Чернівецький національний університет імені Юрія Федьковича

Факультет математики та інформатики

Кафедра прикладної математики та інформаційних технологій

КУРСОВА РОБОТА

на тему: **Розробка веб-додатку для розв'язування транспортних задач**

Студентки 4 курсу 422 групи

галузь знань 11 – математика та статистика,

спеціальність 6.113 – прикладна математика,

Лазарюк Діани Іванівни
(прізвище, ім'я, по-батькові)

Керівник
ас. Юрійчук Анастасія Олександрівна
(науковий ступінь, вчене звання, прізвище, ім'я, по-батькові)

До захисту допущено:

Керівник_____ _____
Підпис Прізвище та ініціали

"_____" _____ 2018 р.

Чернівці–2018

АНОТАЦІЯ

Робота присвячена розробці програмного продукту для швидкого та якісного розв’язування транспортних задач за допомогою фреймвока Vue. Додаток демонструє покрокове розв'язування задачі динамічним виведенням даних.

Робота складається з трьох розділів. У першому розглядається постановка транспортної задачі, побудова початкового опорного плану трьома методами та метод потенціалів для її подальшого розв'язування. У другому розділі наведена коротка характеристика використаних програмних засобів. Третій розділ присвячений опису створеного веб-додатку.

ЗМІСТ

ВСТУП

Робота присвячена розв'язуванню транспортних задач за допомогою JavaScript та фреймворка Vue.

Транспортна задача — задача про оптимальний план перевезення продукту із пунктів відправлення до пунктів споживання. Розробка і використання оптимальних схем вантажних потоків дозволяють знизити витрати на перевезення.

Нижче будуть наведені теоретичні і практичні відомості розв’язування задачі, а також приклад робочої програми написаної засобами JavaScript, Vue, HTML та CSS. У програмі користувачу потрібно задати вхідні дані та вибрати метод, яким буде розв'язано транспортну задачу, тоді програма розв’яже поставлену задачу.

Програма має декілька варіантів побудови опорного плану: мінімального елемента, північно-західного кута, подвійної переваги. Для розуміння студентів один із варіантів може бути простіший чи складніший за інший, та кожний потребує уваги.

РОЗДІЛ 1. Транспортна задача

1.1 Постановка задачі

У декількох m пунктах A_i, $i \varepsilon\{1, ..., m\}$ виробляється однорідний продукт призначений для використання в n пунктах B_j, $j \varepsilon \{1, ..., n\}$.

Транспортні витрати на перевезення одиниці вантажу з A_i, $i \varepsilon\{1, ..., m\}$, в B_j, $j \varepsilon \{1, ..., n\}$ становлять відповідно C_{ij} грошових одиниць. Задача полягає в побудові такого плану перевезень, при якому запити всіх пунктів-споживачів було б задоволено, весь продукт із пунктів-виробників був би вивезений і при цьому сумарні транспортні витрати були б мінімальними.

Нехай $x_{ij} \geq 0$, $i \varepsilon\{1, ..., m\}$, $j \varepsilon \{1, ..., n\}$, — кількість одиниць вантажу, який перевозить з пункту A_i, $i \varepsilon\{1, ..., m\}$, у пункт B_j, $j \varepsilon \{1, ..., n\}$.

Очевидно, що $x_{ij} \geq 0$, $i \varepsilon\{1, ..., m\}$, $j \varepsilon \{1, ..., n\}$. Вартість такого перевезення $\sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$. Загальний обсяг продуктів, що вивозиться від i -го постачальника $\sum_{j=1}^{n} c_{ij} x_{ij}$, $i \varepsilon\{1, ..., m\}$, не повинен перевищувати запас продукції цього постачальника. Обсяг продукції, яка надходить j -му споживачу $\sum_{i=1}^{m} c_{ij} x_{ij}$, $j \varepsilon\{1, ..., n\}$, не повинен бути меншим від потреб цього споживача. Отримали задачу

$$f = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \rightarrow min \qquad (1.1)$$

$$\begin{cases} \sum_{j=1}^{n} x_{ij} \leq a_i, \quad i \in \{1, \ldots, m\}, \\ \sum_{i=1}^{m} x_{ij} \geq b_j, \quad j \in \{1, \ldots, n\}; \end{cases} \qquad (1.2)$$

$$x_{ij} \geq 0,\ i \in \{1, ..., m\},\ j \in \{1, ..., n\}. \quad (1.3)$$

Якщо

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j, \qquad (1.4)$$

то транспорта задача називається збалансованою (закритою), а у випадку невиконання умови (1.4) – незбалансованою (відкритою). Умова (1.4) є необхідною і достатньою для розв'язності задачі (1.1)–(1.3).

Збалансована транспортна задача полягає в знаходженні такого плану $X = (x_{ij})_{i\ j}^{m\ n}$, для якого виконується система обмежень

$$\begin{cases} x_{11} + x_{12} + \ldots + x_{1n} = a_1, \\ x_{21} + x_{22} + \ldots + x_{2n} = a_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ x_{m1} + x_{m2} + \ldots + x_{mn} = a_m; \\ x_{11} + x_{21} + \ldots + x_{m1} = b_1, \\ x_{12} + x_{22} + \ldots + x_{m2} = b_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ x_{1n} + x_{2n} + \ldots + x_{mn} = b_n; \end{cases} \qquad (1.5)$$

за умов (1.2), а цільова функція f з (1.1) досягає найменшого значення.

1.2 Побудова початкового опорного плану

Розглянемо збалансовану транспортну задачу (1.1) – (1.3). Транспорна задача є задачею лінійного програмування, тому опорним планом цієї задачі є її базисний допустимий план. Транспортну задачу та її розв'язування зручно подавати за допомогою транспортної таблиці (матриці) вигляду

Пункти відправлення	Пункти споживання				Запаси, од. продукції
	B_1	B_2	...	B_n	
A_1	$x_{11}^{c_{11}}$	$x_{12}^{c_{12}}$	...	$x_{1n}^{c_{1n}}$	a_1
...	...	...	...	...	...
A_m	$x_{m1}^{c_{m1}}$	$x_{m2}^{c_{m2}}$	...	$x_{mn}^{c_{mn}}$	a_m
Потреби, од. продукцій	b_1	b_2	...	b_n	

Табл.1. Представлення транспортної задачі у вигляді таблиці.

Вектори–стовпці матриці коефіцієнтів системи обмежень (1.5)

$$P_{ij} = (\overbrace{0\ldots0\,1}^{i}\,\underbrace{0\ldots0\,1}_{m+j}\,0\ldots0)^T,$$

$$\underbrace{\qquad\qquad\qquad\qquad}_{m+n}$$

$$i \in \{1,\ldots,m\}, \quad j \in \{1,\ldots,n\},$$

називають векторами комунікацій. Ранг матриці коефіцієнтів системи обмежень (1.5) транспортої задачі дорівнює m + n − 1. Тому довільний базис системи векторів комунікацій транспортої задачі містить m + n − 1 вектор.

Перевезення x_{ij} плану X називатимемо основними (відповідні клітинки основними). Тоді план X є опорним планом транспортої задачі, якщо вектори комунікацій, що відповідають його основним перевезенням, утворюють лінійно незалежну систему.

Існує ряд методів побудови початкових опорних планів транспортної задачі. При використанні одних методів початковий опорний план одер-жується дуже швидко, однак, взагалі кажучи, такий опорний план може бути далеким від оптимального. Пошук початкового опорного плану методом інших групи передбачає проведення значних додаткових обчислень, однак початковий опорний план, який при цьому одержується, є близьким до оптимального. Суть методів побудови опорних планів полягає в тому, що опор- ний план знаходять послідовно за m + n − 1 кроків, на кожному з яких у транспортній таблиці заповнюють одну клітинку. Заповнення однієї клітинки забезпечує повністю потреби одного з споживачів, або вивезення усього

вантажу від деякого постачальника. У першому випадку тимчасово виключають з розгляду стовпчик, який відповідає розглядуваному споживачу (в другому випадку – рядок, який відповідає розглядуваному постачальнику), змінивши запаси вантажу одного з пунктів постачання (споживання), за рахунок якого була забезпечена потреба пункту споживання (вичерпані запаси пункту постачання) на попередньому кроці. Після здійснення m + n − 2 вищеописаних кроків, отримується задача з одним пунктом постачання та одним пунктом призначення, для яких запаси, що залишились, дорівнюють потребам споживача. При цьому залишається вільною лише одна клітинка, заповнивши яку на m+n−1 кроці одержуємо опорний план. Зазначимо, якщо на деякому кроці потреби деякого пункту постачання дорівнюють запасам пункту споживання, то з розгляду виключають і рядок, і стовпчик, записуючи нуль у наступну клітинку розглядувагого рядка і стовпця. Такий спосіб гарантує заповнення m+n−1 клітинки, при цьому їх множина буде ациклічною, оскільки вона будується з використанням методу викреслювання. Отже, побудований таким способом план перевезень буде допустимим і ациклічним, тобто опорним.

1.2.1 Метод північно-західного кута

На кожному кроці розглядають перший, з тих що залишилися, пункт відправлення і перший, з тих що залишилися, пункт призначення, тобто верхня ліва (північно-західна) невикреслена клітинка транспортної таблиці. На першому кроці заповнюється клітинка (1;1) і покладають $x_{1\,1} = \min\{a_1\,; b_1\}$. Можливі такі три випадки:

1) якщо $a_1 < b_1$, то запаси пункту a_1 вичерпані $a_1 = a_1 - x_{1\,1} = 0$ і перший рядок таблиці виключаємо з подальшого розгляду, а потреби пункту $b_1 = b_1 - x_{1\,1}$;

2) якщо $a_1 > b_1$, то запаси пункту $a_1 = a_1 - x_{1\,1} = 0$, а потреби пункту B1 задоволені повністю $b_1 = b_1 - x_{1\,1}$ і перший стовпчик таблиці виключаємо з подальшого розгляду;

3) якщо $a_1 = b_1$, то запаси пункту a_1 і потреби пункту b_1 задоволені повністю ($b_1 = a_1 = 0$) і перший рядок та перший стовпчик таблиці виключаємо з подальшого розгляду. На другому кроці аналогічно розглядаємо невикреслену північно-західну клітинка (у випадку 1 це клітинка (2;1), у випадку 2 – (1;2), у випадку 3 – (2;2)). Заповнення клітинок таблиці і закінчують клітинкою (mn), тобто йдуть немовби по діагоналі.

