Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

(СибГУТИ)

Факультет информатики и вычислительной техники

09.03.01 "Информатика и вычислительная техника"

профиль "Программное обеспечение средств вычислительной техники и автоматизированных систем"

Кафедра прикладной математики и кибернетики

**Лабораторная работа №3 вариант 1 по дисциплине  
 Алгоритмы и вычислительные методы оптимизации**

Выполнил:

студент гр. ИП-712 Алексеев Степан Владимирович

ФИО студента

«\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

Проверил

Галкина Марина Юрьевна

ФИО преподавателя

«\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г. Оценка\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Новосибирск 2020 г.

Оглавление

[Задание: 2](#_Toc39958851)

[Листинг 2](#_Toc39958852)

[Результаты тестирования 15](#_Toc39958853)

Задание:

Занятие 12 (лабораторная работа №4)

Нахождение начального опорного плана транспортной задачи

Написать программу, находящую начальный опорный план транспортной

задачи одним из указанных методов (номер метода находится как n mod 3, где

n – номер бригады).

1. Метод северо-западного угла.

2. Метод минимальной стоимости.

3. Метод Фогеля

Матрица тарифов, запасы поставщиков и потребности потребителей вводить

из файла.

Программа должна работать как с открытой, так и закрытой моделью

транспортной задачи. Предусмотреть программное нахождение вырожденного

плана. Вывести распределение перевозок и стоимость.

# Листинг

using CW3;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

/\*Реализую 4 лабораторнцю по АВМО.

\* Нахождение начального опорного плана транспортной задачи

Написать программу, находящую начальный опорный план транспортной

задачи методом северо-западного угла.

Матрица тарифов, запасы поставщиков и потребности потребителей вводить

из файла.

Программа должна работать как с открытой, так и закрытой моделью

транспортной задачи. Предусмотреть программное нахождение вырожденного

плана. Вывести распределение перевозок и стоимость.

\*/

namespace CW3//в общем мой вариант м.б. не очень хорош для решения транспортной задачи, но просто для нахождения опорного плана годится. Т.е. не думать

//слишком глубоко! Хотя для вывода распределения перевозок всё же лучше использовать строки... Или можно продублировать матрицу с указанием

//отправок потребителям. Но тогда не видны будут стоимости. А можно вести две матрицы, а в DataGridView уже конкатенировать строковые значения

//полученных чисел из обеих матриц. Выберу второй вариант.

{

public partial class Form1 : Form

{

public Form1()

{ //НАЧАЛО КОНСТРУКТОРА

InitializeComponent();

mAh = 3;//поставщики А(строки)

nBv = 5;//потребители В(столбцы)

mas = new int[mAh, nBv];//матрица тарифов

mas[0, 0] = 12; mas[0, 1] = 13; mas[0, 2] = 11; mas[0, 3] = 8; mas[0, 4] = 10;

mas[1, 0] = 9; mas[1, 1] = 10; mas[1, 2] = 7; mas[1, 3] = 5; mas[1, 4] = 8;

mas[2, 0] = 10; mas[2, 1] = 8; mas[2, 2] = 7; mas[2, 3] = 7; mas[2, 4] = 8;

Ah = new int[mAh];//поставщики А

Ah[0] = 88; Ah[1] = 6; Ah[2] = 92;

Bv = new int[nBv];//потребители В

Bv[0] = 32; Bv[1] = 62; Bv[2] = 54; Bv[3] = 10; Bv[4] = 78;

gridCols = nBv + 3;// 1 для "Запасы", 1 для "Потребности", 1 для случая, когда число столбцов

//увеличится из-за добавления нового(фиктивного) потребителя...

} //КОНЕЦ КОНСТРУКТОРА

int mAh = 0;//число пунктов отправления(поставщиков Ah)(строки)

int nBv = 0;//число пунктов назначения(потребителей Bv)(столбцы)

int[,] mas;//матрица тарифов(стоимостей перевозок)

int[] Ah;// массив запасов для каждого поставщика

int[] Bv;// массив потребностей для каждого потребителя

public int rowsIncrement = 0;//для прорисовки промежуточных решений(чтобы новые матрицы рисовались ниже)

int[] tempB;//на случай изменения размеров массивов

int[] tempA;//на случай изменения размеров массивов

int[,] tempMas;//на случай изменения размеров массивов

int gridCols;//размер матрицы в ширину для вывода в dataGridView

public string[] elems;//список имён вершин для подписи в матрице

int[,] masSolv;//матрица пермещений(результат работы алгоритма Северо-Западного угла)

bool virojdenniyPlan = false;

private void printMatrixMasBumaga(object sender, EventArgs e) // <--- КНОПКА "Вывести матрицу"

{

dataGridView1.RowCount = mAh \* 300;//просто делаю всю полосу подлиннее с запасом

dataGridView1.ColumnCount = gridCols;

dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.DisplayedCells;

// initilaizeHeaders();

initializeUpperPart();

fillTableOriginal();// в первый раз нарисовал матричку

checkOpenOrClosed(Ah, Bv);//поменял размер матрицы если задача открытая, превратив её в закрытую

masSolv = new int[mAh, nBv];//инициализирую массив перевозок

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

{

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

{//по умолчанию сишарп инициализирует массив нулями, поэтому, чтобы обеспечить возможность нудевой перевозки(только одной клетки),

//было принято решение инициализации отрицательными значениями всего остального

masSolv[i, j] = -1;

}

}

fillTableOriginal();//ещё раз вывел матрицу, чтобы посмотреть, поменялась ли она после проверки на открытость/закрытость

solveNorthWest();//применил метод Северо-Западного угла

fillTableSolved();//вывел объединённые матрицу тарифов и матрицу перевозок

if (!virojdenniyPlan) textBox3.Text = "План не вырожденный";

} // <--- КОНЕЦ КНОПКИ "Вывести матрицу"

public void fillTableSolved()//распечатка матрицы с отправленными товарами

{

int f = 0;

initilaizeHeaders();//прописываю B0, B1 и т.д., в конце прописываю "Запасы"

rowsIncrement++;

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

{

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

{

if (masSolv[i, j] >= 0)//если перемещение товара зафиксировано в masSolv, то рисуем с количеством перемещённого товара

{

dataGridView1.Rows[rowsIncrement + i].Cells[j].Value = ("+" + masSolv[i, j] + " \* " + mas[i, j]);//вывел значения матрицы тарифов и куда сколько направляется

f += (masSolv[i, j] \* mas[i, j]);

}

else

{//если движения товара не зафиксировано, то просто пишем цену

dataGridView1.Rows[rowsIncrement + i].Cells[j].Value = mas[i, j];

}

}

dataGridView1.Rows[rowsIncrement + i].Cells[nBv].Value = Ah[i];// вывел запасы

dataGridView1.Rows[rowsIncrement + i].Cells[nBv + 1].Value = "A" + i;

}

textBox1.Text = "F(x) = " + f;

dataGridView1.Rows[rowsIncrement + mAh].Cells[nBv + 1].Value = "<--Потребности";

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

{

dataGridView1.Rows[rowsIncrement + mAh].Cells[j].Value = Bv[j];//вывел потребности

}

rowsIncrement += (mAh + 2);//для прорисовки матрицы

}

public void solveNorthWest()

{// цикл фор сначала пойдёт по первой строке, посылая товар потребителям, пока запас этой строки не иссякнет,

//каждый раз выдавая товаров по максимуму

int currRow = 0, currCol = 0;

while (true)//mAh - число строк(магазинов)

{

if (currRow == mAh) break;

if (currRow == (mAh - 1) && Ah[currRow] == 0)//если дошли до последнего ряда и его запас равен нулю, то конец

{

break;

}

int ostatok = Ah[currRow];//скопировал число из магазина в рабочую переменную

int raznica = Ah[currRow] - Bv[currCol];// магазины меняются по ряду, потребители по столбцу

if (raznica > 0)

{

masSolv[currRow, currCol] = Bv[currCol];//записал передачу в клетке masSolv[currRow, currCol] объёма Bv[currCol]

Bv[currCol] = 0;//занулил потребность текущей колонки, т.к. она удовлетворена

currCol++;//переходим к след столбцу, оставаясь на том же ряду

Ah[currRow] = raznica;

