

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
**Національний університет «Запорізька політехніка»**

ІНСТИТУТ ІНФОРМАТИКИ ТА РАДІОЕЛЕКТРОНІКИ  
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК І ТЕХНОЛОГІЙ  
(повне найменування інституту, факультету)

КАФЕДРА СИСТЕМОГО АНАЛІЗУ ТА ОБЧИСЛЮВАЛЬНОЇ  
МАТЕМАТИКИ  
(повне найменування кафедри)

## **Пояснювальна записка**

до дипломного проєкту (роботи)

бакалавра

(ступінь вищої освіти)

на тему Дослідження задачі про рюкзак в умовах невизначеності

Виконав: студент(ка) 4 курсу, групи КНТ-818сп

Спеціальності 124 – Системний аналіз  
(код і найменування спеціальності)

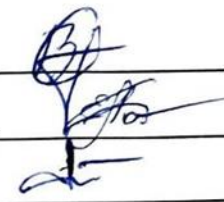
Освітня програма (спеціалізація)

«Інтелектуальні технології та прийняття рішень  
в складних системах»

\_\_\_\_\_ Смола В.І.  
(прізвище та ініціали)

Керівник \_\_\_\_\_ Подковаліхіна О.О.  
(прізвище та ініціали)

Рецензент \_\_\_\_\_ Стеганцев Є.В.  
(прізвище та ініціали)



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
Національний університет «Запорізька політехніка»  
(повне найменування закладу вищої освіти)

Інститут, факультет Інформатики та радіоелектроніки, комп'ютерних наук і технологій

Кафедра Системного аналізу та обчислювальної математики

Ступінь вищої освіти бакалавр

Спеціальність 124 – Системний аналіз  
(код і найменування)

Освітня програма  
(спеціалізація) «Інтелектуальні технології та прийняття рішень в складних системах»  
(назва освітньої програми (спеціалізації))

**ЗАТВЕРДЖУЮ**

Завідувач кафедри \_\_\_\_\_

Г.В. Корніч

« 08 » червня 2021 року

**З А В Д А Н Н Я**  
**НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)**

Смоли Владислава Ігоровича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження задачі про рюкзак в умовах невизначеності

керівник проєкту (роботи) Подковаліхіна Олена Олександрівна, к.ф.-м.н., доц.,  
( прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від « 26 » травня 2021 року № 215

2. Строк подання студентом проєкту (роботи) « 08 » червня 2021 року \_\_\_\_\_



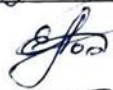



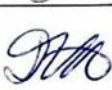
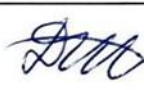
3. Вихідні дані до проєкту (роботи) Умови задачі про рюкзак, перелік літературних джерел за темою дослідження.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) В першому розділі розглядається питання актуальності задачі про рюкзак та її модифікацій, проведено аналіз останніх публікацій та досліджень за темою роботи. В другому та третьому розділах відповідно розглянуто задачу про завантаження та задачу про рюкзак в умовах невизначеності. Наведено загальну та математичну постановки задач. Описано алгоритм знаходження оптимальних розв'язків задач з вхідними параметрами, що підпорядковуються довільному закону розподілу. На основі алгоритму розроблено програмну реалізацію на мові програмування C++, що знаходить оптимальні плани задачі про завантаження та задачі про рюкзак з рівномірними вхідними параметрами та відповідну їм сумарну вартість, і заносить знайдені дані в файл формату csv для заданої кількості експериментів. Розроблено різні варіанти критеріїв оптимальності. Отримано результати розрахунку оптимальних розв'язків для прикладів задачі про завантаження та задачі про рюкзак з рівномірно розподіленими параметрами

для різних значень відхилень. Проведено аналіз оптимальних планів розглянутих прикладів для різних критеріїв оптимальності.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1	Подковаліхіна О.О., к.ф.-м.н., доц.		
2	Подковаліхіна О.О., к.ф.-м.н., доц.		
3	Подковаліхіна О.О., к.ф.-м.н., доц.		
Нормаконтроль	Широкоград Д.В., к.ф.-м.н., ст. викл.		

7. Дата видачі завдання « 08 » жовтня 2020 року.

### КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Сформулювати мету та основні завдання дипломної роботи	08.10.2020 – 30.10.2020	
2	Опрацювати літературу та існуючі дослідження за темою роботи	02.11.2020 – 20.11.2020	
3	Розробка програмної реалізації для вирішення задачі	20.11.2020 – 19.03.2021	
4	Розрахунки та аналіз даних	19.03.2021 – 30.04.2021	
5	Оформлення пояснювальної записки	03.05.2021 – 24.05.2021	
6	Попередній захист дипломної роботи та отримання рецензій.	25.05.2021 – 07.06.2021	
7	Захист дипломної роботи.	08.06.2021	

Студент

  
(підпис)

Смола В.І.

(прізвище та ініціали)

Керівник проєкту (роботи)

  
(підпис)

Подковаліхіна О.О.

(прізвище та ініціали)

## РЕФЕРАТ

Дипломна робота: 48 с., 6 рис., 8 табл., 1 дод., 15 літературних джерел.

Об'єкт дослідження – моделі динамічного програмування в умовах невизначеності.

Предмет дослідження – задача про рюкзак в умовах невизначеності.

Мета роботи – дослідження впливу статистичної невизначеності на оптимальний розв'язок задачі про рюкзак та її модифікацій і розроблення програмного забезпечення для розв'язання задачі, що знаходить оптимальні плани та відповідну їм сумарну вартість, та заносить знайдені дані в файл для заданої кількості експериментів.

Методи дослідження – метод повного перебору.

В дипломній роботі розглянуто задачу про завантаження та задачу про рюкзак в умовах невизначеності. Наведено загальну та математичну постановки задач. Описано алгоритм знаходження оптимальних розв'язків задач з вхідними параметрами, що підпорядковуються довільному закону розподілу. На основі алгоритму розроблено програмну реалізацію на мові програмування C++, що знаходить оптимальні плани задач з рівномірними вхідними параметрами та відповідну їм сумарну вартість, і заносить знайдені дані в файл для заданої кількості експериментів. Розроблено різні варіанти критеріїв оптимальності. Отримано результати розрахунку оптимальних розв'язків для прикладів задачі про завантаження та задачі про рюкзак з рівномірно розподіленими параметрами для різних значень відхилень. Проведено аналіз оптимальних планів розглянутих прикладів для різних критеріїв оптимальності.

МОДЕЛІ ДИНАМІЧНОГО ПРОГРАМУВАННЯ, ЗАДАЧА ПРО РЮКЗАК, ЗАДАЧА ПРО ЗАВАНТАЖЕННЯ, МЕТОД ПОВНОГО ПЕРЕБОРУ, СТАТИСТИЧНА НЕВИЗНАЧЕНІСТЬ, КРИТЕРІЇ ОПТИМАЛЬНОСТІ, МОВА ПРОГРАМУВАННЯ C++, МОВА ПРОГРАМУВАННЯ R.

## ЗМІСТ

Завдання .....	2
Реферат .....	4
Вступ.....	6
1 Огляд літератури та останніх досліджень і публікацій .....	7
2 Задача про завантаження в умовах невизначеності .....	14
2.1 Загальна постановка задачі про завантаження .....	14
2.2 Математична постановка задачі про завантаження .....	14
2.3 Алгоритм розв'язання задачі про завантаження в умовах невизначеності.....	15
2.4 Критерії оптимальності та аналіз оптимального плану задачі про завантаження .....	16
2.5 Приклад задачі про завантаження.....	17
3 Задача про рюкзак в умовах невизначеності.....	25
3.1 Загальна постановка задачі про рюкзак.....	25
3.2 Математична постановка задачі про рюкзак.....	25
3.3 Алгоритм розв'язання задачі про рюкзак в умовах невизначеності ...	26
3.4 Критерії оптимальності та аналіз оптимального плану задачі про рюкзак .....	27
3.5 Приклад задачі про рюкзак .....	28
Висновки .....	35
Перелік посилань .....	36
Додаток А. Програмний код C++ .....	38

## ВСТУП

Наука приділяє багато уваги питанням організації та управління, тому розвиток в даному напрямку призвів до необхідності аналізу складних процесів та знаходження їх розв'язку.

В наш час потрібно знаходити рішення не просто складних систем, а складних динамічних систем, де потрібно працювати з даними, які мають статистичну невизначеність. Тому з'явився ефективний розвиток методів рішення складних динамічних систем різного типу в залежності від призначення та цілей, які необхідно досягти.

Розвиток в даному напрямку призвів до розвитку класичних задач динамічного програмування, які є актуальними в наш час. Моделі динамічного програмування застосовують при рішенні таких задач: розробка правил управління запасами, розробка принципів календарного планування, розподіл дефіцитних капітальних вкладень тощо.

До класичних задач динамічного програмування відноситься задача про рюкзак. В цій задачі необхідно розподілити ресурси так, щоб це принесло найбільшу сумарну вартість при завантаженні човна або літака, вибору багажів для оптимального завантаження транспортного засобу тощо. Для розв'язку задачі про рюкзак достатньо знайти рішення точне або наближене до оптимального.

# 1 ОГЛЯД ЛІТЕРАТУРИ ТА ОСТАННІХ ДОСЛІДЖЕНЬ І ПУБЛІКАЦІЙ

Задача про рюкзак та її модифікації часто з'являються в економіці, прикладній математиці, криптографії, генетиці і логістиці для знаходження оптимального завантаження транспорту чи складу.

Задача про рюкзак є важливою в нашому житті. Один із прикладів її застосування – це в логістиці, а саме: задача оптимального завантаження транспортних засобів, ефективне рішення якої не тільки дозволяє зменшити витрати на перевезення, але і скоротити час вантажно-розвантажувальних робіт. Існують спеціальні програми визначення оптимального розміщення вантажу в кузові автомобіля, контейнері, вагоні тощо.

Програми оптимізації завантаження транспорту можуть:

- заощадити на перевезенні, збільшивши щільність завантаження на 5-20%;
- швидко відповісти на питання про те, скільки місця займе вантаж в контейнері, напівпричепі, вагоні;
- оптимально підібрати транспорт;
- точно визначити, скільки знадобиться контейнерів для великого відвантаження;
- бути абсолютно впевненим, що нічого не залишиться у постачальника.

За короткий час програми для визначення оптимального розміщення стали невід'ємною частиною програмного забезпечення логістів.

