Министерство образования и науки Российской Федерации
Федеральное агентство по образованию
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Вятский государственный университет»

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Отчет по лабораторной работе №4 дисциплины «Исследование операций»

Выполнил студент группы ИВТ-31_____/Крючков И. С/
Проверил_____/Коржавина А. С./

Киров 2022

1. Цель работы

Закрепить на практике знания о методах решения транспортных задач линейного программирования и получить навыки их программной реализации.

2. Задание
    1) Выбрать метод первоначального заполнения базиса и метод решения транспортной задачи, согласовать выбор задач с преподавателем
    2) Реализовать выбранные методы решения транспортной задачи.
3. Решение

Метод первоначального заполнения базиса транспортной задачи – метод минимального элемента.

Метод решение транспортной задачи – метод потенциалов.

4. Алгоритм решение транспортной задачи методом потенциалов в общем виде
    1) Построение транспортной таблицы
    2) Проверка задачи на закрытость
    3) Составление опорного плана
    4) Проверка опорного плана на вырожденность
    5) Вычисление потенциалов для плана перевозки
    6) Проверка опорного плата на оптимальность
    7) Перераспределение поставок
    8) Если оптимальное решение найдено, переход к п. 9, иначе к п. 5
    9) Вычисление общих затрат на перевозку груза

5. Листинг программы

```
import math
import copy

# Ввод данных
def input_data():

    u_num_in = int(input("Введите количество поставщиков: \n"))
    v_num_in = int(input("Введите количество потребителей: \n"))

    u_raw = input(f"Введите запасы поставщиков, ({u_num_in} числ. через пробел): \n").split()

    u_in = list(map(float, u_raw))

    v_raw = input(f"Введите потребности потребителей, ({v_num_in} числ. через пробел):
\n").split()

    v_in = list(map(float, v_raw))

    print(f"Введите стоимость доставки. Построчно (enter), по {v_num_in} числ. в строке через
пробел")

    prices_in = []

    for i in range(u_num_in):
        price_row = input().split()
        price_row_in = list(map(float, price_row))

        prices_in.append(price_row_in)

    print()
    return u_num_in, v_num_in, prices_in, u_in, v_in


def prepare_data(resources, price_in, u_data, v_data, un, vn):
    us = 0
    vs = 0
    ad = 0

    for u in u_data:
        us += u

    for v in v_data:
        vs += v

    if vs != us:
        if vs > us:
            t = [0]*vn
            n = [None]*vn
            resources.append(n)
            price_in.append(t)
            u_data.append(vs - us)
            un += 1
        else:
            for i, r in enumerate(price_in):
                resources[i].append(None)
                price_in[i].append(0)
            v_data.append(us - vs)
            vn += 1
            ad = 1
    else:
        ad = 2
    return un, vn, ad
```

```
def sorting_matrix(matrix, added, un, vn):
    m = copy.deepcopy(matrix)
    t = []
    ta = []

    if added == 0:
        for i, iv in enumerate(m[:-1]):
            for j, jv in enumerate(iv):
                t.append((i, j, jv))

        for k, kv in enumerate(m[-1]):
            ta.append((un-1, k, kv))
    elif added == 1:
        for i, iv in enumerate(m):
            for j, jv in enumerate(iv[:-1]):
                t.append((i, j, jv))
            ta.append((i, vn-1, iv[-1]))

    else:
        for i, iv in enumerate(m):
            for j, jv in enumerate(iv):
                t.append((i, j, jv))

    ot = sorted(t, key=lambda x: x[2])
    ot += ta

    return ot

def get_total_price(res, prices):
    t = 0
    for i, iv in enumerate(res):
        for j, jv in enumerate(iv):
            if jv != None:
                t += jv * prices[i][j]

    return t


def get_potentials(u_n, v_n, res, price):
    u_p = [None] * u_n
    v_p = [None] * v_n

    for i, iv in enumerate(res):
        for j, jv in enumerate(iv):
            if jv != None:
                k = i
                while k < u_n:
                    if res[k][j] != None:
                        if u_p[k] == None and v_p[j] == None:
                            u_p[k] = 0

                        if u_p[k] == None:
                            u_p[k] = price[k][j] - v_p[j]
                        else:
                            v_p[j] = price[k][j] - u_p[i]

                    k += 1

    return u_p, v_p

def get_circle(si, sj, res, u_n, v_n):
```

```
    c_i = si
    c_j = sj

    r_i = si
    r_j = sj

    circle = [{ 'i': c_i, 'j': c_j}]

    j = 0

    t_i = 0
    t_f = 0

    while True:
        st = 0
        if res[c_i][j] != None:

            if not (j == r_j and c_i == r_i):
                if j != sj:
                    while t_i < u_n and t_f == 0:
                        if t_i == c_i or res[t_i][j] == None:
                            t_i += 1
                        else:
                            circle.append({ 'i': c_i, 'j': j })
                            circle.append({ 'i': t_i, 'j': j })

                            r_i = t_i
                            r_j = j
                            j = -1
                            c_i = t_i
                            t_f = 1
                            st = 1
                else:
                    circle.append({ 'i': c_i, 'j': j })
                    break

        t_i = 0
        t_f = 0
        j += 1
        if j >= v_n:
            if st == 0:
                vt = circle.pop()
                circle.pop()
                t_i = vt['i'] + 1
                j = vt['j']
                r_j = circle[-1]['j']
                c_i = circle[-1]['i']
                r_i = circle[-1]['i']



    print(f'circle {circle}')

    return circle

def get_circle_delta(c, res):
    s = res[c[1]['i']][c[1]['j']]
    sgn = 0
    for ij in c[2:]:
        if sgn == 1:
            cv = res[ij['i']][ij['j']]
            if cv < s:
                s = cv
```

```
        sgn ^= 1
    return s

def print_arr(a):
    for v in a:
        print(v)

def transport(u_num_in, v_num_in, prices_in, u_in, v_in):
    u_data = copy.copy(u_in)
    v_data = copy.copy(v_in)
    resources = [[None]*v_num_in for i in range(u_num_in)]

    u_num_in, v_num_in, pad = prepare_data(resources, prices_in, u_data, v_data, u_num_in,
v_num_in)

    prices_sorted = sorting_matrix(prices_in, pad, u_num_in, v_num_in)

    # начальное распределение ресурсов
    for x in prices_sorted:
        if u_data[x[0]] != None and v_data[x[1]] != None:
            if (u_data[x[0]] >= 0 and v_data[x[1]] > 0) or (u_data[x[0]] > 0 and v_data[x[1]]
>= 0):
                v = min(u_data[x[0]], v_data[x[1]])

                resources[x[0]][x[1]] = v

                if u_data[x[0]] != v_data[x[1]]:
                    if u_data[x[0]] == v:
                        u_data[x[0]] = None
                        v_data[x[1]] -= v
                    elif v_data[x[1]] == v:
                        u_data[x[0]] -= v
                        v_data[x[1]] = None
                else:
                    u_data[x[0]] = None
                    v_data[x[1]] -= v

    while True:
        tp = get_total_price(resources, prices_in)
        print('---------------------------------------------')
        print_arr(resources)
        print(f"S = {tp}")

        u_p, v_p = get_potentials(u_num_in, v_num_in, resources, prices_in)

        und_ij = [0, 0]
        und_mv = -1
        und = []
        np = False

        # оценки незадействованных маршрутов
        for i, iv in enumerate(resources):
            for j, jv in enumerate(iv):
                if jv == None:
                    dl = prices_in[i][j] - (u_p[i] + v_p[j])
                    und.append((i, j, dl))

                    if dl < 0:
                        np = True
                        if und_mv < abs(dl):
                            und_ij = [i, j]
                            und_mv = abs(dl)

        if np == True:
```

```
            circle = get_circle(und_ij[0], und_ij[1], resources, u_num_in, v_num_in)
            cdl = get_circle_delta(circle, resources)


            resources[und_ij[0]][und_ij[1]] = 0
            sgn = 0
            for cd in circle:
                if sgn == 0:
                    resources[cd['i']][cd['j']] += cdl
                else:
                    if resources[cd['i']][cd['j']] == cdl:
                        resources[cd['i']][cd['j']] = None
                    else:
                        resources[cd['i']][cd['j']] -= cdl

                sgn ^= 1
        else:
            break



if __name__ == '__main__':
    u_num_in, v_num_in, prices_in, u_in, v_in = input_data()

    transport(u_num_in, v_num_in, prices_in, u_in, v_in)
```

