МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

**«Вятский государственный университет»**

**(ФГБОУ ВО «ВятГУ»)**

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

**Лабораторная работа №6**

**Решение транспортной задачи**

Выполнил студент группы ИВТ-31 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Кудяшев Я.Ю./

Проверил преподаватель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Коржавина А.С./

Киров 2021

1. Цель работы: написать программу для решения транспортной задачи
2. Экранные формы

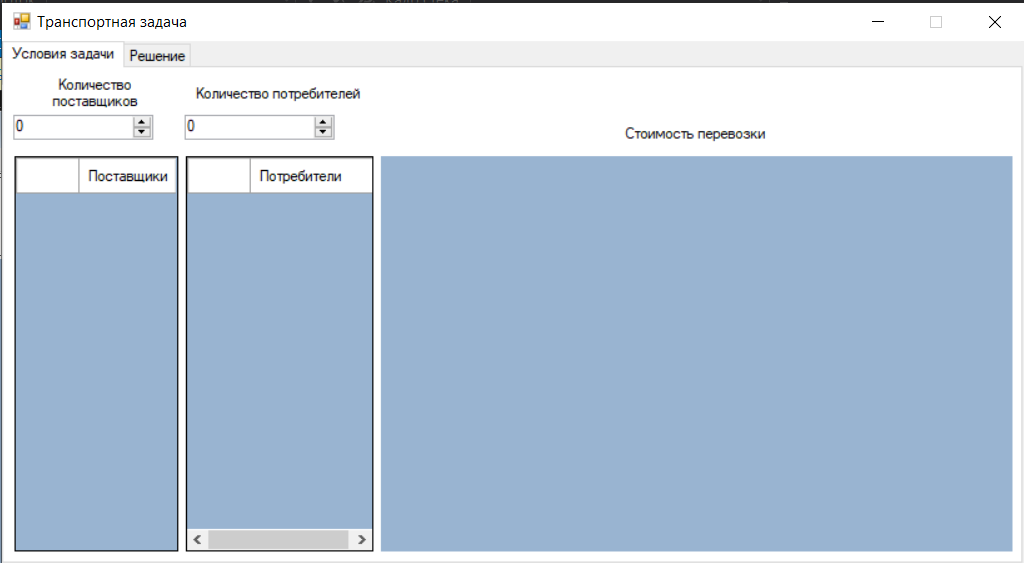


Рисунок 1 – Пример работы программы (1)

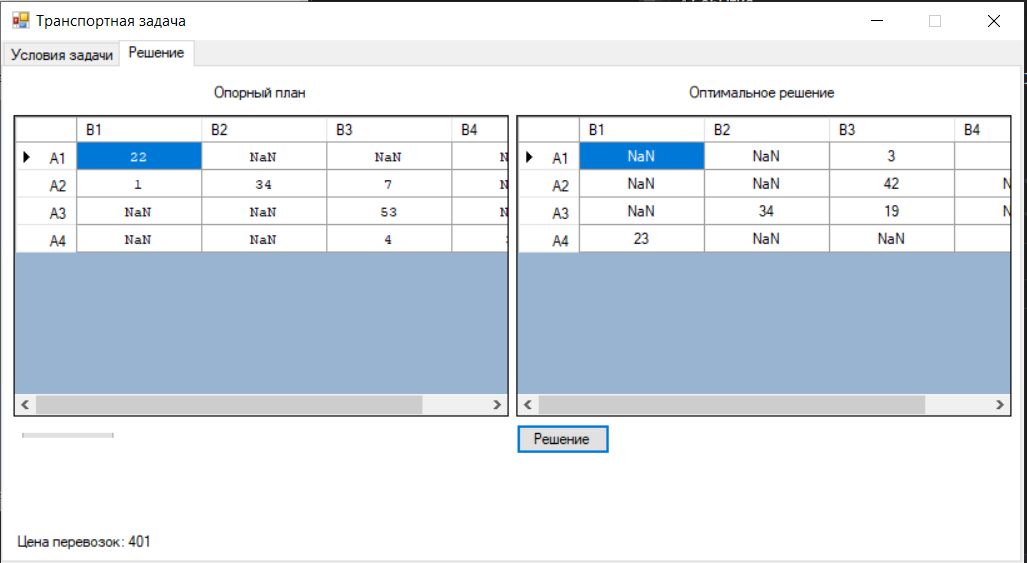


Рисунок 2 – Пример работы программы (2)

