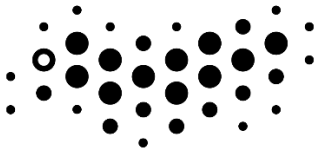
Санкт-Петербургский национальный
исследовательский университет ИТМО

Факультет систем управления и робототехники



ITMO UNIVERSITY

Алгоритмы и структуры данных.
Отчет №5.

Выполнил студент гр. R32362

Лаптев Максим Сергеевич

Санкт-Петербург 2023

Содержание

Цель

Решить данные задачи, написав код, работа которого удовлетворяет условиям

1067. Структура папок

Ограничение времени: 2.0 секунды

Ограничение памяти: 64 МБ

Хакер Билл случайно потерял всю информацию с жесткого диска своего компьютера, и у него нет резервных копий его содержимого. Но он сожалеет не о потере самих файлов, а о потере очень понятной и удобной структуры папок, которую он создавал и сохранял в течение многих лет работы.

К счастью, у Билла есть несколько копий списков папок с его жесткого диска. С помощью этих списков он смог восстановить полные пути к некоторым папкам (например, «WINNT\SYSTEM32\CERTSRV\CERTCO~1\X86»). Он поместил их все в файл, записав каждый найденный путь в отдельную строку.

Напишите программу, которая восстановит структуру папок Билла и выведет ее в виде отформатированного дерева.

Исходные данные

Первая строка содержит целое число N – количество различных путей к папкам ($1 \le N \le 500$). Далее следуют N строк с путями к папкам. Каждый путь занимает одну строку и не содержит пробелов, в том числе, начальных и конечных. Длина каждого пути не превышает 80 символов. Каждый путь встречается в списке один раз и состоит из нескольких имен папок, разделенных обратной косой чертой («\»).

Имя каждой папки состоит из 1-8 заглавных букв, цифр или специальных символов из следующего списка: восклицательный знак, решетка, знак доллара, знак процента, амперсанд, апостроф, открывающаяся и закрывающаяся скобки, знак дефиса, собаки, циркумфлекс, подчеркивание, гравис, открывающаяся и закрывающаяся фигурная скобка и тильда («`!#$%&'()-@^_`{}~`»).

Результат

Выведите отформатированное дерево папок. Каждое имя папки должно быть выведено в отдельной строке, перед ним должно стоять несколько пробелов, указывающих на глубину этой папки в иерархии. Подпапки должны быть перечислены в лексикографическом порядке непосредственно после их родительской папки; перед их именем должно стоять на один пробел больше, чем перед именем их родительской папки. Папки верхнего уровня выводятся без пробелов и также должны быть перечислены в лексикографическом порядке.

Пример

исходные данные	результат

7
WINNT\SYSTEM32\CONFIG
GAMES
WINNT\DRIVERS
HOME
WIN\SOFT
GAMES\DRIVERS
WINNT\SYSTEM32\CERTSRV\CERTCO~1\X86
	GAMES
	 DRIVERS
	HOME
	WIN
	 SOFT
	WINNT
	 DRIVERS
	 SYSTEM32
	  CERTSRV
	  CERTCO~1
	  X86

Исходный код:

```
n = int(input())
cities = dict()
people = []
max_cities = dict()
for i in range(n):
    name, city, money = input().split()
    money = int(money)
    people.append([name, city, money])
    cities[city] = cities.get(city, 0) + money
m, k = map(int, input().split())
moves = []
for i in range(k):
```

```
    day, name, city_move = input().split()
    moves.append((int(day), name, city_move))
for i in range(1, m+1):
    max_sum = 0
    flag = False
    for city, value in cities.items():
        if value == max_sum:
            flag = False
        if value > max_sum:
            max_sum = value
            max_city = city
            flag = True
    if flag: max_cities[max_city] = max_cities.get(max_city, 0) + 1
    count = 0
    for j in range(len(moves)):
        if moves[j][0] == i:
            count += 1
            for human in people:
                if human[0] == moves[j][1]:
                    cities[human[1]] -= human[2]
                    to_city = moves[j][2]
                    human[1] = to_city
                    cities[to_city] = cities.get(to_city, 0) + human[2]
                    break
        else:
            break
    for _ in range(count):
        moves.pop(0)
for city in sorted(max_cities.keys()):
    print(city, max_cities[city])
```

Ограничения небольшие, времени много, так что задача скорее на реализацию – вопрос в том, где хранить данные. В python есть словари, многократной вложенностью получаем имитацию файловой системы – в каждой папке (словаре) свои подпапки (также словари) и тд. Если путь есть, доходим до его конца и создаем новую «ветку». В конце, когда вся файловая система заполнена, просто выводим результат.

