МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

КУДРЯВЦЕВ ВЛАДИСЛАВ СЕРГІЙОВИЧ

Допускається до захисту:
в.о. завідувача кафедри
інформаційних технологій
канд. техн. наук, доцент
_____ О. В. Зелінська
«____»_____ 20__ р.

ПРОКЛАДАННЯ ЛОГІСТИЧНИХ МАРШРУТІВ У ЗОНІ ВЕДЕННЯ БОЙОВИХ ДІЙ З ДОПОМОГОЮ АЛГОРИТМУ ДЕЙКСТРИ

Спеціальність 122 Комп’ютерні науки

Кваліфікаційна (магістерська) робота

Науковий керівник:
П. К. Ніколюк, професор кафедри
інформаційних технологій,
канд. техн. наук, професор

_____
(Підпис)

Оцінка: _____ / _____ / _____
(бали/за шкалою ЄКТС/за національною шкалою)

Голова ЕК: _____
(Підпис)

Вінниця — 2024

ЗМІСТ

ВСТУП

Було пройдено практичну підготовку на підприємстві «IT-СЕРВІС-ВІН». Це компанія, що спеціалізується на наданні широкого спектру інформаційно-технологічних послуг, включаючи розробку програмного забезпечення, IT-консалтинг та технічну підтримку. В умовах сучасного цифрового середовища актуальність задач, які вирішує «IT-СЕРВІС-ВІН», надзвичайно висока, оскільки вони забезпечують функціонування та розвиток IT-інфраструктури багатьох підприємств. Під час практики на мене покладалося завдання аналізу даних та розробки програмних рішень, що сприяють оптимізації бізнес-процесів клієнтів компанії.

ТОВ «IT-СЕРВІС-ВІН» займається такими видами діяльності:

- Розробка та впровадження програмного забезпечення.
- IT-консалтинг та аудит інформаційних систем.
- Технічна підтримка та обслуговування IT-інфраструктури.
- Кібербезпека та захист інформації.

Необхідність використання аналізу даних та інформаційних технологій у роботі підприємства зумовлена складністю та масштабністю завдань, які вирішуються. Сучасні IT-рішення дозволяють автоматизувати процеси збору, обробки та аналізу даних, що забезпечує високу оперативність та ефективність в наданні послуг.

Метою проходження виробничої практичної підготовки на базі ТОВ «IT-СЕРВІС-ВІН» є набуття практичних навичок у сфері інформаційних технологій, удосконалення вмінь з аналізу даних та розробки програмного забезпечення, а також ознайомлення з сучасними методами IT-консалтингу та технічної підтримки.

Основними задачами, які вирішувалися під час практики, були:

- Аналіз великих обсягів даних для покращення якості аналітичних звітів.

- Розробка та тестування програмних рішень для автоматизації бізнес-процесів.
- Участь у проектах з IT-консалтингу, що включає аудит та оптимізацію інформаційних систем клієнтів.
- Забезпечення технічної підтримки та вирішення проблем, що виникають у клієнтів.

Для вирішення поставлених завдань використовувалися такі підходи та методи:

- Методи аналізу даних, включаючи статистичні аналізи та моделі машинного навчання.
- Технології програмування, такі як Python, Java та SQL для розробки та управління базами даних.
- Використання систем управління проектами (наприклад, Jira, Trello) для координації командної роботи.
- Інструменти кібербезпеки для забезпечення захисту інформації та безпеки мережі.

Результати виконаних завдань використовувалися в процесах оптимізації внутрішніх бізнес-процесів клієнтів та підвищення ефективності їх діяльності. Зокрема, аналіз даних допомагав у прийнятті обґрунтованих управлінських рішень, розроблені програмні рішення автоматизували рутинні процеси, а наданий IT-консалтинг та технічна підтримка забезпечували стабільність та безперебійність роботи інформаційних систем клієнтів. Ці результати відіграють ключову роль в загальній системі прийняття рішень, сприяючи підвищенню технологічного розвитку клієнтів.

РОЗДІЛ 1

1.1 Особливості та вимоги до логістичних систем у воєнних конфліктах

У сучасних умовах воєнних конфліктів, особливо в зонах активних бойових дій, ефективне управління логістичними маршрутами є критично важливим завданням. В умовах військових дій важливо не лише забезпечити постачання військових підрозділів необхідними ресурсами, але й зробити це максимально швидко та безпечно.

Наведено теоретичну основу та методичні рекомендації щодо використання алгоритму Дейкстри у контексті логістичного управління у зоні ведення бойових дій. Це включає в себе наступні аспекти:

- Аналіз сучасних тенденцій у логістиці військових операцій, зокрема в контексті управління логістичними маршрутами у зоні бойових дій.
- Вивчення особливостей та вимог до логістичних систем у воєнних конфліктах, зокрема з урахуванням небезпеки та обмежень, які виникають у зоні бойових дій.
- Розробка математичних моделей логістичних маршрутів та їх оптимізація з використанням алгоритму Дейкстри.
- Визначення критеріїв ефективності та методів оцінки розроблених логістичних маршрутів.
- Вивчення можливостей інтеграції алгоритму Дейкстри з іншими методами та технологіями у сфері логістики.

Загальна мета цього дослідження полягає у розробці та обґрунтуванні практичних рекомендацій щодо використання алгоритму Дейкстри для прокладання ефективних логістичних маршрутів у зоні ведення бойових дій. Результати даного дослідження можуть бути використані військовими командуваннями та логістичними підрозділами для покращення управління логістичними процесами у воєнних умовах.

У сучасних умовах воєнних конфліктів, особливо в зонах активних бойових дій, ефективне управління логістичними маршрутами є критично важливим завданням. В умовах військових дій важливо не лише забезпечити постачання військових підрозділів необхідними ресурсами, але й зробити це максимально швидко та безпечно. Наведено теоретичну основу та методичні рекомендації щодо використання алгоритму Дейкстри у контексті логістичного управління у зоні ведення бойових дій. Це включає в себе наступні аспекти:

- Аналіз сучасних тенденцій у логістиці військових операцій, зокрема в контексті управління логістичними маршрутами у зоні бойових дій.
- Вивчення особливостей та вимог до логістичних систем у воєнних конфліктах, зокрема з урахуванням небезпеки та обмежень, які виникають у зоні бойових дій.
- Розробка математичних моделей логістичних маршрутів та їх оптимізація з використанням алгоритму Дейкстри.
- Визначення критеріїв ефективності та методів оцінки розроблених логістичних маршрутів.
- Вивчення можливостей інтеграції алгоритму Дейкстри з іншими методами та технологіями у сфері логістики.

Загальна мета цього дослідження полягає у розробці та обґрунтуванні практичних рекомендацій щодо використання алгоритму Дейкстри для прокладання ефективних логістичних маршрутів у зоні ведення бойових дій. Результати даного дослідження можуть бути використані військовими командуваннями та логістичними підрозділами для покращення управління логістичними процесами у воєнних умовах.

Аналіз історичного досвіду військової логістики

Історія військової логістики є багатим джерелом досвіду, що свідчить про важливість правильно організованих систем постачання для успішного

проведення бойових операцій. З кожним століттям, в міру розвитку технологій та змін в методах ведення бойових дій, зростала й складність логістичних операцій у військових кампаніях. Аналізуючи історичний досвід, можна виокремити кілька ключових етапів розвитку військової логістики та застосування нових технологій для вдосконалення процесів постачання і транспортування.

1. Античність та середньовіччя

У перші етапи розвитку військової логістики основним завданням було забезпечення військ їжею, водою, зброєю та амуніцією, що забезпечувало боєздатність армій. Наприклад, у Давньому Римі існувала добре організована система складів і постачання, що дозволяла Римській імперії вести масштабні військові кампанії. Логістика була критично важливою навіть тоді, коли мова йшла про підтримку мобільності армій в умовах великих відстаней. Власне, логістичні шляхи, такі як римські дороги, були спроектовані для того, щоб швидко доставляти війська та постачання на великі відстані.

У середньовіччі, з розвитком феодальної системи та появою великих армій, постачання ставало дедалі важчим завданням. Важливою частиною логістики було забезпечення військових караванів, які переміщалися через незаселені або ворожі території. Також у цей період з'являється концепція "експедиційної армії", де логістика становить ключову складову для організації довготривалих воєн.

2. Наполеонівські війни

Війни епохи Наполеона стали переломним моментом у розвитку військової логістики. Наполеон Бонапарт розумів, що забезпечення військ постачаннями на великих відстанях є критичним фактором для перемоги. Наполеон активно застосовував концепцію мобільних складів, використовуючи систему централізованого постачання і забезпечуючи регулярні поставки через річки і залізничні шляхи. Однак відсутність належної системи транспорту та розширення фронтів війни призводили до труднощів з

постачанням, особливо на сході Європи, де армії зазнали великих втрат через виснаження та недостатнє забезпечення.

3. Перша та Друга світові війни

У Першій світовій війні зростала роль залізниць, а також автомобільного транспорту для постачання військ. Величезні фронти вимагали комплексного підходу до логістики, яка вже стала не просто постачанням їжі та боєприпасів, а також медичних засобів, техніки і боєзапасів. Логістичні проблеми, що виникали через постійні атаки, погіршення інфраструктури та зміни у стратегічній ситуації, стали важливою частиною військових планувань.

У Другій світовій війні розвиток авіації та автомобільного транспорту ще більше змінив логістичні підходи. Найбільший акцент був зроблений на забезпечення мобільності армій через використання вантажних автомобілів і військових транспортних літаків для доставки ресурсів на передову. Додатково, були введені нові методи управління запасами та склади для швидкого поповнення боєприпасами та іншими необхідними ресурсами.

4. Холодна війна і післявоєнні конфлікти

Під час Холодної війни роль логістики набрала важливості не лише у великих військових конфліктах, але й в аспекті підтримки економічної стабільності через організацію поставок стратегічних ресурсів, таких як паливо і продовольство, між країнами. Для США, як основної глобальної військової потуги, були розроблені масштабні логістичні схеми постачання у зонах конфліктів, таких як В'єтнам, Корея і різноманітні військові кампанії в Африці та Близькому Сході.

5. Сучасні військові конфлікти

У сучасних війнах, наприклад, в Іраку або Афганістані, логістика є важливим фактором для забезпечення успішної операції. Логістичні постачання для підтримки військових підрозділів у таких регіонах часто здійснюються через складну мережу транспортних засобів, включаючи авіацію, автомобільний

транспорт і водні шляхи. В умовах сучасних технологій, таких як супутникові системи навігації, автоматизовані склади і інформаційні системи для моніторингу поставок, логістика набула ще більшої складності, однак забезпечує високу ефективність.

Історія військової логістики вказує на постійну еволюцію технологій і стратегій для забезпечення військових операцій. З кожним століттям методи доставки і постачання удосконалюються, а роль логістичних систем стає все більш критичною для досягнення успіху в бойових умовах. Враховуючи величезну важливість ефективного управління логістичними процесами, історичний досвід дозволяє краще розуміти нинішні виклики та сприяє розробці новітніх стратегій для підтримки військових кампаній в умовах сучасних бойових конфліктів.

Застосування сучасних інформаційних технологій

У сучасних умовах воєнних конфліктів, де швидкість і точність виконання логістичних завдань мають вирішальне значення, застосування сучасних інформаційних технологій дозволяє значно підвищити ефективність управління логістичними процесами. Інформаційні технології дають змогу забезпечити своєчасну доставку ресурсів, мінімізувати ризики та оптимізувати використання доступних ресурсів. Зокрема, впровадження таких технологій, як геоінформаційні системи (ГІС), супутникові системи моніторингу, дрони, а також штучний інтелект, дозволяє значно покращити якість прийнятих рішень в умовах обмеженої інформації та високих ризиків.

Системи GPS і супутникові технології дозволяють в реальному часі визначати місцезнаходження транспортних засобів, військових підрозділів та ресурсів. Це дає змогу точно планувати маршрути постачання, знижуючи ймовірність затримок або втрат у результаті зміни бойової обстановки або технічних проблем. Крім того, супутникові системи моніторингу можуть виявляти блокування шляхів або відсутність доступу до певних районів, що дозволяє своєчасно коригувати логістичні маршрути. Важливим аспектом є

можливість використання даних для побудови інтерактивних карт, що постійно оновлюються в залежності від зміни ситуації.

