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.

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

Метод первоначального заполнения базиса транспортной задачи – метод минимального элемента.

Метод решение транспортной задачи – метод потенциалов.

1. Алгоритм решение транспортной задачи методом потенциалов в общем виде
2. Построение транспортной таблицы
3. Проверка задачи на закрытость
4. Составление опорного плана
5. Проверка опорного плана на вырожденность
6. Вычисление потенциалов для плана перевозки
7. Проверка опорного плата на оптимальность
8. Перераспределение поставок
9. Если оптимальное решение найдено, переход к п. 9, иначе к п. 5
10. Вычисление общих затрат на перевозку груза
11. 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:

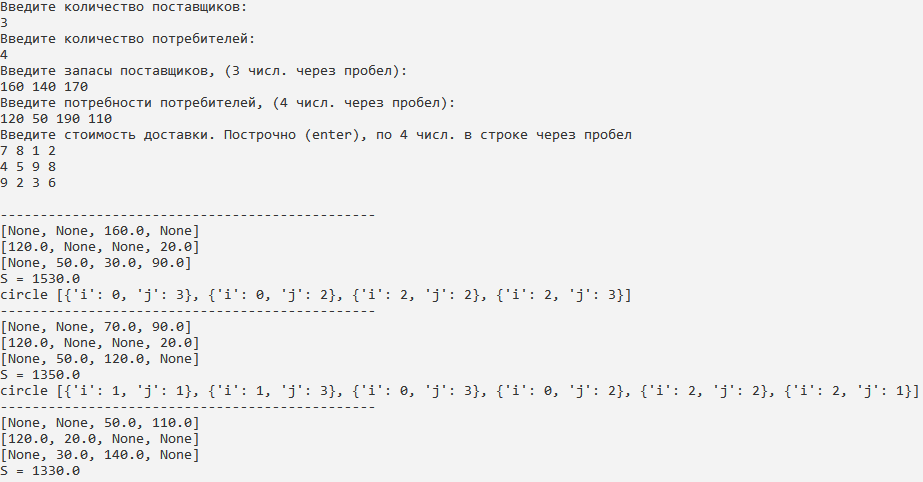
break

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)

1. Экранные формы



1. Вывод

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