логістикою;

- Залучення топ-менеджменту для ефективної комунікації між відділами та формування звітної документації;

- Делегування повноважень, призначення групи працівників, відповідальних за досягнення планових показників.

Після того як дані управлінські рішення будуть реалізовані, потрібно обрати ефективну методику оптимізації управління транспортними системами.

1.3 Проблеми оптимізації транспортних перевезень

Оптимізація автотранспортних перевезень - це використання методів і технологій, що дозволяють максимально точно розрахувати час керування маршрутами та витратами, пов'язаними з перевезеннями. Вирішувати такі завдання можна за допомогою розрахунків, вироблених співробітниками транспортного відділу, складів, управлінських підрозділів по контролю запасів і

інші відділи, за допомогою комп'ютерної програми. Використовує на практиці зараз - і те й інше.

Визначаючи поняття оптимізації автоперевезень, можна підкреслити - постійне, регулярне удосконалення системи перевезення (доставки, завантаження / розвантаження) вантажів клієнтів. Такі технології на сьогодні фахівцями пропонуються у формі програмного забезпечення. Установка і користування комп'ютерною програмою, здатної точно розраховувати маршрути, витрати, напрямки та інші нюанси, дозволяє не утримувати великий штат в транспортному відділі. А деякі організації вже навіть не мають такого зовсім. Досить навчити оператора-диспетчера або бухгалтера працювати в даному сервісі, і підприємство буде повністю забезпечено ефективним рішенням експедиторських, посередницьких та інших завдань. [4]

Багато підприємств, що займаються вантажоперевезеннями, намагаються знайти шляхи вирішення для наступних питань:

1. Як визначити оптимальність маршрутів?

2. Наскільки можна максимально намітити короткі шляхи для доставки вантажу?

3. На чому, як і де можна заощадити?

Як розпорядитися веденням господарської діяльності та контролю маршрутів, якщо диспетчера немає на місці або диспетчер - новачок в роботі? І інші питання. Будь-яке завдання містить в собі не тільки суть проблеми, що потрібно рішення, а й сам спосіб її врегулювання і поліпшення.

До завдань процесу, який здатний оптимальним чином поліпшити логістику на автотранспортному підприємстві, відносяться наступні відповідні способи: [5]

· В деяких випадках обмеження кількості тари, а також розмірів вантажоперевізних машин.

· Зведення до мінімуму перевантажень з одного транспортного засобу в інший, з однієї тари для транспортування в іншу.

· Організація зручності навантажувальних або розвантажувальних робіт в самому кузові вантажівки.

· Автоматизація маркування вантажів або повне виключення такого звільняє процес доставки від затягування термінів перевезення.

· Укрупнення кількості одиниць або самого вантажу в цілому по машині, з механізацією розвантаження на місці у замовника.

· Облік типу машини для кращої вантажомісткості. Домогтися того, щоб вантажі мали однаковий розмір від замовлення до замовлення.

· Зробити частоту поставок більше, при цьому не збільшувати вартість одиниці вантажу, який перевозиться.

· Зниження витрат на процес упаковки перевезеного товару разом з тим, щоб максимально зберегти його цілісність в процесі переміщення з однієї точки в іншу.

· Стандартизація підбору типів машин під кожне замовлення, його розмір. Швидко визначити, які поставки клієнтам максимальні, а які - мінімальні.

· Зведення до мінімуму витрат на перевезення бракованого товару, поворотної продукції.

· Знизити величину часу, що витрачається на кожен транзит вантажу. Зменшення часу, що витрачається на кожен простій, який відбувається з природних причин - завантаження / розвантаження.

· Зведення до мінімуму витрат, що припадають на доставку вантажу клієнту.

· Зробити рівномірним потік руху вантажних поставок. Особливо це стосується сезонної специфіки. Якщо мова йде про партії поставок, тоді бажано використовувати мінімум кількості машин, задіяних для цих цілей.

· Сортування вантажів таким чином, щоб максимально його вмістити в машину.

· Зниження часу, що витрачається на надходження і отримання тієї чи іншої інформації щодо вантажів, заявок на поставку і самої транспортування. [5]

Наприклад: відомості про простої; оперативна інформація про місце знаходження вантажівок; коли прибула машина в пункт призначення (час вказується з точністю до секунд); своєчасне повідомлення диспетчеру про поломки в дорозі чи інших що виникли перешкоди до благополучного прибуття в пункт призначення та інше.

Таким чином, ми розуміємо, що крім очевидних, існують також і непрямі причини зниження ефективності роботи в сфері транспортних перевезень. Наприклад, погано організовані вимоги до стана автомобілів та їх своєчасному ремонту може згодом істотно ускладнювати швидкість доставки. Або ж, якщо були допущені помилки в вимогах до плана перевезень - тоді цілком очевидно, що водій може, або збитися з курсу, або не врахувати особливості термінів доставки і інші проблеми.

1.3.1 Принципи побудови автоматизованої системи управління транспортними перевезеннями

Принцип глобальної оптимізації — оптимізація структури логістичної системи потребує узгодженості локальних цілей функціонування елементів (ланок) системи з метою досягнення глобального оптимуму.

Принцип моделювання та інформаційно-комп'ютерної підтримки — використання різних моделей: математичних, економіко-математичних, графічних, фізичних, імітаційних тощо.

Аналіз останніх наукових публікацій показує, що проблемам оптимізації транспортних логістичних систем приділяється значна увага, як в нашій, вітчизняній, так і в зарубіжній науковій літературі [1-9]. Так, в роботах [4–8] розглядаються проблеми оптимізації маршрутизації та переміщення вантажопотоків аналітичними методами; в роботі [4] — моделювання транспортно-складської системи на основі теорії масового обслуговування.

В цих роботах використовуються різні аналітичні методи оптимізації режимів роботи транспортних логістичних систем на основі теорії масового обслуговування, нечіткої логіки та генетичних алгоритмів, або аналітичні методи носять узагальнюючий характер. Прогрес інформаційних технологій та інформаційних систем дав змогу значно підвищити ефективність транспортної логістики, а інформаційно- комп’ютерна підтримка посіла належне місце серед ключових логістичних функцій.

1.4 Постановка задачі

В ході даної роботи основною задачею є розробка підсистеми планування перевезень комп’ютеризованої системи управління транспортного підприємства.

Загальна комп'ютеризована система управління підприємства охоплює всі галузі функціонування даного підприємства, включаючи управління

економічною, огранізаційною та транспортною сферою, яка в свою чергу також підрозділяється на зберігання та переміщення продукції, управління складами, персоналом та логістичні задачі. Впровадження саме комп'ютеризованої системи та загальна модернізація управління транспортною сфею дозволить досягти багатьох переваг та економічної вигоди, в порівнянні із застарілими засобами управління. На рисунку 1.1 представлено схему вдосконалення транспорної сфери підприємства за допомогою комп'ютеризованої системи управління. Місце мого дослідження полягає у виборі методів оптимізації планування перевезень. Оскільки саме планування перевезень на даний момент є досить актуальним для транспортних підприємств та вони стикаються з багатьма проблемами, дослідження буде проведено саме в цьому напрямі.

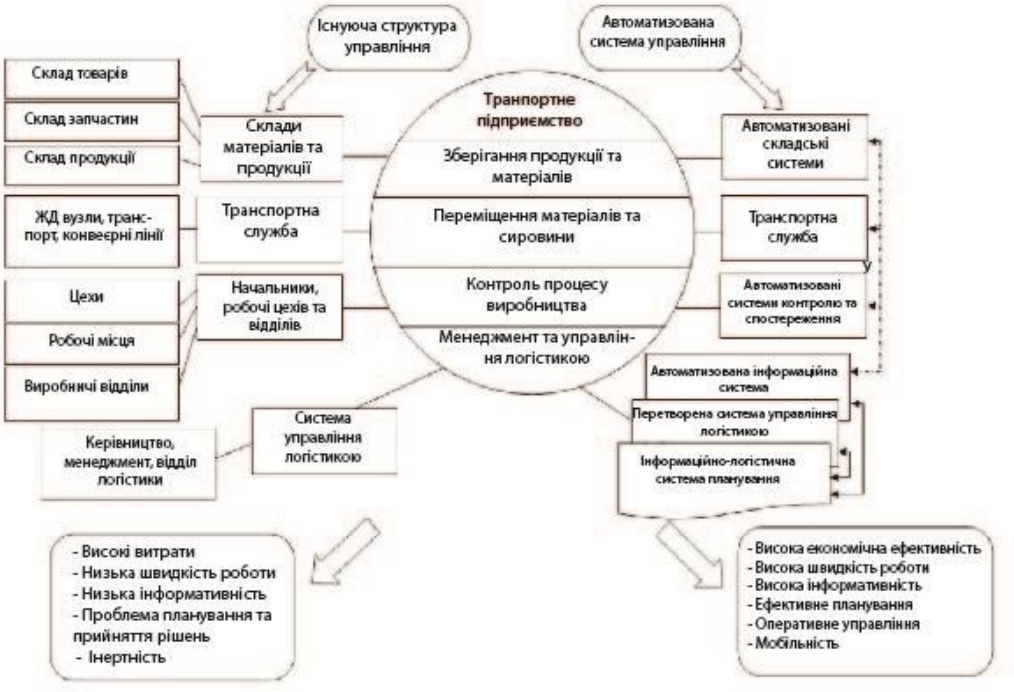


Рис. 1.1 Структура системи управління підприємством

Для вирішення проблем, з якими стикаються транспортні компанії, описаних у попередньому пункті, в ході даної дослідницької роботи, необхідно:

- Проаналізувати основи управління логістичною діяльністю в сфері вантажоперевезень як в межах однієї країни, так і для зарубіжних перевезень

-вибрати математичну модель, яка описує даний процес з метою підбору оптимального алгоритму для побудови маршруту

- провести аналіз математичних методів побудови маршрутів перевезень;

- розглянути критерії оцінки ефективності алгоритмів побудови маршрутів;

- провести дослідження алгоритмів з метою оцінки їх ефективності в залежності від складності маршруту;

- Шляхом порівняння та тестування обрати найбільш оптимальні метода та алгоритми оптимізації побудови маршруту

- Розробити програмний продукт, який вирішить задачу оптимізації та автоматизації транспортних перевезень підприємства для підвищення ефективності планування маршрутів; накопичення і систематизація всієї інформації в базі даних

- Створити сервіс для роботи диспетчерів, що виконуватиме процедуру моніторингу автотранспорту, що являє собою сукупність елементів навігації, організації та аналізу автоперевезень.

Система контролю транспорту здійснює контроль за транспортом по такими параметрами:

- поточне місце розташування (міжнародні перевезення);

- пройдений маршрут по заданих контрольних точках;

- швидкість руху;

Система контролю руху транспорту також дозволяє:

- відобразити маршрути підзвітних об'єктів за будь-який період часу;

- відобразити на карті положення транспортних засобів в поточний момент часу (on-line контроль транспорту);

- зберігати всю інформацію в локальній базі даних системи контролю руху транспорту, що дозволяє не мати постійне підключення до Internet;

- складати шляхові листи в звичній формі і зберігати їх в базі даних; складати звіти про відвідування об'єктів і автоматично зіставити їх з дорожніми листами;

- складати табличні і графічні звіти по витраті палива, пробігу, швидкості, часу в дорозі і т.д. за будь-який період по кожному транспортному засобу або водієві.

