**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №3**

**по дисциплине «Построение и анализ алгоритмов»**

Тема: Максимальный поток

|  |  |  |
| --- | --- | --- |
| Студент гр. 9382 |  | Русинов Д.А. |
| Преподаватель |  | Фирсов М.А. |

Санкт-Петербург

2021

## Цель работы.

Научиться находить величину потока в ориентированном графе, изучить и реализовать алгоритм Форда-Фалкерсона.

## Задание.

1. Найти максимальный поток в сети, а также фактическую величину потока, протекающего через каждое ребро, используя алгоритм Форда-Фалкерсона.
2. Сеть (ориентированный взвешенный граф) представляется в виде триплета из имён вершин и целого неотрицательного числа - пропускной способности (веса).

Входные данные:  
*N* - количество ориентированных рёбер графа  
*v0​ -* исток  
*vn*​ - сток  
*vi* ​​*vj* ​​*ωij*​​ - ребро графа  
*vi* ​​*vj* ​​*ωij*​​ - ребро графа  
...

Выходные данные:  
*Pmax*​ - величина максимального потока  
*vi* ​​*vj* ​​*ωij*​​ - ребро графа с фактической величиной протекающего потока  
*vi* ​​*vj* ​​*ωij*​​ - ребро графа с фактической величиной протекающего потока  
...

1. В ответе выходные рёбра отсортируйте в лексикографическом порядке по первой вершине, потом по второй (в ответе должны присутствовать все указанные входные рёбра, даже если поток в них равен 0).
2. **Sample Input:**
3. 7
4. a
5. f
6. a b 7
7. a c 6
8. b d 6
9. c f 9
10. d e 3
11. d f 4
12. e c 2
13. **Sample Output:**
14. 12
15. a b 6
16. a c 6
17. b d 6
18. c f 8
19. d e 2
20. d f 4
21. e c 2

Вар. 3. Поиск в глубину. Рекурсивная реализация.

## Описание структуры данных, используемой для представления графа.

Для представления графа используется встроенная структура данных – dict. В качестве ключа выступает вершина, а в качестве значения выступает список вершин, до которых есть путь. Вершина в списке имеет три значения:

1. Наименование вершины.
2. Поток, который до нее можно пропустить.
3. Поток, который уже пропущен до вершины.

**Описание алгоритма.**

Считываем входные данные, заполняем словарь, тем самым формируя граф. После создается экземпляр класса Solver, который хранит в себе граф и данные для формирования промежуточного вывода в ходе алгоритма. Данный класс имеет рекурсивный метод \_solve, который предназначен для поиска пути, от начальной вершины до конечной. Также класс имеет публичный метод solve, который вызывает метод \_solve с необходимыми аргументами. Метод \_solve вызывается поэтапно, он находит возможный путь от начальной до конечной вершины, в графе записываются значения потока, который был пропущен через ту или иную вершину, когда удалось найти путь до конечной вершины. Поток, который запишется до вершины – минимум между доступным потоком до этой вершины и потоком, который был пропущен до этой вершины ранее. Доступный поток вычисляется следующим образом – разность изначально заданного потока и уже пропущенного потока до этой вершины. Данный метод вызывается до тех пор, пока удается найти путь, по которому возможно пропустить поток до конечной вершины. Поскольку в графе на каждом этапе записываются значения потока, которые были пропущены от одной вершине к другой, то алгоритм рано или поздно окончится, так как возможного пути может более не остаться. Поиск пути от начальной до конечной вершины работает на базе поиска в глубину. Используется список посещенных вершин, чтобы повторно не заходить в вершину. Также путь до смежной вершины должен иметь свободный поток, иначе посещать данную вершину не имеет смысла.

**Оценка сложности по памяти**

## Для каждой вершины необходимо хранить список вершин, до которых есть путь, поэтому сложность равна O(N^2).

**Оценка сложности по времени**

На каждом шаге алгоритм добавляет поток увеличивающего пути к уже имеющемуся потоку. На каждом шаге алгоритм увеличивает поток хотя бы на единицу. Значит, что алгоритм завершится не более чем за O(F) шагов, где F — максимальный поток в графе. Можно выполнить каждый шаг за время O(*E*), где *E* — число рёбер в графе, тогда общее время работы алгоритма ограничено O(*Ef*).

**Тестирование**

Результаты тестирования программы можно посмотреть в приложении В.

