Part 1- Theory

Problem 1:

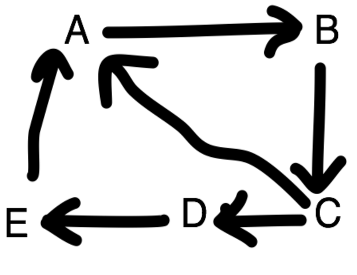
(a) It will be a dynamic programming problem, they way to do this is find out the minimum path from actor to Bacon and it is an unweight path graph problem. Firstly, we need find out the bacon number for people who was in same movie with this actor. The way to find out the bacon number for people is kind of dynamic programming, just keep call it when it reaches the people who have movie with Bacon and gets return with value 1. The loop keeps go back, and return value will always be return value plus one. So the actor's Bacon number will be the minimum bacon number for people who was in same movie with this actor and then plus one. It is more likely build a graph, for people who has the same movie with the bacon we add edge between bacon and those people. And weight will be 1. For people who are in same movie we always add the edge between them. It is just based on the breadth first search algorithm.

(b) Firstly, find the people who do not have movie with the Bacon call those people as group A. Firstly set K=1. For group A, find the people who do not have movie with the peoples have Bacon Number K. And then K++. And call those people as group B, find the people who do not have movie with the peoples have Bacon Number K. Then k++; Keep this loop until there are only one number left or until the left people have same bacon number. Those left people's will be the actor have the highest Bacon number. This problem is kind similar with the longest increasing subsequence. Or much more easily, base on A, find all the bacon number and sort it, and return the highest one.

(c) For example, if we have two arbitrary actors which is A and B. The way to find out the minimum number of links between A and B will be firstly check is A and B have same movie if true return 1; Else check if there are any actor C is having same movie with A and B, if yes return 1+1; Else check if there are any actor D is having same movie with C and C have same movie with A and B, if yes return 1+1+1 else keep go though this loop until it has return value. The value it gets return will be the minimum number of links between two arbitrary nodes. It is kind same with problem a – graph problem. If people are in same movie, we just add edge between them. And then find the minimum path between A and B. Using the algorithm like breadth first search.

Problem 2

(a) The reason why it didn't work, because it is possible to build an infinite loop.
For example: We try to find the shortest weighted path from A to E.



When at the node C, it is possible for node C go to the Node A, and it will add a no-need loop. It will build an infinite loop. Because this recursion will only pass through one single vertex. So this loop will go forever.

(b) Because we assume it is directed graph, and it is acyclic graph so there is no loop, so it will never have a chance to point back. If it is never point back. It will never build a no need loop. And this algorithm is a deepest first algorithm. It will find the t first, and base on the value calculate back.

(c) O (|V|)