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

КАФЕДРА ИНФОРМАТИКИ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

По учебной дисциплине Методы оптимизации и управления

Выполнил:
студент группы 853505
Нетецкая Ю.В.

Проверила:
Костюкова О.И.

Минск 2021

Код программы:

```
import numpy as np
import math


def potentials_method(matrix_c, Jb):
    m, n = np.shape(matrix_c)
    a = np.zeros((m + n, m + n))
    b = np.zeros(m + n)
    count = 0
    a[-1][0] = 1
    b[-1] = 0
    for (i, j) in Jb:
        a[count][i] = 1
        a[count][j + m] = 1
        b[count] = matrix_c[i][j]
        count += 1
    x = np.linalg.solve(a, b)
    return x[:m], x[m:]


def check_optimality(Jn, Jb, x, d, delta):
    for (i, j) in Jn:
        if delta[i][j] > 0 and x[i][j] == 0:
            Jb.append((i, j))
            return False, 1, i, j
        elif delta[i][j] < 0 and x[i][j] == d[i][j]:
            Jb.append((i, j))
            return False, -1, i, j
    return True, None, None, None


def matrix_corner_positions(m, n, i0, j0, Jb):
    row_h, col_h = [[] for _ in range(m)], [[] for _ in range(n)]
    cycle = []
    deleted, vert = True, True
    for i, j in Jb:
        row_h[i].append(j)
        col_h[j].append(i)
    while deleted:
        deleted = False
        for i, row in enumerate(row_h):
            if len(row) < 2:
                for j in row:
                    col_h[j].remove(i)
                    deleted = True
                row.clear()
        for j, column in enumerate(col_h):
            if len(column) < 2:
                for i in column:
                    row_h[i].remove(j)
```

```
                deleted = True
            column.clear()
    cycle.append((i0, j0))
    i, j = cycle[0]
    while True:
        if vert:
            vert = False
            i = col_h[j][1] if col_h[j][0] == i else col_h[j][0]
        else:
            vert = True
            j = row_h[i][1] if row_h[i][0] == j else row_h[i][0]
        if i == cycle[0][0] and j == cycle[0][1]:
            break
        else:
            cycle.append((i, j))
    return cycle


def get_theta(n, m, k, cycle, X, D):
    theta = np.array([[math.inf] * n for _ in range(m)])
    if k == -1:
        for i in range(0, len(cycle), 2):
            i0 = cycle[i][0]
            j0 = cycle[i][1]
            theta[i0][j0] = X[i0][j0]
        for i in range(1, len(cycle), 2):
            i0 = cycle[i][0]
            j0 = cycle[i][1]
            theta[i0][j0] = D[i0][j0] - X[i0][j0]
    elif k == 1:
        for i in range(1, len(cycle), 2):
            i0 = cycle[i][0]
            j0 = cycle[i][1]
            theta[i0][j0] = X[i0][j0]
        for i in range(0, len(cycle), 2):
            i0 = cycle[i][0]
            j0 = cycle[i][1]
            theta[i0][j0] = D[i0][j0] - X[i0][j0]
    return theta


def get_min_value(theta):
    return min(map(min, theta))


def transport_method(a, b, c, x, Jb, d):
    m = len(a)
    n = len(b)
    iter = 1
    while True:
        print("**********" + str(iter) + "**********")
        u, v = potentials_method(c, Jb)
```

```
        print("Вектор u = ", u)
        print("Вектор v = ", v)
        Jn = []
        for i in range(m):
            for j in range(n):
                if (i, j) not in Jb:
                    Jn.append((i, j))
        print("Вектор Jn = ", Jn)
        print('Матрица оценок: ')
        delta = np.array([[np.inf] * n for _ in range(m)])
        for (i, j) in Jn:
            delta[i][j] = u[i] + v[j] - c[i][j]
        print(delta)
        is_optimality, k, i0, j0 = check_optimality(Jn, Jb, x, d, delta)
        if is_optimality == True:
            print('Условие оптимальности выполнилось')
            return x
        cycle = matrix_corner_positions(m, n, i0, j0, Jb)
        theta = get_theta(n, m, k, cycle, x, d)
        print("Знаяения Theta: ")
        print(theta)
        min_theta = get_min_value(theta)
        print("Минимальная Theta: ", min_theta)
        k_0 = 1
        if k == -1:
            k_0 = -1
        for i, j in cycle:
            x[i][j] = x[i][j] + k_0 * min_theta
            if k_0 == 1:
                k_0 = -1
            else:
                k_0 = 1
        for i, j in cycle:
            if theta[i][j] == min_theta:
                Jb.remove((i, j))
                break
        print('Новый вектор Jb = ', Jb)
        iter += 1


if __name__ == '__main__':
    a = np.array([27, 22, 20, 17, 29])
    b = np.array([24, 8, 26, 9, 25, 23])
    x = np.array([[4, 3, 10, 0, 10, 0],
        [10, 0, 0, 2, 0, 10],
        [0, 0, 6, 0, 5, 9],
        [0, 0, 10, 7, 0, 0],
        [10, 5, 0, 0, 10, 4]])
    c = np.array([[1, 1, 20, -4, 15, -1],
        [10, -5, -3, 1, -1, 2],
        [-1, 2, 4, -5, 2, 1],
        [4, 3, 40, 6, -6, -20],
```

```
    [5, 10, -10, -3, 15, 5]])
d = [[10, 10, 10, 10, 10, 10],
    [10, 10, 10, 10, 10, 10],
    [10, 10, 10, 10, 10, 10],
    [10, 10, 10, 10, 10, 10],
    [10, 10, 10, 10, 10, 10]]
Jb = [(0, 0), (0, 1), (1, 3), (2, 2), (2, 4), (2, 5), (3, 3), (3, 5), (4, 1), (4, 5)]
x = transport_method(a, b, c, x, Jb, d)
print(x)
```