## Выводы.

Был изучен поиск потока в ориентированном графе алгоритмом Форда-Фалкерсона и написана программа, которая его реализует.

**ПРИЛОЖЕНИЕ А**

**ИСХОДНЫЙ КОД ПРОГРАММЫ**

Название файла: main.py

from typing import Union

class GraphCreator:

@staticmethod

def readGraph():

countEdges = int(input())

source = input()

destination = input()

graph = dict()

for \_ in range(countEdges):

src, dest, length = input().split()

if src not in graph:

graph[src] = []

graph[src].append([dest, int(length), 0])

return source, destination, graph

class Solver:

def \_\_init\_\_(self, source: str, destination: str, graph: dict):

self.\_source = source

self.\_destination = destination

self.\_graph = graph

self.\_visited = []

self.\_stepCounter = 1

self.\_recursionLevel = 0

def \_solve(self, vertex: str, thread: Union[int, float]):

print(f"{'\*' \* self.\_recursionLevel \* 2}Вызов рекурсивной функции с вершиной {vertex},"

f" текущий возможный поток {thread}")

# выходим из рекурсии, если вершина конечная

if vertex == self.\_destination:

print(f"{'\*' \* self.\_recursionLevel \* 2}Текущая рассматриваемая вершина - конечная,"

f" поэтому работа на текущем шагу завершается.")

return thread

# добавляем вершину в список посещенных

self.\_visited.append(vertex)

# проверяем, есть ли смежные вершины к рассматриваемой. Если нет, выходим из рекурсии

if vertex not in self.\_graph:

print(f"{'\*' \* self.\_recursionLevel \* 2}Из данной вершины нет возможный путей,"

f" поэтому оканчиваем работу с ней")

return 0

# Сортируем список смежных вершин (требуется для верного ответа на Степике)

children = sorted(self.\_graph[vertex], key=lambda x: x[0], reverse=True)

print(f"{'\*' \* self.\_recursionLevel \* 2}Вершина {vertex} имеет следующие смежные вершины"

f" - {' '.join([child[0] for child in children])}")

# Рассматриваем смежные вершины

for i, child in enumerate(children):

print(f"{'\*' \* self.\_recursionLevel \* 2}[{i+1}] Рассматриваю смежную вершину {child[0]}")

# Если смежная вершина еще не посещена, а также через нее возможно еще пропустить поток

# То начинаем рассматривать возможный поток из смежной вершины до конечной

if child[0] not in self.\_visited and child[2] < child[1]:

print(f"{'\*' \* self.\_recursionLevel \* 2}[{i+1}] Вершина {child[0]} еще не посещена,"

f" а также из нее еще возможно пропустить поток")

print(f"{'\*' \* self.\_recursionLevel \* 2}[{i+1}] Поэтому начинаем искать возможный поток, который можно"

f" пропустить из данной вершины до конечной")

# текущий возможный поток, который можно пропустить через смежную вершину

maxFlowFromChild = min(thread, child[1] - child[2])

print(f"{'\*' \* self.\_recursionLevel \* 2}[{i+1}] Текущий максимальный поток,"

f" который можно через нее пропустить {maxFlowFromChild}")

self.\_recursionLevel += 1

# вызов рекурсивной функции

delta = self.\_solve(child[0], maxFlowFromChild)

self.\_recursionLevel -= 1

# если через смежную вершину можно пропустить поток до конечной, то delta будет > 0

if delta > 0:

print(f"{'\*' \* self.\_recursionLevel \* 2}[{i+1}] Поток, который можно пропустить"

f" через вершину {child[0]} до конечной вершины - {delta}")

# записываем пропущенный через эту вершину поток

child[2] += delta

return delta

else:

print(f"{'\*' \* self.\_recursionLevel \* 2}[{i+1}] Через вершину {child[0]} нельзя пропустить"

f" поток до конечной вершины")

else:

print(f"{'\*' \* self.\_recursionLevel \* 2}[{i+1}] Вершина {child[0]} уже была"

f" посещена на текущем шагу, либо через нее пропущен максимально возможный поток")

return 0

def solve(self):

result = 0

print(f"Текущий шаг {self.\_stepCounter}")

answer = self.\_solve(self.\_source, float("+inf"))

print()

while answer:

self.\_stepCounter += 1

# запускаем следующий шаг для поиска пути, обнуляем посещенные вершины

result += answer

self.\_visited = []

print(f"Текущий шаг {self.\_stepCounter}")

answer = self.\_solve(self.\_source, float("+inf"))

print()

print(f"С шага {self.\_stepCounter} невозможно пропустить более поток до конечной вершины,"

f" программа завершает работу")

return result

def \_\_str\_\_(self):

text = ""

for vertex in sorted(self.\_graph.keys()):

for neighbor in sorted(self.\_graph[vertex], key=lambda x: x[0]):

text += f"{vertex} {neighbor[0]} {neighbor[2]}\n"

return text

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

solver = Solver(\*GraphCreator.readGraph())

print(solver.solve())

print(solver)

