|  |  |  |
| --- | --- | --- |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное автономное  образовательное учреждение высшего образования  «Пермский государственный национальный  исследовательский университет» | | |
|  | Институт компьютерных наук и технологий | |
| **ОТЧЁТ**  по индивидуальной работе №2  по дисциплине «Языки программирования»  Вариант 17 | | |
|  | | Работу выполнил  студент группы ПМИ-9,10-2023 1 курса  Стахеева Е.А.  «18» июня 2024 г. |
| Работу проверил  Рубцова М.Б.  «18» июня 2024 г. |
| Пермь 2024 | | |

СОДЕРЖАНИЕ

[Постановка задачи 3](#_Toc153130027)

[Алгоритм решения 3](#_Toc153130028)

[Тестирование 3](#_Toc153130029)

[Код программы 3](#_Toc153130030)

# Постановка задачи

«Почтальон» Мудрый тритон работает почтальоном в Лукоморье. В перечень его обязанностей входит доставка корреспонденции птицам, разместившим свои гнезда на ветвях дуба (постепенно вытеснив оттуда русалку). Птицы вьют свои гнезда там, где ветка дерева разделяется на несколько более тонких веток. Лукоморье является районом с развитой инфраструктурой, поэтому все разветвления на дубе заняты гнездами птиц. За время работы тритон не только запомнил, где кто живет, но и уяснил, что если его путь проходит через чье-либо гнездо, и его обитатели не получают при этом корреспонденции, ему приходится извиняться за доставленные неудобства. Естественно, интеллигентный тритон стремится как можно меньше беспокоить обитателей дуба, поэтому он заранее планирует свой маршрут так, чтобы приносить минимальное число извинений.

Напишите программу, которая по описанию мест обитания птиц и списку корреспонденции определяет, сколько раз придется извиняться мудрому тритону.

Каждое гнездо дуба обозначено уникальным номером, что соответствует номеру квартиры, который мы иногда указываем, отправляя письма родным и близким по обычной почте. Гнездо, расположенное на первой развилке ствола имеет номер один. Тритон может только ползти по веткам или стволу дуба, прыжки и перелеты полностью исключены. Никакие ветки не соприкасаются и не пересекаются.

Формат входного файла: в первой строке входного файла находится число M – количество гнезд на дубе (0<M<1000) В следующих M строках находится информация о размещении гнезд и количестве писем для каждого и возможных адресатов. Первые два числа ni и li – количество соседей гнезда с номером i и количество направленных по этому адресу писем соответственно. Далее в строке приводится ni чисел – номера гнезд, являющихся соседними с гнездом c номером i. Формат выходного файла: в единственной строке выходного файла должно содержаться единственное число – минимальное количество извинений, которые придется приносить мудрому тритону, для того чтобы доставить все письма.

Пример:

(входной файл)

5

3 2 2 5 4

1 1 1

1 1 4

2 2 3 1

1 3 1

(выходной файл)

2

# Алгоритм решения

1. Сначала программа читает входной файл, который содержит информацию о гнездах и их взаимосвязях. Для каждого гнезда программа создает экземпляр класса «Nest», хранящий информацию о количестве писем, которое нужно доставить в гнездо («nest\_letters»), количество писем, которые должны быть переданы в поддерево («branch\_letters»), список соседних гнезд («neighbourhood») и флаг посещения («visited»).

2. Функция «calc\_letters» рекурсивно обходит дерево гнезд, начиная с заданного гнезда, и рассчитывает общее количество писем, которые необходимо доставить в каждое гнездо, включая поддеревья. Для этого она увеличивает «branch\_letters» каждого гнезда на количество писем из его соседей.

3. После расчета количества писем программа использует функцию «pardon\_count» для подсчета количества извинений, которые должен принести почтальон за прохождение через гнезда без доставки писем. Эта функция также рекурсивно обходит дерево гнезд, но учитывает только те гнезда, в которых есть письма для доставки или которые были пропущены без извинений.

4. Перед обработкой входных данных программа выполняет проверку на корректность данных в файле, используя функцию «check\_data\_correctness». Это обеспечивает правильную работу программы даже при наличии ошибок в формате файла.

5. После выполнения всех вычислений программа записывает полученный результат (количество извинений) в выходной файл.

6. Все эти шаги объединяет функция «main», которая управляет процессом чтения данных, расчета и записи результата. Она также выводит примерное описание задачи и процесса работы почтальона.

# Тестирование

1. Тест:

Входной файл:

5

3 2 2 5 4

1 1 1

1 1 4

2 2 3 1

1 3 1

Выходной файл:

2

Вывод для пользователя:

Мудрый тритон работает почтальоном в Лукоморье.

В перечень его обязанностей входит доставка корреспонденции птицам, разместившим свои гнезда на ветвях дуба.

Когда тритону приходится проходить через гнездо, не доставляя туда почту, он должен извиниться.

Дуб:

1

/ | \

2 5 4

\

3

Количество извинений, которые должен принести тритон, проходя по этому дубу: 2

2. Тест:

Входной файл:

5

3 3 2 5 4

1 1 1

1 1 4

2 3 3 1

1 2 1

Выходной файл:

1

Вывод для пользователя:

Мудрый тритон работает почтальоном в Лукоморье.

В перечень его обязанностей входит доставка корреспонденции птицам, разместившим свои гнезда на ветвях дуба.

Когда тритону приходится проходить через гнездо, не доставляя туда почту, он должен извиниться.

Дуб:

1

/ | \

2 5 4

\

3

Количество извинений, которые должен принести тритон, проходя по этому дубу: 1

3. Тест:

Входной файл:

50

3 2 2 5 4

1 1 1

1 1 4

2 2 3 1

1 3 1

Выходной файл:

Произошла ошибка при чтении файла input.txt: Число строк не соответствует первому числу в файле.

