Національний авіаційний університет

Факультет кібербезпеки, комп’ютерної та програмної інженерії

**ЗВІТ**

**по лабораторній роботі No 2**

**Транспортна задача. Метод потенціалів.**

Дисципліна: «Методи оптимізацій та дослідження операцій»

Кафедра: прикладної математики

ОС: бакалавр

Спеціальність: 113 «Прикладна математика»

ОПП: «Прикладне програмне забезпечення»

Виконав: здобувач вищої освіти 4 курсу. 451 групи

Архіпов Олексій Тімурович

Перевірив: Хребет Валерій Григорович

Київ 2023

**Тема:** Транспортна задача. Метод потенціалів.

**Мета:** Ознайомитися з методами розв’язання транспортної задачі лінійного програмування: алгоритмами побудови опорного плану, визначення оптимального плану методом потенціалів.

**Завдання:**

1. Вивчити методи визначення опорного плану і можливості його поліпшення методом потенціалів.

2 Побудувати модель транспортної задачі (включно з транспортною таблицею)

3. В залежності від варіанту складіть програму для знаходження оптимального розв’язку транспортної задачі

**Порядок обчислень**

Метод потенціалів є ефективним ітераційним алгоритмом для розв'язання транспортної задачі. Давайте детальніше розглянемо кожен з етапів, щоб отримати більше інформації:

**Побудова початкового опорного плану методом північно-західного кута:**

1. **Обчислення об'ємів перевезень:**
   * Визначення, яким чином будуть розподілені товари від постачальників до споживачів. Метод північно-західного кута визначає початковий розподіл, починаючи з верхнього лівого кута таблиці і переміщаючись по спіралі.
2. **Врахування можливих варіантів:**
   * Кожна комірка може повністю задовольняти потреби споживача, вичерпувати запас постачальника, або обидва. Метод обирає один з цих варіантів відповідно до логіки оптимізації.
3. **Побудова опорного плану:**
   * Враховуючи умови задачі та вибрані варіанти, будується початковий опорний план.

**Визначення оптимальності отриманого плану:**

1. **Задання потенціалів:**
   * Кожному рядку та стовпцю присвоюються числа, які називаються потенціалами. Ці потенціали визначаються шляхом вирішення системи лінійних рівнянь.
2. **Вирішення системи лінійних рівнянь:**
   * Знайдення значень потенціалів, які відповідають умовам оптимальності. Це може виконуватися за допомогою методів лінійної алгебри.
3. **Перевірка оптимальності:**
   * Перевірка, чи виконуються умови оптимальності для всіх комірок таблиці. Якщо так, то отриманий план є оптимальним. В іншому випадку, переходимо до наступного етапу.

**Поліпшення отриманого плану:**

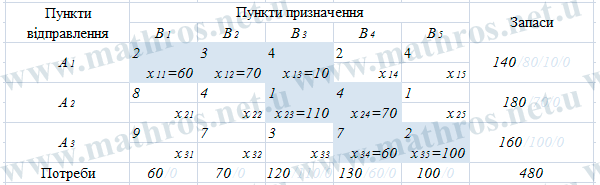
1. **Визначення чисел для вільних комірок:**
   * Визначення чисел, які впливають на вартість перевезення від вільних постачальників до вільних споживачів.
2. **Будівля циклу перерахунку:**
   * Обрання вільної комірки з максимальним від'ємним числом та побудова циклу перерахунку. Це визначає шлях для змін об'ємів перевезень.
3. **Зміна об'ємів перевезень:**
   * Зміна об'ємів перевезень вздовж циклу відповідно до визначених правил.
4. **Повторення обчислень та перевірка оптимальності:**
   * Після внесення змін перевіряється оптимальність плану. Якщо умова не виконується, процес повторюється знову.

Цей ітераційний процес продовжується, доки не буде досягнуто оптимального плану, що задовольняє умови транспортної задачі. Метод потенціалів є ефективним інструментом для оптимізації розподілу ресурсів у транспортних системах.