Свою назву задача про рюкзак отримала від оптимізаційної задачі укладання якомога більшого числа цінних речей в рюкзак за умови, що загальний обсяг (або вага) усіх предметів, здатних поміститися в рюкзак, обмежений.

Задача має різні змістовні трактування. В роботі [1] авторів Беллмана Р.Е. та Дрейфуса С.Е. задача про рюкзак інтерпретована як задача про завантаження

судна. Постановка задачі виглядає так: «Припустимо необхідно завантажити судно вантажем окремими предметами різних типів. Так як ці предмети мають різну вагу та вартість, з'являється задача про те, як завантажити судно, яке обмежене вантажопідйомністю таким чином, щоб вантаж набрав найбільшої цінності».

В роботі [2] автором Сааті Т.Л. було розглянуто задачу про ранець в наступній постановці. Однією з найпростіших задач цілочисельної оптимізації є лінійна задача, в якій треба мінімізувати або максимізувати в невід'ємних цілих числах лінійний вираз з обмеженнями у вигляді лінійних нерівностей. Задача формулюється наступним чином. Знайти

$$\max \sum_{j=1}^n c_j x_j$$

при обмеженнях

$$\sum_{j=1}^n a_j x_j \leq b,$$

де всі  $x_j \geq 0$  – цілі числа. У векторному вигляді задача полягає в тому, щоб максимізувати  $cx$  при обмеженнях  $Ax \leq b$ , де  $c = (c_1, \dots, c_n)$ ,  $x^T = (x_1, \dots, x_n)^T$ ,  $A = (a_1, \dots, a_n)$  і компоненти вектора  $x$  – невід'ємні цілі числа. Задача полягає у тому, що в ранець ємністю  $b$  треба упакувати  $n$  видів предметів з вагами  $c_1, \dots, c_n$  і розмірами  $a_1, \dots, a_n$  відповідно так, щоб завантаження ранця було максимальним.

Також задачу про ранець розглядали в роботі [3] Корбут О.А. та Фінкельштейн Ю.Ю. Автори формулювали постановку задачі наступним чином. Є  $n$  предметів; задані величини:

$a_j$  – вага предмета,



$c_j$  – цінність предмета.

Потрібно завантажити рюкзак, вантажопідйомність якого дорівнює  $A$ , набором предметів з максимальною сумарною цінністю. Якщо ввести змінні  $x_j$ ,  $j = 1, 2, \dots, n$ , що мають наступний сенс:

$$x_j = \begin{cases} 1, & \text{якщо } j - \text{й предмет підлягає завантаженню,} \\ 0 & \text{в іншому випадку,} \end{cases}$$

то задача про рюкзак зведеться до максимізації функції

$$c_1x_1 + c_2x_2 + \dots + c_nx_n$$

за умов

$$x_j = \begin{cases} 0, \\ 1, \end{cases} \quad j = 1, 2, \dots, n,$$

$$a_1x_1 + a_2x_2 + \dots + a_nx_n \leq A.$$

В інших задачах такого типу може бути кілька обмежень виду  $a_1x_1 + a_2x_2 + \dots + a_nx_n \leq A$  (наприклад, обмеженою може бути не тільки сумарна вага предметів, що завантажуються, але і їх сумарний обсяг і т.п.). Задачі, про які сказано вище, називають багатовимірними задачами про рюкзак. Припускаючи, що кожен предмет може завантажуватися не в одному, а в декількох примірниках, то обмеження  $x_j = \begin{cases} 0, \\ 1, \end{cases} \quad j = 1, 2, \dots, n$  заміниться умовою невід'ємності і цілочисельності всіх змінних.

В роботі [4] Окулова С.М. задача має таке трактування: у гіпотетичний рюкзак, вага якого не повинна перевищувати  $P$  умовних одиниць (у.о.) загрузаються предмети  $n$  типів. Предмети кожного типу мають масу  $m_i$  і вартість  $c_i$ , на їх кількість обмежень немає. Необхідно завантажити рюкзак так, щоб сумарна вартість взятих предметів була максимальна.

Позначимо через  $u_i$  кількість предметів типу  $i$ , поміщених в рюкзак. Формальний запис задачі має такий вигляд.

Знайти  $\max_{\langle u_1, u_2, \dots, u_n \rangle} \sum_{i=1}^n c_i u_i$  при обмеженнях  $\sum_{i=1}^n m_i u_i \leq P$  ( $m_i \geq 0$ ,  $c_i \geq 0$ ,  $p > 0$ ,  $i = 1, 2, \dots, n$ ) і  $u_i \geq 0$  цілі.

Процес завантаження рюкзак є багатошаровим: завантаження предметів першого типу, другого і т.д., до останнього типу. Стан системи на кроці  $i$  визначається значенням змінної  $x_i$  – сумарною вагою предметів, завантажених в рюкзак з першого по  $i$ -й крок. Керуючі впливи на кроці  $i$  мають кількість предметів типу  $i$  ( $u_i$ ), що завантажуються в рюкзак. Функціональна залежність  $f$  виражається як  $x_i = x_{i-1} + m_i u_i$  ( $x_0 = 0$ ), а критерії ефективності – як  $w_i = c_i u_i$ . Задача адитивна:  $W = \sum_{i=1}^n w_i$ .

В роботі [5] Таха Х.А. задача про завантаження – це конструктивне завантаження судна (літака, автомашини і т.п.), яке має обмеження за обсягом та вантажопідйомністю. Кожен вантаж розташований на судні приносить певний прибуток. Задача полягає у завантаженні судна такими вантажами, що приносять найбільший сумарний прибуток.

В роботі [6] автор Левітин А.В. розглядає постановку задачі в такому вигляді: «Дано  $n$  предметів з вагою  $w_1, \dots, w_n$  і ціною  $v_1, \dots, v_n$ , а також рюкзак, який витримує вагу  $W$ . Потрібно знайти підмножину предметів, які можна розмістити в рюкзак і яка має при цьому максимальну ціну.»

Задача про рюкзак може бути розв'язана точними, наближеними або стохастичними методами. До точних методів належать повний перебір, метод гілок і меж та інші. Група наближених методів представлена різновидами методу динамічного програмування. Стохастичними методами є жадібний, генетичний алгоритми та інші.

В роботі [6] розглядається метод гілок та меж і вичерпний перебір. Алгоритм розв'язання задачі методом гілок і меж виглядає наступним чином (постановка задачі наведена вище).

Потрібно впорядкувати предмети в порядку спадання за їх питомою ціною (стосовно ціни до ваги), з розв'язанням неоднозначності довільним чином:

$$\frac{v_1}{w_1} \geq \frac{v_2}{w_2} \geq \dots \geq \frac{v_n}{w_n}.$$

Очевидною структурою дерева простору станів для даної задачі є бінарне дерево. Кожен вузол на рівні  $0 \leq i \leq n$  представляє всі підмножини з  $n$  елементів, які включають певний вибір з перших  $i$  упорядкованих елементів. Цей частковий вибір однозначно визначається шляхом від кореня до вузла: гілка, що йде вліво, вказує на включення чергового елемента в підмножину, в той час як права гілка вказує на відсутність елемента в підмножині. Записуємо загальну вагу  $w$  і загальну вартість  $v$  вибору, відповідно вузлу, разом з верхньою межею  $ub$  значення для будь-якої підмножини, яка може бути отримана шляхом додавання деяких елементів (можливо, ніяких) до цього вибору.

Також автор розглядає вичерпний перебір (повний перебір).

Вичерпний перебір в цій задачі призводить до розгляду всіх підмножин даної множини з  $n$  предметів, обчисленню загальної ваги кожного з них для того, щоб з'ясувати, чи допустимий такий набір предметів (тобто не перевищує його загальна вага можливості рюкзака) і вибір з допустимих підмножин з максимальною вагою. Оскільки загальна кількість підмножин  $n$ -елементної множини дорівнює  $2^n$ , вичерпний перебір призводить до алгоритму з часом роботи  $\Omega(2^n)$ , незалежно від того, наскільки ефективним методом генеруються розглянуті підмножини.

В статті [7] автор Кагіров Р.Р. розглядає задачу про багатовимірний рюкзак та спосіб розв'язку задачі за допомогою алгоритмів мурашиних колоній, який є новим перспективним методом оптимізації та базується на моделюванні їх поведінки.

В статті [8] досліджують різні розширення класичної задачі про рюкзак на випадок, коли різні елементи постановки задачі схильні до деякої

невизначеності, описуваної випадковими величинами. Також пропонується модель двоступеневого квадратичного ранця з регресом, який має ймовірне обмеження на місткість ранця на першому етапі. Прийоми розв'язку основані на напіввизначених релаксаціях. Це дозволяє розв'язувати великі приклади, які не можна вирішувати точними методами.

В статті [9] Канцедала С.А., Костікової М.В. і Скрипіни І.В. викладено та обґрунтовано основні поняття й можливості рішення задачі про рюкзак точними методами гілок і границь та методами динамічного програмування. Описані методи та алгоритм розв'язання задачі, а також порівнюються алгоритми, які були одержані експериментальним шляхом.

В статті [10] Купріяшина М.А. та Борзунова Г.І. розглядається алгоритм точного розв'язку задачі про рюкзак методом динамічного програмування. Особливістю даного алгоритму є лінійна залежність часової складності алгоритму від розмірності задачі. При виконанні алгоритму будується таблиця, рядки якої є рішеннями підзадач, що характеризуються меншим набором предметів. Також розглянуто процес побудови шуканого вектора укладання по заповненій таблиці та досліджуються оцінки складності алгоритму за часом і по пам'яті, а також досліджується залежність цих показників від показника щільності рюкзакового вектору, який широко застосовується при аналізі уразливості рюкзакових систем шифрування до атак. Результати дозволяють оцінити можливість практичного застосування алгоритму динамічного програмування для вирішення задачі про рюкзак при заданій щільності рюкзакового вектору.

В статті [11] Васільчікова В.В. пропонується ефективний паралельний алгоритм рішення NP-повної задачі про рюкзак в її вихідному, так званому 0-1 варіанті. Для знаходження її точного рішення застосовуються алгоритми, які відносяться до категорії «методів гілок і границь». Для прискорення отримання результату з різною ефективністю використовуються також різні варіанти організації паралельних обчислень. Також пропонується алгоритм розв'язку задач, які основані на парадигмі рекурсивно-паралельних обчислень. Основною

метою експериментів є дослідження прискорення, за рахунок рекурсивно-паралельної організації обчислень. Докладний опис алгоритма та експеримента, а також отримані результати приводяться в роботі.

