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-231
Столяров Захар

Проверил:
Вирченко Юрий Петрович

Белгород, 2025 г.

Лабораторная работа №4 «Закрытая транспортная задача.»

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

Вариант 14

Задания для подготовки к работе

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. В рамках подготовки тестовых данных решить аналитически следующую задачу:

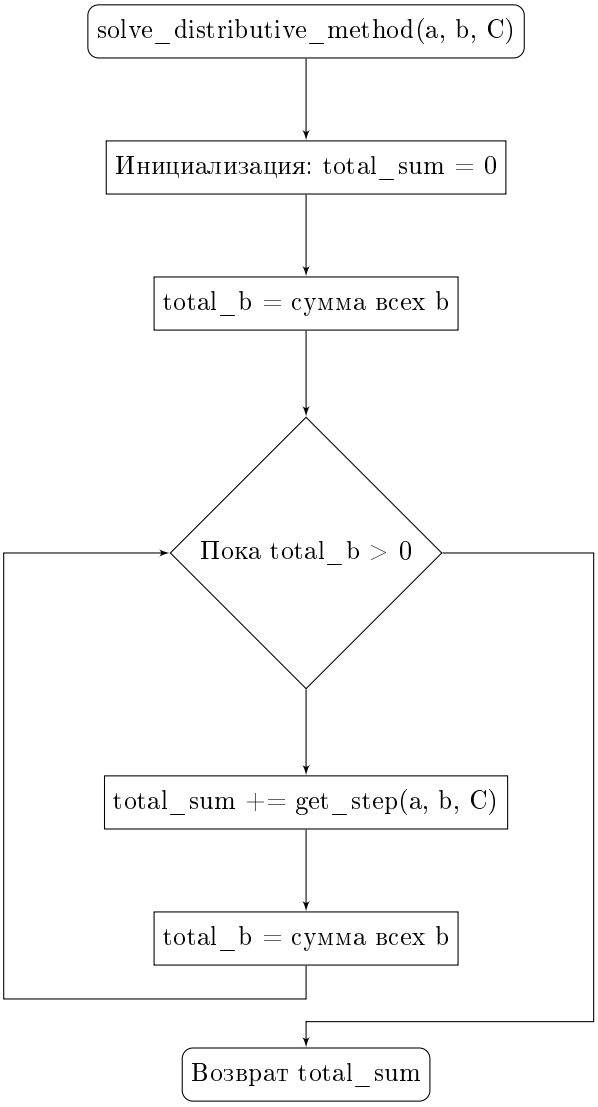
$$\vec{a} = (23, 24, 21, 15);$$

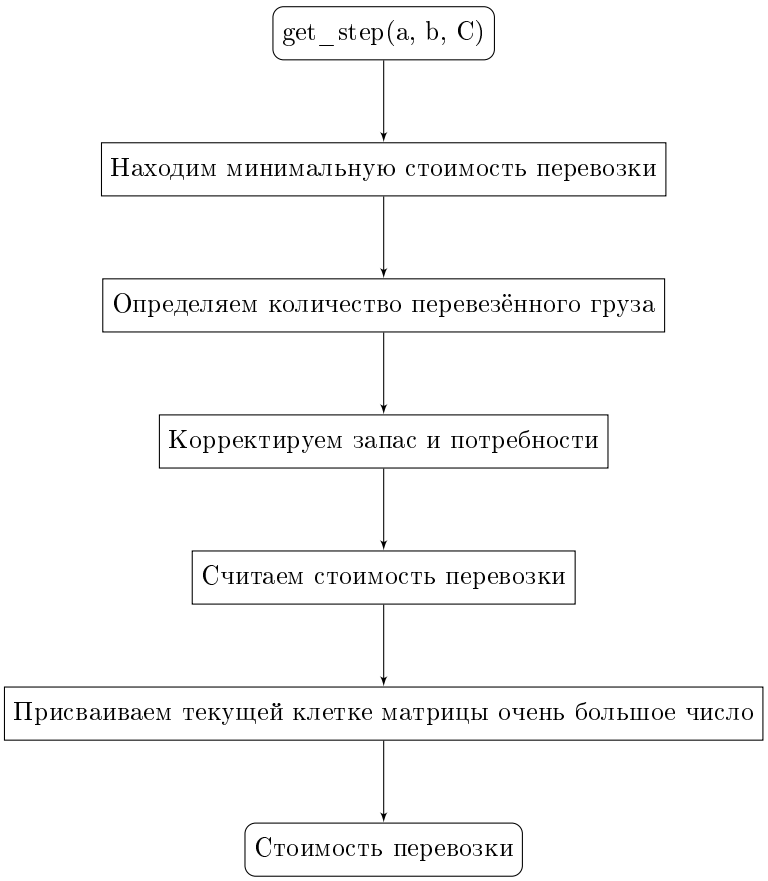
$$\vec{b} = (19, 16, 16, 16, 16);$$

$$\begin{pmatrix} 8 & 28 & 17 & 19 & 11 \\ 27 & 5 & 10 & 6 & 19 \\ 29 & 11 & 3 & 7 & 8 \\ 25 & 16 & 19 & 24 & 13 \end{pmatrix}$$

Программное решение

Блок-схемы основных функций программ:




get_step(a, b, C)
Находим минимальную стоимость перевозки
Определяем количество перевезённого груза
Корректируем запас и потребности
Считаем стоимость перевозки
Присваиваем текущей клетке матрицы очень большое число
Стоимость перевозки


find_cycle(basis, entering_cell)
Строим граф базисных клеток
Добавляем входящую клетку в граф
Начиная со строки входящей клетки ищем цикл
Преобразуем цикл в координаты клеток
Цикл пересчёта


solve_potentials_method (a, b, C)
Считаем количество поставщиков и потребителей
Строим начальный опорный план
Пока сумма потребностей > 0
Находим клетку с минимальной стоимостью
Заполняем клетку
Пока план не является оптимальным
Находим потенциалы и задаём базис
Создаём систему уравнений для потенциалов
Решаем систему уравнений для потенциалов
Проверяем оптимальность
Ищем цикл пересчёта
Находим минимальное значение в минусовых клетках
Корректируем план
Оптимальный план

Распределительный метод

Код программы:

```cpp
#include <iostream>
#include <vector>
#include <climits>

using namespace std;

int get_step(vector<int>& a, vector<int>& b, vector<vector<int>>& C) {
    int min_el = C[0][0];
    pair<int, int> min_el_ind = { 0, 0 };

    // Находим минимальную стоимость перевозки
    for (size_t i = 0; i < a.size(); ++i) {
        for (size_t j = 0; j < b.size(); ++j) {
            if (C[i][j] < min_el) {
                min_el = C[i][j];
                min_el_ind = { i, j };
            }
        }
    }

    // Количество перевезённого груза
    int sum = min(a[min_el_ind.first], b[min_el_ind.second]);

    // Корректируем запас и потребность