На контрольоване транспортний засіб встановлюється комплект бортового обладнання (GPS трекер). GPS трекер визначає місце розташування, швидкість, напрямок руху транспорту. Всі звіти про стан об'єкта і свідченнях датчиків (GPS трекер), з заданою періодичністю архівуються в незалежній пам'яті, незалежно від наявності з'єднання з сервером. Це дозволяє здійснювати повний контроль над автомобілем. На сервері, що надається компанією-постачальником, функціонує база даних, в якій зберігаються дані, прийняті від GPS / GSM терміналів.

Диспетчерський Центр представляє собою звичайний комп'ютер, що має будь-якої доступ до Internet. На ньому, як і на сервері, функціонує база даних, в якій зберігаються дані про контрольовані автомобілях (за кожним автомобілем контроль ведеться індивідуально). періодично здійснюється реплікація даних (передача нової інформації від серверної бази даних до диспетчерської). Така побудова системи моніторингу дозволяє диспетчеру підключатися до Internet періодично і не накладає вимог на швидкість підключення. Диспетчер будує

маршрути на карті, звіти про витрату палива, пробігу, просте і т.д., використовуючи дані з локальної бази даних, аналізує статистику, приймає рішення. Диспетчерський Центр може бути реалізований автономно на стороні Замовника або на стороні Оператора послуг моніторингу з захищеним доступом через Internet.

2 ОГЛЯД ІСНУЮЧИХ МАТЕМАТИЧНИХ МЕТОДІВ ОПТИМІЗАЦІЇ ПЕРЕВЕЗЕНЬ

2.1 Математичний опис задачі

Задача про найкоротший шлях - завдання пошуку найкоротшого шляху (ланцюга) між двома точками (вершинами) на графі, в якій мінімізується сума ваг ребер, що складають шлях.

Задача про найкоротший шлях є однією з найважливіших класичних задач теорії графів. Сьогодні відомо багато алгоритмів для її вирішення

У даній задачі існують і інші назви: завдання про мінімальний шляху або, в застарілому варіанті, задача про диліжанси.

Значимість даного завдання визначається її різними практичними застосуваннями. Наприклад, в GPS-навігаторах здійснюється пошук найкоротшого шляху між двома перехрестями. В якості вершин виступають перехрестя, а дороги є ребрами, які лежать між ними. Якщо сума довжин доріг між перехрестями мінімальна, тоді знайдений шлях найкоротший.

Завдання пошуку найкоротшого шляху на графі може бути визначена для неорієнтованого, орієнтованого або змішаного графа. Далі буде розглянута постановка задачі в найпростішому вигляді для неорієнтованого графа. Для змішаного і орієнтованого графа додатково повинні враховуватися напрямки ребер.

Граф являє собою сукупність непорожньої множини вершин і ребер (наборів пар вершин). Дві вершини на графі суміжні, якщо вони з'єднуються загальним ребром. Шлях в неорієнтованому графі являє собою послідовність

вершин, які зєднані ребрами послідовно. Такий шлях P називається шляхом з довжиною n- вершин. [7]

Нехай ребро з'єднує дві вершини. Дана вагова функція, яка відображає ребра на їх ваги, значення яких виражаються дійсними числами, і неорієнтований граф G. Тоді найкоротшим шляхом з вершини до вершини буде називатися шлях, який має мінімальне значення суми. Якщо всі ребра на графі мають одиничну вагу, то задача зводиться до визначення найменшої кількості обхідних ребер.

Існують різні постановки задачі про найкоротший шлях: [8]

Задача про найкоротший шлях до заданого пункту призначення. Потрібно знайти найкоротший шлях в задану вершину призначення t, який починається в кожній з вершин графа (крім t). Змінивши напрямок кожного належить графу ребра, це завдання можна звести до задачі про єдиної вихідної вершині (в якій здійснюється пошук найкоротшого шляху з заданої вершини в усі інші).

Задача про найкоротший шлях між даною парою вершин. Потрібно знайти найкоротший шлях із заданої вершини u в дану вершину v.

Задача про найкоротший шлях між усіма парами вершин. Потрібно знайти найкоротший шлях з кожної вершини u в кожну вершину v. Це завдання теж можна вирішити за допомогою алгоритму, призначеного для вирішення завдання про одну вихідної вершині, однак зазвичай вона вирішується швидше.

У різних постановках задачі, роль довжини ребра можуть грати не тільки самі довжини, але і час, вартість, витрати, обсяг витрачених ресурсів (матеріальних, фінансових, паливно-енергетичних і т. п.) Або інші характеристики, пов'язані з проходженням кожного ребра. Таким чином, завдання знаходить практичне застосування у великій кількості областей (інформатика, економіка, географія та ін.).

Задача про найкоротший шлях з урахуванням додаткових обмежень.

1. Задача про найкоротший шлях дуже часто зустрічається в ситуації, коли необхідно враховувати додаткові обмеження. Наявність їх може значно підвищити складність завдання. Приклади таких з:

2. Найкоротший шлях, що проходить через задану множину вершин. Можна розглядати два обмеження: найкоротший шлях повинен проходити через виділене безліч вершин, і найкоротший шлях повинен містити якомога менше не виділених вершин. Перше з них добре відома в теорії дослідження операцій.

3. Мінімальне покриття вершин орієнтованого графа шляхами. Здійснюється пошук мінімального за кількістю шляхів покриття графа, а саме підмножини всіх s-t шляхів, таких що, кожна вершина орієнтованого графа належить хоча б одному такому шляху.

4. Задача про обхідні шляхи. Потрібно знайти мінімальну за потужністю множину s-t шляхів таку, що для будь-якого знайдеться шлях, що накриває його — множину деяких шляхів в орієнтованому графі G.

5. Мінімальне покриття дуг орієнтованого графа шляхами. Завдання полягає в знаходженні мінімального за кількістю шляхів підмножини всіх шляхів, такого, що кожна дуга належить хоча б одному такому шляху. При цьому можливе додаткове вимога про те, щоб всі шляхи виходили з однієї вершини. [8]

2.2 Оглад існуючих алгоритмів

У зв'язку з тим, що існує багато різних постановок даної задачі, є найбільш популярні алгоритми для вирішення завдання пошуку найкоротшого шляху на графі: [7]

Алгоритм Дейкстри знаходить найкоротший шлях від однієї з вершин графа до всіх інших. Алгоритм працює тільки для графів без ребер негативного ваги.

Алгоритм Беллмана - Форда знаходить найкоротші шляхи від однієї вершини графа до всіх інших в підвішеному графі. Вага ребер може бути негативним.

Алгоритм пошуку A * знаходить маршрут з найменшою вартістю від однієї вершини (початкової) до іншої (цільової, кінцевої), використовуючи алгоритм пошуку по першому найкращому збігу на графі.

Алгоритм Флойда - Воршелла знаходить найкоротші шляхи між усіма вершинами зваженого орієнтованого графа.

Алгоритм Джонсона знаходить найкоротші шляхи між усіма парами вершин зваженого орієнтованого графа.

Алгоритм Лі (хвильовий алгоритм) заснований на методі пошуку в ширину. Знаходить шлях між вершинами s і t графа (s не збігається з t), який містить мінімальну кількість проміжних вершин (ребер). Основне застосування - трасування електричних з'єднань на кристалах мікросхем і на друкованих платах. Також використовується для пошуку найкоротшої відстані на карті в стратегічних іграх.

Задача про найкоротший шлях між усіма парами вершин для незваженого орієнтованого графа була поставлена Сімбелом у 1953 році, який виявив, що вона може бути вирішена за лінійну кількість операцій (множення) над матрицею. Складність такого алгоритму O (V^4).

Так само для вирішення даної задачі існують інші більш швидкі алгоритми, такі як Алгоритм Флойда - Воршелла зі складністю O (V^3), і Алгоритм Джонсона (є комбінацією алгоритмів Беллмана-Форда і Дейкстри) зі складністю O ($V^E + V^2 \log V$).

Завдання про пошук найкоротшого шляху на графі може бути інтерпретоване по-різному і застосовуватися в різних областях. Далі наведені приклади різних застосувань задачі.

Алгоритми знаходження найкоротшого шляху на графі застосовуються для знаходження шляхів між фізичними об'єктами на таких картографічних сервісах, як карти Google або OpenStreetMap. У навчальному відео від Google можна дізнатися різні ефективні алгоритми, які застосовуються в даній сфері.

Якщо уявити недетермінірована абстрактну машину як граф, де вершини описують стану, а ребра визначають можливі переходи, тоді алгоритми пошуку найкоротшого шляху можуть бути застосовані для пошуку оптимальної послідовності рішень для досягнення головної мети. Завдання пошуку найкоротшого шляху на графі широко використовується при визначенні найменшої відстані в мережі доріг.

Мережу доріг можна представити у вигляді графа з позитивною вагою. Вершини є дорожніми розв'язками, а ребра дорогами, які їх з'єднують. Ваги ребер можуть відповідати протяжності цієї ділянки, часу необхідного для його подолання чи вартості подорожі по ньому. Орієнтовані ребра можна використовувати для подання односторонніх вулиць. В такому графі можна ввести характеристику, яка вказує на те, що одні дороги важливіше інших для тривалих подорожей (наприклад автомагістралі). Вона була формалізована в понятті (ідеї) про магістралях.

Для реалізації підходу, де одні дороги важливіше інших, існує безліч алгоритмів. Вони вирішують завдання пошуку найкоротшого шляху набагато швидше, ніж аналогічні на звичайних графах. Подібні алгоритми складаються з двох етапів:

- етап попередньої обробки. Проводиться попередня обробка графа без урахування початкової і кінцевої вершини (може тривати до декількох днів, якщо

працювати з реальними даними). Зазвичай виконується один раз і потім використовуються отримані дані.

- етап запиту. Здійснюється запит і пошук найкоротшого шляху, при цьому відомі початкова і кінцева вершина.[6]

Інші підходи (техніки), які застосовуються в даній сфері:

-ALT

-Arc Flags

-Contraction hierarchies

-Transit Node Routing

-Reach based Pruning

-Labeling

2.3 Огляд досліджуваних методів

Нехай дано спрямований граф (V, A), де V - безліч вершин і A - безліч ребер, з початковою вершиною обходу s, кінцевою t і вагами дослыджДля кожного ребра (i, j) в A. Вага кожного ребра відповідає змінної програми.

Тоді задача ставиться таким чином: знайти мінімум функції.

Кожній дузі (x, y) вихідного графа G поставимо у відповідність число a (x, y). Якщо в графі G відсутня деяка дуга (x, y), покладемо a (x, y) = ∞. Будемо називати число a (x, y) довжиною дуги (x, y), хоча a (x, y) можна також інтерпретувати як відповідні витрати або відповідний ваговий коефіцієнт. Визначимо довжину шляху як суму довжин окремих дуг, що складають цей шлях.

Для будь-яких двох вершин s і t графа G можуть існувати кілька шляхів, що з'єднують вершину s з вершиною t. В даному розділі буде розглянуто алгоритм, який визначає такий шлях, що веде з вершини s у вершину t, який має мінімально можливу довжину. Цей шлях називається найкоротшим шляхом між вершинами з і t.

Задача полягає в тому, що транспортній компанії, яка перевозить вантаж, необхідно проїхати з пункта А до пункту В, використовуючи магістральні шосейні дороги, що з'єднують ці точки на місцевості.