Цей метод не враховує вартості і тому може виявитись велике відхилення початкового опорного плану від оптимального. Цей метод простіший від інших нижче розглянених методів побудови опорних планів у випадку великих розмірів таблиці.

1.2.2 Метод мінімального елемента

Спочатку по всій транспортній таблиці ведеться пошук клітини з найменшою вартістю. Потім змінній в цій клітині присвоюється найбільше значення, що допускається обмеженнями A_i та B_j (якщо таких змінних декілька, вибір довільний.) Далі викреслюється відповідний стовпець або рядок, і відповідним чином коректуються значення A_i та B_j. Якщо одночасно виконуються обмеження і A_i і B_j, викреслюється або рядок, або стовпець (точно так само, як у методі північно-західного кута). Тоді проглядаються невикреслені клітини, і вибирається нова клітина з мінімальною вартістю. Описаний процес триває до тих пір, поки не залишиться ні одного невикресленого рядка або стовпця.

1.2.3 Метод подвійної переваги

Якщо транспортна таблиця велика, то перебрати всі клітинки важко. У цьому випадку використовують метод подвійної переваги, суть якого полягає в наступному. У кожному стовпчику помічаємо значком $\vee$ клітинку з найменшою ціною. Потім це саме робимо в кожному рядку. Як результат матимемо у деяких клітинках помітки $\vee\vee$. Цим клітинкам відповідає мінімальна ціна як по стовпчику, так і по рядку. В ці клітинки записуємо максимально можливі обсяги, кожного разу виключаючи з розгляду відповідні стовпчики або рядки. Потім розподіляємо обсяги перевезень по клітинках, які помічені значком $\vee$. У тій частині таблиці, що залишилася, вантаж розподіляємо за методом мінімального елемента.

1.3 Знаходження розв'язку транспортної задачі. Метод потенціалів

Побудований за допомогою одного з описаних вище методів опорний план можна довести до оптимального, скориставшись симплексним методом. Оскільки це громіздко, то використовуватимемо простіші методи, одним з яких є метод потенціалів.

Доведено, що якщо для деякого невиродженого опорного плану $X = (x_{ij})_i^m{}_j^n$ транспортної задачі існують такі числа u_i, $i \in \{1, \dots, m\}$, та v_j, $j \in \{1, \dots, n\}$, що

$$v_j - u_i = c_{ij} \text{ при } x_{ij} > 0, \qquad (3.1)$$

$$v_j - u_i \le c_{ij} \text{ при } x_{ij} = 0, \qquad (3.2)$$

для всіх $i \in \{1, \dots, m\}$, $j \in \{1, \dots, n\}$, то цей план є оптимальним. Числа u_i та v_j називаються потенціалами відповідно пунктів відправлення A_i і пунктів

призначення B_j. Зазначимо, що умова (3.1) відповідає базисним клітинкам транспортної таблиці, а (3.2) – небазисним. Оскільки, опорний план невироджений, то система (3.1) складається з m + n − 1 рівнянь з m + n невідомими, тому має не єдиний розв'язок. Якщо B_X – множина базисних векторів P_{ij} деякого невиродженого опорного плану X, E_x – множина відповідних їм базисних клітинок транспортної таблиці, (i_k, j_l) – деяка небазисна клітинка, тоді $P_{i_k j_l}$ можна розкласти за базисом B_X. Іншими словами приєднати клітинки (i_k, j_l) до множини базисних клітинок E_x, що породжує цикл, при цьому клітинки циклу С діляться на дві множини C^+ і C^- (відповідно коефіцієнти розкладу $P_{i_k j_l}$ рівні 1 та -1).

Якщо (i_r, j_s) – клітинка, якій відповідає

$$\theta := \min\{x_{ij}\}, \text{ де min береться з значень } (i, j) \varepsilon C^- \qquad (3.3)$$

то перейшовши до нового опорного плану $X' = (x'_{ij})_{i\ j}^{m\ n}$, де

$$x'_{ij} = \begin{cases} x_{ij} - \theta, & (i,j) \in C^-, \\ x_{ij} + \theta, & (i,j) \in C^+, \\ x_{ij}, & (i,j) \notin C, \end{cases} \qquad (3.4)$$

виводимо вектор $P_{i_r j_s}$ з базису B_X шляхом розриву циклу С на клітинці (i_r, j_s), ввівши при цьому в базис вектор $P_{i_k j_l}$. Сформулюємо алгоритм методу потенціалів розв'язування транспортої задачі.

1. Одним з відомих методів знайти початковий опорний план X.

2. Обчислити потенціали рядків u_i, $i \in \{1, \ldots, m\}$, та стовпчиків v_j, $j \in \{1, \ldots, n\}$, транспортної таблиці, поклавши один з потенціалів рівним нулю та розв'язавши на множині E_X базисних клітинок плану систему

$$v_j - u_i = c_{ij}, (i, j) \in E_{X'}. \qquad (3.5)$$

3. Обчислити для небазисних клітинок оцінки транспортної таблиці

$$\Delta_{ij} := v_j - u_i \le c_{ij}\ ,\ (i, j) \in E_X \qquad (3.8)$$

та перевірити умову $\Delta_{ij} \le 0$, (i, j) $\in$ E_X . Якщо ця умова виконується, то план є оптимальним.

4. Визначити клітинку (i_k, j_l) з найбільшою додатною оцінкою $\Delta_{i_k j_l}$ приєднати її до множини E_X базисних клітинок плану X' та знайти єдиний цикл С, який породжує таке приєднання.

5. Розбити цикл С на C^+ і C^- (обхидячи цикл першою відносячи до C^+ клітинку (i_k, j_l), наступну за нею по циклу клітинку – до C^- наступну за цією клітинкою – знову до C^+, далі до C^- і так, поки не повернеться до клітинки (i_k, j_l), з якої почали. Знайти клітинку (i_r,j_s), яка має найменше перевезення серед клітинок C^- згідно з (3.5) (якщо таких клітинок декілька, то вибирають будь-яку з них) та згідно з (3.4) перейти до нового опорного плану.

6. Повернутись до другого пункту.

РОЗДІЛ 2. Програмні засоби, використані для створення веб-додатку

Для реалізацій функцій веб-додатку, які розв'язують транспортну задачу використовується JavaScript, а для динамічного виведення даних використовується фреймворк Vue. Для стилізацій використовується CSS і HTML.

2.1 JavaScript

JavaScript — динамічна, об'єктно-орієнтована прототипна мова програмування. Найчастіше використовується для створення сценаріїв веб-сторінок, що надає можливість на стороні клієнта (пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки.

JavaScript класифікують як прототипну (підмножина об'єктно-орієнтованої), скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам'яттю, прототипне наслідування, функції як об'єкти першого класу.

Мова JavaScript використовується для:

- написання сценаріїв веб-сторінок для надання їм інтерактивності;
- створення односторінкових веб-додатків (React, Angular, Vue.js);
- програмування на стороні сервера (Node.js);
- стаціонарних застосунків (Electron, NW.js);

- мобільних застосунків (React Native, Cordova);
- сценаріїв в прикладному ПЗ (наприклад, в програмах зі складу Adobe Creative Suite чи Apache JMeter);

Незважаючи на схожість назви, мови Java та JavaScript є двома різними мовами, що мають відмінну семантику, хоча й мають схожі риси в стандартних бібліотеках та правилах іменування. Синтаксис обох мов отриманний «у спадок» від мови C, але семантика та дизайн JavaScript є результатом впливу мов Self та Scheme.

2.2 Фреймворк Vue

Vue — це прогресивний фреймворк для створення користувацьких інтерфейсів. На відміну від фреймворків-монолітів, Vue створений придатним для поступового впровадження. Його ядро в першу чергу вирішує завдання рівня уявлення (view), що спрощує інтеграцію з іншими бібліотеками та існуючими проектами. З іншого боку, Vue повністю підходить і для створення складних односторінкових додатків, якщо використовувати його спільно з сучасними інструментами та додатковими бібліотеками.

Підключення фреймворка:

```
<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>,
```

або через webpack.

В основі Vue.js - це система, яка дозволяє декларативно перетворювати дані в DOM за допомогою простого синтаксису шаблонів:

html:

```
<div id = "app">
  {{message}}
```

```
</div>
```

Vue:

```
var app = new Vue ({
 el: '#app' ,
 data: {
   message: 'Hello Vue!'
 }
})
```

Директиви v-вказуються на те, що вони є особливими атрибутами, наданими Vue, і, як ви вже здогадалися, вони застосовують особливу реактивну поведінку до видозміненого DOM.

html:

```
 <div id = "app-3" > <span v-if = "saw"> Тепер ви бачите мене </span> </div>
```

Vue:

```
var app3 = new Vue ({
    el: '#app-3' ,
    data: {
       seen: true
    }
 })
```

2.3. Дизайн UI веб-додатку за допомогою HTML, CSS

HTML (англ. HyperText Markup Language – мова розмітки гіпертекстових документів) стандартна мова розмітки веб-сторінок в Інтернеті. Більшість веб-сторінок створюються за допомогою мови HTML (або XHTML). Документ HTML оброблюється браузером та відтворюється на екрані у звичному для людини вигляді.

Попри те, що HTML – штучна комп'ютерна мова, вона не є мовою програмування. HTML разом із каскадними таблицями стилів та вбудованими скриптами — це три основні технології побудови веб-сторінок.

