НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
Факультет прикладної математики
Кафедра прикладної математики

«До захисту допущено»
Завідувач кафедри
_____ Олег ЧЕРТОВ
«___» _____ 2023 р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Наука про дані та математичне моделювання»
спеціальності 113 «Прикладна математика»
на тему: «Логістична система оптимізації виробничих технологій»

Виконала:
студентка IV курсу, групи КМ-92
Лутак Маріна Василівна _____

Керівник:
Доцент, канд. фіз.-мат. наук, доцент
Третиник Віолета Вікентіївна _____

Консультант з нормоконтролю:
Старший викладач,
Мальчиков Володимир Вікторович _____

Рецензент:
Доцент каф. ММСА, канд. техн. наук, доцент
Дідковська М.В. _____

Засвідчую, що в цій дипломній роботі немає запозичень із праць інших авторів без відповідних посилань.
Студентка Лутак Маріна Василівна

Київ — 2023 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет прикладної математики
Кафедра прикладної математики

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 113 «Прикладна математика»

Освітньо-професійна програма «Наука про дані та математичне моделювання»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ О. Р. ЧЕРТОВ

«___» _____ 2023 р.

ЗАВДАННЯ

на дипломну роботу студентці

Лутак Маріні Василівні

1. Тема роботи: «Логістична система оптимізації виробничих технологій», керівник роботи Третиник Віолета Вікентіївна, доцент, канд фіз.-мат. наук, затверджені наказом по університету від «31» травня 2023 р. № 2108-С.
2. Термін подання студенткою роботи: «12» червня 2023 р.
3. Вихідні дані до роботи: оптимізовані логістичні маршрути матеріальних потоків.
4. Зміст роботи: провести огляд проблемної області та існуючих рішень за темою «Логістична система оптимізації виробничих технологій», знайти та вибрати набір даних, обрати метод реалізації поставленої задачі, описати математичне забезпечення, здійснити програмну реалізацію, провести тестування програми.
5. Перелік ілюстративного матеріалу: схема логістичної системи, діаграми компонентів, діяльності та послідовностей, блок-схема розробленого алгоритму, знімки екрану, що демонструють роботу програми.
6. Дата видачі завдання: «06» лютого 2023 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Огляд літератури за тематикою та збір даних	12.11.2022	
2	Проведення порівняльного аналізу існуючих програмних рішень	14.12.2022	
3	Проведення порівняльного аналізу математичних методів обробки даних	24.12.2022	
4	Підготовка матеріалів першого розділу роботи	01.02.2023	
5	Розроблення логістичної системи оптимізації виробничих технологій	01.03.2023	
6	Підготовка матеріалів другого розділу роботи	15.03.2023	
7	Підготовка матеріалів третього розділу роботи	05.04.2023	
8	Розроблення програмного забезпечення для логістичної системи оптимізації виробничих технологій	15.04.2023	
9	Підготовка матеріалів четвертого розділу роботи	03.05.2023	
10	Оформлення пояснювальної записки	01.06.2023	

Студентка _____ Маріна ЛУТАК

Керівник роботи _____ Віолета ТРЕТИНИК

АНОТАЦІЯ

Дипломну роботу виконано на 58 аркушах, вона містить 2 додатки та перелік посилань на використані джерела з 8 найменувань. У роботі наведено 47 рисунків та 6 таблиць.

Метою даної дипломної роботи є дослідження та розробка логістичної системи оптимізації виробничих технологій, яка дозволить забезпечити максимальну ефективність виробничих процесів, знизити витрати на транспортну логістику та оптимізувати час на виконання виробничих процесів

У роботі проведено аналіз проблемної області та огляд існуючих рішень указаної задачі — системи оптимізації виробничих технологій. Виконано їх короткий огляд та вказано їх переваги та недоліки. Для розв’язання задачі було використано граф, алгоритм Дейкстри та жадібний алгоритм.

Розроблено програмне забезпечення для моделювання логістичної системи оптимізації виробничих технологій на основі математичних методів

Ключові слова: логістична система, виробничі технології, логістичний ланцюжок, граф, алгоритм Дейкстри, жадібний алгоритм.

ABSTRACT

The thesis is presented in 58 pages. It contains 2 appendixes and bibliography of 8 references. Forty-seven figures and 6 tables are given in the thesis.

The goal of the thesis is to research and develop a logistics system for optimizing the production technologies, which will allow to ensure the maximum efficiency of production processes, reduce the costs of transport logistics and optimize the time for the execution of production processes.

In the thesis, the problem area is analyzed and existing solutions to the specified problem — the system of optimization of production technologies — are considered. They are briefly reviewed and their advantages and disadvantages are indicated. A graph, Dijkstra's algorithm, and a greedy algorithm are used in the thesis to solve the problem.

The software for modeling the logistic system of optimization of production technologies based on mathematical methods is developed.

Keywords: logistics system, production technologies, logistics chain, graph, Dijkstra's algorithm, greedy algorithm.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

DRP (distribution requirement planning) – логістична система планування потреб у розподілі.

JIT (just-in-time) – логістична система «точно в строк».

MRP (material requirements planning) – система планування потреб у матеріалах.

OPT (optimized production technology) – оптимізована виробнича технологія.

ВСТУП

В сучасному світі, де підприємства постійно змагаються за конкурентну перевагу, оптимізація виробничих технологій та логістичних процесів стає невід'ємною складовою успішного функціонування і підвищення ефективності підприємств. Логістична система оптимізації виробничих технологій є одним із ключових інструментів, який дозволяє раціонально керувати ресурсами, транспортуванням та процесами виробництва з метою досягнення оптимальних результатів.

Метою даної дипломної роботи є дослідження та розробка логістичної системи оптимізації виробничих технологій, яка дозволить забезпечити максимальну ефективність виробничих процесів, знизити витрати на транспортну логістику та оптимізувати час на виконання виробничих процесів. Ця система буде базуватися на використанні передових методів аналізу, моделювання та оптимізації, а також на використанні сучасних інформаційних технологій.

1 ПОСТАНОВКА ЗАДАЧІ

Метою роботи є оптимізація виробничих технологій за рахунок налагодження безперебійної логістики між різними модулями виробничих процесів.

Для досягнення поставленої мети необхідно:

1. провести огляд проблемної області;
2. провести огляд наявних рішень;
3. обрати математичну модель та методи розв’язку поставленої задачі;
4. розробити програмне забезпечення;
5. протестувати розроблену систему.

Реалізована система має задовольняти такі вимоги:

1. можливість введення даних користувачем;
2. система має оптимізовувати маршрути між точками логістичного ланцюжка.

2 ОГЛЯД ПРОБЛЕМНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ

2.1 Огляд проблемної області

Поняття "логістична система" є одним з основних термінів в галузі логістики. Під логістичною системою мається на увазі організаційно-управлінський механізм, який забезпечує узгоджену діяльність фахівців різних служб, що беруть участь у керуванні рухом матеріального потоку. Логістична система включає матеріальні ресурси, які забезпечують переміщення товарів по логістичному ланцюжку (склади, механізми для завантаження та розвантаження, транспортні засоби), запаси на виробництво та засоби керування усіма ланками ланцюжка.

Логістична система є важливою складовою сучасного бізнесу, оскільки вона допомагає забезпечити оптимальний рівень обігу товарів і послуг, зменшити витрати та покращити задоволення потреб клієнтів. Вона забезпечує ефективну координацію усіх етапів логістичного процесу, включаючи постачання, зберігання, транспортування, управління запасами та інформаційний обмін.

Основні функції логістичної системи включають планування та прогнозування потреб, організацію доставки та розподілу товарів, контроль за якістю і кількістю запасів, оптимізацію маршрутів транспортування, а також забезпечення інформаційної підтримки всього логістичного процесу [1].

Для налагодження безперебійної логістики між усіма компонентами системи розглянемо більш детально її будову. Логістична система складається з наступних підсистем (рис 2.1):

- закупівля;
- склади;
- запаси;
- транспорт;
- виробництво;
- розподіл;
- збут;

- інформація;
- кадри.

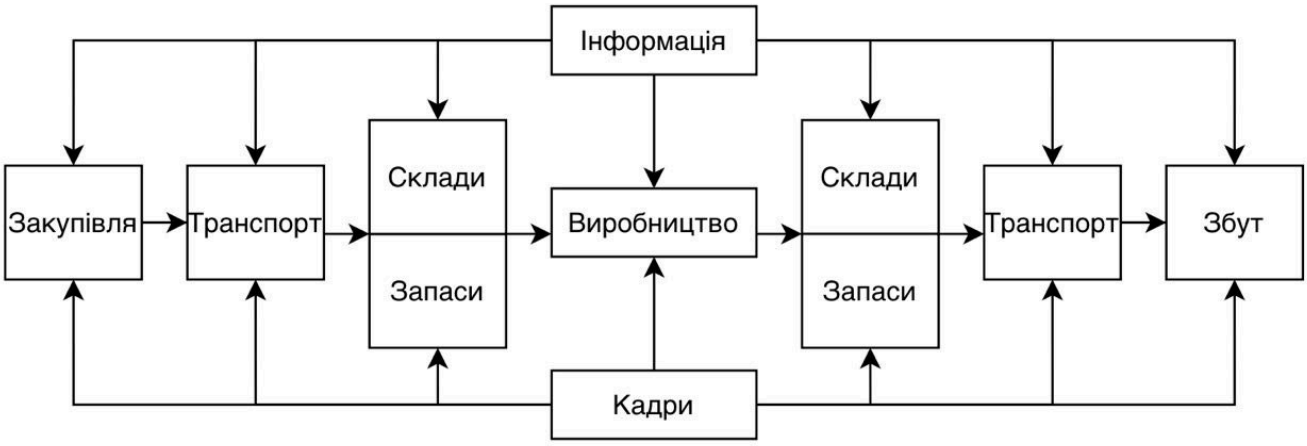

Рисунок 2.1 – Класична логістична система

Закупівля – це частина логістичної системи, яка забезпечує отримання матеріалів, сировини та інших ресурсів безпосередньо від постачальників.

Склади – це підсистема логістичної системи, яка складається з будівель, споруд, площадок та необхідного обладнання для переміщення та обробки матеріального потоку всередині "складського простору". Основна функція складів – розміщення та зберігання матеріального потоку, який перетворений на запаси, його обробка на складі та формування у зручний для транспортування "формат" для споживачів.

Запаси – це важлива складова логістичної системи, яка відображає наявність матеріальних резервів. Наявність запасів дозволяє системі бути гнучкою та адаптивною до змін на ринку. Однак, запаси також є витратою для логістичної системи, тому ефективність їх управління залежить від розумного визначення оптимального рівня запасів з економічної точки зору.

Транспорт забезпечує зв'язок між різними підсистемами логістичної системи (закупівля, склади, виробництво, збут) шляхом забезпечення безперебійного та

своєчасного транспортного процесу. Це дозволяє забезпечити ефективне функціонування системи в цілому.

Виробництво – це процес, який перетворює матеріальний потік на готову продукцію згідно з вимогами ринку, з використанням мінімальних витрат і з заданою якістю.

Розподіл – це складова логістичної системи, яка забезпечує переміщення матеріального потоку з виробництва до місць споживання через логістичні канали та ланцюжки з мінімальними витратами.

Збут – це підсистема, що інтегрується з маркетингом. Основна мета збуту - це своєчасна поставка готової продукції споживачам з відповідним логістичним сервісом у необхідному місці та часі.

Інформація, або інформаційна підсистема, є однією з основних складових логістичної системи, яка забезпечує передачу інформації між усіма підсистемами. Інформаційна підсистема виконує функцію управління та контролю, забезпечуючи потрібну комунікацію всередині системи.

Кадри – це важлива підсистема логістичної системи, яка залучається до виконання логістичних дій і забезпечує цілеспрямовану діяльність системи. Кадрова підсистема грає важливу роль у забезпеченні ефективної роботи логістичної системи [2].

2.2 Огляд існуючих рішень

2.2.1 Система «Kanban»

Головна мета логістичної системи полягає в доставці необхідної кількості та асортименту товарів і виробів до визначеного місця з максимальною готовністю для виробничого або особистого споживання при мінімальних витратах. Логістична система є єдиним планом, який узгоджує інтереси різних підрозділів підприємства з метою досягнення цілей логістики.

З початку 70-х років у Японії та інших країнах поширена система "Kanban", яка є механізмом для організації безперервного виробничого потоку, гнучкої перебудови та функціонує майже без страхових запасів. Традиційна концепція організації виробництва передбачає, що для уникнення зупинок і забезпечення безперервного потоку необхідно мати страховий запас.

Система "Kanban" дозволяє досягти ефективного використання ресурсів та уникнути надмірних запасів. Кожен етап виробничого процесу фокусується на виробництві лише тих товарів, які необхідні для наступного етапу чи безпосереднього споживача. Це сприяє зниженню витрат на утримання запасів та оптимізації логістичних процесів.

Організація виробництва за принципом "Kanban" передбачає постійну комунікацію та співпрацю між різними підрозділами підприємства. Картки "Kanban" служать інформаційними засобами, які передають сигнали про потребу у виробництві конкретних товарів чи виробів. Це сприяє гнучкості та швидкій реакції на зміни в попиті та виробничому процесі.

Однією з ключових принципів системи "Kanban" є постійне покращення та навчання. За допомогою аналізу даних, зібраних з карток "Kanban", підприємство може виявляти можливості для оптимізації процесів, зменшення часу виготовлення та збільшення якості продукції. Такий постійний саморозвиток допомагає забезпечувати конкурентоспроможність підприємства на ринку.

Переваги системи "Kanban":

1. Гнучке планування: у системі Kanban команда зосереджується на поточній роботі, а пріоритет задач встановлює менеджер. Це дозволяє команді швидко реагувати на зміни пріоритетів і адаптуватися до нових вимог.

2. Включеність команди в процес розробки: завдяки постійним зборам, прозорості процесів і можливостям самоорганізації, працівники відчувають високу включеність в роботу. Вони гуртуються як команда і проявляють щирий інтерес до досягнення спільних цілей.