1. Программный код

using System;

using System.IO;

using System.Windows.Forms;

using System.Drawing;

namespace lab1

{

public partial class mainForm : Form

{

TransportProblem TP = null;

float[,] SupportPlan = null;

float[,] Optimum;

public mainForm( )

{

InitializeComponent();

}

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

{

Stream myStream = null;

OpenFileDialog openFileDialog1 = new OpenFileDialog();

openFileDialog1.InitialDirectory = "D:\\";

openFileDialog1.Filter = "txt files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*";

openFileDialog1.FilterIndex = 1;

openFileDialog1.RestoreDirectory = true;

if ( openFileDialog1.ShowDialog() == DialogResult.OK )

{

try

{

if ( ( myStream = openFileDialog1.OpenFile() ) != null )

{

StreamReader SR = new StreamReader( myStream );

String[] Sizes = SR.ReadLine().Split( ' ' );

int Asize = 0, Bsize = 0;

int.TryParse( Sizes[0], out Asize );

int.TryParse( Sizes[1], out Bsize );

String A = SR.ReadLine();

String B = SR.ReadLine();

String[] C = new String[Asize];

for ( int i = 0; i < Asize; i++ ) C[i] = SR.ReadLine();

try

{

TP = new TransportProblem( Asize, Bsize, A, B, C );

}

catch ( Exception exc )

{ MessageBox.Show( exc.Message ); }

}

myStream.Close();

}

catch ( Exception ex )

{

MessageBox.Show( "Error: Could not read file from disk. Original error: " + ex.Message );

}

}

nudA.Value = TP.dilersCount;

nudB.Value = TP.customersCount;

gridA.RowCount = TP.dilersCount;

gridB.RowCount = TP.customersCount;

gridC.ColumnCount = TP.customersCount;

gridC.RowCount = TP.dilersCount;

FillGrids();

}

private void FillGrids( )

{

for ( int i = 0; i < TP.dilersCount; i++ )

{

gridA.Rows[i].Cells[0].Value = TP.dilers[i].ToString();

gridA.Rows[i].HeaderCell.Value = "A" + ( i + 1 ).ToString();

}

for ( int i = 0; i < TP.customersCount; i++ )

{

gridB.Rows[i].Cells[0].Value = TP.customers[i].ToString();

gridB.Rows[i].HeaderCell.Value = "B" + ( i + 1 ).ToString();

}

FillBigGrid( gridC, TP.transportationPrices );

}

private void FillBigGrid( DataGridView grid, float[,] arr )

{

for ( int i = 0; i < TP.dilersCount; i++ )

{

grid.Rows[i].HeaderCell.Value = "A" + ( i + 1 ).ToString();

for ( int j = 0; j < TP.customersCount; j++ )

{

grid.Rows[i].Cells[j].Value = arr[i, j].ToString();

grid.Columns[j].HeaderText = "B" + ( j + 1 ).ToString();

}

}

}

private void validate()

{

float totalProducts = 0;

for ( int i = 0; i < TP.dilersCount; ++i )

{

totalProducts += TP.dilers[i];

}

float totalNeeds = 0;

for ( int i = 0; i < TP.customersCount; ++i )

{

totalNeeds += TP.customers[i];

}

if ( totalProducts != totalNeeds )

{

MessageBox.Show( "Количество товаров не соответствует потреблению" );

