Instruksjon til å kjøre programmet: Kjør python3 innleveringsoppgave3.py så få du alt

I global scope er det tre variabel:

allMovie, allActors og G(E,w)

Example av variablene:

#allMovies {'tt8093700':['Ant-Man', 8.3], 'tt8093701': ['Iron Man', 5.7]}

#allActors {'nm0886638': ['Tome', ['tt0051407','tt0051407'], 'nm0886638': ['Mamie', ['tt0063790', 'tt0062514']]}

#E {a1:{a2,a3,a6}, a2:{a1,a4,a6},...}

#w {(a1,a2): m1, (a1,a3): m2}

Oppgave1 - Bygg grafen (kjøretid: 14s):

Structuren til grafen er representert av en dictionary med tuple av actors som nøkkel og movie som verdi.

For eksempel:

{

(a1,a2): m1,

(a1,a4): m2,

(a2,a5): m4

(a2,a7): m4,

(a7,a5): m4,

}

der a representere en node, evs actorId

og m representere en kant, dvs movieId

Oppgave2 - Six Degrees of IMDB (kjøretid: 0.1s):

Her bruker jeg bredde-først søk for å finne det korteste stien fra en node til en annen.

Grafen blir kjekket lagvis i denne algoritme.

Oppgave3 - Chilleste vei(kjøretid: 11s):

Her bruker jeg dijkstra algoritme for å finne det chillest stien fra en node til en annen.

Den som er litt forvirrende, er at path som blir retunert fra dijkstra(G, start,end), er ikke

det siste path vi trenger. Fordi en node kan ha flere naboer som har en kant med samme weight.

Derfor i finnChillestSti() opprett jeg en list for å holde styr på den resultaten/stien vi trenger.

Oppgave4 - Komponenter(kjøretid: 6s):

Her bruker jeg dybde-først søke først for å finne en komponent.

For hver node som ikke er besøkt før, bruker jeg dfsVisit() på den.

På denne måten finner jeg alle komponenter i grafen.

I DFSvisit prøvde jeg med rekursjon, men det viser 'maximum recursion depth exceeded while calling a Python object'¨

fordi python støtter ikke så stor mengde av rekursjon.