3. Скорочення тривалості циклу: у системі Kanban, якщо необхідні навички є у кількох людей, тривалість виконання завдання скорочується. Якщо ж

тільки одна людина має необхідні навички, це стає вузьким місцем у процесі. Тому співробітники повинні ділитися знаннями і оптимізувати тривалість циклу. Таким чином, вся команда може взятися за завдання, яке затримувалося, і відновити плавний потік роботи.

4. Менше вузьких місць: використання системи Kanban дозволяє швидко виявляти вузькі і проблемні місця, які можуть виникнути через дефіцит уваги, людей або навичок. Завдяки відкритості та прозорості процесів, команда може швидко реагувати на такі проблеми і знаходити шляхи для їх вирішення.

5. Наочність: коли всі виконавці мають доступ до даних, вузькі місця стають більш помітними. Kanban-команди часто використовують графіки управління і сукупного потоку, які надають візуальне представлення стану роботи. Це дозволяє команді швидко виявляти та реагувати на проблеми і покращувати продуктивність.

Недоліки системи "Kanban":

1. Система погано працює з командами, що містять понад 5 осіб.
2. ЇЇ не можна використовувати для довгострокового планування [3].

2.2.2 Система «Just-In-Time»

Однією з найпоширеніших логістичних систем виробництва є система «JIT» [4], або "точно в строк". З логістичного погляду, ця система базується на простій логіці управління запасами, що дозволяє обмежити мінімальну кількість запасів. Потоки матеріальних ресурсів тісно синхронізовані з їх потребами, які визначаються графіком випуску готової продукції. Ця синхронізація вимагає координації двох основних логістичних функцій: матеріально-технічного забезпечення і управління виробництвом. Система "точно в строк" є сучасною концепцією у виробництві, постачанні і дистрибуції, яка спрямована на синхронізацію процесів доставки матеріальних ресурсів і готової продукції у потрібній кількості, в той момент, коли

вони потрібні ланкам логістики, з метою мінімізації витрат, пов'язаних зі зберіганням запасів.

Переваги логістичної системи "JIT":

1. Скорочені логістичні цикли.
2. Мінімальні запаси незавершеного виробництва та готової продукції.
3. Координація всіх виробничих процесів.
4. Стійкі зв'язки з постачальниками матеріальних ресурсів.
5. Наявність надійного транспортного обслуговування.
6. Високий рівень сервісу.
7. Ефективна інформаційна підтримка.

Недоліки системи "JIT":

1. Високі початкові витрати: впровадження JIT вимагає значних інвестицій та витрат.

2. Нездатність до управління непередбаченими обставинами: JIT вразливий до непередбачуваних ситуацій, таких як поломки обладнання, страйки працівників поставок тощо. Це може спричинити порушення у виробничому процесі та затримки у поставках.

3. Залежність від якості поставок: успішна реалізація JIT вимагає постачання високоякісних матеріалів. Якщо якість поставок знижується або надходять непридатні матеріали, це може спричинити проблеми у виробництві.

4. Необхідність стабільного виробництва: JIT працює краще в стабільних умовах, однак попит на товари часто змінюється. Це може створювати складнощі при забезпеченні попиту в потрібний момент.

5. Залежність від постачальників: успішна реалізація JIT вимагає добре налагодженої співпраці з постачальниками. Однак, деякі постачальники можуть мати складнощі з роботою в режимі JIT, що може вплинути на ефективність всього процесу.

6. Проблеми інтеграції з іншими інформаційними системами: імплементація JIT може вимагати інтеграції з іншими інформаційними системами партнерів, такими як системи управління запасами та логістики. Це може

створювати труднощі та потребувати додаткових зусиль для забезпечення сумісності систем.

2.2.3 Система «Distribution Requirement Planning»

Система планування та управління розподілом продукції DRP [5] є ефективним інструментом, який дозволяє не лише враховувати ринкову ситуацію, але й активно впливати на неї. Ця система забезпечує міцні зв'язки між постачанням, виробництвом та збутом продукції, використовуючи елементи системи планування потреб у матеріалах (МРП).

DRP є основою для планування логістичних та маркетингових функцій та їх взаємодії. Вона дозволяє прогнозувати ринкову ситуацію з певним рівнем достовірності, оптимізувати логістичні витрати шляхом скорочення транспортних витрат та витрат на переміщення товарів. DRP також дозволяє планувати поставки та управляти запасами на різних рівнях розподільчого ланцюжка, включаючи центральні та периферійні склади. Система DRP сприяє інформаційному забезпеченню різних рівнів розподільчого ланцюжка щодо ринкової ситуації. Важливою функцією DRP є планування транспортних перевезень, включаючи обробку заявок на транспортне обслуговування та коригування графіків перевезень.

DRP дозволяє вирішувати комплекс завдань:

- планування і координація логістичних та маркетингових функцій;
- прогнозування ринкової ситуації;
- планування обсягу поставок та місця розподілу товарів;
- оптимізація логістичних витрат на зберігання;
- скорочення часу поставок готової продукції.

Застосовуючи систему DRP, компанії мають можливість знижувати витрати, пов'язані зі зберіганням, транспортуванням та обробкою вантажів.

Останніми роками у США та Західній Європі поширена версія DRP-2, яка є наступним поколінням системи. Ця система використовує сучасні моделі та алгоритми для прогнозування попиту та потреби у готовій продукції, що дозволяє управляти запасами для довгострокових прогнозів попиту на готову продукцію. Вона вирішує комплексні завдання управління виробничою програмою, виробничими потужностями, персоналом, якістю транспортування та логістичного сервісу.

Переваги використання системи DRP:

1. Зменшення витрат: система DRP дозволяє фірмам знизити витрати, пов'язані зі зберіганням, транспортуванням і переробкою вантажів.

2. Оптимізація розмірів складських приміщень: ця система допомагає визначити оптимальні розміри складських площ, що сприяє ефективному використанню простору.

3. Вибір оптимального виду транспорту: система DRP дозволяє вибрати найефективніший вид транспорту для перевезення товарів.

4. Поліпшення координації логістичних функцій: вона сприяє покращенню взаємодії між логістичними функціями в дистрибуції та виробництві.

Недоліки системи DRP:

1. Потреба у точних прогнозах: система DRP потребує достеменного прогнозу і поповнення ресурсів для кожного центру і каналу розподілу готової продукції в товарорухомій мережі.

2. Використання страхових запасів: для уникнення можливих помилок при розподілі, необхідно мати визначені страхові запаси в розподільчих центрах.

3. Можливість помилок: є три види можливих помилок - помилки самого методу прогнозування, неправильне прогнозування попиту, що впливає на розміщення складів або розмір запасів, та помилки в прогнозуванні часу зміни попиту.

2.2.4 Система «Material Requirements Planning»

MRP [5] – це концепція планування потреби в матеріальних ресурсах у виробництві. Вона використовує інформацію про структуру та технологію виробництва кінцевого продукту, календарний план виробництва, дані про складські запаси, договори поставки матеріалів та комплектуючих тощо.

MRP-система – це інтегрована електронна інформаційна система управління, що реалізує концепцію MRP. Розробка MRP-систем почалася ще на початку 60-х років, і на сьогоднішній день вони присутні практично у всіх інтегрованих інформаційних системах управління підприємствами.

Основна мета MRP полягає в забезпеченні наявності будь-якої облікової одиниці ресурсів (товарів, матеріалів) у потрібний час та місце. Принципи функціонування MRP базуються на формуванні, контролі та, за необхідності, коригуванні параметрів постачання матеріалів, щоб забезпечити, що всі необхідні матеріали для виробництва будуть наявні до моменту їх споживання. Ці технології зв'язують різні підрозділи підприємства, що займаються управлінням постачанням, виробничими процесами та обслуговуванням складів.

Переваги системи MRP:

1. Ефективне планування ресурсів: MRP дозволяє підприємствам ефективно планувати свої матеріальні ресурси, враховуючи потреби виробництва. Вона допомагає визначити оптимальні обсяги замовлень, терміни поставки та розміщення матеріалів на складі, що сприяє оптимізації процесу виробництва.

2. Зниження витрат: використання MRP дозволяє зменшити витрати на складські запаси, транспортування та переробку матеріалів. Система дозволяє точно визначити потребу в матеріалах на основі обсягів виробництва і наявних запасів, що допомагає уникнути надмірного запасу і зв'язаних з ним затрат.

3. Покращення управління постачанням: MRP дозволяє підприємствам забезпечити своєчасну поставку матеріалів і комплектуючих для виробництва. Це

сприяє покращенню ланцюжка постачання, зниженню часу простою виробництва і підвищенню загальної ефективності процесу.

Недоліки системи MRP:

1. Залежність від точності даних: Ефективність MRP залежить від точності та актуальності даних, що використовуються для планування.

2. Складність впровадження: Впровадження системи MRP може бути складним і вимагати значних зусиль. Потрібно налагодити інтеграцію з існуючими системами управління, навчити персонал використовувати систему та забезпечити належну підтримку.

3. Високі вимоги до інформаційної інфраструктури: MRP потребує сучасної інформаційної інфраструктури, включаючи комп'ютеризовані системи управління, бази даних, мережі зв'язку тощо. Введення такої інфраструктури може бути затратним та часовою витратою для підприємства.

2.2.5 Система «Optimized Production Technology»

У США та інших країнах заходу з 80-х років широко застосовується оптимізована виробнича технологія (OPT) [5], розроблена вченими з Ізраїлю та США. Ця система, базуючись на ідеях систем "Канбан" та МРП, пропонує новий підхід до виробництва. Основний принцип OPT полягає у виявленні "вузьких" місць виробництва, або, як називають їх творці, критичних ресурсів. Критичними ресурсами можуть бути запаси сировини та матеріалів, машини й устаткування, технологічні процеси та персонал.

Ефективне використання критичних ресурсів визначає темпи розвитку виробничої системи, тоді як оптимізація використання некритичних ресурсів майже не має впливу на результати виробництва. Творці OPT стверджують, що втрати критичних ресурсів негативно впливають на виробництво в цілому, тоді як збереження некритичних ресурсів не приносить реальних користей з точки зору

кінцевих результатів. В середньому, для кожного виробництва потрібно 5 критичних ресурсів.

Компанії, що використовують ОРТ, не намагаються досягти 100% завантаження працівників, що займаються некритичними операціями, оскільки це може призвести до незавершених виробничих процесів та інших небажаних наслідків. Замість цього, компанії заохочують працівників використовувати резервний час для підвищення кваліфікації, участі в якісних гуртках та інших діяльностях.

Переваги системи ОРТ:

1. Підвищення ефективності виробництва: ОРТ дозволяє досягти оптимального розподілу ресурсів та забезпечити оптимальну продуктивність виробничих процесів. Вона допомагає уникнути зайвих затримок, покращити планування та збільшити загальний вихід продукції.

2. Зменшення втрат та витрат: ОРТ дозволяє оптимізувати запаси, уникнути перевиробництва та зменшити витрати на складське зберігання непотрібних матеріалів. Вона спрямована на мінімізацію втрат та оптимізацію використання ресурсів.

3. Покращення керованості: ОРТ надає комплексну інформацію про виробничі процеси, що дозволяє керівникам приймати обґрунтовані рішення. Вона забезпечує точний аналіз даних, прогнозування та планування, що сприяє кращому управлінню виробництвом.

4. Підвищення якості продукції: ОРТ допомагає виявляти та усувати можливі дефекти виробництва на ранніх стадіях. Вона спрямована на підтримку високих стандартів якості та забезпечення відповідності продукції вимогам споживачів.

Недоліки системи ОРТ:

1. Велика вартість впровадження: впровадження ОРТ може вимагати значних інвестицій у придбання та налаштування необхідного обладнання, програмного забезпечення та навчання персоналу. Це може бути фінансово вимогливим для деяких підприємств.

2. Складність впровадження та інтеграції: ОРТ є комплексною системою, що вимагає часу та зусиль для повного впровадження та інтеграції з існуючими процесами виробництва. Це може призвести до тимчасових затримок та необхідності переорганізації діяльності підприємства.

3. Залежність від точності та актуальності даних: ефективність ОРТ залежить від точності та актуальності даних, які використовуються для аналізу та прийняття рішень. Недостовірна або застаріла інформація може призвести до неточних результатів та недоцільних рішень.

4. Потреба у професійних знаннях та навичках: ефективне використання ОРТ вимагає наявності спеціалістів з розуміння системи та навичками її налаштування та управління. Це може вимагати додаткового навчання або залучення зовнішніх експертів.

2.3 Висновки до розділу

У даному розділі було зроблено огляд проблемної області – логістичної системи. Однією з головних її частин є транспорт, який виконує функцію зв’язку між всіма ланками виробництва, саме його було обрано як основу логістичної системи оптимізації виробничих технологій.

Також було зроблено огляд існуючих рішень, які представляють собою різні системи для оптимізації різних ланок логістичної системи.

3 МОДЕЛЬ ЛОГІСТИЧНОЇ СИСТЕМИ ОПТИМІЗАЦІЇ ВИРОБНИЧИХ ТЕХНОЛОГІЙ

3.1 Компонентна модель логістичної системи оптимізації виробничих технологій

Зобразимо структуру системи оптимізації виробничих технологій у вигляді діаграми компонентів (рис. 3.1).

Рисунок 3.1 – Модель логістичної системи оптимізації виробничих технологій. Діаграма компонентів у нотації UML

Компонент «Дані» - це набори даних необхідних для побудови системи.

Набір даних, який є списком замовлень orders типу dictionary, буде створено після введення користувачем кількості вершин, ребер, замовлень та вантажівок. Далі цей словник буде використано для побудови та тестування системи.

Ще один набір даних autos типу dictionary з характеристиками вантажівок, був створений для можливості задання додаткових параметрів та обмежень системи.

Компонент «Обробка даних про вантажівки» - це компонент, який слугує для відбору вантажівок, що задовольняють умови замовлення. Він складається з наступних пунктів:

- виконання жадібного алгоритму;
- повернення списку підходящих машин.

За допомогою жадібного алгоритму відбувається пошук підходящих вантажівок, які задовольняють потребу у вазі продукту в замовленні.

Компонент «Обробка даних про замовлення» призначений для оптимізації маршруту між точками в ланцюжку логістичної системи. За допомогою алгоритму Дейкстри відбувається оптимізація прокладеного маршруту від точки, де знаходиться машина до точки відправки, а також від точки відправки до місця призначення. І після виконання замовлення змінюється місцеположення вантажівки.

