МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

**«Московский технический университет связи и информатики»**

Кафедра «Системное программирование»

Лабораторная работа №3

**«Транспортная задача»**

по дисциплине

Теория систем и системный анализ

Выполнил: студент гр. БЭИ2202

Кулешов А. С.

Вариант 16

Проверил: Полищук Ю.В.

Москва, 2024 г

Содержание

[**1.** **Лабораторная работа №1** 3](#_Toc177284701)

[**1.1.** **Цель работы** 3](#_Toc177284702)

[**1.2.** **Задание** 3](#_Toc177284703)

[**1.3.** **Ход работы** 3](#_Toc177284704)

[**1.4.** **Заключение** 17](#_Toc177284705)

1. **Лабораторная работа №3**
   1. **Цель работы**

Изучить решение транспортной задачи методом потенциалов, в которой начальное решение было получено методом северо-западного угла.

* 1. **Задание**

Решить 3 примеров транспортных задач. Показать результаты решения, сверить ответы

* 1. **Ход работы**

**Теоретическая информация**

Транспортная задача имеет несколько разных методов решения, в одном из них необходимо найти базовое решение методом северо-западного угла (выбор произвольного решения), после чего проверять его оптимальность, в случае не оптимальности произвести оценку методом потенциалов и найти новое, более близкое к оптимальному решение. После чего повторить данные шаги до достижения оптимального решения

**Практическая реализация**

Напишу программу на языке C++, использовав лишь 3 библитеки: «iomanip» и «iostream» – для вывода данных в консоль, «vector» – для упрощения работы с динамическими массивами. Перепроверю результаты программы с результатами, полученными с онлайн-решателя.

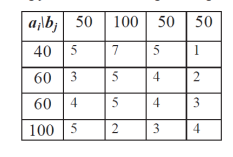


Рисунок 1 – условие первого задания

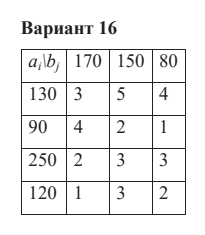


Рисунок 2 – условие индивидуального задания



Рисунок 3 – условие последнего задания

Первое задание решено корректно, алгоритм реализованный в данной работе оперирует числами с плавающей точкой, а не дробями, как в выводе с онлайн-решателя, однако нетрудно заметить, что ответы совпадают.