Геоінформаційні системи (ГІС) забезпечують інтеграцію даних про місцевість, інфраструктуру та бойову ситуацію в єдину платформу, що дозволяє ефективно планувати та оптимізувати логістичні маршрути. В умовах воєнного конфлікту, де швидкість реагування на зміни є критично важливою, ГІС допомагають оцінити стан доріг, відстань, можливі перепони на маршруті (наприклад, мінування або блокади), а також наявність ворожих підрозділів на шляху доставки. Використання ГІС у поєднанні з актуальними даними про бойову обстановку дозволяє знизити ймовірність потрапляння транспортних засобів в небезпечні зони.

Автоматизовані системи управління логістикою (АСУЛ) є невід'ємною частиною сучасних військових логістичних процесів. Вони дозволяють здійснювати моніторинг всіх етапів логістичних операцій: від планування маршрутів і постачання до зворотного зв'язку та оцінки ефективності виконаних завдань. АСУЛ здатні автоматично коригувати маршрути у разі зміни бойової обстановки, відсутності необхідних ресурсів чи виникнення нових небезпек, що підвищує загальну ефективність логістичних процесів. Ці системи також можуть обробляти великі обсяги даних, що дозволяє зменшити ймовірність людської помилки при прийнятті рішень.

Дрони виявилися корисними для розвідки, спостереження та доставки вантажів у зонах, де традиційні засоби транспорту можуть бути уражені або заблоковані. Вони здатні виконувати доставку без участі людини, що дозволяє знизити ризик втрат серед особового складу. Крім того, дрони можуть надавати в режимі реального часу інформацію про стан маршруту, наявність перешкод або активність ворога. Автономні транспортні засоби, зокрема вантажівки з безпілотним управлінням, можуть бути використані для доставки вантажів у небезпечні зони, що зменшує ризики для водіїв і підвищує ефективність логістичних операцій.

Штучний інтелект (ШІ) та методи машинного навчання дозволяють автоматизувати процеси прийняття рішень у складних і динамічних умовах бойових дій. ШІ може бути використаний для прогнозування змін у бойовій обстановці та коригування логістичних маршрутів з урахуванням нових факторів, таких як відсутність доріг, пересування ворожих сил або зміни погодних умов. Машинне навчання також може допомогти в аналізі великих обсягів даних про минулі операції, що дозволяє знаходити найбільш ефективні стратегії та рішення для подальших логістичних процесів.

Хмарні платформи забезпечують доступ до необхідних даних у реальному часі з будь-якої точки, що надзвичайно важливо в умовах воєнного конфлікту, коли переміщення підрозділів та ресурсів є постійним. Використання хмарних технологій дозволяє централізовано зберігати дані, що дає змогу оперативно змінювати стратегії та плани постачання, а також знижувати ризики втрати важливої інформації. Вони також забезпечують можливість координації між різними рівнями командування, що дозволяє злагоджено діяти різним підрозділам при виконанні складних завдань.

У сучасному воєнному конфлікті зростає важливість. Логістичні системи є вразливими до атак, які можуть призвести до захисту інформаційних систем від кібератак втрати даних або дезорганізації поставок. Використання передових систем кібербезпеки допомагає захистити інформацію про маршрути, ресурси та стратегії, що в свою чергу забезпечує безпеку логістичних операцій. Програми для моніторингу та аналізу загроз можуть попередити про можливі атаки та забезпечити своєчасний захист систем.

Таким чином, застосування сучасних інформаційних технологій у військовій логістиці дає змогу значно покращити ефективність, безпеку та швидкість логістичних операцій, що є критично важливим в умовах воєнного конфлікту. Технології, такі як супутниковий моніторинг, ГІС, автоматизовані системи управління, дрони, ШІ та хмарні платформи, дозволяють оперативно

реагувати на зміни в бойовій обстановці, оптимізувати ресурси і знижувати ризики для військових підрозділів та цивільного персоналу.

Оцінка ризиків і управління загрозами

У військовій логістиці в умовах воєнного конфлікту оцінка ризиків та управління загрозами є ключовими елементами для забезпечення безпеки та ефективності операцій. Зростаюча роль сучасних інформаційних технологій (ІТ) у цій сфері зумовлює необхідність інтеграції систем управління ризиками, щоб мінімізувати потенційні загрози та забезпечити належне функціонування логістичних процесів. Оцінка ризиків в умовах воєнних дій передбачає врахування різноманітних факторів, які можуть негативно вплинути на результати логістичних операцій, від фізичних загроз до кібер-атак.

Оцінка ризиків у військовій логістиці передбачає кілька категорій загроз:

1. **Фізичні загрози** включають у себе атаки на транспортні маршрути, пошкодження або блокування стратегічних об'єктів, як-от мости чи склади, а також мінування територій і загрози з боку ворога. Сучасні інформаційні технології, такі як геоінформаційні системи (ГІС) і супутникові дані, дають змогу оцінити ризики на конкретних ділянках, знижуючи ймовірність потрапляння транспортних засобів або військових підрозділів у небезпечні зони.
2. **Технічні та операційні загрози** включають технічні проблеми, такі як поломка транспорту або відсутність ресурсів для підтримки ефективності операцій. Інформаційні технології дозволяють моніторити стан транспортних засобів і обладнання в реальному часі, своєчасно повідомляючи про технічні несправності та дозволяючи швидко вжити заходи для їх усунення.
3. **Кіберзагрози**. У умовах активного використання цифрових технологій для управління логістикою, ризики кібератак стають серйозною проблемою. Хакерські атаки можуть спричинити збої в

системах управління, маніпулювання даними або блокування комунікаційних каналів. Для управління цими загрозами застосовуються передові технології кібербезпеки, такі як шифрування даних, системи виявлення вторгнень (IDS) і захист від DDoS-атак, що дозволяє запобігти компрометації важливої інформації.

Оцінка ризиків в рамках військових логістичних операцій повинна базуватись на методах, які враховують як ймовірність виникнення загроз, так і їх потенційний вплив на операцію. Одним із найбільш ефективних підходів є використання моделей аналізу ймовірності, таких як методи Монте-Карло або ймовірностно-статистичні методи. Ці моделі дозволяють оцінити потенційний ризик у випадку різних сценаріїв, що можуть виникнути в ході логістичних операцій. За допомогою ГІС та супутникових даних можна створювати карти ризиків, що дозволяє оперативно коригувати маршрути і приймати обґрунтовані рішення.

Сучасні інформаційні технології, зокрема програмне забезпечення для управління логістичними операціями, здатні автоматизувати процеси оцінки та управління ризиками. Вони дозволяють своєчасно виявляти потенційні загрози та відповідно коригувати стратегії доставки, планування маршрутів і управління ресурсами. Використання систем, що базуються на великих даних, дає можливість здійснювати моніторинг в реальному часі та оперативно реагувати на зміни в ситуації на фронті або на транспортних шляхах. Наприклад, автоматичне коригування маршруту в разі виявлення мін або підвищеної активності ворога на певному шляху є важливим аспектом для забезпечення безпеки постачання.

Прогнозування можливих загроз в умовах воєнного конфлікту дозволяє планувати заходи щодо їх мінімізації ще на етапі підготовки. Системи на основі штучного інтелекту та машинного навчання можуть аналізувати історичні дані та виявляти закономірності, що дозволяє прогнозувати дії

ворога або зміни в бойовій обстановці. Це дає змогу своєчасно адаптувати логістичні стратегії та шляхи доставки, а також скоротити час реакції на виникнення нових загроз.

Для мінімізації ризиків необхідна інтеграція різноманітних технологій, що дає змогу створити комплексну систему управління загрозами. Використання даних з безпілотних літальних апаратів (БПЛА), дронів, а також даних від супутникових систем дозволяє не лише оцінювати реальний стан ситуації, але й забезпечує постійний моніторинг загроз, таких як активність ворожих сил або природні перешкоди. Це в свою чергу дає змогу своєчасно адаптувати логістичні плани та знижувати ймовірність негативних наслідків.

Таким чином, ефективне управління ризиками і загрозами в контексті військової логістики потребує інтеграції сучасних інформаційних технологій, що дозволяють не лише оцінювати ймовірність загроз, але й оперативно реагувати на зміни в бойовій ситуації. Автоматизовані системи, геоінформаційні технології, супутникові моніторинги, а також новітні методи кібербезпеки створюють надійну основу для запобігання критичним загрозам, що може значно підвищити ефективність логістичних операцій та забезпечити безпеку в зоні бойових дій.

1.2 Існуючі механізми та підходи до вирішення поставленої задачі

Існуючі механізми вирішення проблематики логістичних маршрутів у зонах ведення бойових дій включають кілька ключових підходів та технологій. Одним з основних методів є використання оптимізаційних алгоритмів для побудови найкоротших та найбезпечніших маршрутів. Алгоритм Дейкстри, зокрема, є одним з найефективніших і найпоширеніших алгоритмів для вирішення задач пошуку найкоротшого шляху в графах з невід'ємними вагами. Він дозволяє швидко та ефективно знаходити оптимальні маршрути, враховуючи відстань та інші параметри, такі як безпека та доступність.

Ще одним важливим механізмом є використання гео-інформаційних систем (ГІС), які забезпечують візуалізацію та аналіз географічних даних. Військові логістичні системи інтегрують ГІС для планування та моніторингу маршрутів, що дозволяє оперативно реагувати на зміни в бойовій обстановці та знаходити альтернативні шляхи.

Крім того, сучасні логістичні системи часто використовують методи машинного навчання та штучного інтелекту для прогнозування можливих загроз та оцінки ризиків при плануванні маршрутів. Ці технології дозволяють аналізувати великі обсяги даних та виявляти закономірності, що сприяє більш точному та надійному плануванню логістичних операцій.

Також важливим аспектом є співпраця з місцевими органами влади та міжнародними організаціями для забезпечення безпеки маршрутів та координації дій. Це включає обмін інформацією про поточну обстановку, потенційні загрози та наявність безпечних коридорів для транспортування ресурсів.

Система управління запасами є ще одним критичним елементом. Ефективне управління запасами дозволяє знизити ризики, пов'язані з недостатнім постачанням або затримками в доставці необхідних ресурсів.

Існуючі механізми вирішення проблематики також включають використання безпілотних літальних апаратів (БПЛА) для розвідки та доставки невеликих вантажів у важкодоступні або небезпечні зони. Це дозволяє знизити ризики для персоналу та підвищити ефективність доставки критично важливих ресурсів.

Одним із ключових трендів є використання **розподілених систем планування**, що дозволяє координувати логістичні дії між кількома точками управління в реальному часі. Це забезпечує швидку адаптацію до змін на полі бою та підвищує гнучкість у прийнятті рішень. Розподілені системи також допомагають синхронізувати різні логістичні операції, зменшуючи затримки та оптимізуючи потоки постачання.

Технології блокчейн набувають популярності для забезпечення надійного та безпечного обміну даними у логістичних системах. В умовах воєнних конфліктів важливо захистити інформацію про маршрути, вантажі та місця доставки від можливих втручань або кібератак. Блокчейн дозволяє зберігати інформацію у незмінному вигляді та забезпечує прозорість ланцюгів постачання, що критично важливо для збереження цілісності логістичних операцій.

Ще одним важливим підходом є інтеграція **технологій Інтернету речей (IoT)**. Підключені до інтернету пристрої дозволяють у режимі реального часу відстежувати стан транспорту або вантажів, моніторити маршрут і отримувати дані про умови перевезення. Ця технологія забезпечує оперативний контроль над ситуацією, що сприяє зниженню ризиків, пов'язаних з пошкодженням вантажів або втратою зв'язку з транспортними засобами.

Важливу роль у плануванні маршрутів також відіграють **адаптивні алгоритми маршрутизації**, які дозволяють автоматично коригувати маршрути в залежності від змін на місцевості. Наприклад, ці алгоритми можуть змінювати маршрут, коли виявляються нові небезпечні зони, змінюється обстановка на блокпостах або погіршуються погодні умови. Такий підхід підвищує ефективність логістичних операцій, оскільки маршрути завжди залишаються актуальними і безпечними, навіть у мінливих бойових умовах.

Одним із сучасних інструментів є **моделювання сценаріїв та симуляції**. За допомогою симуляцій можна проаналізувати різні сценарії розвитку подій та оцінити їхній вплив на логістичні операції. Це дозволяє не лише підготуватися до різноманітних викликів, але й знизити ймовірність прийняття неправильних рішень в умовах невизначеності. Моделювання також допомагає передбачити можливі перешкоди на маршруті та заздалегідь розробити альтернативні шляхи.