Компонент «Результати» слугує для відображення результатів роботи побудованої системи оптимізації виробничих технологій.

Відношення між компонентами зображується за допомогою відповідних інтерфейсів.

Інтерфейс ILDP (Interface Logistics Data Processing) – інтерфейс передання даних для побудови системи на вхід процесу обробки даних про вантажівки.

Інтерфейс ISDP (Interface System Data Processing) – інтерфейс передання отриманих даних про вантажівки на вхід процесу обробки даних про замовлення.

Інтерфейс ILP (Interface Logistics Processing) – інтерфейс передання оптимізованого маршруту по точкам логістичного ланцюжка на вхід процесу відображення результатів.

3.2 Діаграма діяльності логістичної системи оптимізації виробничих технологій

Діаграма діяльності – це візуалізація основних циклів функціонування процесів і їхньої взаємодії першого рівня. Дана діаграма зображена на рис. 3.2.

Ця діаграма показує послідовність дій у програмі для виконання замовлень. Вона зображує наступні кроки:

- початок програми;
- обирається замовлення зі списку;
- виконується жадібний алгоритм для вибору вантажівок, які підходять за вантажопідйомністю для даного замовлення;
- повертається список підходящих вантажівок;
- використовується алгоритм Дейкстри для знаходження найкоротшого шляху до пункту відправки для кожної вантажівки зі списку;
- повертається вантажівка з найкоротшим шляхом до пункту відправки;
- вантажівка призначається для виконання замовлення;
- замовлення виконується;
- машина залишається на кінцевому місці прибуття;
- перевіряється наявність наступного замовлення. Якщо таке замовлення є, програма переходить до кроку 2;
- кінець програми.

Ця діаграма демонструє послідовність кроків, які повторюються для кожного замовлення. Після виконання замовлення програма перевіряє наявність наступного замовлення і, якщо є, повертається до початку для обробки наступного замовлення.

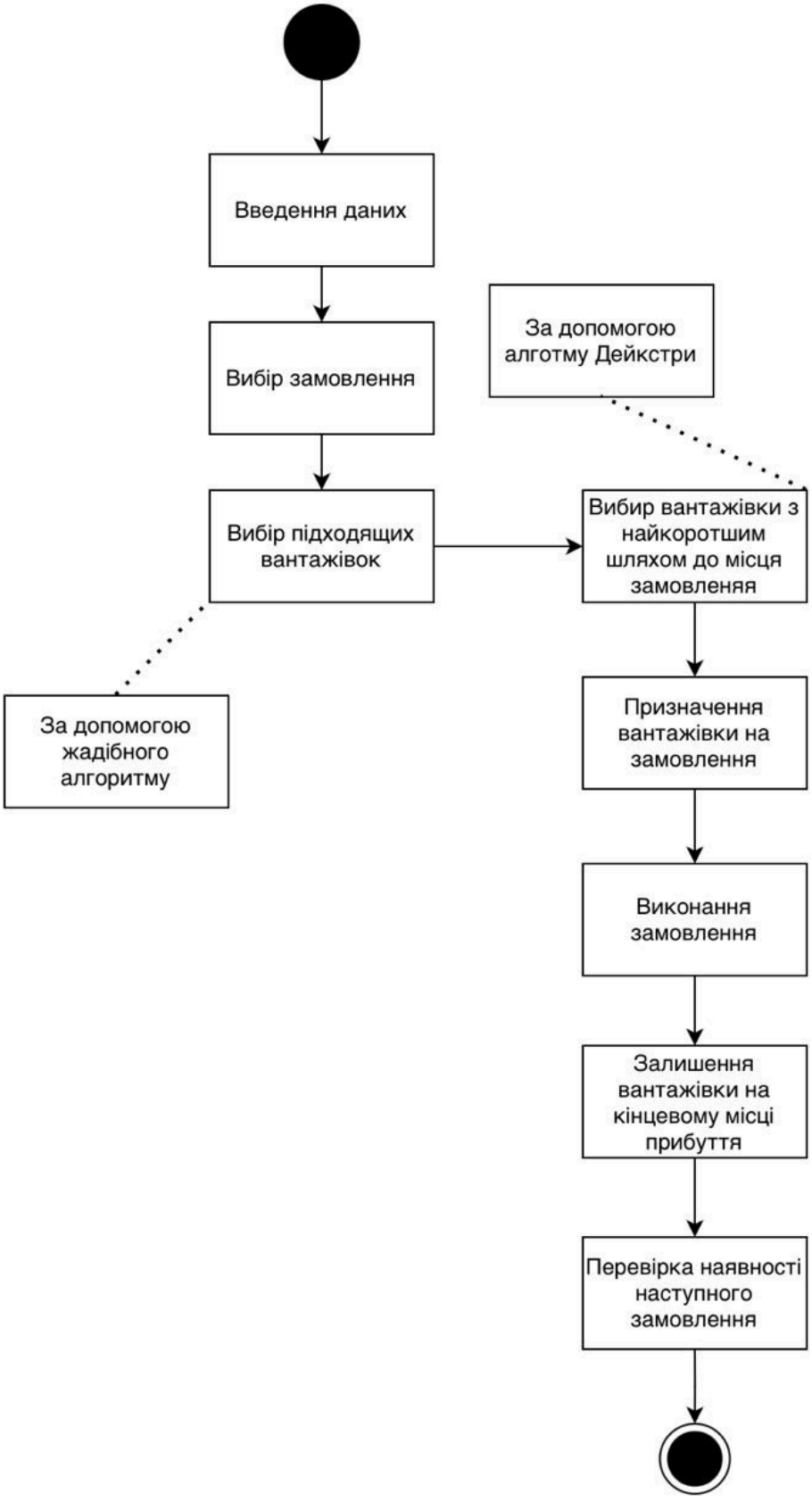

Рисунок 3.2 – Модель логістичної системи оптимізації виробничих технологій. Діаграма діяльності у нотації UML

3.3 Діаграма послідовностей логістичної системи оптимізації виробничих технологій

Для відображення послідовності повідомлень між об’єктами було побудовано діаграму послідовностей, яка зображена на рис. 3.3.

Ця діаграма демонструє послідовність дій у програмі. Опишемо більш детально кожен крок:

- запуск програми;
- програма вибирає замовлення для обробки;
- програма виконує жадібний алгоритм для вибору підходящих вантажівок для виконання даного замовлення;
- жадібний алгоритм повертає список підходящих вантажівок до програми;
- програма використовує алгоритм Дейкстри для знаходження найкоротшого шляху до пункту відправки для кожної вантажівки зі списку;
- повертається вантажівка з найкоротшим шляхом;
- програма призначає вибрану вантажівку для виконання даного замовлення;
- вантажівка підтверджує виконання замовлення програмі;
- програма перевіряє наявність інших замовлень;
- повторюється обробка замовлень;
- якщо більше немає замовлень, програма завершує роботу і припиняє виконання.

Ця діаграма показує послідовність дій, що повторюються для кожного замовлення. Коли всі замовлення виконані, програма завершує свою роботу.

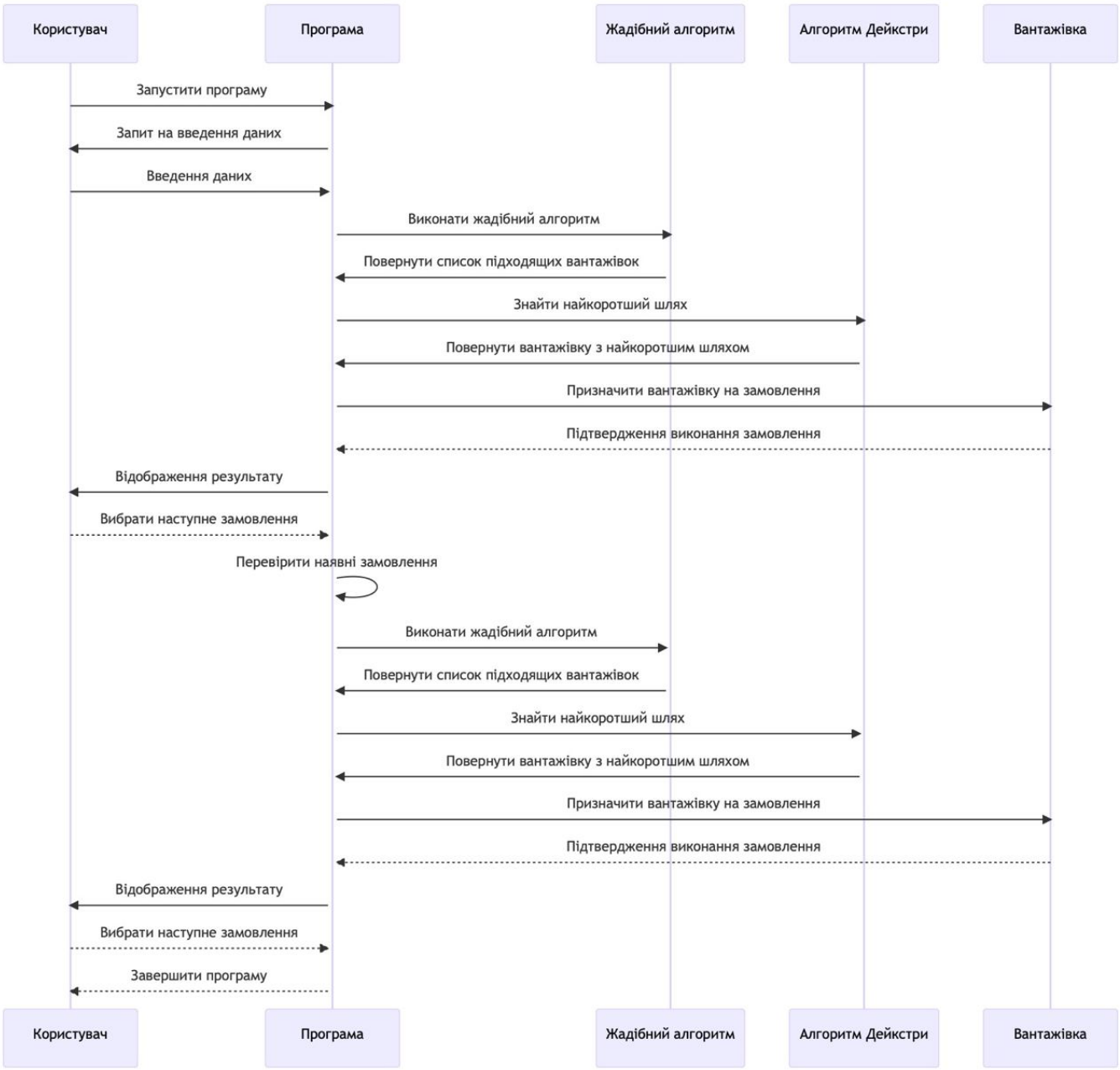

Рисунок 3.3 – Модель логістичної системи оптимізації виробничих технологій. Діаграма послідовностей у нотації UML

3.4 Висновки до розділу

У даному розділі було побудовано модель логістичної системи оптимізації виробничих технологій за допомогою діаграм. Було побудовано діаграму

компонентів для відображення компонентів програмного забезпечення та їх взаємодії, діаграму діяльності для моделювання послідовності процесів та діаграму послідовностей для зображення взаємодії повідомлень між об’єктами.

4 ОПИС МАТЕМАТИЧНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Теорія графів

Граф можна представити як геометричну фігуру, що складається з точок (вершин), що сполучені лініями (ребрами). Строге визначення можна записати наступним чином: граф – це пара множин $G = (V, E)$, де V- підмножина будь-якої зліченної множини, а $E \subset V \times V$. На рисунку 4.1 зображено найпростіший неорієнтований граф.

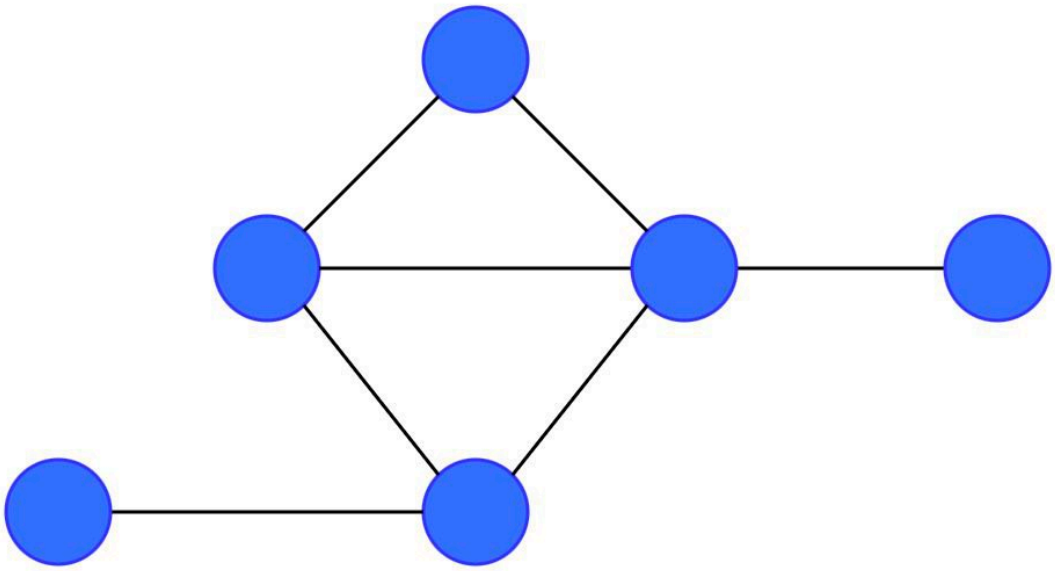

Рисунок 4.1 – Неорієнтований граф

Для виконання поставленої задачі було використано модель неорієнтованого графа. Неорієнтований граф - це граф, у якому ребра не мають напрямку або орієнтації. Він складається з множини вершин та множини неорієнтованих ребер, де кожне ребро з'єднує дві вершини без вказівки напрямку.

Неорієнтований граф може бути представлений у вигляді набору вершин V і набору неорієнтованих ребер E.