HTML впроваджує засоби для:

- створення структурованого документа шляхом позначення структурного складу тексту: заголовки, абзаци, списки, таблиці, цитати та інше;
- отримання інформації із Всесвітньої мережі через гіперпосилання;
- створення інтерактивних форм;
- включення зображень, звуку, відео, та інших об'єктів до тексту.
- Документ HTML 4.01 складається з трьох частин:
    - Декларація типу документа (англ. Document type declaration, Doctype), на початку документа, в якій визначається тип документа (DTD).
    - Шапка документа (знаходиться в межах елемента head), в якій записано загальні технічні відомості або додаткова інформація про документ, яка не відтворюється безпосередньо в браузері;
    - Тіло документа (може знаходитися в елементах body або frameset), в якому міститься основна інформація документа.

Каскадні таблиці стилів (англ. Cascading Style Sheets або скорочено CSS) — спеціальна мова, що використовується для опису сторінок, написаних мовами розмітки даних.

РОЗДІЛ 3. Опис проекту

3.1. Постановка задачі

Робота присвячена розв'язуванню транспортних задач. Веб-додаток дає можливість ввести вхідні дані транспортної задачі і вибрати метод побудови опорного плану та проглянути покроковий розв’язок задачі з описом до виконаних дії. Таким чином можна перевірити та розібратися як розв'язуються транспортні задач.

3.2. Введення даних

Форма (рис.3.1) містить поля для введення кількості виробників, кількості споживачів та поле вибору методу побудови опорного плану. Динамічно будують таблицю для введення таких даних:

- кількість продукції, яку виготовляє A_i;
- кількість продукції, яку потребує B_j споживач;
- ціну перевезення за одиницю продукції.

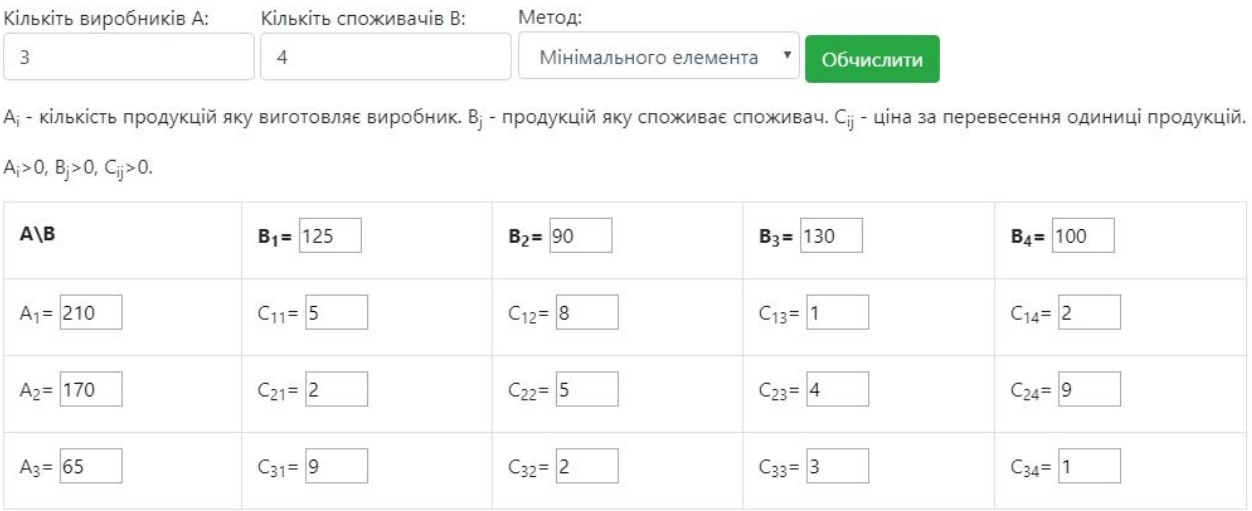

Рис.3.1. Форма для введення даних.

Після введення даних і натиску кнопки обчислення відбувається перевірка даних. Якщо дані введені неправильно, то додаток повідомить про це користувача і не буде розв'язувати задачу (рис.3.2).

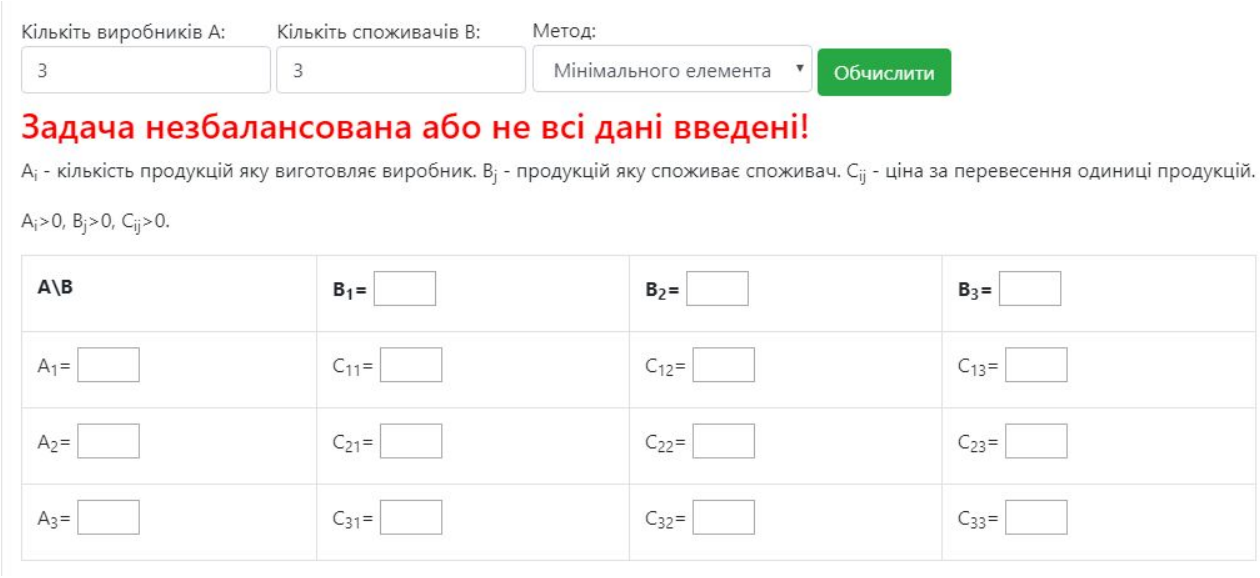

Рис.3.2. Виведення помилки.

При правильних даних програма виводить початкові дані та розв'язки. Дані виводяться динамічно за допомогою vue.

```
<div v-for="state in data"> //за допомогою директиви циклу ми проходимося по всіх масивах
//які зберігають дані задачі на кожному кроці
  <h2 v-html="state.title">{{state.title}}</h2>//переводимо звичайний текс в html
   <h6 v-html="state.teor">{{state.teor}}</h6>//в дів виводиться теоретична частина
 <div v-html="state.description">{{state.description}}</div>//в дів виводиться опис прикладу
    <table class="table table-bordered">
      <tr>
        <th>A \ B</th>
        <th v-for="(item, index) in state.b">
          <label>
            B<sub>{{index + 1}}</sub>= {{item}}//виводяться значення Bj
          </label>
        </th>
        <th v-if="state.showVU">U</th>//як що ми рахували V і U то ми до таблиці додаємо ще
один стовпчик
```

```
            </tr>
            <tr v-for="(row, i) in state.data">
              <td>
                <label>
                  A<sub>{{i + 1}}</sub>= {{state.a[i]}}//виводим всі Ai
                </label>
              </td>
                  <td class="cell" v-for="(item, j) in row" v-bind:class="{ // за допомогою директиви
//задаємо класи залежності від даних
                  baz:(item.x !== null),
                  onecheck:item.minV,
                  twocheck:item.minG
                }">
                  <div>
                    C<sub>{{i + 1}}{{j + 1}}</sub>={{item.c}}//виводимо ціну перевезення
                        <div v-if="item.x !== null">X<sub>{{i + 1}}{{j + 1}}</sub>={{item.x}}</div>
//виводимо значення X як що воно не порожнє
                    <div v-if="item.d !== null">D<sub>{{i + 1}}{{j + 1}}</sub>={{item.d}}</div>
          //виводимо значення  D як що воно не порожнє
                        </div>
              </td>
              <td v-if="state.showVU">
                <label>
                  U<sub>{{i + 1}}</sub>= {{state.u[i]}}//виводимо U
                </label>
              </td>
            </tr>
            <tr v-if="state.showVU">
              <th>V</th>
              <td v-for="(item, index) in state.v">
                <label>
                  V<sub>{{index + 1}}</sub>= {{item}}//виводимо V
                </label>
              </td>
            </tr>
          </table>
        </div>
```

3.3. Опис функцій веб-додатку

3.3.1. Функція побудови опорного плану

Є три методи для побудови опорного плану: метод мінімального елемента (рис.3.3) , метод північно-західного кута (рис.3.4) та метод подвійної переваги (рис.3.5).

Будуємо опорний план методом мінімального елемента

Крок 1

Для побудови опорного плану потрібно знайти найменшу ціну (C_{ij}) з незаповнених клітинок, а також $A_i > 0$ і $B_j > 0$. Тоді в $X_{ij}=\min\{A_i; B_j\}$ і $A_i=A_i-X_{ij}$, $B_j=B_j-X_{ij}$.

Найменша ціна з клітинок, які не містить X - це $C_{1\,3}=1$ в клітинці [1;3]. Заповнюємо $X_{1\,3}= \min\{210;130\}$. А також змінюємо значення $A_1=210-130$ і $B_3=130-130$.

A \ B	**B_1= 125**	**B_2= 90**	**B_3= 0**	**B_4= 100**
A_1= 80	C_{11}=5	C_{12}=8	C_{13}=1 X_{13}=130	C_{14}=2
A_2= 170	C_{21}=2	C_{22}=5	C_{23}=4	C_{24}=9
A_3= 65	C_{31}=9	C_{32}=2	C_{33}=3	C_{34}=1

Рис.3.3. Побудова опорного плану методом мінімальних елементів.

