Министерство образования Республики Беларусь  
Учреждение образования  
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

КАФЕДРА ИНФОРМАТИКИ

Отчёт по лабораторной работе No5

По теме “Метод потенциалов для решения

матрично-транспортной задачи”

Выполнила: студентка гр. 053503 Зырянова М.М.

Проверил: ассистент кафедры информатики Туровец Н. О.

Минск 2023

**СОДЕРЖАНИЕ**

[1 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ 3](#_Toc134136524)

[2 АЛГОРИТМ РЕШЕНИЯ ТРАНСПОРТНОЙ ЗАДАЧИ 4](#_Toc134136525)

[3 ДЕМОНСТРАЦИЯ РАБОТЫ ПРОГРАММЫ 7](#_Toc134136526)

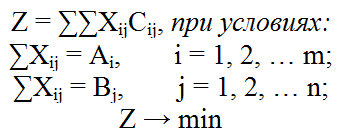
[ПРИЛОЖЕНИЕ А 8](#_Toc134136527)

# **1 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Транспортная задача (задача Монжа — Канторовича) — математическая задача линейного программирования специального вида о поиске оптимального распределения однородных объектов из аккумулятора к приемникам с минимизацией затрат на перемещение

Для простоты понимания рассматривается как задача об оптимальном плане перевозок грузов из пунктов отправления (например, складов) в пункты потребления (например, магазины), с минимальными общими затратами на перевозки.

Математическая модель транспортной задачи имеет следующий вид:



где:

Z — затраты на перевозку грузов;

X — объем груза;

C — стоимость (тариф) перевозки единицы груза;

A — запас поставщика;

B — запрос потребителя;

m — число поставщиков;

n — число потребителей.

Решить транспортную задачу можно различными методами, начиная от симплекс-метода и простого перебора, и заканчивая методом графов. Один из наиболее применяемых и подходящих для большинства случаев методов — итерационное улучшение плана перевозок.

Суть его в следующем: находим некий опорный план и проверяем его на оптимальность (Z → min). Если план оптимален — решение найдено. Если нет — улучшает план столько раз, сколько потребуется, пока не будет найден оптимальный план.

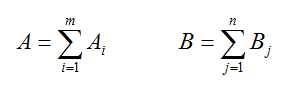
# **2 АЛГОРИТМ РЕШЕНИЯ ТРАНСПОРТНОЙ ЗАДАЧИ**

1. **Построение транспортной таблицы.**

Заполняем транспортную таблицу с исходными данными, где указываем запасы материалов, имеющиеся на складах поставщиков (Ai), и потребности заводов (Bj) в этих материалах. В нижний правый угол ячеек таблицы заносим значение тарифов на перевозку груза (Cij)

1. **Проверка задачи на закрытость.**

Обозначим суммарный запас груза у всех поставщиков символом A, а суммарную потребность в грузе у всех потребителей — символом B. Тогда:



Транспортная задача называется закрытой, если A = B. Если же A ≠ B, то транспортная задача называется открытой. В случае закрытой задачи от поставщиков будут вывезены все запасы груза, и все заявки потребителей будут удовлетворены. В случае открытой задачи для ее решения придется вводить фиктивных поставщиков или потребителей.

1. **Составление опорного плана.**

Составляет предварительный (опорный) план перевозок. Он не обязательно должен быть оптимальный. Это просто своеобразный «черновик» или «набросок», итерационно улучшая который мы постепенно придем к оптимальному плану.

Есть разные методы нахождения опорного плана. Наиболее распространены следующие:

– Метод Северо-Западного угла

Суть метода проста — ячейки транспортной таблицы последовательно заполняются максимально возможными объемами перевозок, в направлении сверху вниз и слева направо. То есть сперва заполняется самая верхняя левая ячейка («северо-западная» ячейка), потом следующая справа и т. д. Затем переходят на новую строку и вновь заполняют ее слева направо. И так пока таблица не будет заполнена полностью.

– Метод минимального элемента

Метод заключается в том, что для заполнения ячеек транспортной таблицы выбирается клетка с минимальным тарифом. Затем выбирается следующая клетка с наименьшим значением тарифа и так продолжается до тех пор, пока таблица не будет заполнена (все запасы и потребности при этом обнулятся).