Для вирішення цієї проблеми треба побудувати граф, вершини якого відповідають точкам з'єднання розглянутих доріг. Нехай кожна дуга графа відповідає шосейній дорозі, що з'єднує пункти, представлені відповідними вершинами. Нехай довжина дуги визначається довжиною (в кілометрах) відповідної ділянки дороги. Тепер завдання пошуку оптимального маршруту руху між А і В може бути зведена до задачі відшукання в побудованому графі найкоротшого шляху між вершинами, відповідними пункту А (складу), звідки починається подорож, і пункту В, де подорож закінчується.[7]

2.4 Алгоритм Дейкстри

У більшості алгоритмів пошуку найкоротших шляхів використовується той факт, що будь-який найкоротший шлях сам складається з найкоротших щляхів. Найбільш ефективний алгоритм вирішення задачі про пошук найкоротшого шляху між двома будь-якими фіксованими вершинами в зв'язковому графі (наприклад, між вершинами s і t) з невід'ємними вагами запропонував в 1959 р Дейкстра [9]. В основі цього алгоритму лежить принцип «жадібності» (на

кожному кроці вибирається локально кращий варіант). Це дозволяє послідовно обчислювати відстані спочатку до найближчої до s вершині, а потім до наступної найближчій і т. д. Метод заснований на приписуванні вершин тимчасових позначок, причому позначка вершини дає верхню межу довжини шляху від s до цієї вершини. Величини цих позначок поступово зменшуються за допомогою деякої ітераційної процедури. На кожному кроці ітерації тільки одна з тимчасових позначок стає постійною і відповідає точній довжині найкоротшого шляху від s до відповідної вершини. Послідовність вершин, через які проходить найкоротший шлях, визначається за допомогою послідовного вирахування з довжини шляху довжин ребер, що лежать на найкоротшому шляху (метод послідовного повернення, зовнішній спосіб).[9]

Розглянемо етапи цього алгоритму.

Нехай l (s) позначає позначку вершини xi (поточна відстань від s до x_i).

Присвоєння початкових значень.

Крок 1. Припустити, що l (s) = 0 і вважати цю позначку постійною. Припустити, що l (xi) = ∞ для всіх xi ≠ s і вважати ці позначки тимчасовими. Припустити, що p = s.

Оновлення позначок

Крок 2. Для всіх $x_i \in$ Г (p), позначки яких тимчасові, змінити їх у відповідності з наступним виразом:

$$l(x_i) = min[l(x_i), l(p) + c(x, p)],$$

c(p,x_i)- довжина шляху з p до x_i, Г(p)- множина суміжних з p вершин графу.

Перетворення тимчасової позначки в постійну

Крок 3.Серед всіх вершин$x_i \in$ Г (p) з тимчасовими позначками знайти таку вершину x_i^* для якої

$$l\ (x_i^*) = \min\ (l(x_i^*)].$$

Крок 4. Вважати позначку вершини x_i^* постійною і припустити, що p=x_i^*.

Крок 5. Якщо p = t, то l (p) є довжиною найкоротшого шляху, якщо $p \neq t$, перейти до кроку 2.[9]

2.5 Паралельний алгоритм Флойда

Як випливає із загальної схеми алгоритму Флойда, основна розрахункового навантаження при вирішенні завдання пошуку найкоротших шляхів полягає у виконанні операції вибору мінімальних значень і номерів попередніх вершин. Очевидно, що алгоритм множення матриць (3.4) легко модифікується в алгоритм підрахунку кратчайших відстаней, відповідних матриці A (N) p. Тому для N >> 1 підрахунок найкоротших відстаней в графі можна реалізувати на основі відомих паралельних алгоритмів обчислення добутку двох матриць [38-40]. В якості фундаментальної підзадачі на k-тій ітерації можна вибрати обчислення елементів aij матриці

шляхів з (3.2), оскільки для будь-якої пари індексів (i, j) на k-тій ітерації елементи a_{ik}^{k+1} та a_{kj}^{k+1} не змінюються. Тому якщо для q процесорів можна виділити $\frac{n^2}{2}$ укрупнених підзадач, то за n ітерацій потрібно виконати $\frac{n^3}{q}$ операцій. Тоді для прискорення і ефективності алгоритму Флойда отримаємо

$$S_q = \frac{n^3}{(n^3/q)} = q, \qquad E_q = \frac{E_q}{q} = 1$$

Відомий паралельний алгоритм, заснований на використанні блочної схеми розбиття матриці примикань A (k) p і дороміжної матриці B (k). Такий підхід передбачає горизонтальне або вертикальне розбиття матриць на стрічки (блоки) однакової розмірності. Розглянемо варіант розбиття матриць на вертикальні

полоси, що відповідають алгоритмічним мовам, у яких елементи матриць зберігаються за стовпцями. При такому способі розбиття даних на кожній ітерації алгоритму Флойда потрібно передавати між процесорними елементами (ПЕ) тільки елементи одного з блоків матриць $A^{(k)}$ і $B^{(k)}$. Паралельний алгоритм для графа з n вершин складається з наступних етапів:

1. Визначення цілого N, де N - найменше 2^s, яке перевищує n-1. Величина s відповідає кількості добутків матриць з послідовності

$$A_p^{(1)} \cdot A_p^{(1)}, A_p^{(2)} \cdot A_p^{(2)}, \ldots, A_p^{(n/2)} \cdot A_p^{(n/2)}$$

2. Завдання розмірності n, елементів матриці примикань $A_p^{(1)}$, допоміжної матриці $B^{(1)}$.Задання номерів і числа ПЄ: $ПЄ_0$, ...,$ПЄ_{q-1}$;

n = L*q, L - кількість смуг.

3. $ПЄ_0$пересилає всім іншим матрицю$A_p^{(1)}$ і допоміжну матрицю $B^{(1)}$.

4. Обчислення елементів відповідної вертикальної смуги матриці $A_p^{(2i)}$, і оновлення вертикальної смуги допоміжної матриці $B_p^{(2i)}$.

5. Кожен ПЕ відправляє свою розраховану смугу матриць $A_p^{(2i)}$ і $B_p^{(2i)}$.

всім іншим ПЕ і заповнює відсутні смуги даними, які надходять від інших процесорів.

6. Виконуються пункти 4-5, доки кількість ітерацій НЕ перевищуватиме розраховане число s.[9]

Згідно із законом Амдаля для прискорення алгоритму, отриманого при

виконанні на ідеальної паралельної машині з q процесорів (без урахування обмінів), має місце формула

$$S_q = \frac{1}{r+(1-r)/q},$$

де r - частка операцій послідовної частини алгоритму.

Реальне прискорення на практиці можна визначити як відношення процесорного часу на виконання послідовної програми до часу виконання обчислень паралельною програмою на q-процесорній машині. При цьому передбачається, що обидві програми реалізують один і той самий алгоритм.

У табл. 2.1 наведено час обчислення за паралельним алгоритмом Флойда для різних значень числа вершин і використаних процесорів для кластера СКІФ Cyberia [9].

Таблиця 2.1 Залежність часу розрахунку від числа вершин графа і числа процесорів

n	Время работы алгоритма					
	q=10	S_q	q=50	S_q	q=100	S_q
500	0,44	8,5	0,26	14,4	0,32	12,5
1000	6,34	8,6	2,78	19,7	1,89	28,9
3000	268,61	9,6	59,06	43,8	32,25	80,2

Як випливає з табл. 2.1, при великих n з ростом числа процесорів час розрахунків істотно скорочується.

Метод Дейкстри шукає найкоротший шлях від заданої вершини (джерела) до всіх інших вершин на графі. Метод Флойда шукає найкоротший шлях по черзі від кожної вершини до всіх інших і цим же він відрізняється від Дейкстри. Матричний метод дуже схожий з методом Флойда, він дозволяє знайти матрицю

D, елементи якої будуть рівні мінімальним величинам маршрутів, а також знайти відстань з кожної вершин в саму себе. [7]

2.6 Алгоритм A*

Пошук A * - алгоритм пошуку по першому найкращому збігу на графі, який знаходить маршрут з найменшою вартістю від однієї вершини (початкової) до іншої (цільової, кінцевої).

Порядок обходу вершин визначається евристичною функцією «відстань + вартість» (зазвичай позначається як f (x)). Ця функція - сума двох інших: функції вартості досягнення даної вершини (x) з початковою (зазвичай позначається як g (x) і може бути як евристичною, так і ні), і функції евристичної оцінки відстані від розглянутої вершини до кінцевої (позначається як h (x)).

Функція h(x) повинна бути допустимою евристичною оцінкою, тобто не повинна переоцінювати відстані до цільової вершині. Наприклад, для завдання маршрутизації h(x) може являти собою відстань до цілі по прямій лінії, оскільки це фізично найменша можлива відстань між двома точками.

Цей алгоритм був вперше описаний в 1968 році Пітером Хартом , Нільсом Нильсоном і Бертрамом Рафаелем. Це по суті було розширення алгоритму Дейкстри, створеного в 1959 році. Новий алгоритм досягав більш високої продуктивності (за часом) за допомогою евристики. В їх роботі він згадується як «алгоритм A». Але так як він обчислює кращий маршрут для заданої евристики, він був названий A*. Узагальненням для нього є двонаправлений евристичний алгоритм пошуку.[9]

У 1964 році Нільс Нільсон винайшов евристичний підхід до збільшення швидкості алгоритму Дейкстри. Цей алгоритм був названий A1. У 1967 році Бертрам Рафаель зробив значні поліпшення за цим алгоритмом, але йому не вдалося досягти оптимальності. Він назвав цей алгоритм A2. Тоді в 1968 році Пітер Е. Харт представив аргументи, які доводили, що A2 був оптимальним при

використанні послідовної евристики лише з незначними змінами. У його доказ алгоритму також включений розділ, який показував, що новий алгоритм A2 був можливо найкращим алгоритмом, враховуючи умови. Таким чином, він позначив новий алгоритм в синтаксисі зірочкою, він починається на A і включав в себе всі можливі номери версій.

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

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

Чим менше евристика h (x), тим більше пріоритет, тому для реалізації черги можна використовувати сортуванні дерева.[9]

Множина переглянутих вершин зберігається в closed, а ті, які потребують розгляду шляху - в черзі з пріоритетом open. Пріоритет шляху обчислюється за допомогою функції f (x) всередині реалізації черги з пріоритетом. Нижче представлено псевдокод алгоритму:

```
function A*(start, goal, f)
    % множество уже пройденных вершин
    var closed := the empty set
    % множество частных решений
    var open := make_queue(f)
    enqueue(open, path(start))
    while open is not empty
        var p := remove_first(open)
        var x := the last node of p
        if x in closed
            continue
        if x = goal
            return p
        add(closed, x)
        % добавляем смежные вершины
        foreach y in successors(x)
            enqueue(open, add_to_path(p, y))
    return failure
```

Властивості

Як і алгоритм пошуку в ширину, A * є повним в тому сенсі, що він завжди знаходить рішення, якщо таке існує.[7]

Якщо евристична функція h допустима, тобто ніколи не переоцінює дійсну мінімальну вартість досягнення мети, то A* сам є допустимим (або оптимальним), також за умови, що ми не відкидаємо пройдені вершини. Якщо ж ми це робимо,

то для оптимальності алгоритму потрібно, щоб h (x) була ще й монотонною, або спадковою евристикою. Властивість монотонності означає, що якщо існують шляхи A-B-C і A-C (не обов'язково через B), то оцінка вартості шляху від A до C повинна бути менше або дорівнює сумі оцінок шляхів A-B і B-C. (Монотонність також відома як нерівність трикутника: одна сторона трикутника не може бути довшим, ніж сума двох інших сторін.) Математично, для всіх шляхів x, y (де y - нащадок x) виконується:

$$g(x) + h(x) \leq g(y) + h(y).$$

A * також оптимально ефективний для заданої евристики h. Це означає, що будь-який інший алгоритм досліджує не менше вузлів, ніж A* (за винятком випадків, коли існує кілька приватних рішень з однаковою евристикою, точно відповідної вартості оптимального шляху).[9]

У той час як A* оптимальний для «випадково» заданих графів, немає гарантії, що він зробить свою роботу краще, ніж більш прості, але і більш інформовані щодо проблемної області алгоритми. Наприклад, в якомусь лабіринті може знадобитися спочатку йти у напрямку від виходу, і тільки потім повернути назад. В цьому випадку обстеження спочатку тих вершин, які розташовані ближче до виходу (по прямій дистанції), буде втратою часу.