В статті [12] Топки В.В. розглянуто оптимальний відбір портфелю проектів і розподіл ресурсів на його виконання у вигляді задачі дискретної оптимізації – багатовимірної задачі про рюкзак. Наводяться результати для жадібного алгоритму розв'язання задачі про одновимірний рюкзак; необхідні і достатні умови оптимальності; оцінка похибки алгоритму і його асимптотична похибка для поведінки в середньому. Пропонується прямий жадібний алгоритм рішення багатовимірної задачі про рюкзак. Для підвищення його ефективності застосовується локальний обмежений перебір, що поліпшує жадібне рішення, а також локальна оптимізація і оптимізація попередніх етапів.

## 2 ЗАДАЧА ПРО ЗАВАНТАЖЕННЯ В УМОВАХ НЕВИЗНАЧЕНОСТІ

### 2.1 Загальна постановка задачі про завантаження

У гіпотетичний рюкзак (склад, кузов транспортного засобу тощо), вага або об'єм якого не повинні перевищувати  $V$  умовних одиниць (у.о.), завантажуються предмети  $n$  типів. Предмети кожного типу мають вагу або об'єм  $a_i$  та вартість або цінність  $c_i$ . Вартість або цінність одного предмету кожного типу може варіюватися відповідно заданому закону розподілу. Необхідно завантажити рюкзак (склад, кузов транспортного засобу тощо) так, щоб сумарна вартість або сумарна цінність взятих предметів була максимальна. Сумарна вага або об'єм завантажених предметів не повинна перевищувати вагу або об'єм рюкзаку  $V$  [8].

### 2.2 Математична постановка задачі про завантаження

$$F = \sum_{j=1}^n c_j x_j \rightarrow \max$$

$$\sum_{j=1}^n a_j x_j \leq V,$$

$c_j$  можуть варіюватися відповідно заданому закону розподілу,

де  $n$  – кількість типів предметів;

$x_j$  – кількість предметів типу  $j$ ;

$c_j$  – вартість або цінність одного завантаженого предмету типу  $j$ ;

$a_j$  – вага або об'єм одного завантаженого предмету типу  $j$ ;

$V$  – вага або об'єм рюкзаку;

$F$  – сумарна вартість або цінність завантажених предметів без перевищення ваги або об'єму рюкзаку.

## 2.3 Алгоритм розв'язання задачі про завантаження в умовах невизначеності

1. Задаємо дані для розв'язання задачі, а саме:

- Об'єм або вагу рюкзаку (склад, кузов транспортного засобу тощо).
- Вагу або об'єм для предметів кожного типу.
- Вартість або цінність для предметів кожного типу.
- Кількість експериментів.
- Місце, де буде знаходитись файл з розрахунками.
- Максимальна кількість предметів кожного типу, яку можливо

помістити в рюкзак (склад, кузов транспортного засобу тощо).

Щоб дізнатися максимальну кількість предметів кожного типу, потрібно спочатку обрати предмет певного типу. Потім вагу або об'єм рюкзаку поділити на вагу або об'єм предмета обраного типу і з отриманого числа виділити цілу частину. І ця ціла частина буде показувати максимальну кількість предметів обраного типу, яку можливо помістити в рюкзак . Такі розрахунки потрібно потвори для предметів кожного типу.

- Мінімальна кількість предметів кожного типу, яку необхідно завантажити в рюкзак (склад, кузов транспортного засобу тощо).

2. Знайдемо кількість всіх оптимальних планів, які підходять за всіма обмеженнями, а саме за обмеженням об'єму або ваги рюкзаку (склад, кузов транспортного засобу тощо), а також обмеженням за кількістю тих чи інших типів предметів з умови задачі.

Для того, щоб створити масив, потрібно дізнатися скільки оптимальних планів може мати дана задача так, щоб не перевищувати вагу або об'єм рюкзаку (склад, кузов транспортного засобу тощо). Тому була розроблена спеціальна функція, що підраховує кількість за допомогою повного перебору всіх оптимальних планів.

3. Створимо масив та занесемо всі оптимальні плани, що не перевищують вагу або об'єм рюкзаку, до нього.

4. Формуємо випадкові числа цінності або вартості за допомогою генератора випадкових чисел для рівномірного розподілу з різними відхиленнями для предметів кожного типу.

5. Зробимо повний перебір оптимальних планів для даних випадкових чисел цінності або вартості.

На даному кроці робиться повний перебір оптимальних планів, щоб дізнатись, який план дає найбільшу сумарну вартість або цінність та відповідає усім обмеженням задачі.

6. Обираємо найкращий оптимальний план, який дає найбільшу сумарну вартість, а також задовольняє усім обмеженням задачі, і заносимо його до масиву.

7. Повторюємо кроки 4-6 відповідно з вказаною кількістю експериментів.

8. Заносимо всі оптимальні плани, що з'явилися, в файл.

9. Заносимо всі сумарні вартості відповідні цим оптимальним планам також в цей файл.

Даний алгоритм є описом роботи програмної реалізації для задачі про завантаження, що написана на мові програмування C++ [13]. Щоб обрати даний алгоритм в програмі потрібно обрати задача №1 (див. додаток А).

## 2.4 Критерії оптимальності та аналіз оптимального плану задачі про завантаження

В результаті розв'язку задачі про завантаження в умовах невизначеності можуть бути такі два варіанти:

- Один оптимальний план.
- Більше ніж один оптимальний план.



Для першого випадку розробка критеріїв оптимальності немає сенсу, так як маємо один оптимальний план, що і буде рекомендований для реалізації. Для другого випадку необхідно провести аналіз отриманих оптимальних планів. Для розглянутої задачі про завантаження можна рекомендувати наступні критерії оптимальності:

1. Кількість з'явлень відповідного плану у відсотках.
2. Відсоток наповненості складу відповідними наборами предметів.
3. Середньоарифметична сумарна вартість.
4. Імовірність того, що отримана сумарна вартість буде більша, ніж певний прибуток.

## 2.5 Приклад задачі про завантаження

Власник може завантажити склад наборами наступних предмети: труби, електродвигуни, спецодяг, автозапчастини та кабелі. Об'єм складу дорівнює 100%. Один набір труб займає 31% об'єму складу, один набір електродвигунів – 13%, один набір спецодягу – 10%, один набір автозапчастин – 9,5% і один набір кабелів – 8%. Вартість одного набору труб дорівнює 10050 у.о., одного набору електродвигунів – 4210 у.о., одного набору спецодягу – 3130 у.о., одного набору автозапчастин – 3000 у.о. і одного набору кабелів – 2600 у.о. На склад обов'язково потрібно завантажити хоча б один набір труб, один набір електродвигунів та один набір спецодягу.

В залежності від комплектації та призначення вартість одного набору предметів може варіюватися відповідно рівномірному закону розподілу в наступних межах:

- труби – 35%;
- електродвигуни – 20%;
- спецодяг – 15%;
- автозапчастини – 50%;
- кабелі – 30%.

Необхідно завантажити склад так, щоб сумарна вартість завантажених предметів була максимальною.

Задача має наступну математичну постановку:

$$F = c_1x_1 + c_2x_2 + c_3x_3 + c_4x_4 + c_5x_5 \rightarrow \max$$

$$\left\{ \begin{array}{l} a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 + a_5x_5 \leq 100 \\ x_i \geq 1, \quad i = \overline{1,3} \\ x_i \geq 0, \quad i = \overline{4,5} \\ 6532,5 \leq c_1 \leq 13567,5 \\ 3368 \leq c_2 \leq 5052 \\ 2660,5 \leq c_3 \leq 3599,5 \\ 1500 \leq c_4 \leq 4500 \\ 1820 \leq c_5 \leq 3380 \end{array} \right. ,$$

де  $x_1$  – кількість наборів труб;

$x_2$  – кількість наборів електродвигунів;

$x_3$  – кількість наборів спецодягу;

$x_4$  – кількість наборів автозапчастин;

$x_5$  – кількість наборів кабелів;

$c_1$  – вартість одного набору труб;

$c_2$  – вартість одного набору електродвигунів;

$c_3$  – вартість одного набору спецодягу;

$c_4$  – вартість одного набору автозапчастин;

$c_5$  – вартість одного набору кабелів;

$a_1$  – об'єм одного набору труб, який дорівнює 31%;

$a_2$  – об'єм одного набору електродвигунів, який дорівнює 13%;

$a_3$  – об'єм одного набору спецодягу, який дорівнює 10%;

$a_4$  – об'єм одного набору автозапчастин, який дорівнює 9,5%;

$a_5$  – об'єм одного набору трю кабелів, який дорівнює 8%;

$F$  – сумарна вартість завантажених предметів без перевищення об'єму складу.

Робимо запуск програми та вибираємо нашу задачу. Після закінчення роботи програми на робочому столі з'являються три файли з різною кількістю даних. Розподіли вартості для 1000, 5000 та 10000 даних проілюструємо на діаграмах типу «ящик з вусами», що були побудовані за допомогою функції `boxplot` мови програмування R [14]. Результати наведені на рис 2.1 – 2.3.

На діаграмі для кожного оптимального плану зображений «ящик з вусами». Середньою лінією (медіаною) зображене середнє значення оптимальних планів. Мінімальне та максимальне значення показане на кінцях вусика. Самі вуса зображують повний розмах вибірки.

