We propose a greedy algorithm here. The general idea is, we define a value function of a node, and then we greedily fill the bus one by one. We should also run something else to make sure no bus is empty is first and then run the greedy.

The greedy algorithm consider a value function that considers a node's value to be in an unfilled bus:

#G is the graph, u is a vertex that we want to evaluate its value, and b is the unfilled bus that we hope to find the best u to put into b

```
def f(G, u, b):
    V = set of all vertices in G
    y = total edges from u to set V\b
    x = total edges from u to b
    return x - y
```

The intuition of f is that it evaluates the "value" of putting a node u into a bus we are trying to fill.

x represents the gain of friendship if we put u into this bus best.

y represents the lose of potential friendship, because use it in this bus prevents us use it in another bus. (That is his friendship with those who are not in b)

Globally, we are trying to paritition the original graph into b1, b2... where crossing edges between each sets are minimized.

By putting u in b, we minus the x corssing edges but gain y crossing edges from b to reset of the graph. Thus greedily we want to maximized the value x-y we gained from u

Thus, at each step, iterate through all unseated passangers and sort them according to value function relative to b

We want the node with value as big as possible, break tie with priority with larger x, our immediate gain

However, if a node form a rowdy group, we just skip it in consideration. If every node gives rowdy group in current bus, just got to fill next bus

repeat untill we fill a bus and then we consider next bus

untill all passangers are assigned seat

```
#bus is the set of all k buses, n is number of passangers, c is capacity
bus = [[] [] [] ... []]
put k-n//c people with lowest degree into last k-n//c buses (to make sure no bus is empty as last)
for b in bus:
    while b is not full:
        sort all nodes u in G\b according to f(G, u, b), break tie with total edges from u to b
        v = the one with highest rank that does not give rowdy group in b1
        if v = none:
            break
        append v to b
    remove b from G
return bus
```