Будуємо опорний план методом північно-захіного куда

Крок 1

На першому кроці заповнюємо клітинку [1;1] $X_{ij}=\min\{A_i;B_j\}$, $B_1=B_1-X_{11}$, $A_1=A_1-X_{11}$. Далі є 3 варіанти :
Якщо $A_i < B_j$, то в $X_{ij}=A_i$, $B_j=B_j-A_i$, $A_i=0$, та стовчик j не розглядаємо в подальшому;
Якщо $A_i > B_j$, то в $X_{ij}=B_i$, $A_j=A_j-B_i$, $B_i=0$ та рядок і не розглядаємо в подальшому;
Якщо $A_i = B_j$, то в $X_{ij}=B_i$, $A_j=0$, $B_i=0$ та рядок і та стовпчик j не розглядаємо в подальшому;

Покладемо в клітинку [1;1] $X_{11}=\min\{210;125\}=125$, $B_1=125-125=0$, $A_1=210-125=85$

a\b	**B_1= 0**	**B_2= 90**	**B_3= 130**	**B_4= 100**
A_1= 85	C_{11}=5 X_{11}=125	C_{12}=8	C_{13}=1	C_{14}=2
A_2= 170	C_{21}=2	C_{22}=5	C_{23}=4	C_{24}=9
A_3= 65	C_{31}=9	C_{32}=2	C_{33}=3	C_{34}=1

Рис.3.4. Побудова опорного плану методом північно-західного кута.

Будуємо опорний план методом подвійної переваги

Крок1

В кожному рядочку відмічаємо галочкою клітинку з найменшою ціною,і так ще в совчиках

a\b	B_1= 125	B_2= 90	B_3= 130	B_4= 100
A_1= 210	C_{11}=5	C_{12}=8	C_{13}=1 ✓✓	C_{14}=2
A_2= 170	C_{21}=2 ✓✓	C_{22}=5	C_{23}=4	C_{24}=9
A_3= 65	C_{31}=9	C_{32}=2 ✓	C_{33}=3	C_{34}=1 ✓✓

Рис.3.5. Побудова опорного плану методом подвійної переваги.

В залежності від вибраного методу буде виведена відповідна теорія і побудовано опорний план з поясненнями.

3.3.2. Функція для обчислення потенціалів

Наступним кроком в розв'язку задачі є знаходження потенціалів (рис.3.6)

Обчислюємо значення потенціалів

Заповнюємо U_i та V_j. Покладемо U_1=0, та рахуємо інші U_i та V_j з формули $U_i + V_j = C_{ij}$, C_{ij} береться тільки з базисних клітинках. $U_i = C_{ij} - V_j$ та $V_j = C_{ij} - U_i$.

V_2 = 8-0 = 8; U_2 = 5-8 = -3; V_1 = 2--3 = 5; V_3 = 1-0 = 1; V_4 = 2-0 = 2; U_3 = 1-2 = -1;

a\b	B_1= 0	B_2= 0	B_3= 0	B_4= 0	U
A_1= 0	C_{11}=5	C_{12}=8 X_{12}=45	C_{13}=1 ✓✓ X_{13}=130	C_{14}=2 X_{14}=35	U_1= 0
A_2= 0	C_{21}=2 ✓✓ X_{21}=125	C_{22}=5 X_{22}=45	C_{23}=4	C_{24}=9	U_2= -3
A_3= 0	C_{31}=9	C_{32}=2 ✓	C_{33}=3	C_{34}=1 ✓✓ X_{34}=65	U_3= -1
V	V_1= 5	V_2= 8	V_3= 1	V_4= 2	

Рис.3.6. Знаходження потенціалів для транспортної задачі

Розрахунок потенціалів складається з двох рекурсивних функцій, які викликають одна одну та обчислюють U_i і V_j. Розглянемо функцію для пошуку U_i:

//пошук u: в функцію передається поточний стан масиву даних і

```
        //перший рядок та стовчик це NULL
        findU: function (currentIndex, currentI, currentJ) {
        //як що ми ще не шукали потенціали ми ставимо u1=0 і переходимо шукати всі vj
        //залежні від u1.
          if (currentJ === null) {
            this.data[currentIndex].u[currentI] = 0;
            this.findV(currentIndex, currentI, currentJ)
            return;
          }
        //якщо вже шукали  u то шукаємо  в рядку базисні клітинки і вираховуємо ui
        //і переходимо шукати всі vj залежні від ui.
          else {
            for (let i = 0; i < this.n; i += 1) {
              if (this.data[currentIndex].data[i][currentJ].x !== null && i !== currentI &&
this.data[currentIndex].u[i] === null) {
                this.data[currentIndex].u[i] = this.data[currentIndex].data[i][currentJ].c -
this.data[currentIndex].v[currentJ];
this.findV(currentIndex, i, currentJ);
              }
            }
          }

        }
```

Відштовхуючись від першого U_i =0 вираховує V_j через базисні клітинки викликаючи функцію, яка вираховує U_i відштовхуються від вирахуваного V_j. Виходом з рекурсій є не можливість більше вирахувати U_i або V_j. Після чого перевіряється чи план оптимальний.

3.3.3. Функції перевірки оптимального плану

Рахуються D_{ij} (рис.3.7), які мають бути від'ємними. Як що всі D від'ємні то план оптимальний, отже рахуємо вартість перевезення, яка дорівнює сумі

кількості продукції помноженій на ціну. В іншому випадку шукаються мінімальне D_{ij} , яке вводиться в базис.

Перевіряємо чи план оптимальний

Для перевірки плану на оптимальність шукаємо D_{ij} для незаповнених клітинок за формулою $D_{ij}=C_{ij}-V_j-U_i$. Якщо всі D>=0 то план оптимальний, в іншому випадку вибираємо найменше D і вводим його в базис.

$D_{1\,1}$=5-5-0=0; $D_{2\,3}$=4-1--3=6; $D_{2\,4}$=9-2--3=10; $D_{3\,1}$=9-5--1=5; $D_{3\,2}$=2-8--1=-5; $D_{3\,3}$=3-1--1=3;

ПЛАН НЕ ОПТИМАЛЬНИЙ!!! Вводим в базис клітинку [3,2]

a\b	B_1= 0	B_2= 0	B_3= 0	B_4= 0	U
A_1= 0	C_{11}=5 D_{11}=0	C_{12}=8 X_{12}=45	C_{13}=1 X_{13}=130	C_{14}=2 X_{14}=35	U_1= 0
A_2= 0	C_{21}=2 X_{21}=125	C_{22}=5 X_{22}=45	C_{23}=4 D_{23}=6	C_{24}=9 D_{24}=10	U_2= -3
A_3= 0	C_{31}=9 D_{31}=5	C_{32}=2 D_{32}=-5	C_{33}=3 D_{33}=3	C_{34}=1 X_{34}=65	U_3= -1
V	V_1= 5	V_2= 8	V_3= 1	V_4= 2	

Рис.3.7. Знаходженя D_{ij} для перевірки плану на оптимальність.

Від клітинки [i,j] будуються цикл по базисних клітинках за допомогою двох рекурсій, які ідуть по вертикалі і горизонталі. Розглянемо одну із них:

```
//функція отримує індекси поточних клітинок, масив даних уже вибраних клітинок
//створення циклу по ньому поточні індекси елемента і  індекси введеної клітинки в базис
        baseVertical: function (currentIndex, cells, currentI, currentJ, startI, startJ) {
//перевірка чи вертикальна клітинка не може перейти на введену клітинку як що
//може то ми виводимо масив базисних клітинок
            if (currentJ === startJ) {
                return cells;
            }
//в іншому випадку ми ідемо циклом по рядку шукаємо базисні клітинки

            for (let i = 0; i < this.n; i += 1) {
//якщо є базисна клітинка і вона не та з якої ми почали наш цикл  дописуємо її в масив
                if (this.data[currentIndex].data[i][currentJ].x !== null && i !== currentI) {let temp = {'data':
this.data[currentIndex].data[i][currentJ], 'i': i, 'j': currentJ},
//викликаємо пошук горизонтальної базисної клітинки передаючи масив
 //індекси цієї клітинки, яку ми знайшли
                    tempCells = this.baseHorizontal(currentIndex, cells.concat([temp]), i, currentJ, startI, startJ);
                }
            }
//якщо  ми не знайшли базисної клітинки ми повертаємося і перевіряємо
 //наступну базисну клітинку, і передаємо порожній масив
            return [];},
```

Виходом з рекурсій є побудований цикл клітинок. Проходячи по масиву вибраних клітинок додаток змінює дані транспортної задачі починаючи з введеної клітинки в базис по порядку додає та віднімає мінімальну кількість продукції. А також клітинку в якій кількість продукції рівна нулю виводимо з базису (рис.3.8). Після цього знову очищуємо всі u,v,d вирахування потенціалів.

Вводимо в базис клітинку

Вводимо клітинку в базис зі значенням X=0. Будуємо цикл по базисних клітинках починаючи з клітинки, яку ми ввели в базис. Потім по черзі додаємо і віднімаємо, починаючи з введеної клітинки, значення яке вираховується min значення серед клітинок від яких треба віднімати.

$X_{3\,2}$ ->$X_{3\,4}$ ->$X_{1\,4}$ ->$X_{1\,2}$ рахуємо значення яке будем віднімати min{65,45}=45записуємо нові значення X: $X_{3\,2}$=0+45=45; $X_{3;4}$=65-45=20; $X_{1\,4}$=35+45=80; $X_{1;2}$=45-45=0; Виводимо з базису X=0. Знову перевіряємо чи план оптимальний

a\b	B_1= 125	B_2= 90	B_3= 130	B_4= 100
A_1= 210	C_{11}=5	C_{12}=8	C_{13}=1 X_{13}=130	C_{14}=2 X_{14}=80
A_2= 170	C_{21}=2 X_{21}=125	C_{22}=5 X_{22}=45	C_{23}=4	C_{24}=9
A_3= 65	C_{31}=9	C_{32}=2 X_{32}=45	C_{33}=3	C_{34}=1 X_{34}=20

Рис.3.8. Введено в базис клітинку для знаходження оптимального плану.

3.4. Вивід відповіді

Задача вважається розв'язаною коли ми знайшли оптимальний план (рис.3.9). Виводиться вартість доставки і таблиця з кількістю продуктів, які перевозяться від продавця споживачу(рис.3.10).