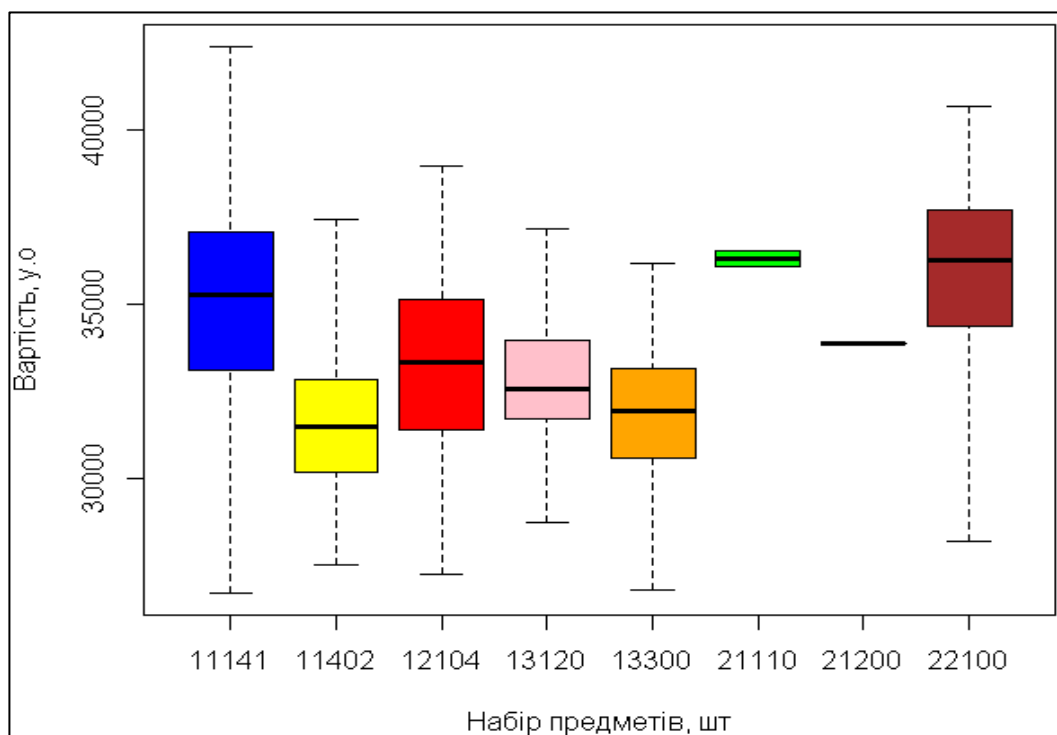


Рисунок 2.1 – Результати розрахунків для 1000 експериментів

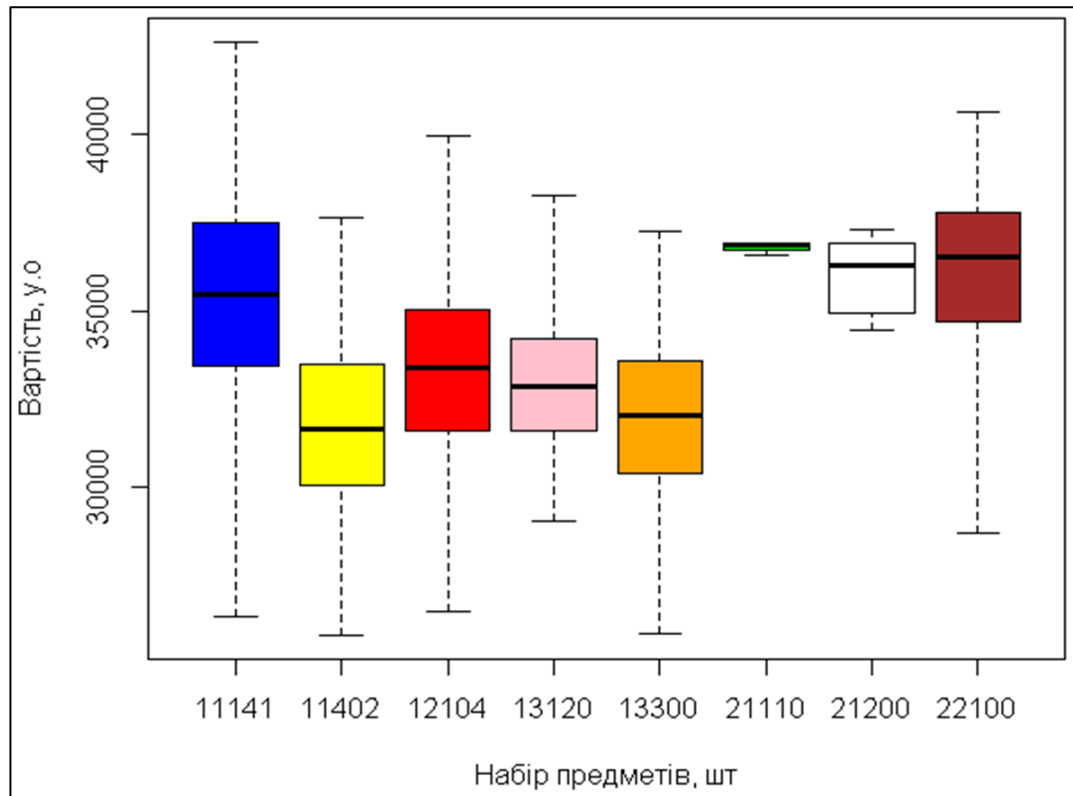


Рисунок 2.2 – Результати розрахунків для 5000 експериментів

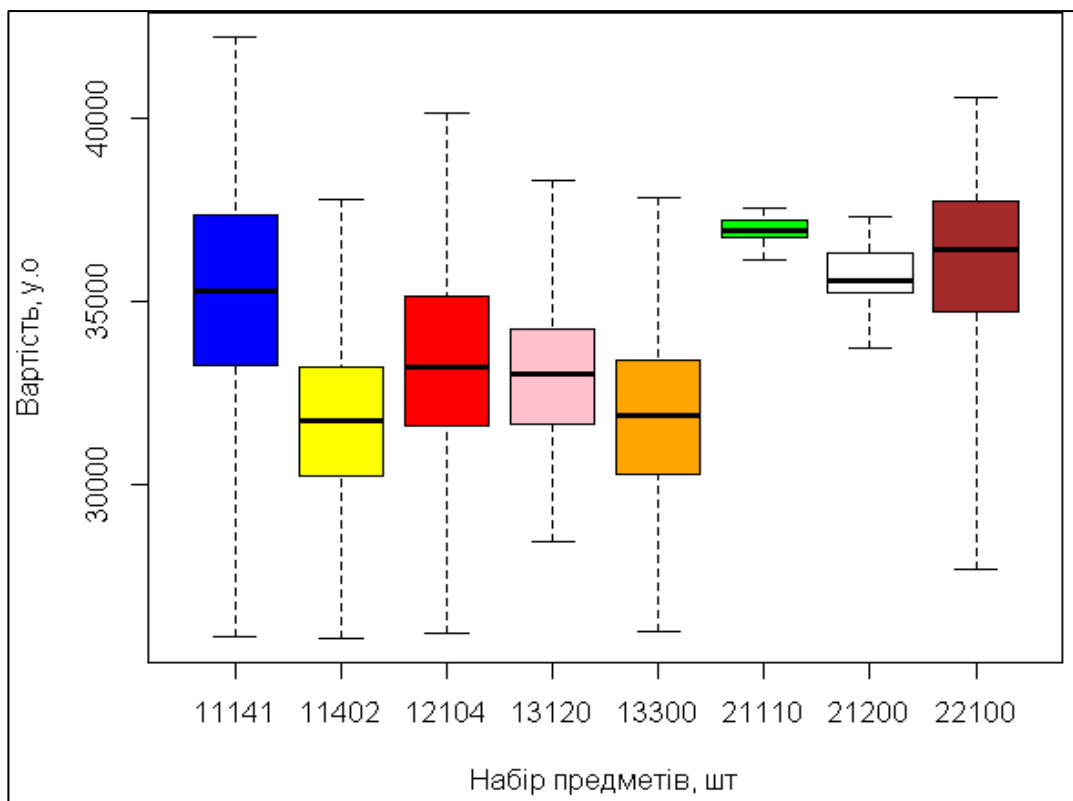


Рисунок 2.3 – Результати розрахунків для 10000 експериментів

За результатами всіх експериментів було отримано однакову кількість оптимальних планів. Тому маємо такі плани: (1 1 1 4 1), (1 1 4 0 2), (1 2 1 0 4), (1 3 1 2 0), (1 3 3 0 0), (2 1 1 1 0), (2 1 2 0 0), (2 2 1 0 0).

Для того, щоб обрати найкращий план проаналізуємо отримані оптимальні плани за наступними критеріями:

1. Кількість з'явлень відповідного плану у відсотках.
2. Відсоток наповненості складу відповідними наборами предметів.
3. Середньоарифметична сумарна вартість.
4. Імовірність того, що отримана сумарна вартість, більше ніж 30000 у.о.
5. Імовірність того, що отримана сумарна вартість, більше ніж 35000 у.о.

Аналіз результатів відповідно даних критеріїв оптимальності наведено у таблицях 2.1–2.5.

Таблиця 2.1 – Аналіз планів за першим критерієм

Набір	Кількість, відсоток		
	1000	5000	10000
1 1 1 4 1	35,20%	36,20%	37,40%
1 1 4 0 2	9,80%	8,00%	8,13%
1 2 1 0 4	20,60%	19,12%	18,01%
1 3 1 2 0	1,70%	2,82%	2,99%
1 3 3 0 0	13,00%	13,08%	12,80%
2 1 1 1 0	0,20%	0,06%	0,10%
2 1 2 0 0	0,10%	0,16%	0,21%
2 2 1 0 0	19,40%	20,56%	20,36%

Таблиця 2.2 – Аналіз планів за другим критерієм

Набір	Наповненість складу, відсоток		
	1000	5000	10000
1 1 1 4 1	100,00%	100,00%	100,00%
1 1 4 0 2	100,00%	100,00%	100,00%
1 2 1 0 4	99,00%	99,00%	99,00%
1 3 3 0 0	100,00%	100,00%	100,00%
2 2 1 0 0	98,00%	98,00%	98,00%

Таблиця 2.3 – Аналіз планів за третім критерієм

Набір	Середнє арифметичне, у.о.		
	1000	5000	10000
1 1 1 4 1	35068,546	35389,93	35266,79
1 1 4 0 2	31672,99	31800,11	31812,89
1 2 1 0 4	33348,62	33392,08	33317,69
1 3 3 0 0	31878,47	32009,67	31890,5
2 2 1 0 0	35960,57	36137,37	36081,14

Таблиця 2.4 – Аналіз планів за четвертим критерієм

Набір	Ймовірність с.в. >30000, відсотки		
	1000	5000	10000
1 1 1 4 1	95,17%	96,63%	96,18%
1 1 4 0 2	80,61%	75,75%	77,98%
1 2 1 0 4	90,78%	92,36%	91,34%
1 3 3 0 0	80,00%	79,82%	79,77%
2 2 1 0 0	97,42%	99,12%	98,62%

Таблиця 2.5 – Аналіз планів за п'ятим критерієм

Набір	Ймовірність с.в. >35000, відсотки		
	1000	5000	10000
1 1 1 4 1	52,84%	55,97%	53,26%
1 2 1 0 4	27,67%	25,84%	26,21%
2 2 1 0 0	67,53%	71,01%	71,27%

Спочатку розглядаємо всі оптимальні плани за першим критерієм, а саме кількості з'явлень. Порівнюючи оптимальні плани за даним критерієм бачимо, що є оптимальні плани, які мають кількість з'явлень менше, ніж 5%. Тому виключимо такі оптимальні плани на даному етапі: (1 3 1 2 0), (2 1 1 1 0), (2 1 2 0 0). Розглядаємо інші критерії без цих оптимальних планів.