Кібербезпека логістичних систем є критично важливою, оскільки під час бойових дій військові системи піддаються численним кібератакам. Захист логістичних даних та інфраструктури є необхідною умовою для забезпечення безперервності постачання. Використання сучасних методів захисту, таких як шифрування та багатофакторна автентифікація, дозволяє мінімізувати ризики витоку інформації та втрати контролю над логістичними процесами.

Ще одним потужним інструментом у вирішенні проблематики логістичних маршрутів є аналіз **великих даних (Big Data)**. Збір та обробка великих обсягів інформації дозволяє прогнозувати потенційні ризики, аналізувати зміни у бойовій обстановці та оптимізувати логістичні процеси. Великі дані можуть бути використані для виявлення закономірностей у постачанні ресурсів, прогнозування можливих затримок або загроз на маршрутах, що сприяє підвищенню точності і надійності планування.

Таким чином, вирішення задач логістики у зонах бойових дій ґрунтується на інтеграції сучасних технологій та підходів, які забезпечують безпеку, гнучкість та ефективність логістичних операцій. Інновації, такі як блокчейн, IoT, адаптивні алгоритми та Big Data, у поєднанні з традиційними підходами, надають можливість військовим логістичним системам швидко реагувати на зміни і підтримувати високу оперативну готовність у складних умовах війни.

1.3 Моделювання логістичних маршрутів за допомогою напрямлених зважених графів

Граф - це абстрактна математична структура, що складається з множини вершин (вузлів) та множини ребер, які з'єднують ці вершини. Графи використовуються для моделювання та вивчення різноманітних ситуацій та взаємозв'язків між об'єктами.

Існують різні види графів, серед яких основні - це:

- Ненапрямлений граф: це граф, де ребра не мають напрямку. Вони просто з'єднують дві вершини і можуть бути переходом у обидва напрямки.
- Напрямлений граф: у цьому типі графа ребра мають напрямок. Вони вказують на одну вершину, яка є початковою, та на іншу, що є кінцевою.
- Зважений граф: кожне ребро має асоційований з ним вагу або вартість. Це число вказує на відстань або вартість переходу від однієї вершини до іншої.

Для теми прокладання логістичних маршрутів у зоні ведення бойових дій за допомогою алгоритму Дейкстри найбільш доцільним є використання напрямленого зваженого графа. Це обумовлено наступними причинами:

- Напрямленість: у контексті ведення бойових дій важливо враховувати напрямок переміщення військових частин та руху ресурсів. Напрямлені ребра графа дозволяють точно моделювати ці процеси.
- Зваженість ребер: оскільки вартість руху в зоні бойових дій може відрізнятися в залежності від різних факторів, таких як безпека маршруту, відстань, стан доріг тощо, важливо мати можливість враховувати ці ваги при обчисленні найкоротшого шляху. Зважені ребра дозволяють моделювати цей аспект.
- Ефективність алгоритму Дейкстри: алгоритм Дейкстри найбільш ефективний для пошуку найкоротшого шляху в напрямленому зваженому графі з додатними вагами. Враховуючи, що у воєнній сфері негативні ваги (наприклад, від'ємні ваги) можуть бути несенсовними або навіть небезпечними, використання алгоритму Дейкстри в цьому контексті є доцільним.

Отже, напрямлений зважений граф є належним інструментом для моделювання маршрутів у воєнній логістиці, оскільки він дозволяє

враховувати напрямок переміщення військових частин та ресурсів, а також різні ваги, пов'язані з безпекою, відстанню та іншими чинниками.

Напрямлений зважений граф підходить для сценаріїв, де необхідно швидко обчислити оптимальний маршрут доставки важливих ресурсів, таких як боєприпаси, медичне обладнання або харчові запаси, у зону бойових дій. Ваги на ребрах можуть відображати різні фактори, такі як довжина маршруту, рівень безпеки чи доступність доріг.

Застосування алгоритму Дейкстри в такому контексті дозволяє ефективно знаходити найкоротший шлях у напрямленому зваженому графі, що допомагає оптимізувати процес доставки ресурсів і забезпечити їхню швидку та безпечну передачу в зону ведення бойових дій.

Таким чином, використання напрямленого зваженого графа разом із алгоритмом Дейкстри надає ефективний інструмент для моделювання та оптимізації логістичних маршрутів у воєнних умовах.

Історичний контекст використання графів у логістиці

Графи, як математична структура, виникли у XVIII столітті завдяки роботі Леонарда Ейлера, який запропонував їх у своєму дослідженні задачі про Кенігсберзькі мости. У цьому випадку графи були використані для моделювання топології міста, а задача полягала у знаходженні маршруту, який проходив би кожним мостом лише раз. Ця задача вважається однією з перших у теорії графів і започаткувала розвиток цієї галузі.

З плином часу графи почали застосовувати у різноманітних сферах. У XIX столітті вони стали основою для вивчення електричних мереж. Роботи Густава Кірхгофа демонстрували, як графи можуть описувати зв'язки між вузлами в електричних ланцюгах. Подібні моделі дозволили ефективно аналізувати складні мережі проводів, що значно вплинуло на розвиток електротехніки.

На початку XX століття графи почали активно використовуватися у транспортній сфері. Наприклад, залізничні маршрути та дорожні мережі моделювалися у вигляді графів для оптимізації логістики та пошуку найкоротших шляхів між містами. Такі задачі набули важливості у період індустріальної революції, коли швидкість і вартість перевезення ресурсів стали критичними для успіху економік.

У військовій сфері графи відіграли ключову роль під час Другої світової війни. Для планування переміщення військ та ресурсів використовували математичні моделі, що включали елементи теорії графів. Графи дозволяли враховувати складні зв'язки між різними елементами логістики, такими як маршрути постачання, вузли концентрації ресурсів та критичні точки оборони. Їхня ефективність зростала завдяки автоматизації обчислень за допомогою перших обчислювальних машин.

У другій половині XX століття графи стали важливим інструментом для моделювання комунікаційних мереж. Вони описували телефонні мережі, а згодом і комп'ютерні мережі, зокрема Інтернет. Графові структури дозволяли аналізувати стійкість мереж, їхню пропускну здатність і оптимальні маршрути передачі даних.

У сучасному світі графи широко застосовуються у різних галузях. Вони стали основою для таких технологій, як соціальні мережі, де графи описують взаємодії між користувачами, а також у біології, наприклад, для аналізу метаболічних мереж. У логістиці графові моделі залишаються ключовими для оптимізації транспортування ресурсів, включаючи складні задачі військової логістики в умовах бойових дій. Сучасні алгоритми, такі як алгоритм Дейкстри, дозволяють ефективно вирішувати задачі пошуку найкоротшого шляху у напрямлених зважених графах, що робить їх надзвичайно корисними для моделювання реальних процесів.

Графи пройшли довгий шлях від теоретичних задач до інструменту, який впливає на функціонування багатьох ключових аспектів сучасного світу, включаючи логістику, комунікації, економіку та військову стратегію.

Реальні фактори, які впливають на ваги графів у зоні бойових дій

У зоні бойових дій ваги на ребрах графів відображають реальні умови, що впливають на безпечність, швидкість і вартість переміщення ресурсів або військових підрозділів. Ці ваги є ключовими параметрами, які визначають ефективність маршруту і залежать від низки складних і динамічних факторів.

Одним із найважливіших факторів є безпека маршруту. У зоні бойових дій наявність мінних полів, зон обстрілу або контрольованих ворогом територій значно впливає на вибір шляхів. Наприклад, дороги, які проходять через потенційно небезпечні зони, отримують значно більші ваги, адже ризик втрати ресурсів або життя військових є критичним. Інформація про безпеку постійно змінюється залежно від розвитку подій на полі бою, тому ваги мають оновлюватися динамічно.

Другим важливим фактором є стан транспортної інфраструктури. Унаслідок бойових дій дороги можуть бути пошкоджені або зруйновані, що робить їх практично непрохідними для військової техніки або конвоїв. Мости, тунелі, а також ключові дорожні вузли стають стратегічними об’єктами, і їхня доступність впливає на ваги ребер графа. Крім того, погодні умови, наприклад, дощі або снігопади, також можуть погіршувати прохідність маршрутів.

Ще одним важливим аспектом є час, необхідний для подолання певного маршруту. В умовах бойових дій швидкість доставки ресурсів може мати вирішальне значення. Наприклад, доставка боєприпасів або медичних ресурсів у визначений час може вплинути на результати операції або врятувати життя. Відповідно, ваги ребер графа можуть враховувати не лише фізичну відстань, але й затримки, спричинені перешкодами чи обхідними шляхами.

Економічний аспект також не можна ігнорувати. Логістичні маршрути в зоні бойових дій часто враховують витрати на пальне, технічне обслуговування транспортних засобів та інші ресурси. Наприклад, маршрути через гірські місцевості можуть вимагати більше пального через складність підйомів, тоді як рівнинні дороги зазвичай є економічно вигіднішими.

Пріоритетність ресурсів і вантажів є ще одним критичним фактором. У зоні бойових дій різні типи ресурсів мають різний рівень важливості. Наприклад, боєприпаси та медичне обладнання можуть мати найвищий пріоритет, тоді як продукти харчування або будівельні матеріали можуть розглядатися як менш термінові. Відповідно, ваги на маршрутах можуть змінюватися залежно від типу вантажу.

Не менш важливим є врахування динамічних змін ситуації. У зоні бойових дій маршрути можуть ставати небезпечними буквально за лічені хвилини через раптові обстріли, переміщення ворожих військ або зміни в погодних умовах. Це вимагає постійного оновлення даних у графі, щоб забезпечити актуальність моделей і оптимальність обраних шляхів.

Таким чином, ваги в графах, які моделюють маршрути у зоні бойових дій, є відображенням реальних умов, що постійно змінюються. Вони враховують безпеку, стан інфраструктури, час, економічну доцільність і пріоритетність завдань. Ефективне моделювання з урахуванням цих факторів дозволяє швидко адаптувати логістичні рішення до змінної ситуації та мінімізувати ризики під час виконання критично важливих завдань.

Приклади реальних застосувань

Реальні приклади застосування графів для моделювання логістичних маршрутів демонструють їхню ефективність у різних сферах, включаючи воєнну логістику, міське планування, транспортні системи та управління ресурсами в умовах кризи. Військова логістика є особливо важливою сферою,

де використання напрямлених зважених графів дозволяє забезпечувати оперативність і безпеку постачання.

Під час Другої світової війни математичні моделі на основі графів використовувалися для планування маршрутів доставки ресурсів союзниками. Військові аналітики враховували безпеку морських і наземних маршрутів, розташування баз постачання, стан доріг та потенційні загрози. Це дозволило ефективно управляти великими обсягами поставок і мінімізувати втрати від атак противника.

У сучасних збройних конфліктах графи використовуються для планування евакуації цивільного населення, доставки гуманітарної допомоги та оптимізації військових операцій. Наприклад, в умовах збройного конфлікту на сході України логістичні маршрути для постачання медичних ресурсів і продовольства враховували безпеку коридорів, ступінь їхнього пошкодження та можливість обстрілів. Напрямлені зважені графи використовувалися для моделювання цих маршрутів, де ваги відповідали ризику та часу доставки.

Використання графів також є важливим у логістиці НАТО. У рамках операцій у Афганістані та Іраку графові моделі допомагали планувати переміщення військових частин через складний рельєф і території з високим ризиком. Ваги на графах враховували не лише відстань і стан доріг, але й можливість атак на маршруті, а також погодні умови, що могли впливати на швидкість руху.

Ще одним яскравим прикладом є використання графів у кризових ситуаціях, таких як природні катастрофи. Під час землетрусу на Гаїті у 2010 році логістичні компанії та гуманітарні організації застосовували графові моделі для оптимізації доставки допомоги. В умовах зруйнованих доріг і обмежених ресурсів графи дозволяли знаходити найкоротші й найбезпечніші маршрути між постраждалими районами та пунктами розподілу допомоги.

У військовій сфері графові моделі активно інтегруються у сучасні системи управління, такі як автоматизовані платформи командування та контролю. Ці системи використовують графи для аналізу логістичних мереж у реальному часі, враховуючи дані з безпілотників, супутників і розвідувальних джерел. Це дозволяє оперативно реагувати на зміни ситуації на полі бою та ефективно планувати маршрути доставки критично важливих ресурсів.