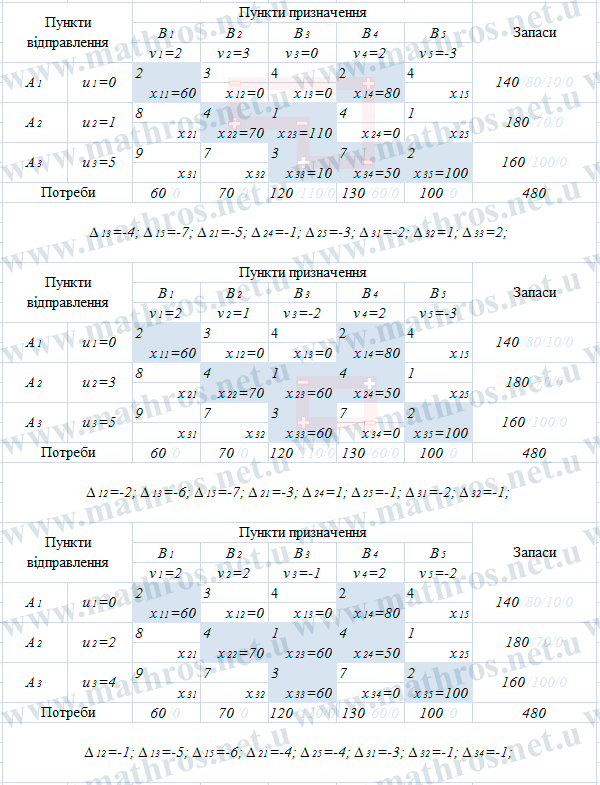
**Результати тестування**

Розглянемо результати тестування на прикладах взятих із джерел інтернету.

Приклад 1:



Результат:



https://www.mathros.net.ua/wp-content/uploads/2012/07/method_potential41.gif

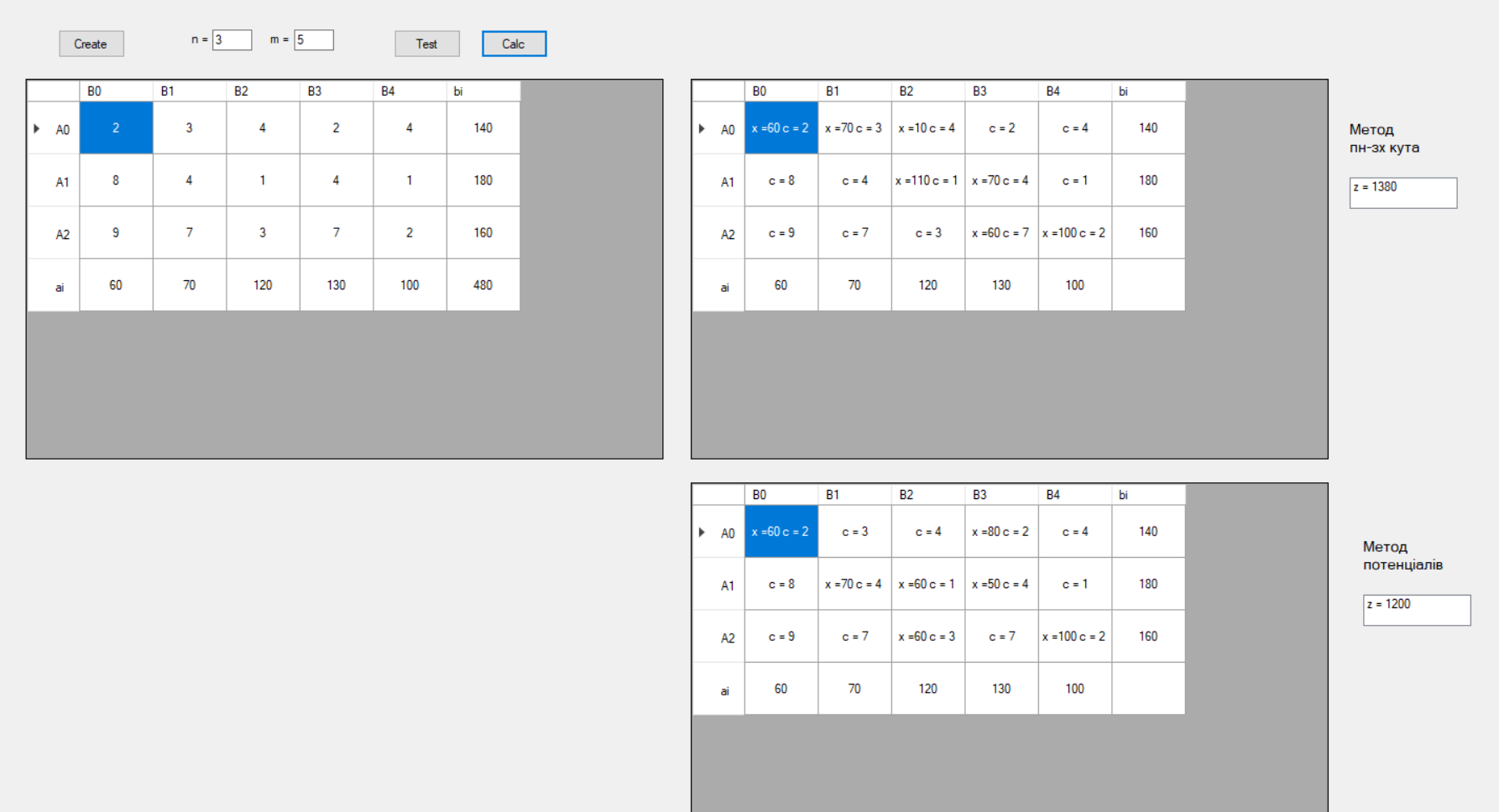
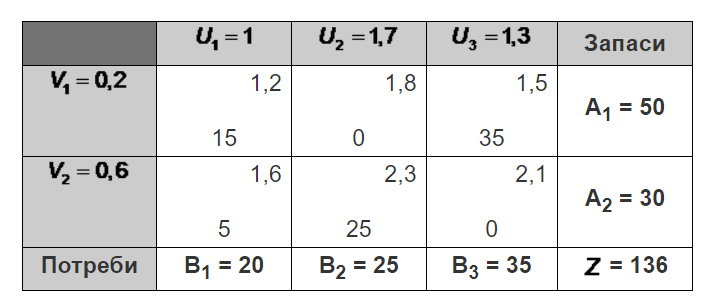