За другим критерієм наповненості складу можемо бачити, як оптимальні плани наповнюють склад. На даному етапі всі оптимальні плани показали хороший результат по даному критерію, тому ніякий план не виключаємо.

За критерієм середнім арифметичним бачимо їх середню сумарну вартість, але на даному критерії важко зробити вибір або виключити якийсь план тому, що всі оптимальні плани не дуже сильно відрізняються один від одного. Тому переходимо до наступного критерію.

Розглядаючи критерій ймовірність сумарної вартості більше ніж 30000 у.о. можна побачити оптимальні плани, які дають менше ніж 90%, тому на даному етапі виключимо з рекомендованих такі оптимальних планів: (1 1 4 0 2), (1 3 3 0 0). Переходимо до останнього критерію.

Залишився критерій ймовірність сумарної вартості більше ніж 35000 у.о., розглядаючи даний критерій бачимо, що є оптимальний план (1 2 1 0 4), який дає менше ніж 50%. Тому виключимо його на даному етапі.

Серед всіх оптимальних планів залишились (1,1,1,4,1) та (2,2,1,0,0). Порівнюючи плани, можна побачити, що план (2,2,1,0,0) має кращі показники за критерієм ймовірності сумарної вартості більше 35000 та за середнім арифметичним, а план (1,1,1,4,1) – за критерієм кількості з'явлень та за

наповненістю складу. Порівнюючи ці два плани за критерієм ймовірності сумарної вартості більше 30000 має перевагу план (2,2,1,0,0).

Тому можна рекомендувати два плани в залежності від ситуації. Якщо в даній задачі необхідна стабільна ймовірність з'явлень, то рекомендується оптимальний план (1,1,1,4,1). Якщо необхідна більша середня сумарна вартість, то рекомендуємо оптимальний план (2,2,1,0,0).



### 3 ЗАДАЧА ПРО РЮКЗАК В УМОВАХ НЕВИЗНАЧЕНОСТІ

#### 3.1 Загальна постановка задачі про рюкзак

У гіпотетичний рюкзак (склад, кузов транспортного засобу тощо), вага або об'єм якого не повинні перевищувати  $V$  умовних одиниць (у.о.), завантажуються предмети  $n$  типів. Предмети кожного типу мають вагу або об'єм  $a_i$  та вартість або цінність  $c_i$ . Вага або об'єм одного предмету кожного типу може варіюватися відповідно заданому закону розподілу. Необхідно завантажити рюкзак (склад, кузов транспортного засобу тощо) так, щоб сумарна вага або об'єм завантажених предметів не повинні перевищувати вагу або об'єм рюкзаку  $V$ . Сумарна вартість або сумарна цінність взятих предметів має бути максимальною [8].

#### 3.2 Математична постановка задачі про рюкзак

$$F = \sum_{j=1}^n c_j x_j \rightarrow \max,$$

$$\sum_{j=1}^n a_j x_j \leq V$$

$a_j$  можуть варіюватися відповідно заданому закону розподілу,

де  $n$  – кількість типів предметів;

$x_j$  – кількість предметів типу  $j$ ;

$c_j$  – вартість або цінність одного завантаженого предмету типу  $j$ ;

$a_j$  – вага або об'єм одного завантаженого предмету типу  $j$ ;

$V$  – вага або об'єм рюкзаку;

$F$  – сумарна вартість або цінність завантажених предметів без перевищення ваги або об'єму рюкзаку.

### 3.3 Алгоритм розв'язання задачі про рюкзак в умовах невизначеності

1. Задаємо дані для розв'язання задачі, а саме:

- Об'єм або вагу рюкзаку (склад, кузов транспортного засобу тощо).
- Вагу або об'єм для предметів кожного типу.
- Вартість або цінність для предметів кожного типу.
- Кількість експериментів.
- Місце, де буде знаходитись файл з експериментами.
- Мінімальна кількість предметів кожного типу, яку необхідно завантажити в рюкзак (склад, кузов транспортного засобу тощо).
- Максимальну кількість предметів кожного типу, яку можливо помістити в рюкзак (склад, кузов транспортного засобу тощо) без урахування відхилень для предметів кожного типу.

Щоб дізнатися максимальну кількість предметів кожного типу, потрібно спочатку обрати предмет певного типу. Потім вагу або об'єм рюкзаку поділити на вагу або об'єм предмета обраного типу і з отриманого числа виділити цілу частину. І ця ціла частина буде показувати максимальну кількість предметів обраного типу, яку можливо помістити в рюкзак. Такі розрахунки потрібно потвори для предметів кожного типу.

Програма потім сама розраховує максимальну кількість з урахуванням відхилень для предметів кожного типу.

2. Знайдемо кількість всіх оптимальних планів, які підходять за всіма обмеженнями, а саме за обмеження об'єму або ваги рюкзаку (склад, кузов транспортного засобу тощо), а також обмеження за кількістю тих чи інших типів предметів з умови задачі.

Для того, щоб створити масив потрібно дізнатися скільки оптимальних планів може мати дана задача так, щоб не перевищувати вагу або об'єм рюкзаку (склад, кузов транспортного засобу тощо). Тому була розроблена спеціальна

функція, що підраховує кількість за допомогою повного перебору всіх оптимальних планів.

3. Створимо масив та занесемо всі оптимальні плани, що не перевищують вагу або об'єм рюкзаку (склад, кузов транспортного засобу тощо) до нього.

4. Формуємо випадкові числа ваги або об'єму для предметів кожного типу за допомогою генератора випадкових чисел для рівномірного розподілу з різними відхиленнями.

5. Зробимо повний перебір оптимальних планів для даних випадкових чисел ваги або цінності.

На даному кроці робиться повний перебір оптимальних планів, щоб дізнатись, який план дає найбільшу сумарну вартість або цінність та відповідає усім обмеженням задачі.

6. Обираємо найкращий оптимальний план, який дає найбільшу сумарну вартість, а також задовольняє усім обмеженням задачі, і заносимо його до масиву.

7. Повторюємо крок 4-6 відповідно з вказаною кількістю експериментів.

8. Заносимо всі оптимальні плани, що з'явилися в файл.

9. Заносимо всі сумарні вартості відповідні цим оптимальним планам також в цей файл.

Даний алгоритм є описом роботи програмної реалізації для задачі про рюкзак, що написана на мові програмування C++ [13]. Щоб обрати даний алгоритм в програмі потрібно обрати задача №2 (див. додаток А).

### 3.4 Критерії оптимальності та аналіз оптимального плану задачі про рюкзак

В результаті розв'язку задачі про рюкзак в умовах невизначеності можуть бути такі два варіанта :

- Один оптимальний план.
- Більше ніж один оптимальний план.

Для першого випадку розробка критеріїв оптимальності немає сенсу так як маємо один оптимальний план. Для другого випадку необхідно провести аналіз отриманих оптимальних планів. Для розглянутої задачі про рюкзак можна рекомендувати наступні критерії оптимальності:

1. Кількість з'явлень відповідного плану у відсотках.
2. Максимальна сумарна вартість.

### 3.5 Приклад задачі про рюкзак

Власник може завантажити склад наборами наступних предмети: труби, електродвигуни, спецодяг, автозапчастини та кабелі. Об'єм складу дорівнює 100%. Один набір труб займає 31% об'єму складу, один набір електродвигунів – 13%, один набір спецодягу – 10%, один набір автозапчастин – 9,5% і один набір кабелів – 8%. Вартість одного набору труб дорівнює 10050 у.о., одного набору електродвигунів – 4210 у.о., одного набору спецодягу – 3130 у.о., одного набору автозапчастин – 3000 у.о. і одного набору кабелів – 2600 у.о. На склад обов'язково потрібно завантажити хоча б один набір труб, один набір електродвигунів та один набір спецодягу.

В залежності від комплектації та призначення об'єм одного набору предметів може варіюватися відповідно рівномірному закону розподілу в наступних межах:

- труби – 35%;
- електродвигуни – 20%;
- спецодяг – 15%;
- автозапчастини – 50%;
- кабелі – 30%.

Необхідно завантажити склад так, щоб сумарний об'єм завантажених предметів не повинен перевищувати об'єм рюкзаку та сумарна вартість завантажених предметів була максимальною.

Задача має наступну математичну постановку:

$$F = c_1x_1 + c_2x_2 + c_3x_3 + c_4x_4 + c_5x_5 \rightarrow \max$$

$$\left\{ \begin{array}{l} a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 + a_5x_5 \leq 100 \\ x_i \geq 1, \quad i = \overline{1,3} \\ x_i \geq 0, \quad i = \overline{4,5} \\ 20,15 \leq a_1 \leq 41,85 \\ 10,4 \leq a_2 \leq 15,6 \\ 8,5 \leq a_3 \leq 11,5 \\ 4,75 \leq a_4 \leq 14,25 \\ 5,6 \leq a_5 \leq 10,4 \end{array} \right. ,$$

де  $x_1$  – кількість наборів труб;

$x_2$  – кількість наборів електродвигунів;

$x_3$  – кількість наборів спецодягу;

$x_4$  – кількість наборів автозапчастин;

$x_5$  – кількість наборів кабелів;

$c_1$  – вартість одного набору труб, яка дорівнює 10050 у.о.;

$c_2$  – вартість одного набору електродвигунів, яка дорівнює 4210 у.о.;

$c_3$  – вартість одного набору спецодягу, яка дорівнює 3130 у.о.;

$c_4$  – вартість одного набору автозапчастин яка дорівнює 3000 у.о.;

$c_5$  – вартість одного набору кабелів яка дорівнює 2600 у.о.;

$a_1$  – об'єм одного набору труб у відсотках

$a_2$  – об'єм одного набору електродвигунів у відсотках;

$a_3$  – об'єм одного набору спецодягу у відсотках;

$a_4$  – об'єм одного набору автозапчастин у відсотках;

$a_5$  – об'єм одного набору кабелів у відсотках;

$F$  – сумарна вартість або цінність завантажених предметів без перевищення об'єму складу.

Робимо запуск програми та вибираємо дану задачу. Після закінчення роботи програми на робочому столі з'являються три файли з різною кількістю даних. Для 1000 даних маємо 148 планів, для 5000 даних – 216 планів, для 10000 даних – 242 плани.