Особливі випадки

Загалом кажучи, пошук в глибину і пошук в ширину є двома окремими випадками алгоритму A *. Для пошуку в глибину візьмемо глобальну змінну-лічильник C, ініціалізуємо її якимось великим значенням. Всякий раз при розкритті вершини будемо присвоювати значення лічильника суміжних вершин, зменшуючи його на одиницю після кожного присвоювання. Таким чином, чим раніше буде відкрита вершина, тим більшого значення h (x) вона отримає, а значить, буде переглянута в останню чергу. Якщо покласти h (x) = 0 для всіх вершин, то ми отримаємо ще один спеціальний випадок - алгоритм Дейкстри.

Чому А * допустимий і оптимальний

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

Коли А * завершує пошук, він, згідно з визначенням, знайшов шлях, справжня вартість якого менше, ніж оцінка вартості будь-якого шляху через будь-який відкритий вузол. Але оскільки ці оцінки є оптимістичними, відповідні вузли можна без сумнівів відкинути. Інакше кажучи, А * ніколи не упустить можливості мінімізувати довжину шляху, і тому є допустимим. [9]

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

3 ОЦІНКА ЕФЕКТИВНОСТІ АЛГОРИТМІВ

3.1 Критерії оцінки складності алгоритму

Складність алгоритмів зазвичай оцінюють за часом виконання або по використовуванні пам’яті. В обох випадках складність залежить від розмірів вхідних даних: масив з 100 елементів буде оброблений швидше, ніж аналогічний з 1000. При цьому точний час мало кого цікавить: воно залежить від процесора, типу даних, мови програмування і інших параметрів. Важлива лише асимптотична складність, тобто складність при прагненні розміру вхідних даних до нескінченності.

Припустим, певним алгоритмом потрібно виконати 4n3 + 7n умовних операцій, щоб обробити n елементів вхідних даних. При збільшенні n на підсумкове час роботи буде значно більше впливати зведення n в куб, ніж множення його на 4 або ж поповнення 7n. Тоді кажуть, що тимчасова складність цього алгоритму дорівнює O (n^3), тобто залежить від розміру вхідних даних кубічно.

Використання великої літери O (або так звана O-нотація) прийшло з математики, де її застосовують для порівняння асимптотичної поведінки функцій. Формально O (f (n)) означає, що час роботи алгоритму (або обсяг займаної пам’яті) росте в залежності від обсягу вхідних даних не швидше, ніж деяка константа, помножена на f (n).

Наприклад:

O (n) – лінійна складність

Такою складністю володіє, наприклад, алгоритм пошуку найбільшого елемента в не відсортованому масиві. Нам доведеться пройтися по всіх n елементів масиву, щоб зрозуміти, який з них максимальний.

O (log n) – логарифмічна складність

Найпростіший приклад – бінарний пошук. Якщо масив відсортований, ми можемо перевірити, чи є в ньому якесь конкретне значення, шляхом розподілу навпіл. Перевіримо середній елемент, якщо він більше пошукового, то відкинемо другу половину масиву – там його точно немає. Якщо ж менше, то навпаки – відкинемо початкову половину. І так будемо продовжувати ділити навпіл, в результаті перевіримо log n елементів.

O (n^2) – квадратична складність

Таку складність має, наприклад, алгоритм сортування вставками. У канонічній реалізації він вдає із себе два вкладених цикла: один, щоб проходити по всьому масиву, а другий, щоб знаходити місце чергового елементу в уже відсортованій частині. Таким чином, кількість операцій буде залежати від розміру масиву як n * n, тобто n^2.

Бувають і інші оцінки за складністю, але всі вони засновані на тому ж принципі.

Також трапляється, що час роботи алгоритму взагалі не залежить від розміру вхідних даних. Тоді складність позначають як O (1). Наприклад, для визначення значення третього елемента масиву не потрібно ні запам’ятовувати елементи, ні проходити по ним скільки-то раз. Завжди потрібно просто дочекатися в потоці вхідних даних третій елемент і це буде результатом, на обчислення якого для будь-якої кількості даних потрібен один і той же час.

Аналогічно проводять оцінку і по пам’яті, коли це важливо. Однак алгоритми можуть використовувати значно більше пам’яті при збільшенні

розміру вхідних даних, ніж інші, але працюватимуть швидше. І навпаки. Це допомагає вибирати оптимальні шляхи вирішення завдань виходячи з поточних умов і вимог.

3.2 Розробка програмного забечення

3.2.1 Огляд середовища та засобів розробки

Для проведення необідних обчислювальних експериментів необідно розробити програмне забезпечення, що буде тестувати методи вибору шляху для планування транспортних перевезень підприємства. Для цього програмний додаток повинен відображати основні функції, які вимагатиме від нього компанія, враховувати різні обставини, які можуть виникати при вирішенні цільових задач та залишатись актуальнім протягом довгого часу. Тому перш за все стоїть питання вибору засобів та мови розробки ПЗ.

Для розробки даного програмного засобу була вибрана Java. Ця мова програмування стабільно користується попитом на ринку. Java довела свою актуальність та активно супруводжується, розививається, що дозволить додатку довго залишатись незастарілим. Завдяки простоті та надійності Java використовують у різних сферах життя: для розробки ПЗ в державній сфері, в науці, освіті, сфері охорони здоров’я, в приватному секторі при створенні програм для трейдингу, серверні додатки для банкінгу та для багатьох інших корпоративних і ентерпрайз цілей.

Java хороша майже для усього. Тому в майбутньому, якщо підприємство буде розширювати спектр застосування ПЗ на інші сфери своєї діяльності, її легко буде адоптувати та супруводжувати. Навіть якщо сьогодні винайдуть достойну альтернативу, повний відхід від Java буде тривати дуже довго. Тому немає причин вважати, що найближчим часом ця мова програмування стане неактуальною.

3.2.2 Розробка додатку

Для початку розробки додатку, спершу необхідно детально та по кроках описати алгоритми, побудувати схему їх роботи та представити у вигляді программного коду.

Алгоритм Дейкстри: нехай дано орієнтований зважений граф G = (V, E), де V – множина вершин, E – множина ваг ребер. Вага ребер графа невід'ємна і визначається ваговою функцією w: E> R. Алгоритм Дейкстри знаходить довжини найкоротших шляхів із заданої вершини s до всіх інших. Графічна інтерпретація алгоритму представлена на рисунку у вигляді блок-схеми (рис.3.2), де i, j, k, - аргументи проходу по циклу, N - розмір масиву відстаней, D[N][N] - масив відстаней. Неформальний опис алгоритму представлено далі.

Кожній вершині з V зіставимо мітку - мінімальне відоме відстань від цієї вершини до заданої, позначимо її a. Алгоритм працює покроково - на кожному кроці він «відвідує» одну вершину і намагається зменшувати мітки. Робота алгоритму завершується, коли всі вершини відвідані.

Мітка самої вершини a покладається рівною 0, мітки інших вершин - нескінченності. Це відображає те, що відстані від a до інших вершин поки невідомі. Всі вершини графа позначаються що не відвідані.

Алгоритм Флойда: робить N ітерацій, після i-й ітерації матриця A буде містити довжини найкоротших шляхів між будь-якими двома парами вершин за умови, що ці шляхи проходять через вершини від першої до i-й. На кожній ітерації перебираються всі пари вершин і шлях між ними скорочується за допомогою i-й вершини. Перед роботою алгоритму матриця A заповнюється довжинами ребер графа. Для заданого зваженого графа G = (V, E) алгоритм знаходить найкоротші

шляхи з заданої вершини до всіх інших вершин. В, разі, коли в графі містяться негативні цикли, досяжні з, алгоритм повідомляє, що найкоротших шляхів не існує.

На рисунку 3.1 представлена розроблена блок-схема алгоритму Флойда, де показано, яким чином, працює цей алгоритм. i, j, k, - аргументи проходу по циклу, N - розмір масиву відстаней, D [N] [N] - масив відстаней.