– Аппроксимация Фогеля

Основа метода в нахождении разности (по модулю) между парой минимальных тарифов в каждой строке и столбце. Затем в строке или столбце с наибольшей разностью заполняется клетка с наименьшим тарифом. Затем все эти действия повторяются заново, только при этом уже не учитываются заполненные клетки.

– Метод двойного предпочтения

Суть метода в том, что отмечаются клетки с наименьшим тарифом по строкам, а затем по столбцам. Затем ячейки заполняются в следующей очередности: сначала клетки с двумя отметками, потом с одной, наконец без отметок.

1. **Проверка опорного плана на вырожденность.**

Клетки таблицы, в которые записаны отличные от нуля перевозки, называются базисными, а остальные (пустые) — свободными. План называется вырожденным, если количество базисных клеток в нем меньше, чем m + n — 1. Если во время решения задачи получился вырожденный план, то его необходимо пополнить, проставив в недостающем числе клеток нулевую перевозку и превратив, тем самым, эти клетки в базисные (общий баланс и суммарная стоимость перевозок плана при этом не изменятся).

Проводить пополнение плана, выбирая клетки произвольно, нельзя. План должен быть ациклическим! План называется ациклическим, если его базисные клетки не содержат циклов. Циклом в транспортной таблице называется несколько клеток, соединенных замкнутой ломаной линией так, чтобы две соседние вершины ломаной были расположены либо в одной строке, либо в одном столбце. Ломаная линия может иметь точки самопересечения, но не в клетках цикла.

1. **Вычисление потенциалов для плана перевозки.**

Для анализа полученных планов и их последующего улучшения удобно ввести дополнительные характеристики пунктов отправления и назначения, называемые потенциалами.

Этот метод улучшения плана перевозок называется методом потенциалов. Есть другие методы итерационного улучшения плана перевозок, но здесь мы их рассматривать не будем.

Итак, сопоставим каждому поставщику Ai и каждому потребителю Bj соответствующие величины Ui и Vj так, чтобы для всех базисных клеток плана было выполнено следующее соотношение: Ui + Vj = Cij. Добавим к транспортной таблице дополнительную строку и столбец для Ui и Vj.

1. **Проверка опорного плана на оптимальность.**

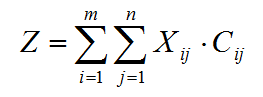
Для каждой свободной клетки плана вычислим разности ΔCij = Cij — (Ui + Vj), и запишем полученные значения в левых нижних углах соответствующих ячеек. План является оптимальным, если все разности ΔCij ≥ 0. В данном случае план — неоптимальный (ΔC22 < 0), и его следует улучшить путем перераспределения поставок.

1. **Перераспределение поставок.**

Найдем ячейку с наибольшей по абсолютной величине (т. е. без учета знака, по модулю) отрицательной разностью ΔCij и построим цикл, в котором кроме этой клетки все остальные являются базисными. Такой цикл всегда существует и единственен.

1. **Если оптимальное решение найдено, переходим к п. 9, если нет — к п. 5.**
2. **Вычисление общих затрат на перевозку груза.**

Вычислим общие затраты на перевозку груза (Z), соответствующие найденному нами оптимальному плану, по формуле:



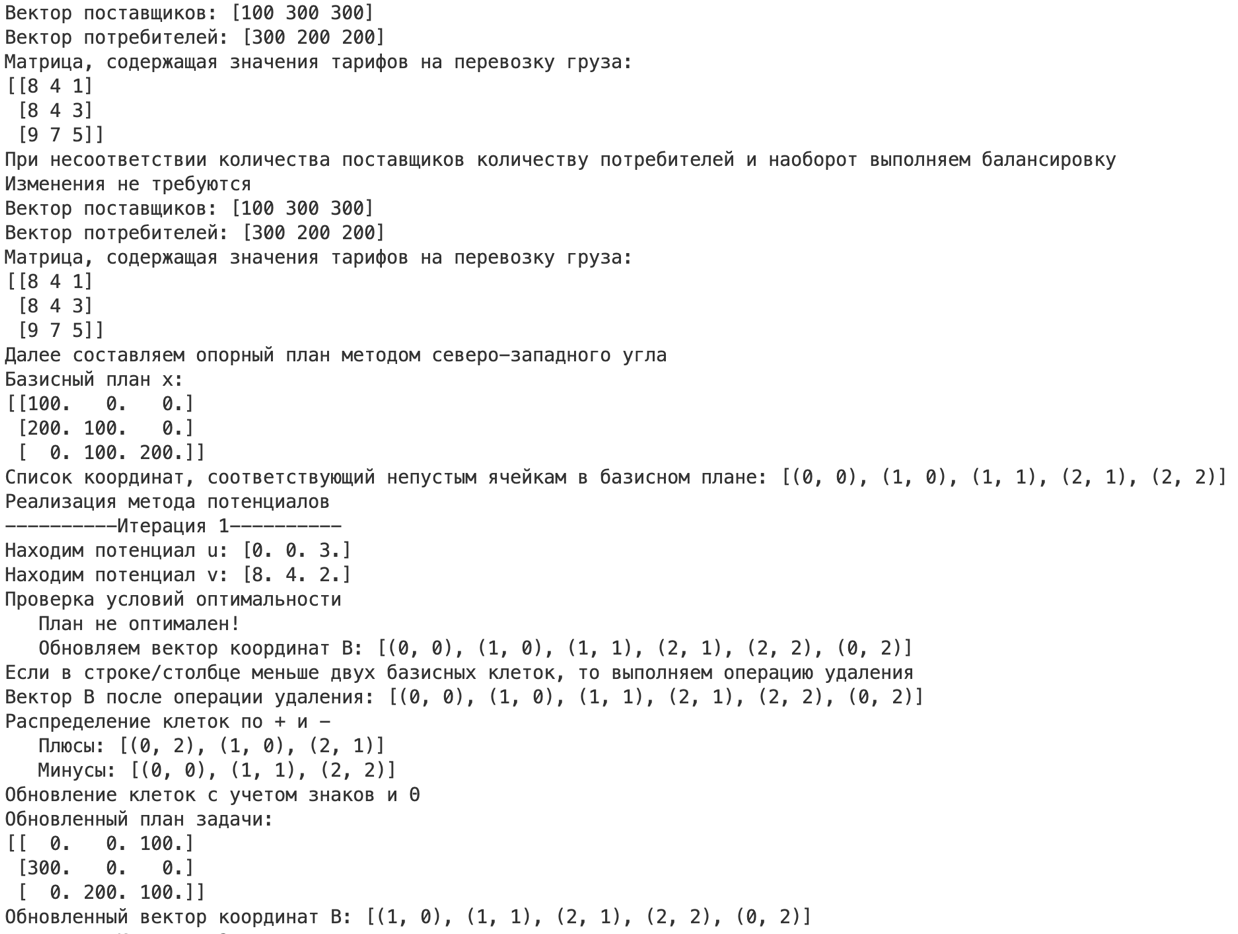
То есть нужно перемножить значения объемов грузоперевозок на соответствующие им тарифы.