Перевіряємо чи план оптимальний

Для перевірки плану на оптимальність шукаємо D_{ij} для незаповнених клітинок за формулою $D_{ij}=C_{ij}-V_j-U_i$. Якщо всі D>=0 то план оптимальний, в іншому випадку вибираємо найменше D і вводим його в базис.

$D_{1\,1}$=5-0-0=5; $D_{1\,2}$=8-3-0=5; $D_{2\,3}$=4-1-2=1; $D_{2\,4}$=9-2-2=5; $D_{3\,1}$=9-0--1=10; $D_{3\,3}$=3-1--1=3;

ПЛАН ОПТИМАЛЬНИЙ

A \ B	B_1= 125	B_2= 90	B_3= 130	B_4= 100	U
A_1= 210	C_{11}=5 D_{11}=5	C_{12}=8 D_{12}=5	C_{13}=1 X_{13}=130	C_{14}=2 X_{14}=80	U_1= 0
A_2= 170	C_{21}=2 X_{21}=125	C_{22}=5 X_{22}=45	C_{23}=4 D_{23}=1	C_{24}=9 D_{24}=5	U_2= 2
A_3= 65	C_{31}=9 D_{31}=10	C_{32}=2 X_{32}=45	C_{33}=3 D_{33}=3	C_{34}=1 X_{34}=20	U_3= -1
V	V_1= 0	V_2= 3	V_3= 1	V_4= 2	

Рис. 3.9. Знайдено оптимальний план.

Вартість доставки продукції: 875

A\B	B_1= 125	B_2= 90	B_3= 130	B_4= 100
A_1= 210	0	0	130	80
A_2= 170	125	45	0	0
A_3= 65	0	45	0	20

Рис.3.10. Вивід результату.

Висновки

За допомогою розробленого веб-додатку користувач може легко зрозуміти процес розв'язування транспортної задачі і водночас перевірити свої знання. В роботі наведено 3 методи побудови опорного плану. Покрокове виведення даних набагато краще демонструє всі нюанси в розв'язку транспортної задачі.

Плюсом таких навчальних інтерактивних програмних засобів є те, що їх можна використовувати в університеті для перевірки знань студентів. А також те, що за їх допомоги можна явно демонструвати процес розв'язування транспортної задачі, чим полегшити й пришвидшити процес вивчення даної теми.

Список використаної літератури

1. Пасічник Г.С., Кушнірчук В.Й. Методи оптимізації: лінійне програмування: Навчальний посібник – Чернівці: Золоті литаври, 2012. – 32-65 с.
2. Пасічник Г.С., Кушнірчук В.Й. Методи оптимізації: транспортна задача: Навчальний посібник – Чернівці: Золоті литаври, 2013. – 13-120 с.
3. Навчальні програми і приклади для вивчення HTML, CSS, JS та багатьох інших технологій веб-розробки [Електронний ресурс]. – Режим доступу**:** https://www.w3schools.com
4. WEB-сторінку кластерних систем Інституту кібернетики імені В.М. Глушкова НАН України [Електронний ресурс]. – Режим доступу: http://icybcluster.org.ua/
5. Документаця по фреймворку Vue.js [Електронний ресурс]. – Режим доступу**:** https://vuejs.org/v2/guide/index.html