Кожне ребро $\{u, v\}$ в неорієнтованому графі вказує, що є зв'язок або зв'язок між вершинами u і v, і цей зв'язок може бути двостороннім, тобто можна переходити в обох напрямках. Неорієнтовані графи можуть мати кілька ребер, які з'єднують одну пару вершин, утворюючи кратні ребра. Також можуть існувати ребра, які з'єднують вершину з собою, утворюючи цикли [6].

В даній дипломній роботі для побудови маршруту переміщення товарів по логістичному ланцюгу було використано неорієнтований граф (рис 4.2), в якому вершини – це точки зберігання, виробництва та збуту товарів, а ребра – це час переміщення від однієї вершини до іншої.

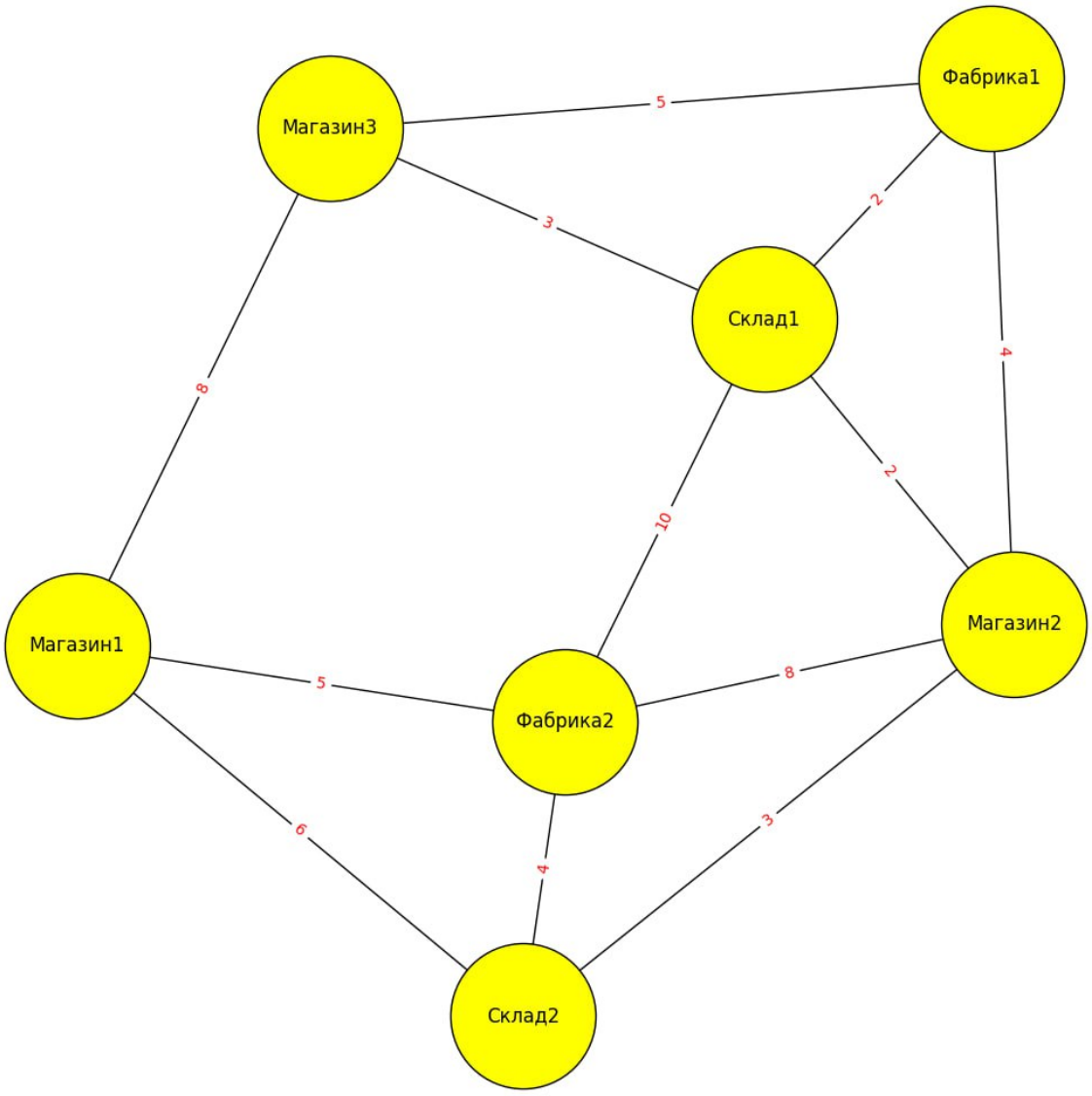

Рисунок 4.2 – Приклад неорієнтованого графа, для зображення переміщення товарів по логістичному ланцюгу

4.2 Алгоритм Дейкстри

Для оптимізації транспортних потоків використано алгоритм Дейкстри [7]. Це алгоритм на графах, який знаходить найкоротший шлях від однієї вершини до інших і працює лише з ребрами додатної довжини.

Оптимальне транспортування машини з товаром з одного місця в інше на основі алгоритму Дейкстри можна описати наступним чином.

Нехай $G = (V, E)$ – це граф, де:

- V – множина вершин, які зображують пункти, звідки і куди потрібно перевезти товари;
- E – множина ребер, які з'єднують пункти;
- $d(i, j)$ – вага (час) переходу від вершини i до вершини j;
- s – початкова вершина, звідки потрібно доставити товар;
- t – кінцева вершина, куди потрібно доставити товар.

Необхідно знайти найкоротший шлях (з найменшою вагою) від вершини s до вершини t у графі G.

Математичну модель на основі алгоритму Дейкстри можна записати наступним чином.

1. Ініціалізувати:
   - відстань до кожної вершини, крім s, як нескінченні;
   - відстань до s як 0.
2. Для кожної вершини $v \in V$:
   - якщо v не відвідана, то обчислити нову відстань $d(s, v)$ як суму відстані до вершини s та ваги ребра (s, v). Якщо нова відстань менша, ніж поточна відстань до v, оновити поточну відстань.
3. Позначити поточну вершину як s.
4. Поки не всі вершини відвідані:

- вибрати невідвідану вершину v, для якої відстань до v найменша;
- позначити v як відвідану;
- для кожного сусіда u невідвіданої вершини v: обчислити нову відстань $d(s,u)$ як суму відстані до вершини s та ваги ребра (s,u). Якщо нова відстань менша, ніж поточна відстань до u, оновити поточну відстань.

5. В результаті отримаємо найкоротший шлях від вершини s до вершини t та його вагу.

4.3 Жадібний алгоритм

Жадібний алгоритм – це евристичний алгоритм, який приймає оптимальне рішення на кожному кроці за допомогою наявних даних, сподіваючись вкінці отримати оптимальне рішення. Його перевагою є ефективність у часі виконання [8].

Математична модель жадібного алгоритму для логістичної системи оптимізації виробничих технологій може бути сформульована наступним чином.

Нехай:

- вхідними даними будуть вантажопідйомність вантажівки, її місцезнаходження, а також пункт відправки;
- визначені за допомогою алгоритму Дейкстри відстані між вантажівками та пунктом відправки;
- визначені вагові обмеження для замовлення та вантажівки;

Тоді жадібний алгоритм матиме наступний вигляд:

- пройтись по кожній машині і перевірити, чи задовольняє вона вагове обмеження. Якщо так, порівняти відстань між цією машиною та пунктом відправки з відстанню іншої машини до цього ж пункту.
- після перевірки всіх машин, вибрати машину з найкоротшим шляхом до пункту відправки.

4.4 Висновки до розділу

У даному розділі було описано математичну модель для логістичної системи оптимізації виробничих технологій. Для її побудови буде використано: теорію графів – для зображення руху товарів по логістичному ланцюгу системи, алгоритм Дейкстри для оптимізації маршруту перевезень між точками та жадібний алгоритм для вибору вантажівок, що задовольняють вантажопідйомність замовлень.

5 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

5.1 Структура програми

Для реалізації поставленої задачі було створено програмне забезпечення за схемою, що зображена на рисунку 5.1.

Рисунок 5.1 – Схема програмного забезпечення логістичної системи оптимізації виробничих технологій

На початку роботи програми відбувається введення користувачем наступних параметрів: кількість вершин (точки зберігання, виробництва та збуту товарів), кількість ребер (зв’язки між вершинами), кількість замовлень та вантажівок. Всі інші параметри обираються випадковим чином. За допомогою жадібного алгоритму обираються вантажівки, які підходять за вантажопідйомністю для виконання даного замовлення. Потім з них обирається вантажівка з найкоротшим шляхом до пункту відправки за допомогою алгоритму Дейкстри, яка і буде призначена на дане замовлення. Після виконання замовлення машина залишається на кінцевому місці прибуття до обрання її на наступне замовлення. Така послідовність дій відбувається з кожним замовленням.

5.2 Опис розроблених алгоритмів

Для даної системи був розроблений алгоритм, що зображений за допомогою схеми на рисунках 5.2 – 5.3.

Після початку виконання програми виводиться запит про введення кількості вершин count_nodes. Після цього буде відбуватися генерація вершин графа, які зображують точки зберігання, виробництва та збуту продукції (склади, фабрики та магазини). Генерація відбувається за допомогою методу random.randint( ), який генерує цілі числа в заданому проміжку.

Потім відображається запит на введення кількості ребер count_edge, відбувається перевірка на введене число. Число повинно задовольняти наступну умову: $count\_nodes \leq count\_edge \leq C^2_{count\_nodes}$, де $C^2_{count\_nodes}$ – це число ребер повного графа. Якщо count_edge не пройшло перевірку, то відображається повторний запит на введення. Після проходження перевірки, відбувається побудова ребер графа між вершинами, що були згенеровані раніше. За допомогою методу random.choice( ) по черзі обираються випадковим чином вершини зі списку і

будуються зв’язки між ними і іншими вершинами. Також за допомогою random.randint( ) генеруються ваги кожного ребра, що символізують час переміщення між вершинами.

Після отримання всіх даних відбувається ініціалізація графа, який зображує весь логістичний ланцюг.

Відбувається введення кількості замовлень та кількості вантажівок. Генерація замовлень відбувається наступним чаном: випадково зі списку обирається початкова та кінцева точка замовлення, найменування замовленого продукту та генерується вага замовлення. Дані про вантажівки заповнюються схожим чином: початкове місцеположення випадковим чином обирається зі списку вершин, а вантажопідйомність генерується за допомогою методу random.randint( ).

Потім відбувається виконання замовлення за допомогою алгоритму Дейкстри та жадібного алгоритму. Детальний опис наведено у п. 4.2 та 4.3 відповідно.

Програма завершує свою роботу після виконання всіх замовлень, виводить на екран граф побудованої логістичної системи та список замовлень, що виконані тою, чи іншою машиною з зображенням повного маршруту.

Рисунок 5.2 – Схема алгоритму програмного забезпечення для «Логістичної системи оптимізації виробничих технологій». Початок

Рисунок 5.3 – Схема алгоритму програмного забезпечення для «Логістичної системи оптимізації виробничих технологій». Кінець

5.3 Формат вихідних даних

Вихідними даними для роботи програми є кількість вершин, ребер, замовлень та вантажівок, що вводяться користувачем.

На основі введених даних генерується список замовлень orders типу dictionary, що має наступні ключі:

- «from» – початкова точка маршруту, з якої необхідно забрати товар (приклад значень: «Склад1», «Фабрика3»);
- «to» – кінцева точка маршруту, до якої необхідно доставити товар (приклад значень: «Фабрика1», «Магазин2»);
- «product» – товар, який необхідно доставити (приклад значень: «Пшеничний хліб», «»Житнє борошно);
- «weight in kg» – вага товару, який необхідно доставити (в кілограм).

Та ще один список autos типу dictionary з характеристиками вантажівок, що мітить наступні ключі:

- «from» – поточне місцезнаходження вантажівки (приклад значень: «Склад1», «Фабрика3»);
- «capacity in kg» – вантажопідйомність вантажівок (приклад значень: 1000, 850).

5.4 Формат результуючих даних

Результат роботи програми – це розподіл замовлень між різними точками виробничого ланцюга, з вибором оптимального маршруту та часом його виконання.

Ці дані буде зображено у вигляді списку з номером замовлення; машиною, яка виконує замовлення; з повним ланцюгом маршруту (які вершини проходить вантажівка від початкової точки до місця призначення) та часом його виконання (вагами ребер).

5.5 Результати випробування

5.5.1 Приклад тестування програми №1

Для тестування програми було введене наступні параметри:

- кількість вершин 11;
- кількість ребер 15;
- кількість замовлень 14;
- кількість машин 9.

Було отримано згенеровані дані, що представлено в таблиці 5.1.

Таблиця 5.1 – Згенеровані дані №1 для тестування програми

№	from	to	product	weight in kg
1	Фабрика0	Склад8	Пшеничний хліб	621
2	Фабрика3	Фабрика6	Вівсяне борошно	767
3	Фабрика0	Фабрика11	Житнє борошно	452
4	Фабрика1	Склад8	Житній хліб	584
5	Фабрика11	Фабрика0	Житнє борошно	2392
6	Фабрика3	Склад8	Вівсяний хліб	1751
7	Фабрика1	Фабрика11	Вівсяний хліб	1711
8	Фабрика1	Склад4	Житній хліб	2170

Продовження таблиці 5.1

9	Фабрика5	Фабрика6	Пшеничний хліб	211
10	Фабрика11	Фабрика5	Пшеничний хліб	1829
11	Магазин8	Фабрика0	Пшеничне борошно	2179
12	Склад8	Фабрика1	Житнє борошно	2407
13	Фабрика0	Фабрика4	Вівсяне борошно	321
14	Фабрика7	Фабрика0	Пшеничний хліб	871

Вантажопідйомність кожної вантажівки та початкове місце перебування наведено в таблиці 5.2.

Таблиця 5.2 – Дані про вантажівки для прикладу №1

Номер вантажівки	Вантажопідйомність	Початкова точка
1	3841	Фабрика1
2	2532	Фабрика11
3	2454	Магазин8
4	2948	Фабрика0
5	1369	Фабрика7
6	1263	Фабрика3
7	3793	Склад4
8	2104	Фабрика5
9	4080	Фабрика1

Результатом роботи програми є оптимізовані перевезення товарів згідно кожного замовлення за оптимальними маршрутами, враховуючи вантажопідйомність кожної вантажівки та їх розміщення.