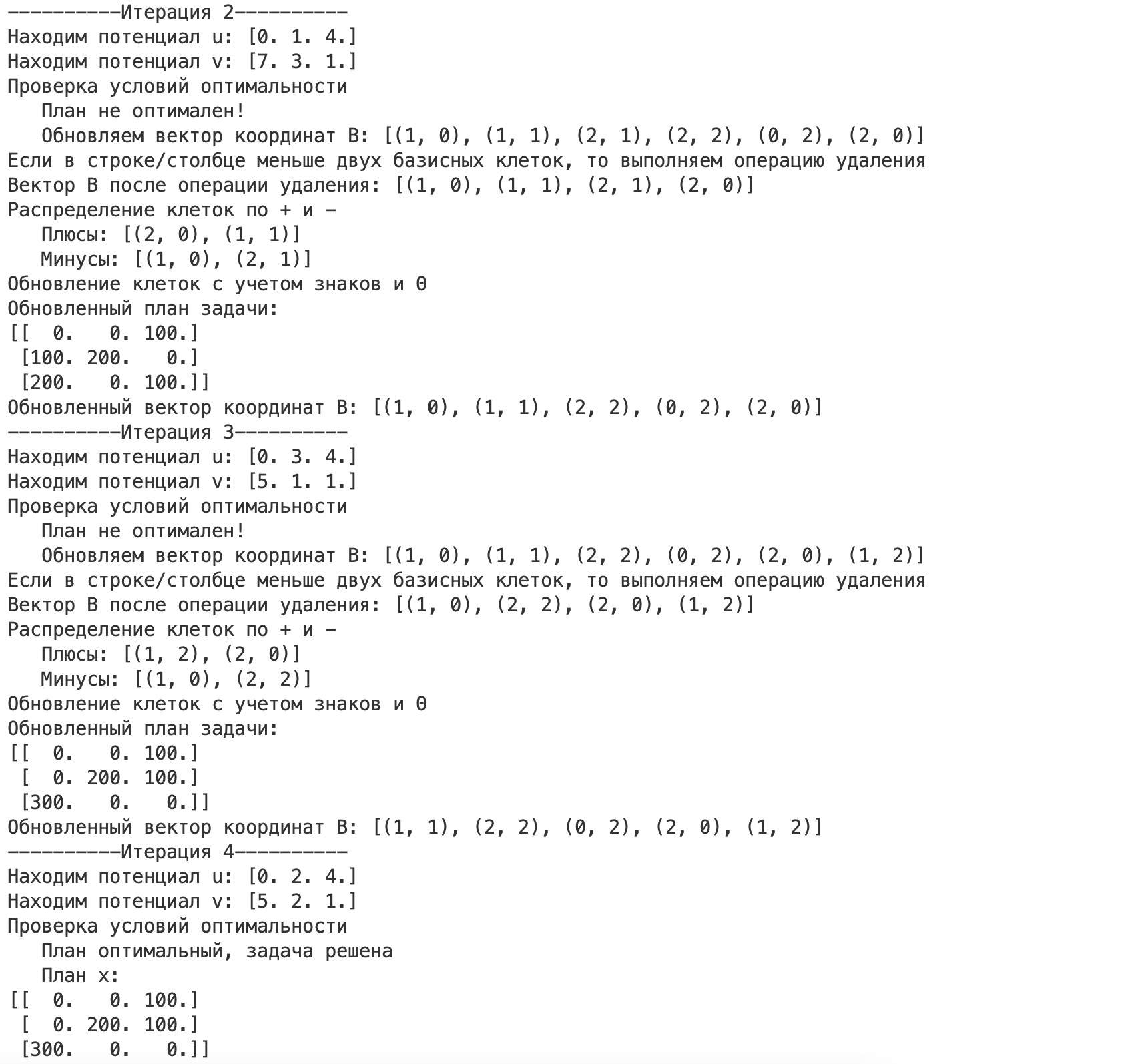
1. **Построение графа перевозок.**

Найдя оптимальный план перевозок, построим граф. Вершинами графа будут «склады» и «магазины». В вершинах укажем соответствующие объемы запасов и потребностей. Дугам, соединяющим вершины графа, будут соответствовать ненулевые перевозки.

Каждую такую дугу подпишем, указав объем перевозимого груза.

# **3** **ДЕМОНСТРАЦИЯ РАБОТЫ ПРОГРАММЫ**





# **ПРИЛОЖЕНИЕ А**

(Листинг программы)

from collections import Counter

from copy import deepcopy

import numpy as np

def balace\_cond(a, b, c):

if (diff := sum(a) - sum(b)) > 0:

b\_new = np.append(b,diff)

a\_new = a

c\_new = np.append(c, [[0] \* c.shape[0]], axis=1)

print('Количество поставщиков больше числа потребителей, следовательно, добавим фиктивного потребителя и обновим матрицу c')

elif diff < 0:

b\_new = b

a\_new = np.append(a,-diff)

c\_new = np.append(c, [[0] \* c.shape[1]], axis=0)

print('Количество потребителей больше числа поставщиков, следовательно, добавим фиктивного поставщика и обновим матрицу c')

else:

a\_new = a

b\_new = b

c\_new = c

print('Изменения не требуются')

print('Вектор поставщиков: ', a\_new, sep='')

