МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»

Кафедра інженерії програмного забезпечення

**КУРСОВИЙ ПРОЕКТ**

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

з дисципліни: «Математичні методи дослідження операцій»

на тему:

**«Розв’язання задачі математичного програмування»**

студента ІII курсу групи ІПЗ-21-4

спеціальності 121 «Інженерія програмного забезпечення»

Савченко Олександра Олександровича

(прізвище, ім’я та по-батькові)

Керівник ст. викладач кафедри ІПЗ

Локтікова Т.М.

Дата захисту: "\_\_" \_\_\_\_\_\_ 2024 р.

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оцінка: ECTS \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Локтікова Т.М \_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Кушнір Н.О. .

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Полторак В.П. .

(підпис) (прізвище та ініціали)

Житомир – 2024

Міністерство освіти і науки України

Державний університет «Житомирська політехніка»

Факультет інформаційно-комп’ютерних технологій

Кафедра інженерії програмного забезпечення

ЗАТВЕРДЖУЮ

Зав. каф. ІПЗ, д.п.н., проф.

\_\_\_\_\_\_\_\_\_\_\_\_ Тетяна ВАКАЛЮК

*(підпис)*

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 р.

**ЗАВДАННЯ**

**до курсової роботи**

студенту *3-го курсу групи ІПЗ-21-4 Олександру САВЧЕНКУ*

1. **Тема роботи:** Розв’язання задачі математичного програмування (МП).

2. **Термін здачі** студентом закінченої роботи: червня 2024 р.

3. **Вихідні дані** до роботи: для будівництва будинків на 100 будівельних майданчиках вибрали 5 типових проектів. За кожним з проектів відомі тривалість закладки фундаментів і будівництва решти частин споруди, а також житлова площа споруди (табл.1). Паралельно можна виконувати закладку 10 фундаментів і будівництво 15 будинків.

Скласти план будівництва, який максимізує введення житлової площі протягом року (300 робочих днів), за умови, що будинків 2 типу має бути побудовано не менше 10.

Таблиця 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Види робіт | Тривалість виконання (дні) для типового проекту | | | | |
| 1 проект | 2 проект | 3 проект | 4 проект | 5 проект |
| Закладка фундаменту | 20 | 30 | 35 | 30 | 40 |
| Решта робіт | 40 | 20 | 60 | 35 | 25 |
| Житлова площа, м2 | 3000 | 2000 | 5000 | 4000 | 6000 |

4. **Перелік питань, які мають бути розроблені:** 1) Вступ, 2) Побудова математичної моделі задачі, 3) Вибір й обґрунтування методу розв’язання задачі МП, 4) Розв’язання задачі МП вручну за допомогою вибраного методу, 5) Розробка алгоритму розв’язання задачі МП, 6) Вибір й обґрунтування інструментальних засобів для реалізації програми розв’язання задачі МП, 7) Написання програми розв’язання задачі МП, 8) Тестування програмного продукту, 9) Висновки, 10) Джерела інформації, 11) Додатки.

5. **Перелік графічного матеріалу:** алгоритм розв’язання задачі МП, скріншоти роботи програми.

6. **Дата видачі завдання:** « » лютого 2024 р.

**Завдання прийняв до виконання** студент групи ІПЗ-21-4

Олександр САВЧЕНКО

**Керівник курсової роботи,** ст. викл. каф. ІПЗ Тамара ЛОКТІКОВА

РЕФЕРАТ

Пояснювальна записка до курсового проекту на тему «Розв’язання задачі математичного програмування» складається зі вступу, трьох розділів,

висновків, джерел інформації та додатків.

Текстова частина викладена на 53 сторінках друкованого тексту.

Джерела інформації містять 5 найменуваннь і займають 1 сторінку. Пояснювальна записка має 19 сторінок додатків. У роботі наведено 30

рисунків.

Загальний обсяг роботи – 74 сторінок.

У першому розділі було проаналізовано завдання та складено математичну модель.

У другому розділі обрано метод розв’язання задачі математичного програмування та розв’язано задачу вручну.

У третьому розділі наведено метод реалізації програми, вибір засобів реалізації, проектування схеми роботи алгоритмів та проведено тестування програмного забезпечення.

Висновки містять у собі результати виконаної роботи з розв’язання задачі математичного програмування та створення програми розв’язання задачі за даними, введеними користувачем.

У додатках представлений лістинг створеної програми.

**ЗМІСТ**

[ВСТУП 5](#_Toc163412892)

[1. ПОБУДОВА МАТЕМАТИЧНОЇ МОДЕЛІ ЗАДАЧІ 7](#_Toc163412893)

[2. РОЗВ’ЯЗАННЯ ЗАДАЧІ МП 10](#_Toc163412894)

[2.1 Вибір й обґрунтування методу розв’язання задачі 10](#_Toc163412895)

[2.2 Розв’язання задачі МП 12](#_Toc163412896)

[3. РОЗРОБКА ПРОГРАМИ ДЛЯ РОЗВ’ЯЗАННЯ ЗАДАЧІ МП 21](#_Toc163412897)

[3.1 Аналіз методів реалізації та опис алгоритмів 21](#_Toc163412898)

[3.2 Вибір й обґрунтування інструментальних засобів для реалізації програмного додатку 23](#_Toc163412899)

[3.3 Розробка програмного додатку 25](#_Toc163412900)

[3.4 Тестування програмного додатку 37](#_Toc163412901)

[ВИСНОВКИ 53](#_Toc163412902)

[СПИСОК ДЖЕРЕЛ 54](#_Toc163412903)

[ДОДАТКИ 55](#_Toc163412904)

## ВСТУП

Математичні методи дослідження операцій є фундаментальною дисципліною, яка знаходить широке застосування у різних галузях науки і техніки, економіці, логістиці, менеджменті та інших сферах. Вони забезпечують аналітичні інструменти для моделювання і оптимізації складних систем, що дозволяє приймати обґрунтовані рішення в умовах обмежених ресурсів і невизначеності.

Однією з центральних тем математичних методів дослідження операцій є математичне програмування, яке займається розв’язанням задач оптимізації, де необхідно знайти найкраще (за певним критерієм) рішення з множини можливих. Задачі математичного програмування поділяються на лінійні, нелінійні, цілочислові, динамічні та інші типи, кожен з яких вимагає спеціальних підходів і методів для їх вирішення.

У цій курсовій роботі розглядається проблема розв’язання задачі математичного програмування. Мета дослідження полягає у вивченні теоретичних основ математичного програмування, аналізі методів розв’язання таких задач та їх практичному застосуванні. Особлива увага приділяється лінійному програмуванню як одному з найбільш поширених і досліджених підходів, а також методам його розв’язання, таким як симплекс-метод, метод внутрішніх точок та інші.

Актуальність даної теми обумовлена широким спектром застосувань математичного програмування у реальних практичних задачах. Зокрема, методи математичного програмування використовуються для оптимізації виробничих процесів, управління запасами, планування транспортних перевезень, фінансового аналізу, розробки стратегій маркетингу та багатьох інших напрямків. Ефективне вирішення таких задач сприяє підвищенню продуктивності, зниженню витрат та покращенню загальної ефективності діяльності організацій.

Таким чином, метою даної курсової роботи є не тільки теоретичне обґрунтування методів розв’язання задач математичного програмування, але й їх практичне застосування на конкретних прикладах, що дозволить продемонструвати їх ефективність і доцільність у реальних умовах.

## ПОБУДОВА МАТЕМАТИЧНОЇ МОДЕЛІ ЗАДАЧІ

**Вихідні дані** до роботи: для будівництва будинків на 100 будівельних майданчиках вибрали 5 типових проектів. За кожним з проектів відомі тривалість закладки фундаментів і будівництва решти частин споруди, а також житлова площа споруди (табл.1). Паралельно можна виконувати закладку 10 фундаментів і будівництво 15 будинків.

Скласти план будівництва, який максимізує введення житлової площі протягом року (300 робочих днів), за умови, що будинків 2 типу має бути побудовано не менше 10.

Таблиця 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Види робіт | Тривалість виконання (дні) для типового проекту | | | | |
| 1 проект | 2 проект | 3 проект | 4 проект | 5 проект |
| Закладка фундаменту | 20 | 30 | 35 | 30 | 40 |
| Решта робіт | 40 | 20 | 60 | 35 | 25 |
| Житлова площа, м2 | 3000 | 2000 | 5000 | 4000 | 6000 |

Обираємо як змінні кількість побудованих будинків за кожним з типових проектів.

Для того щоб максимізувати введення житлової площі нам потрібно досягти максимальної сумми житлової площі помноженої на кількість побудованих будинків за кожним з проектів. Отримаємо цільову функцію:

Обмеження:

Так як паралельно можна виконувати закладку тільки 10 фундаментів то за рік (300 робочих днів) можливо виконати тільки 300 \* 10 = 3000 умовних днів закладення фундаменту. Нам потрібно не перевищити це значення, тому сума виконаних закладень фундаментів повинна бути менше або дорівнювати значення отриманому вище:

Паралельно можна виконувати тільки 15 решт робіт то за рік (300 робочих днів) можливо виконати тільки 300 \* 15 = 4500 умовних днів виконання решт робіт. Нам потрібно не перевищити це значення, тому сума виконаних решт робіт повинна бути менше або дорівнювати значення отриманому вище:

Маємо 100 будівельних майданчиків, потрібно побувати будинок на кожному з майданчиків, тому сума побудованих будинків за кожним з проектів повинна дорівнювати 100:

В умові описана додаткова умова, будинків другого типу повинно бути не менше 10:

Із практичних міркувань додамо умову невідємності:

Так як будинок потрібно побудувати цілком, маємо умову цілочисельності змінних:

Отримаємо математичну модель:

**Висновки до розділу 1**

У першому розділі було проведено детальний аналіз задачі будівництва будинків на 100 будівельних майданчиках з використанням п'яти типових проектів. На основі вихідних даних, включаючи тривалість виконання робіт для кожного проекту та житлову площу споруд, сформовано математичну модель задачі. Модель включає цільову функцію, яка максимізує загальну житлову площу, та набір обмежень, що враховують обмеження на кількість одночасно виконуваних робіт та кількість будинків певного типу.

## РОЗВ’ЯЗАННЯ ЗАДАЧІ МП

## ****Вибір й обґрунтування методу розв’язання задачі****

Для розв'язання задачі математичного програмування, сформованої у попередньому розділі, обрано метод Гоморі. Метод Гоморі є одним з методів розв'язання задач цілочислового програмування і базується на використанні симплекс-методу та спеціальних зрізів для знаходження оптимальних цілочислових розв'язків.

2.1. Основні принципи методу Гоморі

Метод Гоморі включає наступні кроки:

1. Розв'язання лінійної задачі: На першому етапі розв'язується задача лінійного програмування без урахування цілочислових обмежень за допомогою симплекс-методу. Якщо отримані оптимальні розв'язки вже є цілочисловими, то задача вважається розв'язаною.

2. Формування зрізів Гоморі: Якщо оптимальні розв'язки не є цілочисловими, будується зріз Гоморі для кожного дробового базисного змінного. Зрізи є додатковими обмеженнями, які поступово виключають дробові частини розв'язків з допустимої області.

3. Додавання зрізів та повторне розв'язання: Додаються отримані зрізи до системи обмежень, після чого задача знову розв'язується симплекс-методом. Цей процес повторюється доти, доки не буде знайдено цілочисловий оптимальний розв'язок.

2.2. Переваги методу Гоморі

Метод Гоморі дозволяє ефективно розв'язувати задачі цілочислового програмування, забезпечуючи точність і надійність отриманих розв'язків. Завдяки використанню зрізів Гоморі, цей метод поступово звужує область допустимих розв'язків, виключаючи дробові частини і наближаючись до оптимальних цілочислових значень. Це дозволяє досягати оптимальних рішень навіть для складних задач з великою кількістю змінних та обмежень.

2.3. Можливі альтернативні методи

Окрім методу Гоморі, для розв'язання задач цілочислового програмування можуть використовуватись й інші методи. Серед них:

1. Метод відгалужень і меж (Branch and Bound): Цей метод базується на побудові дерева рішень, де кожне відгалуження відповідає певному обмеженню. Застосовується для пошуку оптимальних цілочислових рішень через розділення задачі на підзадачі з подальшим відсіканням неприйнятних рішень.

2. Метод розгалужень і оцінок (Branch and Cut): Це вдосконалений метод відгалужень і меж, який поєднує розгалуження з додаванням лінійних зрізів для покращення ефективності пошуку оптимальних рішень.

3. Динамічне програмування: Метод використовується для задач, які можна розділити на підзадачі, що вирішуються послідовно. Цей підхід особливо ефективний для задач з спеціальною структурою обмежень і функцій.

4. Евристичні та метаевристичні методи: Методи, такі як генетичні алгоритми, табуйовані пошуки та метод імітації відпалу, використовуються для знаходження наближених рішень великих і складних задач цілочислового програмування, коли точний розв'язок є важким або неможливим.

2.4. Практичне застосування

Метод Гоморі буде застосований для нашої задачі максимізації житлової площі при будівництві 100 будинків, враховуючи всі задані обмеження. На основі описаного алгоритму буде реалізовано послідовне додавання зрізів та повторне розв'язання задачі до досягнення оптимального цілочислового розв'язку.

Вибір методу Гоморі для розв'язання даної задачі обумовлений його ефективністю та можливістю знаходження точних цілочислових розв'язків, що є критично важливим для реальних будівельних проектів.

## ****Розв’язання задачі МП****

У відповідності до методу правильних відсічень спочатку відкидаємоумову цілочисельності й розв’язуємо отриману задачу лінійного програмування:

Обернемо нерівності в рівності шляхом введення вільних змінних відповідно:

Введемо в задачу дві штучні змінні :

Отримується початковий допустимий базисний розв’язок задачі, який задовольняє всім умовам-обмеженням:

Складається вихідна симплекс-таблиця:

Таблиця 1

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | C | **–** | 3000 | 2000 | 5000 | 4000 | 6000 | 0 | 0 | 0 | -M | -M |
|  | B |  |  |  |  |  |  |  |  |  |  |  |
| 0 |  | 3000 | 20 | 30 | 35 | 30 | 40 | 1 | 0 | 0 | 0 | 0 |
| 0 |  | 4500 | 40 | 20 | 60 | 35 | 25 | 0 | 1 | 0 | 0 | 0 |
| -M |  | 100 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| -M |  | 10 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | -1 | 1 | 0 |
|  |  | -110M | -M-3000 | -2M-2000 | -M-5000 | -M-4000 | -M-6000 | 0 | 0 | M | 0 | 0 |

Визначимо:

напрямний стовпець – 𝐴2,

напрямний рядок – 𝑥9,

напрямний елемент – 𝑥9 2 = 1.

З базису виводиться штучна змінна, видаляємо відповідний стовпець з таблиці.

Розрахуємо елементи наступної симплекс-таблиці:

Таблиця 2

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | C | **–** | 3000 | 2000 | 5000 | 4000 | 6000 | 0 | 0 | 0 | -M |
|  | B |  |  |  |  |  |  |  |  |  |  |
| 0 |  |  |  | 0 |  |  |  |  |  |  |  |
| 0 |  |  |  | 0 |  |  |  |  |  |  |  |
| -M |  |  |  | 0 |  |  |  |  |  |  |  |
| 2000 |  |  |  | 1 |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |

Наступна симплекс табличка:

Таблиця 3

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | C | **–** | 3000 | 2000 | 5000 | 4000 | 6000 | 0 | 0 | 0 | -M |
|  | B |  |  |  |  |  |  |  |  |  |  |
| 0 |  | 2700 | 20 | 0 | 35 | 30 | 40 | 1 | 0 | 30 | 0 |
| 0 |  | 4300 | 40 | 0 | 60 | 35 | 25 | 0 | 1 | 20 | 0 |
| -M |  | 90 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 2000 |  | 10 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | -1 | 0 |
|  |  | -90M+ 20000 | -M-3000 | 0 | -M-5000 | -M-4000 | -M-6000 | 0 | 0 | -M-2000 | 0 |

Визначимо:

напрямний стовпець – 𝐴5,

напрямний рядок – 𝑥6,

напрямний елемент – 𝑥6 5 = 40.

Розрахуємо елементи наступної симплекс-таблиці:

Таблиця 4

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | C | **–** | 3000 | 2000 | 5000 | 4000 | 6000 | 0 | 0 | 0 | -M |
|  | B |  |  |  |  |  |  |  |  |  |  |
| 6000 |  |  |  |  |  |  | 1 |  |  |  |  |
| 0 |  |  |  |  |  |  | 0 |  |  |  |  |
| -M |  |  |  |  |  |  | 0 |  |  |  |  |
| 2000 |  |  |  |  |  |  | 0 |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |

Наступна симплекс табличка:

Таблиця 5

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | C | **–** | 3000 | 2000 | 5000 | 4000 | 6000 | 0 | 0 | 0 | -M |
|  | B |  |  |  |  |  |  |  |  |  |  |
| 6000 |  |  |  | 0 |  |  | 1 |  | 0 |  | 0 |
| 0 |  |  |  | 0 |  |  | 0 |  | 1 |  | 0 |
| -M |  |  |  | 0 |  |  | 0 |  | 0 |  | 1 |
| 2000 |  | 10 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | -1 | 0 |
|  |  |  |  | 0 |  |  | 0 |  | 0 |  | 0 |

Визначимо:

напрямний стовпець – 𝐴1,

напрямний рядок – 𝑥10,

напрямний елемент – 𝑥10 1 = .

З базису виводиться штучна змінна, видаляємо відповідний стовпець з таблиці.

Розрахуємо елементи наступної симплекс-таблиці:

Таблиця 6

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | C | **–** | 3000 | 2000 | 5000 | 4000 | 6000 | 0 | 0 | 0 |
|  | B |  |  |  |  |  |  |  |  |  |
| 6000 |  |  | 0 |  |  |  |  |  |  |  |
| 0 |  |  | 0 |  |  |  |  |  |  |  |
| 3000 |  |  | 1 |  |  |  |  |  |  |  |
| 2000 |  |  | 0 |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |

Наступна симплекс табличка:

Таблиця 3

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | C | **–** | 3000 | 2000 | 5000 | 4000 | 6000 | 0 | 0 | 0 |
|  | B |  |  |  |  |  |  |  |  |  |
| 6000 |  | 45 | 0 | 0 |  |  | 1 |  | 0 |  |
| 0 |  | 1375 | 0 | 0 |  |  | 0 |  | 1 |  |
| 3000 |  | 45 | 1 | 0 |  |  | 0 |  | 0 |  |
| 2000 |  | 10 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | -1 |
|  |  | 425000 | 0 | 0 | 250 | 500 | 0 | 150 | 0 | 2500 |

Задача на максимум, індексному рядку відсутні від'ємні елементи, отже, отриманий план є оптимальним.

Отримані результати:

— цілі числа, тому отримано оптимальний цілочисельний розв’язок задачі.