**ПРИЛОЖЕНИЕ В**

**ТЕСТИРОВАНИЕ**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 8  a  h  a c 8  a d 2  c d 16  c f 4  d g 6  g f 18  g h 5  f h 5 | Текущий шаг 1  Вызов рекурсивной функции с вершиной a, текущий возможный поток inf  Вершина a имеет следующие смежные вершины - d c  [1] Рассматриваю смежную вершину d  [1] Вершина d еще не посещена, а также из нее еще возможно пропустить поток  [1] Поэтому начинаем искать возможный поток, который можно пропустить из данной вершины до конечной  [1] Текущий максимальный поток, который можно через нее пропустить 2  \*\*Вызов рекурсивной функции с вершиной d, текущий возможный поток 2  \*\*Вершина d имеет следующие смежные вершины - g  \*\*[1] Рассматриваю смежную вершину g  \*\*[1] Вершина g еще не посещена, а также из нее еще возможно пропустить поток  \*\*[1] Поэтому начинаем искать возможный поток, который можно пропустить из данной вершины до конечной  \*\*[1] Текущий максимальный поток, который можно через нее пропустить 2  \*\*\*\*Вызов рекурсивной функции с вершиной g, текущий возможный поток 2  \*\*\*\*Вершина g имеет следующие смежные вершины - h f  \*\*\*\*[1] Рассматриваю смежную вершину h  \*\*\*\*[1] Вершина h еще не посещена, а также из нее еще возможно пропустить поток  \*\*\*\*[1] Поэтому начинаем искать возможный поток, который можно пропустить из данной вершины до конечной  \*\*\*\*[1] Текущий максимальный поток, который можно через нее пропустить 2  \*\*\*\*\*\*Вызов рекурсивной функции с вершиной h, текущий возможный поток 2  \*\*\*\*\*\*Текущая рассматриваемая вершина - конечная, поэтому работа на текущем шагу завершается.  \*\*\*\*[1] Поток, который можно пропустить через вершину h до конечной вершины - 2  \*\*[1] Поток, который можно пропустить через вершину g до конечной вершины - 2  [1] Поток, который можно пропустить через вершину d до конечной вершины - 2  Текущий шаг 2  Вызов рекурсивной функции с вершиной a, текущий возможный поток inf  Вершина a имеет следующие смежные вершины - d c  [1] Рассматриваю смежную вершину d  [1] Вершина d уже была посещена на текущем шагу, либо через нее пропущен максимально возможный поток  [2] Рассматриваю смежную вершину c  [2] Вершина c еще не посещена, а также из нее еще возможно пропустить поток  [2] Поэтому начинаем искать возможный поток, который можно пропустить из данной вершины до конечной  [2] Текущий максимальный поток, который можно через нее пропустить 8  \*\*Вызов рекурсивной функции с вершиной c, текущий возможный поток 8  \*\*Вершина c имеет следующие смежные вершины - f d  \*\*[1] Рассматриваю смежную вершину f  \*\*[1] Вершина f еще не посещена, а также из нее еще возможно пропустить поток  \*\*[1] Поэтому начинаем искать возможный поток, который можно пропустить из данной вершины до конечной  \*\*[1] Текущий максимальный поток, который можно через нее пропустить 4  \*\*\*\*Вызов рекурсивной функции с вершиной f, текущий возможный поток 4  \*\*\*\*Вершина f имеет следующие смежные вершины - h  \*\*\*\*[1] Рассматриваю смежную вершину h  \*\*\*\*[1] Вершина h еще не посещена, а также из нее еще возможно пропустить поток  \*\*\*\*[1] Поэтому начинаем искать возможный поток, который можно пропустить из данной вершины до конечной  \*\*\*\*[1] Текущий максимальный поток, который можно через нее пропустить 4  \*\*\*\*\*\*Вызов рекурсивной функции с вершиной h, текущий возможный поток 4  \*\*\*\*\*\*Текущая рассматриваемая вершина - конечная, поэтому работа на текущем шагу завершается.  \*\*\*\*[1] Поток, который можно пропустить через вершину h до конечной вершины - 4  \*\*[1] Поток, который можно пропустить через вершину f до конечной вершины - 4  [2] Поток, который можно пропустить через вершину c до конечной вершины - 4  Текущий шаг 3  Вызов рекурсивной функции с вершиной a, текущий возможный поток inf  Вершина a имеет следующие смежные вершины - d c  [1] Рассматриваю смежную вершину d  [1] Вершина d уже была посещена на текущем шагу, либо через нее пропущен максимально возможный поток  [2] Рассматриваю смежную вершину c  [2] Вершина c еще не посещена, а также из нее еще возможно пропустить поток  [2] Поэтому начинаем искать возможный поток, который можно пропустить из данной вершины до конечной  [2] Текущий максимальный поток, который можно через нее пропустить 4  \*\*Вызов рекурсивной функции с вершиной c, текущий возможный поток 4  \*\*Вершина c имеет следующие смежные вершины - f d  \*\*[1] Рассматриваю смежную вершину f  \*\*[1] Вершина f уже была посещена на текущем шагу, либо через нее пропущен максимально возможный поток  \*\*[2] Рассматриваю смежную вершину d  \*\*[2] Вершина d еще не посещена, а также из нее еще возможно пропустить поток  \*\*[2] Поэтому начинаем искать возможный поток, который можно пропустить из данной вершины до конечной  \*\*[2] Текущий максимальный поток, который можно через нее пропустить 4  \*\*\*\*Вызов рекурсивной функции с вершиной d, текущий возможный поток 4  \*\*\*\*Вершина d имеет следующие смежные вершины - g  \*\*\*\*[1] Рассматриваю смежную вершину g  \*\*\*\*[1] Вершина g еще не посещена, а также из нее еще возможно пропустить поток  \*\*\*\*[1] Поэтому начинаем искать возможный поток, который можно пропустить из данной вершины до конечной  \*\*\*\*[1] Текущий максимальный поток, который можно через нее пропустить 4  \*\*\*\*\*\*Вызов рекурсивной функции с вершиной g, текущий возможный поток 4  \*\*\*\*\*\*Вершина g имеет следующие смежные вершины - h f  \*\*\*\*\*\*[1] Рассматриваю смежную вершину h  \*\*\*\*\*\*[1] Вершина h еще не посещена, а также из нее еще возможно пропустить поток  \*\*\*\*\*\*[1] Поэтому начинаем искать возможный поток, который можно пропустить из данной вершины до конечной  \*\*\*\*\*\*[1] Текущий максимальный поток, который можно через нее пропустить 3  \*\*\*\*\*\*\*\*Вызов рекурсивной функции с вершиной h, текущий возможный поток 3  \*\*\*\*\*\*\*\*Текущая рассматриваемая вершина - конечная, поэтому работа на текущем шагу завершается.  \*\*\*\*\*\*[1] Поток, который можно пропустить через вершину h до конечной вершины - 3  \*\*\*\*[1] Поток, который можно пропустить через вершину g до конечной вершины - 3  \*\*[2] Поток, который можно пропустить через вершину d до конечной вершины - 3  [2] Поток, который можно пропустить через вершину c до конечной вершины - 3  Текущий шаг 4  Вызов рекурсивной функции с вершиной a, текущий возможный поток inf  Вершина a имеет следующие смежные вершины - d c  [1] Рассматриваю смежную вершину d  [1] Вершина d уже была посещена на текущем шагу, либо через нее пропущен максимально возможный поток  [2] Рассматриваю смежную вершину c  [2] Вершина c еще не посещена, а также из нее еще возможно пропустить поток  [2] Поэтому начинаем искать возможный поток, который можно пропустить из данной вершины до конечной  [2] Текущий максимальный поток, который можно через нее пропустить 1  \*\*Вызов рекурсивной функции с вершиной c, текущий возможный поток 1  \*\*Вершина c имеет следующие смежные вершины - f d  \*\*[1] Рассматриваю смежную вершину f  \*\*[1] Вершина