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-233
Мороз Роман Алексеевич

Проверил:
Вирченко Юрий Петрович

Белгород 2025 г.

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

## Постановка задачи

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 одну из следующих ниже задач.

## Вариант 9

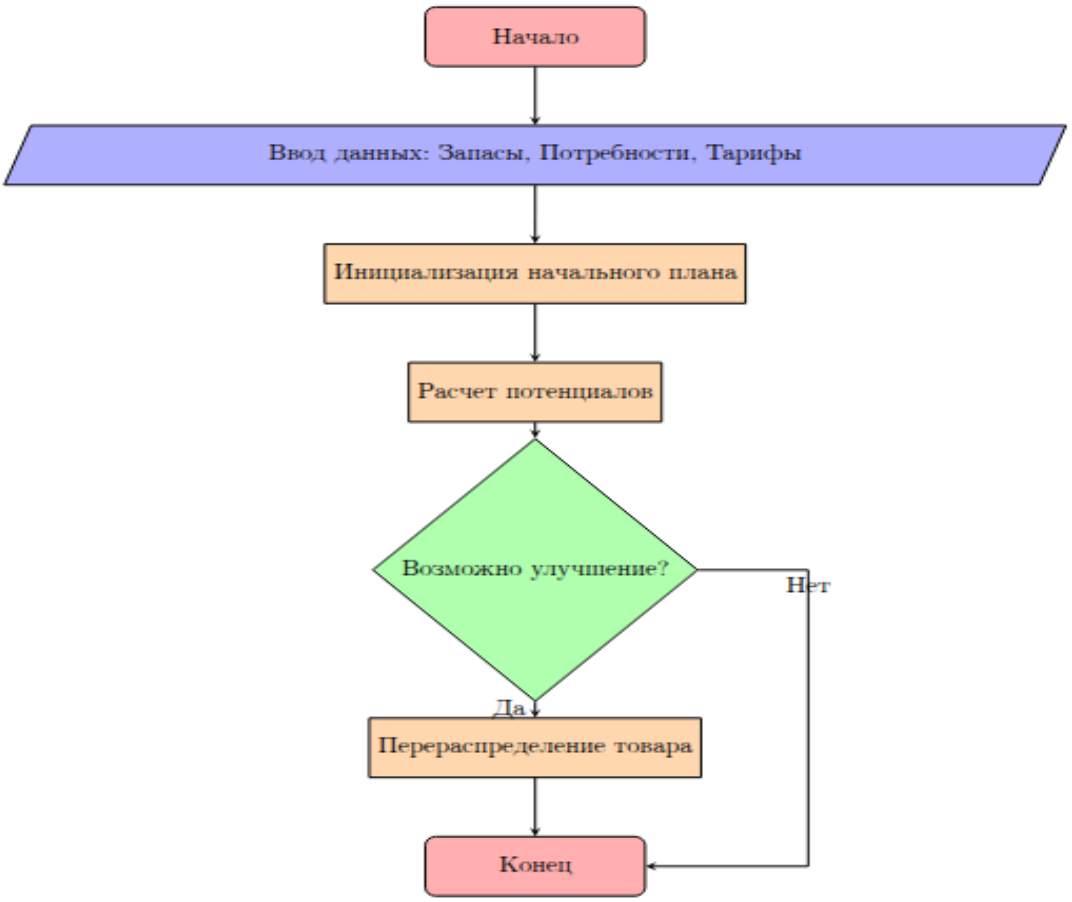
9.

$$\vec{a} = (15, 16, 15, 16);$$

$$\vec{b} = (11, 12, 13, 14, 12);$$

$$C = \begin{pmatrix} 29 & 4 & 8 & 11 & 5 \\ 10 & 19 & 26 & 1 & 27 \\ 16 & 7 & 4 & 29 & 23 \\ 9 & 10 & 24 & 25 & 17 \end{pmatrix}$$