Додатки

Дотаток 1

```
<template>
  <div class="wrapper">
    <div v-if="flag">
      <label>            Кількість виробників A:
        <input type="number" min="1" class="input-sm form-control" v-model="n">
      </label>
      <label>            Кількість споживачів B:
        <input type="number" min="1" class="input-sm form-control" v-model="m">
      </label>
      <label>
        Метод:
        <select v-model="method" class="form-control">
          <option v-for="method in methodsList" :value="method">{{method.title}}</option>
        </select>
      </label>
      <button class="btn btn-success" @click="checkEnteredData();">Обчислити
      </button>
      <h3 class="has-error">{{error}}</h3>
      <p> A<sub>i</sub> - кількість одиниць продукції, виготовленої виробниками.
        B<sub>j</sub> - кількість одиниць продукції, яку потребують споживачі.
        C<sub>ij</sub> - ціна за перевесення одиниці продукції.</p>
      <p>A<sub>i</sub>>0, B<sub>j</sub>>0, C<sub>ij</sub>>0.</p>
      <table class="table table-bordered">
        <tr>
          <th>A\B</th>
          <th v-for="(item, index) in data[0].b">
            <label>
              B<sub>{{index + 1}}</sub>=
              <input type="number" min="1" v-model="data[0].b[index]">
            </label>
          </th>
        </tr>
        <tr v-for="(row, i) in data[0].data">
          <td>
            <label>
              A<sub>{{i + 1}}</sub>=
              <input type="number" min="1" v-model="data[0].a[i]">
            </label>
```

```
            </td>
            <td v-for="(item, j) in row">
              <label>
                C<sub>{{i + 1}}{{j + 1}}</sub>=
                <input type="number" min="1" v-model="data[0].data[i][j].c">
              </label>
            </td>
          </tr>
        </table>
      </div>
      <div v-else>
        <button class=" btn btn-success button" @click="edit()">Змінити умову</button>
        <button class="btn badge-pill" @click="print()">Друк</button>
        <div v-for="state in data">
          <h2 v-html="state.title">{{state.title}}</h2>
          <h4 v-html="state.subtitle">{{state.subtitle}}</h4>
          <h6 v-html="state.teor">{{state.teor}}</h6>
          <div v-html="state.description">{{state.description}}</div>
          <table class="table table-bordered">
            <tr>
              <th>A \ B</th>
              <th v-for="(item, index) in state.b">
                <label>
                  B<sub>{{index + 1}}</sub>= {{item}}
                </label>
              </th>
              <th v-if="state.showVU">U</th>
            </tr>
            <tr v-for="(row, i) in state.data">
              <td>
                <label>
                  A<sub>{{i + 1}}</sub>= {{state.a[i]}}
                </label>
              </td>
              <td class="cell" v-for="(item, j) in row" v-bind:class="{
                baz:(item.x !== null),
                onecheck:item.minV,
                twocheck:item.minG
              }">
```

```
            <div>
              C<sub>{{i + 1}}{{j + 1}}</sub>={{item.c}}
              <div v-if="item.x !== null">X<sub>{{i + 1}}{{j + 1}}</sub>={{item.x}}</div>
              <div v-if="item.d !== null">D<sub>{{i + 1}}{{j + 1}}</sub>={{item.d}}</div>
            </div>
          </td>
          <td v-if="state.showVU">
            <label>
              U<sub>{{i + 1}}</sub>= {{state.u[i]}}
            </label>
          </td>
        </tr>
        <tr v-if="state.showVU">
          <th>V</th>
          <td v-for="(item, index) in state.v">
            <label>
              V<sub>{{index + 1}}</sub>= {{item}}
            </label>
          </td>
        </tr>
      </table>
    </div>
    <template v-if="data[data.length-1].f!==0">
      <h1> Вартість доставки продукції: {{data[data.length - 1].f}}</h1>
      <table class="table table-bordered" v-bind="state=data[data.length-1]">
        <tr>
          <th>A\B</th>
          <th v-for="(item, index) in state.b">
            <label>
              B<sub>{{index + 1}}</sub>= {{item}}
            </label>
          </th>
        </tr>
        <tr v-for="(row, i) in state.data">
          <td>
            <label>
              A<sub>{{i + 1}}</sub>= {{state.a[i]}}
            </label>
          </td>
```

```
                <td class="cell" v-for="(item, j) in row" >
                    <div v-if="item.x !== null">{{item.x}}</div>
                    <div v-else>0</div>
                </td>
            </tr>
        </table>
    </template>
    <button class=" btn btn-success button" @click="edit()">Змінити умову</button>
    <button class="btn badge-pill" @click="print()">Друк</button>
  </div>
 </div>
</template>
<script>
  export default {
    name: 'app',
    data: function () {
      return {
        title: "",
        sumA: 0,
        sumB: 0,
        n: 1,
        m: 1,
        method: null,
        methodsList: [
          {
            title: 'Мінімального елемента',
            functionName: 'Minimal'
          },
          {
            title: 'Північно-західного кута',
            functionName: 'NorthWest'
          },
          {
            title: 'Подвійної переваги',
            functionName: 'Two'
          }
        ],
        flag: true,
        error: "",
```

```
referencePlanFlag: false,
data: [],
state: {
    a: [],
    b: [],
    data: [],
    u: [],
    v: [],
    f: 0,
    showVU: false,
    countFullX: 0,
    title: "Початкові дані",
    subtitle: "",
    teor: "",
    description: "",
},
item: {
    c: null,
    x: null,
    d: null
},
tempCount: 0,
defaultData: {
    n: 3,
    m: 4,
    data: [
        {
            "a": [
                "210",
                "170",
                "65"
            ],
            "b": [
                "125",
                "90",
                "130",
                "100"
            ],
            "showVU": false,
```

```
"data": [
  [
    {
      "c": "5",
      "x": null,
      "d": null,
      'minV': false,
      'minG': false
    },
    {
      "c": "8",
      "x": null,
      "d": null,
      'minV': false,
      'minG': false
    },
    {
      "c": "1",
      "x": null,
      "d": null,
      'minV': false,
      'minG': false
    },
    {
      "c": "2",
      "x": null,
      "d": null,
      'minV': false,
      'minG': false
    }
  ],
  [
    {
      "c": "2",
      "x": null,
      "d": null,
      'minV': false,
      'minG': false
    },
```

```
    {
      "c": "5",
      "x": null,
      "d": null,
      'minV': false,
      'minG': false
    },
    {
      "c": "4",
      "x": null,
      "d": null,
      'minV': false,
      'minG': false
    },
    {
      "c": "9",
      "x": null,
      "d": null,
      'minV': false,
      'minG': false
    }
  ],
  [
    {
      "c": "9",
      "x": null,
      "d": null,
      'minV': false,
      'minG': false
    },
    {
      "c": "2",
      "x": null,
      "d": null,
      'minV': false,
      'minG': false
    },
    {
      "c": "3",
```

```
                    "x": null,
                    "d": null,
                    'minV': false,
                    'minG': false
                },
                {
                    "c": "1",
                    "x": null,
                    "d": null,
                    'minV': false,
                    'minG': false
                }
            ]
        ],
        "u": [
            null,
            null,
            null
        ],
        "v": [
            null,
            null,
            null,
            null
        ],
        "f": 0,
        "countFullX": 0,
        "title": "Початкові дані",
        "description": "",
    }
],
method: {
    title: 'Мінімального елемента',
    functionName: 'Minimal'
}  } }  },
created: function () {
  for (let key in this.defaultData) {
    this[key] = JSON.parse(JSON.stringify(this.defaultData[key]));
  }
```

```
},
methods: {
    edit: function () {
        this.data = [this.data[0]];
        this.flag = true;
    },
    print: function () {
        window.print();
    },
    checkEnteredData: function () {
        this.sumA = 0;
        this.sumB = 0;
        let a, b, c;
        for (let j = 0; j < this.m; j += 1) {
            b = parseInt(this.data[0].b[j], 10);
            if (isNaN(b) || b <= 0) {
                this.error = "Задача незбалансована або не всі дані введені!";
                this.flag = true;
                return;
            }
            this.data[0].b[j] = b;
            this.sumB += b;
        }

        for (let i = 0; i < this.n; i += 1) {
            a = parseInt(this.data[0].a[i], 10);
            if (isNaN(a) || a <= 0) {
                this.error = "Задача незбалансована або не всі дані введені!";
                this.flag = true;
                return;
            }
            this.data[0].a[i] = a;
            this.sumA += a;
            for (let j = 0; j < this.m; j += 1) {
                c = parseInt(this.data[0].data[i][j].c, 10);
                if (isNaN(c) || c <= 0) {
                    this.error = "Задача незбалансована або не всі дані введені!";
                    this.flag = true;
                    return;
```

```
            }
            this.data[0].data[i][j].c = c;
          }
        }
        if (this.sumB !== this.sumA || this.sumB === 0 || this.sumA === 0) {
          this.error = "Задача незбалансована або не всі дані введені!";
          this.flag = true;
          return;
        }
        this.error = "";
        this.flag = false;
        this.referencePlan();
      },
      referenceTwoPlan: function (k) {
        let currentIndex = this.data.length - 1;
        k += 1;
        this.data.push(JSON.parse(JSON.stringify(this.data[currentIndex])));
        currentIndex = this.data.length - 1;
        this.data[currentIndex].subtitle = "Крок     " + k;
        if (k === 1) {
          this.data[currentIndex].title = "Будуємо опорний план методом подвійної переваги";
        }
        this.data[currentIndex].teor = "В кожному рядку відмічаємо галочкою клітинку з найменшою ціною,
так само в стовчиках.";
        for (let i = 0; i < this.n; i += 1) {
          let minC = {
            value: null,
            indexJ: null,
            indexI: null,
          };
          for (let j = 0; j < this.m; j += 1) {
            if (minC.value === null || minC.value > this.data[currentIndex].data[i][j].c) {
              if (this.data[currentIndex].data[i][j].x === null &&
                this.data[currentIndex].a[i] !== 0 &&
                this.data[currentIndex].b[j] !== 0) {
                minC.value = this.data[currentIndex].data[i][j].c;
                minC.indexI = i;
                minC.indexJ = j;
              }} }
```

```
            this.data[currentIndex].data[minC.indexI][minC.indexJ].minV = true; }
        for (let j = 0; j < this.m; j += 1) {
            let minC = {
                value: null,
                indexJ: null,
                indexI: null,
            };
            for (let i = 0; i < this.n; i += 1) {
                if (minC.value === null || minC.value > this.data[currentIndex].data[i][j].c) {
                    if (this.data[currentIndex].data[i][j].x === null &&
                        this.data[currentIndex].a[i] !== 0 &&
                        this.data[currentIndex].b[j] !== 0) {
                        minC.value = this.data[currentIndex].data[i][j].c;
                        minC.indexI = i;
                        minC.indexJ = j;
                    } } }
            this.data[currentIndex].data[minC.indexI][minC.indexJ].minG = true;
        }
        this.data.push(JSON.parse(JSON.stringify(this.data[currentIndex])));
        currentIndex = this.data.length - 1;
        k += 1;
        this.data[currentIndex].subtitle = "Крок     " + k;
        this.data[currentIndex].title = "";
        this.data[currentIndex].teor = "Клітинки, які мають дві галочки заповнюємо першими.