    a[min_el_ind.first] -= sum;
    b[min_el_ind.second] -= sum;

    // Считаем стоимость перевозки
    sum *= C[min_el_ind.first][min_el_ind.second];

    // Помечаем клетку как использованную
    C[min_el_ind.first][min_el_ind.second] = INT_MAX;

    return sum;
}

int solve_distributive_method(vector<int>& a, vector<int>& b, vector<vector<int>>& C) {
    int total_sum = 0;

    // Выполняем шаги пока есть потребности
    int total_b = 0;
    for (int val : b) total_b += val;

```

```
    while (total_b > 0) {
        total_sum += get_step(a, b, C);
        total_b = 0;
        for (int val : b) total_b += val;
    }

    return total_sum;
}

int main() {
    setlocale(LC_ALL, "Russian");

    vector<int> a = { 23, 24, 21, 15 };        // Вектор запасов
    vector<int> b = { 19, 16, 16, 16, 16 };    // Вектор потребностей
    vector<vector<int>> C = {                  // Матрица стоимостей
        {8, 28, 17, 19, 11},
        {27, 5, 10, 6, 19},
        {29, 11, 3, 7, 8},
        {25, 16, 19, 24, 13}
    };

    int result = solve_distributive_method(a, b, C);
    cout << "Минимальная стоимость перевозок: " << result << endl;

    return 0;
}
```

Результат работы программы:

Метод потенциалов

Код программы:

```cpp
#include <iostream>
#include <vector>
#include <limits>
#include <algorithm>
#include <unordered_map>
#include <unordered_set>
#include <stack>
#include <string>
#include <sstream>

using namespace std;

const int INF = numeric_limits<int>::max();

struct Cell {
    int i, j;
};

vector<Cell> find_cycle(const vector<Cell>& basis, Cell entering_cell) {
    unordered_map<string, vector<string>> graph;
    unordered_map<string, string> parent;
    unordered_set<string> visited;

    // Строим граф
    for (size_t idx = 0; idx < basis.size(); ++idx) {
        const Cell& cell = basis[idx];
        string row = "r" + std::to_string(cell.i);
        string col = "c" + std::to_string(cell.j);
        graph[row].push_back(col);
        graph[col].push_back(row);
    }

    // Добавляем входящую клетку
    string start_row = "r" + std::to_string(entering_cell.i);
    string start_col = "c" + std::to_string(entering_cell.j);
    graph[start_row].push_back(start_col);
    graph[start_col].push_back(start_row);

    // Ищем цикл
    stack<pair<string, string>> s;
    s.push(make_pair(start_row, ""));
    vector<string> cycle;