**Висновки до розділу 2**

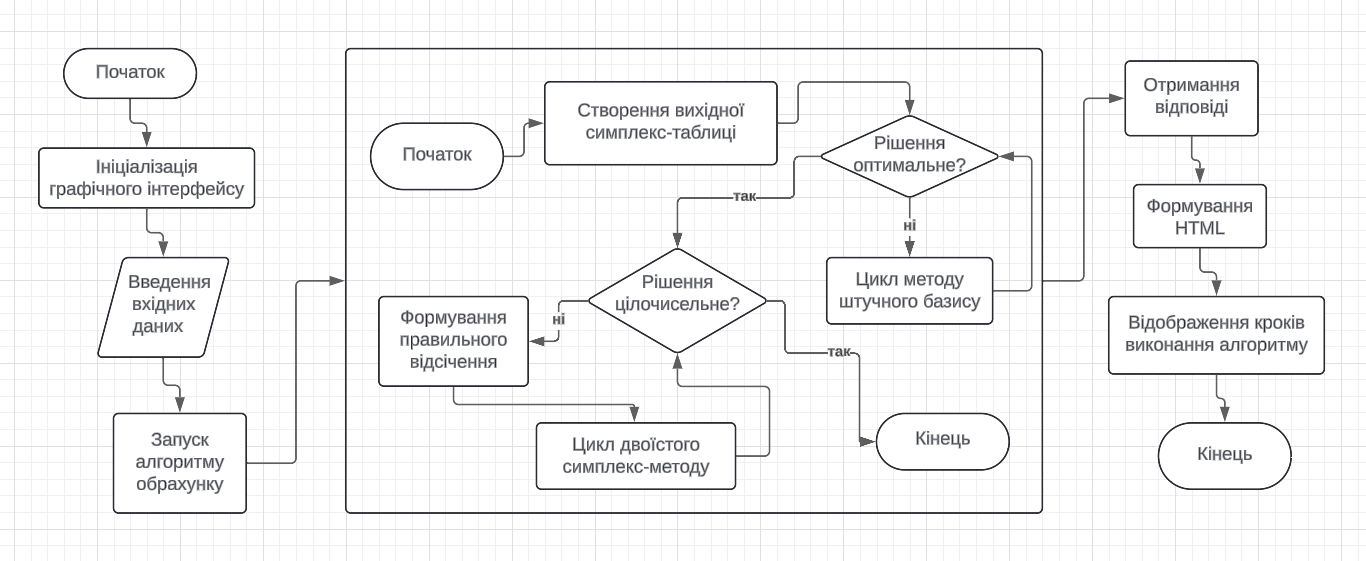
У другому розділі було обрано метод Гоморі для розв'язання задачі цілочислового програмування, сформованої в попередньому розділі. Метод Гоморі, завдяки своїй здатності ефективно обробляти задачі з цілочисловими змінними, дозволив оптимізувати план будівництва будинків, забезпечуючи максимальне використання ресурсів та дотримання всіх обмежень.

Під час розв'язання задачі було детально розглянуто основні етапи методу Гоморі: початкове розв'язання лінійної задачі, формування зрізів для дробових розв'язків та їх поступове додавання до системи обмежень. У результаті застосування цього методу вдалося отримати оптимальні цілочислові рішення, які забезпечують максимальну житлову площу при будівництві 100 будинків за 300 робочих днів, включаючи вимогу побудови не менше 10 будинків другого типу.

Таким чином, метод Гоморі виявився ефективним інструментом для вирішення задачі, що підтверджує його доцільність та надійність у практичному застосуванні для будівельних проектів з подібними характеристиками та вимогами.

## РОЗРОБКА ПРОГРАМИ ДЛЯ РОЗВ’ЯЗАННЯ ЗАДАЧІ МП

## ****Аналіз методів реалізації та опис алгоритмів****



*Рис.3.1. Планування структури додатку*

Розробка програми для розв'язання цілочислової задачі лінійного програмування включає кілька ключових етапів:

1. Введення вхідних даних: Програма повинна забезпечувати можливість введення вхідних даних, таких як кількість видів продукції, кількість видів сировини, кількість одиниць сировини, необхідної для виготовлення одиниці кожного з видів продукції, запаси сировини та прибуток підприємства від реалізації одиниці кожного з видів продукції.

2. Побудова початкової лінійної задачі: На основі введених даних програма будує початкову лінійну задачу, яка включає цільову функцію та обмеження задачі.

3. Застосування методу правильних відсічень (методу Гоморі): Реалізація алгоритму методу Гоморі для поступового додавання відсічних площин до початкової лінійної задачі. Цей алгоритм дозволяє ефективно знаходити цілочисловий розв'язок задачі.

4. Визначення оптимального розв'язку: Після застосування методу Гоморі програма визначає оптимальний цілочисловий розв'язок задачі на основі отриманих даних.

5. Виведення результатів: Останній етап роботи програми полягає у виведенні отриманих результатів, які включають значення цільової функції та оптимальні значення змінних.

У процесі реалізації програми необхідно також враховувати можливість обробки помилок, оптимізацію алгоритмів для підвищення швидкості роботи програми, а також розробку користувацького інтерфейсу, який забезпечить зручний та інтуїтивно зрозумілий доступ користувача до функціоналу програми..

## ****Вибір й обґрунтування інструментальних засобів для реалізації програмного додатку****

Для реалізації програмного додатку з розв'язання цілочислової задачі лінійного програмування було обрано JavaScript (JS). Цей вибір обґрунтований наступними причинами:

3.1. Універсальність та популярність JavaScript

JavaScript є одним з найбільш популярних мов програмування у світі, що забезпечує високу підтримку спільноти та наявність великої кількості бібліотек і фреймворків. JS використовується для розробки як фронтенд, так і бекенд компонентів веб-додатків, що дозволяє створювати інтегровані рішення з єдиною мовою програмування.

3.2. Використання сучасних фреймворків і бібліотек

JS має великий набір фреймворків і бібліотек, які значно спрощують розробку складних додатків. Для реалізації нашого програмного додатку можуть бути використані наступні інструменти:

- React або Vue.js: Для створення інтуїтивно зрозумілого користувацького інтерфейсу. Ці фреймворки забезпечують високу продуктивність та зручність розробки компонентів.

- Node.js: Для реалізації бекенд частини додатку. Node.js дозволяє використовувати JavaScript на сервері, що забезпечує швидкість та ефективність обробки запитів.

- Express.js: Легкий і гнучкий фреймворк для Node.js, який спрощує створення веб-серверів та API.

3.3. Підтримка асинхронного програмування

JavaScript має вбудовану підтримку асинхронного програмування, що є важливим для створення ефективних і продуктивних додатків. Асинхронні функції та обіцянки (promises) дозволяють обробляти великі обсяги даних та виконувати численні запити без блокування головного потоку виконання.

3.4. Взаємодія з користувачем у реальному часі

JS дозволяє створювати динамічні веб-додатки, які можуть взаємодіяти з користувачами в режимі реального часу. Це важливо для нашого додатку, оскільки користувачі повинні мати можливість вводити дані, отримувати результати та взаємодіяти з інтерфейсом швидко і без затримок.

3.5. Багатоплатформеність

Програмний додаток, написаний на JavaScript, може бути легко адаптований для роботи на різних платформах, включаючи веб, мобільні додатки та десктопні додатки. Це забезпечує широку доступність та зручність використання додатку для різних категорій користувачів.

3.6. Безпека та надійність

JS-фреймворки та бібліотеки активно розвиваються та підтримуються, що забезпечує високу безпеку і надійність додатків. Використання сучасних стандартів кодування та інструментів для тестування допомагає знижувати ризики та забезпечувати стабільну роботу програмного забезпечення.

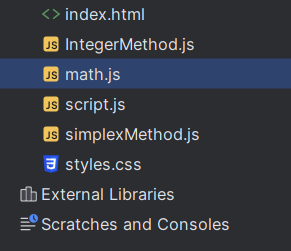
3.7. Модульність та повторне використання коду

JavaScript підтримує модульний підхід до розробки, що дозволяє створювати окремі компоненти, які можуть бути повторно використані у різних частинах додатку. Це сприяє зниженню часу розробки та підвищенню якості коду.

Вибір JavaScript як інструментального засобу для реалізації програмного додатку з розв'язання цілочислової задачі лінійного програмування є оптимальним рішенням завдяки його універсальності, широкій підтримці спільноти, можливості використання сучасних фреймворків і бібліотек, підтримці асинхронного програмування, взаємодії з користувачем у реальному часі, багатоплатформеності, безпеці та надійності, а також модульному підходу до розробки. Це дозволяє створити ефективний, продуктивний та зручний для користувачів програмний додаток.

## ****Розробка програмного додатку****

Було створено таку структуру проекту:



*Рис.3.2. Структура програми*

Для початку було створено клас Frac для реалізації можливостей дробового числа. Клас дозволяє зберігати дробове число (чисельник та знаменник), реалізує необхідні математичні операції додавання, віднімання, множення, ділення. При створенні екзепляру класу дріб спрощується. Реалізується переведення екземпляру в HTML.

class Frac {  
 constructor(a, b) {  
 this.a = a;  
 this.b = b;  
  
 if (a < 0 && b < 0) {  
 this.a = *Math*.abs(a);  
 this.b = *Math*.abs(b);  
 }  
 else if (b < 0) {  
 this.a = -a;  
 this.b = *Math*.abs(b);  
 }  
  
 let t = Frac.*#gcd*(this.a, this.b);  
 this.a /= t;  
 this.b /= t;  
 }  
  
 add(other) {  
 other = Frac.*convertOther*(other);  
 return new Frac(this.a \* other.b + other.a \* this.b, this.b \* other.b);  
 }  
  
 sub(other) {  
 other = Frac.*convertOther*(other);  
 return new Frac(this.a \* other.b - other.a \* this.b, this.b \* other.b);  
 }  
  
 mul(other) {  
 other = Frac.*convertOther*(other);  
 return new Frac(this.a \* other.a, this.b \* other.b);  
 }  
  
 div(other) {  
 other = Frac.*convertOther*(other);  
 return this.mul(new Frac(other.b, other.a));  
 }  
  
 abs() {  
 return new Frac(*Math*.abs(this.a), this.b);  
 }  
  
 valueOf() {  
 return this.a / this.b;  
 }  
 toString() {  
 if (this.b === 1)  
 return `${this.a}`;  
 return `<div class="fraction">${this.a} <div class="line"></div> ${this.b}</div>`;  
 }  
  
 static *convertOther*(other) {  
 if (!(other instanceof Frac))  
 return new Frac(other, 1);  
 else  
 return other;  
 }  
  
 static *#gcd*(a,b) {  
 a = *Math*.abs(a);  
 b = *Math*.abs(b);  
 if (b > a) {let temp = a; a = b; b = temp;}  
 while (true) {  
 if (b === 0) return a;  
 a %= b;  
 if (a === 0) return b;  
 b %= a;  
 }  
 }  
}

Наступним кроком була реалізація класу великого числа GreatNum. Клас дозволяє зберігати M числа. Реалізує додавання, віднімання, множення ділення екземплярів класу, як з іншими екземплярами класу так і з простими числами (цілими чи дробовими).

class GreatNumber {  
 constructor(m, a) {  
 if (!(m instanceof Frac))  
 m = new Frac(m, 1);  
 this.m = m;  
  
 if (!(a instanceof Frac))  
 a = new Frac(a, 1);  
 this.a = a;  
 }  
  
 add(b) {  
 if (b instanceof GreatNumber)  
 return new GreatNumber(this.m.add(b.m), this.a.add(b.a));  
 else  
 return new GreatNumber(this.m, this.a.add(b));  
 }  
  
 sub(b) {  
 if (b instanceof GreatNumber)  
 return new GreatNumber(this.m.sub(b.m), this.a.sub(b.a));  
 else  
 return new GreatNumber(this.m, this.a.sub(b));  
 }  
  
 mul(b) {  
 if (b instanceof GreatNumber)  
 throw new Error("Спроба множення двух об'єктів классу BigNum");  
 else  
 return new GreatNumber(this.m.mul(b), this.a.mul(b));  
 }  
  
 div(b) {  
 if (b instanceof GreatNumber)  
 throw new Error("Спроба ділення двух об'єктів классу BigNum");  
 else  
 return new GreatNumber(this.m.div(b), this.a.div(b));  
 }  
  
 abs() {  
 return new GreatNumber(this.m.abs(), this.a.abs());  
 }  
  
 toString() {  
 let res = "";  
  
 if (this.m == 1)  
 res = `M`;  
 else if (this.m == -1)  
 res = `-M`;  
 else  
 res = `${this.m.toString()}M`;  
  
 if (this.m == 0)  
 return this.a.toString();  
  
 if (this.a < 0)  
 res += ` - ${this.a.abs().toString()}`;  
 else if (this.a != 0)  
 res += ` + ${this.a.toString()}`;  
  
 return res;  
 }  
  
 valueOf() {  
 return this.m \* 9999999999999999999 + this.a;  
 }  
}

Далі було створено клас для реалізації методу штучного базису. Клас реалізує всі можливості симплекс таблиці: зберігає необхідні значення, має функції обрахування індексного рядку, обрахування напрямного елементу, обрахування наступної симплекс таблиці, видалення штучної змінної.

class SimplexArr {  
 \_n;  
 \_m;  
 \_fl = false;  
 constructor(arr, x\_n) {  
 this.arr = arr;  
 this.x\_n = x\_n;  
 this.pivotElement = null;  
  
  
 this.\_n = this.arr.length - 2;  
 this.\_m = this.arr[0].length - 2;  
 }  
  
 calcDelta() {  
 for (let j = 2; j < this.\_m + 2; j++) {  
 if (j !== 2)  
 this.arr[this.\_n + 1][j] = this.arr[0][j].mul(-1);  
 else  
 this.arr[this.\_n + 1][j] = new GreatNumber(0, 0);  
  
 for (let i = 2; i < this.\_n + 1; i++)  
 this.arr[this.\_n + 1][j] = this.arr[this.\_n + 1][j].add(this.arr[i][0].mul( this.arr[i][j]));  
 }  
 }  
  
 calcPivotElement() {  
  
 let min\_j, min\_delta, min\_i = null;  
  
 min\_delta = *Math*.min(...this.arr[this.\_n + 1].slice(3));  
 if (min\_delta >= 0)  
 return;  
  
 for (min\_j = 2; min\_j < this.\_m + 2; min\_j++) {  
 if (+this.arr[this.\_n + 1][min\_j] === min\_delta) {  
  
 for (let i = 2; i < this.\_n + 1; i++)  
 if (this.arr[i][min\_j] > 0) {  
 if (min\_i === null)  
 min\_i = i;  
 else  
 if (this.arr[i][2].div(this.arr[i][min\_j]) < this.arr[min\_i][2].div(this.arr[min\_i][min\_j]))  
 min\_i = i;  
  
 }  
  
 }  
  
 if (min\_i !== null) {  
 break;  
 }  
 if (min\_j === (this.\_m + 1)) {  
 this.\_fl = true;  
 return;  
 }  
  
 }  
  
 this.pivotElement = {  
 i: min\_i,  
 j: min\_j  
 };  
 }  
  
 checkIsNextPossible() {  
 return this.pivotElement !== null;  
 }  
  
 checkResult() {  
 if (this.pivotElement !== null)  
 return "Задача має продовження";  
 if (this.\_fl)  
 return "Всі Δ ≥ 0 і в напрямному стовпці всі значення ≤ 0, тому задача рішення не має";  
 for (let i = 2; i < this.\_n + 1; i++)  
 if (this.arr[i][2] < 0)  
 return "Всі Δ ≥ 0 і в стовпці вільних коефіцієнтів є відємне значення, тому задача рішення не має";  
 else if (this.arr[i][0].m != 0)  
 return "Всі Δ ≥ 0 і в базисі все ще є штучна змінна, тому задача рішення не має"  
 return true;  
 }  
  
 next() {  
 if (!this.checkIsNextPossible())  
 return null;  
 return new SimplexArr(this.\_calcNextArr(), this.x\_n);  
 }  
  
 \_calcNextArr() {  
  
  
 let new\_arr = AdditionalFunctions.*copy\_arr*(this.arr);  
  
 new\_arr[this.pivotElement.i][0] = this.arr[0][this.pivotElement.j];  
 new\_arr[this.pivotElement.i][1] = this.arr[1][this.pivotElement.j];  
  
 for (let i = 2; i < this.\_n + 1; i++) {  
 for (let j = 2; j < this.\_m + 2; j++)  
 if (i === this.pivotElement.i)  
 new\_arr[i][j] = this.arr[this.pivotElement.i][j].div(this.arr[this.pivotElement.i][this.pivotElement.j]);  
 else  
 new\_arr[i][j] = this.arr[i][j].sub(  
 (this.arr[this.pivotElement.i][j].mul(this.arr[i][this.pivotElement.j]).div(  
 this.arr[this.pivotElement.i][this.pivotElement.j])));  
 new\_arr[i][this.pivotElement.j] = new Frac(0, 1);  
 }  
 new\_arr[this.pivotElement.i][this.pivotElement.j] = new Frac(1, 1);  
  
 if (this.arr[this.pivotElement.i][0].m != 0) {  
 //console.log(this.arr[this.pivotElement.i][0].m)  
 for (let j = 3; j < this.\_m + 2; j++) {  
 if (this.arr[this.pivotElement.i][1] === this.arr[1][j]) {  
 new\_arr = AdditionalFunctions.*delete\_row*(new\_arr, j);  
 break;  
 }  
 }  
 }  
  
 return new\_arr;  
 }  
}

Після було створено клас симплекс таблиці двоїстого методу. Клас наслідує всі можливості класу звичайної симплекс таблиці і перевизначає визначення напрямного елементу.

class DualSimplexArr extends SimplexArr{  
 constructor(...arr) {  
 super(...arr);  
 this.theta\_arr = null;  
 }  
 calcPivotElement() {  
 let min\_j = null, min\_i\_val = this.arr[2][2], min\_i = 2;  
  
 for (let i = 2; i < this.\_n + 1; i++)  
 if (min\_i\_val > this.arr[i][2])  
 min\_i\_val = this.arr[i][2];  
  
 if (min\_i\_val >= 0)  
 return;  
  
 for (min\_i = 2; min\_i < this.\_n + 1; min\_i++) {  
 if (this.arr[min\_i][2] === min\_i\_val) {  
 this.q = {  
 arr\_str: [],  
 arr: [],  
 element: null  
 }  
 this.theta\_arr = ["", "θ", "-"];  
  
 for (let j = 3; j < this.\_m + 2; j++)  
 if (this.arr[min\_i][j] < 0) {  
 if (min\_j === null)  
 min\_j = j;  
 else  
 if (this.arr[this.\_n + 1][j].div( this.arr[min\_i][j]).abs() < this.arr[this.\_n + 1][min\_j].div(this.arr[min\_i][min\_j]).abs())  
 min\_j = j;  
 this.theta\_arr.push(this.arr[this.\_n + 1][j].div(this.arr[min\_i][j]).abs());  
 }  
 else  
 this.theta\_arr.push("-");  
 }  
  
 if (min\_j !== null) {  
 break;  
 }  
  
 if (min\_i === (this.\_n)) {  
 this.\_fl = true;  
 return;  
 }  
  
 }  
  
 this.pivotElement = {  
 i: min\_i,  
 j: min\_j  
 };  
 }  
  
 next() {  
 if (!this.checkIsNextPossible())  
 return null;  
 return new DualSimplexArr(this.\_calcNextArr().slice(0, this.\_n + 2), this.x\_n);  
 }  
}

Далі було створено клас цілочисельної задачі. Клас приймає на вхід симплекс табличку (як двоїстого симплекс методу так і методу штучного базису) і має можливості перевірки цілочисельності рішення та формування правильного відсічення.

class IntegerMethod {  
 constructor(table) {  
 this.table = table;  
 this.x\_cut = null;  
 }  
  
 calculateCutOff() {  
 let n = this.table.arr.length - 2, m = this.table.arr[0].length - 2;  
 let new\_x = `x${this.table.arr[0].length - 2}`;  
  
  
 let new\_row = [new GreatNumber(0, 0), new\_x];  
 for (let j = 2; j < m + 2; j++) {  
 new\_row.push(new Frac(0, 1).sub(this.table.arr[this.x\_cut][j].sub(*Math*.floor(this.table.arr[this.x\_cut][j]))));  
 }  
  
 new\_row.push(new Frac(1, 1));  
  
 let new\_col = [new GreatNumber(0, 0), new\_x, ...new *Array*(m).fill(new Frac(0, 1))];  
  
 let new\_arr = AdditionalFunctions.*copy\_arr*(this.table.arr).slice(0, n + 1);  
 for (let i = 0; i < n + 1; i++)  
 new\_arr[i].push(new\_col[i]);  
  
 new\_arr.push(new\_row);  
 new\_arr.push(["", "Δ", ...new *Array*(m + 1).fill(0)]);  
  