/\*float[] tmp = new float[TP.customersCount + 1];

for ( int i = 0; i < TP.customersCount; ++i )

{

tmp[i] = TP.customers[i];

}

tmp[TP.customersCount] = totalProducts - totalNeeds;

TP.customersCount++;

TP.customers = tmp;\*/

}

/\*else if ( totalProducts < totalNeeds )

{

float[] tmp = new float[TP.dilersCount + 1];

for ( int i = 0; i < TP.dilersCount; ++i )

{

tmp[i] = TP.dilers[i];

}

tmp[TP.dilersCount] = totalNeeds - totalProducts;

TP.dilersCount++;

TP.customers = tmp;

}\*/

}

private void nudA\_ValueChanged( object sender, EventArgs e )

{

int rowCount = Convert.ToInt32( nudA.Value );

gridA.RowCount = rowCount;

gridC.RowCount = rowCount;

gridA.Rows[rowCount - 1].HeaderCell.Value = "A" + rowCount.ToString();

gridC.Rows[rowCount - 1].HeaderCell.Value = "A" + rowCount.ToString();

}

private void nudB\_ValueChanged( object sender, EventArgs e )

{

int colCount = Convert.ToInt32( nudB.Value );

gridB.RowCount = colCount;

gridC.ColumnCount = colCount;

gridB.Rows[colCount - 1].HeaderCell.Value = "B" + colCount.ToString();

gridC.Columns[colCount - 1].HeaderText = "B" + colCount.ToString();

}

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

{

fill();

validate();

gridFinal.Visible = false;

label2.Visible = false;

gridSupport.RowCount = TP.dilersCount;

gridFinal.RowCount = TP.dilersCount;

gridSupport.ColumnCount = TP.customersCount;

gridFinal.ColumnCount = TP.customersCount;

// if ( rbNW.Checked )

{

SupportPlan = TP.NordWest();

}

//else if ( rbMinElem.Checked )

//{

// SupportPlan = TP.MinEl();

//}

FillBigGrid( gridSupport, SupportPlan );

Optimum = TP.PotenMeth( SupportPlan );

FillBigGrid( gridFinal, Optimum );

float Sum = 0;

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

{

int j = ( i - i % TP.customersCount ) / TP.customersCount;

int k = i % TP.customersCount;

if ( Optimum[j, k] == Optimum[j, k] )

Sum += Optimum[j, k] \* TP.transportationPrices[j, k];

}

lblOptimum.Text = "Цена перевозок: " + Sum.ToString();

}

private void fill( )

{

TP = new TransportProblem();

TP.dilersCount = gridA.RowCount;

TP.customersCount = gridB.RowCount;

TP.dilers = new float[TP.dilersCount];

TP.customers = new float[TP.customersCount];

TP.transportationPrices = new float[TP.dilersCount, TP.customersCount];

try

{

for ( int i = 0; i < TP.dilersCount; ++i )

{

TP.dilers[i] = Convert.ToInt32( gridA.Rows[i].Cells[0].Value );

}

for ( int i = 0; i < TP.customersCount; ++i )

{

TP.customers[i] = Convert.ToInt32( gridB.Rows[i].Cells[0].Value );

}

for ( int i = 0; i < TP.dilersCount; ++i )

{

for ( int j = 0; j < TP.customersCount; ++j )

{

TP.transportationPrices[i, j] = Convert.ToInt32( gridC.Rows[i].Cells[j].Value );

}

}

}

catch (System.Exception ex)

{

MessageBox.Show( "ex" );

}

}

private void label6\_Click(object sender, EventArgs e)

{

}

private void button1\_Click(object sender, EventArgs e)

{

gridFinal.Visible = true;

label2.Visible = true;

}

private void label5\_Click(object sender, EventArgs e)

{

}

private void gridA\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

}

public class TransportProblem

{

class InvalidInpFormat : ApplicationException

{

public InvalidInpFormat( ) : base() { }

public InvalidInpFormat( string str ) : base( str ) { }

public override string ToString( )