```

```
    while (!s.empty()) {
        pair<string, string> node_pair = s.top();
        s.pop();
        string node = node_pair.first;
        string prev = node_pair.second;

        if (visited.count(node)) {
            // Нашли цикл
            string current = node;
            while (current != "") {
                cycle.push_back(current);
                current = parent[current];
            }
            reverse(cycle.begin(), cycle.end());
            break;
        }

        visited.insert(node);
        parent[node] = prev;

        for (size_t i = 0; i < graph[node].size(); ++i) {
            const string& neighbor = graph[node][i];
            if (neighbor != prev) {
                s.push(make_pair(neighbor, node));
            }
        }
    }

    // Преобразуем в координаты клеток
    vector<Cell> cycle_coords;
    for (size_t i = 0; i < cycle.size(); ++i) {
        if (cycle[i][0] == 'r') {
            if (i + 1 < cycle.size()) {
                string row_str = cycle[i].substr(1);
                string col_str = cycle[i + 1].substr(1);
                int row = atoi(row_str.c_str());
                int col = atoi(col_str.c_str());
                cycle_coords.push_back({ row, col });
            }
        }
    }

    return cycle_coords;
}

vector<vector<int>> solve_potentials_method(vector<int> a, vector<int> b, vector<vector<int>> C) {
    int m = static_cast<int>(a.size());
```

```
    int n = static_cast<int>(b.size());
    vector<vector<int>> X(m, vector<int>(n, 0));
    vector<int> remaining_a = a;
    vector<int> remaining_b = b;

    // Построение начального опорного плана методом минимальной стоимости
    while (true) {
        bool has_remaining = false;
        for (size_t i = 0; i < remaining_a.size(); ++i) {
            if (remaining_a[i] > 0) {
                has_remaining = true;
                break;
            }
        }
        if (!has_remaining) break;

        int min_cost = INF;
        Cell min_cell = { -1, -1 };

        for (int i = 0; i < m; ++i) {
            for (int j = 0; j < n; ++j) {
                if (remaining_a[i] > 0 && remaining_b[j] > 0 && C[i][j] < min_cost) {
                    min_cost = C[i][j];
                    min_cell.i = i;
                    min_cell.j = j;
                }
            }
        }

        if (min_cell.i == -1) break;

        int amount = min(remaining_a[min_cell.i], remaining_b[min_cell.j]);
        X[min_cell.i][min_cell.j] = amount;
        remaining_a[min_cell.i] -= amount;
        remaining_b[min_cell.j] -= amount;
    }

    // Метод потенциалов
    int iteration = 0;
    const int max_iterations = 100;

    while (iteration++ < max_iterations) {
        vector<int> u(m, 0);
        vector<int> v(n, 0);
        vector<Cell> basis;

        // Собираем базисные клетки
```

```
        for (int i = 0; i < m; ++i) {
            for (int j = 0; j < n; ++j) {
                if (X[i][j] > 0) {
                    Cell cell;
                    cell.i = i;
                    cell.j = j;
                    basis.push_back(cell);
                }
            }
        }

        // Вычисляем потенциалы
        bool changed;
        do {
            changed = false;
            for (size_t idx = 0; idx < basis.size(); ++idx) {
                const Cell& cell = basis[idx];
                if (u[cell.i] != 0 && v[cell.j] == 0) {
                    v[cell.j] = C[cell.i][cell.j] - u[cell.i];
                    changed = true;
                }
                else if (v[cell.j] != 0 && u[cell.i] == 0) {
                    u[cell.i] = C[cell.i][cell.j] - v[cell.j];
                    changed = true;
                }
            }
        } while (changed);

        // Проверка оптимальности
        bool optimal = true;
        Cell entering_cell = { -1, -1 };
        int min_delta = 0;

        for (int i = 0; i < m; ++i) {
            for (int j = 0; j < n; ++j) {
                if (X[i][j] == 0) {
                    int delta = C[i][j] - (u[i] + v[j]);
                    if (delta < min_delta) {
                        min_delta = delta;
                        optimal = false;
                        entering_cell.i = i;
                        entering_cell.j = j;
                    }
                }
            }
        }