 //console.log(AdditionalFunctions.copy\_arr(new\_arr), this.table.x\_n);  
  
 return {  
 simplexTable: new DualSimplexArr(new\_arr, this.table.x\_n),  
 newRestrictionHTML: this.restrictionHTML(n, m, new\_x)  
 };  
 }  
  
 restrictionHTML(n, m, new\_x) {  
 let newRestrictionHTML = "<div>Розв'язок не є цілочисельним, додаємо правильне відсічення:</div>";  
 let newRestriction = [];  
  
 for (let j = 2; j < m + 2; j++) {  
 let x = this.table.arr[this.x\_cut][j], round\_x = *Math*.floor(this.table.arr[this.x\_cut][j]);  
 newRestriction.push(new Frac(0, 1).sub(x.sub(round\_x)));  
 }  
  
 newRestriction.push(new Frac(1, 1));  
 let coef = newRestriction.shift();  
  
  
  
 let vars = this.table.arr[1].slice(3);  
 vars.push(new\_x);  
 vars = IntegerMethod.*#varsToHTML*(vars);  
  
 newRestrictionHTML += `<div class=\"equations\">${IntegerMethod.*genEq*(vars, newRestriction, coef, "=")}</div>`;  
 return newRestrictionHTML;  
 }  
  
 checkIsResult() {  
  
 for (let j = 3; j < this.table.x\_n + 3; j++) {  
 //console.log(this.table.arr[1][j]);  
 for (let i = 2; i < this.table.arr.length - 1; i++) {  
 //console.log(` ${this.table.arr[i][1]}`);  
 if (this.table.arr[1][j] === this.table.arr[i][1]) {  
 let t = this.table.arr[i][2];  
 if (t.b !== 1) {  
 this.x\_cut = i;  
 return false;  
 }  
  
 }  
 }  
  
 }  
  
 return true;  
 }  
  
 static *#varsToHTML*(vars) {  
 let res = [];  
 for (let i = 0; i < vars.length; i++)  
 res.push(`x<sub>${vars[i].substr(1)}</sub>`);  
 return res;  
 }  
  
 static *genEq*(vars, coefs, freeCoef, sign) {  
 let eq = IntegerMethod.*#coefsToString*(coefs, vars);  
 if (!eq)  
 eq = "0";  
 eq += ` ${sign} ${freeCoef}`;  
 return `<div class='equation'>${eq}</div>`;  
 }  
  
 static *#coefsToString*(coefs, vars) {  
 let res = "";  
 for (let i = 0; i < coefs.length; i++) {  
 if (coefs[i] == 1)  
 res += ` + ${vars[i]}`;  
 else if (coefs[i] == -1)  
 res += ` - ${vars[i]}`;  
 else if (coefs[i] < 0)  
 if (coefs[i] instanceof Frac)  
 if (res === "")  
 res += ` + ${coefs[i]}${vars[i]}`;  
 else  
 res += ` - ${coefs[i].abs()}${vars[i]}`;  
 else  
 res += ` - ${*Math*.abs(coefs[i])}${vars[i]}`;  
 else if (coefs[i] > 0)  
 res += ` + ${coefs[i]}${vars[i]}`;  
 }  
  
 if (res[1] === "-") {  
 res = res.substr(3);  
 res = "-" + res;  
 return res;  
 }  
 return res.substr(3);  
 }  
}

Далі було створено функцію яка реалізує цикл рішення симплекс таблиці (як двоїстого симплекс методу так і методу штучного базису). Клас приймає на вхід симплекс таблицю і циклічно вирішує симплекс таблицю поки не отримається оптимальне рішення (не обов’язково цілочисельно).

function simplexLoop(simplexTable, solve) {  
 while (true) {  
 simplexTable.calcDelta();  
 simplexTable.calcPivotElement();  
  
 let simplexVisual = new SimplexVisual(simplexTable);  
 solve.innerHTML += simplexVisual.tableToHTML();  
  
 if (!simplexTable.checkIsNextPossible())  
 break;  
 else  
 simplexTable = simplexTable.next();  
 }  
 return simplexTable;  
}

На наступному етапі було створено функцію яка реалізує метод Гоморі. На вхід вона приймає початкову симплекс таблицю. Спочатку запускається звичайний симплекс метод і після отримання оптимального рішення запускається цикл створення правильних відсічень доти поки не отримається оптимальне цілочисельне рішення (якщо кількість правильних відсіченнь перевищує 20 програма видає помилку).

function simplexSolve(simplexTable) {  
 SimplexVisual.*table\_i* = 1;  
 let solve = *document*.querySelector("#solve"), i = 1;  
 solve.innerHTML = "";  
 simplexTable.arr = refactorArr(simplexTable.arr);  
  
 simplexTable = simplexLoop(simplexTable, solve);  
 solve.innerHTML += new SimplexVisual(simplexTable).resultsToHTML();  
  
 if (simplexTable.checkResult() !== true)  
 return;  
  
 let integerTask = new IntegerMethod(simplexTable);  
 //console.log(integerTask, integerTask.checkIsResult());  
  
 let int\_i = 0;  
 while (!integerTask.checkIsResult()) {  
 let newRestrictionHTML = "";  
 ({simplexTable, newRestrictionHTML} = integerTask.calculateCutOff());  
 solve.innerHTML += newRestrictionHTML;  
  
 simplexTable = simplexLoop(simplexTable, solve);  
  
 //console.log(simplexTable);  
  
 solve.innerHTML += new SimplexVisual(simplexTable).resultsToHTML();  
  
 integerTask = new IntegerMethod(simplexTable);  
  
 if (i++ > 20) {  
 alert("Правильних відсічень більше 20, неможливо знайти оптимальний цілочисельний розв'язок!!!");  
 return;  
 }  
  
 }  
  
 if (simplexTable.checkResult() === true)  
 solve.innerHTML += "<div style='font-size: 25px; font-weight: bold;'>Отримано оптимальний цілочисельний розв'язок задачі!</div>";  
}

Функція яка первіряє на можливі помилкові значення полей.

function check\_input(inp) {  
 let t = parseFloat(inp.value);  
  
 if (isNaN(t) || t < 0 || (t - *Math*.floor(t) !== 0)) {  
 if (isNaN(t))  
 show\_error("Всі поля повинні бути заповнені числами!");  
 else if (t < 0)  
 show\_error("Всі числа повинні бути додатними!");  
 else if (t - *Math*.floor(t) !== 0)  
 show\_error("Всі значення повинні бути цілими числами!");  
  
 return false;  
 }  
 return true;  
}

При зміненні кількості проектів запускається функція для формування таблиці в формі.

let *project\_n* = 5;  
let *resultElement* = *document*.getElementById("result");  
  
*document*.querySelector("#project\_n").oninput = function () {  
 let proj\_inp = *document*.querySelector("#project\_n");  
  
 if (!check\_input(proj\_inp)) {  
 proj\_inp.value = *project\_n*;  
 return;  
 }  
  
 let t = parseInt(proj\_inp.value);  
  
 if (t < 1) {  
 proj\_inp.value = *project\_n*;  
 show\_error("Кількість проектів не може бути меньше одного!");  
 return;  
 }  
  
 let dif = t - *project\_n*;  
  
 if (dif > 0) {  
 for (let i = 0; i < dif; i++)  
 *document*.querySelector("#project\_row").innerHTML += `<td>${*project\_n* + i + 1} проект</td>`;  
 for (let i = 0; i < 3; i++) {  
 let row = *document*.querySelector(`#task\_table tr:nth-child(${i + 3})`);  
 for (let j = 0; j < dif; j++) {  
 let td = *document*.createElement("td");  
 td.innerHTML = "<input min=\"0\" type=\"number\" value=\"0\">";  
 row.appendChild(td);  
 }  
  
 }  
 } else {  
 for (let i = 0; i < 4; i++) {  
 let row = *document*.querySelector(`#task\_table tr:nth-child(${i + 2})`);  
 for (let j = 0; j < *Math*.abs(dif); j++) {  
 row.children[row.children.length - 1].outerHTML = "";  
 }  
  
 }  
 }  
  
 *project\_n* = t;  
 *document*.querySelector("#long-cell").colSpan = *project\_n*;  
 *document*.querySelector("#project\_restrict").max = *project\_n*;  
}

Функція яка перевіряє кожне поле для введення даних на сторінці.

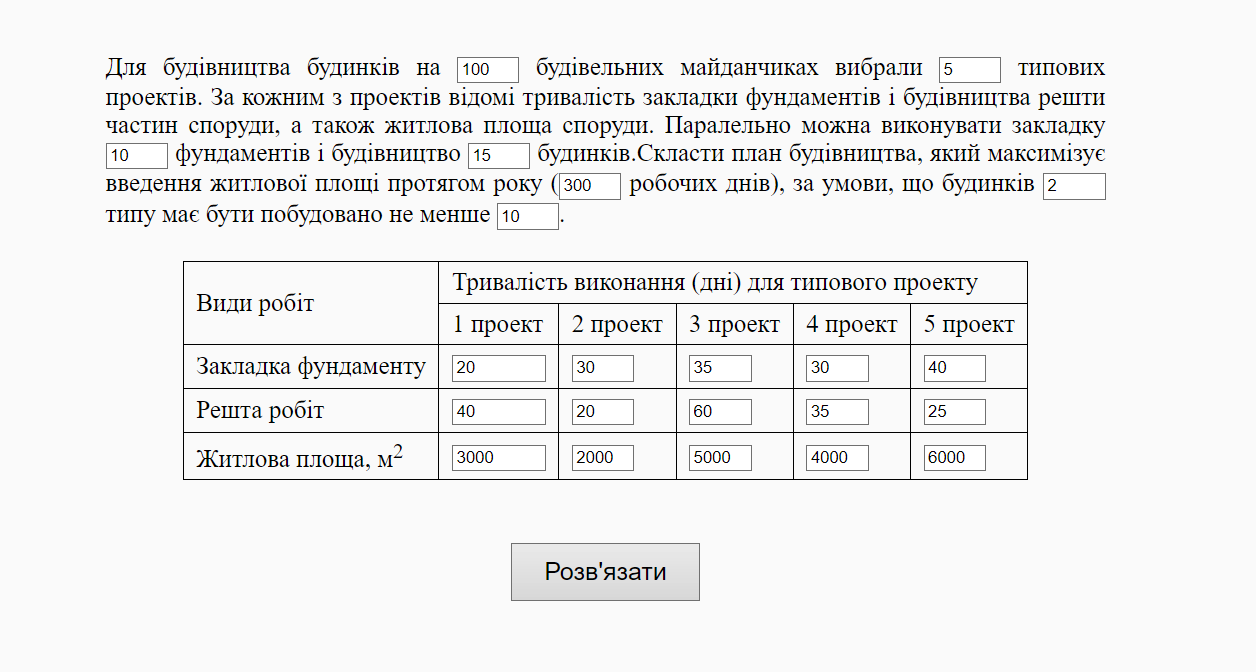
function check\_inputs() {  
 let fl = true;  
 *document*.querySelectorAll("input").forEach((inp) => {  
 if (!check\_input(inp))  
 fl = false;  
 else if (inp.id === "project\_n") {  
 if (!fl)  
 inp.value = *project\_n*;  
 if (parseFloat(inp.value) < 1) {  
 show\_error("Кількість проектів не може бути меньше одного!");  
 fl = false;  
 }  
 }  
 })  
 return fl;  
}

При натисненні на кнопку «Розв’язати» запускається функція яка перевіряє всі поля та формує початкову симплекс таблицю. Після запускається цикл методу Гоморі і виведення результатів.

*document*.getElementById("start").onclick = () => {  
 if (!check\_inputs())  
 return;  
  
 let inp = *document*.querySelector("#project\_restrict");  
 if (inp.id === "project\_restrict") {  
 if (parseFloat(inp.value) > *project\_n*) {  
 show\_error("Неможливий номер будинку!");  
 return;  
 }  
 }  
  
 let arr = [];  
 for (let i = 0; i < 3; i++) {  
 let tr = *document*.querySelector("#task\_table tbody").children[2 + i];  
 arr.push([]);  
  
 for (let j = 0; j < *project\_n*; j++) {  
 arr[i].push(tr.children[j + 1].children[0].value);  
 }  
 }  
  
 let const\_arr = [];  
 *document*.querySelectorAll("#task\_text input").forEach((inp) => {  
 const\_arr.push(parseInt(inp.value));  
 });  
  
 arr.forEach((val, i) => {  
 arr[i] = parseArr(val);  
 })  
  
 hide\_error();  
 *resultElement*.classList.add("visible");  
  
 setTimeout(() => {  
 *resultElement*.scrollIntoView({behavior: "smooth"});  
 }, 1);  
  
 let simplex\_arr = [  
 ["", "C", 0, ...arr[2], 0, 0, 0, new GreatNumber(-1, 0), new GreatNumber(-1, 0)],  
 ["", "B"],  
 [0, `x${*project\_n* + 1}`, const\_arr[4] \* const\_arr[2], ...arr[0], 1, 0, 0, 0, 0],  
 [0, `x${*project\_n* + 2}`, const\_arr[4] \* const\_arr[3], ...arr[1], 0, 1, 0, 0, 0],  
 [new GreatNumber(-1, 0), `x${*project\_n* + 5}`, const\_arr[0], ...new *Array*(*project\_n*).fill(1), 0, 0, 0, 0, 1],  
 [new GreatNumber(-1, 0), `x${*project\_n* + 4}`, const\_arr[6]],  
 ["", "Δ", ...new *Array*(*project\_n* + 6).fill(0)]  
 ];  
  
 for (let i = 0; i < *project\_n* + 6; i++)  
 simplex\_arr[1].push(`x${i}`);  
 for (let i = 0; i < *project\_n*; i++)  
 if ((i + 1) === const\_arr[5])  
 simplex\_arr[5].push(1);  
 else  
 simplex\_arr[5].push(0);  
  
 //console.log(simplex\_arr)  
 simplex\_arr[5].push(0, 0, -1, 1, 0);  
  
 let simplexTable = new SimplexArr(simplex\_arr, *project\_n*);  
  
 simplexSolve(simplexTable);  
  
}

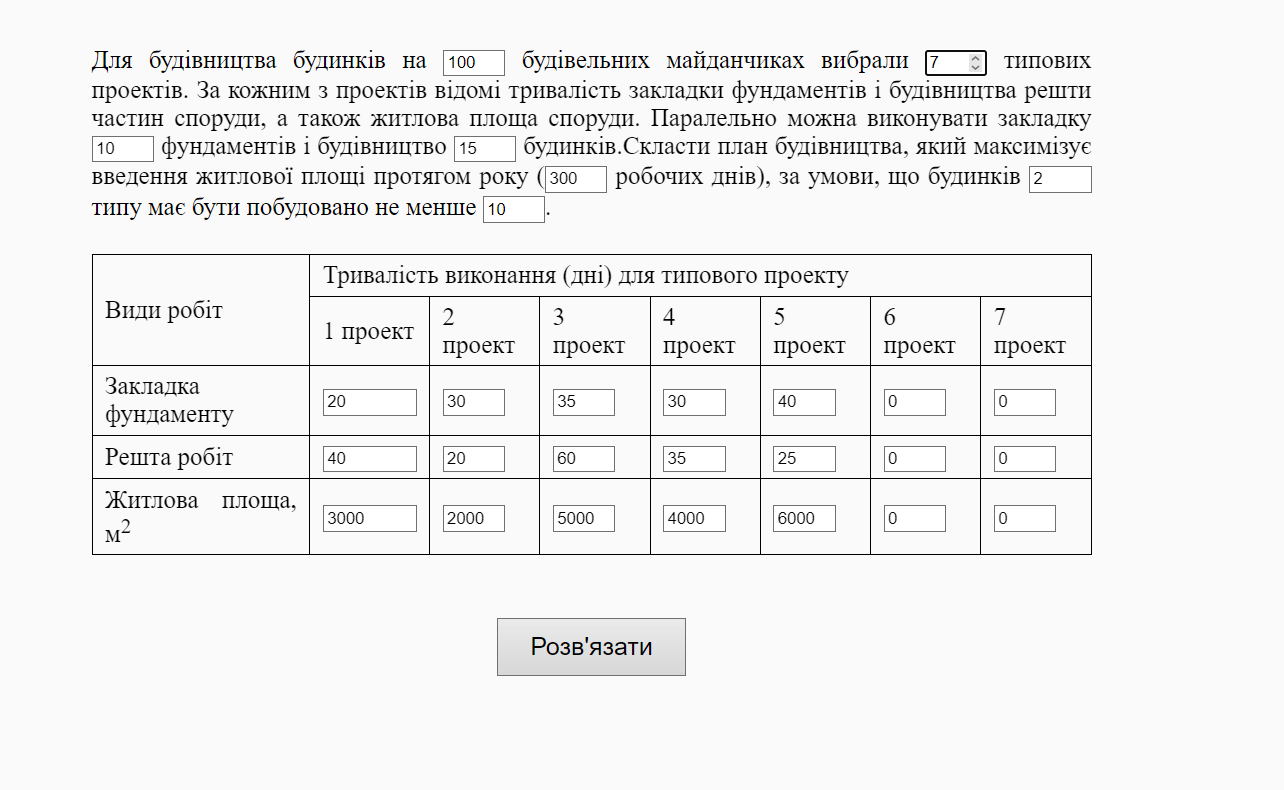
## ****Тестування програмного додатку****

Користувач при запуску програми бачить наступний вид інтерфейсу (рис. 3.3):

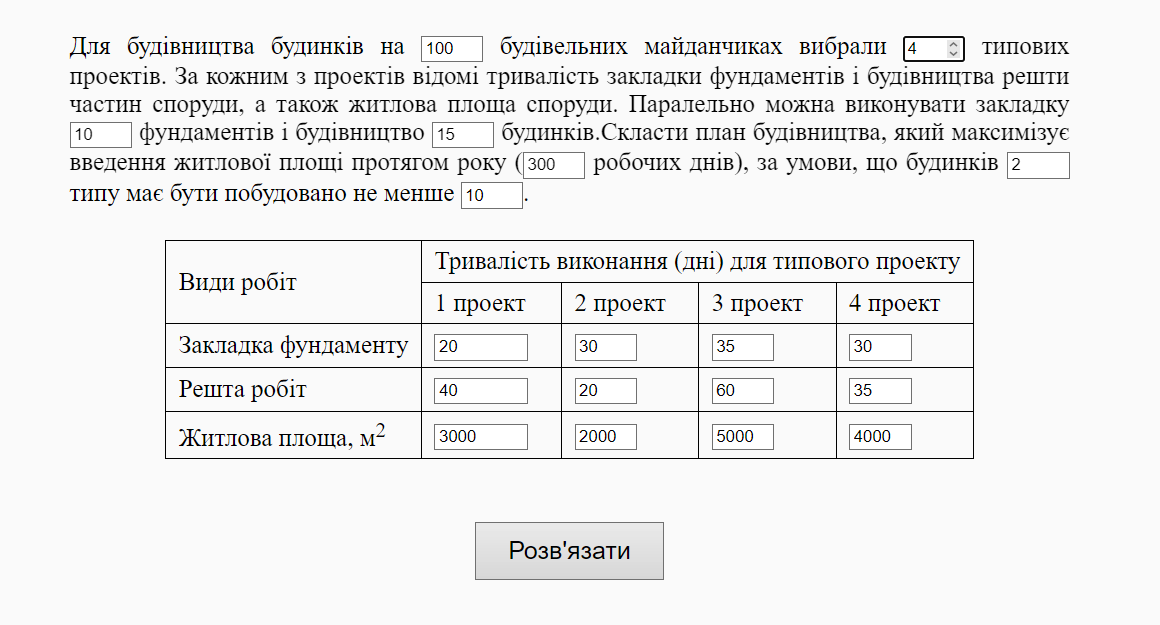


*Рис.3.3. Меню програми*

На кожне з значення з умови передбачено поле значення якого можливо змінити. При зміненні кількості проектів таблиця змінюється для відображення потрібної кількості полів (рис. 3.4 – 3.5):