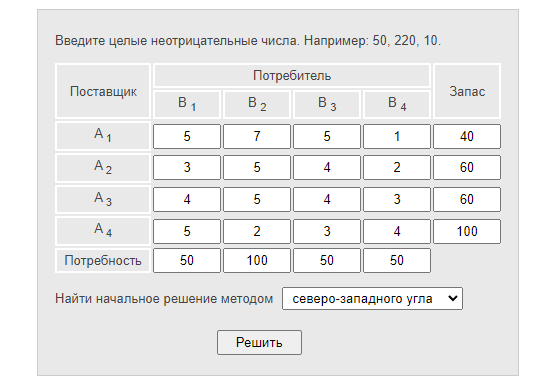


Рисунок 4 – условие первого задания в решателе

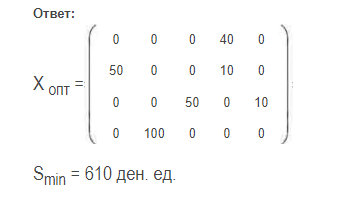


Рисунок 5 – ответ решателя

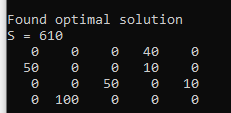


Рисунок 6 – найденный ответ

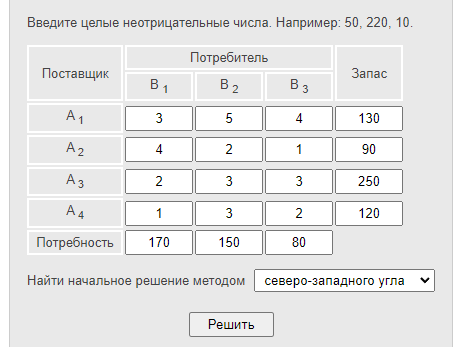


Рисунок 7 – условие второго задания в решателе

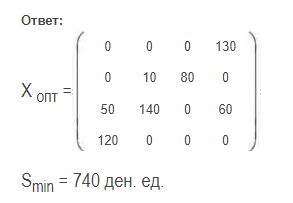


Рисунок 8 – ответ решателя

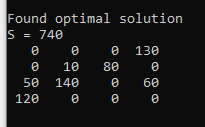


Рисунок 9 – найденный ответ

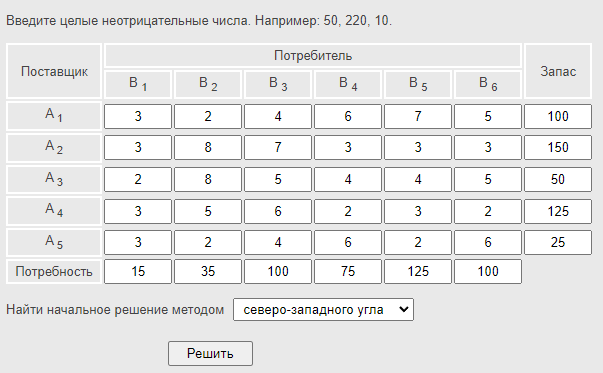


Рисунок 10 – условие третьего задания в решателе

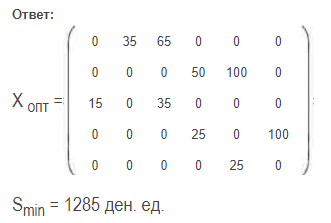


Рисунок 11 – ответ решателя

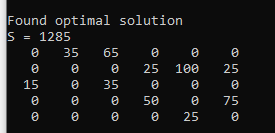


Рисунок 12 – найденный ответ

Программный код представлен ниже.

#include <iostream>

#include <vector>

#include <iomanip>

**using** **namespace** std;

**void** **print**(**const** vector<vector<**long**> >& vc, **int** starti = **0**, **int** startj = **0**){

**for**(**int** i = starti; i < vc.size(); ++i){

**for**(**int** j = startj; j < vc[i].size(); ++j){

cout << setw(**4**) << vc[i][j] << " ";

}

cout << endl;

}

cout << endl;

}

**long** **evaluate**(**const** vector<vector<**long**> >& plan, **const** vector<vector<**long**> >& matrix){

**long** value = **0**;

**for**(**int** i = **1**; i < plan.size(); ++i){

**for**(**int** j = **1**; j < plan[i].size(); ++j){

value += plan[i][j] \* matrix[i][j];

}

}

**return** value;

}

**bool** **optimize**(**const** vector<pair<**int**,**int**> >& elements, vector<vector<**long**> >& plan, **const** vector<vector<**long**> >& matrix){

**for**(**int** i = **0**; i < elements.size(); ++i){

**for**(**int** j = i + **1**; j < elements.size(); ++j){

**long** prev = matrix[elements[i].first ][elements[i].second] + matrix[elements[j].first ][elements[j].second];

**long** optimization = matrix[elements[i].first ][elements[j].second] + matrix[elements[j].first ][elements[i].second];

**if** (optimization < prev){

**long** mini = min(plan[elements[i].first ][elements[i].second], plan[elements[j].first ][elements[j].second]);

plan[elements[i].first ][elements[i].second] -= mini;

plan[elements[j].first ][elements[j].second] -= mini;

plan[elements[i].first ][elements[j].second] += mini;

plan[elements[j].first ][elements[i].second] += mini;

**return** **0**;

}

}

}

**return** **1**;

}

**bool** **optimize2**(**const** vector<pair<**int**,**int**> >& elements, vector<vector<**long**> >& plan, **const** vector<vector<**long**> >& matrix){

**for**(**int** i = **0**; i < elements.size(); ++i){

**for**(**int** j = **0**; j < elements.size(); ++j){

**for**(**int** k = **0**; k < elements.size(); ++k){

**long** prev = matrix[elements[i].first ][elements[i].second] + matrix[elements[j].first ][elements[j].second] + matrix[elements[k].first ][elements[k].second];

**long** optimization = matrix[elements[i].first ][elements[j].second] + matrix[elements[j].first ][elements[k].second] + matrix[elements[k].first ][elements[i].second];

**if** (optimization < prev){

**long** mini = min(min(plan[elements[i].first ][elements[i].second], plan[elements[j].first ][elements[j].second]), plan[elements[k].first ][elements[i].second]);

plan[elements[i].first ][elements[i].second] -= mini;

plan[elements[j].first ][elements[j].second] -= mini;

plan[elements[k].first ][elements[k].second] -= mini;

plan[elements[i].first ][elements[j].second] += mini;

plan[elements[j].first ][elements[k].second] += mini;

plan[elements[k].first ][elements[i].second] += mini;

**return** **0**;

}

}

}

}

**return** **1**;

}

**int** **main**()

{

/\*\*

/// Задание 1 (нулевое)

vector<vector<long> > matrix = {

{0, 50, 100, 50, 50},

{40, 5, 7, 5, 1},

{60, 3, 5, 4, 2},

{60, 4, 5, 4, 3},

{100, 5, 2, 3, 4}

};

