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 ВТ-231

Масленников Даниил

Проверил:

Вирченко Юрий Петрович

Белгород 2025 г.

**Цель работы**: изучить математическую модель транспортной

задачи, овладеть методами решения этой задачи.

**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.

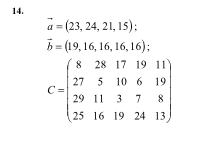
**2**. Составить и отладить программы решения транспортной задачи

распределительным методом и методом потенциалов.

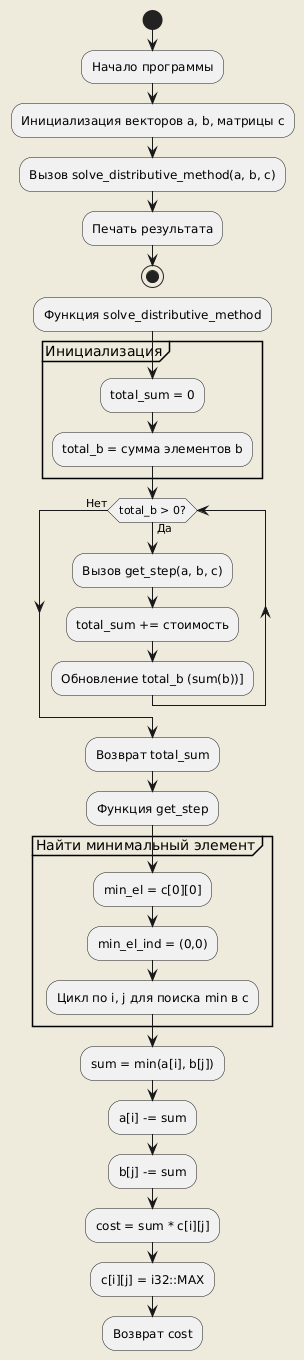
**3**. Для подготовки тестовых данных решить вручную одну из

следующих ниже задач.