{

return Message;

}

}

// склады

public float[] dilers;

// потребители

public float[] customers;

// Издержки

public float[,] transportationPrices;

public int dilersCount;

public int customersCount;

// Конструкторы

public TransportProblem( float[] nA, float[] nB, float[,] nC )

{

if ( ( nA.Length != nC.GetLength( 0 ) ) || ( nB.Length != nC.GetLength( 1 ) ) )

throw new InvalidInpFormat( "Размеры массива затрат не соответствуют размерам массивов поставщиков и складов" );

this.dilers = nA;

this.customers = nB;

this.transportationPrices = nC;

this.dilersCount = nA.Length;

this.customersCount = nB.Length;

}

public TransportProblem( int \_Asize, int \_Bsize, string sA, string sB, string[] sC )

{

dilersCount = \_Asize; customersCount = \_Bsize;

float x = 0;

string[] StrArr = sA.Split( ' ' );

if ( StrArr.Length != dilersCount )

throw new InvalidInpFormat( "Размеры массива А не соответствуют заявленным" );

dilers = new float[dilersCount];

for ( int i = 0; i < dilers.Length; i++ ) if ( float.TryParse( StrArr[i], out x ) ) dilers[i] = x;

StrArr = sB.Split( ' ' );

if ( StrArr.Length != customersCount )

throw new InvalidInpFormat( "Размеры массива B не соответствуют заявленным" );

customers = new float[customersCount];

for ( int i = 0; i < customers.Length; i++ ) if ( float.TryParse( StrArr[i], out x ) ) customers[i] = x;

float sumA = 0;

Array.ForEach( dilers, delegate( float f ) { sumA += f; } );

float sumB = 0;

Array.ForEach( customers, delegate( float f ) { sumB += f; } );

float dif = sumA - sumB;

if ( dif > 0 )

{

float[] bufArr = customers;

customers = new float[bufArr.Length + 1];

bufArr.CopyTo( customers, 0 );

customers[customers.Length - 1] = Math.Abs( dif );

customersCount++;

}

else if ( dif < 0 )

{

float[] bufArr = dilers;

dilers = new float[bufArr.Length + 1];

bufArr.CopyTo( dilers, 0 );

dilers[dilers.Length - 1] = Math.Abs( dif );

dilersCount++;

}

transportationPrices = new float[dilersCount, customersCount];

for ( int j = 0; j < sC.Length; j++ )

{

StrArr = sC[j].Split( ' ' );

if ( StrArr.Length != \_Bsize )

throw new InvalidInpFormat( "Длина одной из строк входного файла не соответствует длине массива В" );

for ( int i = 0; i < \_Bsize; i++ ) if ( float.TryParse( StrArr[i], out x ) ) transportationPrices[j, i] = x;

}

}

public TransportProblem( )

{

}

// Строим опорные планы тут

bool isEmpty( float[] arr )

{

return Array.TrueForAll( arr, delegate( float x ) { return x == 0; } );

}

private void NanToEmpty( float[,] outArr )

{

int i = 0, j = 0;

for ( i = 0; i < dilersCount; i++ )

for ( j = 0; j < customersCount; j++ )

if ( outArr[i, j] == 0 ) outArr[i, j] = float.NaN;

}

float findMin( float[,] Arr, bool[,] pr, out int indi, out int indj )

{

indi = -1; indj = -1;

float min = float.MaxValue;

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

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

if ( ( pr[i, j] ) && ( Arr[i, j] < min ) )

{

min = Arr[i, j];

indi = i; indj = j;

}

return min;

}

// Метод северо-западного угла

public float[,] NordWest( )

{

float[] Ahelp = dilers;

float[] Bhelp = customers;

int i = 0, j = 0;

float[,] outArr = new float[dilersCount, customersCount];