Язык программирования: python

Результат

10275255	15:00:43 12 май 2023	Maxim	1067. Структура папок	Python 3.8 x64	Accepted		0.156	4 964 КБ

1650. Миллиардеры

Ограничение времени: 3.0 секунды

Ограничение памяти: 64 МБ

Возможно, вы знаете, что из всех городов мира больше всего миллиардеров живёт в Москве. Но, поскольку работа миллиардера подразумевает частые перемещения по всему свету, в определённые дни какой-то другой город может занимать первую строчку в таком рейтинге. Ваши приятели из ФСБ, ФБР, MI5 и Шин Бет скинули вам списки перемещений всех миллиардеров за последнее время. Ваш работодатель просит посчитать, сколько дней в течение этого периода каждый из городов мира был первым по общей сумме денег миллиардеров, находящихся в нём.

Исходные данные

В первой строке записано число n — количество миллиардеров ($1 \le n \le 10000$). Каждая из следующих n строк содержит данные на определённого человека: его имя, название города, где он находился в первый день данного периода, и размер состояния. В следующей строке записаны два числа: m — количество дней, о которых есть данные ($1 \le m \le 50000$), k — количество зарегистрированных перемещений миллиардеров ($0 \le k \le 50000$). Следующие k строк содержат список перемещений в формате: номер дня (от 1 до $m - 1$), имя человека, название города назначения. Вы можете считать, что миллиардеры путешествуют не чаще одного раза в день, и что они отбывают поздно вечером и прибывают в город назначения рано утром следующего дня. Список упорядочен по возрастанию номера дня. Все имена и названия городов состоят не более чем из 20 латинских букв, регистр букв имеет значение. Состояния миллиардеров лежат в пределах от 1 до 100 миллиардов.

Результат

В каждой строке должно содержаться название города и, через пробел, количество дней, в течение которых этот город лидировал по общему состоянию миллиардеров, находящихся в нём. Если таких дней не было, пропустите этот город. Города должны быть отсортированы по алфавиту (используйте обычный порядок символов: ABC...Zabc...z).

Пример

исходные данные	результат
5 Abramovich London 15000000000 Deripaska Moscow 10000000000 Potanin Moscow 5000000000 Berezovsky London 2500000000 Khodorkovsky Chita 1000000000 25 9 1 Abramovich Anadyr 5 Potanin Courchevel 10 Abramovich Moscow 11 Abramovich London 11 Deripaska StPetersburg 15 Potanin Norilsk 20 Berezovsky Tbilisi 21 Potanin StPetersburg 22 Berezovsky London	Anadyr 5 London 14 Moscow 1

Исходный код:

```
n = int(input())
cities = dict()
people = dict()
max_cities = dict()
for i in range(n):
    name, city, money = input().split()
    money = int(money)
    people[name] = [city, money]
    cities[city] = cities.get(city, 0) + money

m, k = map(int, input().split())
prev_day = 0
```

```
for i in range(k):
    day, name, city_move = input().split()
    day = int(day)
    human = people.get(name)
    if day != prev_day:
        max_sum = 0
        for city, value in cities.items():
            if value == max_sum:
                flag = False
            elif value > max_sum:
                max_sum = value
                max_city = city
                flag = True
        if flag: max_cities[max_city] = max_cities.get(max_city, 0) + (day -
prev_day)

    cities[human[0]] -= human[1]
    if cities[human[0]] == 0:
        del cities[human[0]]
    human[0] = city_move
    cities[city_move] = cities.get(city_move, 0) + human[1]

    prev_day = day

max_sum = 0
for city, value in cities.items():
    if value == max_sum:
        flag = False
    elif value > max_sum:
        max_sum = value
        max_city = city
        flag = True
if flag: max_cities[max_city] = max_cities.get(max_city, 0) + (m - prev_day)

for city in sorted(max_cities.keys()):
    print(city, max_cities[city])
```

Хранить данные также будем в словарях – города, люди и ответ (города с максимальной посещаемостью). Заполняем входными данными и при вводе путешествий начинаем алгоритм. Каждый раз, когда меняется К, будем вести подсчёт какой город сейчас лидирует (для этого находим 2 максимальных). И на следующий итерации добавляем этому городу дни, когда он лидировал в словарь с ответом.