# Вариант – 14

****

**Распределительный метод**

**Блок схема программы:**

**Код программы:**

**use std::i32;**

**fn get\_step(a: &mut Vec<i32>, b: &mut Vec<i32>, c: &mut Vec<Vec<i32>>) -> i32 {**

**let mut min\_el = c[0][0];**

**let mut min\_el\_ind = (0, 0);**

**// Находим минимальную стоимость перевозки**

**for i in 0..a.len() {**

**for j in 0..b.len() {**

**if c[i][j] < min\_el {**

**min\_el = c[i][j];**

**min\_el\_ind = (i, j);**

**}**

**}**

**}**

**// Количество перевозимого груза**

**let sum = a[min\_el\_ind.0].min(b[min\_el\_ind.1]);**

**// Обновляем запас и потребность**

**a[min\_el\_ind.0] -= sum;**

**b[min\_el\_ind.1] -= sum;**

**// Считаем стоимость**

**let cost = sum \* c[min\_el\_ind.0][min\_el\_ind.1];**

**// Помечаем клетку как использованную**

**c[min\_el\_ind.0][min\_el\_ind.1] = i32::MAX;**

**cost**

**}**

**fn solve\_distributive\_method(mut a: Vec<i32>, mut b: Vec<i32>, mut c: Vec<Vec<i32>>) -> i32 {**

**let mut total\_sum = 0;**

**// Вычисляем сумму всех потребностей**

**let mut total\_b: i32 = b.iter().sum();**

**while total\_b > 0 {**

**total\_sum += get\_step(&mut a, &mut b, &mut c);**

**total\_b = b.iter().sum();**

**}**

**total\_sum**

**}**

**fn main() {**

**let a = vec![23, 24, 21, 15]; // Вектор запасов**

**let b = vec![19, 16, 16, 16, 16]; //Вектор потребностей**

**let c = vec![**

**vec![8, 28, 17, 19, 11],**

**vec![27, 5, 10, 6, 19],**

**vec![29, 11, 3, 7, 8],**

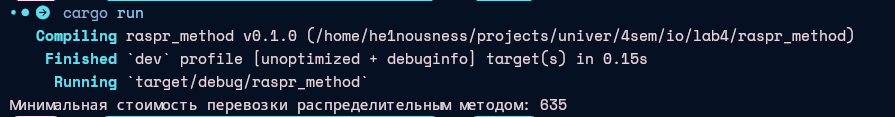
**vec![25, 16, 19, 24, 13],**

**]; // Матрица стоимостей**

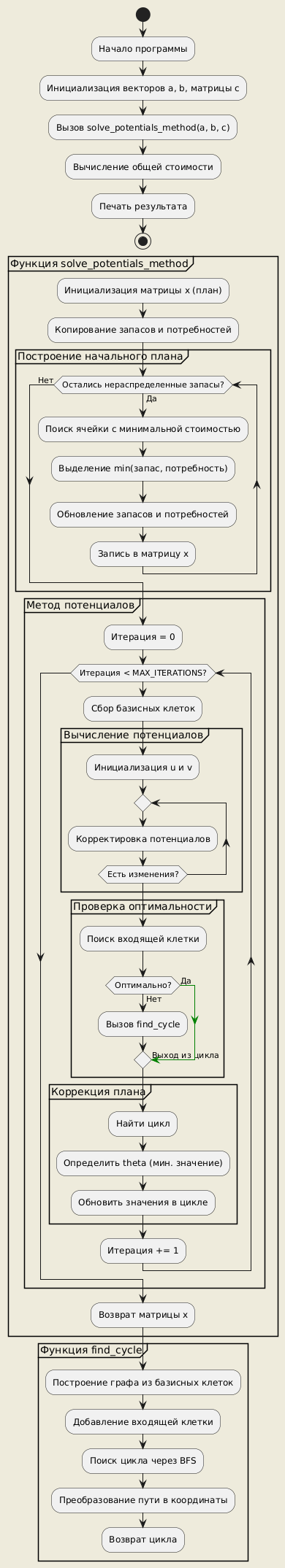
**let result = solve\_distributive\_method(a, b, c);**