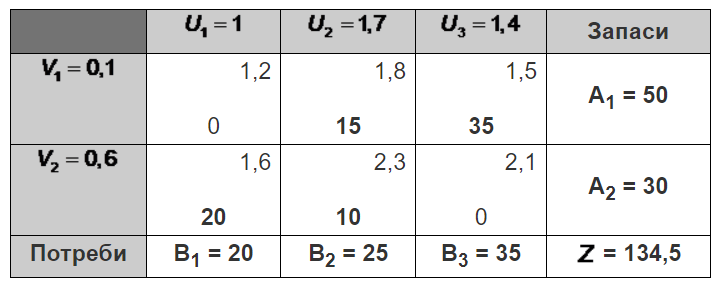
Рисунок 1. Приклад 1

Як видно з рисунку 1 значення співпадають.

Приклад 2:



Результат:



https://elib.lntu.edu.ua/sites/default/files/elib_upload/%D0%95%D0%9D%D0%9F_%D0%9C%D0%9C%D0%9E%D0%95%D0%A1_19/page20.files/image056.gif

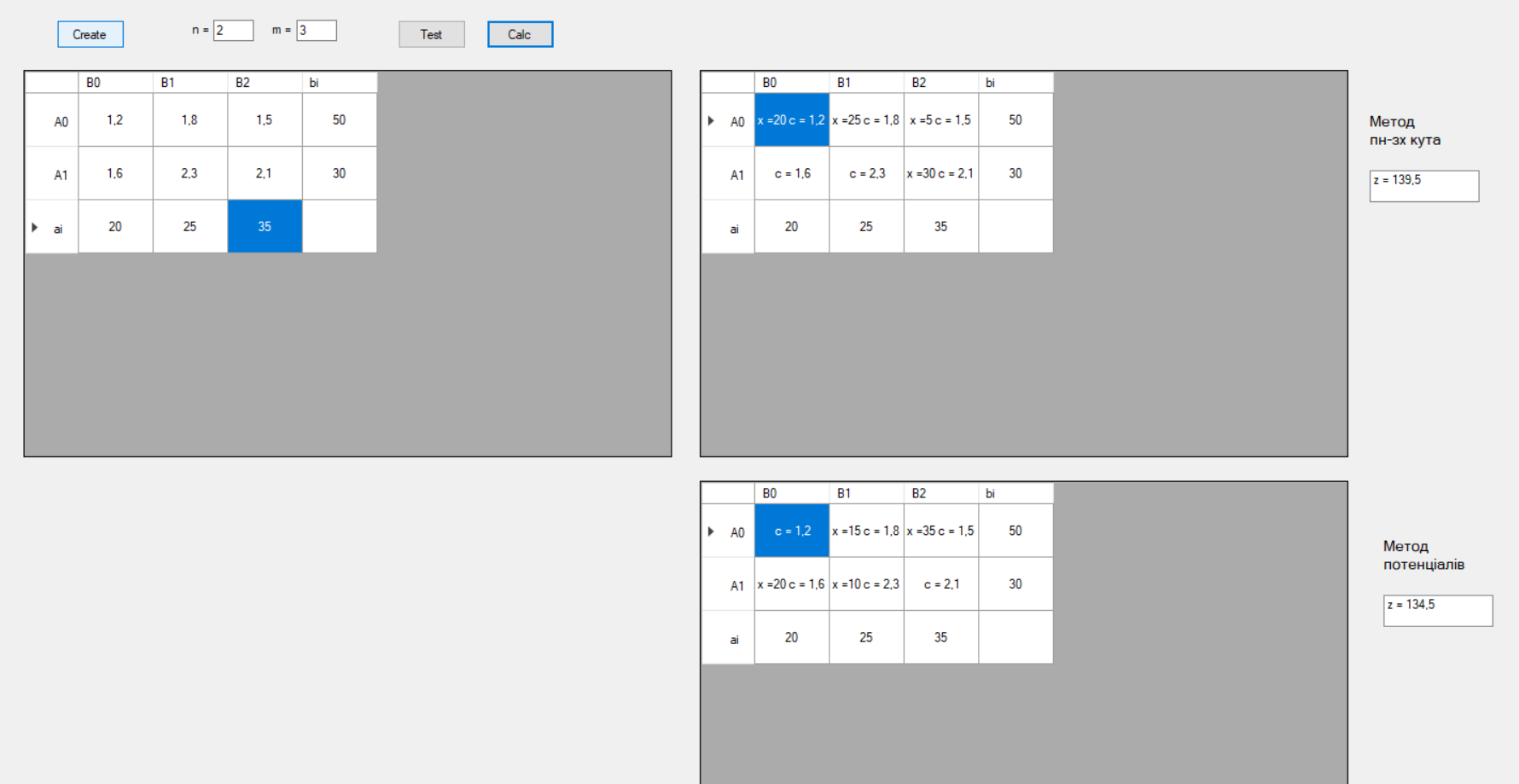
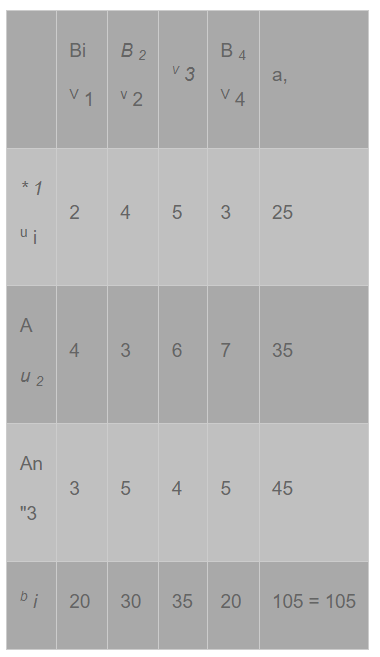


