САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №3

по курсу «Алгоритмы и структуры данных»

Тема: Графы.

Вариант 29

Выполнил:

Логачев Д.С.

К3139/К32402

Проверила:

Артамонова В.Е.

Санкт-Петербург

2022 г.

# Содержание отчета

[Содержание отчета 2](#_Toc115810048)

[Задачи по варианту 3](#_Toc115810049)

[4 задача. Порядок курсов (1 балл) 3](#_Toc115810050)

[10 задача. Оптимальный обмен валюты (2 балла) 4](#_Toc115810051)

[15 задача. Герои (3 балла) 6](#_Toc115810052)

[Вывод 8](#_Toc115810053)

# Задачи по варианту

## 4 задача. Порядок курсов (1 балл)

Теперь, когда вы уверены, что в данном учебном плане нет циклических зависимостей, вам нужно найти порядок всех курсов, соответствующий всем зависимостям. Для этого нужно сделать топологическую сортировку соответствующего ориентированного графа. Дан ориентированный ациклический граф (DAG) с n вершинами и m ребрами. Выполните топологическую сортировку.

from collections import defaultdict

def topsortutil(g, v, vis, st):

vis[v] = True

for i in g[v]:

if vis[i] == False:

topsortutil(g, i, vis, st)

st.insert(0, v)

def topsort(g, v):

vis = [False] \* (v + 1)

st = []

for i in range(v + 1):

if vis[i] == False:

topsortutil(g, i, vis, st)

o.write(str(st[:len(st) - 1]))

f = open("input.txt", "r")

o = open("output.txt", "w")

n, m = map(int, f.readline().split())

g = defaultdict(list)

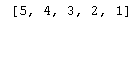
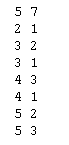
for i in range(m):

q = list(map(int, f.readline().split()))

g[q[0]].append(q[1])

topsort(g, n)

В данном решении основывались на алгоритме поиска в глубину. В отличие от DFS, в топологической сортировке мы используем временный стек, при этом не печатаем вершину сразу, мы сначала вызываем рекурсивно топологическую сортировку для всех соседних вершин, а затем отправляем ее в стек. Наконец, печатаем стек. В DFS же мы начинаем с вершины, печатаем ее, а затем вызываем рекурсивно DFS для соседних вершин.



## 10 задача. Оптимальный обмен валюты (2 балла)

Теперь вы хотите вычислить оптимальный способ обмена данной вам валюты ci на все другие валюты. Для этого вы находите кратчайшие пути из вершины ci во все остальные вершины. Дан ориентированный граф с возможными отрицательными весами ребер, у которого n вершин и m ребер, а также задана одна его вершина s. Вычислите длину кратчайших путей из s во все остальные вершины графа.

class Graph:

def \_\_init\_\_(self, vertices):

self.V = vertices

self.graph = []

def addEdge(self, u, v, w):

self.graph.append([u, v, w])

def printArr(self, dist, root):

with open('output.txt', 'w') as f:

for i in range(self.V):

if dist[i] != float("inf"):

if i == root:

f.write('0\n')

elif dist[i] == 0:

f.write('-\n')

else:

f.write(f'{dist[i]}\n')

else:

f.write('\*\n')

def BellmanFord(self, src):

dist = [float("inf")] \* self.V

dist[src] = 0

for i in range(self.V - 1):

for u, v, w in self.graph:

if dist[u - 1] != float("inf") and dist[u - 1] + w < dist[v - 1]:

dist[v - 1] = dist[u - 1] + w

for u, v, w in self.graph:

if dist[u - 1] != float("inf") and dist[u - 1] + w < dist[v - 1]:

dist[u - 1] = 0

dist[v - 1] = 0

self.printArr(dist, src)

def main():

with open("input.txt") as f:

s = f.readlines()

n, m = map(int, s[0].split())

g = Graph(n)

root = int(s[-1]) - 1

for i in range(1, m + 1):

u, v, w = map(int, s[i].split())

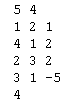
g.addEdge(u, v, w)

g.BellmanFord(root)

if \_\_name\_\_ == "\_\_main\_\_":

main()

Так как допустимы отрицательные циклы, то пришлось использовать алгоритм Беллмана-Форда.