NanToEmpty( outArr );

while ( !( isEmpty( Ahelp ) && isEmpty( Bhelp ) ) )

{

float Dif = Math.Min( Ahelp[i], Bhelp[j] );

outArr[i, j] = Dif;

Ahelp[i] -= Dif; Bhelp[j] -= Dif;

if ( ( Ahelp[i] == 0 ) && ( Bhelp[j] == 0 ) && ( j + 1 < customersCount ) )

{

outArr[i, j + 1] = 0;

}

if ( Ahelp[i] == 0 )

{

i++;

}

if ( Bhelp[j] == 0 )

{

j++;

}

if ( i >= dilersCount || j >= customersCount )

{

break;

}

}

return outArr;

}

class FindWay

{

FindWay Father;

Point Root;

FindWay[] Childrens;

Point[] mAllowed;

Point Begining;

bool flag;

public FindWay( int x, int y, bool \_flag, Point[] \_mAllowed, Point \_Beg, FindWay \_Father )

{

Begining = \_Beg;

flag = \_flag;

Root = new Point( x, y );

mAllowed = \_mAllowed;

Father = \_Father;

}

public Boolean BuildTree( )

{

Point[] ps = new Point[mAllowed.Length];

int Count = 0;

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

if ( flag )

{

if ( Root.Y == mAllowed[i].Y )

{

Count++;

ps[Count - 1] = mAllowed[i];

}

}

else

if ( Root.X == mAllowed[i].X )

{

Count++;

ps[Count - 1] = mAllowed[i];

}

FindWay fwu = this;

Childrens = new FindWay[Count];

int k = 0;

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

{

if ( ps[i] == Root ) continue;

if ( ps[i] == Begining )

{

while ( fwu != null )

{

mAllowed[k] = fwu.Root;

fwu = fwu.Father;

k++;

};

for ( ; k < mAllowed.Length; k++ ) mAllowed[k] = new Point( -1, -1 );

return true;

}

if ( !Array.TrueForAll<Point>( ps, p => ( ( p.X == 0 ) && ( p.Y == 0 ) ) ) )

{

Childrens[i] = new FindWay( ps[i].X, ps[i].Y, !flag, mAllowed, Begining, this );

Boolean result = Childrens[i].BuildTree();

if ( result ) return true;

}

}

return false;

}

}

// Метод минимального элемента

public float[,] MinEl( )

{

float[] Ahelp = this.dilers;

float[] Bhelp = this.customers;

int i = 0;

int j = 0;

float min = float.MaxValue;

float[,] outArr = new float[this.dilersCount, this.customersCount];

bool[,] pArr = new bool[this.dilersCount, this.customersCount];

for ( i = 0; i < this.dilersCount; i++ )

{

for ( j = 0; j < this.customersCount; j++ )

{

pArr[i, j] = true;

}

}

i = 0;

j = 0;

int k;

int count = 0;

while ( !this.isEmpty( Ahelp ) || !this.isEmpty( Bhelp ) )

{

min = this.findMin( this.transportationPrices, pArr, out i, out j );

float Dif = Math.Min( Ahelp[i], Bhelp[j] );

outArr[i, j] += Dif; count++;

Ahelp[i] -= Dif;

Bhelp[j] -= Dif;

if ( Ahelp[i] == 0f )

{

k = 0;

while ( k < this.customersCount )

{

pArr[i, k] = false;

k++;

}

}

if ( Bhelp[j] == 0f )

{

for ( k = 0; k < this.dilersCount; k++ )

{

pArr[k, j] = false;

}

}

}

this.NanToEmpty( outArr );

// Нуль-загрузка

int difference = ( dilersCount + customersCount - 1 ) - count;

for ( int l = 0; l < difference; l++ )

{

//выбираем непустые

Allowed = new Point[count + 1];

k = 0;

for ( i = 0; i < dilersCount; i++ )

for ( j = 0; j < customersCount; j++ )

if ( outArr[i, j] == outArr[i, j] )