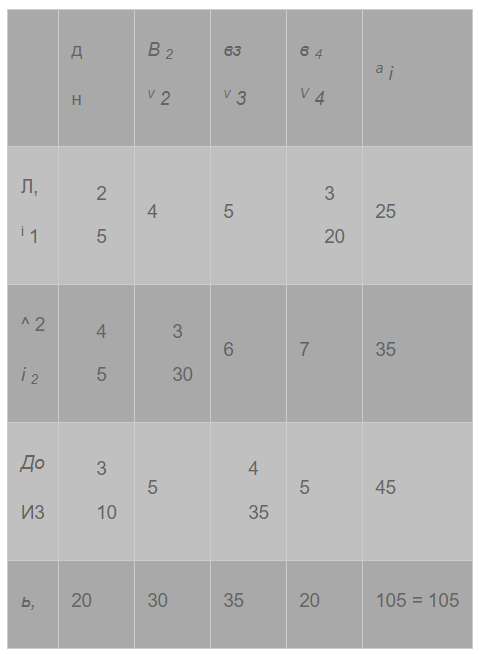
Рисунок 2. Приклад 2

Як видно з рисунку 2 значення співпадають.

Приклад 3:



Результат:



Z = 350

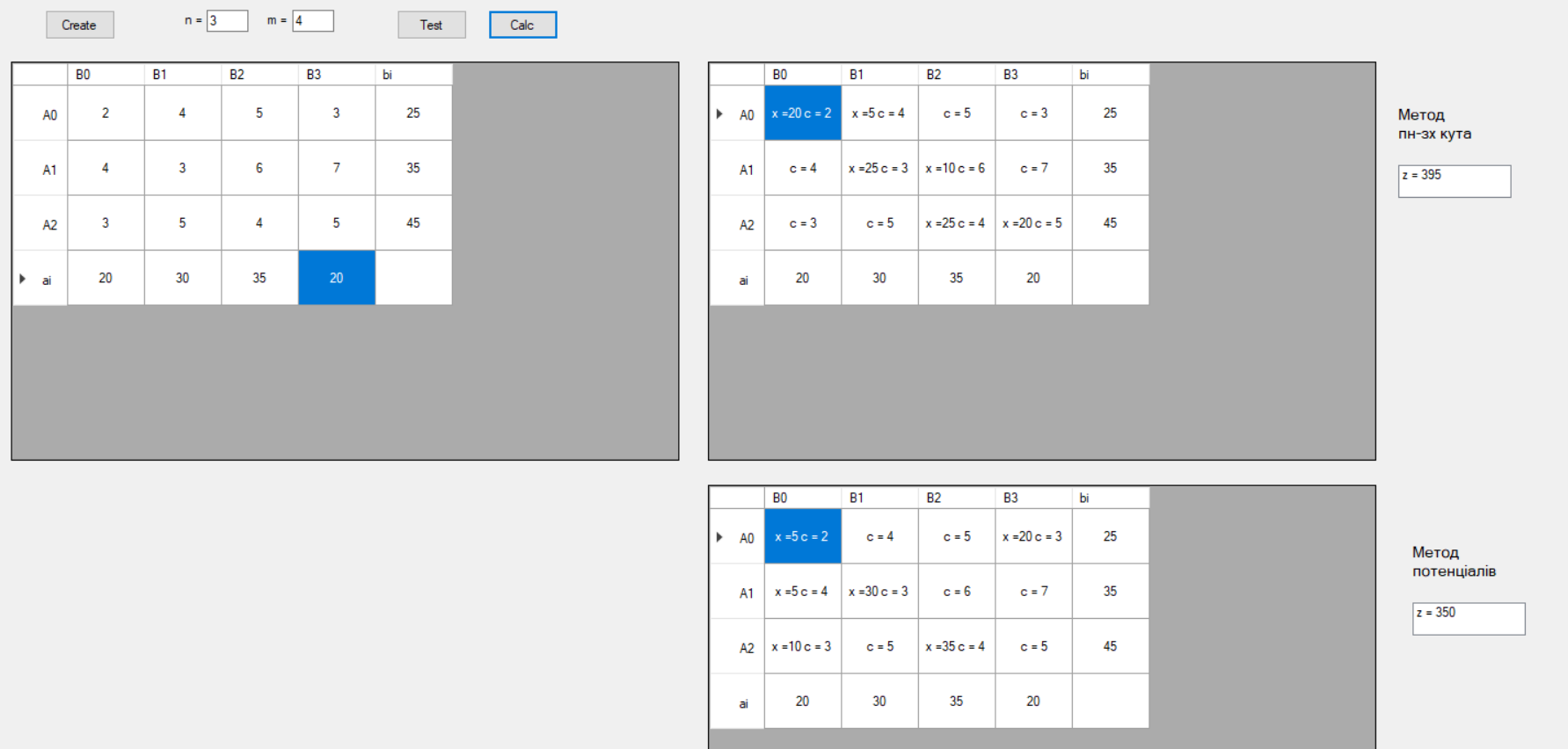


Рисунок 3. Приклад 3

Як видно з рисунку 3 значення співпадають.

**Висновоки**

1. Вивчив методи визначення опорного плану і можливості його поліпшення методом потенціалів.
2. Побудував модель транспортної задачі (включно з транспортною таблицею).
3. В залежності від варіанту склав програму для знаходження оптимального розв’язку транспортної задачі.

**Код програми**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Drawing.Drawing2D;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace lab2

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

//dg.RowHeadersVisible = false;

dg.AllowUserToAddRows = false;

dg.RowTemplate.Height = 50;

dg2.AllowUserToAddRows = false;

dg2.RowTemplate.Height = 50;

dg3.AllowUserToAddRows = false;

dg3.RowTemplate.Height = 50;

}

List<List<double>> C;

List<List<double>> P;

List<List<double>> P\_kuta;

List<double> A;

List<double> B;

int n = 0, m = 0;

int e = 0;

bool result = false;

void Calc()

{

ReadAll();

Calc\_opornui\_plan();//рахуємо опорний план за методом пн-зх кута

Method\_Potentialiv();

}

void Create\_C\_A\_B()

{

n = Convert.ToInt32(tbn.Text);

m = Convert.ToInt32(tbm.Text);

dg.Rows.Clear();

dg.Columns.Clear();