Рисунок 3.1 – Результат варіювання об'єму в файлі на 1000 даних

Рисунок 3.2 – Результат варіювання об'єму в файлі на 5000 даних



Рисунок 3.3 – Результат варіювання об'єму в файлі на 10000 даних

Будемо проводити аналіз отриманих результатів за двома критеріями (кількість появлень відповідного плану та максимальна вартість) наступним чином.

Спочатку розглянемо результати 1000 експериментів. Було отримано 148 оптимальних планів. Виберемо чотири плани з найбільшою кількістю появлень та зафіксуємо відповідну їм вартість (табл. 3.1). Далі обираємо чотири плани з максимальною вартістю та фіксуємо відповідну їм кількість появлень (табл. 3.1). Серед обраних планів виберемо максимальний і за кількістю появлень, і за вартістю.

Таблиця 3.1 – Аналіз планів за двома критеріями для 1000 з'явлень

Набір		Кількість з'явлень	Вартість
3 1 1 0 0	За кількістю	4,40%	37490
1 1 1 7 0		4,40%	38390
1 1 1 6 0		3,90%	35390
1 1 1 0 6		3,80%	32990
1 1 1 11 0	За максимальною вартістю	0,40%	50390
1 1 1 10 0		0,80%	47390
3 1 1 3 0		0,10%	46490
2 1 1 6 0		0,10%	45440

Якщо порахувати середню кількість з'явлень на кожен план, то маємо 0,68%. Бачимо, що деякі плани мають більший відсоток, але не можемо брати даний фактор за базис.

Тому проводимо далі аналіз для файлів з 5000 та 10000 появлень.

Проаналізувавши файл бачимо, що з'являється новий конкурент серед попередніх чотирьох планів, тому добавимо новий план та поглянемо на результати попередніх. Така ж сама ситуація з'явилась і для планів за максимальним середнім.

Таблиця 3.2 – Аналіз планів за двома критеріями для 5000 з'явлень

Набір		Кількість з'явлень	Середнє арифметичне
3 1 1 0 0	За кількістю	5,44%	37490
1 1 1 7 0		4,54%	38390
1 1 1 8 0		4,04%	41390
1 1 1 6 0		3,72%	35390
1 1 1 0 6		3,36%	32990
1 1 1 1 1 0	За максимальним середнім ариф.	0,46%	50390
1 1 1 1 0 0		1,32%	47390
3 1 1 3 0		0,02%	46490
1 2 1 8 0		0,02%	45600
2 1 1 6 0		0,08%	45440

Середня кількість з'явлень для кожного плану в файлі з 5000 з'явленнями дорівнює 0,46%. Деякі оптимальні плани підходять по даному фактору, але знову ж таки не взяти його за основу.

Проводимо далі аналіз для файлів з 10000 з'явлень.

Проаналізувавши дані, бачимо, що нового конкурента за кількістю планів не з'явилося, але для максимального середнього з'явилося більше десяти конкурентів, добавимо п'ять оптимальних планів до максимального середнього.



Таблиця 3.3 – Аналіз планів за двома критеріями для 10000 з'явлень

Набір		Кількість з'явлень	Середнє арифметичне
3 1 1 0 0	За кількістю	5,29%	37490
1 1 1 7 0		4,14%	38390
1 1 1 6 0		3,75%	35390
1 1 1 8 0		3,70%	41390
1 1 1 0 6		3,13%	32990
1 1 1 12 0	За максимальним середнім ариф.	0,03%	53390
1 1 2 10 0		0,02%	50520
1 1 1 11 0		0,46%	50390
2 1 1 7 0		0,03%	48440
4 1 1 0 0		0,01%	47540
1 1 2 9 0		0,11%	47520
1 1 1 10 0		1,58%	47390
3 1 1 3 0		0,00%	46490
1 2 1 8 0		0,02%	45600
2 1 1 6 0		0,14%	45440

Середня кількість з'явлень для кожного плану в файлі з 10000 з'явленнями дорівнює 0,41%.

Проводячи аналіз даних можна виділити декілька оптимальних планів для рекомендації:

1. План (3,1,1,0,0) можна рекомендувати, якщо в даній ситуації необхідна стабільність.

2. План (1,1,1,7,0) можна рекомендувати, але план чуть гірше за кількістю з'явлень, ніж попередній, але дає більшу сумарну вартість.

3. План (1,1,1,8,0) є один із найкращий варіантів по відношенню кількість – сумарна вартість.

4. План (1,1,1,10,0) є також один із найкращих варіантів кількості – сумарна вартість. Цей план має не таку стабільність, але має хорошу сумарну вартість.

Підводячи підсумки можливо виділимо декілька факторів:

1. Чим більша кількість даних, тим більше оптимальних планів. Це пов'язано з тим, що 1000, 5000 та 10000 даних мало для того, щоб побачити всі оптимальні плани так як об'єм предметів не є стабільним і кожен раз різний. Тому можливо потрібно збільшити кількість даних. Тоді б з'явилися всі оптимальні плани.

2. Можливо для даної задачі потрібно поставити більш строгі обмеження.

3. Можливо потрібно було розробити декілька критерії, які б були обов'язкові для виконання, щоб рекомендувати даний чи дані плани.

## ВИСНОВКИ

В даній дипломній роботі виконано такі завдання:

- Методами комп'ютерного моделювання досліджено важливі задачі оптимального керування та планування – задачу про завантаження та задачу про рюкзак в умовах невизначеності.
- Описано алгоритм знаходження оптимальних розв'язків задач з вхідними параметрами, що підпорядковуються довільному закону розподілу.
- Розроблену програмну реалізацію на мові програмування C++, що знаходить оптимальні плани задач з рівномірними вхідними параметрами та відповідну їм сумарну вартість, і заносить знайдені дані в файл формату csv для заданої кількості експериментів.
- Розроблено різні варіанти критеріїв оптимальності аналізу розв'язків задачі в умовах невизначеності.
- Отримано результати розрахунку оптимальних розв'язків для прикладів задачі про завантаження та задачі про рюкзак з рівномірно розподіленими параметрами для різних значень відхилень. Проведено аналіз оптимальних планів розглянутих прикладів для різних критеріїв оптимальності.

## ПЕРЕЛІК ПОСИЛАНЬ

1. **Беллман, Р.** Прикладные задачи динамического программирования [Текст] / Р. Беллман, С. Дрейфус. – М. : «Наука», 1965 г. – 460 стр., с илл.
2. **Саати, Т.** Целочисленные методы оптимизации и связанные с ними экстремальные проблемы [Текст] / Т. Саати – М. : «Мир», 1973 г. – 305 стр., с илл.
3. **Корбут, А.А.** Дискретное программирование [Текст] / А.А. Корбут, Ю.Ю. Финкельштейн – М. : «Наука», 1969 г. – 368 стр., с илл.
4. **Окулов, С.М.** Динамическое программирование [Электронный ресурс] / С.М. Окулов, О.А. Пестов. – М. : БИНОМ. Лаборатория знаний, 2012 г. – 296 стр., с илл.
5. **Таха, Х.А.** Введение в исследование операций [Текст] / Х. А. Таха. – 7-е издание. : пер. с англ. – М.: Издательский дом «Вильямс», 2005. – 912 с.: ил. – Парал. тит. англ.
6. **Левитин, А.В.** Алгоритмы: введение в разработку и анализ. : Пер. с англ. – М. : Издательский дом «Вильямс», 2006. – 576 с. : ил. – Парал. тит. англ.
7. **Кагиров, Р.Р.** Многомерная задача о рюкзаке: новые методы решения [Текст] / Р. Р. Кагиров // Вестник Сибирского государственного аэрокосмического университета имени академика М. Ф.Решетнева. Математика, механика, информатика. – 2007 г. – №3. – с. 16–20.
8. **Lisser, A.** Stochastic Quadratic Knapsack with Recourse [Текст] / A. Lisser, R. Lopez // Electronic Notes in Discrete Mathematics. Vol.36 – 2010 p. – с. 97 – 104.
9. **Канцедал, С.А.** Конструювання й дослідження алгоритмів рішення задачі про рюкзак [Текст] / С. А. Канцедал, М. В. Костікова, І. В. Скрипіна // Автомобільний транспорт. Математика – 2015 р. – №36 – с. 154 – 160.
10. **Куприяшин, М.А.** Исследование алгоритма точного решения задачи о рюкзаке методом динамического программирования [Текст] / М.А.

Куприяшин, Г.И. Борзунов // Вестник Пермского национального исследовательского политехнического университета. Электротехника, информационные технологии, системы управления – 2016 г. – №17 – с. 121 – 130.

11. **Васильчиков, В.В.** О рекурсивно-параллельном алгоритме решения задачи о рюкзаке [Текст] / В.В. Васильчиков // Вестник Моделирование и анализ информационных систем. Т.25 – 2017 г. – №2 – с. 155 – 164.

12. **Топка, В.В.** Многомерная задача о рюкзаке: эффективный метод решения и возможные приложения [Текст] / В.В. Топка // Труды ИСА РАН. Том 69 – 2019 г. – №2 – с. 54 – 64.

13. **Прата, С.** Язык программирования C++ (C++11). Лекции и упражнения, 6-е издание [Текст] / С. Прата – Пер. с англ. — М. : ООО «И.Д.Вильямс», 2012. — 1248 с. : ил. – Парал. тит. англ.

14. **Джеймс, Г.** Введение в статистическое обучение с примерами на языке R [Текст] / Г. Джеймс, Д. Уиттон, Т. Хасти, Р. Тибширани – Пер. с англ. С. Э. Мастицкого – М.: ДМК Пресс, 2016. – 450 с.: ил.

