Homework X: Q2

Name: Xinkai Lin, xinkaili

*Don't forget to input your list of collaborators and sources on **AutoLab**.*
Please submit this file as a PDF.

1 Formalizing the Problem

i. The graph is formatted as an adjacency list storing p node, where p is the group of ADFs, and their friends as neighbor node in the list. For each node, b, we can access all of b's friends.

ii. Each ADF will sit at a minimum distance away from the start node, where start node is the one sitting at the first row.

iii. The start node will be a ADF who is assigned in the first row, and for any node will have a distance of 1 of their neighbor node and distance of 2 of their neighbor's neighbor node and so on. Then, they will sit at that distance row away from the start node. In some case, there might be some ADF who have no friendship with the previous group, and then I will be assigned one of them in the first row, and compute the distance of his friend and his friend's friend.
So, for every pair of friends, they can talk to each other during the movies.

2 Algorithm Idea

Similar to question 1, I will use BFS to solve this problem. I'll formatted as an adjacency list. There are p nodes, and each of them will save all of his friends who is pair with him in a list. Also, there is a start node which is a ADF who is assigned in the first row. Since in the beginning we don't know their distance yet and also there might be a case that some ADF will have no friendship with the other. So, I'll initialized the distance as -1. Then, I'll pick one ADF as the start node, and set the distance as 0 to himself. Now, I'll run the BFS algorithm to find out the distance of all other nodes from the start node. That is same as to find out the distance between the ADF who is sitting in the first row with his friend and his friend's friend. If b_1 and b_2 has friendship, then their distance is 1. If b_2 is a friend of a friend, then they have a distance of 2 and so on. which it is satisfies the distance compatible property. Next, the for loop will loop through all the nodes to see if there are still exit distance of -1, if there is, then it means there will be some ADFs who have no friendship with the previous group of ADFs. Then, I'll pick one of them to be the start node, and use the BFS algorithm again to find out all the distance of his friends and his friend's friend and so on, and repeat the process until I find out all the distances and assigned them to sit at that distance row away from their start node. At the end, for every pair of friends, they can talk to each other during the movies.

3 Algorithm Details

Algorithm: Algorithm1

```
Allocate array distance length of p
Initialized for all p and its distance is -1

//loop through all the nodes
for (i=0; i<p; i++)
        if (distance[i] = -1)        //if this node has not visited
                set q = {i}         //set i to be the startnode and add to the LinkedList of queue
                distance[i] = 0;           //set the distance of i as 0
                //BFS(s), run the BFS algorithm
                While q is not empty
                        currentADF = q.dequeue;        //remove the front node and assign to
                                                       //currentADF
                        dist = distance[currentADF] +1; //the distance of the friends of
                                                        //currentADF is its self's distance plus 1
                        for every friendship (p, f)       //check for every friends of current ADF
                        if (distance[f] = -1)              //if the friend of current ADF's is not visited
                                distance[f] = dist;     //set the distance of its friend as dist
                                q.add(f);               //adding currentADF's friend to the LinkedList

return distance  //when the distance is 0, means he will sit in the first row and the other
                 //ADF will sit at distance row away from him
```

4 Proof of Correctness Idea

There could be different ways of assigning the admissible seat based on the ADF you choose to be sit in the first row. In this algorithm, I will show one of the possible way of assigning the admissible seat by using the BFS algorithm. For any giving p, the group of ADFs, and f the pairs of friends. For any ADF to be the start node and assigned to sit in the first row, I'll use the BFS algorithm to find out the distance between him with his friend and the distance between his friend's friend. Since, there might be a case that for some ADF might not have any friendship with the other. I'll loop through all the nodes to check if that actually happened. If it does, I'll have them to use the BFS algorithm again, setting a start node as sitting in the first row and to find out the distance between his friend and their friend's friend and so on. At the end of this loop, every ADFs should have a distance from the start node and will sit at that distance row away from the start node. Thus, the seating is admissible, and everyone could talk to their friends during the movies.

5 Proof Details

First, I'll created an array of distance with length of p to save the distance between each ADF with their friendship, where p is the number of ADFs. Then I will initialize all the distance to be -1 because at this time we don't know their distance yet regarding to their friends. Now, we loop through all the ADFs. If the current ADF is equal to -1, meaning its either not visited yet or it has no friendship with any of the previous ADF group. Then, I will run the BFS algorithm to find out all the distance between his friend and the distance between his friend's friend. For the BFS algorithm, first I have created a LinkedList of queue to save the node from L_0 to L_{i+1}. The L_0 will always be the start node. And so, I add the start node to the queue and set its distance as 0 to itself. And for every friend of him will have the distance of current ADF which is 0 and plus 1. Then, add all his friends to queue and repeat the same process to find out the distance between the start node with its friend's friend, until the list is empty. This will only give us one set of the admissible seat. As the for loop continues, I might find out there are some ADFs that have no friendship with anyone else. So, I'll use him as the start node and compute his distance with his friend and the distance between his friend's friend by using the BFS algorithm just like what I did before. When the for-loop end, I would be able to find out one or more admissible seating based on the friendship between the ADFs.

6 Runtime Analysis

The algorithm has one for loop for initialized for all distance as -1, each iteration will take one assignment, which will run at O(n) time. Then, there is another for loop to check for the distance. In this loop, since we loop from 0 to n-1, also when the condition met, it will run the BFS algorithm, which will take up to O(n(m+n)) time. Thus, overall, we have O(n) + O(n(m+n)), which is O(n(m+n)).