## 15 задача. Герои (3 балла)

Коварный кардинал Ришелье вновь организовал похищение подвесок королевы Анны; вновь спасать королеву приходится героическим мушкетерам. Атос, Портос, Арамис и д’Артаньян уже перехватили агентов кардинала и вернули украденное; осталось лишь передать подвески королеве Анне. Королева ждет мушкетеров в дворцовом саду. Дворцовый сад имеет форму прямоугольника и разбит на участки, представляющие собой небольшие садики, содержащие коллекции растений из разных климатических зон. К сожалению, на некоторых участках, в том числе на всех участках, расположенных на границах сада, уже притаились в засаде гвардейцы кардинала; на бой с ними времени у мушкетеров нет. Мушкетерам удалось добыть карту сада с отмеченными местами засад; теперь им предстоит выбрать наиболее оптимальные пути к королеве. Для надежности друзья разделили между собой спасенные подвески и проникли в сад поодиночке, поэтому начинают свой путь к королеве с разных участков сада. Двигаются герои по максимально короткой возможной траектории. Марлезонский балет вот-вот начнется; королева не в состоянии ждать героев больше L минут; ровно в начале L+1-ой минуты королева покинет парк, и те мушкетеры, что не успеют к этому времени до нее добраться, не смогут передать ей подвески. На преодоление одного участка у мушкетеров уйдет ровно по минуте. С каждого участка мушкетеры могут перейти на 4 соседние. Требуется выяснить, сколько подвесок будет красоваться на платье королевы, когда она придет на бал.

from collections import deque

from math import inf

with open('input.txt', 'r') as f:

n, m = map(int, f.readline().split())

garden = []

distance = []

for i in range(n):

garden.append(list(f.readline().strip('\n')))

distance.append([inf] \* m)

qx, qy, L = map(int, f.readline().split())

qx -= 1

qy -= 1

knights = []

for i in range(4):

x, y, p = map(int, f.readline().split())

knights.append((x, y, p))

steps = [(0, 1), (1, 0), (-1, 0), (0, -1)]

distance[qx][qy] = 0

queue = deque()

queue += [[qx, qy]]

while len(queue):

vx, vy = queue.popleft()

for step in steps:

x = vx + step[0]

y = vy + step[1]

if garden[x][y] == '0' and distance[x][y] == inf:

queue += [[x, y]]

distance[x][y] = distance[vx][vy] + 1

count = 0

for k in knights:

if 0 <= distance[k[0] - 1][k[1] - 1] <= L:

count += k[2]

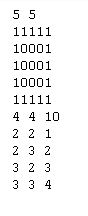
with open('output.txt', 'w') as f:

f.write(str(count))

Представим сад в виде графа. Если ячейка поля ‒ это символ “0”, то данная

ячейка является вершиной графа, и герой может на ней находиться. Положим, что переход в соседнюю ячейку возможен, если у них одна общая сторона. Назначим всем имеющимся связям-переходам в графе вес 1 ‒ столько времени занимает переход. Таким образом, если применить алгоритм Дейкстры, то можно будет понять, сколько минут требуется на преодоление кратчайшего из возможных маршрутов от выбранной вершины до остальных. Воспользуемся этим и применим алгоритм относительно графа и поля, в котором находится королева: тогда в массиве

distances останутся значения ‒ кратчайшие пути между королевой и соответствующими ячейками поля. Конечно, из всех ячеек интересуют в решении те, из которых начинают путь герои с подвесками. Если кратчайший путь героя занимает меньше данного значения, значит, он точно дойдет до королевы и донесет все имеющиеся у него подвески.



# Вывод

В результате выполнения лабораторной работы были реализованы алгоритм DFS с модификациями (в том числе для проверки достижимости вершин, нахождения петель в графе) и алгоритм Беллмана-Форда для нахождения цикла отрицательного веса.