*Блок-схема программы:*




Код программы:

```go
package main


import (
    "fmt"
    "math"
)


// Структура транспортной задачи
type TransportProblem struct {
    supplies []int      // Поставки
    demands  []int      // Потребности
    costs    [][]int    // Стоимость перевозки из одного пункта в другой
    solution [][]int    // Решение задачи
}
```

```go
// Поиск цикла в графе с помощью поиска в глубину
func findCycle(tp *TransportProblem) ([]int, bool) {
   visited := make([]bool, len(tp.supplies)) // отслеживание посещенных узлов
   parent := make([]int, len(tp.supplies))  // отслеживание родительских узлов

   // Для каждого поставщика мы начинаем поиск в глубину
   for i := 0; i < len(tp.supplies); i++ {
       if !visited[i] {
           if cycle := dfs(i, -1, tp, visited, parent); len(cycle) > 0 {
               return cycle, true
           }
       }
   }
   return nil, false
}

// Поиск в глубину для нахождения цикла
func dfs(u, p int, tp *TransportProblem, visited []bool, parent []int) []int {
   visited[u] = true
   parent[u] = p

   for v := 0; v < len(tp.demands); v++ {
       if tp.solution[u][v] > 0 {
           // Если нашли обратный путь
           if visited[v] && v != p {
               // Восстанавливаем цикл
               cycle := []int{}
```

```
                for i := u; i != v; i = parent[i] {
                    cycle = append(cycle, i)
                }
                cycle = append(cycle, v)
                return cycle
            }
            // Рекурсивно продолжаем искать
            if !visited[v] {
                if cycle := dfs(v, u, tp, visited, parent); len(cycle) > 0
{
                    return cycle
                }
            }
        }
    }

    return nil
}

// Метод минимального пересчета для найденного цикла
func getMinFlow(cycle []int, tp *TransportProblem) int {
    // Находим минимальное количество товара в цикле
    minFlow := math.MaxInt

    // Пройдем по всему циклу и найдем минимальное количество товара
    for i := 0; i < len(cycle)-1; i++ {
        u := cycle[i]
        v := cycle[i+1]
        minFlow = min(minFlow, tp.solution[u][v])
    }
```

```
    return minFlow
}

// Функция для нахождения минимума
func min(a, b int) int {
    if a < b {
        return a
    }
    return b
}

// Перераспределение товара по найденному циклу
func redistributeCycle(cycle []int, minFlow int, tp *TransportProblem) {
    for i := 0; i < len(cycle)-1; i++ {
        u := cycle[i]
        v := cycle[i+1]

        // Уменьшаем количество товара на пути
        tp.solution[u][v] -= minFlow
        // Увеличиваем количество товара на обратном пути
        tp.solution[v][u] += minFlow
    }
}

// Основной цикл перераспределения товаров
func optimizeSolution(tp *TransportProblem) {
    for {
        // Ищем цикл
        cycle, found := findCycle(tp)
        if !found {
```

```
            break // Если цикла нет, выходим из цикла
        }

        // Находим минимальное количество товара для перераспределения
        minFlow := getMinFlow(cycle, tp)

        // Перераспределяем товар по найденному циклу
        redistributeCycle(cycle, minFlow, tp)
    }

    // Выводим итоговое решение
    printSolution(tp)
}

// Функция для вывода решения
func printSolution(tp *TransportProblem) {

fmt.Println("\n==========================================================
")
    fmt.Println("РЕШЕНИЕ МЕТОДОМ ПОТЕНЦИАЛОВ")

fmt.Println("==========================================================
")
    totalCost := calculateTotalCost(*tp)
    fmt.Printf("\n Минимальная стоимость: %d\n", totalCost)

    // Заголовки столбцов
    fmt.Printf("\n%-15s", "Поставщики / Потребители")
    for j := 0; j < len(tp.demands); j++ {
        fmt.Printf("%-10s", fmt.Sprintf("D%d", j+1))
    }
    fmt.Println()
```

```
    // Данные для поставщиков
    for i := 0; i < len(tp.supplies); i++ {
        // Заголовок строки для поставщика
        fmt.Printf("P%d%-10s", i+1, "| ")

        // Выводим распределение товара
        for j := 0; j < len(tp.demands); j++ {
            fmt.Printf("%-10d", tp.solution[i][j])
        }
        fmt.Println()
    }

    // Итоги
fmt.Println("\n==========================================================
")
    fmt.Println("РЕШЕНИЕ РАСПРЕДЕЛЬТЕЛЬНЫМ МЕТОДОМ")
fmt.Println("==========================================================
")
    fmt.Printf("\n Минимальная стоимость: %d\n", totalCost)
}

// Функция для расчета общей стоимости
func calculateTotalCost(tp TransportProblem) int {
    totalCost := 0
    for i := 0; i < len(tp.supplies); i++ {
        for j := 0; j < len(tp.demands); j++ {
            totalCost += tp.solution[i][j] * tp.costs[i][j]
        }
    }
```

```
    return totalCost
}


// Основная функция
func main() {
    // Исходные данные из задачи
    tp := TransportProblem{
        supplies: []int{15, 16, 15, 16},
        demands:  []int{11, 12, 13, 14, 12},
        costs: [][]int{
            {29, 4, 8, 11, 5},
            {10, 19, 26, 1, 27},
            {16, 7, 4, 29, 23},
            {9, 10, 24, 25, 17},
        },
        solution: [][]int{
            {0, 0, 0, 0, 0},
            {0, 0, 0, 0, 0},
            {0, 0, 0, 0, 0},
            {0, 0, 0, 0, 0},
        },
    }

    // Решение транспортной задачи методом циклов
    optimizeSolution(&tp)
}
```