print('Вектор потребителей: ', b\_new, sep='')

print('Матрица, содержащая значения тарифов на перевозку груза: ', c\_new, sep='\n')

return a\_new, b\_new, c\_new

if \_\_name\_\_ == "\_\_main\_\_":

a = np.array([100, 300, 300])

print('Вектор поставщиков: ', a, sep='')

b = np.array([300, 200, 200])

print('Вектор потребителей: ', b, sep='')

c = np.array([[8, 4, 1],

[8, 4, 3],

[9, 7, 5]])

print('Матрица, содержащая значения тарифов на перевозку груза: ', c, sep='\n')

print('При несоответствии количества поставщиков количеству потребителей и наоборот выполняем балансировку')

a, b, c = balace\_cond(a, b, c)

# Шаг 1. Метод северо-западного угла

# План перевозок и список базисных позиций

print('Далее составляем опорный план методом северо-западного угла')

x = np.zeros((len(a), len(b)))

B = []

i, j = 0, 0

while i < len(a) and j < len(b):

minimum = min([a[i], b[j]])

x[i, j] = minimum

B.append((i, j))

a[i] -= minimum

b[j] -= minimum

if a[i] == 0 and i < len(a) - 1:

i += 1

elif b[j] == 0:

j += 1

print('Базисный план x: ', x, sep='\n')

print('Список координат, соответствующий непустым ячейкам в базисном плане: ', B, sep='')

print('Реализация метода потенциалов')

n, m = len(a), len(b)

counter = 0

while True:

counter += 1

print(f'----------Итерация {counter}----------')

A = np.zeros((m + n, m + n))

b = np.zeros(m + n)

for num, (i, j) in enumerate(B):

A[num][i] = 1

A[num][m + j] = 1

b[num] = c[i][j]

A[-1][0] = 1

# Находим u и v

u\_v = np.linalg.solve(A, b)

u = u\_v[:m]

v = u\_v[m:]

print('Находим потенциал u: ', u, sep='')

print('Находим потенциал v: ', v, sep='')

print('Проверка условий оптимальности')

optimal, flag = True, True

for i in range(m):

if flag:

for j in range(n):

if u[i] + v[j] > c[i][j]:

optimal, flag = False, False

B.append((i, j))

print(' План не оптимален!')

print(' Обновляем вектор координат B: ', B, sep='')

break

if optimal:

print(' План оптимальный, задача решена')

print(' План x: ', x, sep='\n')

break

print('Если в строке/столбце меньше двух базисных клеток, то выполняем операцию удаления')

B\_copy = deepcopy(B)

while True:

i\_list = [i for (i, j) in B\_copy]

j\_list = [j for (i, j) in B\_copy]

i\_counter = Counter(i\_list)

j\_counter = Counter(j\_list)

i\_to\_rm = [i for i in i\_counter if i\_counter[i] == 1]

j\_to\_rm = [j for j in j\_counter if j\_counter[j] == 1]

if not i\_to\_rm and not j\_to\_rm:

break

B\_copy = [(i, j) for (i, j) in B\_copy if i not in i\_to\_rm

and j not in j\_to\_rm]

print('Вектор В после операции удаления: ', B\_copy, sep='')

print('Распределение клеток по + и -')

plus, minus = [], []

plus.append(B\_copy.pop())

while B\_copy:

if len(plus) > len(minus):

for index, (i, j) in enumerate(B\_copy):

if plus[-1][0] == i or plus[-1][1] == j:

minus.append(B\_copy.pop(index))

break

else:

for index, (i, j) in enumerate(B\_copy):

if minus[-1][0] == i or minus[-1][1] == j:

plus.append(B\_copy.pop(index))

break

print(' Плюсы: ', plus, sep='')

print(' Минусы: ', minus, sep='')

print('Обновление клеток с учетом знаков и Θ')

theta = min(x[i][j] for i, j in minus)

for i, j in plus:

x[i][j] += theta

for i, j in minus:

x[i][j] -= theta

for i, j in minus:

if x[i][j] == 0:

B.remove((i, j))

break

print('Обновленный план задачи: ', x, sep='\n')

print('Обновленный вектор координат B: ', B, sep='')