}

else if (raznica == 0)/\*Лекция 11, стр 4: Если на каком-нибудь шаге метода одновременно удовлетворяются все потребности потребителя B[j]

и вывозится весь груз от поставщика A[i] , то опорный план – вырожденный, и в какую-нибудь клетку j-го столбца и

i-ой строки заносят нулевую поставку и из дальнейшего рассмотрения исключают и строку, и столбец.\*/

{

if (!(currCol == (nBv - 1)))// если речь не о последнем столбце?... Ведь если рассматривать последний столбец, то там в любом случае обязательно

//происходит удовлетворение всех потребностей и вывоз всех грузов. Тогда в соответствии с правилом из лекции, все закрытые задачи будут

//с вырожденным опорным планом. Делаю вывод, что последний столбец является исключением из правила.

//Также не понятно как "в какую-нибудь клетку j-го столбца и!(одновременно?) i-ой строки" занести нулевую поставку.

//В общем сделаю так, что если в столбце ниже ещё есть маршруты, то поставлю туда ноль, если нет, то

//поставлю справа. Т.е. буду считать, что вместо "и" в лекции д.б. "или".

{

virojdenniyPlan = true;

textBox3.Text = "План вырожденный, текущая колонка: " + currCol + ", текущий ряд: " + currRow;

if (currRow < (mAh - 1))//если текущий ряд выше последнего, то в ряду ниже текущего можно разместить нулевую перевозку

{

masSolv[currRow + 1, currCol] = 0;

}

else// иначе размещаем нулевую перевозку справа от текущей клетки(в которой произошло одновременное зануление Ah & Bv

{

masSolv[currRow, currCol + 1] = 0;

}

}

masSolv[currRow, currCol] = Bv[currCol];//записал передачу в клетке masSolv[currRow, currCol] объёма Bv[currCol]

Bv[currCol] = 0;//занулил потребность текущей колонки, т.к. она удовлетворена

Ah[currRow] = raznica;//занулил запас текущего магазина

currCol++;//переходим к след столбцу, и ряду

currRow++;

}

else if (raznica < 0)//в этом случае запас магазина истощён, а потребность осталась в размере модуля raznica

{

masSolv[currRow, currCol] = ostatok;

Bv[currCol] = (raznica \*= -1);//потребность обновляем

Ah[currRow] = 0;

currRow++;

}

}

}

public void checkOpenOrClosed(int[] Ah, int[] Bv)//проверяю открытая или закрытая задача. Если открытая, то преобразую в закрытую.

{

int sumA = 0, sumB = 0;

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

{

sumA += Ah[i];

}

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

{

sumB += Bv[i];

}

if ((sumA - sumB) == 0)

{//Если срос равен предложению, то матрица остаётся неизменной

textBox2.Text = "Спрос равен предложению. Матрица остаётся прежней.";

}

else if ((sumA - sumB) > 0)

{//Если предложение больше спроса, то вводим фиктивного потребителя(В)(увеличиваем число столбцов). При этом в сишарпе нет

//хороших функций для ресайза массивов типа List. Если бы я создал такой массив, его всё равно пришлось бы переделывать

//вручную. Поэтому просто сделаю копию примитивов вручную.

textBox2.Text = "Предложение больше спроса, вводим фиктивного потребителя, чтобы сделать задачу закрытой.";

tempB = new int[++nBv];//создал временный массив, в который скопирую В с увеличенным на 1 размером. nBv теперь на 1 больше.

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

{

tempB[i] = Bv[i];

}

tempB[nBv - 1] = (sumA - sumB);//в последний элемент добавляю недостающее число

this.Bv = tempB;

{//test

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

{

Console.WriteLine("Bv[" + i + "] = " + Bv[i]);

}

}

// матрицу тарифов тоже надо переделать: увеличилось число столбцов

tempMas = new int[mAh, nBv + 1];// дальше пока просто скопировал из третьего случая

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

{

for (int j = 0; j < nBv - 1; j++)

{

tempMas[i, j] = mas[i, j];//просто скопировал

}

tempMas[i, nBv] = 0;

}

this.mas = tempMas;

//for (int i = 0; i < nBv; i++)

//{

// mas[mAh, i] = 0;

//}

}

else if ((sumA - sumB) < 0)

{//Если спрос больше предложения, то вводим фиктивного поставщика(А)

textBox2.Text = "Спрос больше предложения, вводим фиктивного поставщика, чтобы задача стала закрытой.";

tempA = new int[++mAh];//создал временный массив, в который скопирую А с увеличенным на 1 размером. mAh теперь на 1 больше.