## ДОДАТОК А

### Програмный код C++

```

#include<iostream>
#include<random>
#include<fstream>
#include<sstream>
#include<string>
#include<direct.h>
#pragma warning(disable : 4996)
using namespace std;
int How_Many_Combination_This_Things(int* x_max, int* x_min, double* type_x, int
count_x, double sum_type_x){
    int* number_x = new int[count_x];
    for (int i = 0; i < count_x; i++) {
        number_x[i] = x_min[i]; //0
    }
    int general = 0, exit_with_while = 0;
    double general_sum_combi;
    bool true_if = false;
    while (true) {

        general_sum_combi = 0;
        if (true_if == true) {
            number_x[count_x - 1] += 1;
        }
        else {
            true_if = true;
        }
        for (int i = 0; i < count_x; i++) {
            general_sum_combi += number_x[i] * type_x[i];
        }
        if (general_sum_combi <= sum_type_x) {//==
            general++;
        }
        for (int i = 1; i < count_x; i++) {
            if (number_x[i] > x_max[i]) {
                number_x[i - 1] += 1;
                number_x[i] = x_min[i]; //0
                true_if = false;
            }
        }
        for (int i = 0; i < count_x; i++) {
            if (number_x[i] == x_max[i]) {
                exit_with_while++;
            }
        }
        if (exit_with_while == count_x) {
            break;
        }
        else {
            exit_with_while = 0;
        }
    }
    return general;
}

void Write_Combination_This_Things(int* x_max, int* x_min, double* type_x, int
count_x, double sum_type_x, int** count_sample) {
    int* number_x = new int[count_x];

```

```

for (int i = 0; i < count_x; i++) {
    number_x[i] = x_min[i]; //0
}
int general = 0, exit_with_while = 0;
double general_sum_combi;
bool true_if = false;
while (true) {
    general_sum_combi = 0;
    if (true_if == true) {
        number_x[count_x - 1] += 1;
    }
    else {
        true_if = true;
    }
    for (int i = 0; i < count_x; i++) {
        general_sum_combi += number_x[i] * type_x[i];
    }
    if (general_sum_combi <= sum_type_x) { //==
        for (int i = 0; i < count_x; i++) {
            count_sample[general][i] = number_x[i];
        }
        general++;
    }
    for (int i = 1; i < count_x; i++) {
        if (number_x[i] > x_max[i]) {
            number_x[i - 1] += 1;
            number_x[i] = x_min[i]; //0
            true_if = false;
        }
    }
    for (int i = 0; i < count_x; i++) {
        if (number_x[i] == x_max[i]) {
            exit_with_while++;
        }
    }
    if (exit_with_while == count_x) {
        break;
    }
    else {
        exit_with_while = 0;
    }
}
}
void Problem_1() {

    int count_base[3] = { 1000,5000,10000 };

    string path;
    path = "C:/Users/crazy/Desktop/Вартість";
    int ko = path.length();
    char* pathh = new char[ko + 1];
    strcpy(pathh, path.c_str());

    int count_x = 5;

    double type_x[5] = { 31, 13, 10, 9.5, 8 };

    cout << "Об'єм кожного предмету" << endl;
    for (int i = 0; i < count_x; i++)
        cout << type_x[i] << " ";
    cout << endl;

    cout << "Об'єм складу" << endl;
    double sum_type_x = 100;
    cout << sum_type_x << endl;
}

```

```

cout << "Вартість кожного предмету" << endl;
double cost_x[5] = { 10050,4210,3130,3000,2600 };

for (int i = 0; i < count_x; i++)
    cout << cost_x[i] << " ";
cout << endl;

double* C = new double[count_x];

int x_max[5] = { 3,7,10,10,12 };
int x_min[5] = { 1,1,1,0,0 };

int general = How_Many_Combination_This_Things(x_max, x_min, type_x, count_x,
sum_type_x);

int** count_sample = new int* [general];
for (int i = 0; i < general; i++) {
    count_sample[i] = new int[count_x];
}

Write_Combination_This_Things(x_max, x_min, type_x, count_x, sum_type_x,
count_sample);

int* x = new int[count_x];
int* x_opt = new int[count_x];

double f, f_opt;
int* num_mass = new int[general];
int num_while, _j;
double _sum_type_x;
mkdir(pathh);

string gyper[] = { path + "/" + to_string(count_base[0]) + ".csv" ,
    path + "/" + to_string(count_base[1]) + ".csv" ,
    path + "/" + to_string(count_base[2]) + ".csv" };
int count_while = 0;
int count_right = 0;
while (count_while < 3) { //количество файлов //3
    double** mass = new double* [count_base[count_while]]; //1000 //count_base
    for (int i = 0; i < count_base[count_while]; i++) //count_base
        mass[i] = new double[general]; //general
    for (int i = 0; i < general; i++) //general
        num_mass[i] = 0;
    num_while = 0;
    while (num_while < count_base[count_while]) { //1000 //count_base
        random_device rd;
        mt19937 gen(rd());

        uniform_real_distribution<> rndC1(10050 - 10050 * 0.35, 10050 + 10050 *
0.35);
        C[0] = rndC1(gen);
        uniform_real_distribution<> rndC2(4210 - 4210 * 0.20, 4210 + 4210 *
0.20);
        C[1] = rndC2(gen);
        uniform_real_distribution<> rndC3(3130 - 3130 * 0.15, 3130 + 3130 *
0.15);
        C[2] = rndC3(gen);
        uniform_real_distribution<> rndC4(3000 - 3000 * 0.5, 3000 + 3000 * 0.5);
        C[3] = rndC4(gen);
        uniform_real_distribution<> rndC5(2600 - 2600 * 0.30, 2600 + 2600 *
0.30);
        C[4] = rndC5(gen);

        for (int i = 0; i < count_x; i++)

```



```

        x_opt[i] = x_min[i]; //0
    for (int i = 0; i < count_x; i++)
        x[i] = x_min[i]; //0
    f = 0;
    f_opt = -1000;
    _j = count_x - 1;
    _sum_type_x = 0;
    while (true) {
        if (x[_j] <= x_max[_j]) {
            for (int i = 0; i < count_x; i++) {
                _sum_type_x += type_x[i] * x[i];
            }
            if (_sum_type_x <= sum_type_x) {
                for (int i = 0; i < count_x; i++) {
                    f += C[i] * x[i];
                }
                if (f_opt <= f) {
                    f_opt = f;
                    for (int i = 0; i < count_x; i++)
                        x_opt[i] = x[i];
                }
            }
            _j = count_x - 1;
        }
        else {
            x[_j] = x_min[_j]; //0
            _j = _j - 1;
            if (_j < 0) {
                break;
            }
        }
        x[_j] += 1;
        _sum_type_x = 0;
        f = 0;
    }
    count_right = 0;
    for (int i = 0; i < general; i++) {
        for (int k = 0; k < count_x; k++) {
            if (count_sample[i][k] == x_opt[k])
                count_right++;
        }
        if (count_right == count_x) {
            mass[num_mass[i]][i] = f_opt;
            num_mass[i]++;
            count_right = 0;
        }
        else {
            count_right = 0;
        }
    }
    num_while++;
}

int max = 0;
for (int i = 0; i < general; i++) {
    if (max < num_mass[i])
        max = num_mass[i];
}

ofstream myFile(gyper[count_while]);
for (int i = 0; i < general; i++) {
    if (num_mass[i] != 0) {
        myFile << "_";
        for (int k = 0; k < count_x; k++) {

```

```

        myFile << count_sample[i][k];
    }
    if (i != general - 1) {
        myFile << "_,";
    }
}
if (i == general - 1) {
    myFile << "_";
}
}
myFile << "\n";

for (int i = 0; i < max; i++) {
    for (int mk = 0; mk < general; mk++) {
        if (num_mass[mk] != 0) {
            if (num_mass[mk] > i) {
                myFile << mass[i][mk];
            }
            if (mk != general - 1)
                myFile << ",";
        }
        if (mk == general - 1)
            myFile << "\n";
    }
}
myFile.close();

count_while++;
}
}

void Problem_2() {

    int count_base[3] = { 1000,5000,10000 };

    string path;
    path = "C:/Users/crazy/Desktop/06'ем";
    int ko = path.length();
    char* pathh = new char[ko + 1];
    strcpy(pathh, path.c_str());

    int count_x = 5;

    double type_x[5] = { 31, 13, 10, 9.5, 8 };

    cout << "06'ем кожного предмету" << endl;
    for (int i = 0; i < count_x; i++)
        cout << type_x[i] << " ";
    cout << endl;

    cout << "06'ем складу" << endl;
    double sum_type_x = 100;
    cout << sum_type_x << endl;

    cout << "Вартість кожного предмету" << endl;
    double cost_x[5] = { 10050,4210,3130,3000,2600 };

    for (int i = 0; i < count_x; i++)
        cout << cost_x[i] << " ";
    cout << endl;

    double* C = new double[count_x];

    int* x_max = new int[count_x];
    x_max[0] = sum_type_x / (type_x[0] - type_x[0] * 0.35);
}

```

```

x_max[1] = sum_type_x / (type_x[1] - type_x[1] * 0.2);
x_max[2] = sum_type_x / (type_x[2] - type_x[2] * 0.15);
x_max[3] = sum_type_x / (type_x[3] - type_x[3] * 0.5);
x_max[4] = sum_type_x / (type_x[4] - type_x[4] * 0.3);

int x_min[5] = { 1,1,1,0,0 };

double type_x[5];
type_x[0] = type_x[0] - type_x[0] * 0.35;
type_x[1] = type_x[1] - type_x[1] * 0.2;
type_x[2] = type_x[2] - type_x[2] * 0.15;
type_x[3] = type_x[3] - type_x[3] * 0.5;
type_x[4] = type_x[4] - type_x[4] * 0.3;
int general = How_Many_Combination_This_Things(x_max, x_min, type_x, count_x,
sum_type_x);

int** count_sample = new int* [general];
for (int i = 0; i < general; i++) {
    count_sample[i] = new int[count_x];
}

Write_Combination_This_Things(x_max, x_min, type_x, count_x, sum_type_x,
count_sample);

int* x = new int[count_x];
int* x_opt = new int[count_x];

double f, f_opt;
int* num_mass = new int[general];
int num_while, _j;
double _sum_type_x;
mkdir(pathh);

string gyper[] = { path + "/" + to_string(count_base[0]) + ".csv" ,
    path + "/" + to_string(count_base[1]) + ".csv" ,
    path + "/" + to_string(count_base[2]) + ".csv" };
int count_while = 0;
int count_right = 0;
while (count_while < 3) { //количество файлов //3
    double** mass = new double* [count_base[count_while]]; //1000 //count_base
    for (int i = 0; i < count_base[count_while]; i++) //count_base
        mass[i] = new double[general]; //general
    for (int i = 0; i < general; i++) //general
        num_mass[i] = 0;
    num_while = 0;
    while (num_while < count_base[count_while]) { //1000 //count_base
        random_device rd;
        mt19937 gen(rd());

        uniform_real_distribution<> rndC1(31 - 31 * 0.35, 31 + 31 * 0.35);
        C[0] = rndC1(gen);
        uniform_real_distribution<> rndC2(13 - 13 * 0.20, 13 + 13 * 0.20);
        C[1] = rndC2(gen);
        uniform_real_distribution<> rndC3(10 - 10 * 0.15, 10 + 10 * 0.15);
        C[2] = rndC3(gen);
        uniform_real_distribution<> rndC4(9.5 - 9.5 * 0.5, 9.5 + 9.5 * 0.5);
        C[3] = rndC4(gen);
        uniform_real_distribution<> rndC5(8 - 8 * 0.30, 8 + 8 * 0.30);
        C[4] = rndC5(gen);

        for (int i = 0; i < count_x; i++)
            x_opt[i] = x_min[i]; //0
        for (int i = 0; i < count_x; i++)
            x[i] = x_min[i]; //0
        f = 0;
    }
}