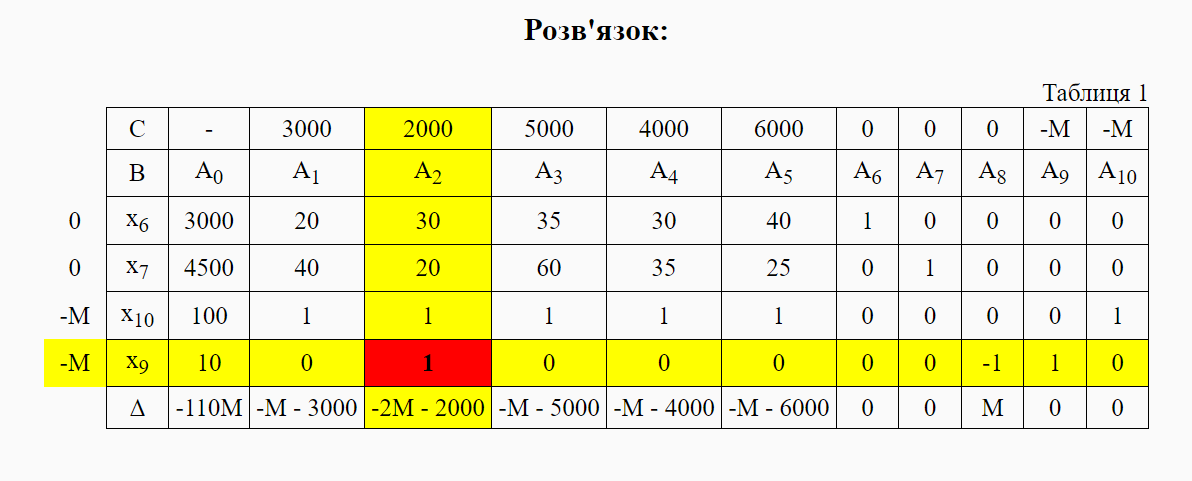


*Рис.3.4. 7 проектів*

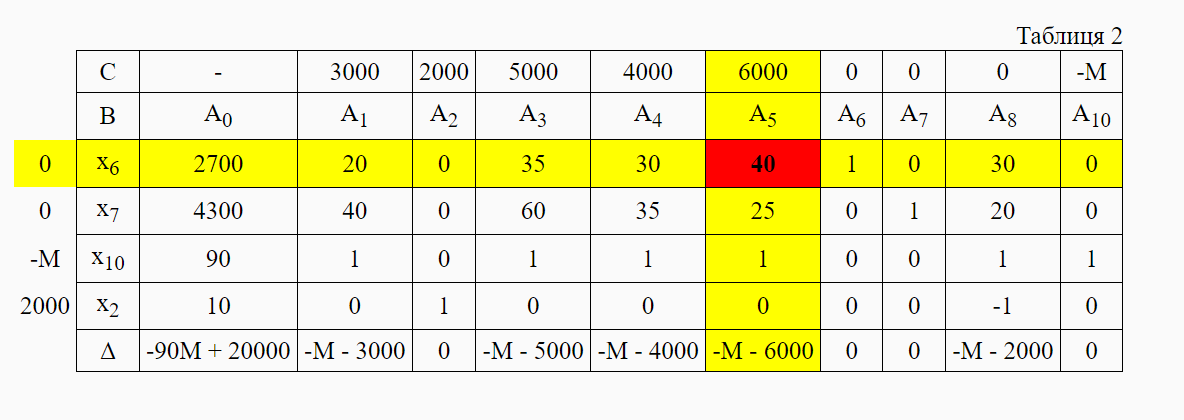
**

*Рис.3.4. 4 проекти*

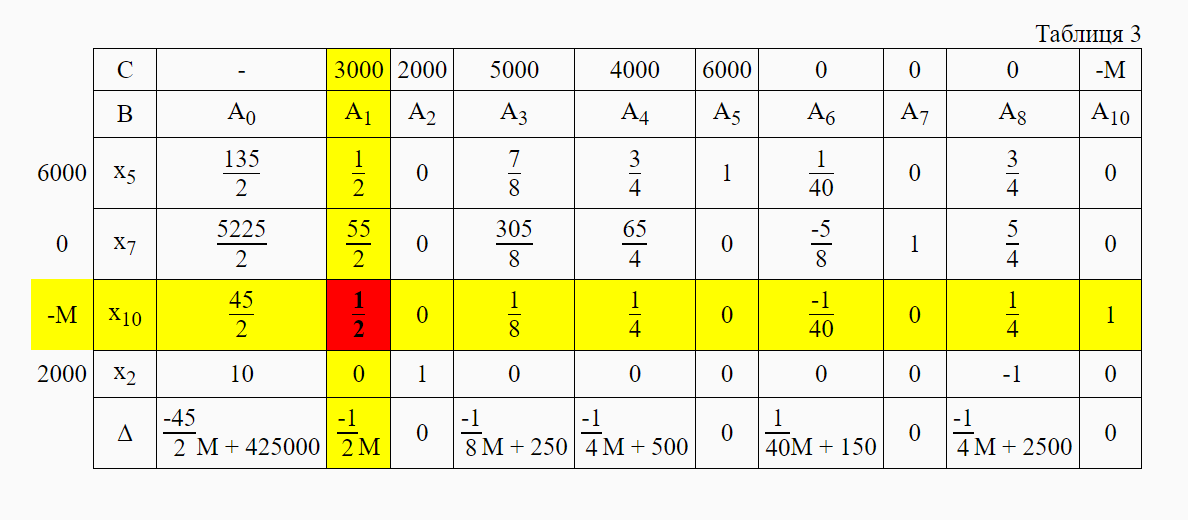
Для зручності проведення тестуваня в полях уже введено значення з задачі з умови. Для отримання розв’зку потрібно натиснути на кнопку «Розв’язати» (рис. 3.5 – 3.9):



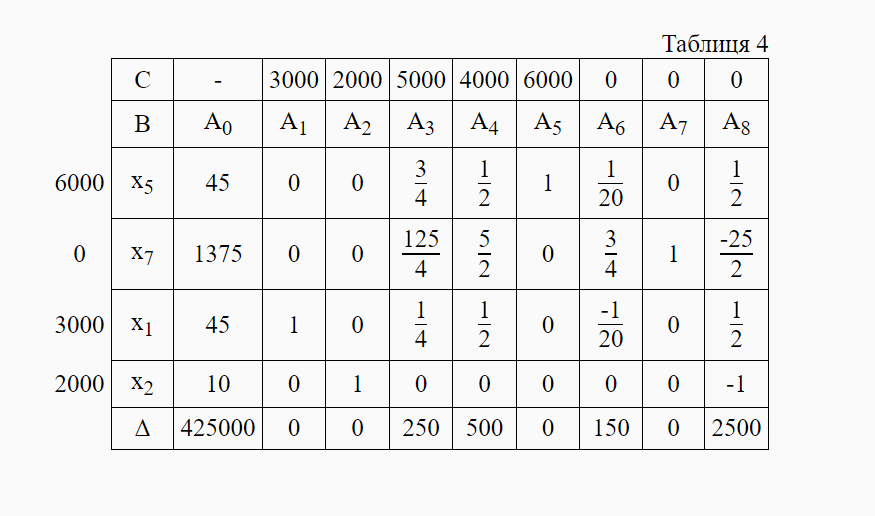
*Рис.3.5. Вихідна симплекс таблиця*

**

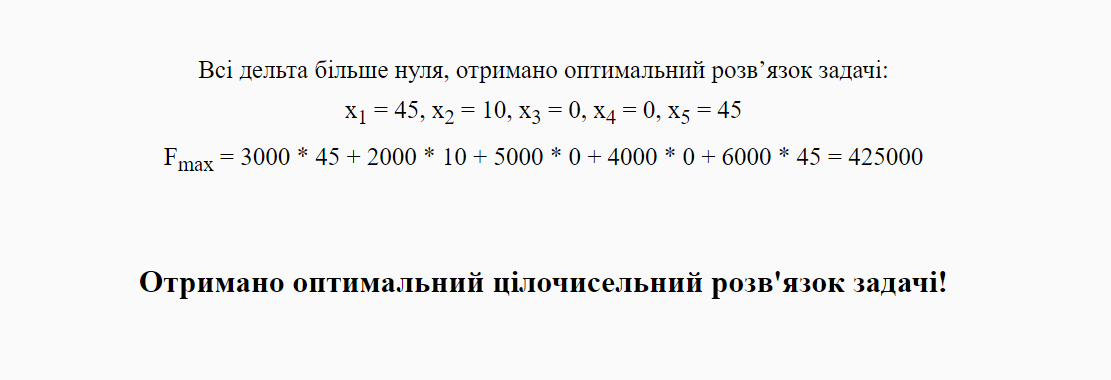
*Рис.3.6. Друга симплекс таблиця*

**

*Рис.3.7. Третя симплекс таблиця*

**

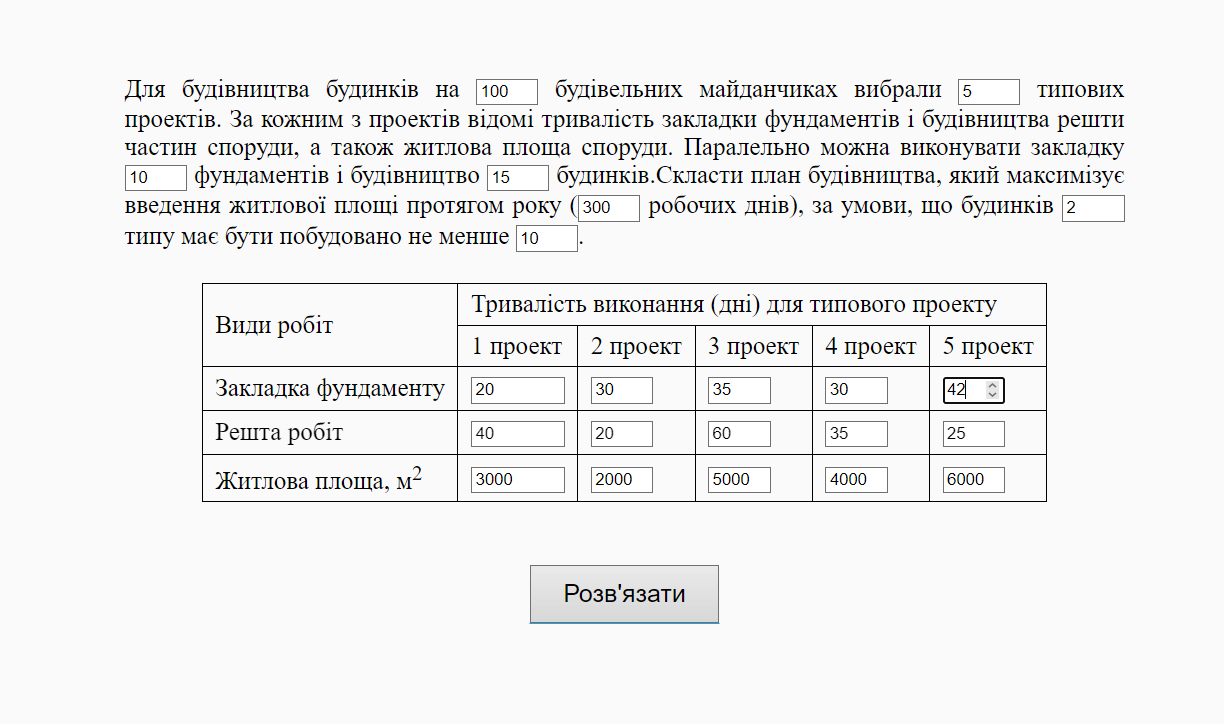
*Рис.3.8. Четверта симплекс таблиця*

**

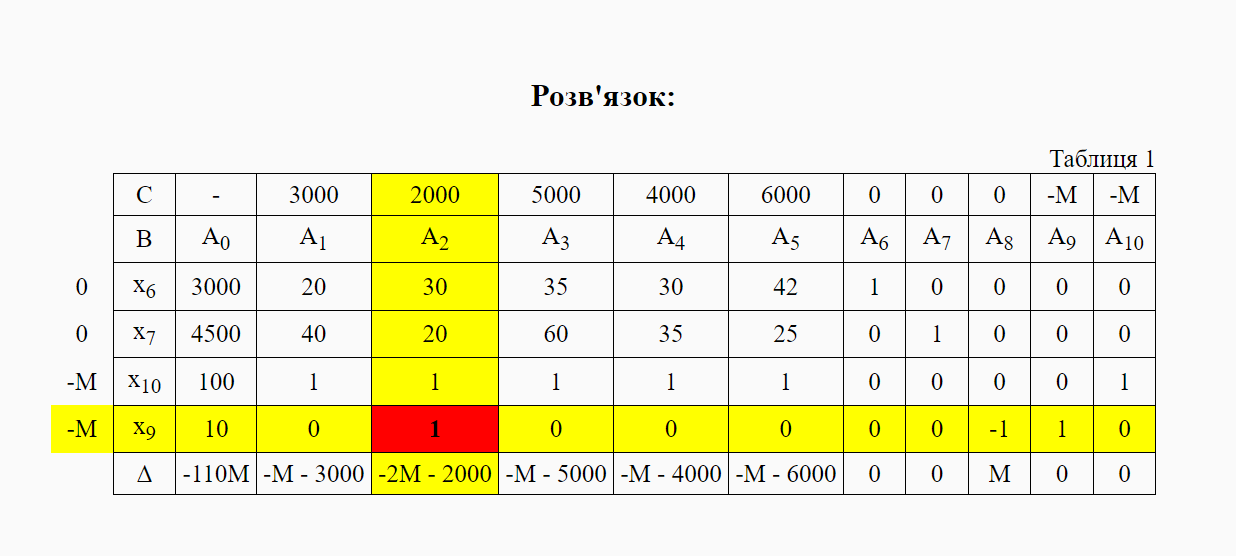
*Рис.3.9. Відповідь до поставленої задачі*

Проведемо тестування на інші можливі сценарії.

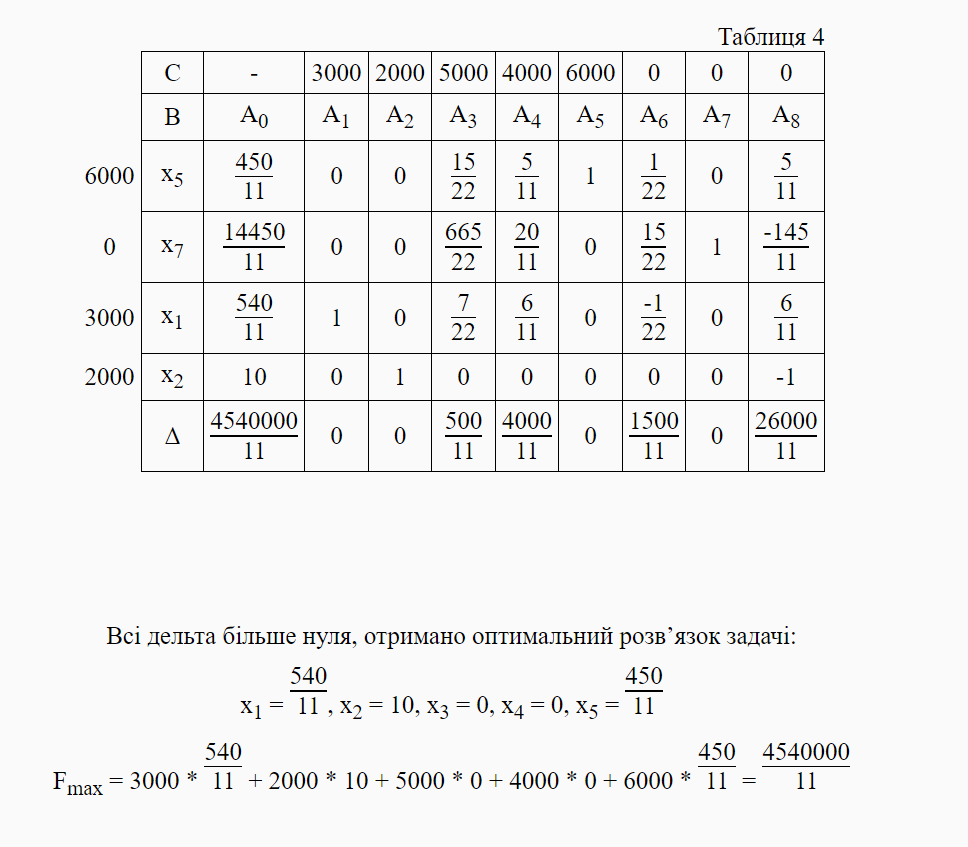
Змінимо вхідні дані щоб побачити як програма вирішує задачу цілочисельного програмування (рис. 3.10 – 3.18):