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

{

tempA[i] = Ah[i];

}

tempA[mAh - 1] = (sumB - sumA);

// Ah = new int[mAh];

this.Ah = tempA;//перенаправил ссылку. //в сишарпе ведь как и в джаве копируются ссылки только?

//Да. Причём не важно, что размер исходной матрицы был другим. А вот как раз важно. Тут странность сишарпа? Нет. Решение ниже.

//Выводится здесь Ah увеличенного размера, а вот в функции filltable() Ah всё также на 1 короче почему-то...

//Оказывается надо было указать this.Ah, несмотря на то, что Ah объявлена в теле класса, т.е. не является локальной ссылкой.

// Недопонимание получилось, ведь Ah - действительно локальная, я её прописал в списке параметров этой функции. Так что сишарп не при чём.

{//test

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

{

Console.WriteLine("Ah[" + i + "] = " + Ah[i]);

}

Console.WriteLine("mAh = " + mAh);

}

// матрицу тарифов тоже надо переделать: появилась новая строка с нулевыми стоимостями для всех путей.

tempMas = new int[mAh + 1, nBv];//попробовать +1 убрать

for (int i = 0; i < mAh - 1; i++)// -1 т.к. в mas ещё нет этой строки

{

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

{

tempMas[i, j] = mas[i, j];//просто скопировал

}

}

mas = tempMas;

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

{

mas[mAh, i] = 0;

}

}

}

private void initilaizeHeaders()//------------------------------------расписал заголовки вверху таблицы. Число переменных может возрасти,

//поэтому данная функция более неактуальна, и должна быть переработана под новые условия.

//Вызов этой функции переезжает в fillTableOriginal();

{

elems = new string[gridCols];

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

{

elems[i] = "B" + Convert.ToString(i);//создал строки для обозначения потребителей

}

elems[nBv] = "Запасы";

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

{

dataGridView1.Rows[rowsIncrement].Cells[i].Value = elems[i];//расписал все заголовки в dataGridView1

}

}

private void fillTableOriginal()//в принципе одинаково правильно работает как с оригиналом, так и с изменённой матрицей

{

initilaizeHeaders();//прописываю B0, B1 и т.д., в конце "Запасы"

rowsIncrement++;

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

{

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

{

dataGridView1.Rows[rowsIncrement + i].Cells[j].Value = mas[i, j];//вывел значения матрицы тарифов

}

dataGridView1.Rows[rowsIncrement + i].Cells[nBv].Value = Ah[i];// вывел запасы

dataGridView1.Rows[rowsIncrement + i].Cells[nBv + 1].Value = "A" + i;

}

dataGridView1.Rows[rowsIncrement + mAh].Cells[nBv + 1].Value = "<--Потребности";

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

{

dataGridView1.Rows[rowsIncrement + mAh].Cells[j].Value = Bv[j];//вывел потребности

}

rowsIncrement += (mAh + 2);//для прорисовки матрицы

}

public void initializeUpperPart()

{

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

{

dataGridView1.Columns[i].HeaderCell.Value = "----";

}

}

private void button4\_Click(object sender, EventArgs e)// "Solve"

{

}

private void button1\_Click(object sender, EventArgs e)// "Build a table"

{

}

private void button2\_Click(object sender, EventArgs e) // "Load"

{

checkOpenOrClosed(Ah, Bv);

}

private void button3\_Click(object sender, EventArgs e) // "test"

{

}

private void Form1\_Load(object sender, EventArgs e)

{

}

private void textBox3\_TextChanged(object sender, EventArgs e)

{

}

}

}

# Результаты тестирования