Таким чином, реальні приклади застосування графів підтверджують їхню універсальність і ефективність для вирішення складних логістичних задач у різних сферах. У військовій логістиці напрямлені зважені графи забезпечують можливість моделювати динамічні та ризиковані сценарії, оптимізуючи використання ресурсів і зменшуючи втрати.

Можливості інтеграції з сучасними технологіями

Інтеграція напрямлених зважених графів із сучасними технологіями відкриває нові перспективи для вирішення логістичних задач, особливо в умовах високої динаміки, таких як зона бойових дій. Використання технологій аналізу даних, автоматизації, інтернету речей (IoT) та штучного інтелекту значно розширює функціональні можливості графових моделей.

Однією з ключових можливостей є застосування великих даних (Big Data) для автоматичного оновлення ваг графів у реальному часі. Наприклад, дані з безпілотних літальних апаратів, супутників, камер спостереження та сенсорів на транспортних засобах можуть надавати інформацію про стан доріг, погодні умови, наявність загроз або перешкод. Такі дані автоматично обробляються і використовуються для коригування моделей графів, що дозволяє адаптувати маршрути до поточної ситуації.

Інтернет речей (IoT) забезпечує безперервний моніторинг транспортних засобів і вантажів. Обладнання, яке включає GPS-трекери, датчики температури, вологості та стану пального, дозволяє з високою точністю

відстежувати місцезнаходження та стан ресурсів. Дані з IoT-пристроїв можуть інтегруватися у графові моделі, щоб враховувати технічний стан транспортних засобів і оптимізувати маршрути на основі їхньої доступності.

Штучний інтелект (AI) і машинне навчання дозволяють не лише аналізувати великі обсяги даних, але й прогнозувати майбутні зміни. Наприклад, алгоритми можуть передбачати, як зміняться умови безпеки на маршруті залежно від активності противника чи погодних умов. Це дозволяє заздалегідь планувати альтернативні шляхи та мінімізувати ризики. Крім того, AI може автоматично визначати найбільш ефективні маршрути з урахуванням усіх факторів, таких як безпека, швидкість і витрати.

Хмарні обчислення відкривають можливості для масштабованого моделювання графів. Завдяки хмарним платформам військові та логістичні організації можуть зберігати та обробляти великі графові структури, виконувати складні обчислення та забезпечувати доступ до даних у реальному часі з будь-якої точки світу. Це особливо важливо в умовах розподілених командувань, які потребують оперативного доступу до інформації.

Розвиток технологій доповненої реальності (AR) дозволяє інтегрувати графові моделі у візуальні інтерфейси. Наприклад, командири можуть використовувати AR-окуляри для перегляду маршруту в тривимірному просторі, що дозволяє краще зрозуміти ситуацію на місцевості. Такий підхід сприяє прийняттю більш обґрунтованих рішень у складних умовах.

Кібербезпека також є важливим аспектом інтеграції графових моделей. У зоні бойових дій захист інформації про логістичні маршрути є критичним, адже їхнє розкриття може призвести до втрат. Сучасні технології шифрування та захисту даних забезпечують безпечну передачу інформації між вузлами системи.

Таким чином, інтеграція напрямлених зважених графів із сучасними технологіями, такими як Big Data, IoT, AI, хмарні обчислення та AR, значно

підвищує ефективність планування логістичних маршрутів. Це дозволяє адаптуватися до динамічних умов, зменшувати ризики й оптимізувати використання ресурсів навіть у найскладніших ситуаціях.

Аналіз потенційних ризиків

Аналіз потенційних ризиків при використанні напрямлених зважених графів для моделювання логістичних маршрутів у зоні бойових дій є критичним аспектом, який допомагає передбачити можливі проблеми та розробити стратегії їхнього уникнення. У цій сфері ризики пов'язані як із зовнішніми, так і внутрішніми факторами, що впливають на функціонування графових моделей і загальну ефективність логістичних операцій.

Одним із ключових ризиків є недостатня якість даних, які використовуються для побудови графів і визначення ваг. У зоні бойових дій джерела інформації, такі як супутники, безпілотники або розвідка, можуть бути неповними, застарілими або неточними через обмеження в доступі до місцевості чи активні дії противника. Використання недостовірних даних може призвести до помилкових рішень, наприклад, вибору маршруту, який є небезпечним або непридатним для пересування.

Ще одним серйозним ризиком є динамічність ситуації на місцевості, яка може змінюватися в реальному часі через бойові дії, блокування доріг або зміни погодних умов. Якщо граф не оновлюється оперативно, це може зробити запропоновані маршрути неактуальними. Наприклад, зруйнований міст чи раптовий обстріл на маршруті можуть унеможливити рух транспорту.

Залежність від технологій створює додаткові виклики. Умови бойових дій часто супроводжуються перешкодами для роботи технологічної інфраструктури, такими як переривання зв'язку, збої в роботі GPS або проблеми із забезпеченням живлення для пристроїв. У таких випадках система моделювання графів може втратити актуальність, і прийняття рішень

доведеться переводити в ручний режим, що значно знижує швидкість та ефективність.

Кіберзагрози є ще одним критичним ризиком. Противник може намагатися зламати систему для отримання даних про маршрути або навіть внести зміни до графів, що призведе до помилкових рішень. Наприклад, спотворені ваги в графі можуть спрямувати ресурси через небезпечну зону або вивести їх на маршрут, де очікується атака.

Обмеженість ресурсів є також важливим фактором ризику. У воєнних умовах ресурси, такі як транспорт, паливо чи персонал, можуть бути недостатніми для обслуговування навіть оптимальних маршрутів. Це особливо актуально, якщо модель графа не враховує наявні обмеження під час розрахунків.

Нелінійні взаємозв'язки між факторами можуть ускладнювати точний розрахунок ваг. Наприклад, одночасний вплив погодних умов, стану доріг і бойових дій може створювати складні сценарії, які важко змоделювати точно. Така ситуація може призвести до вибору субоптимального маршруту.

Ще одним ризиком є недостатня адаптивність алгоритму до негативних сценаріїв, наприклад, до розширення конфліктної зони. Алгоритм Дейкстри передбачає, що граф не змінюється під час обчислення, проте у реальних умовах зони бойових дій структура графа може суттєво трансформуватися навіть за короткий проміжок часу.

Для мінімізації цих ризиків важливо впроваджувати:

- Механізми оперативного збору та обробки даних для забезпечення актуальності графа
- Технології шифрування і кіберзахисту для захисту даних
- Інтеграцію альтернативних джерел інформації, таких як локальна розвідка

- Резервні сценарії для ручного прийняття рішень у разі збоїв

Аналіз ризиків дозволяє не лише передбачити можливі проблеми, але й закласти стратегії їхнього уникнення, що є критично важливим для ефективного управління логістикою в умовах бойових дій.

Висновки і подальший розвиток

Напрямлені зважені графи є потужним інструментом для моделювання логістичних маршрутів у зоні бойових дій. Їх використання дозволяє враховувати складні умови воєнного середовища, зокрема динамічність ситуації, обмеження ресурсів, ризики та різноманітні фактори, що впливають на ефективність маршруту. Завдяки алгоритму Дейкстри можна оперативно знаходити оптимальні рішення, що є критично важливим для забезпечення безперебійної доставки ресурсів і підтримки бойових операцій.

Однак аналіз також виявив ряд викликів, таких як динамічні зміни ваг графа, залежність від точності даних, ризики кіберзагроз і технічні обмеження в умовах збройних конфліктів. Ці аспекти вимагають уваги і стають важливими напрямками для подальшого розвитку цієї методології.

Для вдосконалення підходів і розширення можливостей графових моделей доцільно зосередитися на таких напрямках:

1. **Розробка адаптивних алгоритмів**: Алгоритми, здатні враховувати динамічні зміни графа в реальному часі, зокрема нові ваги, розриви маршрутів чи зміну пріоритетів. Це дозволить швидше реагувати на зміни в бойових умовах.
2. **Інтеграція технологій штучного інтелекту**: Використання методів машинного навчання для прогнозування ризиків і виявлення закономірностей у даних, які впливають на ваги графів. Це підвищить точність і надійність рішень.

3. **Посилення кібербезпеки**: Розробка більш надійних механізмів шифрування даних і захисту від атак на системи, що використовують графові моделі.
4. **Створення розподілених систем управління**: Використання хмарних технологій для забезпечення доступності та масштабованості графових моделей навіть за умов обмежених ресурсів.
5. **Розширення сфер застосування**: Використання графових моделей у суміжних областях, таких як евакуація цивільного населення, планування гуманітарних місій і розподіл ресурсів у кризових ситуаціях.

Подальший розвиток передбачає також експериментальну перевірку нових моделей у симуляційних середовищах, що імітують реальні бойові умови. Такий підхід дозволить оцінити ефективність запропонованих рішень та їхню адаптованість до складних сценаріїв.

В цілому, інтеграція сучасних технологій, вдосконалення алгоритмів і врахування специфіки бойових умов забезпечать подальше зростання ефективності та надійності напрямлених зважених графів як ключового інструменту для управління логістикою в зоні бойових дій.

1.4 Формалізація предметної області

Предметна область, яка має бути формалізованою в межах завдань, пов'язана з управлінням логістичними маршрутами у зоні ведення бойових дій. Це включає в себе ряд ключових аспектів:

- Військова логістика: Воєнна логістика включає в себе планування, координацію та виконання процесів по забезпеченню військових підрозділів ресурсами, матеріалами та послугами у будь-яких умовах, включаючи зони активних бойових дій.
- Логістичні маршрути: Це шляхи та маршрути, якими рухаються та постачаються ресурси від пунктів постачання до військових одиниць у зоні бойових дій. Вони повинні бути оптимізованими з точки зору часу, витрат та безпеки.
- Алгоритм Дейкстри: Це математичний алгоритм, що використовується для знаходження найкоротшого шляху в графі з невід'ємними вагами ребер. В контексті логістики він може бути застосований для прокладання оптимальних маршрутів доставки.
- Зона ведення бойових дій: Це територія, де відбуваються воєнні дії. Ця зона може бути динамічною та небезпечною для переміщення логістичних засобів.
- Формалізація завдань: Це процес перетворення практичних аспектів управління логістичними маршрутами у математичні моделі, які можна аналізувати та оптимізувати за допомогою алгоритмів, таких як алгоритм Дейкстри.

Таким чином, формалізація предметної області включає в себе розуміння основних принципів та вимог військової логістики, вивчення характеристик логістичних маршрутів у зоні бойових дій, а також розробку математичних моделей та алгоритмів для їх оптимізації.

РОЗДІЛ 2

2.1 Обґрунтування та аналіз переваг й недоліків алгоритму Дейкстри та альтернатив

Алгоритм Дейкстри — це класичний алгоритм пошуку найкоротших шляхів у графі з невід'ємними вагами на ребрах. Він використовується для обчислення найкоротших шляхів від заданого початкового вузла до всіх інших вузлів у графі.

Основні компоненти алгоритму:

1. **Граф**: Множина вершин (вузлів) та ребер (доріг) між ними. Кожне ребро має вагу (вартість, відстань), яка вказує на ціну переміщення між двома вузлами.
2. **Початковий вузол**: Вузол, з якого починається пошук найкоротших шляхів до інших вузлів графа.
3. **Відстань**: Для кожного вузла підтримується найкоротша відома відстань від початкового вузла. Спочатку всі відстані встановлюються в нескінченність (окрім початкового вузла, для якого відстань дорівнює 0).
4. **Попередник**: Для кожного вузла зберігається попередній вузол у найкоротшому шляху.
5. **Множина відвіданих вузлів**: Вузли, для яких вже знайдений найкоротший шлях і вони більше не будуть розглядатися.
6. **Множина невідвіданих вузлів**: Вузли, для яких ще не знайдений остаточний найкоротший шлях. Алгоритм працює з ними в порядку зростання відстаней.

Покроковий опис роботи алгоритму:

1. **Ініціалізація**:
    - Всі вузли позначаються як "невідвідані".

- Для кожного вузла встановлюється початкова відстань: для початкового вузла вона дорівнює 0, для всіх інших — нескінченність.
- Для початкового вузла відстань ggg дорівнює 0.
- Всі попередники вузлів спочатку невизначені.

2. **Основний цикл**:
   - Поки залишаються невідвідані вузли обирається **невідвіданий вузол** з мінімальною відстанню.
   - Якщо цей вузол є кінцевим вузлом (якщо алгоритм шукає найкоротший шлях до конкретного вузла), алгоритм завершується.
   - Вузол позначається як відвіданий і більше не розглядатиметься.
   - Для кожного сусіднього вузла (з’єднаного ребром з поточним) обчислюється потенційна відстань до сусіда через поточний вузол
   - Якщо знайдено коротший шлях до сусіднього вузла оновлюємо: відстань до сусіднього вузла і попередник вузла, щоб знати, через який вузол йде цей найкоротший шлях.
3. **Закінчення роботи**:
   - Коли всі вузли оброблені, алгоритм завершується.
   - Результатом є таблиця найкоротших відстаней від початкового вузла до кожного іншого вузла, а також шлях, яким можна досягти кожного вузла (за допомогою попередників).

Особливості:

Оптимальність: Алгоритм Дейкстри гарантує знаходження найкоротшого шляху для всіх вузлів у графі, якщо всі ребра мають невід'ємні ваги.

Обмеження: Алгоритм не працює з графами, які містять ребра з від'ємними вагами (для таких графів використовується алгоритм Беллмана-Форда).

Переваги алгоритму Дейкстри:

- **Гарантовано оптимальний результат** для графів з невід'ємними вагами.
- **Загальна ефективність**: Широко використовується для задач маршрутизації та навігації.

Недоліки алгоритму Дейкстри:

- **Не підходить для графів з від'ємними вагами**.
- **Потребує багато часу та пам'яті** для великих графів, особливо якщо не використовуються спеціальні структури даних, такі як купи.

Альтернативи алгоритму Дейкстри включають в себе алгоритм **Беллмана-Форда** і **алгоритм A*** (A-star). Кожен з них має свої переваги та недоліки, такі як робота з від'ємними вагами ребер, ефективність обчислень і можливість адаптації до різноманітних умов. Вибір конкретного алгоритму залежить від специфіки задачі та вимог щодо швидкості та точності результів.

Алгоритм Беллмана-Форда — це алгоритм для пошуку найкоротших шляхів у графі, який може містити від'ємні ваги на ребрах. Його особливістю є здатність не лише знаходити найкоротші шляхи, але й виявляти негативні цикли, через які неможливо визначити найкоротший шлях. Алгоритм обробляє граф шляхом "розслаблення" кожного ребра кілька разів, поступово покращуючи відстані до вершин, якщо знаходить коротший шлях.

Основні компоненти алгоритму:

1. **Граф**: Це набір вершин і ребер з вагами. Відстані між вершинами можуть бути як додатними, так і від'ємними.
2. **Початковий вузол**: Вузол, з якого починається пошук найкоротших шляхів до інших вершин.

3. **Відстань**: Для кожної вершини зберігається найкоротша відома відстань від початкової вершини. Спочатку відстань до початкової вершини дорівнює 0, до всіх інших — нескінченність.
4. **Попередник**: Для кожної вершини зберігається попередній вузол, через який проходить найкоротший шлях.

Покроковий опис роботи алгоритму:

1. **Ініціалізація**:
   - Всі вузли позначаються як такі, що мають нескінченну відстань, крім початкового вузла, для якого відстань дорівнює 0.
   - Попередники всіх вузлів спочатку невизначені.
2. **Основний цикл**:
   - Алгоритм виконує V − 1 ітерацій (де V — кількість вершин у графі).
   - На кожній ітерації він перевіряє всі ребра та виконує "розслаблення": якщо відстань до кінцевої вершини через початкову менша за поточну відстань до цієї вершини, відстань оновлюється.
   - Оновлюється також попередник вершини, щоб потім можна було відновити шлях.
3. **Перевірка на негативні цикли**:
   - Після V − 1 ітерацій всі можливі найкоротші шляхи мають бути знайдені.
   - Алгоритм виконує додаткову ітерацію для перевірки наявності негативних циклів: якщо на цій ітерації знаходиться шлях, що може бути ще скорочений, це означає наявність негативного циклу.
4. **Завершення роботи**:
   - Якщо негативних циклів не знайдено, алгоритм завершує роботу, і результати включають найкоротші відстані від початкового вузла до кожної вершини, а також шляхи до цих вершин.

Особливості:

Гарантоване знаходження найкоротших шляхів для графів з від'ємними вагами, за умови, що немає негативних циклів.

Виявлення негативних циклів: Алгоритм виявляє, чи містить граф негативні цикли, що можуть призвести до нескінченного скорочення шляхів.

Складність: Часова складність алгоритму становить O(V · E), де V — кількість вершин, а E — кількість ребер, що робить його менш ефективним для великих графів, ніж алгоритм Дейкстри.

Переваги алгоритму Беллмана-Форда:

- Працює з графами, де є від'ємні ваги ребер.
- Здатен виявляти негативні цикли в графах.
- Простий для реалізації та універсальний для багатьох типів графів.

Недоліки алгоритму Беллмана-Форда:

- Часова складність O(V · E) робить його повільним для великих графів.
- У деяких випадках алгоритм може бути менш ефективним, ніж інші алгоритми, такі як Дейкстра.

Алгоритм A* (A-star) — це один із найефективніших пошукових алгоритмів для знаходження найкоротшого шляху в графах, особливо в задачах пошуку шляху в іграх або навігаційних системах. Він є вдосконаленням алгоритму Дейкстри, оскільки використовує евристичну функцію для керування процесом пошуку, що дозволяє пришвидшити пошук.

Основні компоненти алгоритму A*:

1. **Граф**: Вузли (вершини) та ребра (зв’язки) з вагами, де кожен вузол представляє певне місце або стан, а ребра визначають можливий рух або перехід між цими станами.
2. **Початковий вузол**: Вузол, з якого починається пошук шляху.
3. **Цільовий вузол**: Вузол, до якого ми шукаємо найкоротший шлях.

4. **Відкрита множина (open list)**: Множина вузлів, які планується обробити, але ще не оброблені.
5. **Закрита множина (closed list)**: Множина вузлів, які вже оброблені.
6. **Г-функція**: Вартість шляху від початкового вузла до поточного вузла.
7. **Н-функція**: Евристична функція, яка оцінює приблизну відстань від поточного вузла до цільового вузла (найчастіше використовується евклідова або мангеттенська відстань).
8. **F-функція**: Вартість поточного вузла, яка є сумою f(n)=g(n)+h(n)f(n) = g(n) + h(n)f(n)=g(n)+h(n), де:

   g(n)g(n)g(n) — реальна вартість шляху від початку до поточного вузла nnn

   h(n)h(n)h(n) — евристична оцінка залишку шляху від поточного вузла nnn до цілі

Покроковий опис роботи алгоритму:

1. **Ініціалізація**:
   - Додаємо початковий вузол до відкритої множини (open list), встановивши його значення ggg дорівнює 0 і обчисливши значення fff.
   - Всі інші вузли на початку мають ggg-значення нескінченності, оскільки ми не знаємо шлях до них.
2. **Основний цикл**:
   - Поки відкрита множина не порожня, повторюємо наступні кроки
3. **Вибір вузла з найменшим значенням fff**:
   - Знаходимо вузол з найменшим значенням fff у відкритій множині. Це вузол, для якого сума вартості пройденого шляху та евристичної оцінки найбільш перспективна.
   - Якщо цей вузол є цільовим, алгоритм закінчується — шлях знайдено.
4. **Переміщення вузла до закритої множини**:
   - Оброблений вузол переміщується до закритої множини, що означає, що ми більше не будемо його розглядати.

5. **Оновлення сусідів**:
    - Для кожного сусіда поточного вузла (який з'єднаний з ним ребром):
    - Якщо сусід не в закритій множині, обчислюємо його нове значення ggg (вартість шляху через поточний вузол).
    - Якщо цей шлях до сусіда є кращим (менша вартість ggg), оновлюємо значення ggg і перераховуємо значення fff.
    - Якщо сусід ще не в відкритій множині, додаємо його до неї.
6. **Кінець**:
    - Якщо досягнуто цільовий вузол, алгоритм завершується, і шлях можна відновити, простеживши всі вузли від цілі до початку.
    - Якщо відкрита множина стає порожньою, і ціль не досягнута, шлях не існує.

Евристики:

Евристична функція h — це ключовий елемент, який відрізняє A* від алгоритму Дейкстри. Вона дозволяє оцінити відстань до цілі, щоб сконцентрувати пошук у більш перспективних напрямках. Найпопулярніші евристики:

- **Евклідова відстань**: Використовується, коли можна рухатися по діагоналях і в будь-яких напрямках.

$$\mathrm{h(n)} = \sqrt{(x_n - \mathrm{x}_{goal})^2 + (\mathrm{y}_n - \mathrm{y}_{goal})^2}$$

- **Мангеттенська відстань**: Використовується в системах, де можна рухатися лише горизонтально або вертикально.

$$\mathrm{h(n)} = \left|\mathrm{x}_n - \mathrm{x}_{goal}\right| + \left|\mathrm{y}_n - y_{goal}\right|$$

Приклад роботи алгоритму:

Припустимо, у нас є сітка, де ми можемо рухатися лише по горизонталі та вертикалі. Початковий вузол знаходиться в лівому верхньому куті (A), а цільовий вузол — в правому нижньому (B). В процесі роботи алгоритм A* розширює вузли, які ведуть до цілі, уникаючи тупикових ділянок і зменшуючи кількість оброблених вузлів завдяки евристичній функції.

Переваги A*:

1. **Ефективність**: Завдяки евристичному підходу, A* обробляє меншу кількість вузлів порівняно з алгоритмом Дейкстри, що робить його швидшим.
2. **Гнучкість**: Можна адаптувати за допомогою різних евристик для специфічних умов.
3. **Оптимальність**: Якщо евристика є допустимою (не переоцінює), алгоритм знайде оптимальний шлях.

Недоліки A*:

1. **Час і пам'ять**: Алгоритм може споживати багато пам'яті, якщо пошуковий простір великий.
2. **Ефективність залежить від евристики**: Якщо евристична функція неадекватна, алгоритм може працювати неефективно.

A* — це потужний інструмент для задач пошуку шляху і широко використовується в багатьох сферах, від відеоігор до робототехніки та навігаційних систем.

Алгоритм Дейкстри задовільняє потребам виконання завдання через його ефективність у роботі з графами з невід'ємними вагами ребер, що є важливим у вирішенні задачі прокладання логістичних маршрутів у зоні ведення бойових дій. Цей алгоритм має просту реалізацію та добре досліджену теоретичну базу, що сприяє його швидкому впровадженню та застосуванню у практичних умовах. Крім того, алгоритм Дейкстри підходить для вирішення задачі з прокладанням найкоротших шляхів, що є ключовим аспектом в

управлінні логістичними маршрутами. Враховуючи обмеження даної області, такі як відсутність від'ємних ваг та потреба у швидких та ефективних рішеннях, алгоритм Дейкстри виявляється оптимальним вибором.

Отже, обрання алгоритму Дейкстри зумовлене його відповідністю вимогам задачі, його простотою та ефективністю у роботі з невеликими та середніми графами, які характерні для даного контексту.

2.2 Формалізація математичної моделі дослідження

Формалізація математичної моделі для задачі прокладання логістичних маршрутів у зоні ведення бойових дій базується на використанні напрямлених зважених графів, де вершини представляють пункти постачання, вузли військової інфраструктури або важливі географічні точки, а ребра відображають можливі маршрути між цими точками з відповідними вагами, що характеризують різні фактори.

Вхідні дані:

- G = (V, E) — напрямлений граф, де V — множина вершин, а E — множина ребер.
- w(u, v) — вага ребра між вершинами u та v, яка визначає вартість або ризик пересування між цими точками. Ваги можуть враховувати такі фактори, як:
    - Довжина шляху
    - Час пересування
    - Безпека маршруту
    - Стан дорожньої інфраструктури
    - Рівень ризику через бойові дії
- s — стартова вершина (точка відправлення).
- t — цільова вершина (точка призначення).

Вихідні дані:

- d(s, t) — мінімальна вартість (найкоротша відстань) між стартовою та цільовою вершинами.
- P(s, t) — найкоротший маршрут між стартовою та цільовою вершинами.

Опис процесів:

- **Задача пошуку найкоротшого шляху**: Мета полягає в знаходженні маршруту між точками sss та ttt з мінімальним значенням сумарної ваги ребер, що утворюють шлях. Це формалізується як:

$$\mathrm{P}(\mathrm{s},\mathrm{t}) = \mathrm{arg}^{min}_{\text{шлях P}} \sum_{(u,v)\in P} w(u,v)$$

Тут $\sum_{(u,v)\in P} w(u,v)$ — це сума ваг ребер, що утворюють маршрут PPP.

- **Вплив зони бойових дій**: Для кожного ребра можна ввести додаткові параметри, які відображають вплив зони бойових дій. Наприклад:

$$w'(u,v) = w(u,v) + r(u,v)$$

де $r(u,v)$ — коефіцієнт ризику, пов'язаний з рівнем небезпеки в зоні бойових дій. Це дозволяє адаптувати алгоритм до реальних умов воєнного конфлікту.

- **Алгоритм знаходження найкоротшого шляху**: Для вирішення задачі використовується алгоритм Дейкстри, який ефективно знаходить найкоротший шлях у графі з додатними вагами ребер. Алгоритм будує найкоротші шляхи шляхом поступового розширення множини оброблених вершин, обчислюючи відстань до кожної вершини.

- **Система обмежень**: Крім основного завдання мінімізації відстані, можна додати обмеження, що відображають специфіку зони бойових дій:

  - Обмеження на максимальну відстань або час, що може бути витрачено на маршрут.
  - Обмеження на використання певних ділянок маршруту через небезпечні зони.

- **Формалізація метрик**: Окрім знаходження найкоротшого шляху, результати алгоритму можуть бути проаналізовані за допомогою таких метрик:

- Загальна довжина маршруту: $\sum_{(u,v)\in P(s,t)} w(u,v)$
- Час доставки: $T(P) = \sum_{(u,v)\in P(s,t)} t(u,v)$ — де t(u, v) час, необхідний для пересування між вершинами.
- Рівень безпеки маршруту: $\sum_{(u,v)\in P(s,t)} r(u,v)$

Взаємозв'язок між вхідними параметрами та результатами:

- Вхідні параметри G = (V, E) ваги w(u, v), коефіцієнти ризику r(u, v) та обмеження часу впливають на вибір найкоротшого маршруту та загальну вартість шляху.
- Результати включають найкоротший шлях P(s, t), його вартість d(s, t), а також додаткові метрики, що відображають ефективність та безпеку маршруту.

Програмна реалізація:

Для реалізації моделі використовується Python із бібліотеками:

- **networkx** — для представлення графів і виконання алгоритму Дейкстри.
- **heapq** — для ефективної роботи з чергою з пріоритетом під час пошуку найкоротшого шляху.
- **matplotlib** — для візуалізації графів і результатів обчислень.

Ця математична модель дозволяє формалізувати процес прокладання логістичних маршрутів у зоні ведення бойових дій, з урахуванням різноманітних факторів і ризиків. Вона адаптивна до умов війни та здатна забезпечити ефективне планування логістичних операцій.

2.3 Інструменти для реалізації експеременталної програми

Середовище **Windows** є однією з основних платформ, що забезпечують комфортну та продуктивну роботу для розробників програмного забезпечення. Це середовище володіє широким набором функцій і можливостей, які роблять його ідеальним вибором для реалізації експериментальної програми, пов'язаної з прокладанням логістичних маршрутів у зоні ведення бойових дій. Основні аспекти середовища Windows як інструменту розробки включають:

1. **Доступність програмного забезпечення**:

Windows підтримує велику кількість програмних засобів, необхідних для розробки, включаючи середовища програмування, текстові редактори та спеціалізовані бібліотеки. Серед популярних інструментів — Python, PyCharm, а також бібліотеки, такі як NetworkX, Matplotlib та інші.

2. **Кросплатформна сумісність**:

Середовище Windows підтримує різноманітні інструменти та фреймворки, що дозволяє легко інтегрувати їх у розробку. Наприклад, багато бібліотек, призначених для роботи з графами та алгоритмами, мають версії для Windows, що спрощує процес налаштування та використання.

3. **Графічний інтерфейс**:

Windows забезпечує зручний графічний інтерфейс, що дозволяє розробникам легше взаємодіяти з програмним забезпеченням. Інтуїтивно зрозумілий інтерфейс PyCharm, зокрема, дозволяє швидко налаштовувати середовище, створювати проекти, працювати з файлами та здійснювати налагодження коду.

4. **Підтримка командного рядка**:

Windows надає можливість використовувати командний рядок для виконання команд, що робить його корисним інструментом для управління середовищем розробки. Командний рядок може бути використаний для виконання скриптів Python, установки пакетів через pip, а також для роботи з системами контролю версій, такими як Git.

5. **Системи контролю версій**:

Windows підтримує популярні системи контролю версій, такі як Git. Інтеграція з такими інструментами дозволяє ефективно управляти кодом, відслідковувати зміни, працювати в команді та реалізовувати методи DevOps.

6. **Розробка графічних інтерфейсів**:

Windows також надає можливості для створення графічних інтерфейсів для програм. Це може бути корисним для візуалізації результатів роботи алгоритму прокладання маршрутів, що дозволяє користувачам легше взаємодіяти з програмою.

7. **Можливості налагодження та тестування**:

Windows підтримує потужні інструменти налагодження, такі як вбудовані засоби у PyCharm. Це дозволяє розробникам з легкістю виявляти помилки, перевіряти логіку програми та забезпечувати якість коду.

8. **Спільнота та ресурси**:

Величезна спільнота користувачів Windows забезпечує доступ до безлічі ресурсів, таких як форуми, документація та навчальні матеріали. Це дозволяє розробникам швидко знаходити рішення на питання, що виникають під час роботи.

Отже, середовище Windows є потужним і зручним інструментом для реалізації експериментальної програми, пов'язаної з прокладанням логістичних маршрутів. Його багатофункціональність, підтримка різних

інструментів та можливість інтеграції з численними бібліотеками роблять його оптимальним вибором для виконання поставлених завдань у рамках дослідження.

Python - це інтерпретована, високорівнева мова програмування загального призначення. Однією з її основних переваг є простота вивчення та зрозуміння, що робить її досить популярною серед початківців і досвідчених програмістів. Давай розглянемо деякі переваги Python у контексті даного завдання:

- Зручний синтаксис: Python має простий та зрозумілий синтаксис, що робить код більш зрозумілим та легко читабельним. Це особливо важливо для великих проектів, де чіткість коду стає ключовою.
- Велика кількість бібліотек: Python має широкий вибір сторонніх бібліотек для різних завдань, включаючи роботу з графами (наприклад, networkx), обробку даних, візуалізацію та багато іншого. Це дозволяє реалізувати складні алгоритми та вирішувати різноманітні завдання за короткий час.
- Широке застосування: Python підходить для вирішення різноманітних завдань, від наукових досліджень до веб-розробки та машинного навчання. Це дає можливість легко і швидко розширювати функціональність програми за потреби.
- Спільнота розробників: Python має велику та активну спільноту розробників, яка постійно розвивається та підтримує різні проекти. Це означає, що завжди можна знайти допомогу або пораду на форумах чи у блогах.
- Широке використання в індустрії: Python використовується в багатьох компаніях та проектах, включаючи технологічні гігантів, такі як Google, Facebook та Netflix. Отже, знання Python може бути корисним у подальшій кар'єрі.

З урахуванням цих переваг, Python є відмінним вибором для вирішення різноманітних завдань, включаючи реалізацію алгоритмів для роботи з графами, таких як алгоритм Дейкстри.

PyCharm - це інтегроване середовище розробки (IDE) для мови програмування Python, розроблене компанією JetBrains. Воно надає розширені можливості для написання, відлагодження та тестування програм на Python.

Існують кілька причин, чому PyCharm є відмінним вибором для розробки програм на Python.

По-перше, воно має повний функціонал, який включає автодоповнення коду, відлагодження, підтримку систем контролю версій та інші корисні інструменти. Це дозволяє розробникам працювати ефективно та продуктивно.

По-друге, PyCharm оптимізований спеціально для роботи з Python. Він має ряд функцій, призначених для полегшення написання коду на цій мові, включаючи підтримку вирівнювання, аналіз коду та підказки щодо його структури. Крім того, PyCharm має широкі можливості налаштування, що дозволяє користувачам налаштовувати середовище розробки під свої потреби. Це дозволяє кожному розробнику налаштувати IDE так, як йому зручно, і підвищує загальну продуктивність роботи.

Узагальнюючи, PyCharm є потужним та зручним інструментом для розробки програм на мові програмування Python. Його розширені можливості та зручний інтерфейс дозволяють розробникам ефективно працювати над проектами будь-якої складності.

Модуль **heapq** у Python надає функціонал для роботи з пріоритетними чергами, основаними на купі (heap), що дозволяє швидко виконувати операції додавання, вилучення та отримання мінімального (або максимального) елемента. Він містить функції для створення та роботи з мінімальною купою, включаючи heapify для перетворення ітерованого об'єкту в купу, heappush для додавання елементу до купи, heappop для вилучення та повернення найменшого елемента, і heapreplace для вилучення та повернення найменшого

елемента, а потім додавання нового елемента. Також модуль містить функції nlargest та nsmallest для отримання n найбільших або найменших елементів з ітерованого об'єкту. У нашій програмі, яка моделює прокладання логістичних маршрутів за допомогою алгоритму Дейкстри, модуль heapq використовується для реалізації алгоритму Дейкстри. Зокрема, функція heappop використовується для вилучення найменшого елемента з пріоритетної черги, що дозволяє знаходити вершину з найменшою відстанню на кожному кроці алгоритму. Функція heappush використовується для додавання нових вершин у пріоритетну чергу, коли знайдена відстань до них менша за попередню.

Модуль **networkx** у Python є потужною бібліотекою для створення, маніпулювання та вивчення складних графів і мереж. Він надає широкий набір інструментів для роботи з графами, включаючи функції для генерації різних типів графів (ненапрямлених, напрямлених, зважених, багатографів тощо), аналізу властивостей графів, візуалізації та багатьох інших операцій.

Основні можливості networkx включають:

- Створення графів: Легке створення графів різних типів, таких як ненапрямлені графи, напрямлені графи, багатографи тощо.
- Додавання вузлів та ребер: Можливість додавання і видалення вузлів та ребер, а також атрибутів для них (наприклад, ваги ребер, метаданих вузлів).
- Аналіз графів: Вбудовані функції для обчислення різноманітних параметрів графів, таких як ступінь вузлів, короткі шляхи, центральність, кластеризація та інші метрики.
- Алгоритми: Підтримка багатьох класичних алгоритмів теорії графів, таких як пошук найкоротших шляхів (алгоритм Дейкстри, алгоритм Беллмана-Форда), пошук мінімальних кістякових дерев (алгоритм Крускала, алгоритм Пріма), знаходження компонент зв'язності тощо.

- Візуалізація: Засоби для візуалізації графів, що дозволяють відображати графи з вузлами та ребрами різних кольорів і розмірів, з підписами та іншими атрибутами.

Модуль **matplotlib** - це популярна бібліотека для візуалізації даних у Python. Вона надає широкий набір інструментів для створення різноманітних графіків, діаграм і візуалізацій, що дозволяє користувачам ефективно досліджувати та представляти дані.

Основні можливості matplotlib включають:

- Графіки: Бібліотека підтримує створення різних типів графіків, таких як лінійні графіки, гістограми, стовпчасті діаграми, кругові діаграми, розсіювання, графіки з помилками та багато інших.
- Налаштування графіків: Користувачі можуть налаштовувати зовнішній вигляд графіків, включаючи кольори, маркери, лінії, шрифти, підписи осей, заголовки, легенди та сітки.
- Інтерактивність: Підтримка інтерактивних візуалізацій, що дозволяє користувачам взаємодіяти з графіками в реальному часі, змінювати масштаб, панорамувати та зберігати графіки у різних форматах (PNG, PDF, SVG тощо).
- 3D-графіка: matplotlib дозволяє створювати тривимірні графіки та поверхні, що особливо корисно для візуалізації складних даних.
- Підтримка різних середовищ: Графіки можуть бути вбудовані у різні середовища, такі як Jupyter Notebook, веб-сторінки, настільні додатки, що робить бібліотеку гнучкою та зручною для використання в різних контекстах.
- Сумісність з іншими бібліотеками: matplotlib добре інтегрується з іншими науковими бібліотеками Python, такими як NumPy, pandas, SciPy, що дозволяє легко використовувати її разом з іншими інструментами для обробки та аналізу даних.