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

{

dg.Columns.Add("B" + i.ToString(), "B" + i.ToString());

dg.Columns[i].Width = 70;

dg.Columns[i].DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;

}

dg.Columns.Add("bi", "bi");

dg.Columns[dg.Columns.Count - 1].Width = 70;

dg.Columns[dg.Columns.Count - 1].DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;

dg.Rows.Add(n + 1);

for (int i = 0; i < dg.Rows.Count; i++)

{

if(i < dg.Rows.Count - 1) dg.Rows[i].HeaderCell.Value = "A" + (i);

else dg.Rows[i].HeaderCell.Value = "ai";

}

for (int i = 0; i < n + 1; i++)

{

for (int j = 0; j < m + 1; j++)

{

if(i == n && j == m) dg.Rows[i].Cells[j].Value = "";

else dg.Rows[i].Cells[j].Value = "1";

}

}

}

void CreateMatrixTest()

{

n = 3;

m = 5;

dg.Rows.Clear();

dg.Columns.Clear();

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

{

dg.Columns.Add("B" + i.ToString(), "B" + i.ToString());

dg.Columns[i].Width = 70;

dg.Columns[i].DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;

}

dg.Columns.Add("bi", "bi");

dg.Columns[dg.Columns.Count - 1].Width = 70;

dg.Columns[dg.Columns.Count - 1].DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;

dg.Rows.Add(n + 1);

for (int i = 0; i < dg.Rows.Count; i++)

{

if (i < dg.Rows.Count - 1) dg.Rows[i].HeaderCell.Value = "A" + (i);

else dg.Rows[i].HeaderCell.Value = "ai";