**

*Рис.3.10. Вхідні дані*

**

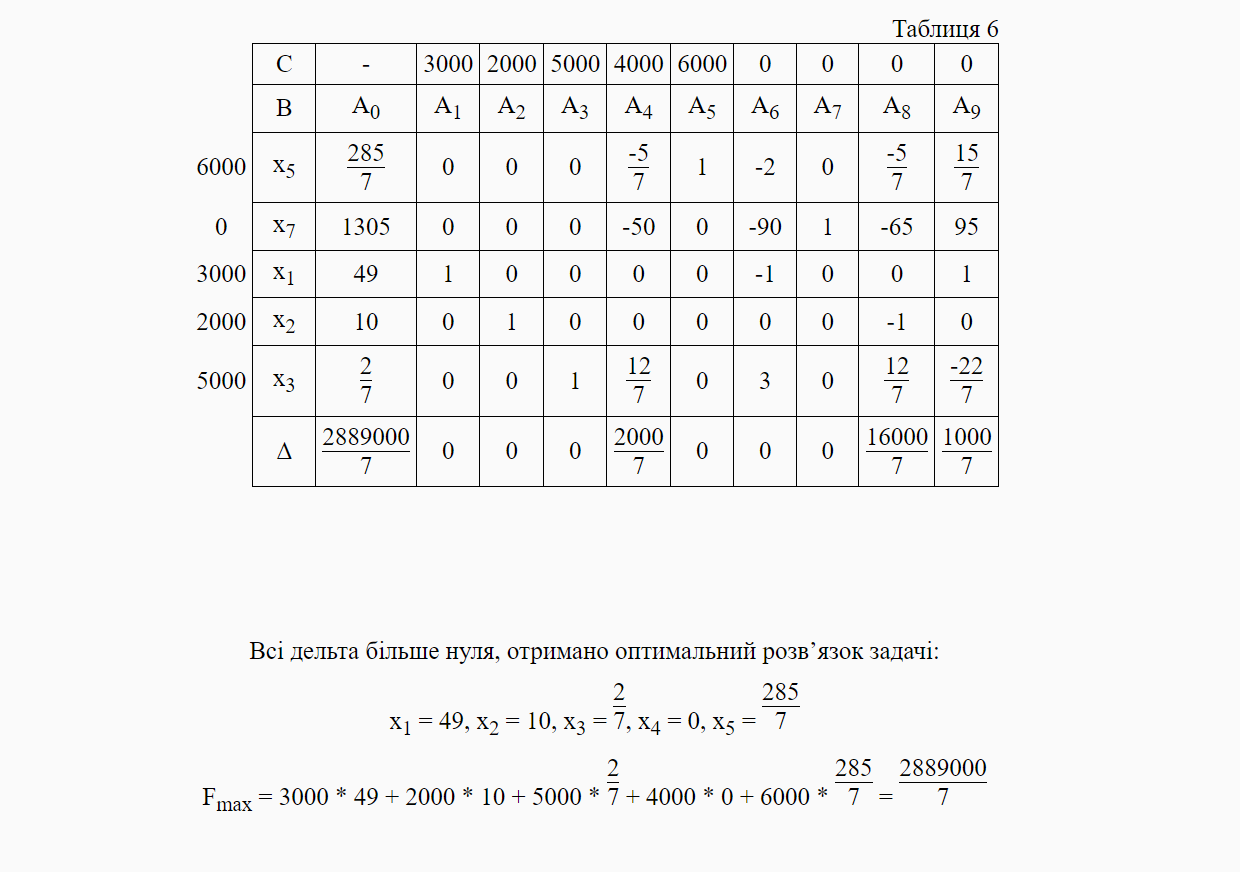
*Рис.3.11. Вихідна симплекс таблиця*

**

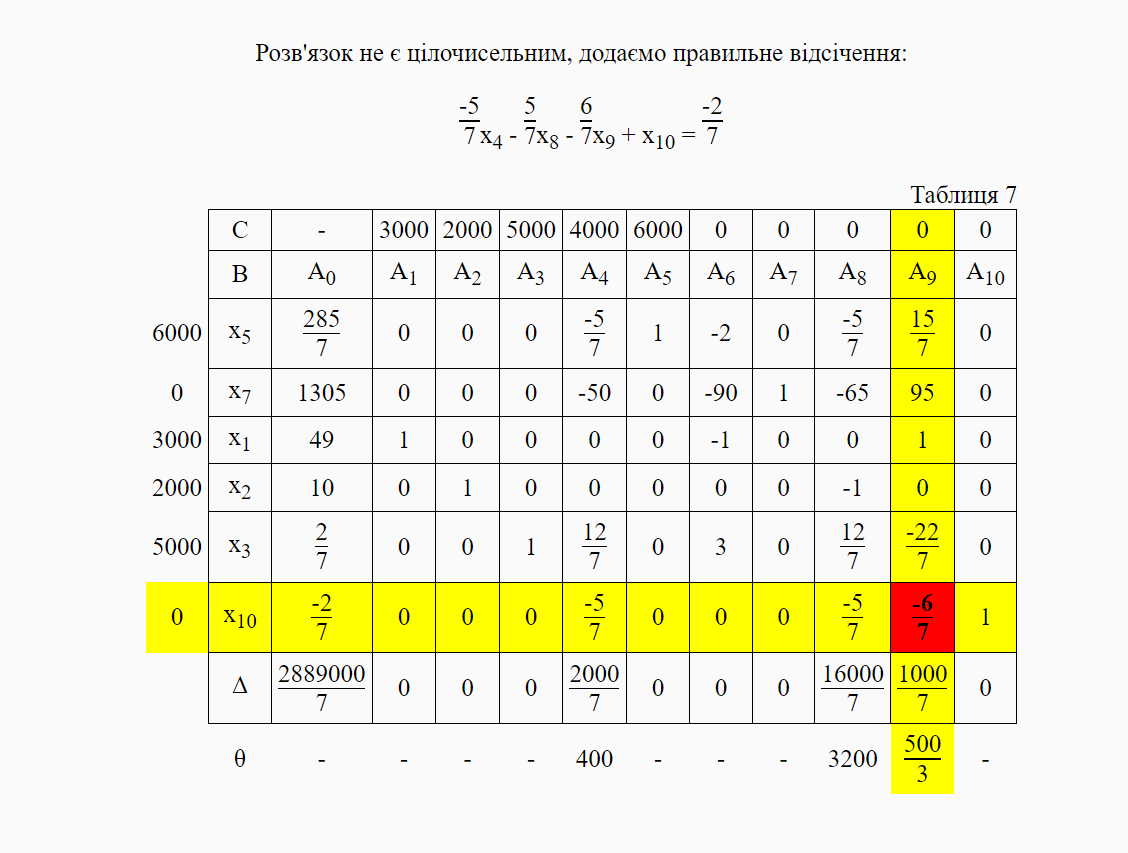
*Рис.3.12. Оптимальне рішення що не є цілочисельним*

**

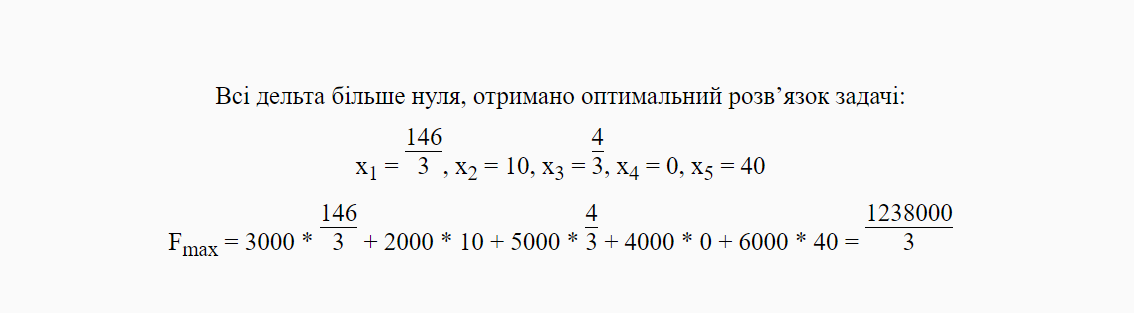
*Рис.3.13. Перше правильне відсічення*

**

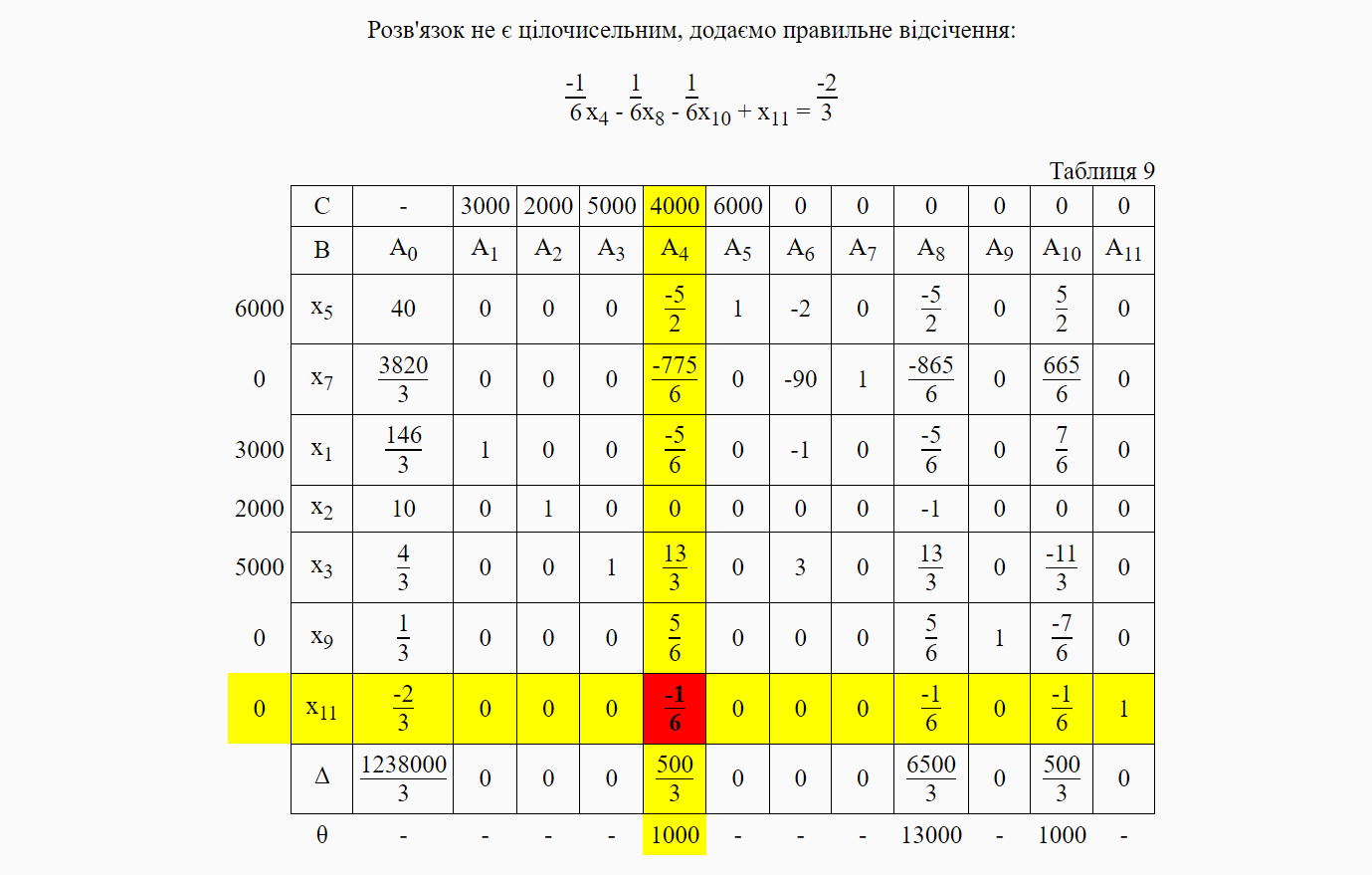
*Рис.3.14. Наступне оптимальне рішення що не є цілочисельним*

**

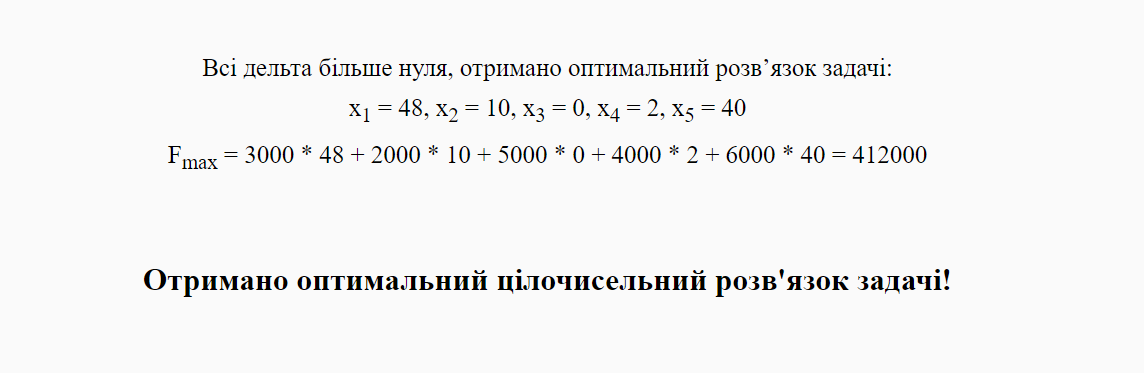
*Рис.3.15. Друге правильне відсічення*

**

*Рис.3.16. Наступне оптимальне рішення що не є цілочисельним*

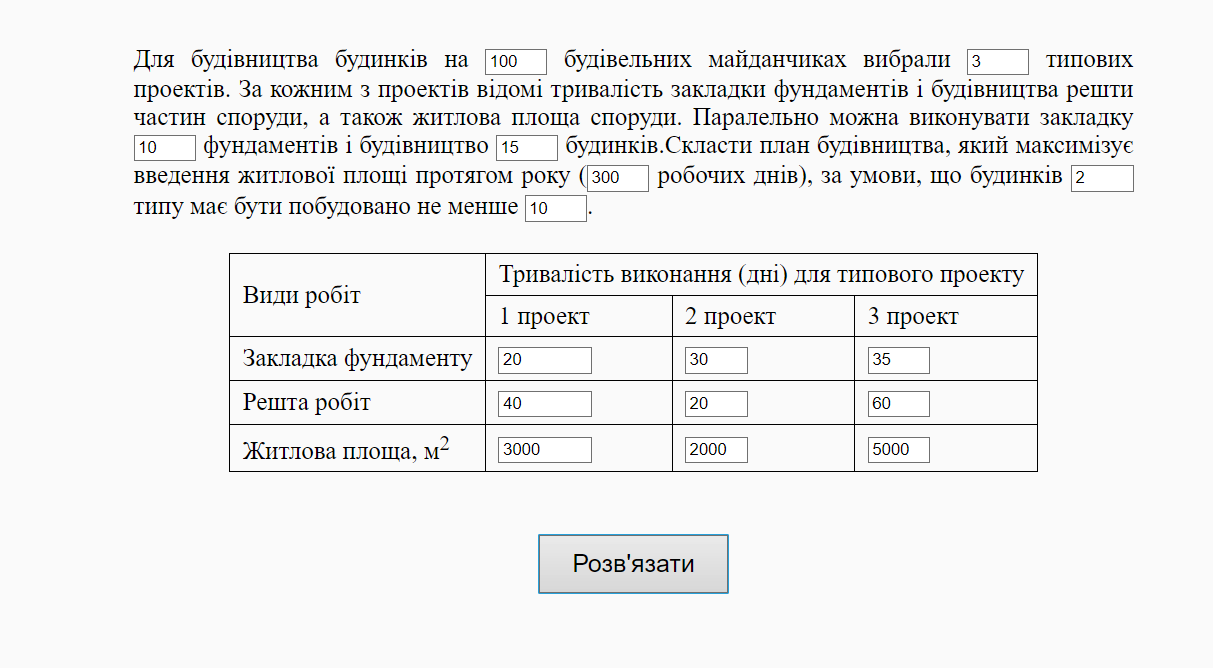
**

*Рис.3.17. Третє правильне відсічення*

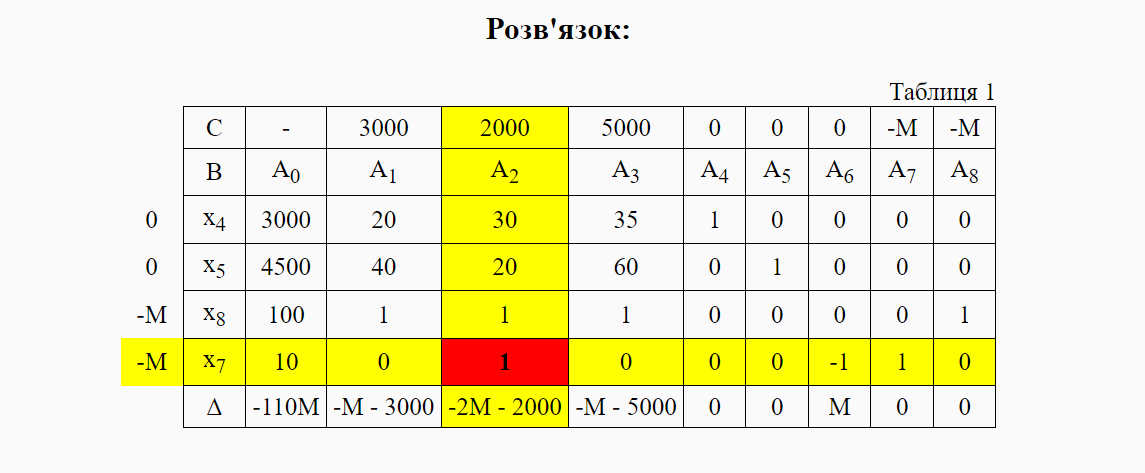
**

*Рис.3.18. Оптимальне цілочисельне рішення*

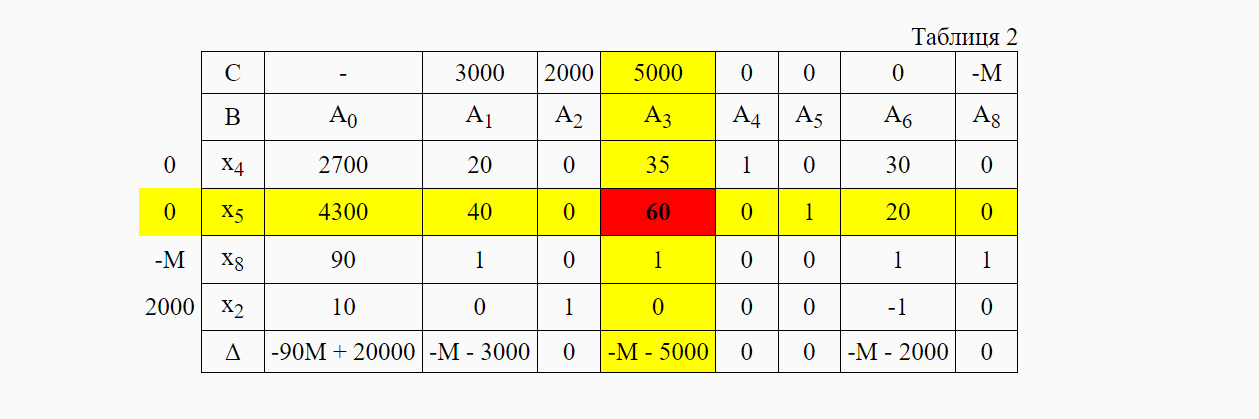
Спробуємо вирішити задачу з меншою кількістю проектів (рис. 3.19 – 3.25):