{

Allowed[k] = new Point( i, j );

k++;

}

// ищем куда загрузить

Boolean p = true;

Point Nl = new Point( 0, 0 );

for ( i = 0; ( i < dilersCount ) && p; i++ )

for ( j = 0; ( j < customersCount ) && p; j++ )

{

Nl = Allowed[9] = new Point( i, j );

FindWay fw = new FindWay( i, j, true, Allowed, new Point( i, j ), null );

p = fw.BuildTree();

}

if ( !p ) outArr[Nl.X, Nl.Y] = 0;

}

return outArr;

}

// пока работает...

private void FindUV( float[] U, float[] V, float[,] HelpMatr )

{

//для проверки вычислена ли Ui Vi будем использовать массив boolean'ов

//даже 2 массива. в одном признак того вычислена ли соответствующий потенциал

//во втором прошлись ли мы по строке/строчке этого потенциала

//алгоритм позволит за конечное число итераций вычислить все потенциалы. ура.

bool[] U1 = new bool[dilersCount];

bool[] U2 = new bool[dilersCount];

bool[] V1 = new bool[customersCount];

bool[] V2 = new bool[customersCount];

//V[BSize - 1] = 0;

//V1[BSize - 1] = true;

// пока все элементы массивов V1 и U1 не будут равны true

while ( !( AllTrue( V1 ) && AllTrue( U1 ) ) )

{

int i = -1;

int j = -1;

for ( int i1 = customersCount - 1; i1 >= 0; i1-- )

if ( V1[i1] && !V2[i1] ) i = i1;

for ( int j1 = dilersCount - 1; j1 >= 0; j1-- )

if ( U1[j1] && !U2[j1] ) j = j1;

if ( ( j == -1 ) && ( i == -1 ) )

for ( int i1 = customersCount - 1; i1 >= 0; i1-- )

if ( !V1[i1] && !V2[i1] )

{

i = i1;

V[i] = 0;

V1[i] = true;

break;

}

if ( ( j == -1 ) && ( i == -1 ) )

for ( int j1 = dilersCount - 1; j1 >= 0; j1-- )

if ( !U1[j1] && !U2[j1] )

{

j = j1;

U[j] = 0;

U1[j] = true;

break;

}

if ( i != -1 )

{

for ( int j1 = 0; j1 < dilersCount; j1++ )

{

if ( !U1[j1] ) U[j1] = HelpMatr[j1, i] - V[i];

if ( U[j1] == U[j1] ) U1[j1] = true;

}

V2[i] = true;

}

if ( j != -1 )

{

for ( int i1 = 0; i1 < customersCount; i1++ )

{

if ( !V1[i1] ) V[i1] = HelpMatr[j, i1] - U[j];

if ( V[i1] == V[i1] ) V1[i1] = true;

}

U2[j] = true;

}

}

private Boolean AllPositive( float[,] m )

{

Boolean p = true;

for ( int i = 0; ( i < dilersCount ) && p; i++ )

for ( int j = 0; ( j < customersCount ) && p; j++ )

if ( m[i, j] < 0 ) p = false;

return p;

}

private bool AllTrue( bool[] arr )

{

return Array.TrueForAll( arr, delegate( bool x ) { return x; } );

}

// дозаполняет матрицу S оценками

private float[,] MakeSMatr( float[,] M, float[] U, float[] V )

{

float[,] HM = new float[dilersCount, customersCount];

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

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

{

HM[i, j] = M[i, j];

if ( HM[i, j] != HM[i, j] )

HM[i, j] = transportationPrices[i, j] - ( U[i] + V[j] );

}

return HM;

}

private Point[] Allowed;// хранит координаты клеток, в которых есть груз

public int[] arra = new int[5];

private Point[] GetCycle( int x, int y )

{

Point Beg = new Point( x, y );