}

int j = 0;

dg.Rows[j].Cells[0].Value = "2";

dg.Rows[j].Cells[1].Value = "3";

dg.Rows[j].Cells[2].Value = "4";

dg.Rows[j].Cells[3].Value = "2";

dg.Rows[j].Cells[4].Value = "4";

dg.Rows[j].Cells[5].Value = "140";

j = 1;

dg.Rows[j].Cells[0].Value = "8";

dg.Rows[j].Cells[1].Value = "4";

dg.Rows[j].Cells[2].Value = "1";

dg.Rows[j].Cells[3].Value = "4";

dg.Rows[j].Cells[4].Value = "1";

dg.Rows[j].Cells[5].Value = "180";

j = 2;

dg.Rows[j].Cells[0].Value = "9";

dg.Rows[j].Cells[1].Value = "7";

dg.Rows[j].Cells[2].Value = "3";

dg.Rows[j].Cells[3].Value = "7";

dg.Rows[j].Cells[4].Value = "2";

dg.Rows[j].Cells[5].Value = "160";

j = 3;

dg.Rows[j].Cells[0].Value = "60";

dg.Rows[j].Cells[1].Value = "70";

dg.Rows[j].Cells[2].Value = "120";

dg.Rows[j].Cells[3].Value = "130";

dg.Rows[j].Cells[4].Value = "100";

dg.Rows[j].Cells[5].Value = "480";

}

void ReadAll()

{

n = Convert.ToInt32(tbn.Text);

m = Convert.ToInt32(tbm.Text);

C = new List<List<double>>();

A = new List<double>();

B = new List<double>();

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

{

List<double> temp = new List<double>();

for (int j = 0; j < m; j++)

{

temp.Add(Convert.ToDouble(dg.Rows[i].Cells[j].Value));

}

C.Add(temp);

A.Add(Convert.ToDouble(dg.Rows[i].Cells[m].Value));

}

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

{

B.Add(Convert.ToDouble(dg.Rows[n].Cells[i].Value));

}

}

void Calc\_opornui\_plan()

{

double sum1 = A.Sum();

double sum2 = B.Sum();

if(sum1 > sum2)

{

B.Add(sum1 - sum2);

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

{

C[i].Add(0);

}

}

else if(sum1 < sum2)

{

A.Add(sum2 - sum1);

List<double> temp = new List<double>();

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

{

temp.Add(0);

}

C.Add(temp);

}

double[,] Plan = new double[C.Count, C[0].Count];

int k = 0; int j = 0;

List<double> A1 = new List<double>();

List<double> B1 = new List<double>();

for (int i = 0; i < A.Count(); i++)

{

A1.Add(A[i]);

}

for (int i = 0; i < B.Count(); i++)

{

B1.Add(B[i]);

}

do

{

do

{

if (A1[k] >= B1[j])

{

Plan[k, j] = B1[j];

B1[j] = 0;

A1[k] -= Plan[k, j];

j++;

}

else

{

Plan[k, j] = A1[k];

B1[j] -= A1[k];

A1[k] -= Plan[k, j];

k++;

}

}while (A1[k] != 0);

} while (A1.Sum() > 0 && B1.Sum() > 0);

P = new List<List<double>>();

P\_kuta = new List<List<double>>();

for (int i = 0; i < C.Count; i++)

{

List<double> temp = new List<double>();

for (j = 0; j < C[i].Count; j++)

{

temp.Add(Plan[i, j]);

}

P.Add(temp);

}

int c = 0;

for(int i = 0; i < P.Count; i++)

{

for(int t = 0; t < P[0].Count; t++)

{

if (P[i][t] == 0)

{

P[i][t] = -1;

}

else c++;

}

}

if(c < A.Count + B.Count - 1)

{

P = Add\_zero(P, A.Count + B.Count - 1 - c);

}

for(int i = 0; i < P.Count; i++)

{

List<double> temp = new List<double>();

for(int v = 0; v < P[0].Count; v++)

{

temp.Add(P[i][v]);

}

P\_kuta.Add(temp);

}

}

List<Tuple<int, int>> indexAll = new List<Tuple<int, int>>();

List<List<double>> Add\_zero(List<List<double>> P, int n)

{

//List<Tuple<int, int>> index = new List<Tuple<int, int>>();

if (indexAll.Count == 0)

{

for (int i = 0; i < P.Count; i++)

{

for (int t = 0; t < P[0].Count; t++)

{

if (P[i][t] == -1)

{

Tuple<int, int> t1 = new Tuple<int, int>(i, t);

//index.Add(t1);

indexAll.Add(t1);

}

}

}

}

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

{

Random r = new Random();

int j = r.Next(0, indexAll.Count);

P[indexAll[j].Item1][indexAll[j].Item2] = 0;

indexAll.RemoveAt(j);

}

return P;

}

void Method\_Potentialiv()

{

e++;

double[] U = new double[P.Count];

double[] V = new double[P[0].Count];

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

{

U[i] = -999;