Язык программирования: python

Результат

ID	Дата	Автор	Задача	Язык	Результат проверки	№ теста	Время работы	Выделено памяти
10275513	18:25:29 12 май 2023	Maxim	1650. Миллиардеры	PyPy 3.8 x64	Accepted		0.64	12 868 КБ

1080. Раскраска карты

Ограничение времени: 1.0 секунды
Ограничение памяти: 64 МБ

Рассмотрим географическую карту с N странами, занумерованными от 1 до N ($0 < N < 99$). Для каждой страны известны номера соседних стран, т.е. имеющих общую границу с данной. Из каждой страны можно попасть в любую другую, перейдя некоторое количество границ. Напишите программу, которая определит, возможно ли покрасить карту только в два цвета — красный и синий — так, что если две страны имеют общую границу, их цвета различаются. Цвет первой страны — красный. Ваша программа должна вывести одну возможную раскраску для остальных стран или сообщить, что такая раскраска невозможна.

Исходные данные

В первой строке записано число N. Из следующих N строк i-я строка содержит номера стран, с которыми i-я страна имеет границу. Каждое целое число в i-й строке больше, чем i, кроме последнего, которое равно 0 и обозначает конец списка соседей i-й страны. Если строка содержит 0, это значит, что i-я страна не соединена ни с одной страной с бо́льшим номером.

Результат

Вывод содержит ровно одну строку. Если раскраска возможна, эта строка должна содержать список нулей и единиц без разделителей между ними. i-я цифра в этой последовательности обозначает цвет i-й страны. 0 соответствует красному цвету, единица — синему. Если раскраска невозможна, выведите целое число –1.

Пример

исходные данные	результат
3 2 0 3 0 0	010

Исходный код:

```
def bfs(graph, node):
    queue.append(node)

    while queue:
        m = queue.pop(0)

        for neighbour in graph[m]:
            if colors[neighbour] == -1:
                colors[neighbour] = 1 if colors[m] == 0 else 0
                queue.append(neighbour)
            else:
                if colors[neighbour] == colors[m]:
                    return False
    return True


n = int(input())
neighbors = dict()
colors = dict()
for i in range(n):
    x = input().split()[:-1]
    for el in x:
        el = int(el)
        if neighbors.get(i + 1, False):
            neighbors[i + 1].append(el)
        else:
            neighbors[i + 1] = [el]

        if neighbors.get(el, False):
            neighbors[el].append(i + 1)
```

```
        else:
            neighbors[el] = [i + 1]
    if x == [] and not neighbors.get(i + 1, False):
        neighbors[i + 1] = []
    colors[i + 1] = -1

queue = []
colors[1] = 0
if bfs(neighbors, 1):
    for i in range(1, n + 1):
        print(colors[i], end="")
else:
    print(-1)
```

Для решения использовался простейший поиск в ширину. С одним лишь отличием – окрашиваем вершину в противоположный цвет предыдущей вершины. Таким образом, соседи окрашиваются в разные цвета. Однако если мы встретим двух соседей с одинаковой окраской, значит покраска невозможна. Возвращаем -1, иначе выводим ответ из словаря colors.

Язык программирования: python

Результат:

ID	Дата	Автор	Задача	Язык	Результат проверки	№ теста	Время работы	Выделено памяти
10283647	13:42:06 16 май 2023	Maxim	1080. Раскраска карты	Python 3.8 x64	Accepted		0.093	396 КБ

Вывод

В результате проделанной работы были успешно решены 3 задачи, используя язык программирования python . Работа программ выдает правильный ответ, а также удовлетворяет условиям задачи (проходит по времени и памяти).