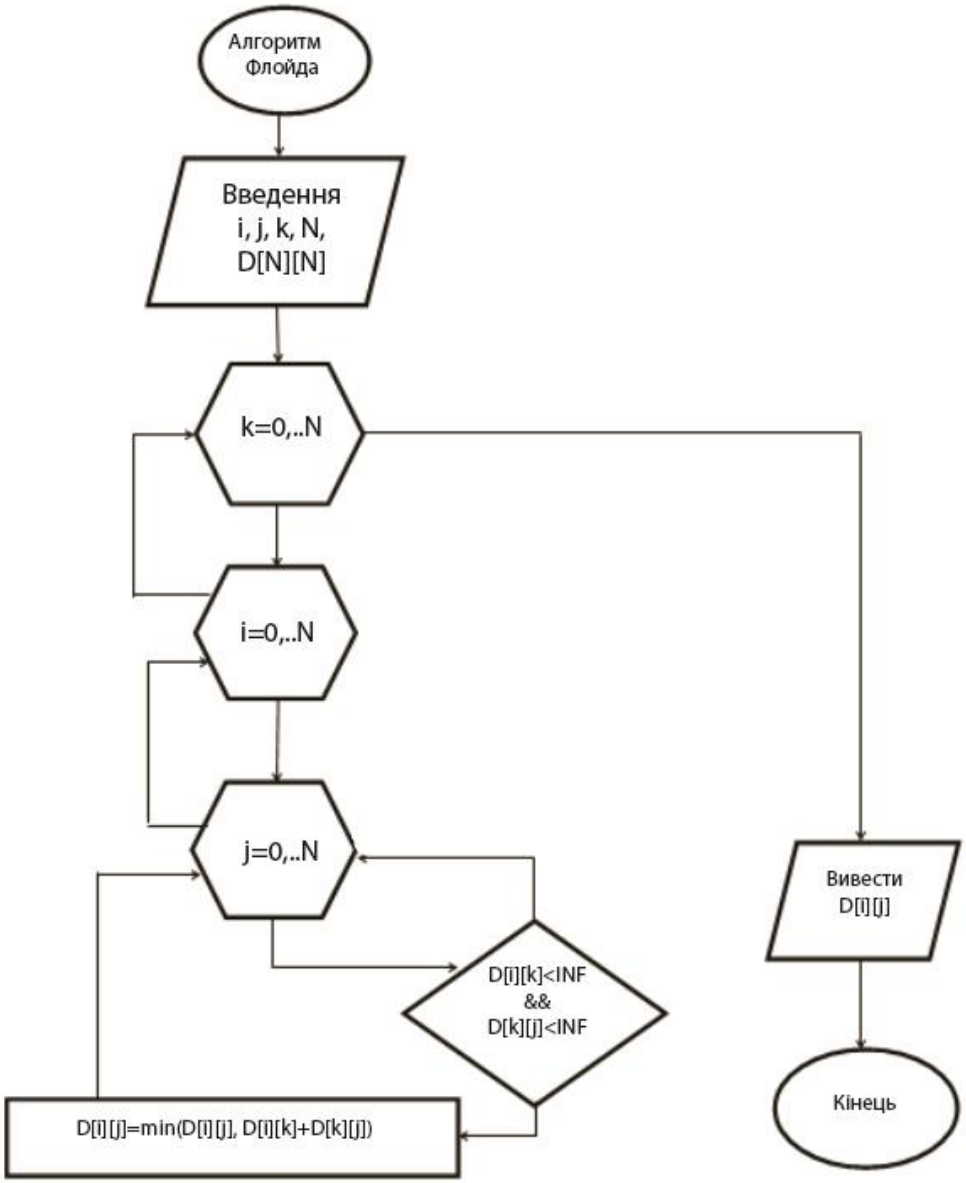


Рис. 3.1 - Графічна інтерпретація алгоритму Флойда

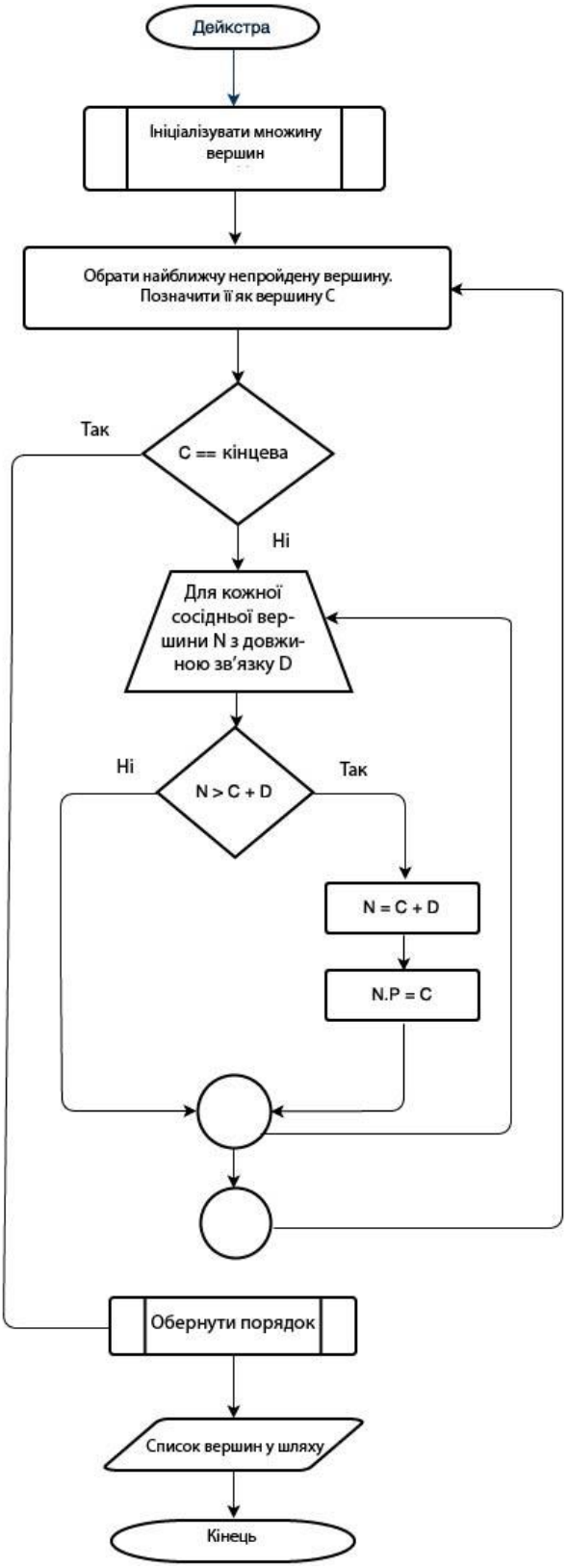


Рис. 3.2 - Графічна інтерпретація алгоритму Дейкстри

Алгоритм A*:

Алгоритм пошуку A * полягає в тому, що на кожному кроці він вибирає вузол відповідно до значення "f", яке є параметром, рівним сумі двох інших параметрів - "g" і "h". На кожному кроці він вибирає вузол / вершину, яка має найнижчу 'f', і обробляє цей вузол / вершину.

Нижче визначення параметрів "g" і "h" :

g - вага руху для переміщення від початкової точки до заданого квадрата в сітці, слідуючи шляхом, що веде до кінцевої вершини призначення

h - розрахункова вага руху для переміщення від даного квадрата на сітці до кінцевого пункту призначення.

Нижче представлена схема алгоритму A*.

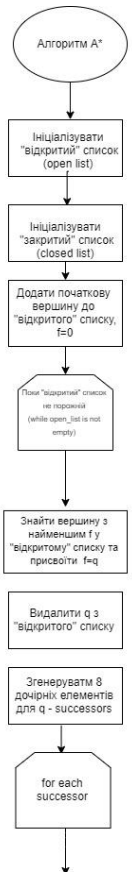


Рис. 3.3 Схема алгоритму A*(a)

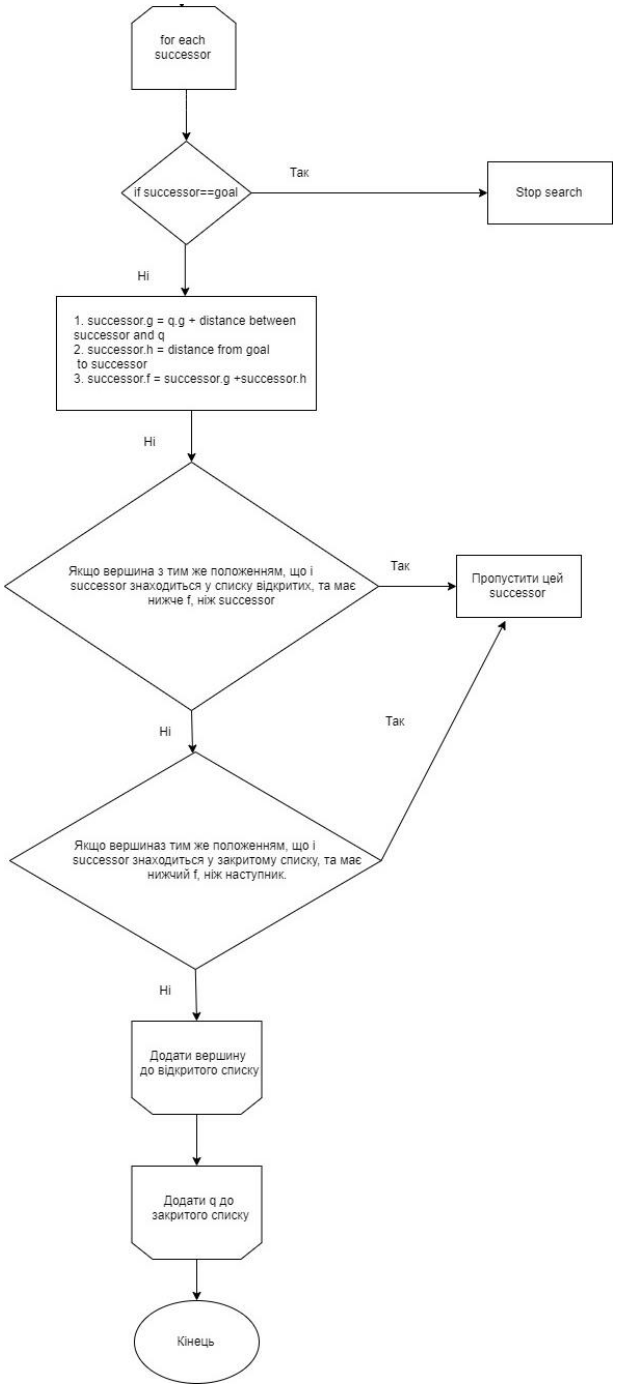


Рис. 3.4 Схема алгоритму A*(б)

3.3 Проведення обчислювальних експериментів

Після опрацювання алгоритмів та написання програмного коду було написано програмний додаток, що тестує ефективність алгоритмів за критерієм часу та кількості проведених операцій. знак

Для проведення екперименту припустимо, що задано деякі локації на місцевості, які мають певні координати. Для більш зручного тестування алгоритмів, кожна локація буде умовно позначена номером, а в залежності від того, чи є шлях між локаціями, вершини будуть з’єднано ребрами. Тобто таку задачу можна представити у вигляді графу. Найчастіше в процесі перевезень виникає необхідність доставити вантаж від 3 до 10 локацій, тому тестування буде проводитись у цих межах. В якості вагів для ребер графа буде представлено час, за який можена дістатись до необхідної локації.

Тестування алгоритмів представлено у наглядному вигляді графу та матриці результатів. Спочатку обираэться кількість вершин, потім ваги для кожного ребра вершини, а потім відображається найкротший шлях у графі

Перший підлягатиме тестуванню підлягатиме алгоритм Флойда. Для програмного додатку було розроблено зручний інтерфейс з наглядною демонстрацією.

