МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ДГТУ)

Факультет «Информатика и вычислительная техника»

Кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»

**Лабораторная работа №8,9**

Выполнил:

Студент группы ВМО 31

Оганесьянц К.П.

2021г

**Цель \_работы**: познакомиться с методом ветвей и границ на примере задачи коммивояжера.

**Задание:\_ \_**решить задачу коммивояжера в системе Mathсad.

Коммивояжеру нужно посетить 5 городов, затраты на пере- ход из города в город в таблице 1 (знак # означает запрещение перехо- да). В программах, реализующих решение задачи коммивояжера, соответст- вующим элементам матрицы можно присвоить значение машинной беско- нечности (∞). i j

Приложение №1: Программная реализация

Файл №1: index.py – основной файл с математическими операциями, решающими данную задачу:

from copy import deepcopy

import sys

# sys.stdin = open('in2.txt', 'r')

INF = 10\*\*6

CUT = '-'

if sys.version.split('.')[0] == '2':

input = raw\_input

# reading

#n, k = map(int, input().split(' '))

k = int(input())

g = [None for i in range(k)]

for i in range(k):

g[i] = list(map(lambda x: int(x) if x != '-' else INF, input().split(' ')))

def min\_in\_raw(g, raw):

m = INF+1

for i in range(len(g[raw])):

if g[raw][i] == CUT: continue

m = min(m, g[raw][i])

if m == INF+1: m = None # is cut string

return m

def min\_in\_column(g, column):

m = INF+1

for i in range(len(g)):

if g[i][column] == CUT: continue

m = min(m, g[i][column])

if m == INF+1: m = None # is cut string

return m

def print\_matrix(matrix, title = ''):

print(title)

for it in matrix: print(it)

print('\n')

############

### algo ###

############

print\_matrix(g, '<Source graph>')

def reduction(g):

min\_sum = 0

for i in range(len(g)):

m = min\_in\_raw(g, i)

if m == None: continue

for j in range(len(g)):

if g[i][j] == CUT: continue

g[i][j] -= m

min\_sum += m

for i in range(len(g)):

m = min\_in\_column(g, i)

if m == None: continue

for j in range(len(g)):

if g[j][i] == CUT: continue

g[j][i] -= m

min\_sum += m

return min\_sum

def reduction\_by(g, xy, hm, vm):

for i in range(len(g)):

if g[xy[0]][i] == CUT: continue

g[xy[0]][i] -= hm

g[xy[0]][xy[1]] = INF

for i in range(len(g)):

if g[i][xy[1]] == CUT: continue

g[i][xy[1]] -= vm

g[xy[0]][xy[1]] = INF

min\_sum = reduction(g)

print('<Start (R = {%i})>\n' % min\_sum)

def get\_min\_r():

t\_min = INF

current\_sheet\_index = None

for i in range(len(R)):

if R[i][2]: continue

if t\_min > R[i][0]:

t\_min = R[i][0]

current\_sheet\_index = i

return current\_sheet\_index

def remove\_cross\_from\_matrix(matrix, xy):

for i in range(len(matrix)):

matrix[xy[0]][i] = CUT

matrix[i][xy[1]] = CUT

# [fine, g, is\_sheet, is\_present, edge, parent\_index]

R = [[min\_sum, g, False, True]]

## Cycle ##

while True:

current\_sheet\_index = get\_min\_r()

assert current\_sheet\_index is not None # or break

r = R[current\_sheet\_index]

g = deepcopy(r[1])

print\_matrix(g, '<Matrix (fine = %i)>' % r[0])

# search zeros

zeros = []

for i in range(len(g)):

for j in range(len(g)):

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

zeros.append((i, j))

if len(zeros) == 1:

break

# search maximum among minimums in zeros

mx = -1

vm0 = hm0 = None

for zero in zeros:

t = g[zero[0]][zero[1]]

g[zero[0]][zero[1]] = INF

hm = min\_in\_raw(g, zero[0])

vm = min\_in\_column(g, zero[1])

if mx < hm + vm:

mx = hm + vm

vm0, hm0 = vm, hm

skip = zero

g[zero[0]][zero[1]] = t

assert skip[0] != skip[1] and mx >= 0

g[skip[0]][skip[1]] = INF

expensive\_g = deepcopy(g)

reduction\_by(expensive\_g, skip, hm0, vm0)

print\_matrix(expensive\_g, '<Reduction cross %s, %s>' % (str(skip), mx))

remove\_cross\_from\_matrix(g, skip)

print\_matrix(g, '<Cross-line removing ' + str(skip) + '>')