Приклад роботи програми зображено на наступних рисунках в порядку виконання замовлень.

```
Машина №3 виконує замовлення:
       from      to         product  weight in kg
0  Фабрика0  Склад8  Пшеничний хліб           621
За маршрутом з часом 9 год.
Фабрика0 -> Склад8
```

Рисунок 5.4 – Виконання замовлення №1 для прикладу №1

```
Машина №5 виконує замовлення:
       from        to          product  weight in kg
1  Фабрика3  Фабрика6  Вівсяне борошно           767
За маршрутом з часом 18 год.
Фабрика3 -> Фабрика5 -> Фабрика11 -> Фабрика6
```

Рисунок 5.5 – Виконання замовлення №2 для прикладу №1

```
Машина №3 виконує замовлення:
       from         to        product  weight in kg
2  Фабрика0  Фабрика11  Житнє борошно           452
За маршрутом з часом 23 год.
Фабрика0 -> Фабрика3 -> Фабрика5 -> Фабрика11
```

Рисунок 5.6 – Виконання замовлення №3 для прикладу №1

```
Машина №1 виконує замовлення:
       from      to      product  weight in kg
3  Фабрика1  Склад8  Житній хліб           584
За маршрутом з часом 24 год.
Фабрика1 -> Фабрика7 -> Склад8
```

Рисунок 5.7 – Виконання замовлення №4 для прикладу №1

```
Машина №1 виконує замовлення:
        from        to        product  weight in kg
4  Фабрика11  Фабрика0  Житнє борошно          2392
За маршрутом з часом 23 год.
Фабрика11 -> Фабрика5 -> Фабрика3 -> Фабрика0
```

Рисунок 5.8 – Виконання замовлення №5 для прикладу №1

```
Машина №4 виконує замовлення:
       from      to        product  weight in kg
5  Фабрика3  Склад8  Вівсяний хліб          1751
За маршрутом з часом 10 год.
Фабрика3 -> Склад4 -> Склад8
```

Рисунок 5.9 – Виконання замовлення №6 для прикладу №1

```
Машина №8 виконує замовлення:
       from         to        product  weight in kg
6  Фабрика1  Фабрика11  Вівсяний хліб          1711
За маршрутом з часом 23 год.
Фабрика1 -> Фабрика4 -> Фабрика6 -> Фабрика11
```

Рисунок 5.10 – Виконання замовлення №7 для прикладу №1

```
Машина №7 виконує замовлення:
       from      to      product  weight in kg
7  Фабрика1  Склад4  Житній хліб          2170
За маршрутом з часом 29 год.
Фабрика1 -> Фабрика7 -> Склад8 -> Склад4
```

Рисунок 5.11 – Виконання замовлення №8 для прикладу №1

```
Машина №5 виконує замовлення:
       from        to         product  weight in kg
8  Фабрика5  Фабрика6  Пшеничний хліб           211
За маршрутом з часом 12 год.
Фабрика5 -> Фабрика11 -> Фабрика6
```

Рисунок 5.12 – Виконання замовлення №7 для прикладу №1

```
Машина №1 виконує замовлення:
        from        to         product  weight in kg
9  Фабрика11  Фабрика5  Пшеничний хліб          1829
За маршрутом з часом 7 год.
Фабрика11 -> Фабрика5
```

Рисунок 5.13 – Виконання замовлення №8 для прикладу №1

```
Машина №1 виконує замовлення:
        from        to           product  weight in kg
10  Магазин8  Фабрика0  Пшеничне борошно          2179
За маршрутом з часом 30 год.
Магазин8 -> Фабрика5 -> Фабрика3 -> Фабрика0
```

Рисунок 5.14 – Виконання замовлення №9 для прикладу №1

```
Машина №3 виконує замовлення:
      from        to        product  weight in kg
11  Склад8  Фабрика1  Житнє борошно          2407
За маршрутом з часом 24 год.
Склад8 -> Фабрика7 -> Фабрика1
```

Рисунок 5.15 – Виконання замовлення №10 для прикладу №1

```
Машина №1 виконує замовлення:
        from        to          product  weight in kg
12  Фабрика0  Фабрика4  Вівсяне борошно           321
За маршрутом з часом 38 год.
Фабрика0 -> Фабрика3 -> Фабрика5 -> Магазин8 -> Фабрика4
```

Рисунок 5.16 – Виконання замовлення №11 для прикладу №1

```
Машина №2 виконує замовлення:
        from        to         product  weight in kg
13  Фабрика7  Фабрика0  Пшеничний хліб           871
За маршрутом з часом 23 год.
Фабрика7 -> Склад8 -> Фабрика0
```

Рисунок 5.17 – Виконання замовлення №12 для прикладу №1

На рисунку 5.18 зображено граф для прикладу №1, який відображає схему по якій переміщується товар в логістичному ланцюжку.

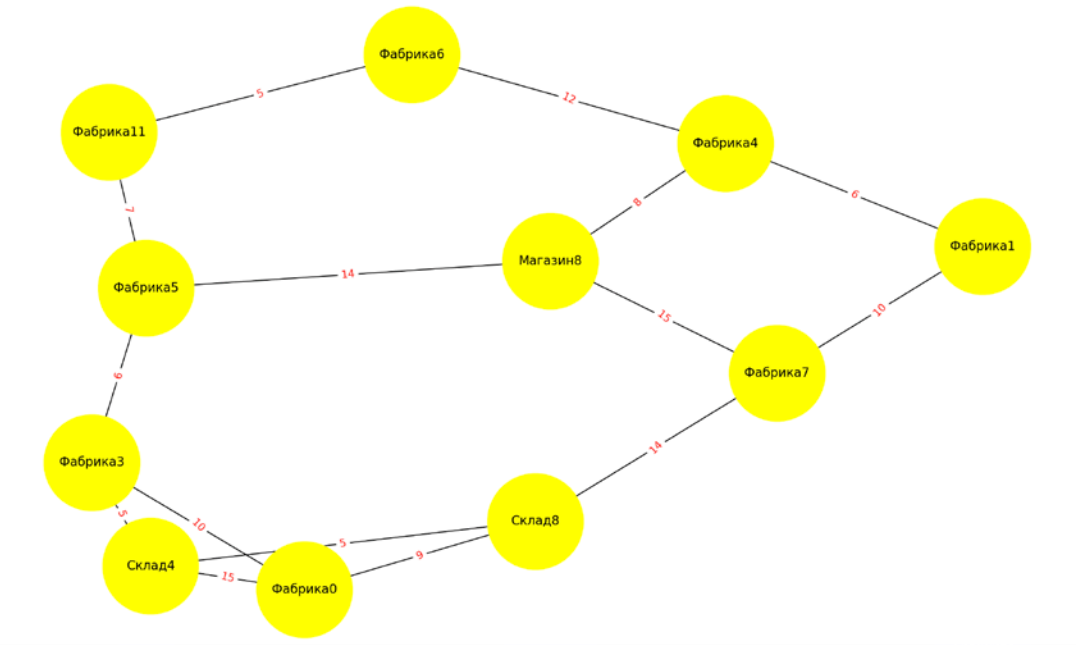

Рисунок 5.18 – Граф, який зображує схему, по якій переміщується товар для прикладу №1

5.5.2 Приклад тестування програми №2

Для тестування програми було введене наступні параметри:

- кількість вершин 15;
- кількість ребер 24;
- кількість замовлень 14;
- кількість машин 5.

Було отримано згенеровані дані, що представлено в таблиці 5.3.

Таблиця 5.3 – Згенеровані дані №2 для тестування програми

№	from	to	product	weight in kg
1	Склад15	Фабрика15	Пшеничне борошно	2000
2	Фабрика13	Склад0	Житнє борошно	593
3	Магазин11	Фабрика13	Житнє борошно	1987
4	Склад3	Фабрика15	Вівсяне борошно	1644
5	Магазин8	Фабрика13	Житнє борошно	1185
6	Магазин3	Фабрика0	Вівсяне борошно	2435
7	Склад15	Магазин11	Вівсяний хліб	1531
8	Магазин11	Склад0	Вівсяне борошно	1903
9	Склад10	Фабрика15	Пшеничне борошно	1211
10	Фабрика15	Фабрика5	Пшеничний хліб	763
11	Фабрика0	Склад15	Вівсяний хліб	141
12	Магазин11	Склад3	Житнє борошно	746
13	Магазин8	Фабрика15	Пшеничне борошно	575
14	Склад8	3	Пшеничний хліб	2272

Вантажопідйомність кожної вантажівки та початкове місце перебування наведено в таблиці 5.4.

Таблиця 5.4 – Дані про вантажівки для прикладу №2

Номер вантажівки	Вантажопідйомність	Початкова точка
1	2311	Магазин8
2	3557	Магазин11
3	4320	Магазин8
4	4873	Магазин11
5	1328	Склад8

Результатом роботи програми є оптимізовані перевезення товарів згідно кожного замовлення за оптимальними маршрутами, враховуючи вантажопідйомність кожної вантажівки та їх розміщення.

Приклад роботи програми зображено на наступних рисунках в порядку виконання замовлень.

```
Машина №1 виконує замовлення:
      from         to           product  weight in kg
0  Склад15  Фабрика15  Пшеничне борошно          2000
За маршрутом з часом 20 год.
Склад15 -> Фабрика14 -> Фабрика5 -> Фабрика15
```

Рисунок 5.19 – Виконання замовлення №1 для прикладу №2

```
Машина №2 виконує замовлення:
        from      to        product  weight in kg
1  Фабрика13  Склад0  Житнє борошно           593
За маршрутом з часом 12 год.
Фабрика13 -> Фабрика5 -> Склад0
```

Рисунок 5.20 – Виконання замовлення №2 для прикладу №2

```
Машина №3 виконує замовлення:
        from         to        product  weight in kg
2  Магазин11  Фабрика13  Житнє борошно          1987
За маршрутом з часом 21 год.
Магазин11 -> Склад0 -> Фабрика5 -> Фабрика13
```

Рисунок 5.21 – Виконання замовлення №3 для прикладу №2

```
Машина №2 виконує замовлення:
     from         to          product  weight in kg
3  Склад3  Фабрика15  Вівсяне борошно          1644
За маршрутом з часом 13 год.
Склад3 -> Склад10 -> Фабрика5 -> Фабрика15
```

Рисунок 5.22 – Виконання замовлення №4 для прикладу №2

```
Машина №2 виконує замовлення:
       from         to        product  weight in kg
4  Магазин8  Фабрика13  Житнє борошно          1185
За маршрутом з часом 3 год.
Магазин8 -> Фабрика13
```

Рисунок 5.23 – Виконання замовлення №5 для прикладу №2

```
Машина №2 виконує замовлення:
       from        to          product  weight in kg
5  Магазин3  Фабрика0  Вівсяне борошно          2435
За маршрутом з часом 18 год.
Магазин3 -> Склад10 -> Фабрика0
```

Рисунок 5.24 – Виконання замовлення №6 для прикладу №2

```
Машина №2 виконує замовлення:
      from         to        product  weight in kg
6  Склад15  Магазин11  Вівсяний хліб          1531
За маршрутом з часом 26 год.
Склад15 -> Фабрика14 -> Склад0 -> Магазин11
```

Рисунок 5.25 – Виконання замовлення №7 для прикладу №2

```
Машина №1 виконує замовлення:
        from      to          product  weight in kg
7  Магазин11  Склад0  Вівсяне борошно          1903
За маршрутом з часом 9 год.
Магазин11 -> Склад0
```

Рисунок 5.26 – Виконання замовлення №8 для прикладу №2

```
Машина №2 виконує замовлення:
      from         to           product  weight in kg
8  Склад10  Фабрика15  Пшеничне борошно          1211
За маршрутом з часом 10 год.
Склад10 -> Фабрика5 -> Фабрика15
```

Рисунок 5.27 – Виконання замовлення №9 для прикладу №2

```
Машина №1 виконує замовлення:
        from        to         product  weight in kg
9  Фабрика15  Фабрика5  Пшеничний хліб           763
За маршрутом з часом 4 год.
Фабрика15 -> Фабрика5
```

Рисунок 5.28 – Виконання замовлення №10 для прикладу №2

```
Машина №4 виконує замовлення:
        from       to        product  weight in kg
10  Фабрика0  Склад15  Вівсяний хліб           141
За маршрутом з часом 24 год.
Фабрика0 -> Склад10 -> Магазин8 -> Склад15
```

Рисунок 5.29 – Виконання замовлення №11 для прикладу №2

```
Машина №1 виконує замовлення:
         from      to        product  weight in kg
11  Магазин11  Склад3  Житнє борошно           746
За маршрутом з часом 25 год.
Магазин11 -> Склад0 -> Фабрика5 -> Склад10 -> Склад3
```

Рисунок 5.30 – Виконання замовлення №12 для прикладу №2

```
Машина №2 виконує замовлення:
        from         to           product  weight in kg
12  Магазин8  Фабрика15  Пшеничне борошно           575
За маршрутом з часом 12 год.
Магазин8 -> Фабрика13 -> Фабрика5 -> Фабрика15
```

Рисунок 5.31 – Виконання замовлення №13 для прикладу №2

```
Машина №2 виконує замовлення:
      from         to         product  weight in kg
13  Склад8  Фабрика13  Пшеничний хліб          2272
За маршрутом з часом 17 год.
Склад8 -> Магазин8 -> Фабрика13
```

Рисунок 5.32 – Виконання замовлення №14 для прикладу №2

На рисунку 5.33 зображено граф для прикладу №2, який відображає схему по якій переміщується товар в логістичному ланцюжку.