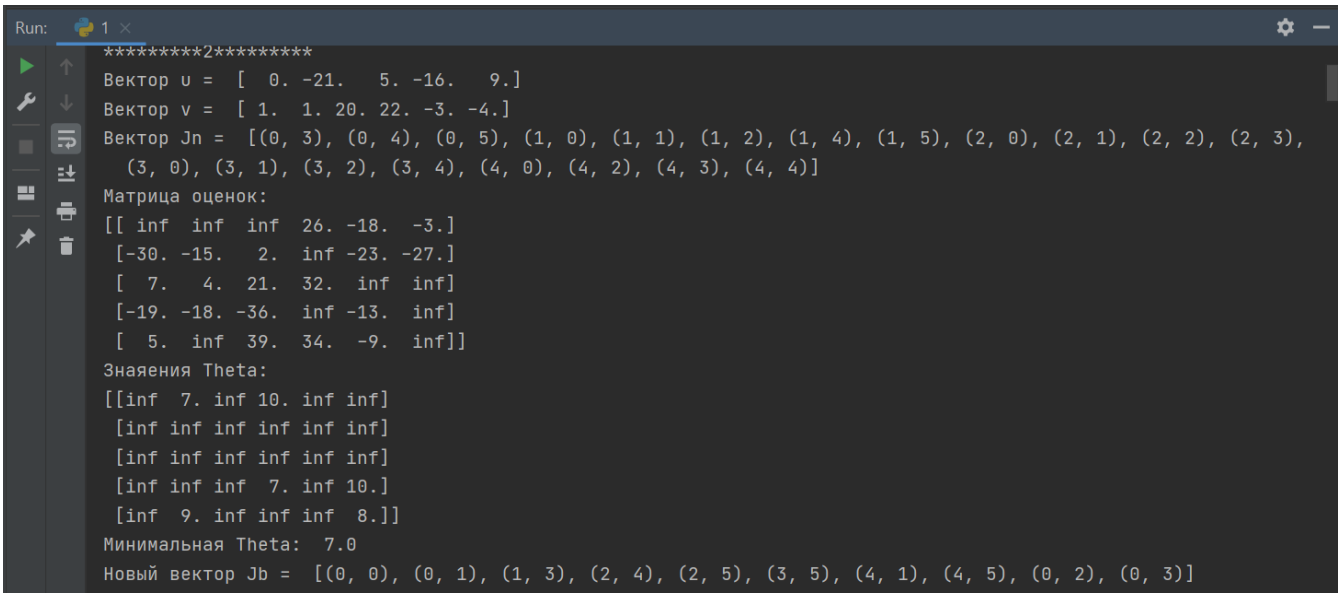
Входные данные:

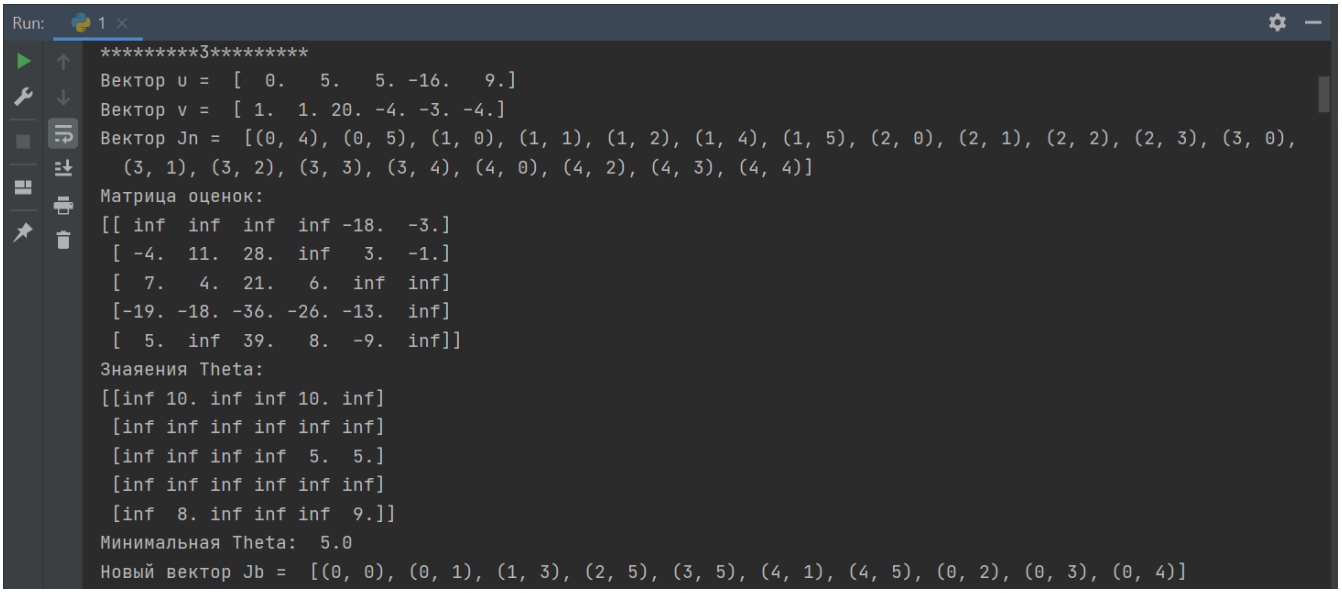
```
a = np.array([27, 22, 20, 17, 29])
b = np.array([24, 8, 26, 9, 25, 23])
x = np.array([[4, 3, 10, 0, 10, 0],
     [10, 0, 0, 2, 0, 10],
     [0, 0, 6, 0, 5, 9],
     [0, 0, 10, 7, 0, 0],
     [10, 5, 0, 0, 10, 4]])
c = np.array([[1, 1, 20, -4, 15, -1],
     [10, -5, -3, 1, -1, 2],
     [-1, 2, 4, -5, 2, 1],
     [4, 3, 40, 6, -6, -20],
     [5, 10, -10, -3, 15, 5]])
d = [[10, 10, 10, 10, 10, 10],
     [10, 10, 10, 10, 10, 10],
     [10, 10, 10, 10, 10, 10],
     [10, 10, 10, 10, 10, 10],
     [10, 10, 10, 10, 10, 10]]
Jb = [(0, 0), (0, 1), (1, 3), (2, 2), (2, 4), (2, 5), (3, 3), (3, 5), (4, 1), (4, 5)]
```

Результат работы:

Первые три итерации:

```
Run:  1
C:\Users\User\Desktop\moiu\dop_task5\venv\Scripts\python.exe C:/Users/User/Desktop/moiu/dop_task5/1.py
*********1*********
Вектор u =  [  0. -21.   5. -16.   9.]
Вектор v =  [ 1.  1. -1. 22. -3. -4.]
Вектор Jn =  [(0, 2), (0, 3), (0, 4), (0, 5), (1, 0), (1, 1), (1, 2), (1, 4), (1, 5), (2, 0), (2, 1), (2, 3),
 (3, 0), (3, 1), (3, 2), (3, 4), (4, 0), (4, 2), (4, 3), (4, 4)]
Матрица оценок:
[[ inf  inf -21.  26. -18.  -3.]
 [-30. -15. -19.  inf -23. -27.]
 [  7.   4.  inf  32.  inf  inf]
 [-19. -18. -57.  inf -13.  inf]
 [  5.  inf  18.  34.  -9.  inf]]
Знаяения Theta:
[[inf  7. 10. inf inf inf]
 [inf inf inf inf inf inf]
 [inf inf  4. inf inf  9.]
 [inf inf inf inf inf inf]
 [inf  5. inf inf inf  6.]]
Минимальная Theta:  4.0
Новый вектор Jb =  [(0, 0), (0, 1), (1, 3), (2, 4), (2, 5), (3, 3), (3, 5), (4, 1), (4, 5), (0, 2)]
```





Результат:

```
Условие оптимальности выполнилось
[[10  6  0  1  0 10]
 [ 0  2 10  0 10  0]
 [ 6  0  6  0  8  0]
 [ 0  0  0  0  7 10]
 [ 8  0 10  8  0  3]]

Process finished with exit code 0
```