**println!("Минимальная стоимость перевозки распределительным методом: {}", result);**

**}**



**Потенциальный метод  
Блок схема программы:**



**Код программы:  
use std::collections::{HashMap, HashSet, VecDeque};**

**use std::i32;**

**const INF: i32 = i32::MAX;**

**#[derive(Clone, Copy, Debug)]**

**struct Cell {**

**i: usize,**

**j: usize,**

**}**

**fn find\_cycle(basis: &[Cell], entering\_cell: Cell) -> Vec<Cell> {**

**let mut graph: HashMap<String, Vec<String>> = HashMap::new();**

**let mut parent: HashMap<String, String> = HashMap::new();**

**let mut visited: HashSet<String> = HashSet::new();**

**// Строим граф**

**for &cell in basis {**

**let row = format!("r{}", cell.i);**

**let col = format!("c{}", cell.j);**

**graph**

**.entry(row.clone())**

**.or\_insert\_with(Vec::new)**

**.push(col.clone());**

**graph.entry(col).or\_insert\_with(Vec::new).push(row);**

**}**

**// Добавляем входящую ячейку**

**let start\_row = format!("r{}", entering\_cell.i);**

**let start\_col = format!("c{}", entering\_cell.j);**

**graph**

**.entry(start\_row.clone())**

**.or\_insert\_with(Vec::new)**

**.push(start\_col.clone());**

**graph**

**.entry(start\_col)**

**.or\_insert\_with(Vec::new)**

**.push(start\_row.clone());**

**// Ищем цикл**

**let mut stack = VecDeque::new();**

**stack.push\_back((start\_row.clone(), String::new()));**

**let mut cycle = Vec::new();**

**while let Some((node, prev)) = stack.pop\_front() {**

**if visited.contains(&node) {**

**// Нашли цикл**

**let mut current = node.clone();**

**while !current.is\_empty() {**

**cycle.push(current.clone());**

**current = parent[&current].clone();**

**}**

**cycle.reverse();**

**break;**

**}**

**visited.insert(node.clone());**

**parent.insert(node.clone(), prev.clone());**

**if let Some(neighbors) = graph.get(&node) {**

**for neighbor in neighbors {**

**if neighbor != &prev {**

**stack.push\_back((neighbor.clone(), node.clone()));**

**}**

**}**

**}**

**}**

**// Преобразуем в координаты ячеек**

**let mut cycle\_coords = Vec::new();**

**for i in 0..cycle.len() {**

**if cycle[i].starts\_with('r') {**

**if i + 1 < cycle.len() {**

**let row\_str = &cycle[i][1..];**

**let col\_str = &cycle[i + 1][1..];**

**let row: usize = row\_str.parse().unwrap();**

**let col: usize = col\_str.parse().unwrap();**

**cycle\_coords.push(Cell { i: row, j: col });**

**}**

**}**

**}**

**cycle\_coords**

**}**

**fn solve\_potentials\_method(a: Vec<i32>, b: Vec<i32>, c: Vec<Vec<i32>>) -> Vec<Vec<i32>> {**

**let m = a.len();**

**let n = b.len();**

**let mut x = vec![vec![0; n]; m];**

**let mut remaining\_a = a.clone();**

**let mut remaining\_b = b.clone();**

**// Построение начального допустимого плана минимальной стоимости**

**loop {**

**let mut has\_remaining = false;**

**for &val in &remaining\_a {**

**if val > 0 {**

**has\_remaining = true;**

**break;**

**}**

**}**

**if !has\_remaining {**

**break;**

**}**

**let mut min\_cost = INF;**

**let mut min\_cell = Cell {**

**i: usize::MAX,**

**j: usize::MAX,**

**};**

**for i in 0..m {**

**for j in 0..n {**

**if remaining\_a[i] > 0 && remaining\_b[j] > 0 && c[i][j] < min\_cost {**

**min\_cost = c[i][j];**

**min\_cell = Cell { i, j };**

**}**

**}**

**}**

**if min\_cell.i == usize::MAX {**

**break;**

**}**

**let amount = remaining\_a[min\_cell.i].min(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**

**let mut iteration = 0;**

**const MAX\_ITERATIONS: i32 = 100;**

**while iteration < MAX\_ITERATIONS {**

**iteration += 1;**

**let mut u = vec![0; m];**

**let mut v = vec![0; n];**

**let mut basis = Vec::new();**

**// Собираем базисные клетки**

**for i in 0..m {**

**for j in 0..n {**

**if x[i][j] > 0 {**

**basis.push(Cell { i, j });**

**}**

**}**

**}**

**// Вычисляем потенциалы**

**let mut changed;**

**loop {**

**changed = false;**

**for &cell in &basis {**

**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;**

**}**

**}**

**if !changed {**

**break;**

**}**

**}**

**// Проверка оптимальности**

**let mut optimal = true;**

**let mut entering\_cell = Cell {**

**i: usize::MAX,**

**j: usize::MAX,**

**};**

**let mut min\_delta = 0;**

**for i in 0..m {**

**for j in 0..n {**

**if x[i][j] == 0 {**

**let delta = c[i][j] - (u[i] + v[j]);**

**if delta < min\_delta {**

**min\_delta = delta;**

**optimal = false;**

**entering\_cell = Cell { i, j };**

**}**

**}**

**}**

**}**

**if optimal {**

**break;**

**}**

**// Находим цикл**

**let cycle = find\_cycle(&basis, entering\_cell);**

**// Находим минимальное ограничение в цикле**

**let mut theta = INF;**

**for i in (1..cycle.len()).step\_by(2) {**

**theta = theta.min(x[cycle[i].i][cycle[i].j]);**

**}**

**// Обновляем план**

**for i in 0..cycle.len() {**

**let row = cycle[i].i;**

**let col = cycle[i].j;**

**if i % 2 == 0 {**

**x[row][col] += theta;**

**} else {**

**x[row][col] -= theta;**

**}**

**}**

**}**

**x**

**}**

**fn main() {**

**let a = vec![23, 24, 21, 15];**

**let b = vec![19, 16, 16, 16, 16];**

**let c = vec![**

**vec![8, 28, 17, 19, 11],**

**vec![27, 5, 10, 6, 19],**

**vec![29, 11, 3, 7, 8],**

**vec![25, 16, 19, 24, 13],**

**];**

**let solution = solve\_potentials\_method(a, b, c.clone());**

**// Вычисляем общую стоимость**

**let total\_cost: i32 = solution**

**.iter()**

**.enumerate()**

**.map(|(i, row)| {**

**row.iter()**

**.enumerate()**

**.map(|(j, &val)| val \* c[i][j])**

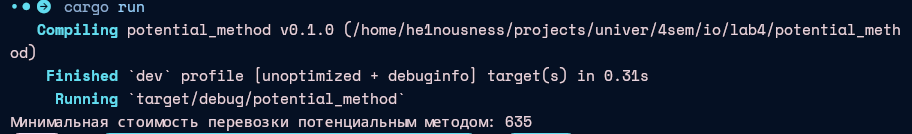
**.sum::<i32>()**

**})**

**.sum();**

**println!("Минимальная стоимость перевозки потенциальным методом: {}", total\_cost);**

**}**



**Аналитическое решение:**

### **Решение транспортной задачи методом потенциалов**

#### 1. ****Построение начального опорного плана****

Выберем клетки с минимальными стоимостями последовательно, соблюдая баланс запасов и потребностей.