**

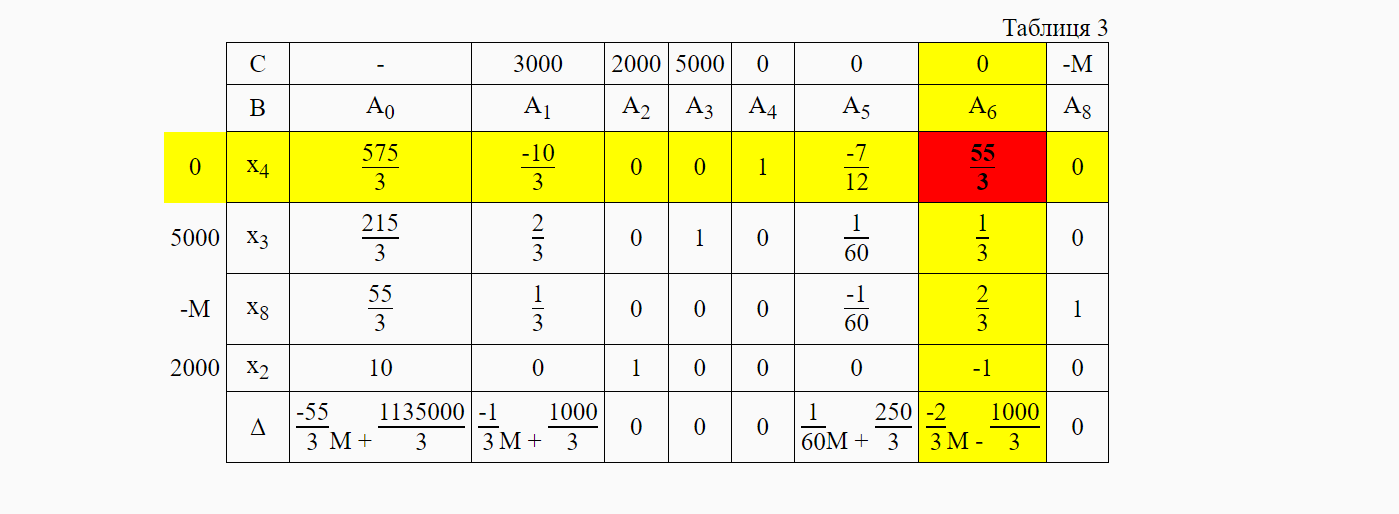
*Рис.3.19. Вхідні дані*

**

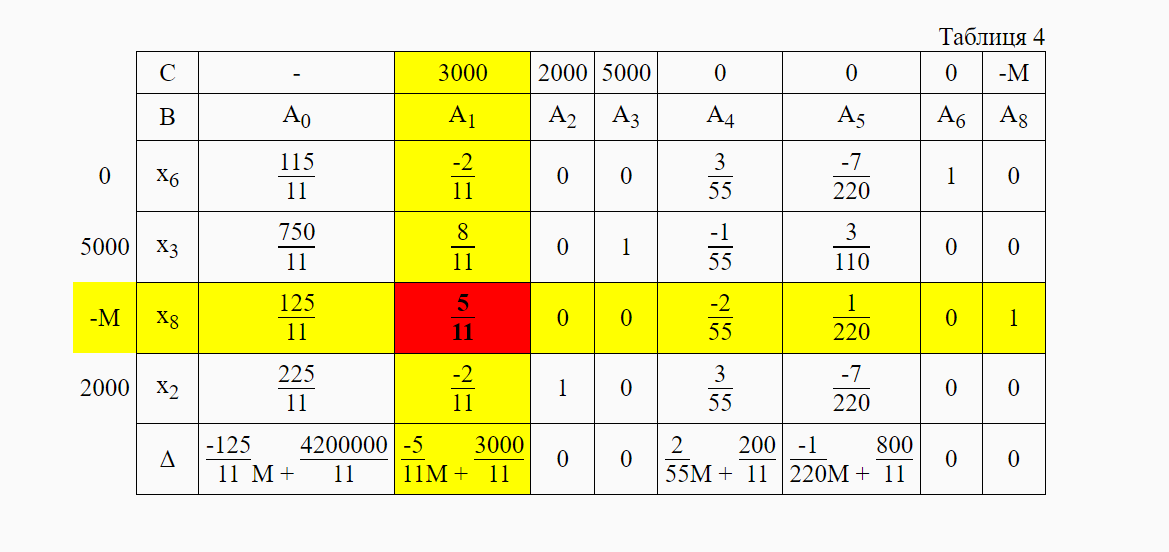
*Рис.3.20. Перша симплекс таблиця*

**

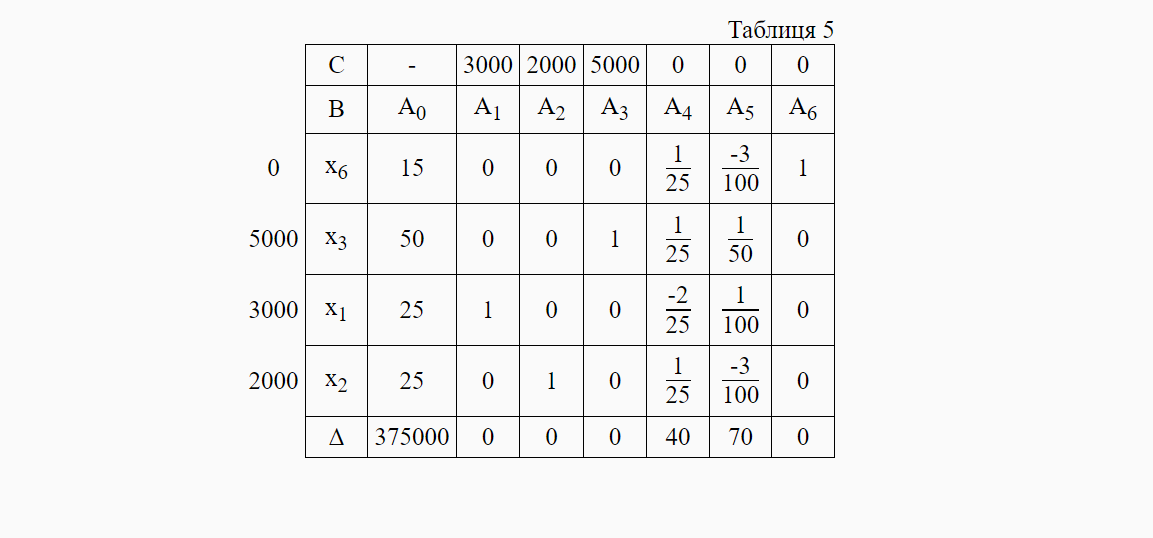
*Рис.3.21. Друга симплекс таблиця*

**

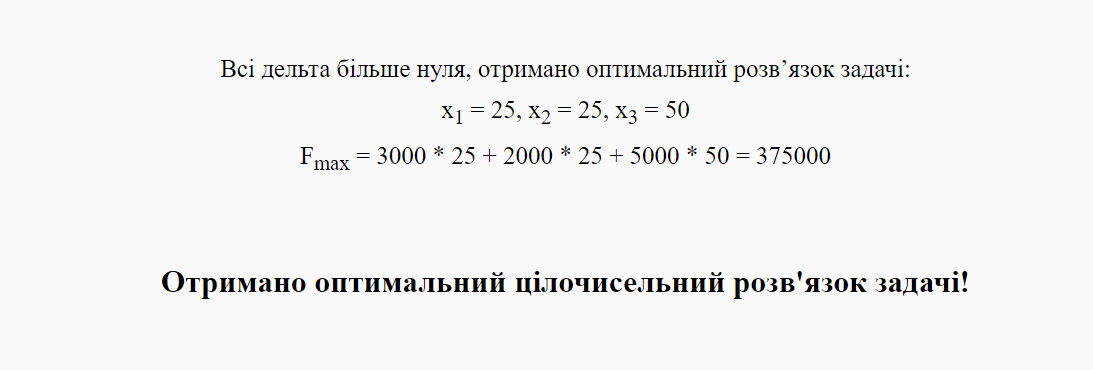
*Рис.3.22. Третя симплекс таблиця*

**

*Рис.3.23. Четрверта симплекс таблиця*

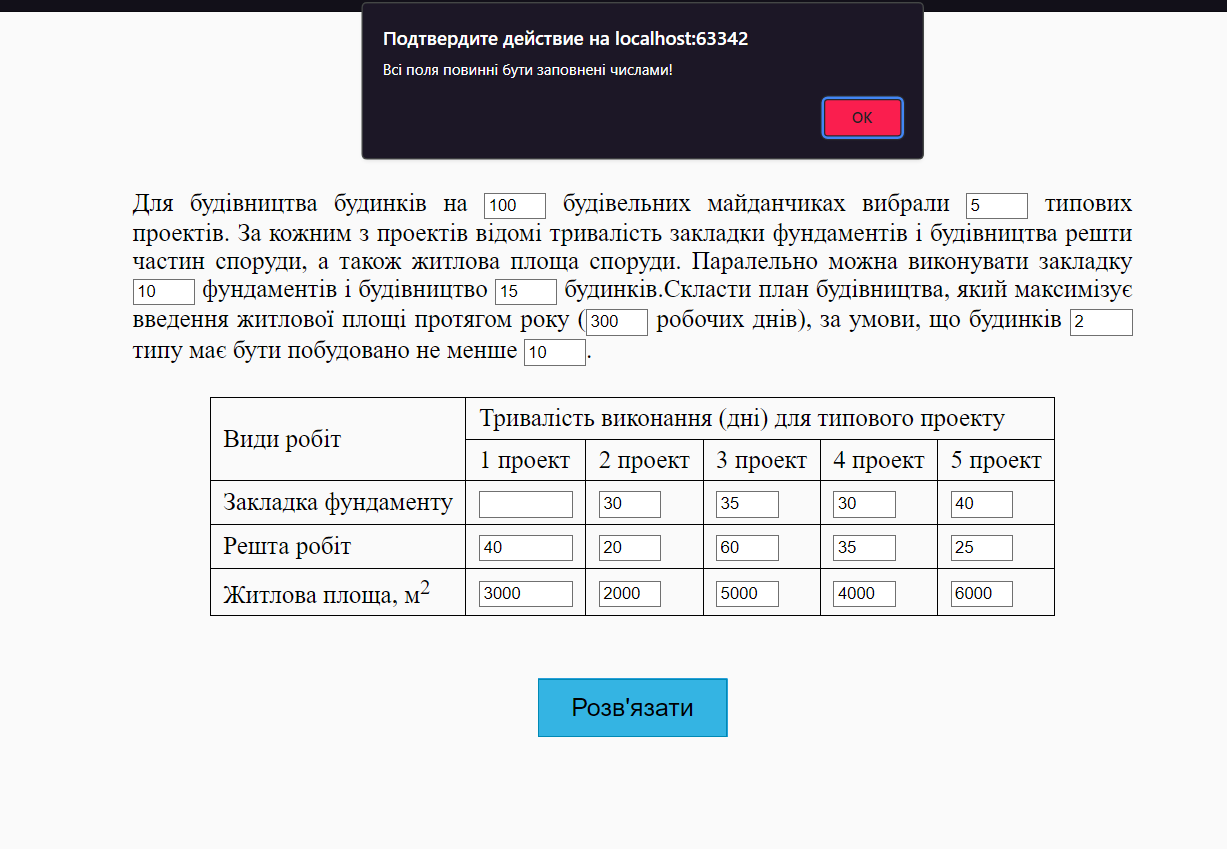
**

*Рис.3.24. П’ята симплекс таблиця*

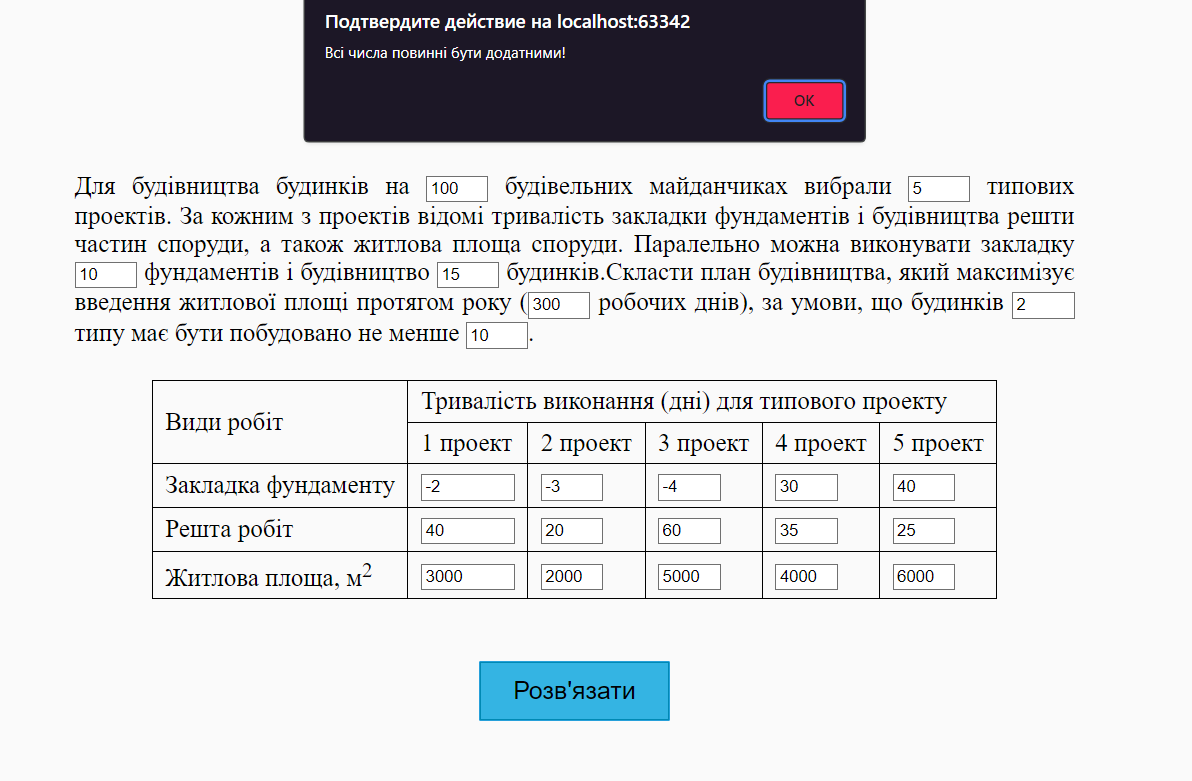
**

*Рис.3.25. Оптимальне цілочисельне рішення*

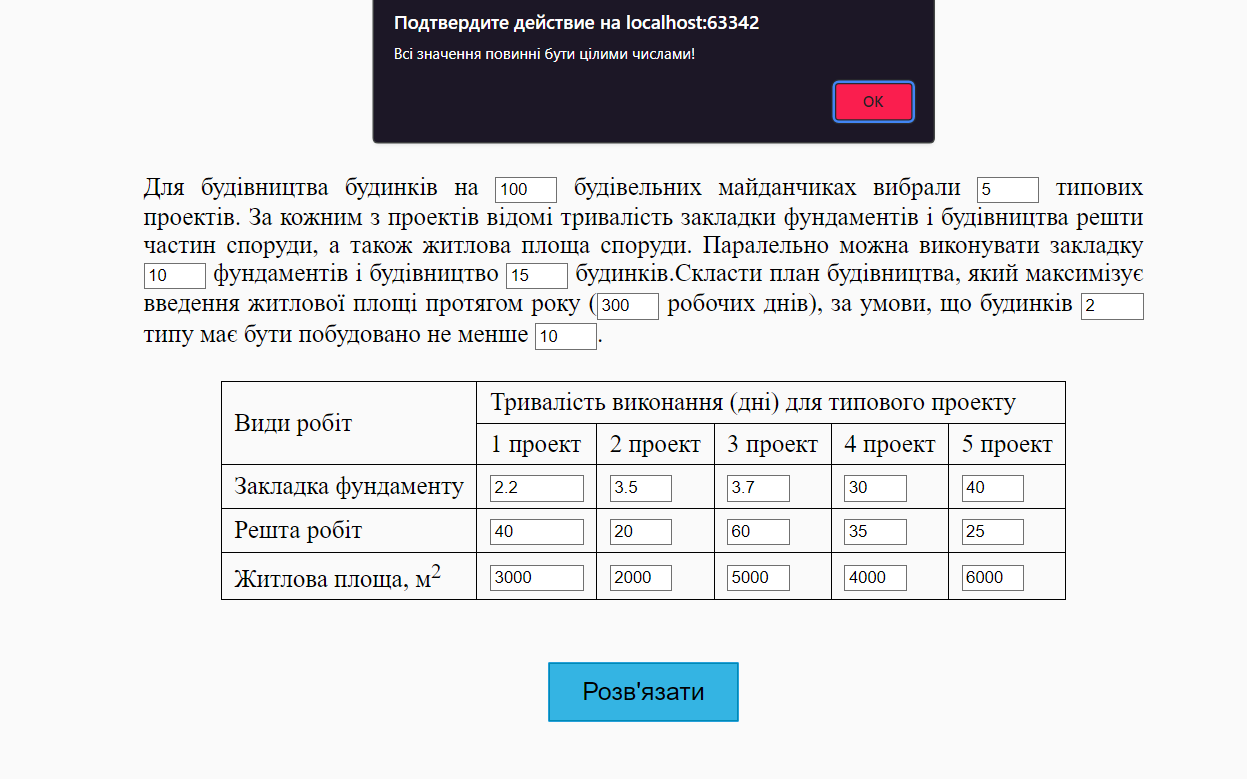
Проведемо тестування на можливі помилки вхідних даних (рис. 3.26 – 3.30):

**

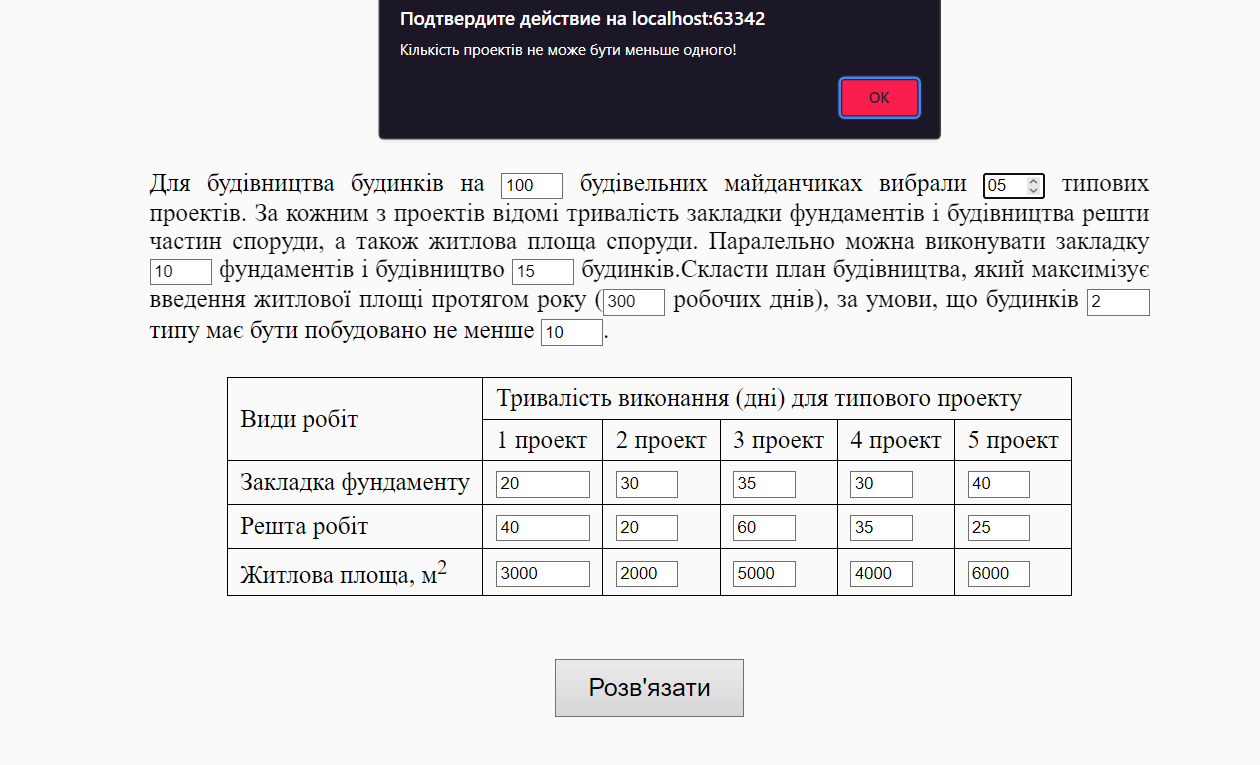
*Рис.3.26. Помилка пустих полей*

**

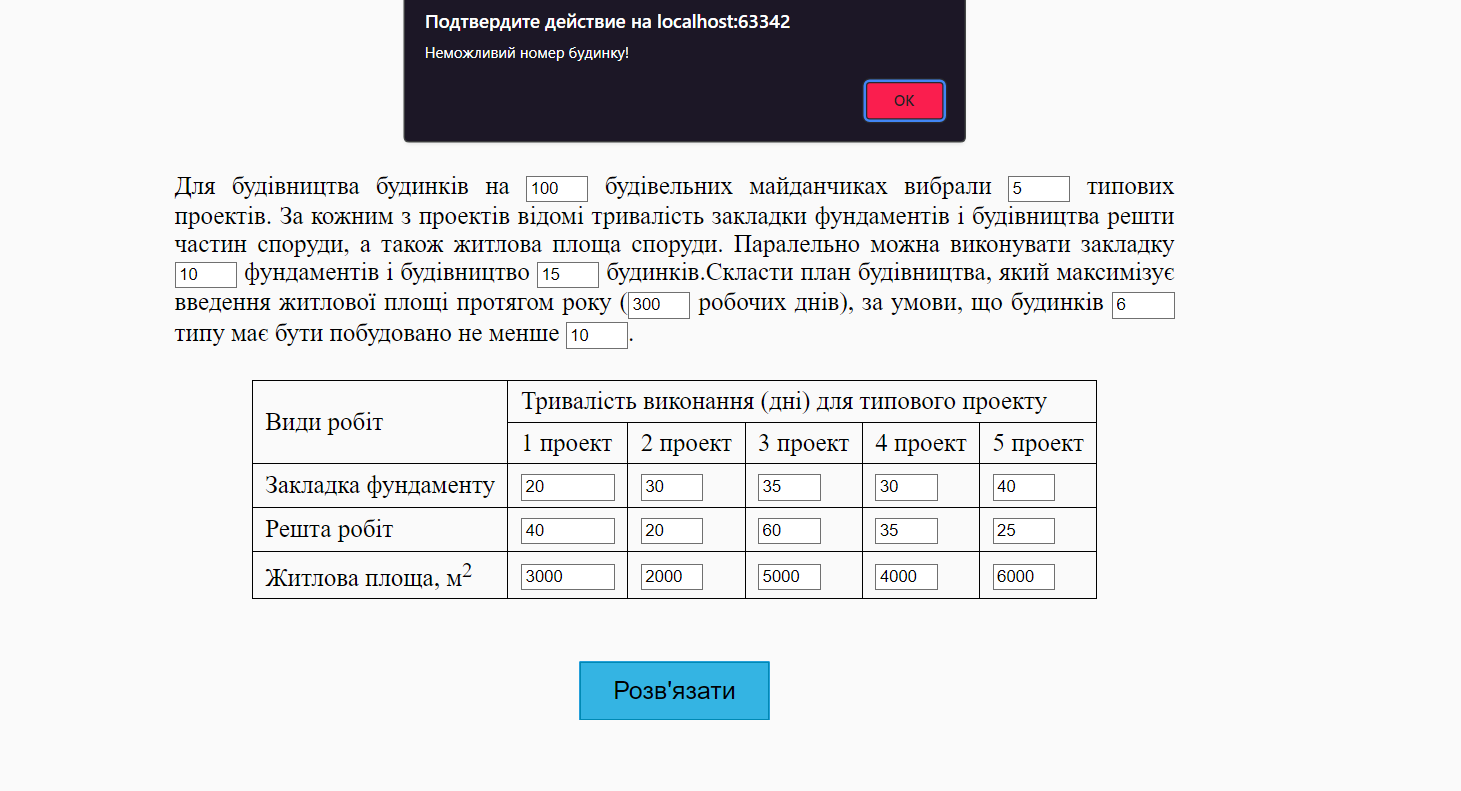
*Рис.3.27. Помилка відємних значень*

**

*Рис.3.28. Помилка дробових чисел*

**

*Рис.3.29. Помилка кількості проектів*

**

*Рис.3.30. Помилка номеру будинку*

Отже, розроблений алгоритм методу Гоморі ефективно вирішує нашу задачу та подібні до неї. Створене програмне забезпечення можна застосовувати для розв'язання реальних завдань, що виникають на підприємствах.