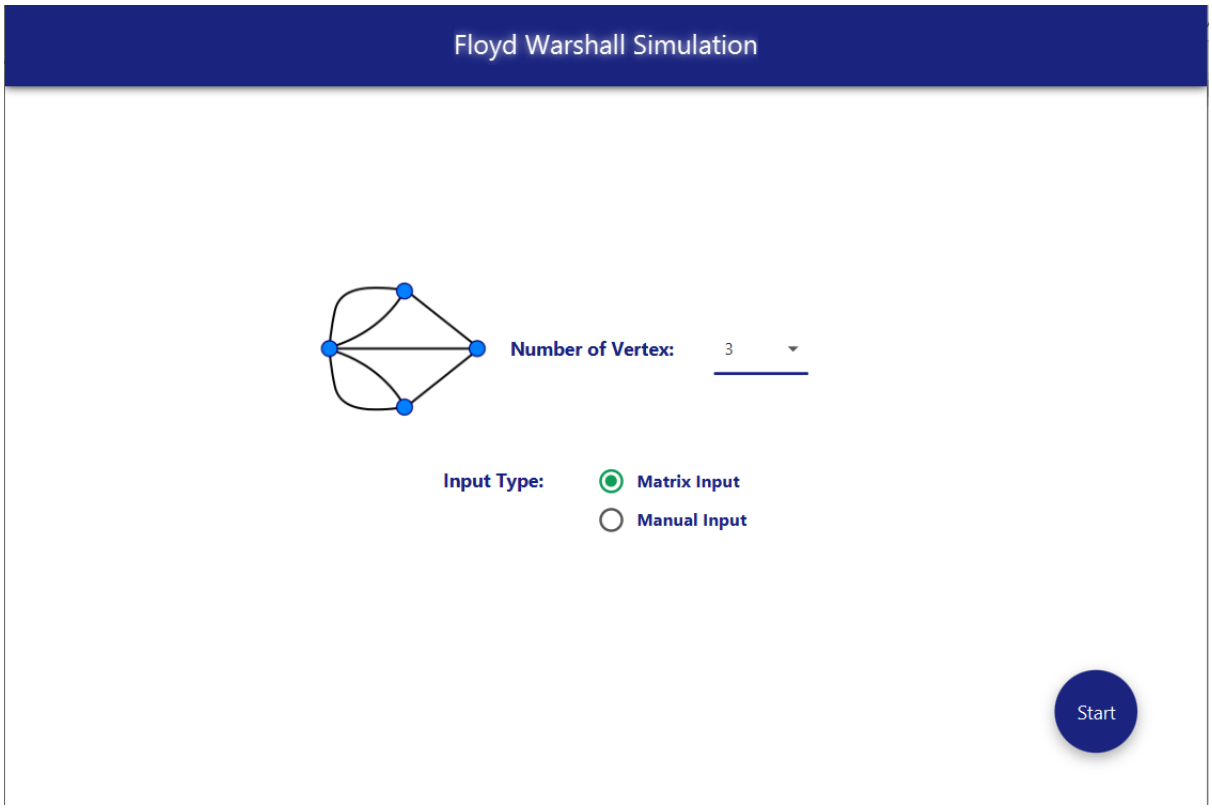


Рис. 3.5 – Задання кількості вершин

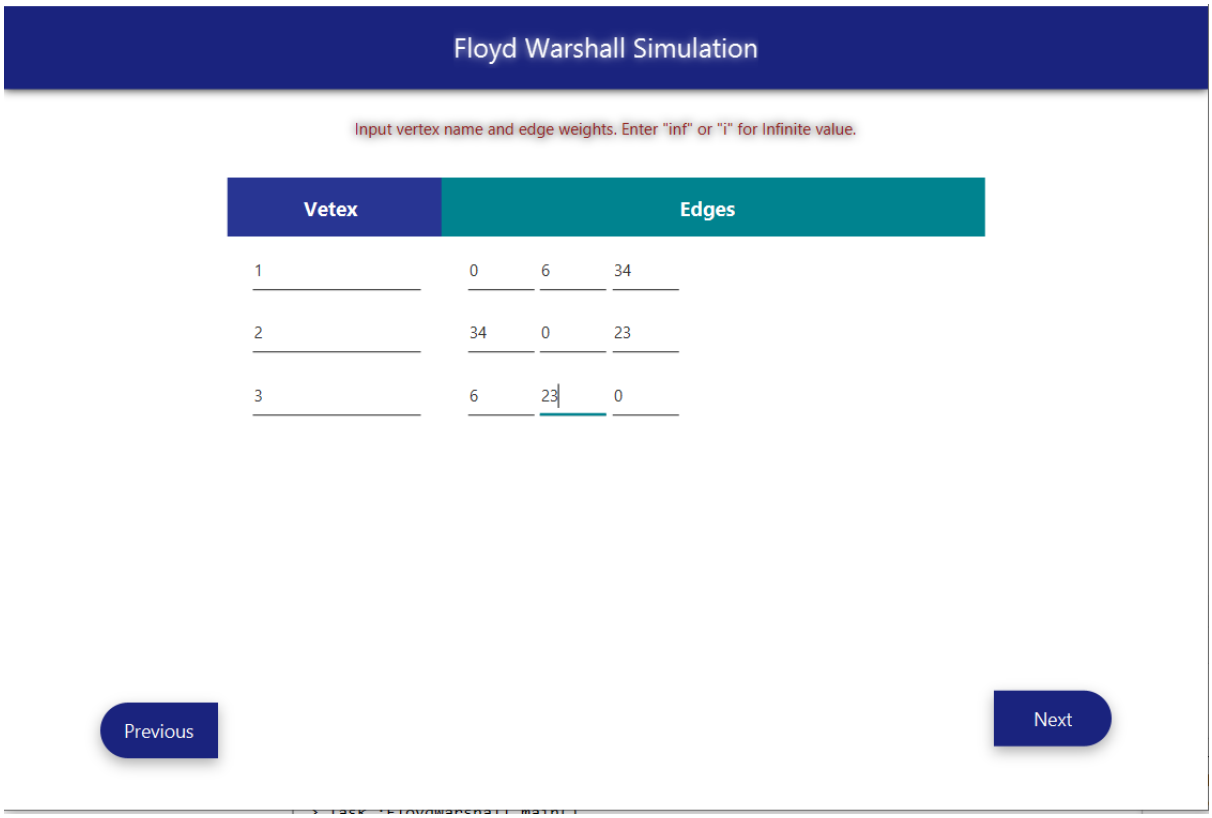


Рис. 3.5 – Задання ваг шляхів для кожної вершини

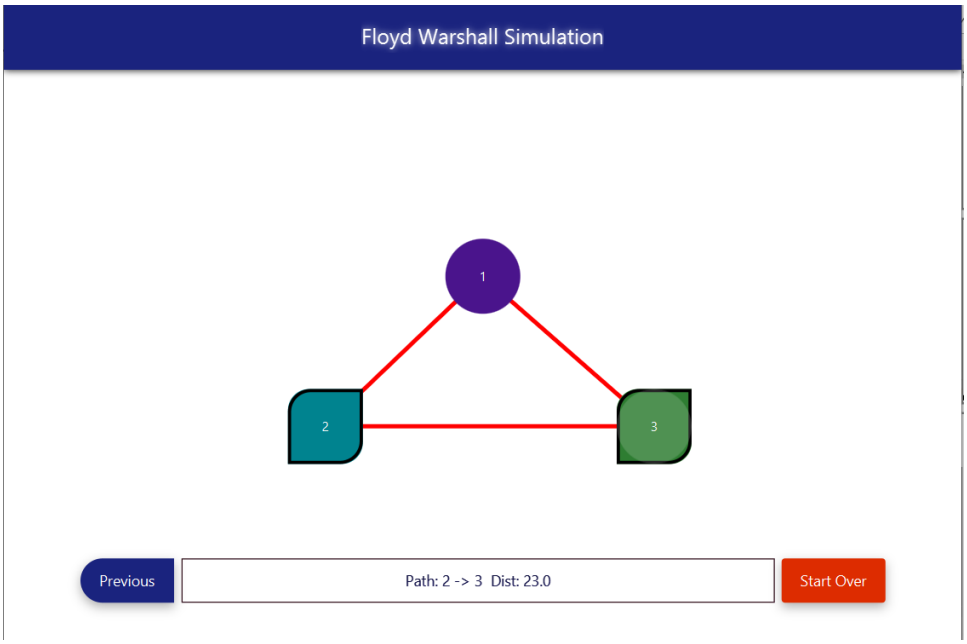


Рис. 3.6 – Результат розрахунку найкоротшого шляху

```
Calculations count27

BUILD SUCCESSFUL in 3m 29s
```

Рис. 3.7 – Кількість виконаних операцій

Спочатку було проведено розрахунок найкоротшого шляху для трех вершин і для цього алгоритму необхідно було зробити 27 операцій.

Далі експеримент з 5 вершинами.

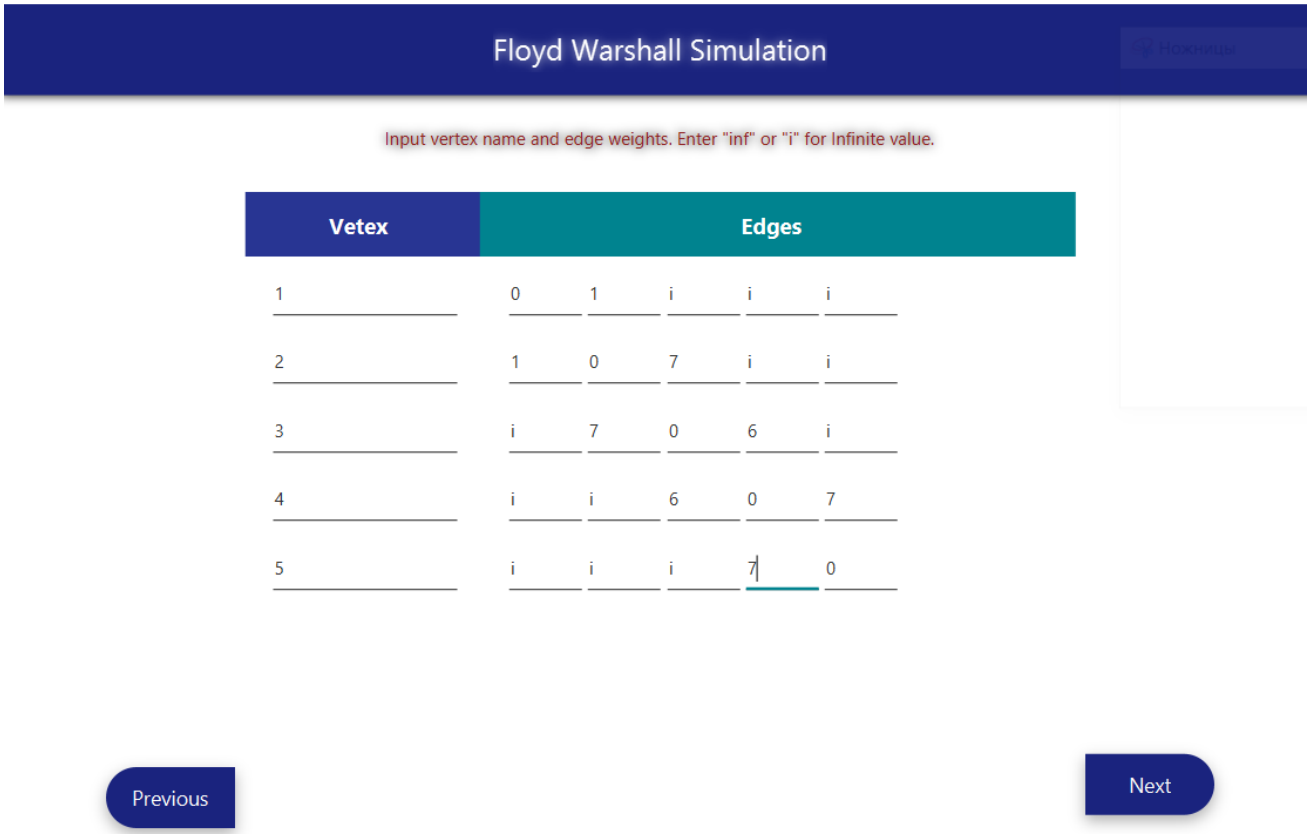
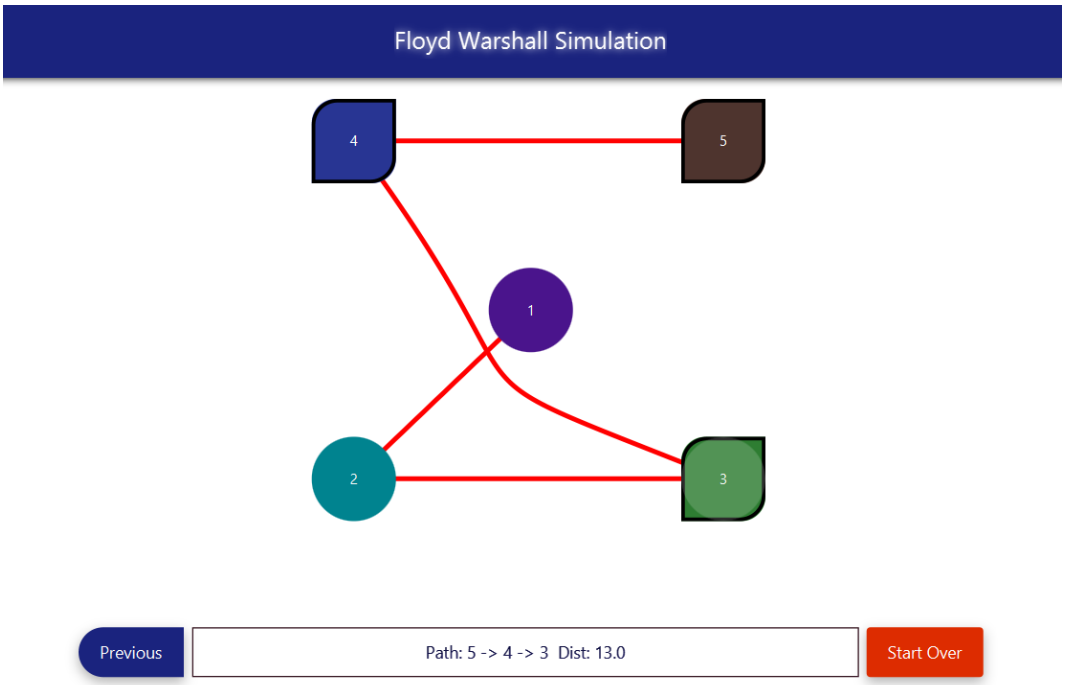