}

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

{

V[i] = -999;

}

List<double[]> list = Get\_potencials(P, C, V, U);//знаходимо потенціали u, v

U = list[0];

V = list[1];

List<Tuple<double, int, int>> D = Get\_delta(P, C, V, U);// знаходимо дельта

Tuple<double, int, int> maxTuple = new Tuple<double, int, int>(-100000,-100000,-10000);

if (D.Count > 0)

{

maxTuple = D.OrderByDescending(t => t.Item1).First();//шукаємо максимльне дельта

}

if (maxTuple.Item1 > 0 && result == false && e < 600)

{

P = ModifyPlan(P, maxTuple.Item2, maxTuple.Item3);

Method\_Potentialiv();

}

else if (result == false || e == 600)

{

double sum = 0;

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

{

for(int j = 0; j < m; j++)

{

if (P[i][j] != -1)

{

sum += P[i][j] \* C[i][j];

}

}

}

dg3.Rows.Clear();

dg3.Columns.Clear();

for (int i = 0; i < P[0].Count; i++)

{

dg3.Columns.Add("B" + i.ToString(), "B" + i.ToString());

dg3.Columns[i].Width = 70;

dg3.Columns[i].DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;

}

dg3.Columns.Add("bi", "bi");

dg3.Columns[dg3.Columns.Count - 1].Width = 70;

dg3.Columns[dg3.Columns.Count - 1].DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;

dg3.Rows.Add(P.Count + 1);

for (int i = 0; i < dg3.Rows.Count; i++)

{

if (i < dg3.Rows.Count - 1) dg3.Rows[i].HeaderCell.Value = "A" + (i);

else dg3.Rows[i].HeaderCell.Value = "ai";

}

for (int i = 0; i < P.Count; i++)

{

for (int j = 0; j < P[i].Count; j++)

{

if (P[i][j] != -1)

{

dg3.Rows[i].Cells[j].Value = "x =" + P[i][j] + " c = " + C[i][j];

}

else dg3.Rows[i].Cells[j].Value = " c = " + C[i][j];

}

}

for(int i = 0; i < A.Count; i++)

{

dg3.Rows[i].Cells[P[0].Count].Value = A[i];

}

for (int i = 0; i < B.Count; i++)

{

dg3.Rows[P.Count].Cells[i].Value = B[i];

}

listBox1.Items.Add("z = " + sum);

//------------------------------------------

sum = 0;

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

{

for (int j = 0; j < m; j++)

{

if (P\_kuta[i][j] != -1)

{

sum += P\_kuta[i][j] \* C[i][j];

}

}

}

dg2.Rows.Clear();

dg2.Columns.Clear();

for (int i = 0; i < P\_kuta[0].Count; i++)

{

dg2.Columns.Add("B" + i.ToString(), "B" + i.ToString());

dg2.Columns[i].Width = 70;

dg2.Columns[i].DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;

}

dg2.Columns.Add("bi", "bi");

dg2.Columns[dg2.Columns.Count - 1].Width = 70;

dg2.Columns[dg2.Columns.Count - 1].DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;

dg2.Rows.Add(P\_kuta.Count + 1);

for (int i = 0; i < dg2.Rows.Count; i++)

{

if (i < dg2.Rows.Count - 1) dg2.Rows[i].HeaderCell.Value = "A" + (i);

else dg2.Rows[i].HeaderCell.Value = "ai";

}

for (int i = 0; i < P\_kuta.Count; i++)

{

for (int j = 0; j < P\_kuta[i].Count; j++)

{

if (P\_kuta[i][j] != -1)

{

dg2.Rows[i].Cells[j].Value = "x =" + P\_kuta[i][j] + " c = " + C[i][j];

}

else dg2.Rows[i].Cells[j].Value = " c = " + C[i][j];

}

}

for (int i = 0; i < A.Count; i++)

{

dg2.Rows[i].Cells[P\_kuta[0].Count].Value = A[i];

}

for (int i = 0; i < B.Count; i++)

{

dg2.Rows[P\_kuta.Count].Cells[i].Value = B[i];

}

listBox2.Items.Add("z = " + sum);

result = true;

}

}

List<double[]> Get\_potencials(List<List<double>> A, List<List<double>> B, double[] V, double[] U)

{

U[0] = 0;

int k = 0;

do

{

if (result == true) break;

for (int i = 0; i < A.Count; i++)

{

for (int j = 0; j < A[0].Count; j++)

{

if (A[i][j] != -1)

{

if (U[i] != -999 && V[j] == -999)

{

V[j] = B[i][j] - U[i];

}

if (U[i] == -999 && V[j] != -999)

{

U[i] = B[i][j] - V[j];

}

}

}

}

int c = 0;

for(int i = 0; i < U.Length; i++)

{

if (U[i] == -999) c++;

}

for (int i = 0; i < V.Length; i++)

{

if (V[i] == -999) c++;