**Шаг 1:**  
Клетка (3,3) с минимальной стоимостью c33​=3.  
Перевозим x33​=min(21,16)=16.  
Обновляем:

* a3​=21−16=5
* b3​=16−16=0.

**Шаг 2:**  
Клетка (2,4) с c24​=6.  
Перевозим x24​=min(24,16)=16.  
Обновляем:

* a2​=24−16=8
* b4​=16−16=0.

**Шаг 3:**  
Клетка (2,2) с c22​=5.  
Перевозим x22​=min(8,16)=8.  
Обновляем:

* a2​=8−8=0
* b2​=16−8=8.

**Шаг 4:**  
Клетка (1,1) с c11​=8.  
Перевозим x11​=min(23,19)=19.  
Обновляем:

* a1​=23−19=4
* b1​=19−19=0.

**Шаг 5:**  
Клетка (1,5) с c15​=11.  
Перевозим x15​=min(4,16)=4.  
Обновляем:

* a1​=4−4=0
* b5​=16−4=12.

**Шаг 6:**  
Клетка (3,5) с c35​=8.  
Перевозим x35​=min(5,12)=5.  
Обновляем:

* a3​=5−5=0
* b5​=12−5=7.

**Шаг 7:**  
Клетка (4,5) с c45​=13.  
Перевозим x45​=min(15,7)=7.  
Обновляем:

* a4​=15−7=8
* b5​=7−7=0.

****Шаг 8:****  
Оставшийся запас a4​=8 распределяем в клетку (4,4) с c44​=24.  
Перевозим x44​=8.

#### ****2. Начальный опорный план****

|  | b1=19 | b2=16 | b3=16 | b4=16 | b5=16 | Запасы |
| --- | --- | --- | --- | --- | --- | --- |
| a1=23 | 19 | – | – | – | 4 | 0 |
| a2=24 | – | 8 | – | 16 | – | 0 |
| a3=21 | – | – | 16 | – | 5 | 0 |
| a4=15 | – | – | – | 8 | 7 | 0 |

#### ****3. Расчёт потенциалов****

Для базисных клеток (xij​>0):  
ui​+vj​=cij​.

* Присвоим u1​=0.
* Из клетки (1,1): 0+v1​=8 → v1​=8.
* Из клетки (1,5): 0+v5​=11 → v5​=11.
* Из клетки (2,2): u2​+v2​=5.
* Из клетки (2,4): u2​+v4​=6.
* Из клетки (3,3): u3​+v3​=3.
* Из клетки (3,5): u3​+v5​=8 → u3​=8−11=−3.
* Из клетки (4,4): u4​+v4​=24.
* Из клетки (4,5): u4​+v5​=13 → u4​=13−11=2.

#### ****4. Проверка оптимальности****

Для свободных клеток вычислим оценки:  
Δij​=cij​−(ui​+vj​).

| Клетка | Δij​ | Результат |
| --- | --- | --- |
| (1,2) | 28−(0+v2​) | 28−21=7 |
| (1,3) | 17−(0+v3​) | 17−18=−1 |
| (1,4) | 19−(0+v4​) | 19−22=−3 |
| (4,2) | 16−(2+v2​) | 16−21=−5 |

**Вывод:** Есть отрицательные оценки. План не оптимален.

#### ****5. Коррекция плана****

* Выбираем клетку (1,4) с Δ14​=−3.
* Строим цикл: (1,4) → (2,4) → (2,2) → (1,2) → (1,4).
* Минимальное значение в "минусовых" клетках цикла: θ=min(16,8)=8.
* Увеличиваем перевозку в (1,4) на 8, уменьшаем в (2,4) и (2,2).

**Обновлённый план:**

|  | b1=19 | b2=16 | b3=16 | b4=16 | b5=16 | Запасы |
| --- | --- | --- | --- | --- | --- | --- |
| a1=23 | 19 | – | – | 8 | 4 | 0 |
| a2=24 | – | 0 | – | 8 | – | 0 |
| a3=21 | – | – | 16 | – | 5 | 0 |
| a4=15 | – | – | – | 0 | 15 | 0 |

#### ****6. Повторная проверка оптимальности****

После пересчёта потенциалов все оценки Δij​≥0. **План оптимален.**

### **Итоговая стоимость перевозок:**

19⋅8+8⋅19+4⋅11+8⋅6+16⋅3+5⋅8+15⋅13=635.

**Вывод:** в ходе выполнения лабораторной работы, были изучены математическая модель транспортной задачи, а так же такие методы решения этой задачи, как распределительный метод и метод потенциалов.