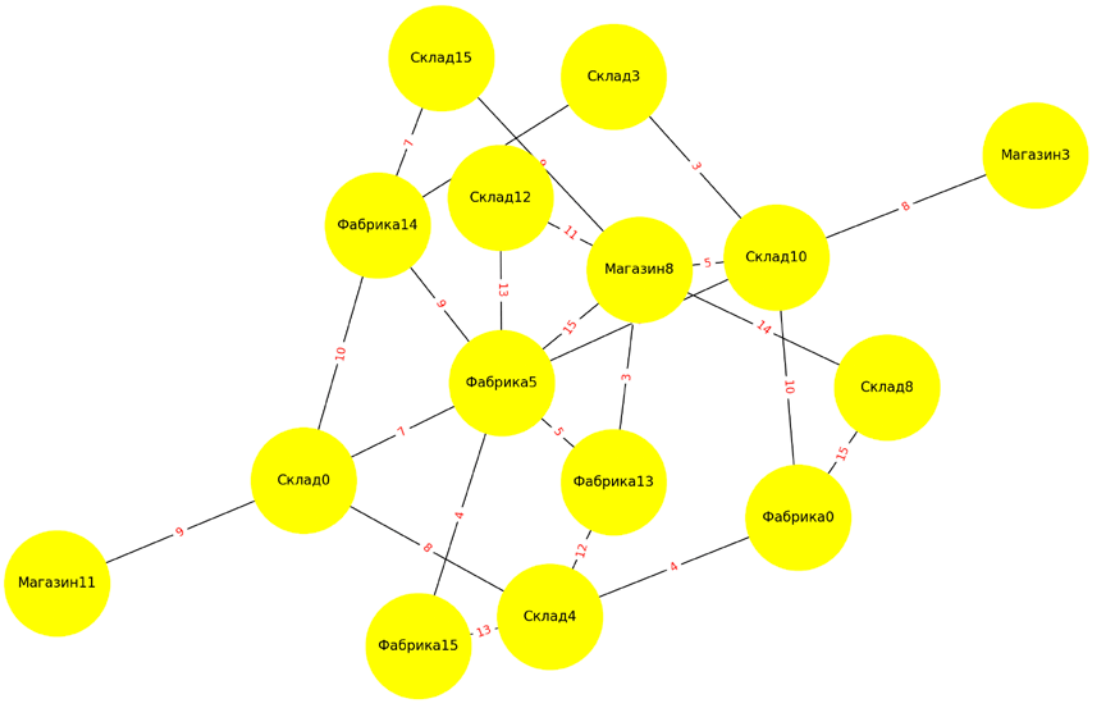

Рисунок 5.33 – Граф, який зображує схему, по якій переміщується товар для прикладу №2

5.5.3 Приклад тестування програми №3

Для тестування програми було введене наступні параметри:

- кількість вершин 5;
- кількість ребер 9;
- кількість замовлень 7;
- кількість машин 4.

Було отримано згенеровані дані, що представлено в таблиці 5.3.

Таблиця 5.5 – Згенеровані дані №3 для тестування програми

№	from	to	product	weight in kg
1	Склад0	Фабрика1	Вівсяне борошно	2081
2	Фабрика1	Фабрика0	Пшеничне борошно	2402
3	Фабрика1	Склад3	Пшеничне борошно	1694
4	Фабрика0	Склад0	Пшеничний хліб	1334
5	Фабрика0	Фабрика1	Вівсяне борошно	1092
6	Фабрика1	Фабрика0	Вівсяне борошно	910
7	Фабрика5	Склад3	Вівсяний хліб	704

Вантажопідйомність кожної вантажівки та початкове місце перебування наведено в таблиці 5.6.

Таблиця 5.6 – Дані про вантажівки для прикладу №3

Номер вантажівки	Вантажопідйомність	Початкова точка
1	3438	Фабрика5
2	859	Фабрика1
3	3012	Фабрика0

Продовження таблиці 5.6

4	2098	Склад0

Приклад роботи програми зображено на наступних рисунках в порядку виконання замовлень.

```
Машина №3 виконує замовлення:
     from        to          product  weight in kg
0  Склад0  Фабрика1  Вівсяне борошно          2081
За маршрутом з часом 9 год.
Склад0 -> Фабрика1
```

Рисунок 5.34 – Виконання замовлення №1 для прикладу №3

```
Машина №1 виконує замовлення:
       from        to           product  weight in kg
1  Фабрика1  Фабрика0  Пшеничне борошно          2402
За маршрутом з часом 7 год.
Фабрика1 -> Фабрика0
```

Рисунок 5.35 – Виконання замовлення №2 для прикладу №3

```
Машина №1 виконує замовлення:
       from      to           product  weight in kg
2  Фабрика1  Склад3  Пшеничне борошно          1694
За маршрутом з часом 10 год.
Фабрика1 -> Склад3
```

Рисунок 5.36 – Виконання замовлення №3 для прикладу №3

```
Машина №3 виконує замовлення:
       from      to         product  weight in kg
3  Фабрика0  Склад0  Пшеничний хліб          1334
За маршрутом з часом 6 год.
Фабрика0 -> Склад0
```

Рисунок 5.37 – Виконання замовлення №4 для прикладу №3

```
Машина №1 виконує замовлення:
       from        to          product  weight in kg
4  Фабрика0  Фабрика1  Вівсяне борошно          1092
За маршрутом з часом 7 год.
Фабрика0 -> Фабрика1
```

Рисунок 5.38 – Виконання замовлення №5 для прикладу №3

```
Машина №1 виконує замовлення:
       from        to          product  weight in kg
5  Фабрика1  Фабрика0  Вівсяне борошно           910
За маршрутом з часом 7 год.
Фабрика1 -> Фабрика0
```

Рисунок 5.39 – Виконання замовлення №6 для прикладу №3

```
Машина №1 виконує замовлення:
       from      to        product  weight in kg
6  Фабрика5  Склад3  Вівсяний хліб           704
За маршрутом з часом 14 год.
Фабрика5 -> Фабрика1 -> Склад3
```

Рисунок 5.40 – Виконання замовлення №7 для прикладу №3

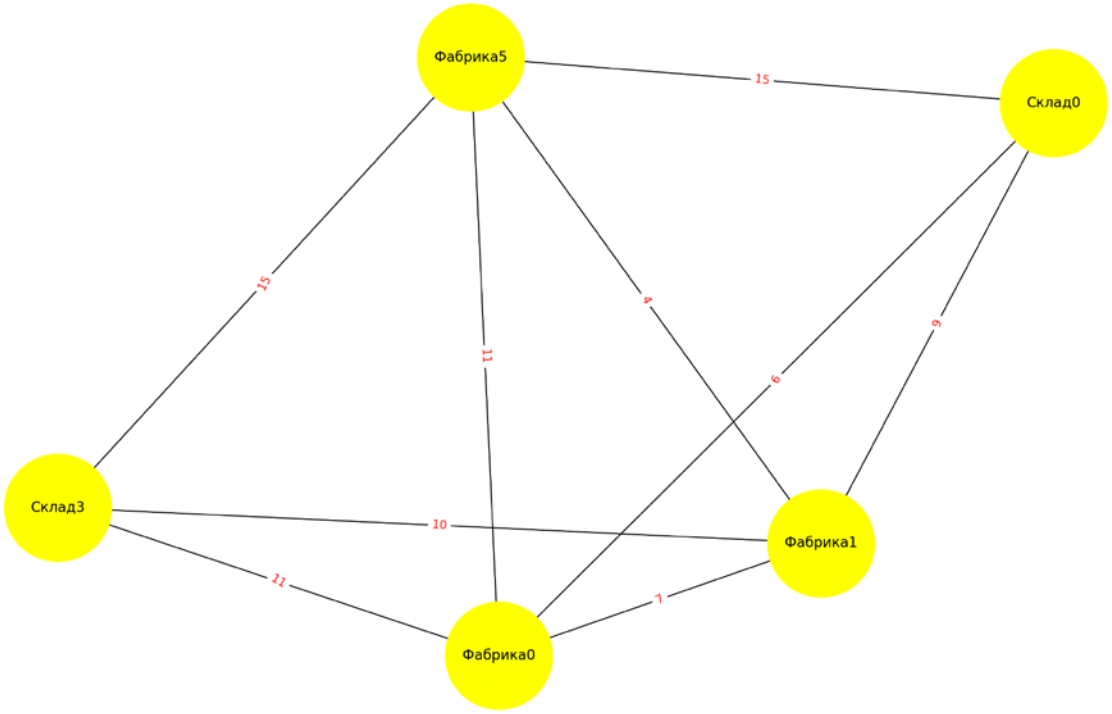

Рисунок 5.41 – Граф, який зображує схему, по якій переміщується товар для прикладу №3

4.6 Висновки до розділу

У цьому розділі було розроблено програмне забезпечення для логістичної системи оптимізації виробничих технологій.

Реалізовано тестування програми на основі власноруч створеного списку необхідних переміщень товарів по логістичному ланцюгу. Результати роботи програми наведено в п. 4.5.

ВИСНОВКИ

Під час виконання дипломної роботи на тему «Логістична система оптимізації виробничих технологій» було:

1. розглянуто проблемну область (логістичну систему);
2. проведено аналіз існуючих рішень (система Kanban, логістична система «точно в строк», логістична система планування потреб у розподілі, концепція планування потреби в матеріальних ресурсах у виробництві, система планування потреби в матеріальних ресурсах та оптимізована виробнича технологія).
3. побудовано модель логістичної системи оптимізації виробничих технологій за допомогою діаграм компонентів, діяльності та послідовностей;
4. описано математичну модель за допомогою якої була побудована система (для її побудови було використано теорію графів, алгоритм Дейкстри та жадібний алгоритм);
5. реалізовано програмне забезпечення логістичної системи оптимізації виробничих технологій;
6. проведено тестування розробленої системи.

ПЕРЕЛІК ПОСИЛАНЬ

1. Кальченко А. Г. Логістика : Підручник / А. Г. Кальченко. – Київ : КНЕУ, 2003. – 284 с.

2. Крикавський Є. Логістика та управління ланцюгами поставок / Є. Крикавський, О. Похильченко, М. Фертч. – Львів : Львів. політехніка, 2017. – 801 с.

3. Ohno T. Toyota Production System [Electronic resource] / Taiichi Ohno, Norman Bodek. – [S. l.] : Productivity Press, 2019.

4. Атамас П. Система «якраз вчасно» та її вплив на облік : навч. посіб. / П. Атамас. – Київ : ЦНЛ, 2006. – 440 с.

5. Гаврилов Д. А. Управление производством на базе стандарта MRP II / Д. А. Гаврилов. – Санкт-Петербур : Питер, 2002. – 320 с.

6. Басакер Р., Сааті Т. Кінцеві графи та мережі / Р. Басакер, Т. Сааті // М.: Наука, 1974. – 368с.

7. Кормен Т. Х. Алгоритми для роботи з графами : підручник / Т. Х. Кормен. – 2-ге вид. – М. : Вільямс, 2006. – 1296 с.

8. Кормен Т. Вступ в алгоритми / Т. Кормен, Ч. Лейзерсон, Р. Рівест, К. Стайн // К.: К.І.С., 2019. – 1288 с.