**Висновки до розділу 3:**

У третьому розділі було обрано JavaScript як основний інструментальний засіб для реалізації програмного додатку з розв'язання цілочислової задачі лінійного програмування. Вибір JS був обґрунтований його універсальністю, широкою підтримкою спільноти розробників, багатством сучасних фреймворків та бібліотек, а також можливістю створення динамічних і інтерактивних веб-додатків.

Завдяки застосуванню JavaScript та його потужних інструментів вдалося розробити ефективний та зручний програмний додаток, який дозволяє користувачам швидко та точно розв'язувати цілочислові задачі лінійного програмування, забезпечуючи високу продуктивність та стабільність роботи.

## ВИСНОВКИ

У даній курсовій роботі було розглянуто задачу математичного програмування, пов’язану з максимізацією житлової площі при будівництві будинків на 100 будівельних майданчиках з використанням п'яти типових проектів. Для розв'язання цієї задачі було сформовано відповідну математичну модель, обрано метод Гоморі та реалізовано програмний додаток на JavaScript.

На першому етапі роботи було детально проаналізовано вимоги задачі, сформовано математичну модель, яка включає цільову функцію та систему обмежень, враховуючи такі параметри, як тривалість робіт, житлова площа будинків і обмеження на кількість будівництва будинків певних типів.

На другому етапі було обґрунтовано вибір методу Гоморі для розв'язання задачі цілочислового програмування. Метод Гоморі, завдяки своїй здатності поступово виключати дробові рішення за допомогою спеціальних зрізів, дозволив ефективно знайти оптимальний цілочисловий розв'язок задачі.

На третьому етапі було обрано JavaScript як основний інструментальний засіб для реалізації програмного додатку. Створено зручний та інтуїтивно зрозумілий користувацький інтерфей, а також ефективний алгоритм для рішення задачі цілочисельного програмування. Досягнено високої продуктивності та швидкої взаємодії з користувачем у реальному часі.

Загалом, результати курсової роботи демонструють успішну реалізацію цілочислової задачі лінійного програмування для будівельного проекту з використанням сучасних математичних методів та інструментальних засобів програмування. Створений програмний додаток дозволяє швидко та точно розв’язувати подібні задачі, що може бути корисним у практичній діяльності будівельних компаній та інших підприємств, які займаються плануванням та оптимізацією ресурсів.

## СПИСОК ДЖЕРЕЛ

1. Локтікова Т., Кушнір Н. Методичні рекомендації для проведення лабораторних занять з навчальної дисципліни "Математичні методи дослідження операцій". – Житомир: Житомирська політехніка, 2023. – 65 с.
2. Локтікова Т., Кушнір Н. Методичні рекомендації для виконання курсової роботи. – Житомир: Житомирська політехніка, 2023. – 35 с.
3. JavaScript Documentation [Електронний ресурс] – Режим доступу: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>.
4. . Фрімен Е., Робсон Е. Head First. Програмування на JavaScript. – Харків:Видавництво "Фабула", 2022. – 672 с.
5. Сучасний підручник з JavaScript [Електронний ресурс] – Режим доступудо ресурсу: https://uk.javascript.info/.

## ДОДАТКИ

**index.html:**

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Курсова робота ММДО</title>  
 <link rel="stylesheet" href="styles.css">  
</head>  
<body>  
<div class="main\_wrapper">  
 <main>  
  
 <div id="task">  
 <div id="task\_text">  
 Для будівництва будинків на <input min="0" type="number" value="100"> будівельних майданчиках  
 вибрали <input min="1" id="project\_n" type="number" value="5"> типових проектів. За кожним з проектів відомі  
 тривалість закладки фундаментів і будівництва решти частин  
 споруди, а також житлова площа споруди. Паралельно  
 можна виконувати закладку <input min="0" type="number" value="10"> фундаментів і будівництво  
 <input min="0" type="number" value="15"> будинків.Скласти план будівництва, який максимізує введення  
 житлової площі протягом року (<input min="0" type="number" value="300"> робочих днів), за умови, що  
 будинків <input id="project\_restrict" min="0" max="5" type="number" value="2"> типу має бути побудовано не менше <input min="0" type="number" value="10">.  
 </div>  
  
 <div id="task\_table">  
  
 <table>  
 <tbody>  
 <tr>  
 <td rowspan="2">Види робіт</td>  
 <td id="long-cell" colspan="5">Тривалість виконання (дні) для типового проекту</td>  
 </tr>  
 <tr id="project\_row">  
 <td>1 проект</td>  
 <td>2 проект</td>  
 <td>3 проект</td>  
 <td>4 проект</td>  
 <td>5 проект</td>  
 </tr>  
 <tr>  
 <td>Закладка фундаменту</td>  
 <td><input min="0" type="number" value="20"></td>  
 <td><input min="0" type="number" value="30"></td>  
 <td><input min="0" type="number" value="35"></td>  
 <td><input min="0" type="number" value="30"></td>  
 <td><input min="0" type="number" value="40"></td>  
 </tr>  
 <tr>  
 <td>Решта робіт</td>  
 <td><input min="0" type="number" value="40"></td>  
 <td><input min="0" type="number" value="20"></td>  
 <td><input min="0" type="number" value="60"></td>  
 <td><input min="0" type="number" value="35"></td>  
 <td><input min="0" type="number" value="25"></td>  
 </tr>  
 <tr>  
 <td>Житлова площа, м<sup>2</sup></td>  
 <td><input min="0" type="number" value="3000"></td>  
 <td><input min="0" type="number" value="2000"></td>  
 <td><input min="0" type="number" value="5000"></td>  
 <td><input min="0" type="number" value="4000"></td>  
 <td><input min="0" type="number" value="6000"></td>  
 </tr>  
 </tbody>  
 </table>  
 </div>  
  
 <button id="start">Розв'язати</button>  
 <div id="error" ><h4>Всі поля мають бути заповнені числами!</h4></div>  
 </div>  
  
 <div id="result">  
 <h3>Розв'язок:</h3>  
 <div id="solve" class="flex-center">  
  
 </div>  
 <div id="final\_res" class="flex-center">  
  
 </div>  
  
 </div>  
 </main>  
</div>  
  
<script src="math.js"></script>  
<script src="simplexMethod.js"></script>  
<script src="IntegerMethod.js"></script>  
<script src="script.js"></script>  
  
</body>  
</html>

**integerMethod.js:**

class IntegerMethod {  
 constructor(table) {  
 this.table = table;  
 this.x\_cut = null;  
 }  
  
 calculateCutOff() {  
 let n = this.table.arr.length - 2, m = this.table.arr[0].length - 2;  
 let new\_x = `x${this.table.arr[0].length - 2}`;  
  
  
 let new\_row = [new GreatNumber(0, 0), new\_x];  
 for (let j = 2; j < m + 2; j++) {  
 new\_row.push(new Frac(0, 1).sub(this.table.arr[this.x\_cut][j].sub(*Math*.floor(this.table.arr[this.x\_cut][j]))));  
 }  
  
 new\_row.push(new Frac(1, 1));  
  
 let new\_col = [new GreatNumber(0, 0), new\_x, ...new *Array*(m).fill(new Frac(0, 1))];  
  
 let new\_arr = AdditionalFunctions.*copy\_arr*(this.table.arr).slice(0, n + 1);  
 for (let i = 0; i < n + 1; i++)  
 new\_arr[i].push(new\_col[i]);  
  
 new\_arr.push(new\_row);  
 new\_arr.push(["", "Δ", ...new *Array*(m + 1).fill(0)]);  
  
 //console.log(AdditionalFunctions.copy\_arr(new\_arr), this.table.x\_n);  
  
 return {  
 simplexTable: new DualSimplexArr(new\_arr, this.table.x\_n),  
 newRestrictionHTML: this.restrictionHTML(n, m, new\_x)  
 };  
 }  
  
 restrictionHTML(n, m, new\_x) {  
 let newRestrictionHTML = "<div>Розв'язок не є цілочисельним, додаємо правильне відсічення:</div>";  
 let newRestriction = [];  
  
 for (let j = 2; j < m + 2; j++) {  
 let x = this.table.arr[this.x\_cut][j], round\_x = *Math*.floor(this.table.arr[this.x\_cut][j]);  
 newRestriction.push(new Frac(0, 1).sub(x.sub(round\_x)));  
 }  
  
 newRestriction.push(new Frac(1, 1));  
 let coef = newRestriction.shift();  
  
  
  
 let vars = this.table.arr[1].slice(3);  
 vars.push(new\_x);  
 vars = IntegerMethod.*#varsToHTML*(vars);  
  
 newRestrictionHTML += `<div class=\"equations\">${IntegerMethod.*genEq*(vars, newRestriction, coef, "=")}</div>`;  
 return newRestrictionHTML;  
 }  
  
 checkIsResult() {  
  
 for (let j = 3; j < this.table.x\_n + 3; j++) {  
 //console.log(this.table.arr[1][j]);  
 for (let i = 2; i < this.table.arr.length - 1; i++) {  
 //console.log(` ${this.table.arr[i][1]}`);  
 if (this.table.arr[1][j] === this.table.arr[i][1]) {  
 let t = this.table.arr[i][2];  
 if (t.b !== 1) {  
 this.x\_cut = i;  
 return false;  
 }  
  
 }  
 }  
  
 }  
  
 return true;  
 }  
  
 static *#varsToHTML*(vars) {  
 let res = [];  
 for (let i = 0; i < vars.length; i++)  
 res.push(`x<sub>${vars[i].substr(1)}</sub>`);  
 return res;  
 }  
  
 static *genEq*(vars, coefs, freeCoef, sign) {  
 let eq = IntegerMethod.*#coefsToString*(coefs, vars);  
 if (!eq)  
 eq = "0";  
 eq += ` ${sign} ${freeCoef}`;  
 return `<div class='equation'>${eq}</div>`;  
 }  
  
 static *#coefsToString*(coefs, vars) {  
 let res = "";  
 for (let i = 0; i < coefs.length; i++) {  
 if (coefs[i] == 1)  
 res += ` + ${vars[i]}`;  
 else if (coefs[i] == -1)  
 res += ` - ${vars[i]}`;  
 else if (coefs[i] < 0)  
 if (coefs[i] instanceof Frac)  
 if (res === "")  
 res += ` + ${coefs[i]}${vars[i]}`;  
 else  
 res += ` - ${coefs[i].abs()}${vars[i]}`;  
 else  
 res += ` - ${*Math*.abs(coefs[i])}${vars[i]}`;  
 else if (coefs[i] > 0)  
 res += ` + ${coefs[i]}${vars[i]}`;  
 }  
  
 if (res[1] === "-") {  
 res = res.substr(3);  
 res = "-" + res;  
 return res;  
 }  
 return res.substr(3);  
 }  
}  
  
  
  
class DualSimplexArr extends SimplexArr{  
 constructor(...arr) {  
 super(...arr);  
 this.theta\_arr = null;  
 }  
 calcPivotElement() {  
 let min\_j = null, min\_i\_val = this.arr[2][2], min\_i = 2;  
  
 for (let i = 2; i < this.\_n + 1; i++)  
 if (min\_i\_val > this.arr[i][2])  
 min\_i\_val = this.arr[i][2];  
  
 if (min\_i\_val >= 0)  
 return;  
  
 for (min\_i = 2; min\_i < this.\_n + 1; min\_i++) {  
 if (this.arr[min\_i][2] === min\_i\_val) {  
 this.q = {  
 arr\_str: [],  
 arr: [],  
 element: null  
 }  
 this.theta\_arr = ["", "θ", "-"];  
  
 for (let j = 3; j < this.\_m + 2; j++)  
 if (this.arr[min\_i][j] < 0) {  
 if (min\_j === null)  
 min\_j = j;  
 else  
 if (this.arr[this.\_n + 1][j].div( this.arr[min\_i][j]).abs() < this.arr[this.\_n + 1][min\_j].div(this.arr[min\_i][min\_j]).abs())  
 min\_j = j;  
 this.theta\_arr.push(this.arr[this.\_n + 1][j].div(this.arr[min\_i][j]).abs());  
 }  
 else  
 this.theta\_arr.push("-");  
 }  
  
 if (min\_j !== null) {  
 break;  
 }  
  
 if (min\_i === (this.\_n)) {  
 this.\_fl = true;  
 return;  
 }  
  
 }  
  
 this.pivotElement = {  
 i: min\_i,  
 j: min\_j  
 };  
 }  
  
 next() {  
 if (!this.checkIsNextPossible())  
 return null;  
 return new DualSimplexArr(this.\_calcNextArr().slice(0, this.\_n + 2), this.x\_n);  
 }  
}

**math.js:**

class Frac {  
 constructor(a, b) {  
 this.a = a;  
 this.b = b;  
  
 if (a < 0 && b < 0) {  
 this.a = *Math*.abs(a);  
 this.b = *Math*.abs(b);  
 }  
 else if (b < 0) {  
 this.a = -a;  
 this.b = *Math*.abs(b);  
 }  
  
 let t = Frac.*#gcd*(this.a, this.b);  
 this.a /= t;  
 this.b /= t;  
 }  
  
 add(other) {  
 other = Frac.*convertOther*(other);  
 return new Frac(this.a \* other.b + other.a \* this.b, this.b \* other.b);  
 }  
  
 sub(other) {  
 other = Frac.*convertOther*(other);  
 return new Frac(this.a \* other.b - other.a \* this.b, this.b \* other.b);  
 }  
  
 mul(other) {  
 other = Frac.*convertOther*(other);  
 return new Frac(this.a \* other.a, this.b \* other.b);  
 }  
  
 div(other) {  
 other = Frac.*convertOther*(other);  
 return this.mul(new Frac(other.b, other.a));  
 }  
  
 abs() {  
 return new Frac(*Math*.abs(this.a), this.b);  
 }  
  
 valueOf() {  
 return this.a / this.b;  
 }  
 toString() {  
 if (this.b === 1)  
 return `${this.a}`;  
 return `<div class="fraction">${this.a} <div class="line"></div> ${this.b}</div>`;  
 }  
  
 static *convertOther*(other) {  
 if (!(other instanceof Frac))  
 return new Frac(other, 1);  
 else  
 return other;  
 }  
  
 static *#gcd*(a,b) {  
 a = *Math*.abs(a);  
 b = *Math*.abs(b);  
 if (b > a) {let temp = a; a = b; b = temp;}  
 while (true) {  
 if (b === 0) return a;  
 a %= b;  
 if (a === 0) return b;  
 b %= a;  
 }  
 }  
}  
  
class GreatNumber {  
 constructor(m, a) {  
 if (!(m instanceof Frac))  
 m = new Frac(m, 1);  
 this.m = m;  
  
 if (!(a instanceof Frac))  
 a = new Frac(a, 1);  
 this.a = a;  
 }  
  
 add(b) {  
 if (b instanceof GreatNumber)  
 return new GreatNumber(this.m.add(b.m), this.a.add(b.a));  
 else  
 return new GreatNumber(this.m, this.a.add(b));  
 }  
  
 sub(b) {  
 if (b instanceof GreatNumber)  
 return new GreatNumber(this.m.sub(b.m), this.a.sub(b.a));  
 else  
 return new GreatNumber(this.m, this.a.sub(b));  
 }  
  
 mul(b) {  
 if (b instanceof GreatNumber)  
 throw new Error("Спроба множення двух об'єктів классу BigNum");  
 else  
 return new GreatNumber(this.m.mul(b), this.a.mul(b));  
 }  
  
 div(b) {  
 if (b instanceof GreatNumber)  
 throw new Error("Спроба ділення двух об'єктів классу BigNum");  
 else  
 return new GreatNumber(this.m.div(b), this.a.div(b));  
 }  
  
 abs() {  
 return new GreatNumber(this.m.abs(), this.a.abs());  
 }  
  
 toString() {  
 let res = "";  
  
 if (this.m == 1)  
 res = `M`;  
 else if (this.m == -1)  
 res = `-M`;  
 else  
 res = `${this.m.toString()}M`;  
  
 if (this.m == 0)  
 return this.a.toString();  
  
 if (this.a < 0)  
 res += ` - ${this.a.abs().toString()}`;  
 else if (this.a != 0)  
 res += ` + ${this.a.toString()}`;  
  
 return res;  
 }  
  
 valueOf() {  
 return this.m \* 9999999999999999999 + this.a;  
 }  
}  
  
class AdditionalFunctions {  
 static *copy\_arr*(arr) {  
 let new\_arr = [];  
 for (let i = 0; i < arr.length; i++) {  
 new\_arr.push([])  
 for (let j = 0; j < arr[0].length; j++)  
 if (arr[i][j].constructor.name === Frac.constructor.name)  
 new\_arr[i].push(new Frac(arr[i][j].a, arr[i][j].b));  
 else if (arr[i][j].constructor.name === GreatNumber.constructor.name)  
 new\_arr[i].push(new GreatNumber(arr[i][j].m, arr[i][j].a));  
 else  
 new\_arr[i].push(arr[i][j]);  
 }  
 return new\_arr;  
 }  
  
 static *delete\_row*(arr, delete\_i) {  
 let new\_arr = AdditionalFunctions.*copy\_arr*(arr);  
 for (let i = 0; i < new\_arr.length; i++)  
 new\_arr[i].splice(delete\_i, 1);  
 return new\_arr;  
 }  
}

**script.js:**

let *project\_n* = 5;  
let *resultElement* = *document*.getElementById("result");  
  