```

```

f_opt = -1000;
_j = count_x - 1;
_sum_type_x = 0;
while (true) {
    if (x[_j] <= x_max[_j]) {
        for (int i = 0; i < count_x; i++) {
            _sum_type_x += C[i] * x[i];
        }
        if (_sum_type_x <= sum_type_x) {
            for (int i = 0; i < count_x; i++) {
                f += cost_x[i] * x[i];
            }
            if (f_opt <= f) {
                f_opt = f;
                for (int i = 0; i < count_x; i++)
                    x_opt[i] = x[i];
            }
        }
        _j = count_x - 1;
    }
    else {
        x[_j] = x_min[_j]; //0
        _j = _j - 1;
        if (_j < 0) {
            break;
        }
    }
    x[_j] += 1;
    _sum_type_x = 0;
    f = 0;
}
count_right = 0;
for (int i = 0; i < general; i++) {
    for (int k = 0; k < count_x; k++) {
        if (count_sample[i][k] == x_opt[k])
            count_right++;
    }
    if (count_right == count_x) {
        mass[num_mass[i]][i] = f_opt;
        num_mass[i]++;
        count_right = 0;
    }
    else {
        count_right = 0;
    }
}
num_while++;
}

int max = 0;
for (int i = 0; i < general; i++) {
    if (max < num_mass[i])
        max = num_mass[i];
}

int count_plan = 0;
ofstream myFile(gyper[count_while]);
for (int i = 0; i < general; i++) {
    if (num_mass[i] != 0) {
        count_plan++;
        myFile << "_";
        for (int k = 0; k < count_x; k++) {
            myFile << count_sample[i][k];
        }
    }
}

```

```

        if (i != general - 1) {
            myFile << "_,";
        }
    }
    if (i == general - 1) {
        myFile << "_";
    }
}
myFile << "\n";

cout << count_plan << endl;

for (int i = 0; i < max; i++) {
    for (int mk = 0; mk < general; mk++) {
        if (num_mass[mk] != 0) {
            if (num_mass[mk] > i) {
                myFile << mass[i][mk];
            }
            if (mk != general - 1)
                myFile << ",";
        }
        if (mk == general - 1)
            myFile << "\n";
    }
}
myFile.close();

count_while++;
}
}

void Problem_3() {

    int count_base[3] = { 1000,5000,10000 };

    string path;
    path = "C:/Users/crazy/Desktop/06'єм і Вартість";
    int ko = path.length();
    char* pathh = new char[ko + 1];
    strcpy(pathh, path.c_str());

    int count_x = 5;

    double type_x[5] = { 31, 13, 10, 9.5, 8 };

    cout << "06'єм кожного предмету" << endl;
    for (int i = 0; i < count_x; i++)
        cout << type_x[i] << " ";
    cout << endl;

    cout << "06'єм складу" << endl;
    double sum_type_x = 100;
    cout << sum_type_x << endl;

    cout << "Вартість кожного предмету" << endl;
    double cost_x[5] = { 10050,4210,3130,3000,2600 };
    for (int i = 0; i < count_x; i++)
        cout << cost_x[i] << " ";
    cout << endl;

    double* C = new double[count_x];
    double* C1 = new double[count_x];

    int* x_max = new int[count_x];
    x_max[0] = sum_type_x / (type_x[0] - type_x[0] * 0.35);
}
}

```

```

x_max[1] = sum_type_x / (type_x[1] - type_x[1] * 0.2);
x_max[2] = sum_type_x / (type_x[2] - type_x[2] * 0.15);
x_max[3] = sum_type_x / (type_x[3] - type_x[3] * 0.5);
x_max[4] = sum_type_x / (type_x[4] - type_x[4] * 0.3);
cout << x_max[0] << " " << x_max[1] << " " << x_max[2] << " " << x_max[3] << " "
<< x_max[4] << endl;
int x_min[5] = { 1,1,1,0,0 };

double type_x[5];
type_x[0] = type_x[0] - type_x[0] * 0.35;
type_x[1] = type_x[1] - type_x[1] * 0.2;
type_x[2] = type_x[2] - type_x[2] * 0.15;
type_x[3] = type_x[3] - type_x[3] * 0.5;
type_x[4] = type_x[4] - type_x[4] * 0.3;
int general = How_Many_Combination_This_Things(x_max, x_min, type_x, count_x,
sum_type_x);

int** count_sample = new int* [general];
for (int i = 0; i < general; i++) {
    count_sample[i] = new int[count_x];
}

Write_Combination_This_Things(x_max, x_min, type_x, count_x, sum_type_x,
count_sample);

int* x = new int[count_x];
int* x_opt = new int[count_x];

double f, f_opt;
int* num_mass = new int[general];
int num_while, _j;
double _sum_type_x;
mkdir(pathh);

string gyper[] = { path + "/" + to_string(count_base[0]) + ".csv" ,
    path + "/" + to_string(count_base[1]) + ".csv" ,
    path + "/" + to_string(count_base[2]) + ".csv" };
int count_while = 0;
int count_right = 0;
while (count_while < 3) { //количество файлов //3
    double** mass = new double* [count_base[count_while]]; //1000 //count_base
    for (int i = 0; i < count_base[count_while]; i++) //count_base
        mass[i] = new double[general]; //general
    for (int i = 0; i < general; i++) //general
        num_mass[i] = 0;
    num_while = 0;
    while (num_while < count_base[count_while]) { //1000 //count_base
        random_device rd;
        mt19937 gen(rd());

        uniform_real_distribution<> rndC1(31 - 31 * 0.35, 31 + 31 * 0.35);
        C[0] = rndC1(gen);
        uniform_real_distribution<> rndC2(13 - 13 * 0.20, 13 + 13 * 0.20);
        C[1] = rndC2(gen);
        uniform_real_distribution<> rndC3(10 - 10 * 0.15, 10 + 10 * 0.15);
        C[2] = rndC3(gen);
        uniform_real_distribution<> rndC4(9.5 - 9.5 * 0.5, 9.5 + 9.5 * 0.5);
        C[3] = rndC4(gen);
        uniform_real_distribution<> rndC5(8 - 8 * 0.30, 8 + 8 * 0.30);
        C[4] = rndC5(gen);

        uniform_real_distribution<> rndC11(10050 - 10050 * 0.35, 10050 + 10050 *
0.35);
        C1[0] = rndC11(gen);

```

```

uniform_real_distribution<> rndC22(4210 - 4210 * 0.20, 4210 + 4210 *
0.20);
C1[1] = rndC22(gen);
uniform_real_distribution<> rndC33(3130 - 3130 * 0.15, 3130 + 3130 *
0.15);
C1[2] = rndC33(gen);
uniform_real_distribution<> rndC44(3000 - 3000 * 0.5, 3000 + 3000 *
0.5);
C1[3] = rndC44(gen);
uniform_real_distribution<> rndC55(2600 - 2600 * 0.30, 2600 + 2600 *
0.30);
C1[4] = rndC55(gen);

for (int i = 0; i < count_x; i++)
    x_opt[i] = x_min[i]; //0
for (int i = 0; i < count_x; i++)
    x[i] = x_min[i]; //0
f = 0;
f_opt = -1000;
_j = count_x - 1;
_sum_type_x = 0;
while (true) {
    if (x[_j] <= x_max[_j]) {
        for (int i = 0; i < count_x; i++) {
            _sum_type_x += C[i] * x[i]; //type_x[i] //C[i]
        }
        if (_sum_type_x <= sum_type_x) { //sum_type_x //3
            for (int i = 0; i < count_x; i++) {
                f += C1[i] * x[i]; //cost_x[i] //C1[i]
            }
            if (f_opt <= f) {
                f_opt = f;
                for (int i = 0; i < count_x; i++)
                    x_opt[i] = x[i];
            }
        }
        _j = count_x - 1;
    }
    else {
        x[_j] = x_min[_j]; //0
        _j = _j - 1;
        if (_j < 0) {
            break;
        }
    }
    x[_j] += 1;
    _sum_type_x = 0;
    f = 0;
}
count_right = 0;
for (int i = 0; i < general; i++) {
    for (int k = 0; k < count_x; k++) {
        if (count_sample[i][k] == x_opt[k])
            count_right++;
    }
    if (count_right == count_x) {
        mass[num_mass[i]][i] = f_opt;
        num_mass[i]++;
        count_right = 0;
    }
    else {
        count_right = 0;
    }
}
}

```

```

        num_while++;
    }

    int max = 0;
    for (int i = 0; i < general; i++) {
        if (max < num_mass[i])
            max = num_mass[i];
    }

    int count_plan = 0;
    ofstream myFile(gyper[count_while]);
    for (int i = 0; i < general; i++) {
        if (num_mass[i] != 0) {
            count_plan++;
            myFile << "_";
            for (int k = 0; k < count_x; k++) {
                myFile << count_sample[i][k];
            }
            if (i != general - 1) {
                myFile << "_";
            }
        }
        if (i == general - 1) {
            myFile << "_";
        }
    }
    myFile << "\n";

    cout << count_plan << endl;

    for (int i = 0; i < max; i++) {
        for (int mk = 0; mk < general; mk++) {
            if (num_mass[mk] != 0) {
                if (num_mass[mk] > i) {
                    myFile << mass[i][mk];
                }
                if (mk != general - 1)
                    myFile << ",";
            }
            if (mk == general - 1)
                myFile << "\n";
        }
    }
    myFile.close();

    count_while++;
}
}
int main()
{
    setlocale(LC_ALL, "Ukrainian");
    cout << "Виберіть варіант задачі" << endl;
    int p;
    cin >> p;
    if (p == 1) {
        Problem_1();
    }
    if (p == 2) {
        Problem_2();
    }
    if (p == 3) {
        Problem_3();
    }
    return 0;
}

```