FindWay fw = new FindWay( x, y, true, Allowed, Beg, null );

fw.BuildTree();

Point[] Way = Array.FindAll<Point>( Allowed, delegate( Point p ) { return ( p.X != -1 ) && ( p.Y != -1 ); } );

return Way;

}

// находит плохой цикл и крутит его

private void Roll( float[,] m, float[,] sm )

{

Point minInd = new Point();

float min = float.MaxValue;

int k = 0;

Allowed = new Point[dilersCount + customersCount];

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

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

{

if ( m[i, j] == m[i, j] )

{

Allowed[k].X = i;

Allowed[k].Y = j;

k++;

}

// заодно ищем макс по модулю отр элемент

if ( sm[i, j] < min )

{

min = sm[i, j];

minInd.X = i;

minInd.Y = j;

}

}

// Ищем цикл

Allowed[Allowed.Length - 1] = minInd;

Point[] Cycle = GetCycle( minInd.X, minInd.Y );

float[] Cycles = new float[Cycle.Length];

Boolean[] bCycles = new Boolean[Cycle.Length];

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

bCycles[i] = i == bCycles.Length - 1 ? false : true;

min = float.MaxValue;

/\* проблема в следующем:

\* цикл мы находим правильно

\* а вот посчитать правильно не можем

\* ниже поиск минимального элемента

\*/

// поиск минимального

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

{

Cycles[i] = m[Cycle[i].X, Cycle[i].Y];

if ( ( i % 2 == 0 ) && ( Cycles[i] == Cycles[i] ) && ( Cycles[i] < min ) )

{

min = Cycles[i];

minInd = Cycle[i];

}

if ( Cycles[i] != Cycles[i] ) Cycles[i] = 0;

}

// вычитание-прибавление

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

{

if ( i % 2 == 0 )

{

Cycles[i] -= min;

m[Cycle[i].X, Cycle[i].Y] -= min;

}

else

{

Cycles[i] += min;

if ( m[Cycle[i].X, Cycle[i].Y] != m[Cycle[i].X, Cycle[i].Y] ) m[Cycle[i].X, Cycle[i].Y] = 0;

m[Cycle[i].X, Cycle[i].Y] += min;

}

}

m[minInd.X, minInd.Y] = float.NaN;

}

// сама оптимизация

public float[,] PotenMeth( float[,] SupArr )

{

// расчитываем Ui и Vi

//подготовка

int i = 0, j = 0;

float[,] HelpMatr = new float[dilersCount, customersCount];

for ( i = 0; i < dilersCount; i++ )

for ( j = 0; j < customersCount; j++ )

if ( SupArr[i, j] == SupArr[i, j] ) HelpMatr[i, j] = transportationPrices[i, j];

else HelpMatr[i, j] = float.NaN;

//расчёт

float[] U = new float[dilersCount];

float[] V = new float[customersCount];

FindUV( U, V, HelpMatr );

float[,] SMatr = MakeSMatr( HelpMatr, U, V );

//пока все потенциалы не станут положительнымми, будем снова и снова считать

while ( !AllPositive( SMatr ) )

{

Roll( SupArr, SMatr );

for ( i = 0; i < dilersCount; i++ )

for ( j = 0; j < customersCount; j++ )

{

if ( SupArr[i, j] == float.PositiveInfinity )

{

HelpMatr[i, j] = transportationPrices[i, j];

SupArr[i, j] = 0;

continue;

}

if ( SupArr[i, j] == SupArr[i, j] ) HelpMatr[i, j] = transportationPrices[i, j];

else HelpMatr[i, j] = float.NaN;

}

FindUV( U, V, HelpMatr );

SMatr = MakeSMatr( HelpMatr, U, V );

}

return SupArr;

}

}

}

1. Вывод

В ходе выполнения лабораторной работы были получены навыки в решении транспортных задач. Была написана программа, позволяющая решить любую транспортную задачу методом северо-западного угла.