X<sub>ij</sub>=min{A<sub>i</sub>; B<sub>j</sub>}, " +
            "A<sub>i</sub>=A<sub>i</sub>-X<sub>ij</sub>, B<sub>j</sub>=B<sub>j</sub>-X<sub>ij</sub>.";
        for (let i = 0; i < this.n; i += 1) {
            for (let j = 0; j < this.m; j += 1) {
                if (this.data[currentIndex].data[i][j].minV && this.data[currentIndex].data[i][j].minG) {
                    if (this.data[currentIndex].data[i][j].x === null &&
                        this.data[currentIndex].a[i] !== 0 &&
                        this.data[currentIndex].b[j] !== 0) {
                        this.data[currentIndex].data[i][j].x = Math.min(this.data[currentIndex].a[i],
this.data[currentIndex].b[j]);
                        this.data[currentIndex].a[i] = this.data[currentIndex].a[i] - this.data[currentIndex].data[i][j].x;
                        this.data[currentIndex].b[j] = this.data[currentIndex].b[j] - this.data[currentIndex].data[i][j].x;
                        this.sumA = this.sumA - this.data[currentIndex].data[i][j].x;
                        this.sumB = this.sumB - this.data[currentIndex].data[i][j].x;
                        this.data[currentIndex].countFullX += 1;
```

```
                } } } }
            this.data.push(JSON.parse(JSON.stringify(this.data[currentIndex])));
            currentIndex = this.data.length - 1;
            k += 1;
            this.data[currentIndex].subtitle = "Крок    " + k;
            this.data[currentIndex].teor = "Клітинки, які мають одну галочки заповнюємо наступними. X<sub>ij</sub>=min{A<sub>i</sub>;B<sub>j</sub>}, " +
                "A<sub>i</sub>=A<sub>i</sub>-X<sub>ij</sub>, B<sub>j</sub>=B<sub>j</sub>-X<sub>ij</sub>.";
            for (let i = 0; i < this.n; i += 1) {
                for (let j = 0; j < this.m; j += 1) {
                    if (this.data[currentIndex].data[i][j].minV || this.data[currentIndex].data[i][j].minG) {
                        if (this.data[currentIndex].data[i][j].x === null &&
                            this.data[currentIndex].a[i] !== 0 &&
                            this.data[currentIndex].b[j] !== 0) {
                            this.data[currentIndex].data[i][j].x = Math.min(this.data[currentIndex].a[i], this.data[currentIndex].b[j]);
                            this.data[currentIndex].a[i] = this.data[currentIndex].a[i] - this.data[currentIndex].data[i][j].x;
                            this.data[currentIndex].b[j] = this.data[currentIndex].b[j] - this.data[currentIndex].data[i][j].x;
                            this.sumA = this.sumA - this.data[currentIndex].data[i][j].x;
                            this.sumB = this.sumB - this.data[currentIndex].data[i][j].x;
                            this.data[currentIndex].countFullX += 1;
                        } } } }
            this.data.push(JSON.parse(JSON.stringify(this.data[currentIndex])));
            currentIndex = this.data.length - 1;
            k += 1;
            this.data[currentIndex].subtitle = "Крок    " + k;
            if (this.sumA !== 0 || this.sumB !== 0) {
                this.data[currentIndex].teor ="";
                this.data[currentIndex].description = "Опорний план ще не побудований. Далі розподіляємо X за методом мінімальних елементів";
                this.referenceMinimalPlan(k);
            }
            return true;
        },
        referenceMinimalPlan: function (k) {
            let referencePlanFlag = true;
            let currentIndex = this.data.length - 1;

            while (referencePlanFlag) {
                k += 1;
```

```
this.data.push(JSON.parse(JSON.stringify(this.data[currentIndex])));
currentIndex = this.data.length - 1;
let minC = {
  value: null,
  indexJ: null,
  indexI: null,
};
if (k === 1) {
  this.data[currentIndex].title = "Будуємо опорний план методом мінімального елемента";
}
else {
  this.data[currentIndex].title = "";
}
this.data[currentIndex].subtitle = "Крок     " + k;
for (let i = 0; i < this.n; i += 1) {
  for (let j = 0; j < this.m; j += 1) {
    if (minC.value === null || minC.value > this.data[currentIndex].data[i][j].c) {
      if (this.data[currentIndex].data[i][j].x === null &&
        this.data[currentIndex].a[i] !== 0 &&
        this.data[currentIndex].b[j] !== 0) {
        minC.value = this.data[currentIndex].data[i][j].c;
        minC.indexI = i;
        minC.indexJ = j;
      }
    }
  }
}
this.data[currentIndex].teor = "Для побудови опорного плану потрібно " +
  "знайти найменшу ціну  (C<sub>ij</sub>) з незаповнених клітинок, а також A<sub>i</sub> >0 і B<sub>j</sub> >0." +
  " Тоді в X<sub>ij</sub>=min{A<sub>i</sub>; B<sub>j</sub>} і A<sub>i</sub>=A<sub>i</sub>-X<sub>ij</sub> ," +
  " B<sub>j</sub>=B<sub>j</sub>-X<sub>ij</sub>.";
this.data[currentIndex].description = "Найменша ціна з клітинок, які не містить X - це C<sub>" +
  parseInt(minC.indexI + 1) + "  " + parseInt(minC.indexJ + 1) + "</sub>=" +
  this.data[currentIndex].data[minC.indexI][minC.indexJ].c + " в  клітинці [" + parseInt(minC.indexI + 1) + ";" + parseInt(minC.indexJ + 1) + "]. " +
  "Заповнюємо X<sub>" + parseInt(minC.indexI + 1) + "  " + parseInt(minC.indexJ + 1) + "</sub>= min{" + this.data[currentIndex].a[minC.indexI] + ";" + this.data[currentIndex].b[minC.indexJ]
  + "}. А також  змінюємо значення  A<sub>" +
```

```
            parseInt(minC.indexI + 1) + "</sub>=" + this.data[currentIndex].a[minC.indexI] + "-" +
Math.min(this.data[currentIndex].a[minC.indexI],
              this.data[currentIndex].b[minC.indexJ]) + " i  B<sub>" + parseInt(minC.indexJ + 1) + "</sub>=" +
this.data[currentIndex].b[minC.indexJ] + "-"
            + Math.min(this.data[currentIndex].a[minC.indexI], this.data[currentIndex].b[minC.indexJ]) + ".";
          this.data[currentIndex].countFullX += 1;
          this.data[currentIndex].data[minC.indexI][minC.indexJ].x =
Math.min(this.data[currentIndex].a[minC.indexI], this.data[currentIndex].b[minC.indexJ]);
          this.data[currentIndex].a[minC.indexI] -= this.data[currentIndex].data[minC.indexI][minC.indexJ].x;
          this.data[currentIndex].b[minC.indexJ] -= this.data[currentIndex].data[minC.indexI][minC.indexJ].x;
          this.sumA -= this.data[currentIndex].data[minC.indexI][minC.indexJ].x;
          this.sumB -= this.data[currentIndex].data[minC.indexI][minC.indexJ].x;
          if (this.sumA === 0 && this.sumB === 0) {
            referencePlanFlag = false;
          }
          else if (this.sumA === 0 || this.sumB === 0) {
            referencePlanFlag = false;
            this.error.notReferencePlan = true;
          }
        }
        return true;
      },
      referenceNorthWestPlan: function (k) {
        let currentIndex = this.data.length - 1;
        for (let i = 0; i < this.n; i += 1) {
          for (let j = 0; j < this.m; j += 1) {
            if (this.data[currentIndex].data[i][j].x === null &&
              this.data[currentIndex].a[i] !== 0 &&
              this.data[currentIndex].b[j] !== 0) {
              k += 1;
              this.data.push(JSON.parse(JSON.stringify(this.data[currentIndex])));
              currentIndex = this.data.length - 1;
              if (k === 1) {
                this.data[currentIndex].title = "Будуємо опорний план методом північно-західного кута";
                this.data[currentIndex].teor = "На першому кроці заповнюємо клітинку [1;1] " +
                  "X<sub>11</sub>=min{A<sub>1</sub>;B<sub>1</sub>}, " + "
B<sub>1</sub>=B<sub>1</sub>-X<sub>11</sub>," +
                  " A<sub>1</sub>=A<sub>1</sub>-X<sub>11</sub>. " +
                  " Далі є 3 варіанти :<br>" +
                  " Якщо A<sub>i</sub>< B<sub>j</sub>, то в X<sub>ij</sub>=A<sub>i</sub>, " +
```

```
                    "B<sub>j</sub>=B<sub>j</sub>-A<sub>i</sub>, " +
                    "A<sub>i</sub>=0, та стовчик j не розглядаємо в подальшому; <br> " +
                    " Якщо A<sub>i</sub> > B<sub>j</sub>, то в X<sub>ij</sub>=B<sub>i</sub>, " +
                    "A<sub>j</sub>=A<sub>j</sub>-B<sub>i</sub>, " +
                    "B<sub>i</sub>=0 та рядок i не розглядаємо в подальшому;<br>" +
                    " Якщо A<sub>i</sub> = B<sub>j</sub>, то в X<sub>ij</sub>=B<sub>i</sub>, " +
                    "A<sub>j</sub>=0, B<sub>i</sub>=0 та рядок i та стовпчик j не розглядаємо в
подальшому;<br>";
                this.data[currentIndex].description = "Покладемо в клітинку [1;1]  X<sub>" +
                    "11</sub>=min{" + this.data[currentIndex].a[i] + ";" + this.data[currentIndex].b[j] + "}=" +
                    Math.min(this.data[currentIndex].a[i], this.data[currentIndex].b[j]) + ", B<sub>" + (j + 1) +
"</sub>=" + this.data[currentIndex].b[j] +
                    "-" + (Math.min(this.data[currentIndex].a[i], this.data[currentIndex].b[j])) + "=" +
                    (this.data[currentIndex].b[j] - Math.min(this.data[currentIndex].a[i],
this.data[currentIndex].b[j]) ) +
                    ", A<sub>" + (i + 1) + "</sub>=" + this.data[currentIndex].a[i] +
                    "-" + (Math.min(this.data[currentIndex].a[i], this.data[currentIndex].b[j])) + "=" +
                    (this.data[currentIndex].a[i] - Math.min(this.data[currentIndex].a[i],
this.data[currentIndex].b[j]) );
                }
                else {
                    this.data[currentIndex].title = "";
                }
                if (this.data[currentIndex].a[i] === this.data[currentIndex].b[j] && k !== 1) {
                    this.data[currentIndex].description = "B<sub>" + (j + 1) + "</sub> = A<sub>" + (i + 1) +
"</sub>. Отже " +
                        "X<sub>" + (i + 1) + " " + (j + 1) + "</sub>=" + this.data[currentIndex].b[j] + ", A<sub>" + (i
+ 1) + "</sub>=0, " +
                        "B<sub>" + (j + 1) + "</sub>=0,  та стовпчик " + (j + 1) + "i рядок " + (i + 1) + " не
розглядаємо в подальшому;";
                } else if (this.data[currentIndex].a[i] === Math.min(this.data[currentIndex].a[i],
this.data[currentIndex].b[j]) && k !== 1) {
                    this.data[currentIndex].description = "A<sub>" + (i + 1) + "</sub> < B<sub>" + (j + 1) +
                        "</sub>. Отже X<sub>" + (i + 1) + " " + (j + 1) + "</sub> =" + this.data[currentIndex].a[i] + ",
" + "B<sub>" + (j + 1) + "</sub>=" + this.data[currentIndex].b[j] + "-" +
                        this.data[currentIndex].a[i] + "=" + (this.data[currentIndex].b[j] - this.data[currentIndex].a[i]) +
                        ", A<sub>" + (i + 1) + "</sub>=0, та рядок " + (i + 1) + " не розглядаємо в подальшому;";
                } else if (this.data[currentIndex].b[j] === Math.min(this.data[currentIndex].a[i],
this.data[currentIndex].b[j]) && k !== 1) {
                    this.data[currentIndex].description = "B<sub>" + (j + 1) + "</sub> < A<sub>" + (i + 1) +
"</sub>. Отже X<sub>"
                        + (i + 1) + " " + (j + 1) + "</sub>=" + this.data[currentIndex].b[j] + ", A<sub>" + (i + 1) +
"</sub>=" + this.data[currentIndex].a[i] + "-" +
```

```
                    this.data[currentIndex].b[j] + "=" + (this.data[currentIndex].a[j] - this.data[currentIndex].b[j]) +
                    ", B<sub>" + (j + 1) + "</sub>=0,  та стовпчик " + (j + 1) + " не розглядаємо в
подальшому;";
                }
                this.data[currentIndex].subtitle = "Крок     " + k;
                this.data[currentIndex].data[i][j].x = Math.min(this.data[currentIndex].a[i],
this.data[currentIndex].b[j]);
                this.data[currentIndex].a[i] = this.data[currentIndex].a[i] - this.data[currentIndex].data[i][j].x;
                this.data[currentIndex].b[j] = this.data[currentIndex].b[j] - this.data[currentIndex].data[i][j].x;
                this.sumA = this.sumA + this.data[currentIndex].data[i][j].x;
                this.sumB = this.sumB + this.data[currentIndex].data[i][j].x;
                this.data[currentIndex].countFullX += 1;
              }
            }
          }
          if (this.sumA !== this.sumB) {
            this.error.notReferencePlan = true;
          }
          return true;