f уже была посещена на текущем шагу, либо через нее пропущен максимально возможный поток  \*\*[2] Рассматриваю смежную вершину d  \*\*[2] Вершина d еще не посещена, а также из нее еще возможно пропустить поток  \*\*[2] Поэтому начинаем искать возможный поток, который можно пропустить из данной вершины до конечной  \*\*[2] Текущий максимальный поток, который можно через нее пропустить 1  \*\*\*\*Вызов рекурсивной функции с вершиной d, текущий возможный поток 1  \*\*\*\*Вершина d имеет следующие смежные вершины - g  \*\*\*\*[1] Рассматриваю смежную вершину g  \*\*\*\*[1] Вершина g еще не посещена, а также из нее еще возможно пропустить поток  \*\*\*\*[1] Поэтому начинаем искать возможный поток, который можно пропустить из данной вершины до конечной  \*\*\*\*[1] Текущий максимальный поток, который можно через нее пропустить 1  \*\*\*\*\*\*Вызов рекурсивной функции с вершиной g, текущий возможный поток 1  \*\*\*\*\*\*Вершина g имеет следующие смежные вершины - h f  \*\*\*\*\*\*[1] Рассматриваю смежную вершину h  \*\*\*\*\*\*[1] Вершина h уже была посещена на текущем шагу, либо через нее пропущен максимально возможный поток  \*\*\*\*\*\*[2] Рассматриваю смежную вершину f  \*\*\*\*\*\*[2] Вершина f еще не посещена, а также из нее еще возможно пропустить поток  \*\*\*\*\*\*[2] Поэтому начинаем искать возможный поток, который можно пропустить из данной вершины до конечной  \*\*\*\*\*\*[2] Текущий максимальный поток, который можно через нее пропустить 1  \*\*\*\*\*\*\*\*Вызов рекурсивной функции с вершиной f, текущий возможный поток 1  \*\*\*\*\*\*\*\*Вершина f имеет следующие смежные вершины - h  \*\*\*\*\*\*\*\*[1] Рассматриваю смежную вершину h  \*\*\*\*\*\*\*\*[1] Вершина h еще не посещена, а также из нее еще возможно пропустить поток  \*\*\*\*\*\*\*\*[1] Поэтому начинаем искать возможный поток, который можно пропустить из данной вершины до конечной  \*\*\*\*\*\*\*\*[1] Текущий максимальный поток, который можно через нее пропустить 1  \*\*\*\*\*\*\*\*\*\*Вызов рекурсивной функции с вершиной h, текущий возможный поток 1  \*\*\*\*\*\*\*\*\*\*Текущая рассматриваемая вершина - конечная, поэтому работа на текущем шагу завершается.  \*\*\*\*\*\*\*\*[1] Поток, который можно пропустить через вершину h до конечной вершины - 1  \*\*\*\*\*\*[2] Поток, который можно пропустить через вершину f до конечной вершины - 1  \*\*\*\*[1] Поток, который можно пропустить через вершину g до конечной вершины - 1  \*\*[2] Поток, который можно пропустить через вершину d до конечной вершины - 1  [2] Поток, который можно пропустить через вершину c до конечной вершины - 1  Текущий шаг 5  Вызов рекурсивной функции с вершиной a, текущий возможный поток inf  Вершина a имеет следующие смежные вершины - d c  [1] Рассматриваю смежную вершину d  [1] Вершина d уже была посещена на текущем шагу, либо через нее пропущен максимально возможный поток  [2] Рассматриваю смежную вершину c  [2] Вершина c уже была посещена на текущем шагу, либо через нее пропущен максимально возможный поток  С шага 5 невозможно пропустить более поток до конечной вершины, программа завершает работу  10  a c 8  a d 2  c d 4  c f 4  d g 6  f h 5  g f 1  g h 5 |
| 7  a  f  a b 7  a c 6  b d 6  c f 9  d e 3  d f 4  e c 2 | 12  a b 6  a c 6  b d 6  c f 8  d e 2  d f 4  e c 2 |
| 5  a  e  a b 8  b c 10  b e 3  a e 4  c e 2 | 9  a b 5  a e 4  b c 2  b e 3  c e 2 |
| 4  a  c  a b 2  b c 1  c d 1  c a 1 | 1  a b 1  b c 1  c a 0  c d 0 |
| 5  b  d  a c 5  a b 6  c d 3  b c 2  a d 4 | 2  a b 0  a c 0  a d 0  b c 2  c d 2 |