# fine

if g[skip[1]][skip[0]] != CUT:

g[skip[1]][skip[0]] = INF

min\_sum = reduction(g)

print\_matrix(g, '<Reduction (sum = %i)>' % min\_sum)

r[2] = True

R.append([r[0] + mx, expensive\_g, False, False, skip, current\_sheet\_index])

R.append([r[0] + min\_sum, deepcopy(g), False, True, skip, current\_sheet\_index])

print('<R += {%i+%i, %i+%i}; %s>' % (r[0], mx, r[0], min\_sum, skip))

print('-'\*20 + '\n')

#print('R: ' + str(list(map(lambda x: (x[0], x[2]), R))))

edges = [None]\*k

edges[zeros[0][0]] = zeros[0][1]

i = 0

while i < k-1:

assert r[4][0] is not None

if r[3]:

edges[r[4][0]] = r[4][1]

i += 1

r = R[r[5]]

j = 0

path = [0+1]

for i in range(k):

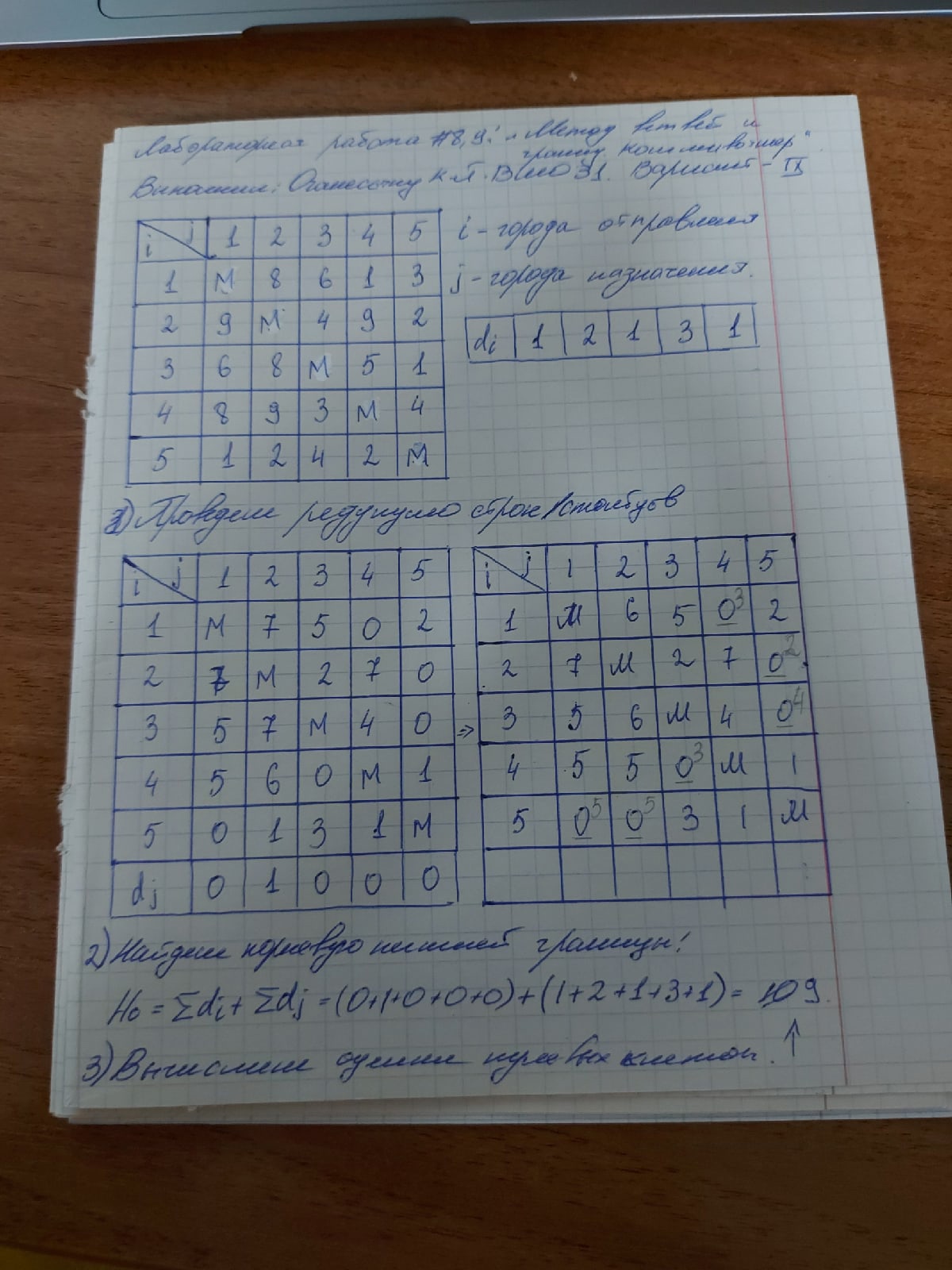
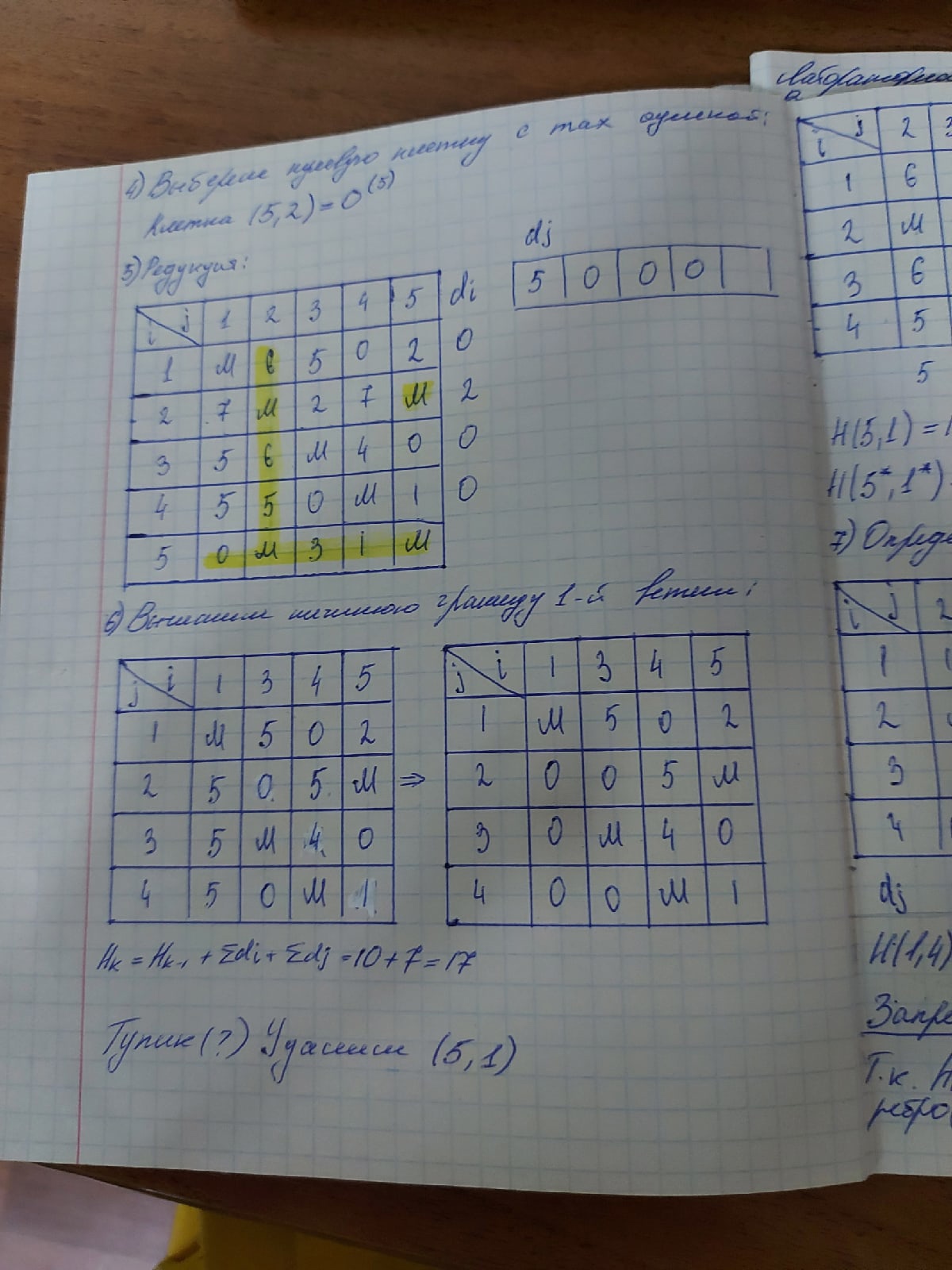
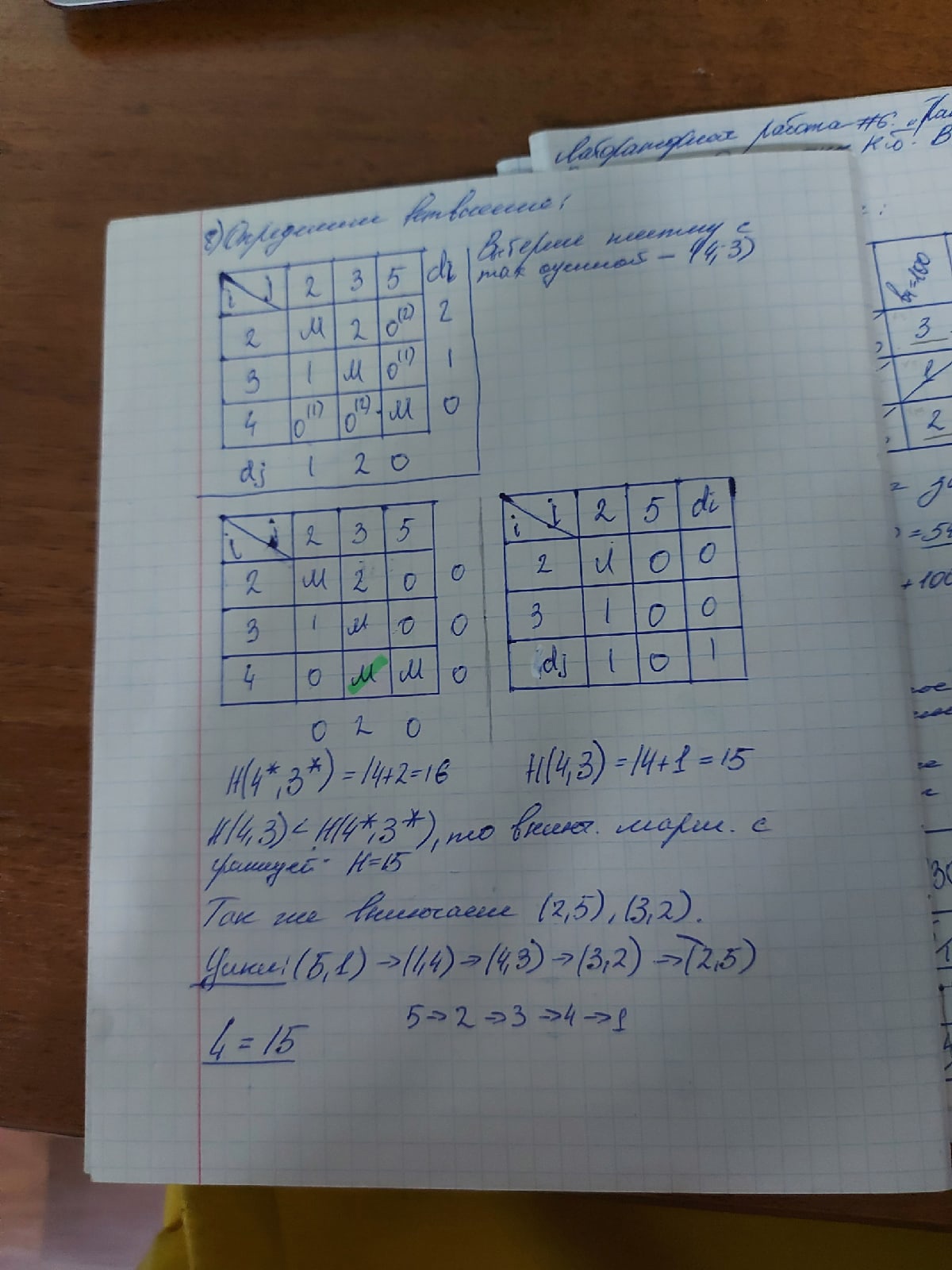
j = edges[j]

path.append(j+1)

print(path)

Приложение №2: Вывод реализованного программного средства:





Приложение №3:Аналитическое решение задачи:

