МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»
(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №4

по дисциплине: Исследование операций

тема: Закрытая транспортная задача

Выполнил: ст. группы ПВ-223
Игнатьев Артур
Проверил:
Вирченко Юрий Петрович

Белгород 2024 г.

Цель работы: изучить математическую модель транспортной задачи, овладеть методами решения этой задачи.

Задания

1. Изучить содержательную и математическую постановки закрытой транспортной задачи, методы нахождения первого опорного решения ее системы ограничений. Изучить понятие цикла пересчета в матрице перевозок. Овладеть распределительным методом и методом потенциалов, а также их алгоритмами.
2. Составить и отладить программы решения транспортной задачи распределительным методом и методом потенциалов.
3. Для подготовки тестовых данных решить вручную следующую задачу.

Вариант 3

3.

$$\vec{a} = (19, 19, 19, 19);$$

$$\vec{b} = (17, 17, 17, 17, 8);$$

$$C = \begin{pmatrix} 22 & 23 & 16 & 12 & 14 \\ 17 & 30 & 1 & 8 & 25 \\ 27 & 15 & 13 & 23 & 22 \\ 3 & 12 & 21 & 26 & 7 \end{pmatrix}$$

	B1	B2	B3	B4	B5	Запасы
A1	22	23	16	12	14	19
A2	17	30	1	8	25	19
A3	27	15	13	23	22	19
A4	3	12	21	26	7	19
Потребности	17	17	17	17	8	

Проверим необходимое и достаточное условие разрешимости задачи.

$\sum a = 19 + 19 + 19 + 19 = 76$

$\sum b = 17 + 17 + 17 + 17 + 8 = 76$

Ручное решение:

Составляем опорный план

Искомый элемент равен $c_{23}=1$. Для этого элемента запасы равны 19, потребности 17. Поскольку минимальным является 17, то вычитаем его.

$x_{23} = \min(19,17) = 17$.

22	23	x	12	14	19
17	30	**1**	8	25	**19 - 17 = 2**
27	15	x	23	22	19
3	12	x	26	7	19
17	17	**17 - 17 = 0**	17	8	

Искомый элемент равен c_{41}=3. Для этого элемента запасы равны 19, потребности 17. Поскольку минимальным является 17, то вычитаем его. x_{41} = min(19,17) = 17.

x	23	x	12	14	19
x	30	1	8	25	2
x	15	x	23	22	19
3	12	x	26	7	**19 - 17 = 2**
17 - 17 = 0	17	0	17	8	

Искомый элемент равен c_{45}=7. Для этого элемента запасы равны 2, потребности 8. Поскольку минимальным является 2, то вычитаем его. x_{45} = min(2,8) = 2.

x	23	x	12	14	19
x	30	1	8	25	2
x	15	x	23	22	19
3	x	x	x	**7**	**2 - 2 = 0**
0	17	0	17	**8 - 2 = 6**	

Искомый элемент равен c_{24}=8. Для этого элемента запасы равны 2, потребности 17. Поскольку минимальным является 2, то вычитаем его. x_{24} = min(2,17) = 2.

x	23	x	12	14	19
x	x	1	**8**	x	**2 - 2 = 0**
x	15	x	23	22	19
3	x	x	x	7	0
0	17	0	**17 - 2 = 15**	6	

Искомый элемент равен c_{14}=12. Для этого элемента запасы равны 19, потребности 15. Поскольку минимальным является 15, то вычитаем его. x_{14} = min(19,15) = 15.

x	23	x	**12**	14	**19 - 15 = 4**
x	x	1	8	x	0
x	15	x	x	22	19
3	x	x	x	7	0
0	17	0	**15 - 15 = 0**	6	

Искомый элемент равен c_{15}=14. Для этого элемента запасы равны 4, потребности 6. Поскольку минимальным является 4, то вычитаем его. x_{15} = min(4,6) = 4.

x	x	x	12	**14**	**4 - 4 = 0**
x	x	1	8	x	0
x	15	x	x	22	19
3	x	x	x	7	0
0	17	0	0	**6 - 4 = 2**	

Искомый элемент равен c_{32}=15. Для этого элемента запасы равны 19, потребности 17. Поскольку минимальным является 17, то вычитаем его. x_{32} = min(19,17) = 17.

x	x	x	12	14	0
x	x	1	8	x	0
x	**15**	x	x	22	**19 - 17 = 2**
3	x	x	x	7	0
0	**17 - 17 = 0**	0	0	2	

Искомый элемент равен c_{35}=22. Для этого элемента запасы равны 2, потребности 2. Поскольку минимальным является 2, то вычитаем его. x_{35} = min(2,2) = 2.

x	x	x	12	14	0
x	x	1	8	x	0
x	15	x	x	**22**	**2 - 2 = 0**
3	x	x	x	7	0
0	0	0	0	**2 - 2 = 0**	

	B1	B2	B3	B4	B5	Запасы
A1	22	23	16	12[15]	14[4]	19
A2	17	30	1[17]	8[2]	25	19
A3	27	15[17]	13	23	22[2]	19
A4	3[17]	12	21	26	7[2]	19
Потребности	17	17	17	17	8	

В результате получен первый опорный план, который является допустимым, так как все грузы из баз вывезены, потребность магазинов удовлетворена, а план соответствует системе ограничений транспортной задачи.

Подсчитаем число занятых клеток таблицы, их 8, а должно быть m + n - 1 = 8. Следовательно, опорный план является *невырожденным*.
Значение целевой функции для этого опорного плана равно:
F(x) = 12*15 + 14*4 + 1*17 + 8*2 + 15*17 + 22*2 + 3*17 + 7*2 = 633

Улучшение опорного палана

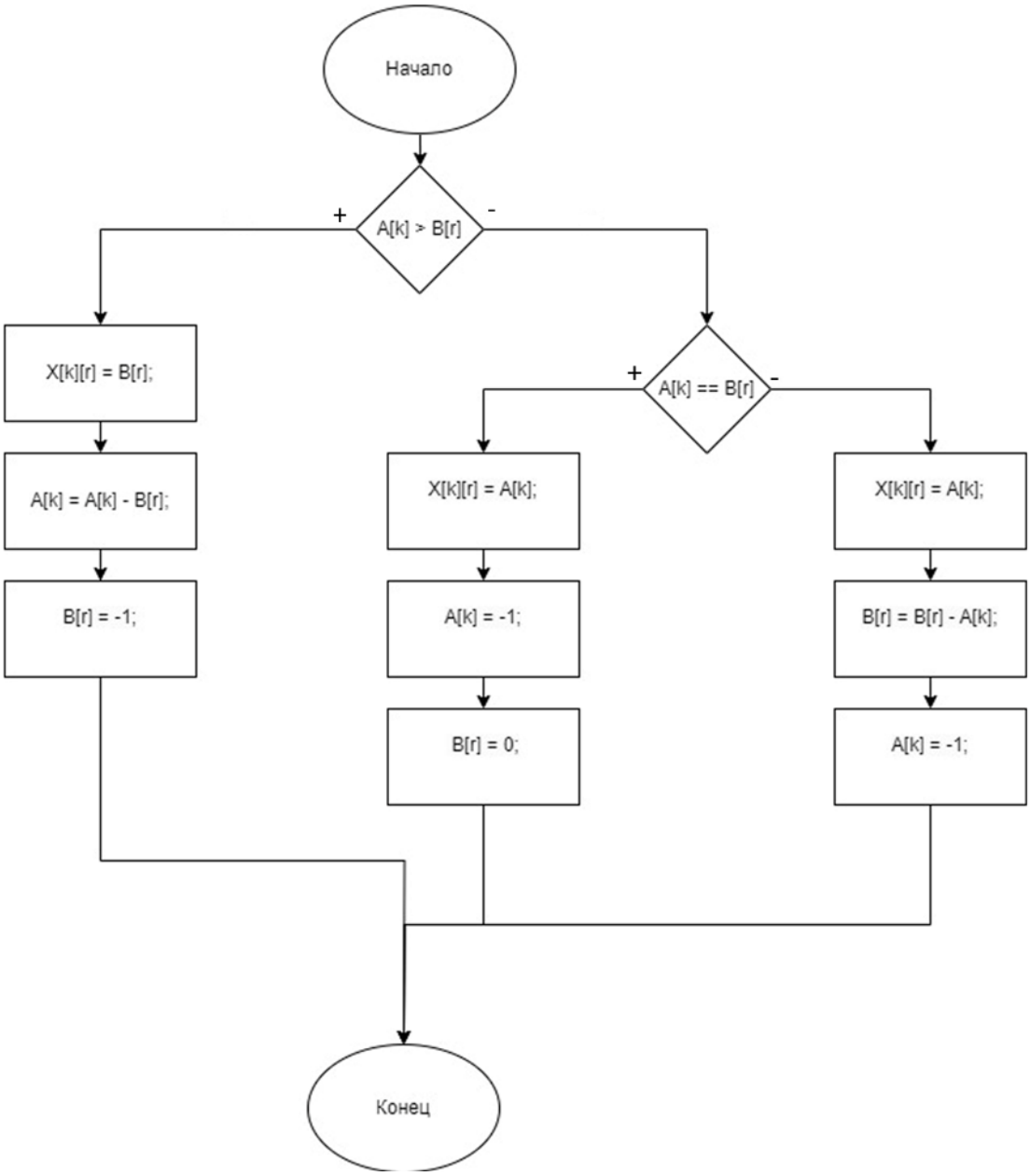
Проверим оптимальность опорного плана. Найдем *предварительные потенциалы* u_i, v_j. по занятым клеткам таблицы, в которых $u_i + v_j = c_{ij}$, полагая, что $u_1 = 0$.
$u_1 + v_4 = 12$; $0 + v_4 = 12$; $v_4 = 12$
$u_2 + v_4 = 8$; $12 + u_2 = 8$; $u_2 = -4$
$u_2 + v_3 = 1$; $-4 + v_3 = 1$; $v_3 = 5$
$u_1 + v_5 = 14$; $0 + v_5 = 14$; $v_5 = 14$
$u_3 + v_5 = 22$; $14 + u_3 = 22$; $u_3 = 8$
$u_3 + v_2 = 15$; $8 + v_2 = 15$; $v_2 = 7$
$u_4 + v_5 = 7$; $14 + u_4 = 7$; $u_4 = -7$
$u_4 + v_1 = 3$; $-7 + v_1 = 3$; $v_1 = 10$

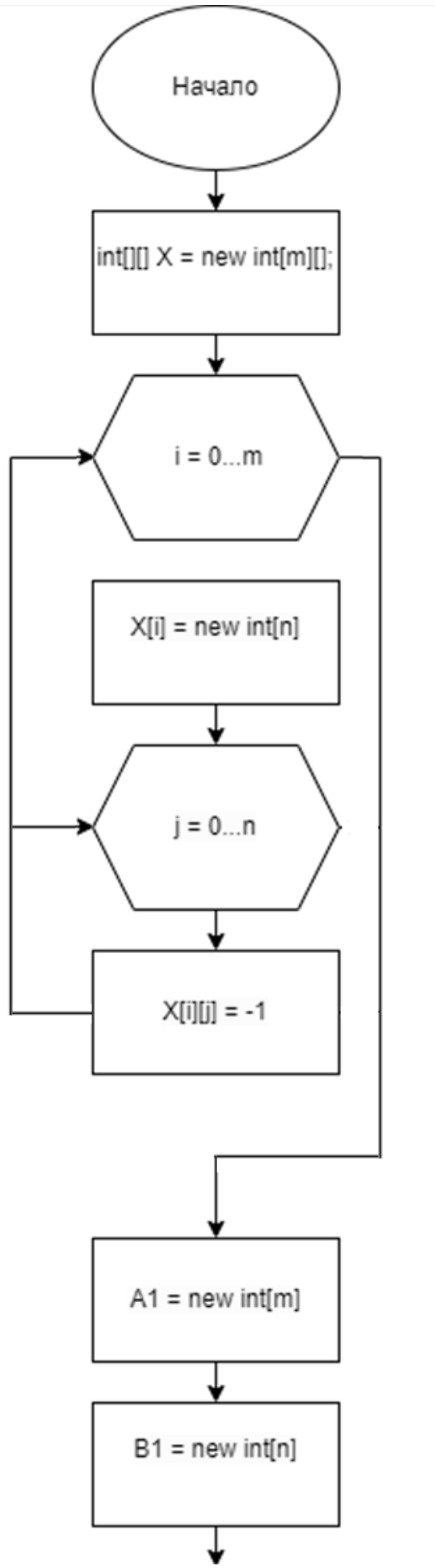
	v_1=10	v_2=7	v_3=5	v_4=12	v_5=14
u_1=0	22	23	16	12[15]	14[4]
u_2=-4	17	30	1[17]	8[2]	25
u_3=8	27	15[17]	13	23	22[2]
u_4=-7	3[17]	12	21	26	7[2]

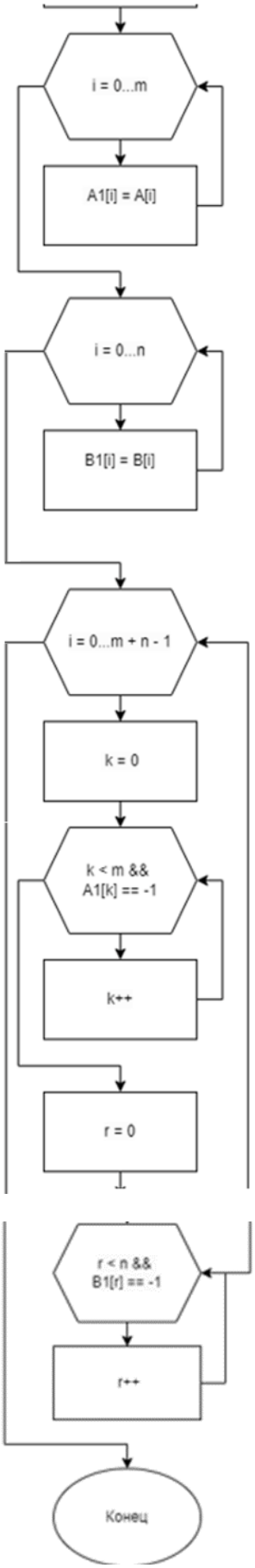
Опорный план является оптимальным, так все оценки свободных клеток удовлетворяют условию $u_i + v_j \leq c_{ij}$.
Минимальные затраты составят: F(x) = 12*15 + 14*4 + 1*17 + 8*2 + 15*17 + 22*2 + 3*17 + 7*2 = 633

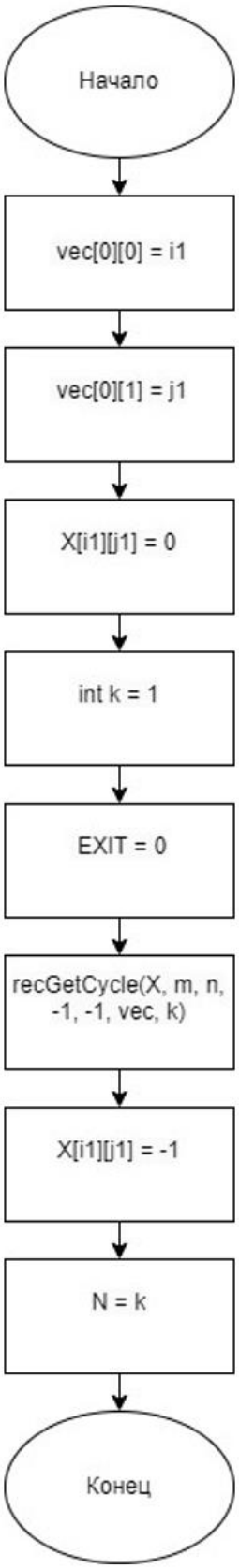
Анализ оптимального плана.
Из 1-го склада необходимо груз направить в 4-й магазин (15 ед.), в 5-й магазин (4 ед.)
Из 2-го склада необходимо груз направить в 3-й магазин (17 ед.), в 4-й магазин (2 ед.)
Из 3-го склада необходимо груз направить в 2-й магазин (17 ед.), в 5-й магазин (2 ед.)
Из 4-го склада необходимо груз направить в 1-й магазин (17 ед.), в 5-й магазин (2 ед.)

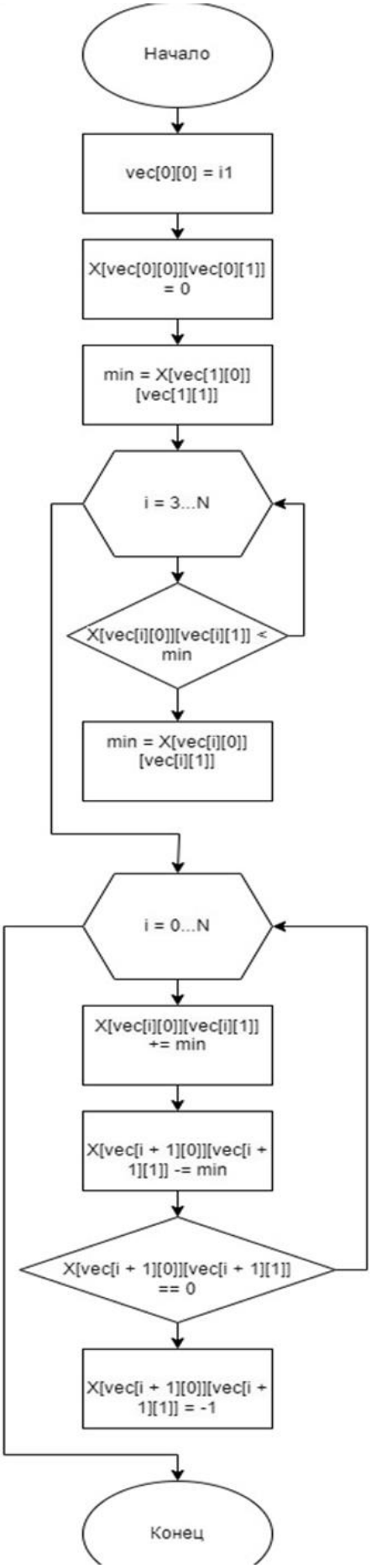
Блок схемы:

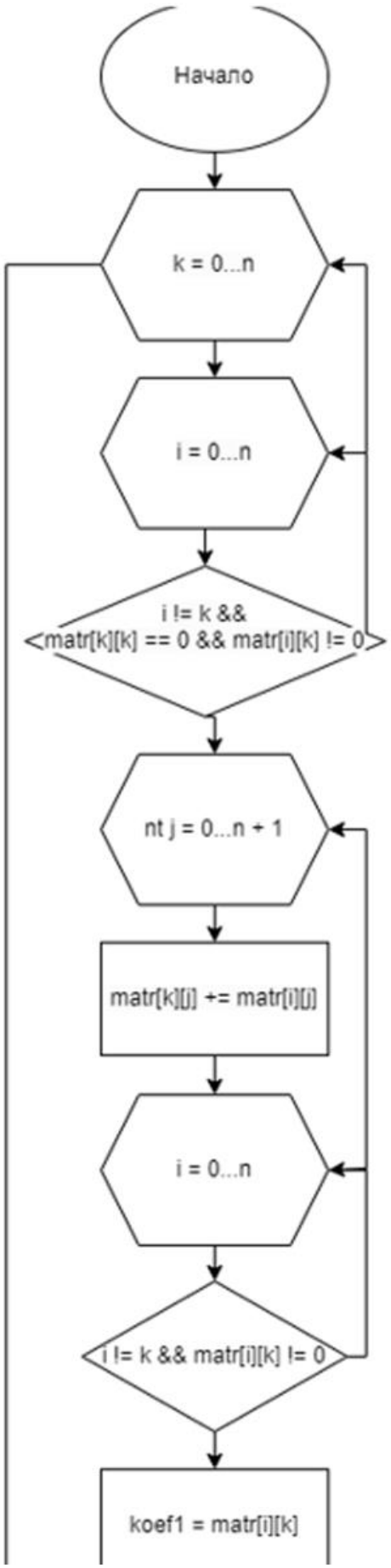

Начало
A[k] > B[r]
+
-
X[k][r] = B[r];
A[k] = A[k] - B[r];
B[r] = -1;
A[k] == B[r]
+
-
X[k][r] = A[k];
A[k] = -1;
B[r] = 0;
X[k][r] = A[k];
B[r] = B[r] - A[k];
A[k] = -1;
Конец

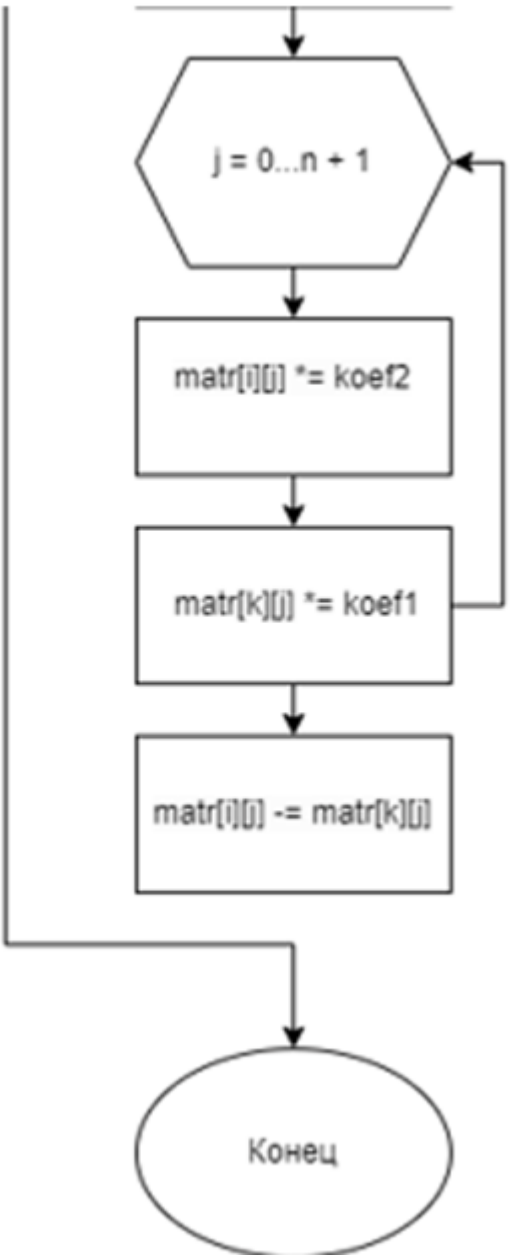

Начало
int[][] X = new int[m][];
i = 0...m
X[i] = new int[n]
j = 0...n
X[i][j] = -1
A1 = new int[m]
B1 = new int[n]


i = 0...m
A1[i] = A[i]
i = 0...n
B1[i] = B[i]
i = 0...m + n - 1
k = 0
k < m &&
A1[k] == -1
k++
r = 0
r < n &&
B1[r] == -1
r++
Конец


Начало
vec[0][0] = i1
vec[0][1] = j1
X[i1][j1] = 0
int k = 1
EXIT = 0
recGetCycle(X, m, n, -1, -1, vec, k)
X[i1][j1] = -1
N = k
Конец


Начало
vec[0][0] = i1
X[vec[0][0]][vec[0][1]] = 0
min = X[vec[1][0]] [vec[1][1]]
i = 3...N
X[vec[i][0]][vec[i][1]] < min
min = X[vec[i][0]] [vec[i][1]]
i = 0...N
X[vec[i][0]][vec[i][1]] += min
X[vec[i + 1][0]][vec[i + 1][1]] -= min
X[vec[i + 1][0]][vec[i + 1][1]] == 0
X[vec[i + 1][0]][vec[i + 1][1]] = -1
Конец


Начало
k = 0...n
i = 0...n
i != k &&
<matr[k][k] == 0 && matr[i][k] != 0
nt j = 0...n + 1
matr[k][j] += matr[i][j]
i = 0...n
i != k && matr[i][k] != 0
koef1 = matr[i][k]


j = 0...n + 1
matr[i][j] *= koef2
matr[k][j] *= koef1
matr[i][j] -= matr[k][j]
Конец

Код программы:

```
using System;
public class TransportationProblem
{
    static int EXIT = 0;

    static int getLength(int a)
    {
        int l = 0;

        while (a > 0)
        {
            l++;
            a = a / 10;
        }

        return l;
    }
    // Метод для отображения таблицы с текущими рузультатами расчетов в
консоль
    static void writeTable(int[][] X, int[][] C, int m, int n)
    {
        int l = 0;

        Console.Write("====================================");

        for (int i = 0; i < m - 1; i++)
        {
            Console.Write('|');

            for (int j = 0; j < n - 1; j++)
            {
                Console.Write(C[i][j]);
                l = getLength(C[i][j]);

                for (int z = 0; z < 3 - l; z++)
                    Console.Write(" ");
            }

            Console.Write(C[i][n - 1]);
            l = getLength(C[i][n - 1]);

            for (int z = 0; z < 3 - l; z++)
                Console.Write(" ");

            for (int j = 0; j < n - 1; j++)
                if (X[i][j] >= 0)
                    Console.Write(X[i][j] + "\t");
                else
                    Console.Write("\t", '|');

            if (X[i][n - 1] >= 0)
                Console.Write(X[i][n - 1] + "\n");
            else
                Console.Write("\t\n", '|', '.');

            Console.Write("====================================");
        }

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

```
        {
            Console.Write(C[m - 1][j]);
            l = getLength(C[m - 1][j]);

            for (int z = 0; z < 3 - l; z++)
                Console.Write(" ");

            Console.Write("\t", '|', '|');
        }

        Console.Write(C[m - 1][n - 1]);
        l = getLength(C[m - 1][n - 1]);

        for (int z = 0; z < 3 - l; z++)
            Console.Write(" ");

        for (int j = 0; j < n - 1; j++)
            if (X[m - 1][j] >= 0)
                Console.Write(X[m - 1][j] + "\t");
            else
                Console.Write("\t", '|');

        if (X[m - 1][n - 1] >= 0)
            Console.Write(X[m - 1][n - 1] + "\t\n");
        else
            Console.Write("\t\n", '|', '.');

        Console.Write("====================================");

        Console.Write("\n", '.');
    }




    // Вспомогательный метод для нахождения опорного плана методом северо-западного угла
    static void choice(int[] A, int[] B, int[][] X, int k, int r)
    {
        if (A[k] > B[r])
        {
            X[k][r] = B[r];
            A[k] = A[k] - B[r];
            B[r] = -1;
        }
        else if (A[k] == B[r])
        {
            X[k][r] = A[k];
            A[k] = -1;
            B[r] = 0;
        }
        else
        {
            X[k][r] = A[k];
            B[r] = B[r] - A[k];
            A[k] = -1;
        }
    }
```

```
    // Метод для нахождения опорного плана методом северо-западного угла
    static int[][] northwestCornerMethod(int[] A, int[] B, int[][] C, int
m, int n)
    {
        int[][] X = new int[m][];

        for (int i = 0; i < m; i++)
        {
            X[i] = new int[n];

            for (int j = 0; j < n; j++)
                X[i][j] = -1;
        }

        int[] A1 = new int[m];
        int[] B1 = new int[n];

        for (int i = 0; i < m; i++)
            A1[i] = A[i];

        for (int i = 0; i < n; i++)
            B1[i] = B[i];

        for (int i = 0; i < m + n - 1; i++)
        {
            int k = 0;

            while (k < m && A1[k] == -1)
                k++;

            int r = 0;

            while (r < n && B1[r] == -1)
                r++;

            choice(A1, B1, X, k, r);
        }

        return X;
    }

    // Метод для нахождения опорного плана методом наименьшей стоимости
    static int[][] smallestPenaltyMethod(int[] A, int[] B, int[][] C, int
m, int n)
    {
        int[][] X = new int[m][];

        for (int i = 0; i < m; i++)
        {
            X[i] = new int[n];

            for (int j = 0; j < n; j++)
                X[i][j] = -1;
        }

        int[] A1 = new int[m];
        int[] B1 = new int[n];

        for (int i = 0; i < m; i++)
            A1[i] = A[i];
```

```
        for (int i = 0; i < n; i++)
            B1[i] = B[i];

        for (int i = 0; i < m + n - 1; i++)
        {
            int km = 0, rm = 0;

            while (km < m && A1[km] == -1)
                km++;

            while (rm < n && B1[rm] == -1)
                rm++;
            int min = C[km][rm];

            for (int k = 0; k < m; k++)
                for (int r = 0; r < n; r++)
                    if (A1[k] != -1 && B1[r] != -1 && C[k][r] < min)
                    {
                        min = C[k][r];
                        km = k;
                        rm = r;
                    }