6. Экранные формы

```
Введите количество поставщиков:
3
Введите количество потребителей:
4
Введите запасы поставщиков, (3 числ. через пробел):
160 140 170
Введите потребности потребителей, (4 числ. через пробел):
120 50 190 110
Введите стоимость доставки. Построчно (enter), по 4 числ. в строке через пробел
7 8 1 2
4 5 9 8
9 2 3 6

-------------------------------------------------
[None, None, 160.0, None]
[120.0, None, None, 20.0]
[None, 50.0, 30.0, 90.0]
S = 1530.0
circle [{'i': 0, 'j': 3}, {'i': 0, 'j': 2}, {'i': 2, 'j': 2}, {'i': 2, 'j': 3}]
-------------------------------------------------
[None, None, 70.0, 90.0]
[120.0, None, None, 20.0]
[None, 50.0, 120.0, None]
S = 1350.0
circle [{'i': 1, 'j': 1}, {'i': 1, 'j': 3}, {'i': 0, 'j': 3}, {'i': 0, 'j': 2}, {'i': 2, 'j': 2}, {'i': 2, 'j': 1}]
-------------------------------------------------
[None, None, 50.0, 110.0]
[120.0, 20.0, None, None]
[None, 30.0, 140.0, None]
S = 1330.0
```

7. Вывод

В ходе выполнения лабораторной работы был изучен метод потенциалов для решение транспортной задачи линейного программирования, получены навыки его программной реализации.