Optimal Account Balancing

A group of friends went on holiday and sometimes lent each other money. For example, Alice paid for Bill's lunch for $10. Then later Chris gave Alice $5 for a taxi ride. We can model each transaction as a tuple (x, y, z) which means person x gave person y $z. Assuming Alice, Bill, and Chris are person 0, 1, and 2 respectively (0, 1, 2 are the person's ID), the transactions can be represented as `[[0, 1, 10], [2, 0, 5]]`.

Given a list of transactions between a group of people, return the minimum number of transactions required to settle the debt.

Note:

1. A transaction will be given as a tuple (x, y, z). Note that `x ≠ y` and `z > 0`.
2. Person's IDs may not be linear, e.g. we could have the persons 0, 1, 2 or we could also have the persons 0, 2, 6.

Example 1:

```
Input:
[[0,1,10], [2,0,5]]

Output:
2

Explanation:
Person #0 gave person #1 $10.
Person #2 gave person #0 $5.

Two transactions are needed. One way to settle the debt is person #1 pays person #0
 and #2 $5 each.
```

Example 2:

```
Input:
[[0,1,10], [1,0,1], [1,2,5], [2,0,5]]

Output:
1

Explanation:
Person #0 gave person #1 $10.
Person #1 gave person #0 $1.
Person #1 gave person #2 $5.
Person #2 gave person #0 $5.

Therefore, person #1 only need to give person #0 $4, and all debt is settled.
```

Solution 1

Given the input [[0,1,1], [2,3,2], [4,5,3], [6,7,4], [8,9,5], [10,11,6], [12,13,7], [14,15,2], [14,16,2], [14,17,2], [14,18,2]]

The provided expected result is 14. However, there are only 11 transactions, so the upper bound of the result should be 11.

written by tcui original link here

Solution 2

Think about this case:
[[0,3,2],[1,4,3],[2,3,2],[2,4,2]]

The correct answer should be 3, but the so-called standard answer is 4.
Actually this question seems to be a NPC problem. And I don't know why there is no data scale.

written by nyunyunyunyu original link here

Solution 3

```
import java.util.*;

public class Solution {
    public int minTransfers(int[][] trans) {
        Map<Integer, Integer> net = new HashMap<>();
        for(int i = 0; i < trans.length; i++){
            net.put(trans[i][0], net.getOrDefault(trans[i][0], 0) - trans[i][2]);
            net.put(trans[i][1], net.getOrDefault(trans[i][1], 0) + trans[i][2]);
        }
        int[] temp = new int[net.size()];
        int i = 0;
        for(int j : net.values()){
            if(j != 0)temp[i++] = j;
        }
        int[] a = new int[i];
        System.arraycopy(temp, 0, a, 0, i);
        transactions.clear();
        number = Integer.MAX_VALUE;
        mintran(a, 0);
        return number;
    }

    private List<int[]> transactions = new ArrayList<>();
    private int number = Integer.MAX_VALUE;

    private void mintran(int[] a, int start){
        //System.out.println(Arrays.toString(a));
        if(transactions.size() >= number) return;
        if(number == (a.length + 1)/2) return;

        if(a.length < 2){
            number = 0;
            return;
        }else if(a.length == 2) {
            number = a[0] == 0 ? 0 : 1;
            return;
        }else{
            int ind = -1;
            int max = Integer.MIN_VALUE;
            int i = start;
            for(; i < a.length; i++){
                if(Math.abs(a[i]) > max){
                    max = Math.abs(a[i]);
                    ind = i;
                }
            }

            if(max == 0 || start == a.length){
                if(transactions.size() < number){
                    number = transactions.size();
                }
                return;
            }

            
```

```
            int temp = a[ind];
            a[ind] = a[start];
            a[start] = temp;

            for(i = start + 1; i < a.length; i++){
                if(a[i] * a[start] < 0) {
                    transactions.add(new int[]{a[i], a[start]});
                    temp = a[i];
                    a[i] += a[start];
                    mintran(a, start + 1);
                    a[i] = temp;
                    transactions.remove(transactions.size()-1);
                }
            }

            temp = a[ind];
            a[ind] = a[start];
            a[start] = temp;

        }
    }

    public static void main(String[] args) {
        int[][] A = new int[][] {{0,1,1}, {2,3,2}, {4,5,3}, {6,7,4}, {8,9,5}, {1
0,11,6}, {12,13,7}, {14,15,2}, {14,16,2}, {14,17,2}, {14,18,2}};
        Solution solution = new Solution();
        System.out.println(solution.minTransfers(A));
    }
}
```

written by fatalme original link here