}

if (c == 0) break;

k++;

if(k == 40)

{

Calc();

break;

}

} while (true);

List<double[]> list = new List<double[]>();

list.Add(U); list.Add(V);

return list;

}

List<Tuple<double, int, int>> Get\_delta(List<List<double>> A, List<List<double>> B, double[] V, double[] U)

{

List<Tuple<double, int, int>> D = new List<Tuple<double, int, int>>();//1 - номер дельта, 2 - номер строки, 3 - номер рядка

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

{

if (result == true) break;

for(int j = 0; j < m; j++)

{

if (A[i][j] == -1)

{

double t = U[i] + V[j] - B[i][j];

Tuple<double, int, int> d = new Tuple<double, int, int>(t, i, j);

D.Add(d);

}

}

}

return D;

}

List<List<double>> ModifyPlan(List<List<double>> P, int row, int col)

{

// Знайдіть шлях змін в опорному плані

List<Tuple<int, int>> changePath = FindChangePath(P, row, col);

List<double> change = new List<double>();

for(int i = 1; i < changePath.Count; i += 2)

{

change.Add(P[changePath[i].Item1][changePath[i].Item2]);

}

// Змініть опорний план на шляху

for (int i = 0; i < changePath.Count; i++)

{

int r = changePath[i].Item1;

int c = changePath[i].Item2;

if (i % 2 == 0)

{

//if(b == 1)

//{

// // Зменште значення на шляху

// if (i == 0) P[r][c] += change.Min() + 1;

// else P[r][c] += change.Min();

//}

//else P[r][c] += change.Min();

if (i == 0)

{

P[r][c] += change.Min() + 1;

}

else P[r][c] += change.Min();

}

else

{

// Збільште значення на шляху

if(P[r][c] == change.Min()) P[r][c] -= change.Min() + 1;

else P[r][c] -= change.Min();

}

}

return P;

}

List<Tuple<int, int>> FindChangePath(List<List<double>> P, int startRow, int startCol)

{

List<Tuple<int, int>> path = new List<Tuple<int, int>>();

int row = startRow;

int col = startCol;

List<List<double>> P\_old = new List<List<double>>();

for (int i = 0; i < P.Count; i++)

{

List<double> p = new List<double>();

for(int j = 0; j < P[0].Count; j++)

{

p.Add(P[i][j]);

}

P\_old.Add(p);

}

P\_old[row][col] = 1;

List<Tuple<int, int>> index = new List<Tuple<int, int>>();

for(int i = 0; i < P\_old[0].Count(); i++)

{

for(int j = 0; j < P\_old.Count; j++)

{

if (P\_old[j][i] != -1)

{

Tuple<int, int> t = new Tuple<int, int>(j, i);

index.Add(t);

}

}

if (index.Count == 1) P\_old[index[0].Item1][index[0].Item2] = -1;

index.Clear();

}

for (int i = 0; i < P\_old.Count(); i++)

{

for (int j = 0; j < P\_old[0].Count; j++)

{

if (P\_old[i][j] != -1)

{

Tuple<int, int> t = new Tuple<int, int>(i, j);

index.Add(t);

}

}

if (index.Count == 1) P\_old[index[0].Item1][index[0].Item2] = -1;

index.Clear();

}

for (int i = 0; i < P\_old.Count(); i++)

{

for (int j = 0; j < P\_old[0].Count; j++)

{

if (P\_old[i][j] != -1)

{

Tuple<int, int> t = new Tuple<int, int>(i, j);

index.Add(t);

}

}

}

int r = 0, c = 0;

int count = index.Count;

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

{

if (result == true) break;

if (i == 0)

{

Tuple<int, int> t1 = new Tuple<int, int>(row, col);

int j = index.IndexOf(t1);

if (j == -1)

{

Calc();

break;

}

path.Add(index[j]);

index.RemoveAt(j);

r = row;

c = col;

}

else

{

if(i % 2 != 0)

{

for(int k = 0; k < index.Count; k++)

{

if (index[k].Item1 == r && index[k].Item2 != c)

{

path.Add(index[k]);

r = index[k].Item1;

c = index[k].Item2;

index.RemoveAt(k);

break;

}

}

}

else

{

for (int k = 0; k < index.Count; k++)

{

if (index[k].Item1 != r && index[k].Item2 == c)

{

path.Add(index[k]);

r = index[k].Item1;

c = index[k].Item2;

index.RemoveAt(k);

break;

}

}

}

}

}

return path;

}

private void btnCreate\_Click(object sender, EventArgs e)

{

Create\_C\_A\_B();

}

private void btnTest\_Click(object sender, EventArgs e)

{

CreateMatrixTest();

}

private void btnCalc\_Click(object sender, EventArgs e)

{

listBox1.Items.Clear();

listBox2.Items.Clear();

Calc();

indexAll.Clear();

result = false;

}

}

}