```

```
        if (optimal) break;

        // Находим цикл пересчета
        vector<Cell> cycle = find_cycle(basis, entering_cell);

        // Находим минимальное значение в минусовых клетках
        int theta = INF;
        for (size_t i = 1; i < cycle.size(); i += 2) {
            theta = min(theta, X[cycle[i].i][cycle[i].j]);
        }

        // Корректируем план
        for (size_t i = 0; i < cycle.size(); ++i) {
            int row = cycle[i].i;
            int col = cycle[i].j;
            if (i % 2 == 0) { // Плюсовые клетки
                X[row][col] += theta;
            }
            else { // Минусовые клетки
                X[row][col] -= theta;
            }
        }
    }

    return X;
}

int main() {
    setlocale(LC_ALL, "Russian");

    vector<int> a = { 23, 24, 21, 15 };
    vector<int> b = { 19, 16, 16, 16, 16 };
    vector<vector<int>> C = {
        {8, 28, 17, 19, 11},
        {27, 5, 10, 6, 19},
        {29, 11, 3, 7, 8},
        {25, 16, 19, 24, 13}
    };

    vector<vector<int>> solution = solve_potentials_method(a, b, C);

    // Вычисляем общую стоимость
    int total_cost = 0;
    for (size_t i = 0; i < solution.size(); ++i) {
        for (size_t j = 0; j < solution[i].size(); ++j) {
            total_cost += solution[i][j] * C[i][j];
        }
```

```
    }

    cout << "Минимальная стоимость перевозок: " << total_cost << endl;

    return 0;
}
```

Результат работы программы:

Аналитическое решение

1. **Теоретические сведения**

   - Имеется 4 поставщика с запасами $a = (23, 24, 21, 15)$
   - Имеется 5 потребителей с потребностями $b = (19, 16, 16, 16, 16)$
   - Матрица стоимостей перевозок:

$$C = \begin{pmatrix} 8 & 28 & 17 & 19 & 11 \\ 27 & 5 & 10 & 6 & 19 \\ 29 & 11 & 3 & 7 & 8 \\ 25 & 16 & 19 & 24 & 13 \end{pmatrix}$$

2. **Решение методом минимальной стоимости**

   (a) Пошаговое решение:

   - Шаг 1: Клетка (3,3), стоимость 3, перевозим 16 ед.
     Стоимость: $16 \times 3 = 48$
   - Шаг 2: Клетка (2,4), стоимость 6, перевозим 16 ед.
     Стоимость: $48 + 16 \times 6 = 144$
   - Шаг 3: Клетка (2,2), стоимость 5, перевозим 8 ед.
     Стоимость: $144 + 8 \times 5 = 184$
   - Шаг 4: Клетка (4,5), стоимость 13, перевозим 15 ед.
     Стоимость: $184 + 15 \times 13 = 379$
   - Шаг 5: Клетка (1,1), стоимость 8, перевозим 19 ед.
     Стоимость: $379 + 19 \times 8 = 531$
   - Шаг 6: Клетка (1,5), стоимость 11, перевозим 4 ед.
     Стоимость: $531 + 4 \times 11 = 575$
   - Шаг 7: Клетка (3,5), стоимость 8, перевозим 1 ед.
     Стоимость: $575 + 1 \times 8 = 583$
   - Шаг 8: Клетка (1,3), стоимость 17, перевозим 3 ед.
     Стоимость: $583 + 3 \times 17 = 634$
   - Шаг 9: Клетка (2,3), стоимость 10, перевозим 1 ед.
     Стоимость: $634 + 1 \times 10 = \boxed{635}$

   (b) Таблица итогового распределения:

	$b_1 = 19$	$b_2 = 16$	$b_3 = 16$	$b_4 = 16$	$b_5 = 16$	Запасы
$a_1 = 23$	**19**	–	**3**	–	**4**	0
$a_2 = 24$	–	**8**	**1**	**16**	–	0
$a_3 = 21$	–	–	**16**	–	**1**	4
$a_4 = 15$	–	–	–	–	**15**	0

   (c) Общая минимальная стоимость перевозок: $\boxed{635}$

Вывод: результат аналитического решения совпадает с результатами написанных мной обеих программ, что подтверждает корректность моего решения методом минимальной стоимости, методом потенциалов и распределительным методом.