Рис. 3.8 – Задання кількості вершин



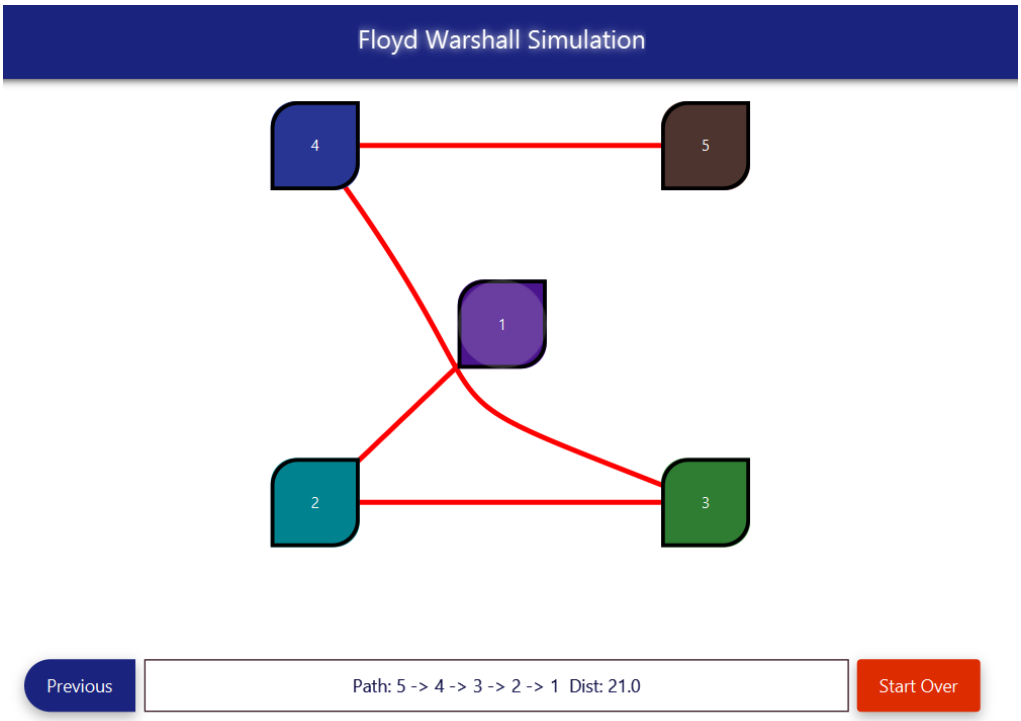


Рис. 3.9 –Результат роботи алгоритму

Як можна побачити на попередніх рисунках, ми можемо обрати будь-які дві вершини і алгоритм покаже найкоротший шлях без перезапуску, тому що він одразу прораховує накоротші шляхи між усіма вершинами.

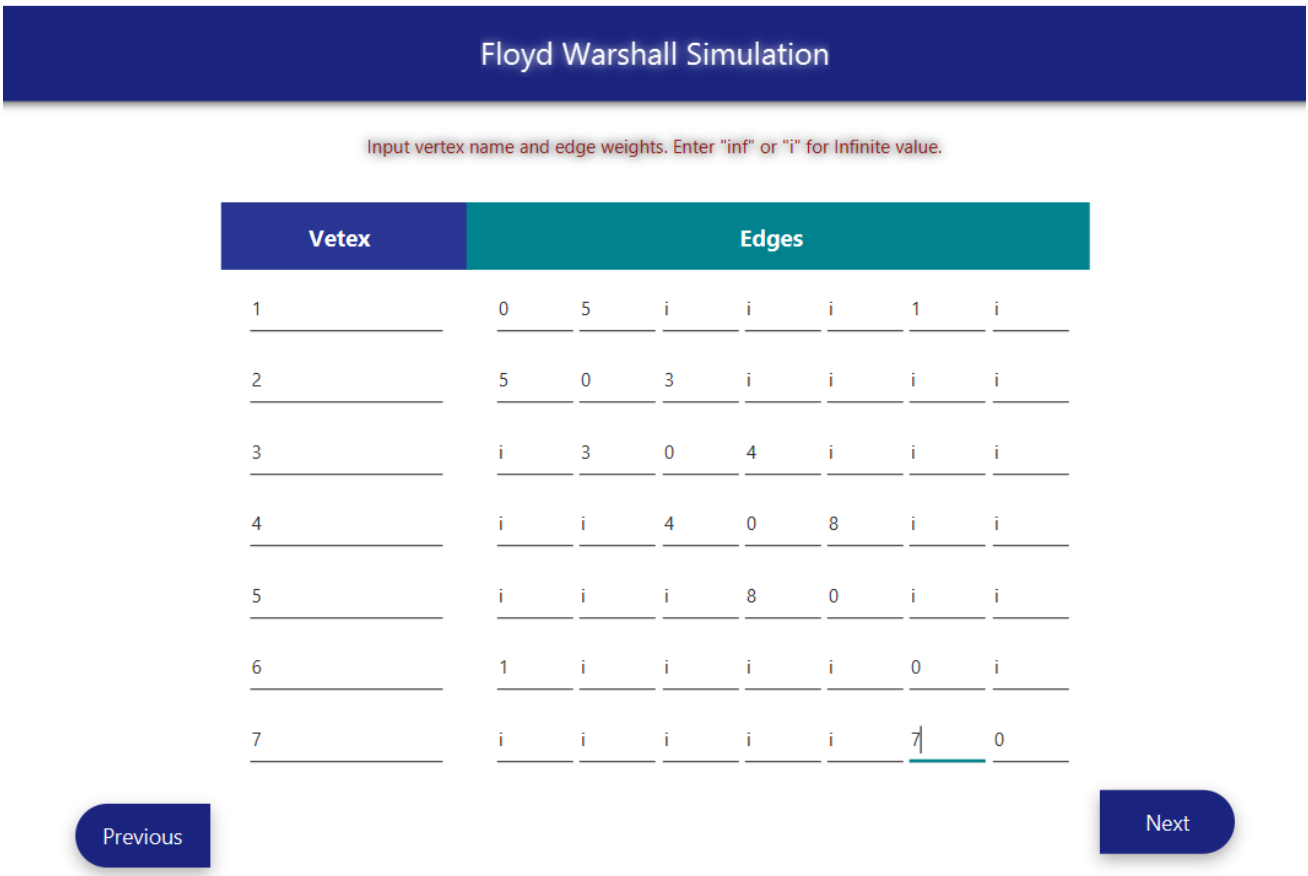


Рис. 3.10 – Задання вагів для 7 вершин

```
FloydWarshall initializaation step: 49
Calculations count343
```

Рис. 3.11 –Кількість операцій для 7 вершин

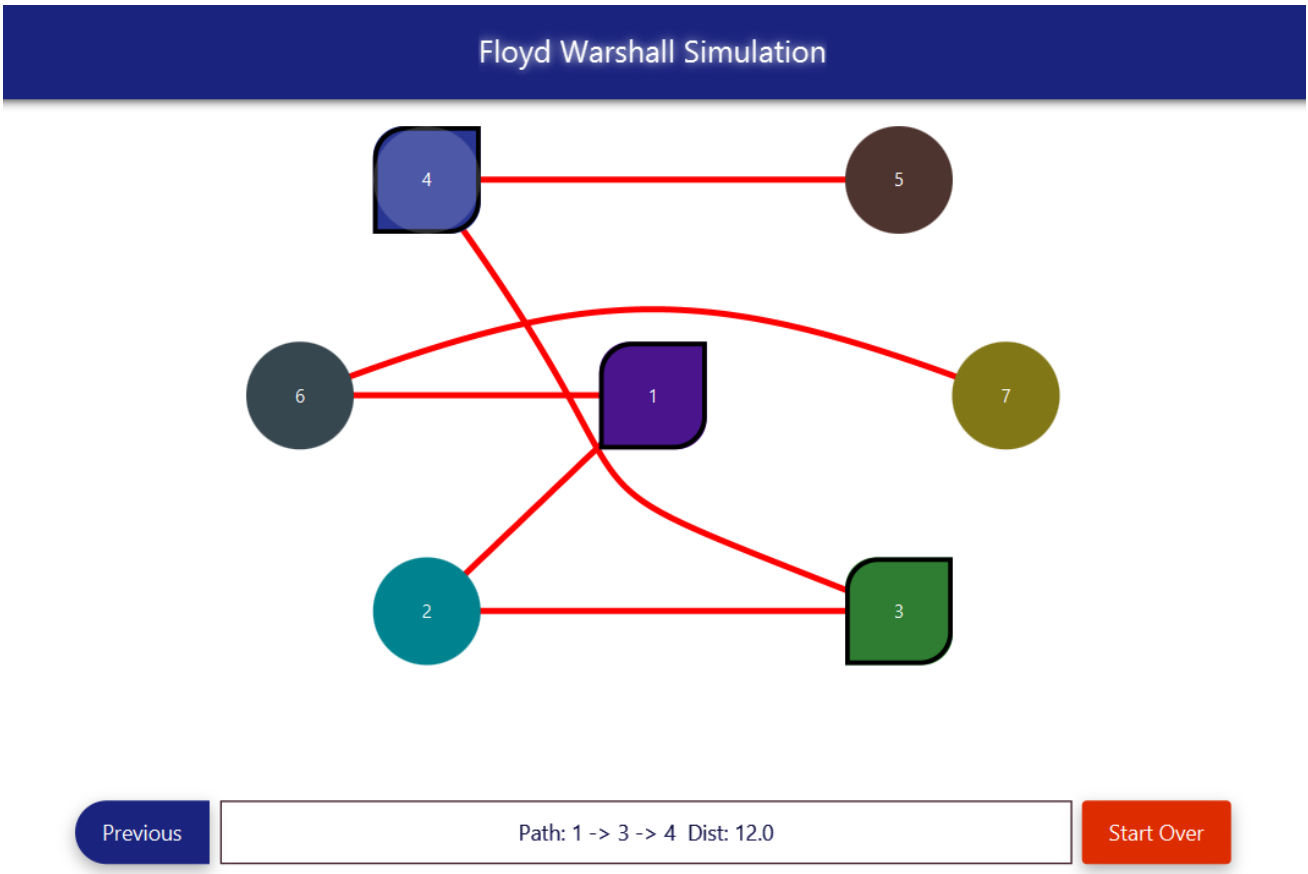


Рис. 3.12 – Результат виконання алгоритму для 7 вершин

Як видно з рис.3.11 для опрацювання 7 вершин алгоритму було необхідно виконати 343 оперції. Для порівняння, алгоритм Дейкстри обробляє таку ж задачу на 7 вершин за 32 операції