*document*.querySelector("#project\_n").oninput = function () {  
 let proj\_inp = *document*.querySelector("#project\_n");  
  
 if (!check\_input(proj\_inp)) {  
 proj\_inp.value = *project\_n*;  
 return;  
 }  
  
 let t = parseInt(proj\_inp.value);  
  
 if (t < 1) {  
 show\_error("Кількість проектів не може бути меньше одного!");  
 proj\_inp.value = *project\_n*;  
 return;  
 }  
  
 let dif = t - *project\_n*;  
  
 if (dif > 0) {  
 for (let i = 0; i < dif; i++)  
 *document*.querySelector("#project\_row").innerHTML += `<td>${*project\_n* + i + 1} проект</td>`;  
 for (let i = 0; i < 3; i++) {  
 let row = *document*.querySelector(`#task\_table tr:nth-child(${i + 3})`);  
 for (let j = 0; j < dif; j++) {  
 let td = *document*.createElement("td");  
 td.innerHTML = "<input min=\"0\" type=\"number\" value=\"0\">";  
 row.appendChild(td);  
 }  
  
 }  
 } else {  
 for (let i = 0; i < 4; i++) {  
 let row = *document*.querySelector(`#task\_table tr:nth-child(${i + 2})`);  
 for (let j = 0; j < *Math*.abs(dif); j++) {  
 row.children[row.children.length - 1].outerHTML = "";  
 }  
  
 }  
 }  
  
 *project\_n* = t;  
 *document*.querySelector("#long-cell").colSpan = *project\_n*;  
 *document*.querySelector("#project\_restrict").max = *project\_n*;  
}  
  
function check\_input(inp) {  
 let t = parseFloat(inp.value);  
  
 if (isNaN(t) || t < 0 || (t - *Math*.floor(t) !== 0)) {  
 if (isNaN(t))  
 show\_error("Всі поля повинні бути заповнені числами!");  
 else if (t < 0)  
 show\_error("Всі числа повинні бути додатними!");  
 else if (t - *Math*.floor(t) !== 0)  
 show\_error("Всі значення повинні бути цілими числами!");  
  
 return false;  
 }  
 return true;  
}  
  
function check\_inputs() {  
 let fl = true;  
 *document*.querySelectorAll("input").forEach((inp) => {  
 if (!fl)  
 return;  
 if (!check\_input(inp))  
 fl = false;  
 else if (inp.id === "project\_n") {  
 if (!fl)  
 inp.value = *project\_n*;  
 if (parseFloat(inp.value) < 1) {  
 show\_error("Кількість проектів не може бути меньше одного!");  
 fl = false;  
 }  
 }  
 })  
 return fl;  
}  
  
  
*document*.getElementById("start").onclick = () => {  
 if (!check\_inputs())  
 return;  
  
 let inp = *document*.querySelector("#project\_restrict");  
 if (inp.id === "project\_restrict") {  
 if (parseFloat(inp.value) > *project\_n*) {  
 show\_error("Неможливий номер будинку!");  
 return;  
 }  
 }  
  
 let arr = [];  
 for (let i = 0; i < 3; i++) {  
 let tr = *document*.querySelector("#task\_table tbody").children[2 + i];  
 arr.push([]);  
  
 for (let j = 0; j < *project\_n*; j++) {  
 arr[i].push(tr.children[j + 1].children[0].value);  
 }  
 }  
  
 let const\_arr = [];  
 *document*.querySelectorAll("#task\_text input").forEach((inp) => {  
 const\_arr.push(parseInt(inp.value));  
 });  
  
 arr.forEach((val, i) => {  
 arr[i] = parseArr(val);  
 })  
  
 hide\_error();  
 *resultElement*.classList.add("visible");  
  
 setTimeout(() => {  
 *resultElement*.scrollIntoView({behavior: "smooth"});  
 }, 1);  
  
 let simplex\_arr = [  
 ["", "C", 0, ...arr[2], 0, 0, 0, new GreatNumber(-1, 0), new GreatNumber(-1, 0)],  
 ["", "B"],  
 [0, `x${*project\_n* + 1}`, const\_arr[4] \* const\_arr[2], ...arr[0], 1, 0, 0, 0, 0],  
 [0, `x${*project\_n* + 2}`, const\_arr[4] \* const\_arr[3], ...arr[1], 0, 1, 0, 0, 0],  
 [new GreatNumber(-1, 0), `x${*project\_n* + 5}`, const\_arr[0], ...new *Array*(*project\_n*).fill(1), 0, 0, 0, 0, 1],  
 [new GreatNumber(-1, 0), `x${*project\_n* + 4}`, const\_arr[6]],  
 ["", "Δ", ...new *Array*(*project\_n* + 6).fill(0)]  
 ];  
  
 for (let i = 0; i < *project\_n* + 6; i++)  
 simplex\_arr[1].push(`x${i}`);  
 for (let i = 0; i < *project\_n*; i++)  
 if ((i + 1) === const\_arr[5])  
 simplex\_arr[5].push(1);  
 else  
 simplex\_arr[5].push(0);  
  
 //console.log(simplex\_arr)  
 simplex\_arr[5].push(0, 0, -1, 1, 0);  
  
 let simplexTable = new SimplexArr(simplex\_arr, *project\_n*);  
  
 simplexSolve(simplexTable);  
  
}  
  
function show\_error(text) {  
 alert(text);  
 /\*let errorElement = document.getElementById("error");  
 errorElement.style.display = "block";  
 errorElement.querySelector('h4').innerText = text;  
 document.getElementById("start").style.display = "none";\*/  
}  
  
function hide\_error() {  
 /\*document.getElementById("error").classList.remove("visible");  
 document.getElementById("error").style.display = "none";  
 document.getElementById("start").style.display = "block";\*/  
}  
  
function checkNumbers(nums) {  
 for (let i = 0; i < nums.length; i++) {  
 let temp = parseFloat(nums[i]);  
 if (isNaN(temp))  
 return 2;  
 else if (temp < 0)  
 return 1;  
 else if (temp - *Math*.floor(temp) !== 0)  
 return 3;  
 }  
 return 0;  
}  
  
function parseArr(arr) {  
 let newArr = [];  
 arr.forEach((i) => {  
 newArr.push(parseFloat(i));  
 });  
 return newArr;  
}

**simplexMethod.js:**

function simplexSolve(simplexTable) {  
 SimplexVisual.*table\_i* = 1;  
 let solve = *document*.querySelector("#solve"), i = 1;  
 solve.innerHTML = "";  
 simplexTable.arr = refactorArr(simplexTable.arr);  
  
 simplexTable = simplexLoop(simplexTable, solve);  
 solve.innerHTML += new SimplexVisual(simplexTable).resultsToHTML();  
  
 if (simplexTable.checkResult() !== true)  
 return;  
  
 let integerTask = new IntegerMethod(simplexTable);  
 //console.log(integerTask, integerTask.checkIsResult());  
  
 let int\_i = 0;  
 while (!integerTask.checkIsResult()) {  
 let newRestrictionHTML = "";  
 ({simplexTable, newRestrictionHTML} = integerTask.calculateCutOff());  
 solve.innerHTML += newRestrictionHTML;  
  
 simplexTable = simplexLoop(simplexTable, solve);  
  
 //console.log(simplexTable);  
  
 solve.innerHTML += new SimplexVisual(simplexTable).resultsToHTML();  
  
 integerTask = new IntegerMethod(simplexTable);  
  
 if (i++ > 20) {  
 alert("Правильних відсічень більше 20, неможливо знайти оптимальний цілочисельний розв'язок!!!");  
 return;  
 }  
  
 }  
  
 if (simplexTable.checkResult() === true)  
 solve.innerHTML += "<div style='font-size: 25px; font-weight: bold;'>Отримано оптимальний цілочисельний розв'язок задачі!</div>";  
}  
  
function simplexLoop(simplexTable, solve) {  
 while (true) {  
 simplexTable.calcDelta();  
 simplexTable.calcPivotElement();  
  
 let simplexVisual = new SimplexVisual(simplexTable);  
 solve.innerHTML += simplexVisual.tableToHTML();  
  
 if (!simplexTable.checkIsNextPossible())  
 break;  
 else  
 simplexTable = simplexTable.next();  
 }  
 return simplexTable;  
}  
  
class SimplexArr {  
 \_n;  
 \_m;  
 \_fl = false;  
 constructor(arr, x\_n) {  
 this.arr = arr;  
 this.x\_n = x\_n;  
 this.pivotElement = null;  
  
  
 this.\_n = this.arr.length - 2;  
 this.\_m = this.arr[0].length - 2;  
 }  
  
 calcDelta() {  
 for (let j = 2; j < this.\_m + 2; j++) {  
 if (j !== 2)  
 this.arr[this.\_n + 1][j] = this.arr[0][j].mul(-1);  
 else  
 this.arr[this.\_n + 1][j] = new GreatNumber(0, 0);  
  
 for (let i = 2; i < this.\_n + 1; i++)  
 this.arr[this.\_n + 1][j] = this.arr[this.\_n + 1][j].add(this.arr[i][0].mul( this.arr[i][j]));  
 }  
 }  
  
 calcPivotElement() {  
  
 let min\_j, min\_delta, min\_i = null;  
  
 min\_delta = *Math*.min(...this.arr[this.\_n + 1].slice(3));  
 if (min\_delta >= 0)  
 return;  
  
 for (min\_j = 2; min\_j < this.\_m + 2; min\_j++) {  
 if (+this.arr[this.\_n + 1][min\_j] === min\_delta) {  
  
 for (let i = 2; i < this.\_n + 1; i++)  
 if (this.arr[i][min\_j] > 0) {  
 if (min\_i === null)  
 min\_i = i;  
 else  
 if (this.arr[i][2].div(this.arr[i][min\_j]) < this.arr[min\_i][2].div(this.arr[min\_i][min\_j]))  
 min\_i = i;  
  
 }  
  
 }  
  
 if (min\_i !== null) {  
 break;  
 }  
 if (min\_j === (this.\_m + 1)) {  
 this.\_fl = true;  
 return;  
 }  
  
 }  
  
 this.pivotElement = {  
 i: min\_i,  
 j: min\_j  
 };  
 }  
  
 checkIsNextPossible() {  
 return this.pivotElement !== null;  
 }  
  
 checkResult() {  
 if (this.pivotElement !== null)  
 return "Задача має продовження";  
 if (this.\_fl)  
 return "Всі Δ ≥ 0 і в напрямному стовпці всі значення ≤ 0, тому задача рішення не має";  
 for (let i = 2; i < this.\_n + 1; i++)  
 if (this.arr[i][2] < 0)  
 return "Всі Δ ≥ 0 і в стовпці вільних коефіцієнтів є відємне значення, тому задача рішення не має";  
 else if (this.arr[i][0].m != 0)  
 return "Всі Δ ≥ 0 і в базисі все ще є штучна змінна, тому задача рішення не має"  
 return true;  
 }  
  
 next() {  
 if (!this.checkIsNextPossible())  
 return null;  
 return new SimplexArr(this.\_calcNextArr(), this.x\_n);  
 }  
  
 \_calcNextArr() {  
  
  
 let new\_arr = AdditionalFunctions.*copy\_arr*(this.arr);  
  
 new\_arr[this.pivotElement.i][0] = this.arr[0][this.pivotElement.j];  
 new\_arr[this.pivotElement.i][1] = this.arr[1][this.pivotElement.j];  
  
 for (let i = 2; i < this.\_n + 1; i++) {  
 for (let j = 2; j < this.\_m + 2; j++)  
 if (i === this.pivotElement.i)  
 new\_arr[i][j] = this.arr[this.pivotElement.i][j].div(this.arr[this.pivotElement.i][this.pivotElement.j]);  
 else  
 new\_arr[i][j] = this.arr[i][j].sub(  
 (this.arr[this.pivotElement.i][j].mul(this.arr[i][this.pivotElement.j]).div(  
 this.arr[this.pivotElement.i][this.pivotElement.j])));  
 new\_arr[i][this.pivotElement.j] = new Frac(0, 1);  
 }  
 new\_arr[this.pivotElement.i][this.pivotElement.j] = new Frac(1, 1);  
  
 if (this.arr[this.pivotElement.i][0].m != 0) {  
 //console.log(this.arr[this.pivotElement.i][0].m)  
 for (let j = 3; j < this.\_m + 2; j++) {  
 if (this.arr[this.pivotElement.i][1] === this.arr[1][j]) {  
 new\_arr = AdditionalFunctions.*delete\_row*(new\_arr, j);  
 break;  
 }  
 }  
 }  
  
 return new\_arr;  
 }  
}  
  
class SimplexVisual {  
 \_fontSize;  
 static *table\_i* = 1;  
 constructor(table) {  
 //console.log(table);  
 this.table = table;  
 this.\_fontSize = 20;  
 }  
  
 tableToHTML() {  
 if (this.table.theta\_arr !== undefined)  
 return SimplexVisual.*#tableToStr*(this.table.arr, this.table.pivotElement, this.table.theta\_arr);  
 else  
 return SimplexVisual.*#tableToStr*(this.table.arr, this.table.pivotElement, null)  
 }  
  
 static *#tableToStr*(arr, pivotElement, theta) {  
 *console*.log(pivotElement);  
 arr = AdditionalFunctions.*copy\_arr*(arr);  
 arr[0][2] = "-";  
 if (!!theta)  
 arr.push(theta);  
  
 let table = "";  
  
 for (let j = 2; j < arr[0].length; j++)  
 arr[1][j] = SimplexVisual.*#getSmall*(`A${arr[1][j].substring(1)}`);  
 for (let i = 2; i < arr.length - 1; i++)  
 arr[i][1] = SimplexVisual.*#getSmall*(arr[i][1]);  
  
 for (let i = 0; i < arr.length; i++) {  
 let tr = "<tr>";  
 if (!!pivotElement && i === pivotElement.i)  
 tr = "<tr class='mark'>";  
 if (!!theta && i === arr.length - 1)  
 tr = "<tr class='theta-row'>"  
 for (let j = 0; j < arr[0].length; j++) {  
 if (!!pivotElement && j === pivotElement.j)  
 tr += `<td class="mark">${arr[i][j]}</td>`;  
 else  
 tr += `<td>${arr[i][j]}</td>`;  
 }  
  
 table += tr + "</tr>";  
 }  
  
 let s = `<div class="right">Таблиця ${SimplexVisual.*table\_i*++}</div>`;  
 return `<div class="simplex-wrapper">${s + `<table class='simplex\_table'> ${table} </table>`}</div>`;  
 }  
  
 resultsToHTML() {  
 //console.log(this);  
 let func\_res = "", func\_res\_int = new Frac(0, 1);  
  
 let res = this.table.checkResult();  
 if (res !== true) {  
 return `<div class="res"><h4 style='color: red'>${res}</h4></div>`;  
 }  
  
 let result = "<div>Всі дельта більше нуля, отримано оптимальний розв’язок задачі:</div>"  
  
  
 let x\_results = "";  
 for (let j = 3; j < this.table.x\_n + 3; j++) {  
 //console.log(this.table.arr[1][j]);  
 let fl = false;  
 for (let i = 2; i < this.table.arr.length - 1; i++) {  
 //console.log(` ${this.table.arr[i][1]}`);  
 if (this.table.arr[i][1] === this.table.arr[1][j]) {  
 let t = this.table.arr[i][2];  
 x\_results += `, ${SimplexVisual.*#getSmall*(this.table.arr[i][1])} = ${t}`;  
 func\_res += ` + ${this.table.arr[i][0]} \* ${t}`;  
 //func\_res\_int = func\_res\_int.add(this.table.arr[i][0].mul(t));  
 fl = true;  
 }  
 }  
  
 if (j - 3 < this.table.x\_n && !fl) {  
 x\_results += `, ${SimplexVisual.*#getSmall*(this.table.arr[1][j])} = 0`;  
 func\_res += ` + ${this.table.arr[0][j]} \* 0`;  
 }  
  
  
 }  
 result += `<div class='x\_results'>${x\_results.substring(2)}</div>`;  
  
 result += `<div class="func\_res">F<sub>max</sub> = ${func\_res.substr(3)} = ${this.table.arr[this.table.arr.length - 1][2]}</div>`;  
  
 return `<div class="res">${result}</div>`;  
 }  
  
 static *#getSmall*(s) {  
 return s[0] + "<sub>" + s.substr(1) + "</sub>";  
 }  
}  
  
function refactorArr(arr) {  
 let new\_arr = AdditionalFunctions.*copy\_arr*(arr);  
 for (let i = 0; i < arr.length; i++)  
 for (let j = 0; j < arr[0].length; j++)  
 if (typeof arr[i][j] === typeof 1) {  
 if (i !== 0 && j !== 0 && i !== (arr.length - 1))  
 new\_arr[i][j] = new Frac(arr[i][j], 1);  
 else  
 new\_arr[i][j] = new GreatNumber(0, arr[i][j]);  
 }  
  
 return new\_arr;  
}

**style.css:**

body {  
 margin: 0;  
 height: 100vh;  
 width: 100vw;  
 overflow: hidden;  
 background-color: #fafafa;  
}  
  
\* {  
 box-sizing: border-box;  
}  
  
.main\_wrapper {  
 z-index: 1;  
 top: 50px;  
 height: 100vh;  
 overflow: scroll;  
}  
  
main {  
 display: flex;  
 flex-direction: column;  
 justify-content: center;  
 align-items: center;  
 padding: 0 25px;  
}  
  
#task {  
 height: calc(100vh - 25px);  
 max-width: 800px;  
 display: flex;  
 row-gap: 25px;  
 flex-direction: column;  
 justify-content: center;  
 align-items: center;  
 text-align: justify;  
 font-size: 20px;  
}  
  
#task\_table table, #task\_table td {  
 border: 1px solid black;  
 border-collapse: collapse;  
}  
  
#task td {  
 padding: 5px 10px;  
}  
  
#task\_table td:nth-child(2) input {  
 width: 75px;  
}  
  
#task input {  
 text-align: left;  
 width: 50px;  
}  
  
#fields div{  
 display: inline-block;  
}  
  
#fields {  
 display: flex;  
 width: 100%;  
 justify-content: space-between;  
}  
  
#task button {  
 font-size: 20px;  
 margin-top: 25px;  
 padding: 10px 25px;  
}  
  
.restrictions {  
 display: flex;  
 position: relative;  
}  
  
.equations {  
 margin-left: 15px;  
 display: flex;  
 flex-direction: column;  
}  
  
#error {  
 color: red;  
 display: none;  
  
}  
  
#error.visible {  
 display: block;  
 position: absolute;  
 bottom: 0;  
}  
  
#result {  
 display: none;  
  
}  
  
#result.visible, .flex-center {  
 display: flex;  
 flex-direction: column;  
 justify-content: center;  
 align-items: center;  
}  
  
.simplex\_table td {  
 border: 1px solid black;  
 text-align: center;  
}  
  
.simplex\_table {  
 border-collapse: collapse;  
}  
  
.simplex\_table td:nth-child(1), .theta-row td {  
 border: none;  
}  
.sub\_math\_model {  
 margin: 5px 0 25px 0;  
}  
  
#solve {  
 row-gap: 20px;  
}  
  
#result h3 {  
 font-size: 25px;  
}  
  
#result {  
 font-size: 20px;  
}  
  
.simplex\_table td {  
 padding: 5px;  
 min-width: 50px;  
}  
  
.simplex\_table {  
 margin-bottom: 50px;  
}  
  
.right {  
 width: 100%;  
 text-align: right;  
 margin-bottom: -20px;  
}  
  
#final\_res, .res {  
 row-gap: 10px;  
 margin: 50px 0;  
}  
  
.res {  
 display: flex;  
 flex-direction: column;  
 align-items: center;  
 row-gap: 10px;  
}  
  
.fraction {  
 display: inline-block;  
 text-align: center;  
}  
  
.fraction div {  
 height: 1px;  
 background-color: black;  
}  
  
.simplex\_table .mark {  
 background-color: yellow;  
}  
  
.simplex\_table tr.mark .mark {  
 background-color: red;  
 font-weight: bold;  
}  
  
.simplex-wrapper {  
 display: flex;  
 flex-direction: column;  
 row-gap: 20px;  
}