Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»

Кафедра теоретических основ компьютерной безопасности и криптографии

**Отчет по практическому заданию №3**

**Вычисление значения функции на графе**

студента 5 курса 531 группы

специальности 10.05.01 «Компьютерная безопасность»

факультета компьютерных наук и информационных технологий

Минуситова Амиля Куанышкалиевича

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| Проверил  Доцент | \_\_\_\_\_\_\_\_\_\_ | И. И. Слеповичев |
|  | подпись, дата |  |

Саратов 2022

**На входе:**

а) Текстовый файл с описанием графа в виде списка дуг.

б) Текстовый файл соответствий арифметических операций именам вершин:

a1 : операция 1

a2 : операция 2

...

an : операция n

где ai - имя i-й вершины, операция i - символ операции, соответствующий вершине ai.

Допустимы следующие символы операций:

+ – cумма значений,

\* – произведение значений,

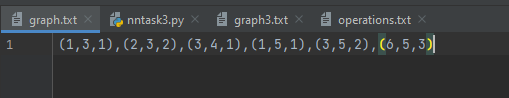
exp – экспонирование входного значения,

число – любая числовая константа.

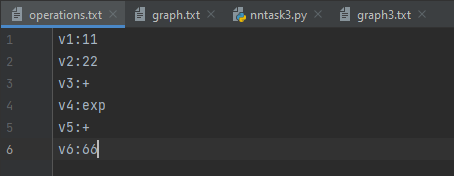
**На выходе:** значение функции, построенной по графу а) и файлу б).

Входные файлы:

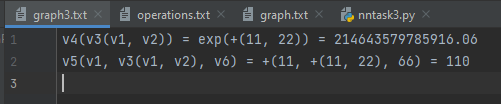
а)



б)



Выходной файл:



Исходный текст программы:

from re import split

import math

def read\_graph():

with open('graph.txt', 'r') as input\_graph:

j = 0

for line in input\_graph:

j += 1

edges = split('\),\(', line)

n = len(edges)

for i in range(n):

edges[i] = edges[i].replace('(', '')

edges[i] = edges[i].replace(')', '')

for i in range(n - 1):

a = edges[i]

# print(edge1)

for j in range(1, n):

b = edges[j]

# print(edge2)

if a != b:

if a[2] == b[2] and a[4] == b[4]:

print('В строке № {} входных данных ошибка введенных данных'.format(str(j)))

exit()

if a[0] == b[0] and a[2] == b[2]:

print('В строке № {} входных данных ошибка введенных данных'.format(str(j)))

exit()

edges.sort(key=lambda i: (i[0], i[2]))

c = edges

k = []

for i in range(n):

k.append(([int(edges[i][0]), int(edges[i][2]), int(edges[i][4])]))

edges = k

# print(edges)

v = []

for i in range(n):

v.append(edges[i][0])

v.append(edges[i][1])

v.sort()

vertex = []

for x in v:

if x not in vertex:

vertex.append(x)

return (edges, vertex)

def matr(edges, vertex):

n = len(vertex)

m = [[0]\*n for i in range(n)]

for a in edges:

m[a[0]-1][a[1]-1] = 1

return m

def stok(matr):

n = len(matr)

crs = []

for r in range(n):

for c in range(n):

if matr[r][c] != 0:

break

else:

crs.append(r + 1)

return crs

def dfs(v, color, d):

color[v] = 'grey'

for y in d[v]:

if color[y] == 'white':

dfs(y, color, d)

if color[y] == 'grey':

print('Ошибка введённых данных. В графе обнаружен цикл')

exit()

color[v] = 'black'

def dfsres(v, color, d):

color[v] = 'grey'

for y in d[v]:

if color[y] == 'white':

dfs(y, color, d)

if color[y] == 'grey':

print('Ошибка введённых данных. В графе обнаружен цикл')

exit()

color[v] = 'black'

def Function(x, func): # вычисление функции графа

if len(d[x]) == 0:

return func

func += '('

t = False

for v in d[x]:

if t == True:

func += ', '

func += f'v{v}'

t = True

func = Function(v, func)

func += ')'

return func

def resultfunc(func):

start = 0

while True:

start = func.rfind('(')

if start == -1:

return func

finish = func.find(')', start)

numbers = func[start + 1:finish:]

if func[start - 1] == '\*':

func = func.replace(func[start - 1:finish + 1:], str(numresultfunc(func[start + 1:finish:], '\*')))

if func[start - 1] == '+':

func = func.replace(func[start - 1:finish + 1:], str(numresultfunc(func[start + 1:finish:], '+')))

if func[start - 3:start:] == 'exp':

func = func.replace(func[start - 3:finish + 1:], str(numresultfunc(func[start + 1:finish:], 'exp')))

def numresultfunc(func, operation):

cnt = []

for x in func.split(', '):

cnt.append(int(x))

if operation == 'exp':

return math.exp(cnt[0])

if operation == '+':

result = 0

for x in cnt:

result += x

return result

if operation == '\*':

result = 1

for x in cnt:

result \*= x

return result

if \_\_name\_\_ == '\_\_main\_\_':

edges, vertex = read\_graph()

d = {}

color = {}

for v in vertex:

color[v] = 'white'

tmp = []

for edge in edges:

if edge[1] == v:

tmp.append(edge[0])

d[v] = tmp

#print(vertex)

vert = []

for v in vertex:

dfs(v, color, d)

crc = stok(matr(edges, vertex))

#print(crc)

f = open('operations.txt', 'r')

operation = {}

f1 = open('graph3.txt', 'w')

for row in f.read().split('\n'):

tmp = row.split(':')

operation[tmp[0]] = tmp[1]

for x in crc:

func = 'v' + Function(x, str(x))

#print(func)

result = func + ' = '

for x in operation:

func = func.replace(x, operation[x])

#f1.write(func + '\n')

#print(func)

result += func + ' = ' + resultfunc(func)

print(result)

f1.write(result + '\n')

#print(operation)