Ця бібліотека є невід'ємною частиною екосистеми Python для наукових обчислень і аналізу даних, забезпечуючи потужні засоби для створення наочних і інформативних візуалізацій.

РОЗДІЛ 3

3.1 Результати експериментального виконання

Почнемо з того, як саме працює алгоритм Дейкстри, основна ідея полягає у тому, щоб поступово оновлювати найкоротші відомі відстані до вузлів, починаючи з початкового вузла і розширюючися на його сусідів.

Алгоритм починається з встановлення відстаней до всіх вузлів як нескінченності, за винятком початкового вузла, який має відстань 0. Початковий вузол поміщається в пріоритетну чергу з відстанню 0. Далі алгоритм витягує вузол з найменшою відстанню з пріоритетної черги і оновлює відстані до всіх його сусідів. Якщо нова відстань менша за попередню, то вона оновлюється, і сусід додається до черги з новою відстанню.

Процес повторюється, доки у черзі є елементи або доки не будуть відомі всі найкоротші відстані до всіх вузлів. В результаті отримуємо найкоротші відстані від початкового вузла до всіх інших вузлів у графі.

У даному розділі наведено результати експериментального виконання практичного завдання з використанням реалізації алгоритму Дейкстри для знаходження найкоротшого шляху в орієнтованому графі.

На початку виконання програми було задано орієнтований граф edges, що представляє собою мережу з визначеними вагами ребер між вузлами. Для визначення найкоротшого шляху між певними вузлами у графі був використаний алгоритм Дейкстри.

Після виконання алгоритму Дейкстри було отримано найкоротший шлях від вузла 'A' до вузла 'H', який складався з наступних вузлів: ['A', 'B', 'E', 'F', 'H']. Результат виведено у відповідному форматі за допомогою функції print.

Далі було побудовано граф для візуалізації отриманого найкоротшого шляху та ваг ребер. Використана бібліотека networkx дозволила побудувати

граф та візуалізувати його разом з найкоротшим шляхом. Граф було відображено з використанням matplotlib.pyplot.

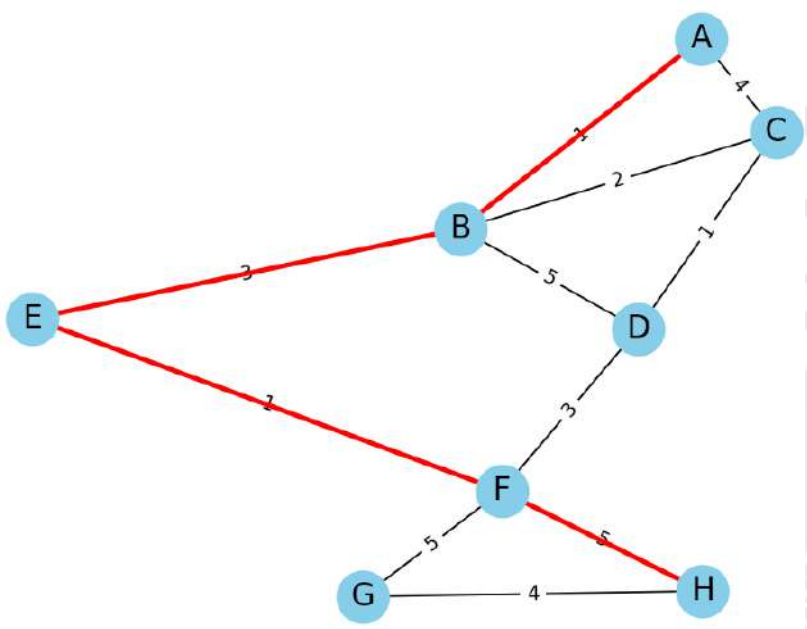

Рис. 2.1 Граф найкоротшого шляху за алгоритмом Дейкстри

Отже, результати експериментального виконання демонструють успішне виконання алгоритму Дейкстри та правильність отриманого найкоротшого шляху у вказаному графі.

3.2 Реалізація експеременталної програми

При розробці експеременталної програми було використано три бібліотеки для реалізації та візуалізації алгоритму Дейкстри.

```
import heapq
import networkx as nx
import matplotlib.pyplot as plt
```

Модуль **heapq** у Python надає функціонал для роботи з пріоритетними чергами, основаними на купі (heap), що дозволяє швидко виконувати операції додавання, вилучення та отримання мінімального (або максимального) елемента. Він містить функції для створення та роботи з мінімальною купою, включаючи heapify для перетворення ітерованого об'єкту в купу, heappush для додавання елементу до купи, heappop для вилучення та повернення найменшого елемента, і heapreplace для вилучення та повернення найменшого елемента, а потім додавання нового елемента. Також модуль містить функції nlargest та nsmallest для отримання n найбільших або найменших елементів з ітерованого об'єкту. У нашій програмі, яка моделює прокладання логістичних маршрутів за допомогою алгоритму Дейкстри, модуль heapq використовується для реалізації алгоритму Дейкстри. Зокрема, функція heappop використовується для вилучення найменшого елемента з пріоритетної черги, що дозволяє знаходити вершину з найменшою відстанню на кожному кроці алгоритму. Функція heappush використовується для додавання нових вершин у пріоритетну чергу, коли знайдена відстань до них менша за попередню.

Модуль **networkx** у Python є потужною бібліотекою для створення, маніпулювання та вивчення складних графів і мереж. Він надає широкий набір інструментів для роботи з графами, включаючи функції для генерації різних типів графів (ненапрямлених, напрямлених, зважених, багатографів тощо), аналізу властивостей графів, візуалізації та багатьох інших операцій.

Основні можливості networkx включають:

- Створення графів: Легке створення графів різних типів, таких як ненапрямлені графи, напрямлені графи, багатографи тощо.

- Додавання вузлів та ребер: Можливість додавання і видалення вузлів та ребер, а також атрибутів для них (наприклад, ваги ребер, метаданих вузлів).
- Аналіз графів: Вбудовані функції для обчислення різноманітних параметрів графів, таких як ступінь вузлів, короткі шляхи, центральність, кластеризація та інші метрики.
- Алгоритми: Підтримка багатьох класичних алгоритмів теорії графів, таких як пошук найкоротших шляхів (алгоритм Дейкстри, алгоритм Беллмана-Форда), пошук мінімальних кістякових дерев (алгоритм Крускала, алгоритм Пріма), знаходження компонент зв'язності тощо.
- Візуалізація: Засоби для візуалізації графів, що дозволяють відображати графи з вузлами та ребрами різних кольорів і розмірів, з підписами та іншими атрибутами.

Модуль **matplotlib** - це популярна бібліотека для візуалізації даних у Python. Вона надає широкий набір інструментів для створення різноманітних графіків, діаграм і візуалізацій, що дозволяє користувачам ефективно досліджувати та представляти дані.

Основні можливості matplotlib включають:

- Графіки: Бібліотека підтримує створення різних типів графіків, таких як лінійні графіки, гістограми, стовпчасті діаграми, кругові діаграми, розсіювання, графіки з помилками та багато інших.
- Налаштування графіків: Користувачі можуть налаштовувати зовнішній вигляд графіків, включаючи кольори, маркери, лінії, шрифти, підписи осей, заголовки, легенди та сітки.
- Інтерактивність: Підтримка інтерактивних візуалізацій, що дозволяє користувачам взаємодіяти з графіками в реальному часі, змінювати масштаб, панорамувати та зберігати графіки у різних форматах (PNG, PDF, SVG тощо).

- 3D-графіка: matplotlib дозволяє створювати тривимірні графіки та поверхні, що особливо корисно для візуалізації складних даних.
- Підтримка різних середовищ: Графіки можуть бути вбудовані у різні середовища, такі як Jupyter Notebook, веб-сторінки, настільні додатки, що робить бібліотеку гнучкою та зручною для використання в різних контекстах.
- Сумісність з іншими бібліотеками: matplotlib добре інтегрується з іншими науковими бібліотеками Python, такими як NumPy, pandas, SciPy, що дозволяє легко використовувати її разом з іншими інструментами для обробки та аналізу даних.

Ця бібліотека є невід'ємною частиною екосистеми Python для наукових обчислень і аналізу даних, забезпечуючи потужні засоби для створення наочних і інформативних візуалізацій.

Функція dijkstra реалізує алгоритм Дейкстри для знаходження найкоротших шляхів у графі з невід'ємними вагами ребер. Вона приймає два аргументи: граф, представлений у вигляді словника, де ключами є вузли, а значеннями — інші словники з сусідніми вузлами та відповідними вагами ребер; та початковий вузол, від якого починається обчислення найкоротших шляхів. Функція створює два словники: один для зберігання відстаней від початкового вузла до всіх інших вузлів, ініціалізуючи їх значення нескінченністю, крім початкового вузла, відстань до якого встановлюється в нуль; та другий для зберігання попередників кожного вузла на найкоротшому шляху. Використовуючи пріоритетну чергу, функція обирає вузол з найменшою відстанню та перевіряє його сусідів. Якщо нова знайдена відстань до сусіднього вузла є меншою за раніше відому, ця відстань оновлюється, і сусідній вузол додається до пріоритетної черги для подальшої обробки. Функція продовжує працювати, поки пріоритетна черга не стане порожньою, що означає обробку всіх вузлів. В результаті вона повертає два словники: один з найкоротшими відстанями до кожного вузла, а другий з попередниками кожного вузла, що дозволяє відновити найкоротші шляхи.

```
def dijkstra(graph, start):
    distances = {node: float('infinity') for node in graph}
    distances[start] = 0
    priority_queue = [(0, start)]
    previous_nodes = {node: None for node in graph}

    while priority_queue:
        current_distance, current_node = heapq.heappop(priority_queue)

        if current_distance > distances[current_node]:
            continue

        for neighbor, weight in graph[current_node].items():
            distance = current_distance + weight
            if distance < distances[neighbor]:
                distances[neighbor] = distance
                previous_nodes[neighbor] = current_node
                heapq.heappush(priority_queue, (distance, neighbor))

    return distances, previous_nodes
```

Граф представлений у вигляді словника, де кожен ключ є вузлом графа, а його значенням є інший словник, що містить сусідні вузли та відповідні ваги ребер. Цей словник описує структуру графа та ваги ребер між вузлами, що використовуються алгоритмом Дейкстри для пошуку найкоротших шляхів.

```
edges = {
    'A': {'B': 1, 'C': 4},
    'B': {'A': 1, 'C': 2, 'D': 5, 'E': 3},
    'C': {'A': 4, 'B': 2, 'D': 1},
    'D': {'B': 5, 'C': 1, 'F': 3},
    'E': {'B': 3, 'F': 1},
    'F': {'D': 3, 'E': 1, 'G': 5, 'H': 5},
    'G': {'F': 5, 'H': 4},
    'H': {'G': 4, 'F': 5}
}
```

Функція get_shortest_path знаходить найкоротший шлях від початкового вузла до кінцевого, використовуючи словник попередніх вузлів, створений алгоритмом Дейкстри.

Основні кроки функції:

- Ініціалізація: Створюється порожній список path для зберігання шляху та змінна current_node, яка починається з кінцевого вузла.
- Відновлення шляху: Функція проходить по ланцюжку попередніх вузлів, починаючи з кінцевого вузла і рухаючись у зворотному напрямку до початкового вузла. Кожен вузол додається до списку path.
- Додавання початкового вузла: Коли досягнуто початкового вузла, він також додається до списку.
- Реверсування списку: Після відновлення шляху у зворотному порядку, список path перевертається, щоб надати правильний порядок від початкового вузла до кінцевого.
- Повернення результату: Функція повертає список path, що містить послідовність вузлів, яка представляє найкоротший шлях від початкового вузла до кінцевого.