*Результат работы программы:*

```
================================================
РЕШЕНИЕ МЕТОДОМ ПОТЕНЦИАЛОВ
================================================

Начальный опорный план:
Поставщик  |   1 |   2 |   3 |   4 |   5
-----------------------------------------
A          |  11 |   4 |   0 |   0 |   0
B          |   0 |   8 |   8 |   0 |   0
C          |   0 |   0 |   5 |  10 |   0
D          |   0 |   0 |   0 |   4 |  12
Стоимость: 1021

Оптимальный план после корректировки:
Поставщик  |   1 |   2 |   3 |   4 |   5
-----------------------------------------
A          |  11 |   4 |   0 |   0 |   0
B          |   0 |  13 |   3 |   0 |   0
C          |   0 |   5 |   0 |  10 |   0
D          |   0 |   0 |   0 |   4 |  12
Минимальная стоимость: 971

================================================
РЕШЕНИЕ РАСПРЕДЕЛИТЕЛЬНЫМ МЕТОДОМ
================================================

Оптимальный план:
Поставщик  |   1 |   2 |   3 |   4 |   5
-----------------------------------------
A          |   0 |  12 |   0 |   0 |   3
B          |   0 |   0 |   0 |  14 |   2
C          |   0 |   0 |  13 |   0 |   2
D          |  11 |   0 |   0 |   0 |   5
Минимальная стоимость: 971
```

*Аналитическое решение методом потенциалов и распределительным методом:*

### *Исходные данные*

### *Поставщики и запасы (a):*

- *A: 15*
- *B: 16*
- *C: 15*
- *D: 16*

### *Потребители и потребности (b):*

- *1: 11*
- *2: 12*
- *3: 13*
- *4: 14*
- *5: 12*

### *Матрица тарифов (C):*

| Поставщик \ Потребитель | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| A | 29 | 4 | 8 | 11 | 5 |
| B | 10 | 19 | 26 | 1 | 27 |
| C | 16 | 7 | 4 | 29 | 23 |
| D | 9 | 10 | 24 | 25 | 17 |

### *Часть 1. Решение методом потенциалов*

### *Шаг 1. Начальный опорный план*

| Поставщик \ Потребитель | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| A | 11 | 4 | - | - | - |
| B | - | 8 | 8 | - | - |
| C | - | - | 5 | 10 | - |
| D | - | - | - | 4 | 12 |