            choice(A1, B1, X, km, rm);
        }

        return X;
    }

    // Рекурсивный вспомогательный метод для нахождения цикла в таблице
    static void recGetCycle(int[][] X, int m, int n, int i1, int j1,
int[][] vec, int k)
    {
        int i, j, f = 0;

        if (vec[0][0] == i1 && vec[0][1] == j1)
            EXIT = 1;

        if (EXIT == 0)
        {
            if (i1 >= 0 && j1 >= 0)
            {
                vec[k][0] = i1;
                vec[k][1] = j1;
                (k)++;
            }

            if (k < 2 || vec[k - 1][0] == vec[k - 2][0])
            {
                j = vec[k - 1][1];
                i = 0;

                while (f == 0 && i < m)
                {
                    if (X[i][j] >= 0 && i != vec[k - 1][0])
                        recGetCycle(X, m, n, i, j, vec, k);
                    i++;
                }

                if (EXIT == 0)
```

```
                    (k)--;
            }
            else
            {
                i = vec[k - 1][0];
                j = 0;

                while (f == 0 && j < n)
                {
                    if (X[i][j] >= 0 && j != vec[k - 1][1])
                        recGetCycle(X, m, n, i, j, vec, k);

                    j++;
                }

                if (EXIT == 0)
                    (k)--;
            }
        }
    }
    // Метод для нахождения цикла в таблице
    static void getCycle(int[][] X, int m, int n, int i1, int j1, int[][]
vec, int N)
    {
        vec[0][0] = i1;
        vec[0][1] = j1;
        X[i1][j1] = 0;
        int k = 1;
        EXIT = 0;

        recGetCycle(X, m, n, -1, -1, vec, k);

        X[i1][j1] = -1;
        N = k;
    }

    // Метод для реализации сдвига по циклу на значение
    static void shiftCycle(int[][] X, int[][] vec, int N)
    {
        X[vec[0][0]][vec[0][1]] = 0;
        int min = X[vec[1][0]][vec[1][1]];

        for (int i = 3; i < N; i += 2)
            if (X[vec[i][0]][vec[i][1]] < min)
                min = X[vec[i][0]][vec[i][1]];

        for (int i = 0; i < N; i += 2)
        {
            X[vec[i][0]][vec[i][1]] += min;
            X[vec[i + 1][0]][vec[i + 1][1]] -= min;

            if (X[vec[i + 1][0]][vec[i + 1][1]] == 0)
                X[vec[i + 1][0]][vec[i + 1][1]] = -1;
        }
    }

    // Метод для улучшения опорного плана распределительным методом
    static void distributionMethod(int[][] X, int[][] C, int m, int n)
    {
        int N = 0;
        int y = 0;
```

```
        int[][] vec = new int[n + m][];

        for (int i = 0; i < n + m; i++)
            vec[i] = new int[2];

        int f = 0;

        while (f == 0)
        {
            f = 1;
            for (int i = 0; i < m; i++)
                for (int j = 0; j < n; j++)
                    if (X[i][j] < 0)
                    {
                        y = 0;
                        getCycle(X, m, n, i, j, vec, N);

                        for (int z = 0; z < N; z += 2)
                            y += C[vec[z][0]][vec[z][1]] - C[vec[z +
1][0]][vec[z + 1][1]];

                        if (y < 0)
                        {
                            shiftCycle(X, vec, N);
                            f = 0;
                        }
                    }
        }
    }

    // Метод для нахождения потенциалов
    static void getBasis(int[][] matr, int n)
    {
        for (int k = 0; k < n; k++)
        {
            for (int i = 0; i < n; i++)
                if (i != k && matr[k][k] == 0 && matr[i][k] != 0)
                    for (int j = 0; j < n + 1; j++)
                        matr[k][j] += matr[i][j];

            for (int i = 0; i < n; i++)
                if (i != k && matr[i][k] != 0)
                {
                    int koef1 = matr[i][k];
                    int koef2 = matr[k][k];

                    for (int j = 0; j < n + 1; j++)
                    {
                        matr[i][j] *= koef2;
                        matr[k][j] *= koef1;
                        matr[i][j] -= matr[k][j];
                    }
                }
        }
    }
    // Метод для улучшения оптимального плана методом потенциалов
    static void getPotential(int[][] X, int[][] C, int m, int n, int[] u,
int[] v)
    {
        int k = 0;
```

