are two minimal such cliques.

[4, -2, -2] and [6, -6]

The beautiful part of those cliques are that we only need to resolve the balance inside the clique. There will be no inter clique transactions.

The final number of transactions will be 5 - 2, which is num_non_zero - num_clique.

To find the minimal clique is the perfect problem to be solved by BFS. We will start with one element in the set as the root, find the minimal length path in the tree which sum up to be zero. Remove that zero sum set and pick another root.

```
class Solution(object):
    def minTransfers(self, transactions):
        """
        :type transactions: List[List[int]]
        :rtype: int
        """
        def remove_one_zero_clique(non_zero):
            n = len(non_zero)
            q = collections.deque()
            # q store ([index set], sum of set)
            q.append(([0], non_zero[0]))
            min_zero_set = None

            while q:
                cur_set, cur_sum = q.popleft()
                if cur_sum == 0:
                    min_zero_set = cur_set
                    break
                for j in range(cur_set[-1] + 1, n):
                    q.append((cur_set + [j], cur_sum + non_zero[j]))

            min_zero_set = set(min_zero_set)
            return [non_zero[i] for i in range(n) if i not in min_zero_set]


        bal = collections.defaultdict(int)
        for t in transactions:
            bal[t[0]] -= t[2]
            bal[t[1]] += t[2]
        non_zero = [bal[k] for k in bal if bal[k] != 0]

        bal_cnt = len(non_zero)
        while len(non_zero) > 0:
            non_zero = remove_one_zero_clique(non_zero)
            bal_cnt -= 1
        return bal_cnt
```

Comments: 7 Best Most Votes Newest to Oldest Oldest to Newest

Type comment here... (Markdown is supported)

Post

chao4 ★ 31 June 20, 2019 11:13 PM

Update: I recently realized that this solution is wrong. The following test cases does not pass
transactions = [
[6, 0, 50],
[1, 6, 40],
[2, 6, 10],
[6, 3, 40],
[6, 4, 10],
[5, 6, 25]
]

What is wrong here? Given a bizarre node as the starting point of BFS, I want to find the minimum containing the source. However, this may not lead to globally optimal solution.

Read More

5 Reply

adamujk ★ 948 Last Edit: November 14, 2018 5:26 AM

Very nice solution, here's **Clean Code and Some Explainations**

```
def minTransfers(self, trans):
    debt = defaultdict(int)
    for i, j, k in trans: debt[i] -= k; debt[j] += k
    debt = list(filter(None, debt.values()))

    def SplitDebt():
        cnt, Q = set(), deque([([0], debt[0])])
        while Q:
            ...
```

Read More

4 Show 1 reply Reply

liangsun ★ -60 February 19, 2019 8:51 AM

This is incorrect. Try this input:

`[[1,0,18],[2,1,9],[4,3,11],[5,4,10],[5,6,7],[7,6,5],[8,7,3]]`

The expected answer is 6 while this code returns 7.

6 Show 9 replies Reply

Hammer001 ★ 200 April 23, 2019 10:37 AM

This is great solution! Thanks for sharing!

0 Reply

leetcodep123 ★ 0 March 5, 2019 5:17 AM

Excellent solution. Thank you for sharing it.

0 Reply

edaengineer ★ 258 March 5, 2020 10:51 AM

@chao4 How do we know that the 1st minimum clique found using BFS is the right one for the optimal solution to the problem?

That is, [4, -2, -2] or some other clique [4, -1, -3] are both minimum cliques for balancing the debt of [4]. But it's possible that choosing [4, -1, -3] is better for finding other minimum cliques for the rest of the people in the problem. What's the proof that choosing any minimum clique would produce the optimal solution?

0 Reply

Jinjt ★ 1 December 23, 2019 6:36 AM

perfect solution!
could you explian why "The final number of transactions will be 5 - 2, which is num_non_zero - num_clique." ?
thanks