        },
        referencePlan: function () {
          if (this['reference' + this.method.functionName + 'Plan'](0)) {
            this.optimalPlan();
          }
        },
        findUV: function () {
          let currentIndex = this.data.length - 1;
          this.data.push(JSON.parse(JSON.stringify(this.data[currentIndex])));
          currentIndex += 1;
          this.data[currentIndex].showVU = true;
          this.data[currentIndex].title = "Обчислюємо значення потенціалів";
          this.data[currentIndex].subtitle = "";
          this.data[currentIndex].teor = "Заповнюємо U<sub>i</sub> та V<sub>j</sub>. Покладемо
U<sub>1</sub>=0," +
            " та рахуємо інші U<sub>i</sub> та V<sub>j</sub> за формули  U<sub>i</sub> +
V<sub>j</sub>=C<sub>ij</sub>,  C<sub>ij</sub> рухаючись  базисними клітинким. " +
            " U<sub>i</sub> =C<sub>ij</sub>-V<sub>j</sub> та
V<sub>j</sub>=C<sub>ij</sub>-U<sub>i</sub>.";
          this.data[currentIndex].description = "";
          this.findU(currentIndex, 0, null);
```

```
    },
    findU: function (currentIndex, currentI, currentJ) {
      if (currentJ === null) {
        this.data[currentIndex].u[currentI] = 0;
        this.findV(currentIndex, currentI, currentJ);
        return null;
      }
      else {
        for (let i = 0; i < this.n; i += 1) {
          if (this.data[currentIndex].data[i][currentJ].x !== null && i !== currentI &&
this.data[currentIndex].u[i] === null) {
            this.data[currentIndex].u[i] = this.data[currentIndex].data[i][currentJ].c -
this.data[currentIndex].v[currentJ];
            this.data[currentIndex].description = this.data[currentIndex].description + "U<sub>" + (i + 1) +
"</sub> = " +
              this.data[currentIndex].data[i][currentJ].c + "-" + this.data[currentIndex].v[currentJ] +
              " = " + (this.data[currentIndex].data[i][currentJ].c - this.data[currentIndex].v[currentJ]) + ";     ";
            this.findV(currentIndex, i, currentJ);
          }} } },
    findV: function (currentIndex, currentI, currentJ) {
      for (let j = 0; j < this.m; j += 1) {
        if (this.data[currentIndex].data[currentI][j].x !== null && this.data[currentIndex].v[j] === null) {
          this.data[currentIndex].v[j] = this.data[currentIndex].data[currentI][j].c -
this.data[currentIndex].u[currentI];
          this.data[currentIndex].description = this.data[currentIndex].description + "V<sub>" + (j + 1) +
"</sub> = " +
            this.data[currentIndex].data[currentI][j].c + "-" + this.data[currentIndex].u[currentI] +
            " = " + (this.data[currentIndex].data[currentI][j].c - this.data[currentIndex].u[currentI]) + ";     ";
          currentJ = j;
          this.findU(currentIndex, currentI, currentJ)
        }
      }
    },
    optimalPlan: function () {
      this.findUV();
      let currentIndex = this.data.length - 1;
      this.data.push(JSON.parse(JSON.stringify(this.data[currentIndex])));
      currentIndex += 1;
      this.data[currentIndex].title = "Перевіряємо чи план оптимальний";
      this.data[currentIndex].teor = "Для перевірки плану на оптимальність шукаємо D<sub>ij</sub> для
незаповнених клітинок за " +
```

```
" формулою D<sub>ij</sub>=C<sub>ij</sub>-V<sub>j</sub>-U<sub>i</sub>. Якщо всі D>=0 то план оптимальний, в іншому випадку вибираємо найменше D і вводим його в базис. ";
        this.data[currentIndex].description = "";
        if (this.data[currentIndex].countFullX === (parseInt(this.n) + parseInt(this.m) - 1)) {
          let min = null,
            minI = null,
            minJ = null;
          for (let i = 0; i < this.n; i += 1) {
            for (let j = 0; j < this.m; j += 1) {
              if (this.data[currentIndex].data[i][j].x === null) {
                this.data[currentIndex].data[i][j].d = this.data[currentIndex].data[i][j].c -
this.data[currentIndex].v[j] - this.data[currentIndex].u[i];
                this.data[currentIndex].description = this.data[currentIndex].description +
                  "D<sub>" + (i + 1) + " " + (j + 1) + "</sub>=" + this.data[currentIndex].data[i][j].c + "-" +
this.data[currentIndex].v[j] + "-" + this.data[currentIndex].u[i] + "=" + this.data[currentIndex].data[i][j].d + ";   ";

                if (min === null || this.data[currentIndex].data[i][j].d < min) {
                  min = this.data[currentIndex].data[i][j].d;
                  minI = i;
                  minJ = j;
                }
              }
              else {
                this.data[currentIndex].f += this.data[currentIndex].data[i][j].x *
this.data[currentIndex].data[i][j].c
              }
            }
          }
          if (min <= 0) {
            this.data[currentIndex].description = this.data[currentIndex].description + "<br>   ПЛАН НЕ
ОПТИМАЛЬНИЙ!!!  Вводим в базис клітинку [" + (minI + 1) + "," + (minJ + 1) + "]  ";
            this.base(currentIndex, minI, minJ);
          } else {
            this.data[currentIndex].description = this.data[currentIndex].description + "<br>   ПЛАН
ОПТИМАЛЬНИЙ";
          }
        }
        else {
          console.error('error');
        }
      },
```

```
base: function (currentIndex, startI, startJ) {
    let tempCells = [{
            'data': this.data[currentIndex].data[startI][startJ],
            'i': startI,
            'j': startJ
        }],
        cells = this.baseHorizontal(currentIndex, tempCells, startI, startJ, startI, startJ);
    this.reBase(currentIndex, cells);
    return null;
},
baseHorizontal: function (currentIndex, cells, currentI, currentJ, startI, startJ) {
    for (let j = 0; j < this.m; j += 1) {
        if (this.data[currentIndex].data[currentI][j].x !== null && j !== currentJ) {
            let temp = {'data': this.data[currentIndex].data[currentI][j], 'i': currentI, 'j': j},
                tempCells = this.baseVertical(currentIndex, cells.concat([temp]), currentI, j, startI, startJ);

            if (tempCells.length) {
                return tempCells;
            }
        }
    }
    return [];
},
baseVertical: function (currentIndex, cells, currentI, currentJ, startI, startJ) {
    if (currentJ === startJ) {
        return cells;
    }
    for (let i = 0; i < this.n; i += 1) {
        if (this.data[currentIndex].data[i][currentJ].x !== null && i !== currentI) {
            let temp = {'data': this.data[currentIndex].data[i][currentJ], 'i': i, 'j': currentJ},
                tempCells = this.baseHorizontal(currentIndex, cells.concat([temp]), i, currentJ, startI, startJ);
            if (tempCells.length) {
                return tempCells;
            }
        }
    }
    return [];
},
reBase: function (currentIndex, cells) {
```

```
let state = JSON.parse(JSON.stringify(this.state));
state.countFullX = this.data[currentIndex].countFullX;
state.a = Object.assign([], this.data[0].a);
state.b = Object.assign([], this.data[0].b);
state.u = Object.assign([], this.data[0].u);
state.v = Object.assign([], this.data[0].v);
state.data = JSON.parse(JSON.stringify(this.data[currentIndex].data));
for (let i = 0; i < this.n; i += 1) {
  for (let j = 0; j < this.m; j += 1) {
    state.data[i][j].d = null;
  }
}
this.data.push(state);
currentIndex += 1;
this.data[currentIndex].title = "Вводимо в базис клітинку";
this.data[currentIndex].teor = "Вводимо клітинку в базис та ставимо значення X=0. Будуємо цикл по базисних клітинках починаючи з клітинки, яку " +
  "ми ввели в базис. Потім  по черзі додаємо і віднімаємо, значення яке вираховується" +
  " мінімальне  значення серед клітинок, від яких потрібно віднімати.";
for (let k = 0; k < cells.length; k += 1) {
  this.data[currentIndex].description = this.data[currentIndex].description + "X<sub>" + (cells[k].i + 1) + "
" + (cells[k].j + 1) + "</sub> ";
  if (k !== cells.length - 1) {
    this.data[currentIndex].description = this.data[currentIndex].description + "->";
  }
}
this.data[currentIndex].description = this.data[currentIndex].description + "рахуємо значення, яке будемо віднімати min{";
this.data[currentIndex].data[cells[0].i][cells[0].j].x = 0;
let min = null;
for (let k = 1; k < cells.length; k += 2) {
  this.data[currentIndex].description = this.data[currentIndex].description + cells[k].data.x;
  if (k !== cells.length - 1) {
    this.data[currentIndex].description = this.data[currentIndex].description + ",";
  }
  if (min === null || cells[k].data.x < min) {
    min = cells[k].data.x;
  }
}
this.data[currentIndex].description = this.data[currentIndex].description + "}=" + min + ". Записуємо нові
```

```
значення X:  ";
            for (let k = 0; k < cells.length; k += 1) {
              if (k % 2 === 0) {
                this.data[currentIndex].description = this.data[currentIndex].description + "X<sub>" + (cells[k].i + 1) +
                  " " + (cells[k].j + 1) + "</sub>=" + this.data[currentIndex].data[cells[k].i][cells[k].j].x + "+" + min +
"=" + (this.data[currentIndex].data[cells[k].i][cells[k].j].x += min) + ";  ";
              } else {
                this.data[currentIndex].description = this.data[currentIndex].description + "X <sub>" + (cells[k].i + 1)
+
                  " " + (cells[k].j + 1) + "</sub>=" + this.data[currentIndex].data[cells[k].i][cells[k].j].x + "-" + min +
"=" + (this.data[currentIndex].data[cells[k].i][cells[k].j].x -= min) + ";  ";
                if (this.data[currentIndex].data[cells[k].i][cells[k].j].x === 0) {
                  this.data[currentIndex].data[cells[k].i][cells[k].j].x = null;
                }
              }
            }
            this.data[currentIndex].description = this.data[currentIndex].description + "Виводимо з базису клітинку,
де X=0. Знову перевіряємо чи план оптимальний.";
            this.optimalPlan();
          }
        },
        watch: {
          n: function () {
            let lengthA = this.data[0].a.length;
            if (this.n > lengthA) {
              for (let i = lengthA; i < this.n; i += 1) {
                this.data[0].a.push(null);
                this.data[0].u.push(null);
                this.data[0].data.push([]);
                for (let j = 0; j < this.m; j += 1) {
                  this.data[0].data[i].push(Object.assign({}, this.item));
                }
              }
            }
            else {
              this.data[0].a.splice(this.n, lengthA - this.n);
              this.data[0].data.splice(this.n, lengthA - this.n);
              this.data[0].v.splice(this.n, lengthA - this.n);
            }
          },
```

```
        m: function () {
            let lengthB = this.data[0].b.length;
            if (this.m > lengthB) {
                for (let j = 0; j < this.m - lengthB; j += 1) {
                    this.data[0].b.push(null);
                    this.data[0].v.push(null);
                }
                for (let i = 0; i < this.n; i += 1) {
                    for (let j = 0; j < this.m - lengthB; j += 1) {
                        this.data[0].data[i].push(Object.assign({}, this.item));
                    }
                }
            }
            else {
                this.data[0].b.splice(this.m, lengthB - this.m);
                this.data[0].v.splice(this.m, lengthB - this.m);
                for (let i = 0; i < this.n; i += 1) {
                    this.data[0].data[i].splice(this.m, lengthB - this.m);
                }
            }
        }
    }
</script>
<style> * {
        margin: 0;
        padding: 0;
    }   table input {
        max-width: 50px;
    }
    .has-error {
        color: red;
    }
    .cell {
        position: relative;
    }
    .baz {
        background-color: #9fcdff;
    }
```

```
.onecheck:after {
    position: absolute;
    top: 0;
    right: 25px;
    content: "\2713";
}
.twocheck:before {
    position: absolute;
    top: 0;
    right: 20px;
    content: "\2713";
}
.wrapper {
    width: 100%;
    max-width: 1440px;
    padding: 15px;
    box-sizing: border-box;
}
.btn.badge-pill{
    padding:5px 50px;
}</style>
```