Ця функція дозволяє відновити повний найкоротший шлях, використовуючи дані про попередні вузли, отримані з алгоритму Дейкстри для подальшої візуалізації на графі коротшого шляху.

```
def get_shortest_path(previous_nodes, start, end):
    path = []
    current_node = end
    while current_node != start:
        path.append(current_node)
        current_node = previous_nodes[current_node]
    path.append(start)
    path.reverse()
    return path
```

Наступний фрагмент коду демонструє процес знаходження найкоротшого шляху в графі, використовуючи функції dijkstra та get_shortest_path. Спочатку встановлюються початковий та кінцевий вузли: змінна start_node встановлюється на вузол 'A', що вказує на початковий вузол, з якого починається обчислення найкоротших шляхів, а змінна end_node

встановлюється на вузол 'H', що вказує на кінцевий вузол, до якого потрібно знайти найкоротший шлях. Далі викликається функція dijkstra з параметрами edges (граф) та start_node (початковий вузол 'A'). Ця функція повертає два словники: distances, який містить найкоротші відстані від початкового вузла до всіх інших вузлів, та previous_nodes, який містить попередні вузли для кожного вузла на шляху до нього. Потім викликається функція get_shortest_path з параметрами previous_nodes (словар попередніх вузлів), start_node (початковий вузол 'A') та end_node (кінцевий вузол 'H'). Ця функція повертає список shortest_path, який містить послідовність вузлів, що представляє найкоротший шлях від початкового вузла 'A' до кінцевого вузла 'H'. Таким чином, цей код знаходить і повертає найкоротший шлях між заданими вузлами 'A' і 'H' у графі, використовуючи алгоритм Дейкстри та додаткову функцію для відновлення самого шляху.

```
start_node = 'A'
end_node = 'H'
distances, previous_nodes = dijkstra(edges,start_node)
shortest_path = get_shortest_path(previous_nodes,start_node,end_node)
```

Наступний фрагмент коду створює граф, використовуючи бібліотеку NetworkX, та додає в нього ребра з відповідними вагами. Спочатку створюється порожній граф G за допомогою команди nx.Graph(). Далі, використовуючи подвійний цикл, граф наповнюється ребрами і вагами з попередньо визначеного словника edges. У зовнішньому циклі перебираються всі вузли графа разом із їхніми сусідами, а у внутрішньому циклі для кожної пари вузлів (сусідніх вузлів) додається ребро з відповідною вагою до графа G. Таким чином, будується повний граф, який містить усі визначені вузли та з'єднання між ними, з урахуванням ваг кожного ребра.

```
# Create graph
G = nx.Graph()

# Adding edges and weights
for node, neighbors in edges.items():
```

```
    for neighbor, weight in neighbors.items():
        G.add_edge(node, neighbor, weight=weight)
```

Цей фрагмент коду відповідає за візуалізацію графа та найкоротшого шляху в ньому. Спочатку визначаються позиції вузлів для візуалізації за допомогою функції nx.spring_layout(G), яка створює координати для вузлів графа, використовуючи алгоритм пружинного компонування. Це допомагає візуально розподілити вузли графа у просторі для зручного перегляду. Далі граф G відображається разом з вузлами та ребрами за допомогою функції nx.draw(). Вузли відображаються з підписами, встановленим розміром та кольором, а ребра відображаються за замовчуванням. Потім до графа додаються ваги ребер, які відображаються як текст біля відповідних ребер. Це досягається за допомогою функції nx.draw_networkx_edge_labels(), яка використовує словник ваг ребер, отриманий за допомогою функції nx.get_edge_attributes(G, 'weight').

Останнім етапом є візуалізація найкоротшого шляху, знайденого раніше. Ребра, що входять до складу цього шляху, виділяються червоним кольором та товстішою лінією. Це робиться шляхом створення списку ребер на основі списку вузлів, що складають найкоротший шлях, та виклику функції nx.draw_networkx_edges() з відповідними параметрами для зміни вигляду цих ребер.

```
# Knot positions for visualisation
pos = nx.spring_layout(G)

# Visualisation of knots and edges
nx.draw(G, pos, with_labels=True, node_size=700, node_color='skyblue',
font_size=16, font_color='black')

# Visualisation of edges and weights
labels = nx.get_edge_attributes(G, 'weight')
nx.draw_networkx_edge_labels(G, pos, edge_labels=labels)

# Visualisation of the shortest path
path_edges = list(zip(shortest_path, shortest_path[1:]))
```

```
nx.draw_networkx_edges(G, pos, edgelist=path_edges, width=2.5,
edge_color='r')
```

Вкінці коду створюється відображення готової візуалізації графа. Спочатку встановлюється заголовок для графіка за допомогою функції plt.title(), який вказує, що візуалізація показує граф із виділеним найкоротшим шляхом. Після цього викликається функція plt.show(), яка відображає створений графік на екрані. Цей виклик відкриває вікно з графіком, де користувач може побачити візуалізацію графа з усіма вузлами, ребрами, вагами та виділеним найкоротшим шляхом.

```
# Show graph
plt.title("Visualization of the Graph with Shortest Path Highlighted")
plt.show()
```

ВИСНОВКИ

Практична підготовка полягала в розробці та використанні алгоритму Дейкстри для пошуку найкоротшого шляху у графі з визначеними вагами ребер. Було проведено комплексне дослідження проблеми прокладання логістичних маршрутів у зоні ведення бойових дій. Робота розкриває важливість ефективної логістики в умовах військових конфліктів, підкреслюючи специфіку та складнощі, з якими стикаються системи управління постачанням у таких ситуаціях.Отримані результати практичного застосування алгоритму були успішно відображені у візуальному вигляді на графіку, де найкоротший шлях між визначеними вузлами був підкреслений.

В рамках дослідження були вивчені сучасні підходи до вирішення задачі прокладання маршрутів, зокрема алгоритми, що забезпечують пошук оптимальних шляхів. Особлива увага була приділена алгоритму Дейкстри, як одному з найбільш ефективних методів у цій галузі, який продемонстрував свою здатність знаходити найкоротші шляхи в умовах зважених графів.

У процесі роботи було розроблено формалізовану модель, що дозволяє відображати сутність логістичних процесів. Ця модель враховує взаємозв'язок між вхідними параметрами та вихідними результатами, що є критично важливим для прийняття рішень у динамічному середовищі бойових дій. Дослідження також підкреслює значення математичних та алгоритмічних структур, які використовуються для реалізації моделі.

Важливу роль у реалізації експериментальної програми відіграло вибране програмне забезпечення та середовище розробки. Використання Python і відповідних бібліотек забезпечило необхідні можливості для моделювання та візуалізації результатів. Це дало змогу перевірити ефективність запропонованого підходу та підтвердити його практичну цінність.

Загалом, результати роботи підтверджують, що обрані алгоритмічні рішення можуть суттєво підвищити ефективність логістичних процесів у складних умовах. Дослідження пропонує основи для подальших розробок у цій галузі, включаючи можливість інтеграції нових технологій та методів, що можуть покращити адаптивність та швидкість реакції логістичних систем.

Таким чином, дана робота робить внесок у розвиток наукового підходу до управління логістикою в умовах бойових дій, підкреслюючи важливість використання сучасних алгоритмічних рішень для оптимізації процесів, що мають критичне значення для успішного виконання завдань у сфері військової логістики.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. https://www.microsoft.com/uk-ua/ - офіціальний сайт Microsoft.
2. https://www.python.org – офіціальний сайт Python з документацією до використання.
3. https://www.jetbrains.com/pycharm/ - офіціальний сайт JetBrains – розробників PyCharm з документацією до використання та посиланнями на завантаження середовища.
4. https://docs.python.org/uk/3/library/heapq.html - документація з офіціального сайту Python щодо використання бібліотеки `heapqueue`.
5. https://networkx.org/documentation/stable/tutorial.html - офіціальний сайт розробників модулю `networkx` для Python з навчальним матеріалом для використання.
6. https://matplotlib.org - офіціальний сайт розробників модулю `matplotlib` для Python з навчальним матеріалом для використання.
7. https://library.maup.com.ua/metod_disc/pdf/1525-vijsk_logist.pdf - невеликий курс по військовій логістиці, серед яких також транспортна логістика.
8. http://tstt.diit.edu.ua/article/view/272057 - стаття посвячена темі військової транспортної-логістики.
9. https://dspace.uzhnu.edu.ua/jspui/bitstream/lib/31527/1/Теорія%20графів.pdf – посилання на PDF файл з лекцією по теорії графів.
10. http://choippo.cn.sch.in.ua/Files/downloadcenter/Алгоритм%20Дейкстри.%20Теорія.pdf – посилання на статтю посвячену опису алгоритму Дейкстри.
11. https://www.mathros.net.ua/znahodzhennja-najkorotshogo-shljahu-v-orijentovanomu-grafi-za-algorytmom-dejkstry.html - посилання на статтю посвячену розвязку задач за допомогою алгоритму Дейкстри.

ДОДАТОК А

```
import heapq
import networkx as nx
import matplotlib.pyplot as plt


def dijkstra(graph, start):
    distances = {node: float('infinity') for node in graph}
    distances[start] = 0
    priority_queue = [(0, start)]
    previous_nodes = {node: None for node in graph}

    while priority_queue:
        current_distance, current_node = heapq.heappop(priority_queue)

        if current_distance > distances[current_node]:
            continue

        for neighbor, weight in graph[current_node].items():
            distance = current_distance + weight
            if distance < distances[neighbor]:
                distances[neighbor] = distance
                previous_nodes[neighbor] = current_node
                heapq.heappush(priority_queue, (distance, neighbor))

    return distances, previous_nodes


def get_shortest_path(previous_nodes, start, end):
    path = []
    current_node = end
    while current_node != start:
        path.append(current_node)
        current_node = previous_nodes[current_node]
    path.append(start)
    path.reverse()
    return path


edges = {
    'A': {'B': 1, 'C': 4},
    'B': {'A': 1, 'C': 2, 'D': 5, 'E': 3},
    'C': {'A': 4, 'B': 2, 'D': 1},
    'D': {'B': 5, 'C': 1, 'F': 3},
    'E': {'B': 3, 'F': 1},
    'F': {'D': 3, 'E': 1, 'G': 5, 'H': 5},
    'G': {'F': 5, 'H': 4},
    'H': {'G': 4, 'F': 5}
}

start_node = 'A'
```

```
end_node = 'H'
distances, previous_nodes = dijkstra(edges, start_node)
shortest_path = get_shortest_path(previous_nodes, start_node,
end_node)
print("Shortest path:", shortest_path)

# Create graph
G = nx.Graph()

# Adding edges and weights
for node, neighbors in edges.items():
    for neighbor, weight in neighbors.items():
        G.add_edge(node, neighbor, weight=weight)

# Knot positions for visualisation
pos = nx.spring_layout(G)

# Visualisation of knots and edges
nx.draw(G, pos, with_labels=True, node_size=700, node_color='skyblue',
font_size=16, font_color='black')

# Visualisation of edges and weights
labels = nx.get_edge_attributes(G, 'weight')
nx.draw_networkx_edge_labels(G, pos, edge_labels=labels)

# Visualisation of the shortest path
path_edges = list(zip(shortest_path, shortest_path[1:]))
nx.draw_networkx_edges(G, pos, edgelist=path_edges, width=2.5,
edge_color='r')

# Show graph
plt.title("Visualization of the Graph with Shortest Path Highlighted")
plt.show()
```

ДЕКЛАРАЦІЯ

про дотримання академічної доброчесності

Я, _____

_____

_____

_____

_____

_____

Повністю вказується ПІБ та статус (освітня (освітньо-наукова) програма – для здобувачів вищої освіти, назва кваліфікаційної роботи)

що нижче підписалась/підписався, розуміючи та підтримуючи загальновизнані засади справедливості, доброчесності та законності,

ЗОБОВ'ЯЗУЮСЬ:

дотримуватися принципів та правил академічної доброчесності, що визначені законодавством України, локальними нормативними актами Донецького національного університету імені Василя Стуса, положеннями, правилами, умовами, визначеними іншими суб'єктами, та не допускати їх порушення.

ПІДТВЕРДЖУЮ:

що мені відомі положення статті 42 Закону України «Про освіту»;

що у даній роботі не представляла/представляв чиїсь роботи повністю або частково як свої власні. Там, де я скористалася/скористався працею інших, я зробила/зробив відповідні посилання на джерела інформації;

що дана робота не передавалась іншим особам і подається вперше, не порушує авторських та суміжних прав закріплених статтями 21-25 Закону України «Про авторське право та суміжні права», а дані та інформація не отримувались в недозволений спосіб.

УСВІДОМЛЮЮ:

що ця робота може бути перевірена університетом на плагіат або інші порушення академічної доброчесності, в тому числі з використанням спеціалізованих сервісів;

що у разі порушення академічної доброчесності, до мене можуть бути застосовані процедури, передбачені законодавством України та Кодексом академічної доброчесності та корпоративної етики Донецького національного університету імені Василя Стуса, іншими локальними нормативними актами університету, та я можу бути притягнута/притягнутий до академічної відповідальності.

_____ _____

(дата) (підпис)