Documents/leetcode-master/myLeetCode/C++/hw3problem4.py

```python
from collections import defaultdict
from queue import Queue

def can_win_games(n, v):
    if sum(v) != (n-1)*n // 2:  # 检查比赛场数是否正确
        return False

    # 构建网络流图
    source = 0
    sink = 2*n+1
    capacity = defaultdict(int)
    for i in range(1, n+1):
        # 每个玩家对应源节点和汇节点
        capacity[source, i] = v[i-1]  # 设置源节点到每个玩家的限制流量
        capacity[i, sink] = n-1  # 每个玩家到汇节点的容量为n-1
        for j in range(i+1, n+1):
            # 每场比赛对应一条有向边，容量为1
            capacity[i, j] = 1

    # 运行最大流算法
    flow = 0
    while True:
        visited = set()
        q = Queue()
        q.put((source, float('inf')))
        while not q.empty():
            node, curr_flow = q.get()
            if node in visited:
                continue
            visited.add(node)
            if node == sink:
                flow += curr_flow
                break
            for neighbor in range(1, 2*n+2):
                if (node, neighbor) in capacity and neighbor not in visited:
                    q.put((neighbor, min(curr_flow, capacity[node, neighbor])))

        if flow == sum(v):
            return True
        elif flow > sum(v):
            return False
        else:
            # 重置visited和q，继续运行最大流算法
            visited = set()
            q = Queue()
            q.put((source, float('inf')))


n = 4
v = [1, 1, 2, 2]
print(can_win_games(n, v))  # 输出True，因为存在一种方案使得每个节点的出度等于其对应的v值

v = [3, 1, 0, 2]
print(can_win_games(n, v))  # 输出True，因为存在一种方案使得每个节点的出度等于其对应的v值

v = [1, 2, 1, 2]
```

```
print(can_win_games(n, v))  # 输出True，因为存在一种方案使得每个节点的出度等于其对应的v值

v = [2, 1, 1, 0]
print(can_win_games(n, v))  # 输出False，因为不存在一种方案使得每个节点的出度等于其对应的v值

# We can use a network flow algorithm to solve this problem, where each player
corresponds
# to a source node and a sink node, and each game corresponds to a directed edge with
a
# capacity of 1. We also need to set a traffic limit for each player, which is the
# number of games that that player needs to win. Then we run the maximum flow
# algorithm and check whether there is a scheme that makes the egress of each
# node equal to its corresponding v value.
```