**Стоимость плана:**

$$11 \times 29 + 4 \times 4 + 8 \times 19 + 8 \times 26 + 5 \times 4 + 10 \times 29 + 4 \times 25 + 12 \times 17 = 1021$$

**Шаг 2. Проверка оптимальности потенциалами**

**Вычисление потенциалов**

Задаём и решаем систему:

$$\begin{cases} u_A + v_1 = 29 \\ u_A + v_2 = 4 \\ u_B + v_2 = 19 \\ u_B + v_3 = 26 \\ u_C + v_3 = 4 \\ u_C + v_4 = 29 \\ u_D + v_4 = 25 \\ u_D + v_5 = 17 \end{cases}$$

После вычислений получаем:

$$\begin{cases} u_A = 0, v_1 = 29, v_2 = 4 \\ u_B = 15, v_3 = 11 \\ u_C = -7, v_4 = 36 \\ u_D = -11, v_5 = 28 \end{cases}$$

**Вычисление оценок свободных клеток**

$\Delta_{C2} = 7 - (-7 + 4) = 10 > 0$ → клетка (C,2) неоптимальна.

**Коррекция плана**

Находим цикл: C2 → C3 → B3 → B2 → C2.

Минимальный элемент .

$\theta = \min(5, 8) = 5.$

**Новый план:**

| Поставщик \ Потребитель | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| A | 11 | 4 | - | - | - |
| B | - | 13 | 3 | - | - |
| C | - | 5 | - | 10 | - |
| D | - | - | - | 4 | 12 |

**Новая стоимость:** 971

Проверяем потенциалы — все $\Delta ij <= 0$, значит **план оптимален**!

## Часть 2. Решение распределительным методом (метод минимального элемента)

### Шаг 1. Выбор минимального тарифа

Находим минимальный элемент в матрице тарифов: это **1** (B4).

### Шаг 2. Заполняем перевозки

| Поставщик \ Потребитель | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| A | - | 12 | - | - | 3 |
| B | - | - | - | 14 | 2 |
| C | - | - | 13 | - | 2 |
| D | 11 | - | - | - | 5 |

### Шаг 3. Подсчёт стоимости

$$12 \times 4 + 3 \times 5 + 14 \times 1 + 2 \times 27 + 13 \times 4 + 2 \times 23 + 11 \times 9 + 5 \times 17 = 971$$

### Шаг 4. Проверка оптимальности

Для проверки оптимальности используем метод потенциалов. Определяем потенциалы $\nu j$ и $\upsilon i$ :

$$\begin{cases} u_A = 0, v_2 = 4 \\ u_A + v_5 = 5 \\ u_B + v_4 = 1 \\ u_B + v_5 = 27 \\ u_C + v_3 = 4 \\ u_C + v_5 = 23 \\ u_D + v_1 = 9 \\ u_D + v_5 = 17 \end{cases}$$

После вычислений:

$$\begin{cases} u_A = 0, v_2 = 4, v_5 = 5 \\ u_B = -13, v_4 = 14, v_5 = 27 \\ u_C = -9, v_3 = 13, v_5 = 23 \\ u_D = -8, v_1 = 9, v_5 = 17 \end{cases}$$

Оцениваем свободные клетки: все $\Delta ij \ <= \ 0$ , значит решение **оптимально**!

**Вывод**: в ходе выполнения лабораторной работы были изучены математическая модель транспортной задачи и методы решения этой задачи.

Проведённые задания по подготовке к работе помогли разработать программу для решения транспортной задачи методом потенциалов и распределительным методом. Программа преобразует исходные данные в таблицу, определяет оптимальное решение путём выбора опорного решения и выполнения шагов методов, и наконец, записывает значение минимальной стоимости. Результатом работы программы являются минимальная стоимость, где функция достигает оптимального значения. Была создана программа, которая правильно реализует метод потенциалов и распределительный метод, что в свою очередь является главным в данной лабораторной работе.