Додаток А

Лістинг програми

```
import sys
import pandas as pd
import copy
import random
import math
import networkx as nx
import matplotlib.pyplot as plt

class Graph(object):
    def __init__(self, nodes, init_graph):
        self.nodes = nodes
        self.graph = self.construct_graph(nodes, init_graph)

    def construct_graph(self, nodes, init_graph):
        graph = {}
        for node in nodes:
            graph[node] = {}

        graph.update(init_graph)

        for node, edges in graph.items():
            for adjacent_node, value in edges.items():
                if graph[adjacent_node].get(node, False) == False:
                    graph[adjacent_node][node] = value

        return graph

    def get_nodes(self):
        return self.nodes  # Повертає вузли графа

    def get_outgoing_edges(self, node):
        connections = []
        for out_node in self.nodes:
            if self.graph[node].get(out_node, False) != False:
                connections.append(out_node)
        return connections  # Повертає сусідів вузла

    def value(self, node1, node2):
        return self.graph[node1][node2] # Повертає значення ребер між вузлами

def dijkstra_algorithm(graph, start_node):
    unvisited_nodes = list(graph.get_nodes())

    shortest_path = {}

    previous_nodes = {}

    max_value = sys.maxsize
    for node in unvisited_nodes:
        shortest_path[node] = max_value
    shortest_path[start_node] = 0

    while unvisited_nodes:
        current_min_node = None
        for node in unvisited_nodes:  # Ітеруємося по вузлах
            if current_min_node == None:
                current_min_node = node
            elif shortest_path[node] < shortest_path[current_min_node]:
                current_min_node = node

        neighbors = graph.get_outgoing_edges(current_min_node)
        for neighbor in neighbors:
            tentative_value = shortest_path[current_min_node] + graph.value(current_min_node, neighbor)
            if tentative_value < shortest_path[neighbor]:
                shortest_path[neighbor] = tentative_value
                # Оновлюємо найкращий шлях до поточного вузла
                previous_nodes[neighbor] = current_min_node
```

```
        unvisited_nodes.remove(current_min_node)

    return previous_nodes, shortest_path

def print_result(previous_nodes, shortest_path, start_node, target_node):
    path = []
    node = target_node

    while node != start_node:
        path.append(node)
        node = previous_nodes[node]

    path.append(start_node)

    path = list(reversed(path))
    return path, shortest_path[target_node]

def appiontment(autos, orders, graph):
    orders_copy = copy.copy(orders)
    capacity_auto = []
    for j in autos['capacity in kg']:
        capacity_auto.append(j)

    capacity_order = []
    for j in orders['weight in kg']:
        capacity_order.append(j)

    k = 0
    q = 0
    for j in orders['from']:
        path_lst = []
        short_lst = []
        count = 0
        capacity_bool = []
        for i in autos['from']:
            if capacity_order[q] <= capacity_auto[count]:
                previous_nodes, shortest_path = dijkstra_algorithm(graph=graph, start_node=i)
                path, short = print_result(previous_nodes, shortest_path, start_node=i, target_node=j)
                path_lst.append(path)
                short_lst.append(short)
                capacity_bool.append(True)
            else:
                capacity_bool.append(False)
            count += 1
        index = short_lst.index(min(short_lst))
        order = orders.iloc[[q]]
        for i in range(len(capacity_bool)):
            if capacity_bool[i]:
                index -= 1
            if True not in capacity_bool[:i+1]:
                continue
            if index == 0 or index == -1:
                print("Машина №{}".format(i+1), "виконує замовлення:")
                print(order)
                previous_nodes, shortest_path = dijkstra_algorithm(graph=graph, start_node=order['from'].values[0])
                path, short = print_result(previous_nodes, shortest_path, start_node=order['from'].values[0], target_node=order['to'].values[0])
                print("За маршрутом з часом {} год.".format(short))
                print(" -> ".join(path))
                print("\n")
                autos['from'][i] = path[-1]
                break

        q += 1

count_nodes = int(input("Введіть кількість вершин: "))
count_sfm = [[], [], []]
name_node = ["Склад", "Фабрика", "Магазин"]
nodes = []
for i in range(count_nodes):
    name = random.choice(name_node)
    if name == "Склад":
        num = random.randint(0, count_nodes)
        while num in count_sfm[0]:
            num = random.randint(0, count_nodes)
        count_sfm[0].append(num)
    if name == "Фабрика":
        num = random.randint(0, count_nodes)
        while num in count_sfm[1]:
            num = random.randint(0, count_nodes)
```

```
        count_sfm[1].append(num)
    if name == "Магазин":
        num = random.randint(0, count_nodes)
        while num in count_sfm[2]:
            num = random.randint(0, count_nodes)
        count_sfm[2].append(num)

for i in count_sfm[0]:
    nodes.append("Склад" + str(i))
for i in count_sfm[1]:
    nodes.append("Фабрика" + str(i))
for i in count_sfm[2]:
    nodes.append("Магазин" + str(i))

count_edge = int(input("Введіть кількість ребер: "))
while count_edge < count_nodes:
    count_edge = int(input("Ви ввели замалу кількість ребер, введіть ще раз: "))
while count_edge > (math.comb(count_nodes, 2)):
    count_edge = int(input("Ви ввели завелику кількість ребер, введіть ще раз: "))
edges = []
node1 = ""
node2 = ""
for i in range(count_edge):
    node1 = random.choice(nodes)
    node2 = random.choice(nodes)
    while node1 == node2:
        node2 = random.choice(nodes)
    while (node1, node2) in edges or (node2, node1) in edges:
        node1 = random.choice(nodes)
        node2 = random.choice(nodes)
        while node1 == node2:
            node2 = random.choice(nodes)
    edges.append((node1, node2))

weights_edges = [random.randint(3, 15) for i in range(count_edge)]

init_graph = {}
for node in nodes:
    init_graph[node] = {}

k = 0
for i in edges:
    init_graph[i[0]][i[1]] = weights_edges[k]
    k += 1

count_orders = int(input("Введіть кількість замовлень "))
from_orders = []
to_orders = []
for i in range(count_orders):
    from_order = random.choice(nodes)
    to_order = random.choice(nodes)
    while from_order == to_order:
        to_order = random.choice(nodes)
    from_orders.append(from_order)
    to_orders.append(to_order)

products = ["Пшеничне борошно", "Пшеничний хліб",  "Житнє борошно", "Житній хліб", "Вівсяне борошно", "Вівсяний хліб"]

orders = {
        'from': from_orders,
        'to': to_orders,
        'product': [random.choice(products) for i in range(count_orders)],
        'weight in kg': [random.randint(10, 2500) for i in range(count_orders)]
}

orders_df = pd.DataFrame(data=orders, index=[i for i in range(count_orders)])
print(orders_df)

count_autos = int(input("\nВведіть кількість машин "))
autos = {'from': [random.choice(nodes) for i in range(count_autos)], 'capacity in kg': [random.randint(500, 5000) for i in range(count_autos)]}

autos_df = pd.DataFrame(data=autos)
print(autos_df)

print("\n")
graph = Graph(nodes, init_graph)
appiontment(autos, orders_df, graph)

G = nx.Graph(directed=True)
```

```
fig = plt.figure(figsize=(15, 15))

for i in nodes:
    G.add_node(i)

options = {
    'node_color': 'yellow',
    'node_size': 8500,
    'width': 1,
    'arrowstyle': '-',
    'arrowsize': 18,
}

for i in edges:
    G.add_edge(i[0], i[1])

pos = nx.spring_layout(G)
nx.draw(G, pos, with_labels = True, arrows=True, **options)

k = 0
for i in edges:
    nx.draw_networkx_edge_labels(
        G, pos,
        edge_labels={
            (i[0], i[1]): weights_edges[k]
        },
        font_color='red'
    )
    k += 1

plt.show()
```

Додаток Б

Ілюстративний матеріал

Рисунок Б.1 – Слайд 1

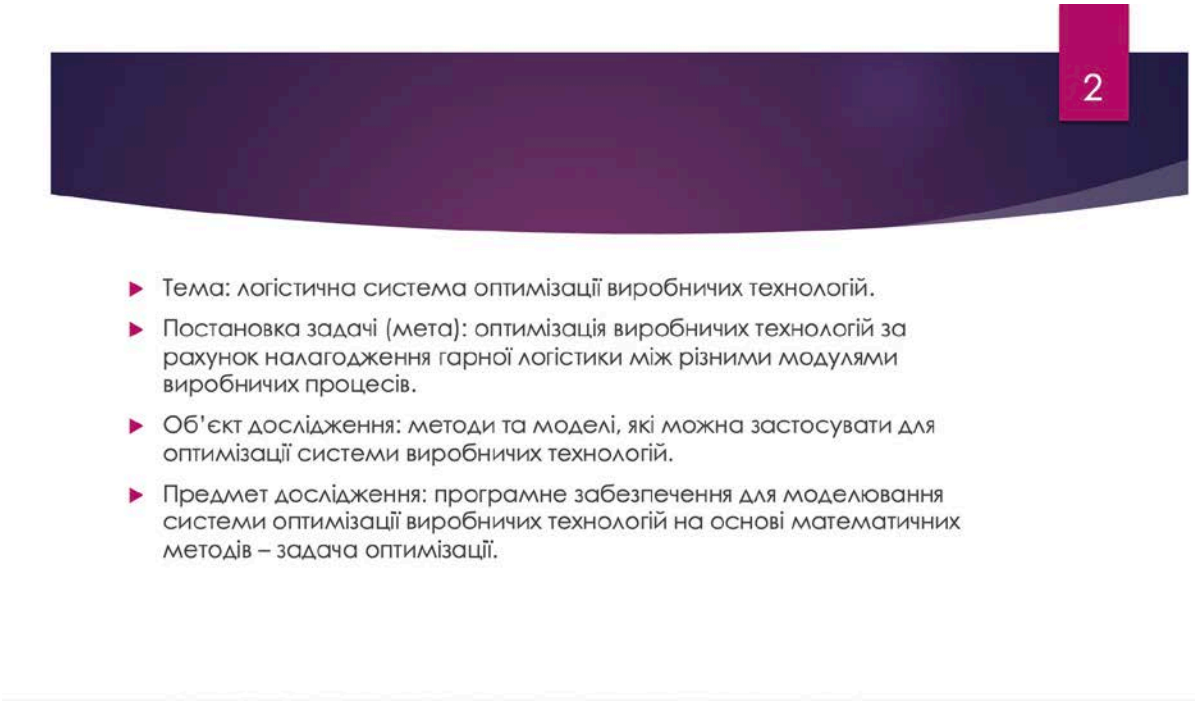

Рисунок Б.2 – Слайд 2

Рисунок Б.3 – Слайд 3

Рисунок Б.4 – Слайд 4

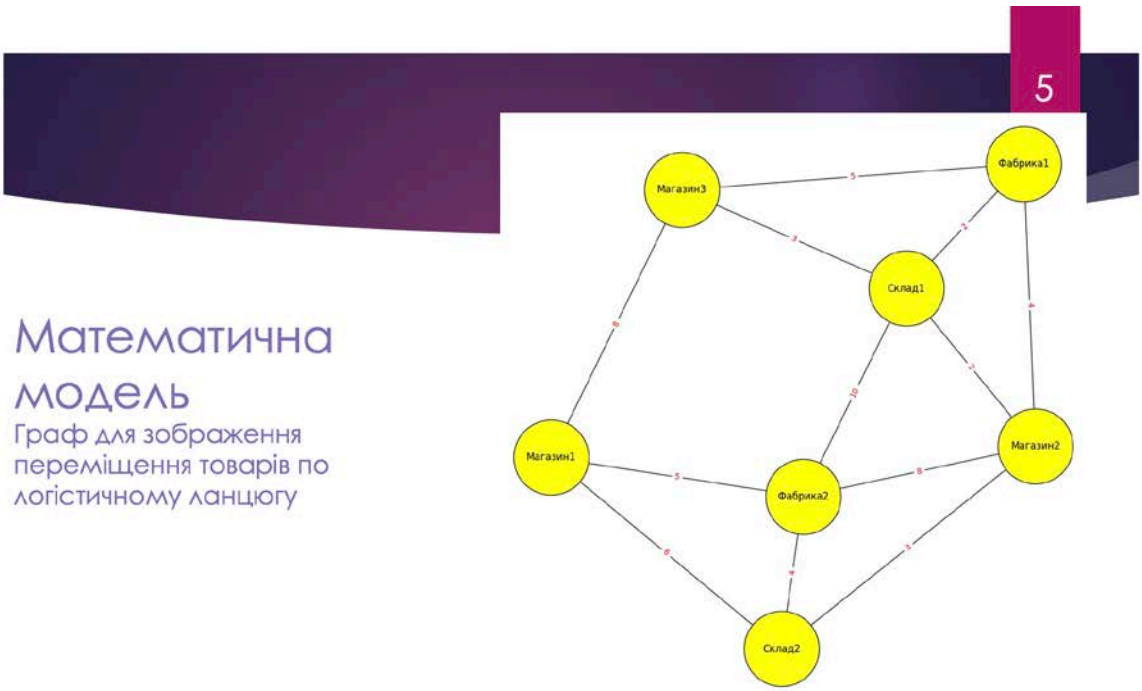

Рисунок Б.5 – Слайд 5

Рисунок Б.6 – Слайд 6

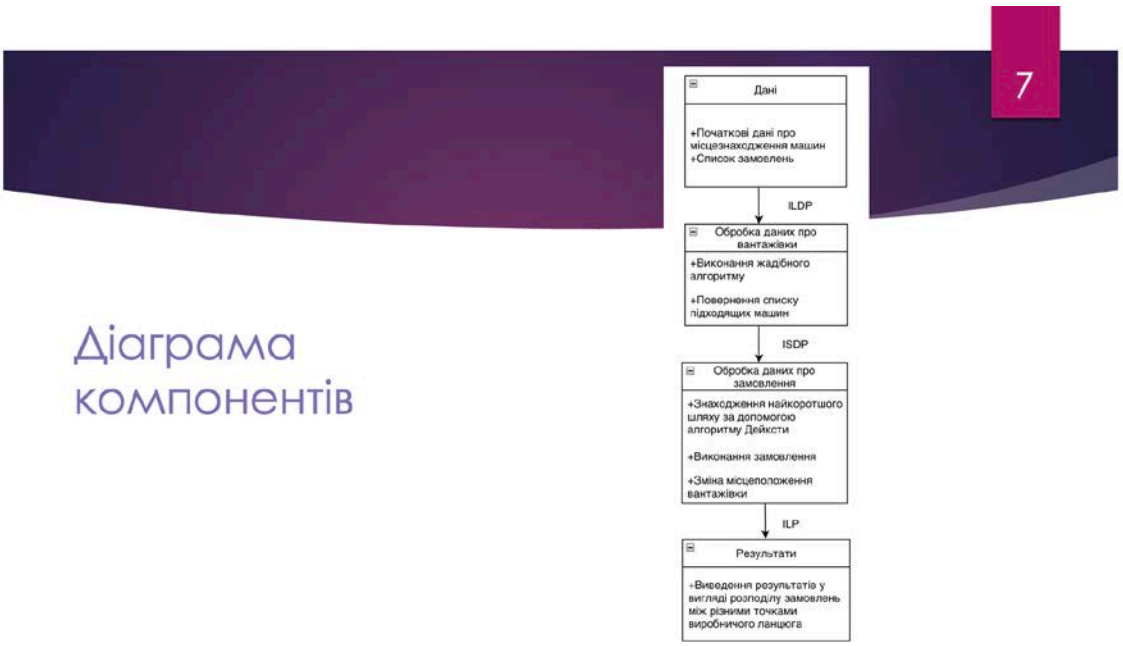

Рисунок Б.7 – Слайд 7

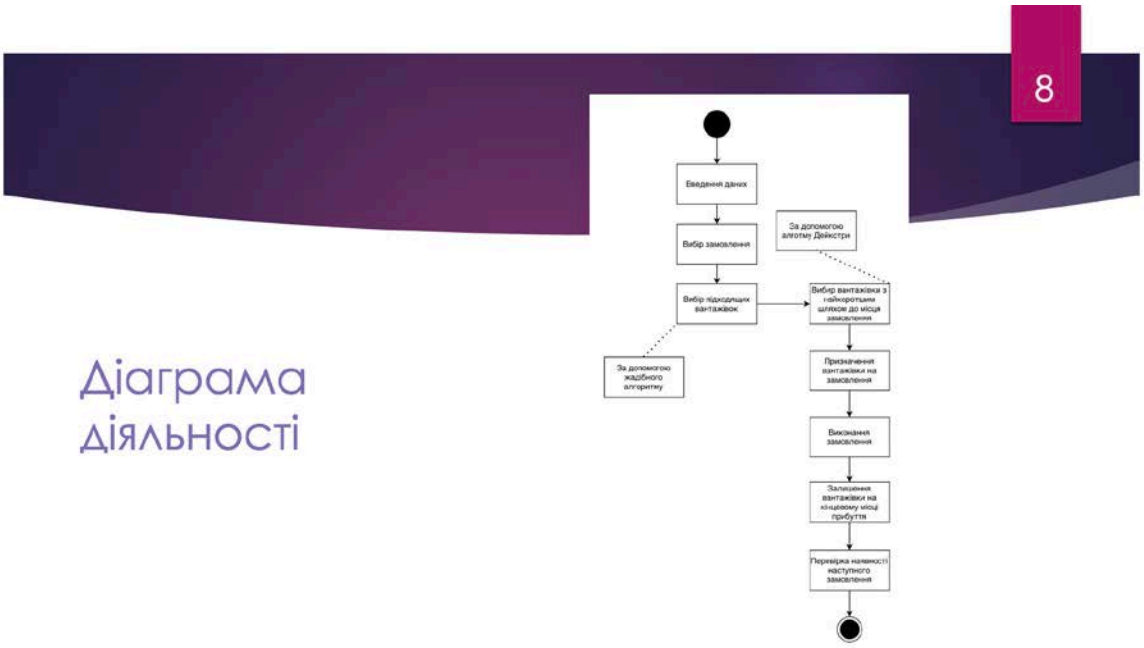

Рисунок Б.8 – Слайд 8

Рисунок Б.9 – Слайд 9

Рисунок Б.10 – Слайд 10

11

Початкові дані

Номер вантажівки	Вантажопідйомність	Початкова точка
1	3841	Фабрика1
2	2532	Фабрика11
3	2454	Магазин8
4	2948	Фабрика0
5	1369	Фабрика7
6	1263	Фабрика3
7	3793	Склад4
8	2104	Фабрика5
9	4080	Фабрика1