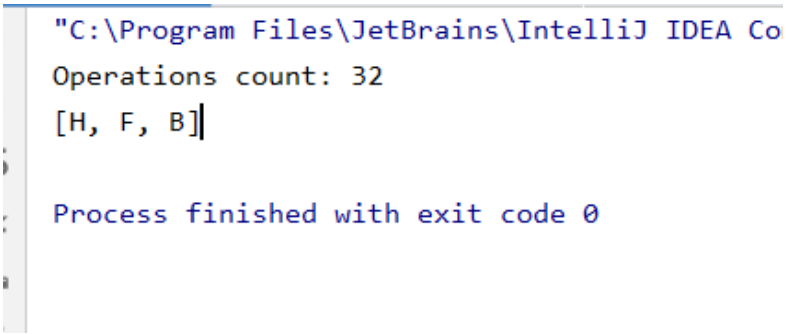


Рис. 3.13 – Результат роботи алгоритму Дейкстри

Оскільки алгоритм Дейкстри проходить пошук для одного шляху за один раз, та одразу перед наступним кроком «обирає» оптимальний наступний крок, що значно скорочує, кількість оперцій.

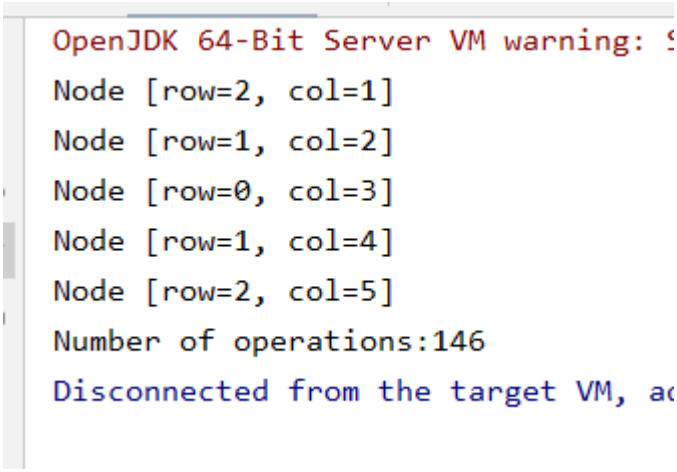


Рис. 3.14 – Результат роботи алгоритму A*

Алгоритм A* схожий за своєю роботою на алгоритм Дейкстри, але при виборі оптимального наступного кроку- вершини він враховує евристичну функцію, яка прогнозує, яка буде відстань від поточної вершини до цільової. Як видно з експерименту, алгоритму знадобилось 146 операцій.

В ході проведення обчислювальних експериментів було виявлено, що найбільш ефективним для вирішення даного типу задач є алгоритм Дейкстри, оскільки здебільшого транспортні системи прцюють у ситуації з великою кількістю пунктів, які необхідно проїхати , та які мають при цьому великі відстані. Як було виявлено у ході експериментів, складність алгоритму Дейкстри при даних умовах складатиме порядка m * log(n), у порівнянні з алгоритмом Флойда, складність якого становитиме порядка n^3

На основі цього була розроблена початкова версія мобільного додатку для розрахунку оптимального шляху на основі алгоритма Дейкстри.

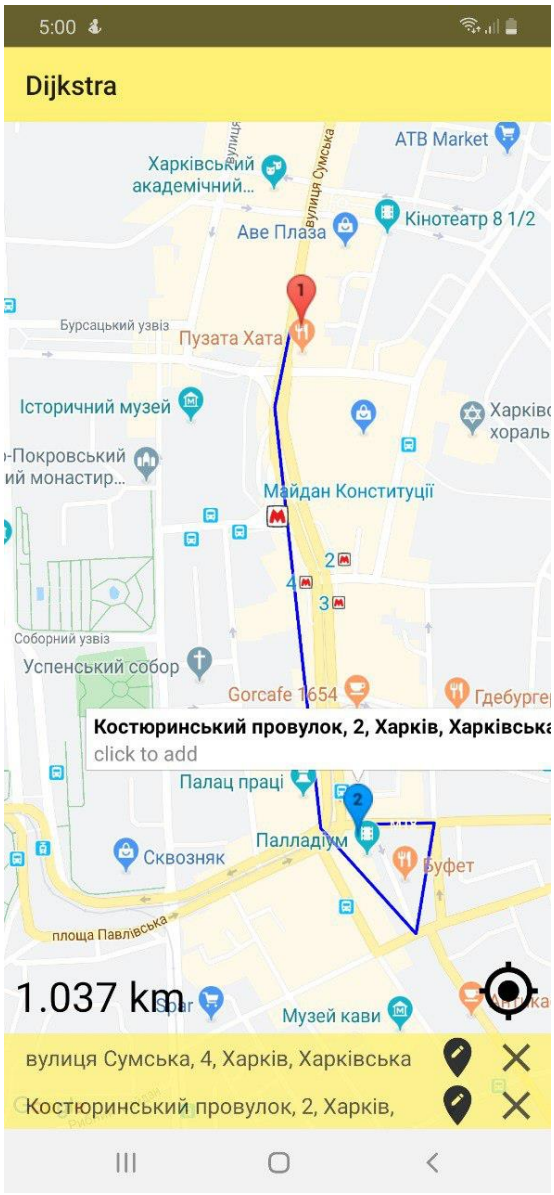


Рис. 3.14 – Результат роботи мобільного додатку

ВИСНОВКИ

В ході дослідницької роботи була поставлена задача розробки підсистеми планування перевезень комп’ютеризованої системи управління транспортного підприємства.

Дослідження полягало у виборі методів оптимізації планування перевезень. Оскільки саме планування перевезень на даний момент є досить актуальним для транспортних підприємств та вони стикаються з багатьма проблемами, дослідження було проведено саме в цьому напрямі.

Перш за все, було розглянуто проблеми, з якими стикаються транспортні компанії, та виконано ряд дій для висунення певних пропозицій, щодо оптимізації роботи частини системи управління транспортним підприємством:

- Проаналізовано основи управління логістичною діяльністю в сфері вантажоперевезень як в межах однієї країни, так і для зарубіжних перевезень

- Обрано математичну модель, яка описує даний процес з метою підбору оптимального алгоритму для побудови маршруту

- Проведено аналіз математичних методів побудови маршрутів перевезень;

- Розглянуто критерії оцінки ефективності алгоритмів побудови маршрутів;

- Проведено дослідження алгоритмів з метою оцінки їх ефективності в залежності від складності маршруту;

- Шляхом проведення обчислювальних експериментів, порівнювались та тестувались деякі методи, з ціллю виділення найбільш оптимального методу побудови маршруту

- Розроблено програмний продукт, який допомагає вирішити задачу оптимізації та автоматизації транспортних перевезень підприємства для підвищення ефективності планування маршрутів; а також виконує накопичення і систематизація всієї інформації в базі даних

Розроблено початкову версію системи контролю транспорту, яка здійснює контроль за транспортом по такими параметрами:

- поточне місце розташування (міжнародні перевезення);

- пройдений маршрут по заданих контрольних точках;

- швидкість руху;

Система контролю руху транспорту також дозволяє:

- відобразити маршрути підзвітних об'єктів за будь-який період часу;

- відобразити на карті положення транспортних засобів в поточний момент часу (on-line контроль транспорту);

- зберігати всю інформацію в локальній базі даних системи контролю руху транспорту, що дозволяє не мати постійне підключення до Internet;

- складати шляхові листи в звичній формі і зберігати їх в базі даних; складати звіти про відвідування об'єктів і автоматично зіставити їх з дорожніми листами;

- складати табличні і графічні звіти по витраті палива, пробігу, швидкості, часу в дорозі і т.д. за будь-який період по кожному транспортному засобу або водієві.

Перед початком обчислювальних експериментів предмет дослідження був описаний у вигляді математичної моделі – графа, вершинами якого можна описати усі пункти, та початкову і кінцеві точки шляху, між якими і необхідно прокласти оптимальний маршрут.

В ході проведення обчислювальних експериментів було виявлено, що найбільш ефективним для вирішення даного типу задач є алгоритм Дейкстри, оскільки здебільшого транспортні системи прцюють у ситуації з великою кількістю пунктів, які необхідно проїхати , та які мають при цьому великі відстані. Як було виявлено у ході експериментів, складність алгоритму Дейкстри при даних умовах складатиме порядка $m * \log(n)$, у порівнянні з алгоритмом Флойда, складність якого становитиме порядка n^3

ПЕРЕЛІК ПОСИЛАНЬ

1 ДСТУ 2226-93 Автоматизовані системи. Терміни та визначення

2. А.А. Бакаєв. «Єкономіко – математичні моделі планування та проектування транспортних систем», Киев, 1973

3. А.А. Силкин, «Грузовые и пассажирские автомобильные перевозки», Москва,1985

4. Оптимизация транспортных перевозок [Електронний ресурс]. 21.03.2018. — Режим доступу: http://provodim24.ru/optimizacija-transportnyh-perevozok.html

5. Методы оптимизации перевозок в транспортных сетях. [Електронний ресурс]. 05.10.2015. — Режим доступу: http://jrnl.nau.edu.ua/index.php/PIU/article/viewFile/11589/15390

6. В.М. Беляев «ОРГАНИЗАЦИЯ АВТОМОБИЛЬНЫХ ПЕРЕВОЗОК И БЕЗОПАСНОСТЬ ДВИЖЕНИЯ», Москва, 2014

7. Бєлов В. В., Воробйов Є. М., Шаталов В. Є. Теорія графів — М .: Вища. школа, 1976. — С. 392.

8. Берж К. Глава 7. Задача о кратчайшем пути // Теория графов и её применения = Theorie des graphes et ses applications / Под ред. И. А. Вайнштейна. — Москва: Издательство иностранной литературы, 1962. — С. 75—81. — 320 с.

9. Берцун В.Н. Математическое моделирование на графах. Часть 2 - Томск: Изд-во Томского ун-та, 2013. — 86 с. — ISBN 978−5−7511−2211−9

10. Методичні вказівки до організації виконання та захисту атестаційної роботи на здобуття другого (магістерського) рівня вищої освіти для студентів усіх форм навчання спеціальності 122 – «Комп’ютерні науки»

за освітньою програмою «Інформаційні технології проектування»