PA4 Extension Report

Jiawei Li PID: A53226117
Ziyan Zhu PID:A53219779

September 6, 2017

In this part of the assignment, given a list of actors, the program will return a list of movies with the shortest length, which are able to connect these actors. For example, calling the program as follows:

```
./extension movie_casts.tsv
```

then inputing actor names "CATLETT, LOYD", "RAVEN, ELSA", "PEREZ, ROSIE" and "TURTURRO, JOHN", the program should return movie "FEARLESS" which was made in 1993.

To solve this problem, we will exploit the idea of greedy algorithm and manipulate the union and intersection operation of set. We add two new elements to the GNode to record the movie he/she participate in. One is movie set: Represent the total movie an actor(component) hold. movie key: The set of movie that can connect different component as the current component.(Note: there could be multiple key and key set in one huge component, for example in a huge component A, there could contain actor ABCDE, there could be three movie key set 1,2,3 to connect AB, BC, CDE, and in each movie key set there could be multiple movies.) As disjoint set, we only use one node(the first actor we get from the vector) to represent the whole component.

We first put every user input to a vector, and conyinue the following step.

1. We get the first actor out and compare the movie he/she has to the second actor B, if there is no common that means there are no edge between these two nodes, therfore we move on to the next actor C.

2. If A and C have common movie, we then merge them as a huge component, with movie set represent all the movies in the component, and movie key represent the set of common movies we need to connect each subcomponent. We then compare the new movie key which is the set of movie that needed to connect A and C to the original movie key in component A(here we asssume A is not an actor but a huge component that has already been a group of actor).

3. If there are even smaller movie key when insersection the new movie key with the set inside the component A, that means the smaller movie key can user to represent the component, we then replace the original movie key with the intersection key. If there are not, we simply input this new key to the component.

4. Therefore we create a even larger component, then we erase the actor(or component) we compared to from the vector, since it has already merged to a bigger component, then we insert the bigger component to the vector and compare it to the rest node(or component)in the vector and continue the step 1,2,3,4.

5. When we reach the last node in the vector, we are sure to have every node that could possibily connect to A through the actor in the vector, the rest actor(or component) in the vector is surly have no connection with A, since every time we check whether an actor have connection with the component, we compare it to the movie set of the component, and the movie set of A is just a sub set of the component movie set, if there are no common movie with the actor and component, there are sure no common movie between this actor and A.

6. We then move this big component out to a vector result, and continue the step 1,2,3,4,5,6 until the size of the vector of input is zero.

7. At last we have several big component in the result vector, we then choose the first movie in each set of movie key of each component, since we only need the minimum number of movie, if a movie key has multiple movies, that means those movie can all be the movie to connect the actors we merge as a component, then we only need to output one of them.

Algorithm 1 Extension algorithm

```
1: procedure MOVIE COM
2:     actorset ← input actor node
3:     start ← first element of actorset
4:     while start is the last element of actorset do
5:         com ← the next element of start
6:         while com is the last element of actorset do
7:             v ← the intersection set of movie of start and com
8:             if size of v is zero then
9:                 com + +
10:            else
11:                movie set of start ← the union set of movie set of start and com
12:                for every movie set in the movie key set of start do
13:                    if intersection between movie key set and v is 0 then
14:                        continue
15:                    else
16:                        the set of start movie key ← the intersection set of movie key and v
17:                erase the node at position com in actorset, the size of actorset is reducing 1
18:        start + +
19:    for each remaining nodes in the actorset do
20:        output the first movie in each movie set of movie key
```

To test out algorithm, some actors from some certain movies will be chosen as input and we run the program. Then we manually checked the correctness of the output.