Assignment 3.2 Practice Problem 2 (Split the Bill)

Problem

In the Splitwise app, people form groups and add the expenses of members of the group. This is especially useful for vacations, where people traveling in a group can maintain an account of their expenses and who paid the bills.

All people in the group are assigned distinct IDs between 1 and N, where N is the size of the group.

In addition to keeping a record of the expenditure, Splitwise also calculates the list of shortest-path transfers (defined later) that will settle up all dues.

Each transaction has the following parameters:-

transaction_id - It is a string representing the unique ID by which the transaction is identified. paid_by - It is a list of lists, where each element of the list is another list having the form [x, y]. Here, x and y denote that person having ID x paid Rs. y. split_as - It is a list of lists, where each element of the list is another list having the form [x, y]. Here, x and y denote that after all dues are settled, a person having ID x will ultimately contribute Rs. y to the transaction.

For any given transaction, the following condition holds true:-

Total_Amt_Paid = Sum_of_all_splits

In other words, the sum total of all amounts in list paid_by equals the sum total of all amounts in list split_as.

Following is the example of a transaction in a group of size N=64:-

transaction_id : "#f1230" paid_by : [[1, 30], [4, 100], [63, 320]] split_as : [[1, 120], [2, 20], [3, 40], [4, 40], [37, 100], [51, 40], [53, 90]]

Shortest-Path Transfers: Shortest-path transfers lead to a reduction in the number of transfers.

Specifically, for a group having multiple transactions, the shortest-path transfers will be a list of payments to be made such that:-

Each payment can be represented by a list of the following form:- [payer_id, payee_id, amount]. There is only 1 payer, and 1 payee in each payment, which are distinct from each other. So, payer_id != payee_id, for any payment. Each person (out of the N people) can only either be the payer (in all payments involving him), or the payee, but not both.

The total amount of money that each person should receive/spend, must be equal to the total amount he would receive/spend according to the given list of transactions.

Clearly, there can be several shortest-path transfers for a particular list of transactions.

Specifically, the lexicographically smallest shortest path has the following:-

Arrange people who have borrowed money in ascending order of their IDs. Do the same for people who have lent money.

Now, construct payments so that the least borrower ID has to pay the least lender ID. Continue this process, till all debts have been settled. Task

Given N members in a group, and lists representing the transactions(expenses), print the payments involved in the lexicographically smallest shortest-path transfers for the group.

Example

Input:

- N = 4

5 transactions, that can be represented as follows:-

- transaction_id = "#a1234", paid_by = [[1, 60]], split_as = [[2, 60]].
- transaction_id = "#a2142", paid_by = [[2, 40]], split_as = [[3, 40]].
- transaction_id = "#b3310", paid_by = [[3, 30]], split_as = [[4, 30]].
- transaction_id = "#b2211", paid_by = [[4, 30]], split_as = [[3, 30]].
- transaction_id = "#f1210", paid_by = [[3, 20]], split_as = [[1, 20]].

Output:

- 2 payments (of the form [payer_id, payee_id, amount]) are to be made, represented by the list:-
- [[1, 2, 20], [1, 3, 20]]

Approach:

- The given list of payments satisfies all three necessary conditions. Hence, it is a Shortest-Path Transfer.

Function description

Complete the function solve. This function takes the following 2 parameters and returns the required answer:

- N: An integer, representing the number of people in the group.
- transaction_list: A list (vector) of transactions. Each transaction is a dictionary, having keys “transaction_id”, “paid_by” and “split_as”. (The contents of each transaction are explained above)

Input format

Note: This is the input format that you must use to provide custom input (available above the Compile and Test button).

- The first line contains two space-separated integers N and M, the number of people in the group, and the number of transactions recorded. The next lines describe the M transactions as follows:- Each new transaction begins from a new line.
- The first line of each transaction contains a string, representing the transaction_id of the transaction.
- The 2nd line of each transaction contains 2 space-separated integers n_payers and n_splits.
- n_payers denotes the number of people in the paid_by list. n_splits denotes the number of people in the split_as list.
- The next n_payers lines contain two space-separated integers, the payer and the amount paid.
- The next n_splits lines contain two space-separated integers, the borrower and the amount borrowed.

Output format

Print the answer in the given format.

- In the first line, print a single integer K, denoting the number of payments involved in the Shortest Path Transfer.
- The next K lines should represent the K payments. Each payment should be printed in a single line as 3 space-separated integers payer_id, payee_id, and amount. Here, payer_id is the ID of the person who needs to pay the amount of money to the person with ID payee_id

Constraints

- 2 ≤ N ≤ $2*10^5$
- 1 ≤ M ≤ 5000
- 1 ≤ len(transaction[i][paid_by]) + len(transaction[i][split_as]) ≤ 50
- 1 ≤ total_money_exchanged_in_each_transaction ≤ 10^7

```
# input number of people and transactions
n_people, n_transactions = map(int, input("Enter number of people and transactions: ").split())

# create balance list for each person
balance = [0] * n_people

# initialize transaction list
transactions = []

# loop number of transactions and track the balance of each person
for _ in range(n_transactions):
  payers, receivers = input("Enter transaction details (payers / receivers): ").split()

  for _ in range(int(payers)):
    p_id, amount = map(int, input("Enter payer details (id / amount): ").split())
    balance[p_id - 1] -= amount

  for _ in range(int(receivers)):
    r_id, amount = map(int, input("Enter receiver details (id / amount): ").split())
    balance[r_id - 1] += amount
```

```
# find and output transactions
for i in range(n_people):
  # if a person has a positive balance, find a person with a negative balance to pair with
  if balance[i] > 0:
    current = balance[i]
    pointer = 0

    while current > 0 and pointer < n_people:
      # continue to next person if person has a positive balance
      if balance[pointer] >= 0:
        pointer += 1
        continue

      # determine smallest amount for the transfer and adjust the balances of both individuals accordingly
      min_amount = min(current, abs(balance[pointer]))
      current -= min_amount
      balance[pointer] += min_amount

      transactions.append(((i + 1, pointer + 1), min_amount))

print("\nResulting transactions:")
for transaction in transactions:
  print(transaction[0][0], transaction[0][1], transaction[1])
```

```
Enter number of people and transactions: 6 5
Enter transaction details (payers / receivers): 2 3
Enter payer details (id / amount): 1 25
Enter payer details (id / amount): 3 15
Enter receiver details (id / amount): 4 10
Enter receiver details (id / amount): 5 25
Enter receiver details (id / amount): 6 5
Enter transaction details (payers / receivers): 1 4
Enter payer details (id / amount): 4 100
Enter receiver details (id / amount): 1 25
Enter receiver details (id / amount): 2 25
Enter receiver details (id / amount): 3 25
Enter receiver details (id / amount): 4 25
Enter transaction details (payers / receivers): 2 2
Enter payer details (id / amount): 5 30
Enter payer details (id / amount): 3 10
Enter receiver details (id / amount): 1 25
Enter receiver details (id / amount): 4 15
Enter transaction details (payers / receivers): 1 3
Enter payer details (id / amount): 2 150
Enter receiver details (id / amount): 1 50
Enter receiver details (id / amount): 2 50
Enter receiver details (id / amount): 3 50
Enter transaction details (payers / receivers): 2 2
Enter payer details (id / amount): 5 13
Enter payer details (id / amount): 6 25
Enter receiver details (id / amount): 4 25
Enter receiver details (id / amount): 1 13

Resulting transactions:
1 2 75
1 4 13
3 4 12
3 5 18
3 6 20
```