```
        int[][] sys = new int[m + n - 1][];

        for (int i = 0; i < m + n - 1; i++)
            sys[i] = new int[m + n + 1];

        for (int i = 0; i < m; i++)
            for (int j = 0; j < n; j++)
                if (X[i][j] >= 0)
                {
                    sys[k][i] = 1;
                    sys[k][m + j] = 1;
                    sys[k][m + n] = C[i][j];
                    k++;
                }

        v[n - 1] = 0;

        for (int i = 0; i < m + n - 1; i++)
            sys[i][m + n - 1] = sys[i][m + n];

        getBasis(sys, k);

        for (int i = 0; i < m; i++)
            u[i] = sys[i][m + n - 1] / sys[i][i];

        for (int i = 0; i < n - 1; i++)
            v[i] = sys[i + 3][m + n - 1] / sys[i + 3][i + 3];
    }

    static void potentialMethod(int[][] X, int[][] C, int m, int n)
    {
        int N = 0;
        int y = 0;
        int[][] vec = new int[n + m][];
        int[] u = new int[m];
        int[] v = new int[n];

        for (int i = 0; i < n + m; i++)
            vec[i] = new int[m];
        int f = 0;

        while (f == 0)
        {
            f = 1;
            getPotential(X, C, m, n, u, v);
            for (int i = 0; i < m; i++)
                Console.Write($"u = {u[i]}\n");
            for (int j = 0; j < n; j++)
                Console.Write($"\tv = {v[j]}\n");
            Console.Write("\n");
            for (int i = 0; i < m; i++)
                for (int j = 0; j < n; j++)
                    if (f == 1 && X[i][j] < 0)
                    {
                        y = C[i][j] - (u[i] + v[j]);
                        Console.Write($"{i} {j}\t Y = {y}\n", i, j, y);

                        if (y < 0)
                        {
                            getCycle(X, m, n, i, j, vec, N);
                            shiftCycle(X, vec, N);
```

```
                            f = 0;
                        }
                    }
        }
    }
    // Метод вычисления минимального значения целевой функции
    static int calculateZMin(int[][]C, int[][] X, int m, int n)
    {
        int z = 0;

        for (int i = 0; i < m; i++)
            for (int j = 0; j < n; j++)
                if (X[i][j] >= 0)
                    z += X[i][j] * C[i][j];

        return z;
    }

    // Главная функция (точка входа в программу)
public static void Main(string[] args)
    {
        int m = 4, n = 5;
        int[][] C = new int[n][];
        int[] A = new int[] { 14, 14, 12, 16 };
        int[] B = new int[] { 11, 11, 11, 8, 15 };
        int[, ] CC =
                        {
                                { 10, 15, 14, 28, 1 },
                                { 16, 7, 30, 8, 29 },
                                { 1, 21, 22, 19, 12 },
                                { 8, 25, 28, 5, 19 }
                        };

        for (int i = 0; i < m; i++)
            C[i] = new int[n];

        for (int i = 0; i < m; i++)
            for (int j = 0; j < n; j++)
                C[i][j] = CC[i, j];

        int[][] X = null;
        int k = 0, h = 0;

        for (int i = 0; i < m; i++)
            k += A[i];

        for (int i = 0; i < n; i++)
            h += B[i];

        if (k != h)
            Console.WriteLine("Нельзя решить задачу, так как она
открыта\n");
        else
        {
            int g = 0;

            switch (g)
            {
                case 0:
                    X = TransportationProblem.smallestPenaltyMethod(A, B,
C, m, n);
```

```
                    break;

                case 1:
                    X = TransportationProblem.northwestCornerMethod(A, B,
C, m, n);
                    break;
            }
        }

        Console.WriteLine("\n\nОпорное решение:\n");
        TransportationProblem.writeTable(X, C, m, n);
        Console.WriteLine("\nОпорный план невырожденный");
        Console.WriteLine("\nВыберете метод для решения \n0 -
распределительный метод \n1 - метод потенциалов");
        int f = Convert.ToInt32(Console.ReadLine());
        switch (f)
        {
            case 0:
                TransportationProblem.distributionMethod(X, C, m, n);
                break;
            case 1:
                break;
        }
        Console.WriteLine("\nРезультат : \n");
        TransportationProblem.writeTable(X, C, m, n);
        int z = calculateZMin(C, X, m, n);
        Console.WriteLine("Zmin: " + z);
    }
}
```

Результат работы программы:

Опорное решение:

№	B1	B2	B3	B4	B5
1	22	23	16	12	14
2	17	30	1	8	25
3	27	15	13	23	22
4	3	12	21	26	7

Опорный план не вырожден

Результат

№	v_1=10	v_2=7	v_3=5	v_4=12	v_5=14
1	22	23	16	12[15]	14[4]
2	17	30	1[17]	8[2]	25
3	27	15[17]	13	23	22[2]
4	3[17]	12	21	26	7[2]

Вывод: Изучение математической модели транспортной задачи и методов её решения важно для логистики и оптимизации перевозок. Математическая модель помогает анализировать и оптимизировать распределение ресурсов в ограниченных условиях. В ходе исследования рассмотрены основные принципы создания модели, определены ключевые понятия и термины, связанные с транспортной задачей.

Методы решения транспортной задачи, такие как метод потенциалов, метод северо-западного угла, метод минимального элемента и метод пересчёта потенциалов, были изучены и использованы для нахождения оптимального решения. Применение разных методов помогло оценить их эффективность и пригодность для различных ситуаций.