4. Тест:

Входной файл:

0

3 3 2 5 4

1 1 1

1 1 4

2 3 3 1

1 2 1

Выходной файл:

Произошла ошибка при чтении файла input.txt: Число гнезд должно быть в диапазоне 0 < m < 1000.

5. Тест:

Если файл не создан:

Вывод для пользователя:

Файл input.txt не найден.

# Код программы

class Nest:

def \_\_init\_\_(self, nest\_letters=0, branch\_letters=0, neighbourhood=None, visited=False):

self.nest\_letters = nest\_letters # письма, которые осталось доставить в это гнездо

self.branch\_letters = branch\_letters # письма, которые нужно передать в поддереве, корнем которого является текущий узел

self.neighbourhood = neighbourhood

self.visited = visited # Флаг, указывающий был ли уже посещен этот объект

def calc\_letters(nests, nest\_number): # Рассчитывает количество писем для гнезда, начиная с заданного номера гнезда

nest = nests[nest\_number]

nest.visited = True # Помечаем гнездо как посещенное

nest.branch\_letters = nest.nest\_letters

for j in range(len(nest.neighbourhood)):

neighbour\_id = nest.neighbourhood[j]

if nests[neighbour\_id].visited == False: # Если соседнее гнездо еще не было посещено

calc\_letters(nests, neighbour\_id)

nest.branch\_letters += nests[neighbour\_id].branch\_letters # Добавляем к общему количеству писем в гнезде количество писем из соседнего гнезда

def pardon\_count(nests, nest\_number, current\_pass=1): # Считает количество извинений

nest = nests[nest\_number]

pardons = 0 # Количество извинений

nest.visited = True # Помечаем гнездо как посещенное

if nest.nest\_letters <= 0: # Если в гнезде нет писем

pardons += 1 # Увеличиваем количество извинений на 1

else:

nest.nest\_letters -= 1 # Иначе уменьшаем количество писем

for j in range(len(nest.neighbourhood)): # Для каждого соседнего гнезда

neighbour\_id = nest.neighbourhood[j]

neighbour\_id = nest.neighbourhood[j]

if nests[neighbour\_id].branch\_letters <= 0: # Если в соседнем гнезде нет писем

continue # Пропускаем это гнездо

if nests[neighbour\_id].visited == False: # Если соседнее гнездо еще не было посещено

pardons += pardon\_count(nests, neighbour\_id)

if nest.nest\_letters <= 0: # Если в текущем гнезде нет писем

pardons += 1 # Увеличиваем количество извинений на 1

else:

nest.nest\_letters -= 1 # Иначе уменьшаем количество писем

return pardons

def check\_data\_correctness(filename):

try:

with open(filename, 'r') as file:

# Чтение первой строки файла, содержащей количество гнезд

m = int(file.readline())

# Проверка, что количество гнезд находится в допустимом диапазоне (от 0 до 1000)

if m <= 0 or m >= 1000:

raise ValueError("Число гнезд должно быть в диапазоне 0 < m < 1000") # Вывод сообщения об ошибке, если файл не найден

# Проверяем, что следующих m строк существуют

for \_ in range(m):

line = file.readline().strip()

if not line:

raise ValueError("Число строк не соответствует первому числу в файле.")

except FileNotFoundError:

raise SystemExit(f"Файл {filename} не найден.") # Вывод сообщения об ошибке, если файл не найден

except Exception as e:

raise SystemExit(f"Произошла ошибка при чтении файла {filename}: {e}") # Обработка других исключений

return True

def read\_input\_file(filename):

nests = [] # Создаем список гнезд

with open(filename, 'r') as file:

# Чтение первой строки файла, содержащей количество гнезд

m = int(file.readline())

# Цикл по количеству гнезд

for \_ in range(m):

line = file.readline().strip() # Чтение строки с информацией о гнезде

if len(line.split()) > 2:

neighbourhood\_count, nest\_letters = map(int, line.split()[:2]) # Разбор количества соседей и писем гнезда

neighbours = [] # Создаем список соседних гнезд

line = file.readline()

numbers = line.split()

# Добавление индексов соседних гнезд в список neighbours, уменьшаем каждый индекс на 1, так как индексы начинаются с 0

for number in numbers:

neighbours.append(int(number) - 1)

nests.append(Nest(nest\_letters=nest\_letters, branch\_letters=0, neighbourhood=neighbours, visited=False))

return nests

def write\_output\_file(filename, result): # Записывает результат в файл

with open(filename, 'w') as file:

file.write(str(result))

def main():

input\_filename = "input.txt"

output\_filename = "output.txt"

if not check\_data\_correctness(input\_filename):

print("Данные в файле некорректны.")

return

nests = read\_input\_file(input\_filename) # Читаем данные из файла

calc\_letters(nests, 0) # Рассчитываем количество писем

for nest in nests: # Сбрасываем флаги посещения для всех гнезд

nest.visited = False

result = pardon\_count(nests, 0) # Считаем количество извиненийа

write\_output\_file(output\_filename, result) # Записываем результат в файл

# Пример использования функции для проверки файла перед его обработкой

print('Мудрый тритон работает почтальоном в Лукоморье.')

print('В перечень его обязанностей входит доставка корреспонденции птицам, разместившим свои гнезда на ветвях дуба.')

print('Когда тритону приходится проходить через гнездо, не доставляя туда почту, он должен извиниться.')

print('Дуб:')

print(' 1')

print(' / | \ ')

print(' 2 5 4')

print(' \ ')

print(' 3')

print('Количество извинений, которые должен принести тритон, проходя по этому дубу:', result)

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

main()