```
def solve(N, transaction_list):
    # Parse the input to extract transaction details
    net_amount = [0] * (N + 1)
    for transaction in transaction_list:
        paid_by = transaction['paid_by']
        split_as = transaction['split_as']
        for payer in paid_by:
            net_amount[payer[0]] -= payer[1]
        for split in split_as:
            net_amount[split[0]] += split[1]

    #  payments for the lexicographically smallest shortest-path transfers
    payments = []
    borrowers = []
    lenders = []
    for i in range(1, N + 1):
        if net_amount[i] < 0:
            borrowers.append((i, -net_amount[i]))
        elif net_amount[i] > 0:
            lenders.append((i, net_amount[i]))

    borrowers.sort(key=lambda x: x[0])  # Sort borrowers by ID
    lenders.sort(key=lambda x: x[0])     # Sort lenders by ID

    for borrower in borrowers:
        borrower_id, borrower_amount = borrower
        while borrower_amount > 0:
            lender_id, lender_amount = lenders[0]
            payment_amount = min(borrower_amount, lender_amount)
            payments.append([lender_id, borrower_id, payment_amount])
            borrower_amount -= payment_amount
            lender_amount -= payment_amount
            if lender_amount == 0:
                lenders.pop(0)
            else:
                lenders[0] = (lender_id, lender_amount)

    # Print the payments involved in the lexicographically smallest shortest-path transfers
    print(len(payments))
    for payment in payments:
        print(*payment)


# parsing and function call
if __name__ == "__main__":
    N, M = map(int, input().split())
    transaction_list = []
    for _ in range(M):
        transaction_id = input().strip()
        n_payers, n_splits = map(int, input().split())
        paid_by = [list(map(int, input().split())) for _ in range(n_payers)]
        split_as = [list(map(int, input().split())) for _ in range(n_splits)]
        transaction = {'transaction_id': transaction_id, 'paid_by': paid_by, 'split_as': split_as}
        transaction_list.append(transaction)

    solve(N, transaction_list)
```