Рисунок Б.11 – Слайд 11

12

Замовлення

№	from	to	product	weight in kg
1	Фабрика0	Склад8	Пшеничний хліб	621
2	Фабрика3	Фабрика6	Вівсяне борошно	767
3	Фабрика0	Фабрика11	Житнє борошно	452
4	Фабрика1	Склад8	Житній хліб	584
5	Фабрика11	Фабрика0	Житнє борошно	2392
6	Фабрика3	Склад8	Вівсяний хліб	1751
7	Фабрика1	Фабрика11	Вівсяний хліб	1711
8	Фабрика1	Склад4	Житній хліб	2170
9	Фабрика5	Фабрика6	Пшеничний хліб	211
10	Фабрика11	Фабрика5	Пшеничний хліб	1829
11	Магазин8	Фабрика0	Пшеничне борошно	2179
12	Склад8	Фабрика1	Житнє борошно	2407
13	Фабрика0	Фабрика4	Вівсяне борошно	321
14	Фабрика7	Фабрика0	Пшеничний хліб	871

Рисунок Б.12 – Слайд 12

Рисунок Б.13 – Слайд 13

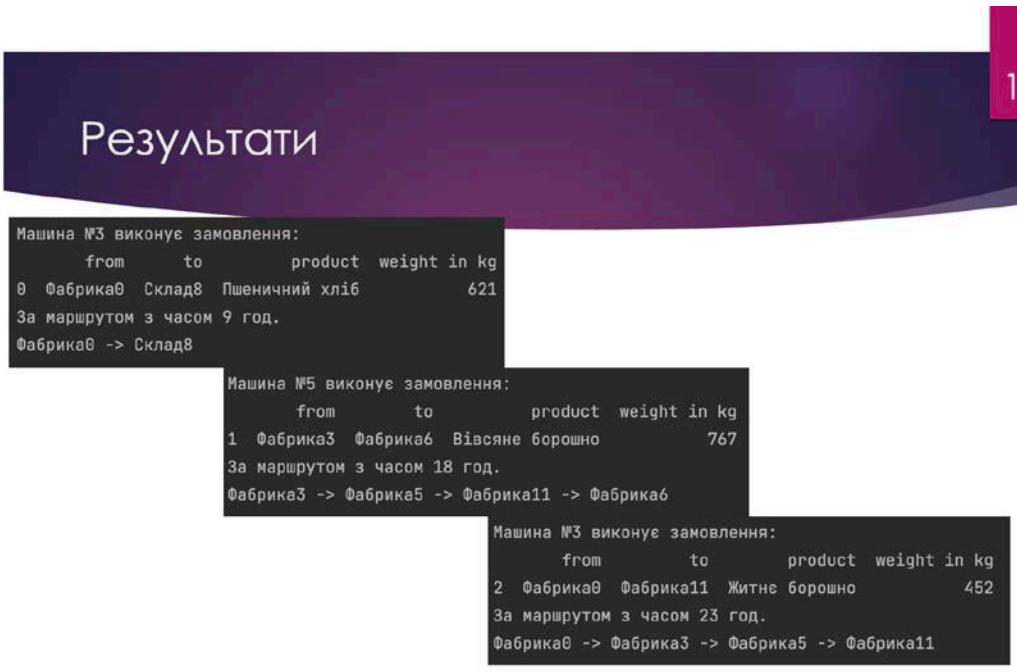

Рисунок Б.14 – Слайд 14

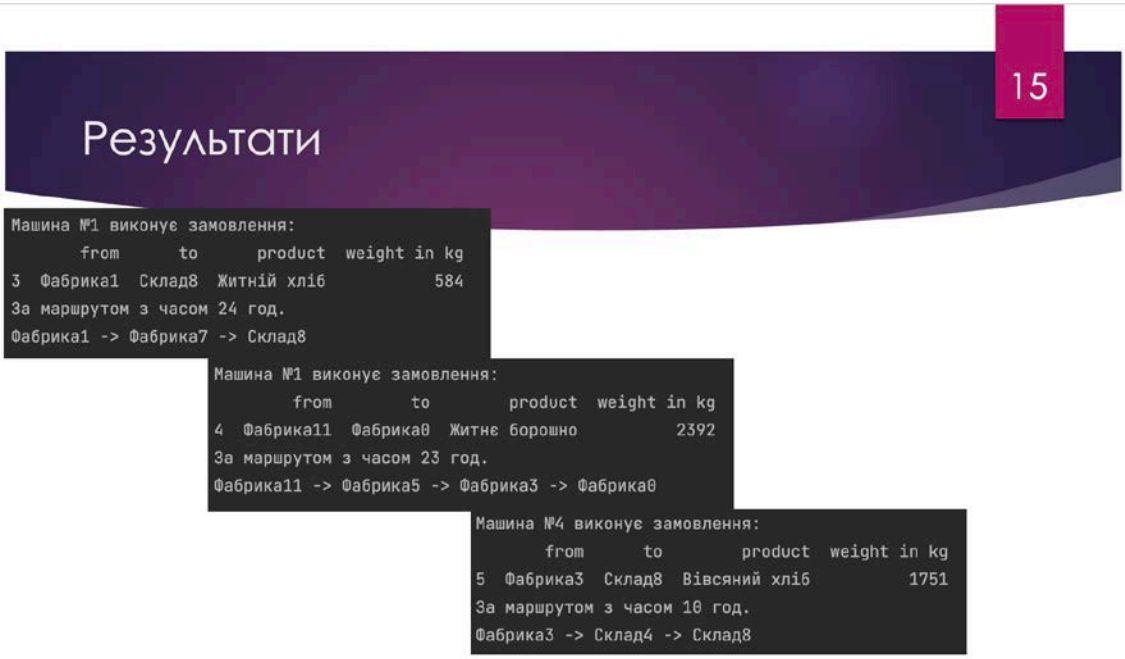

Рисунок Б.15 – Слайд 15

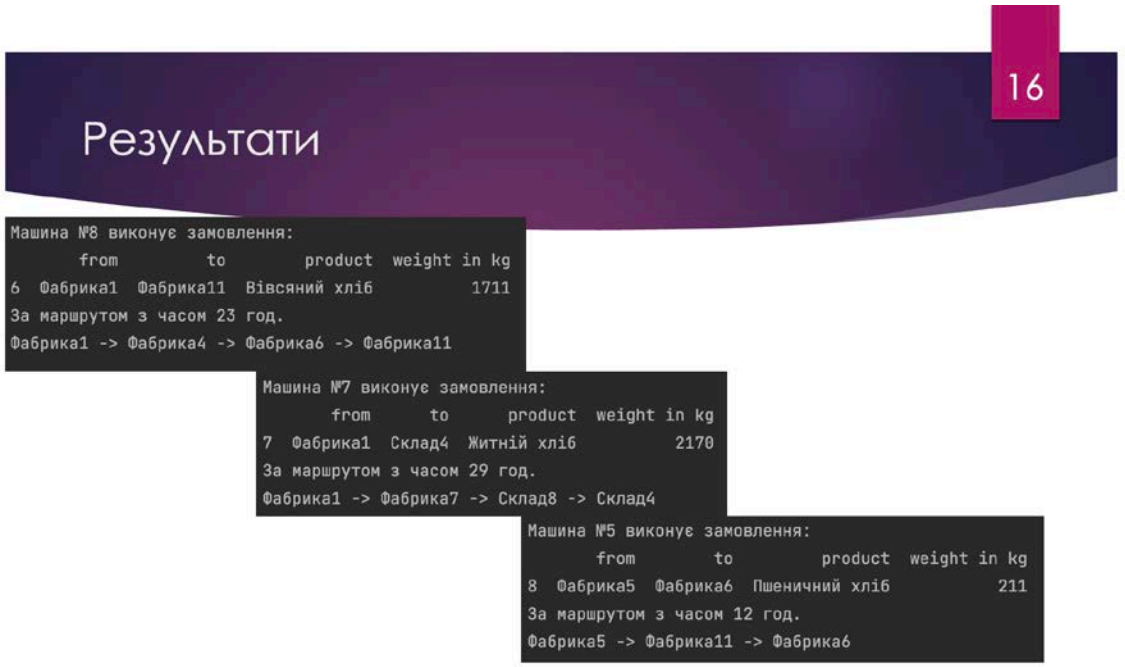

Рисунок Б.16 – Слайд 16

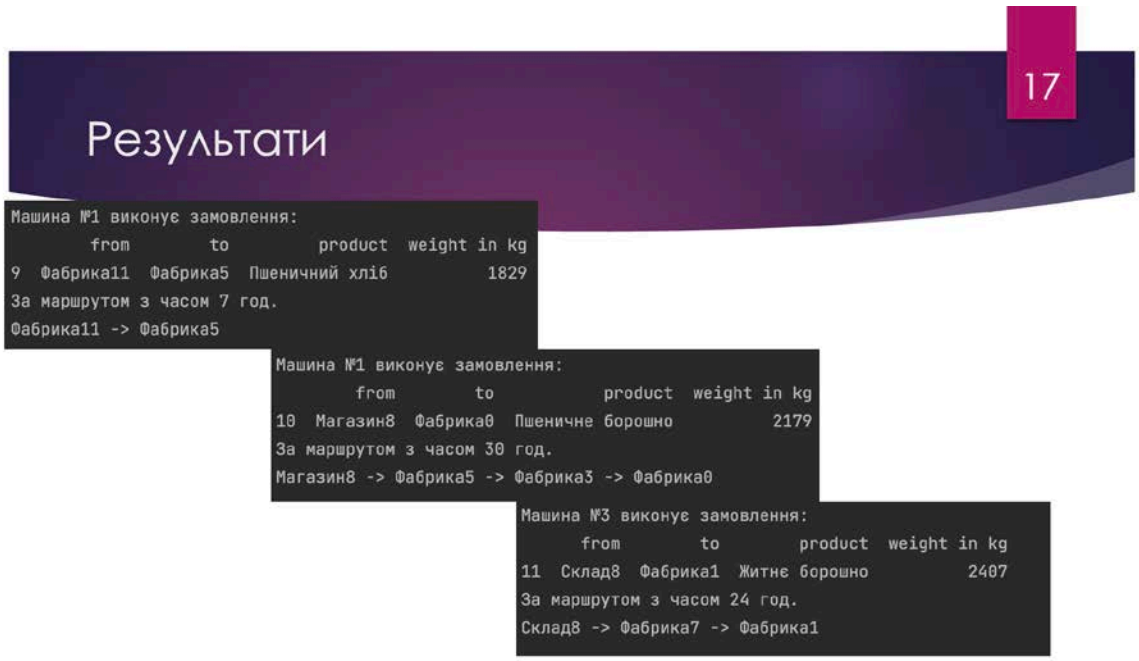

Рисунок Б.17 – Слайд 17

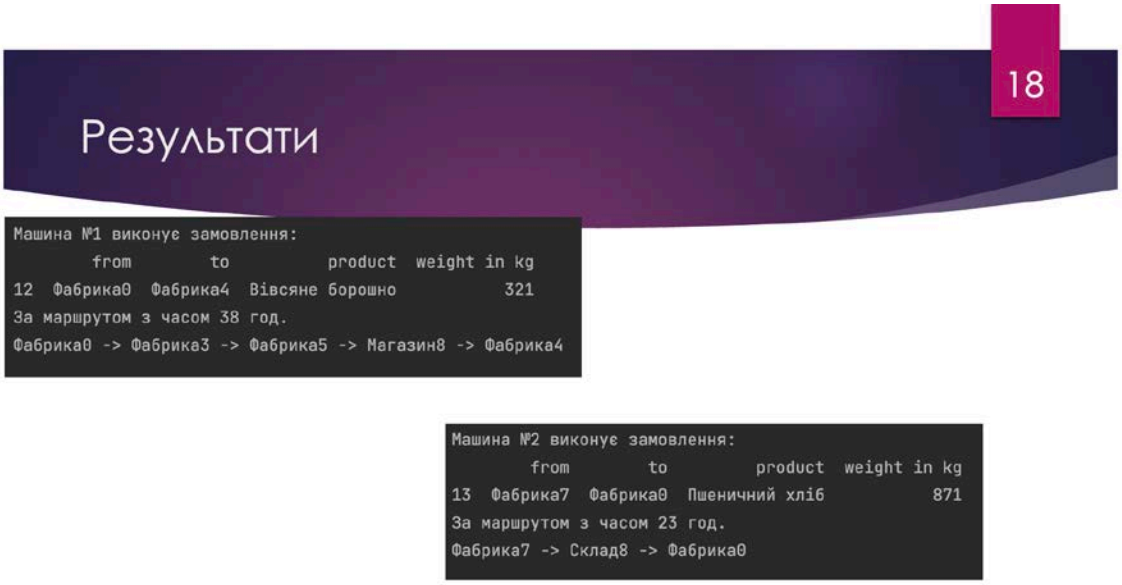

Рисунок Б.18 – Слайд 18

19

Висновки

У ході виконання поставленої задачі:

- було розглянуто проблемну область (логістичну систему) та проведено аналіз існуючих рішень (система Kanban, логістична система «точно в строк», логістична система планування потреб у розподілі, система планування потреби в матеріальних ресурсах, концепція планування потреби в матеріальних ресурсах у виробництві та оптимізована виробнича технологія);
- було побудовано модель логістичної системи оптимізації виробничих технологій за допомогою діаграм компонентів, діяльності та послідовностей;
- було описано математичну модель за допомогою якої була побудована логістична система оптимізації виробничих технологій;
- було реалізовано програмне забезпечення та проведено його тестування.

Рисунок Б.19 – Слайд 19