\*/

/\*\*

/// Задание 2 (индивидуальное)

vector<vector<long> > matrix = {

{0, 170, 150, 80, },

{130, 3, 5, 4},

{90, 4, 2, 1},

{250, 2, 3, 3},

{120, 1, 3, 2}

};

\*/

/// Задание 3 (последнее)

vector<vector<**long**> > matrix = {

{**0**, **15**, **35**, **100**, **75**, **125**, **100**},

{**100**, **3**, **2**, **4**, **6**, **7**, **5**},

{**150**, **3**, **8**, **7**, **3**, **3** ,**3**},

{**50**, **2**, **8**, **5**, **4**, **4**, **5**},

{**125**, **3**, **5**, **6**, **2**, **3**, **2**},

{**25**, **3**, **2**, **4**, **6**, **2**, **6**}

};

**long** sum1 = **0**;

**long** sum2 = **0**;

print(matrix);

**for**(**int** i = **1**; i < matrix.size(); ++i){

sum1 += matrix[i][**0**];

}

**for**(**int** j = **1**; j < matrix[**0**].size();++j){

sum2 += matrix[**0**][j];

}

**if** ( sum1 == sum2 ){

cout << "No need for extra buyers or sellers" << endl;

}**else**{

**if** (sum1 > sum2){

cout << "Extra buyer is required" << endl;

**for**(**int** i = **1**; i < matrix.size(); ++i){

matrix[i].push\_back(**0**);

}

matrix[**0**].push\_back(sum1 - sum2);

print(matrix);

}**else**{

cout << "Extra seller is required" << endl;

matrix.push\_back({sum2 - sum1});

**for**(**int** i = **1**; i < matrix[**0**].size(); ++i){

matrix[matrix.size() - **1**].push\_back(**0**);

}

print(matrix);

}

}

vector<vector<**long**> > plan = matrix;

**for**(**int** i = **1**; i < plan.size(); ++i){

**for**(**int** j = **1**; j < plan[i].size(); ++j){

plan[i][j] = **0**;

}

}

**for**(**int** i = **1**; i < plan.size(); ++i){

**for**(**int** j = **1**; j < plan[i].size(); ++j){

**long** mini = min(plan[i][**0**], plan[**0**][j]);

plan[i][j] = mini;

plan[i][**0**] -= mini;

plan[**0**][j] -= mini;

}

}

cout << "Plan" << endl;

print(plan);

**long** inf = -**1001** \* **1001** \* **1001**;

**for**(**int** i = **0**; i < plan.size(); ++i){

plan[i][**0**] = inf;

}

**for**(**int** i = **0**; i < plan[**0**].size(); ++i){

plan[**0**][i] = inf;

}

plan[**0**][**0**] = **0**;

plan[**0**][**1**] = **0**;

**for**(**int** i = **1**; i < plan.size(); ++i){

**for**(**int** j = **1**; j < plan[i].size(); ++j){

**if** (plan[i][j] == **0**) **continue**;

**if** (plan[i][**0**] == inf && plan[**0**][j] == inf){

cerr << "this is supposed to be unreachable..." << endl;

exit(**0**);

}

**if** (plan[i][**0**] == inf){

plan[i][**0**] = matrix[i][j] - plan[**0**][j];

}

**if** (plan[**0**][j] == inf){

plan[**0**][j] = matrix[i][j] - plan[i][**0**];

}

**if** (plan[**0**][j] + plan[i][**0**] != matrix[i][j]){

cerr << "this is supposed to be unreachable 2..." << endl;

exit(**0**);

}

}

}

cout << "Potential" << endl;

print(plan);

**while**(**1**){

vector<pair<**int**, **int**> > elements;

**for**(**int** i = **1**; i < plan.size(); ++i){

**for**(**int** j = **1**; j < plan[i].size(); ++j){

**if** (plan[i][j] == **0**) **continue**;

elements.push\_back({i,j});

}

}

**bool** optimal = optimize(elements,plan,matrix);

**if** (optimal){

optimal = optimize2(elements,plan,matrix);

}

**if** (optimal){

cout << "Found optimal solution" << endl;

cout << "S = " << evaluate(plan, matrix) << endl;

print(plan, **1**, **1**);

**return** **0**;

}**else**{

cout << "Plan is not optimal" << endl;

cout << "S = " << evaluate(plan, matrix) << endl;

print(plan);

}

}

**return** **0**;

}

* 1. **Заключение**

В ходе проделанной работы мы познакомились с решением транспортной задачи методом потенциалов, в которой начальное решение было получено методом северо-западного угла