UNIVERSITÀ
DEGLI STUDI
FIRENZE

Advanced algorithms and Graph mining

Caso di studio: Internet Movies DataBase

Studentessa: **Fatemah Alhamdoosh**

Relatori: **Prof. Andria Marino and Prof. Massimo Nocentini**

2021/2022

Construct Graph: execution time (2 minutes)

```
class IMDBGraph:
    def __init__(self,`path="data.tsv") -> None:
    def construct_graph(self)-> nx.Graph:
```

1- Read rows from data file to data frame

```
            df=pd.read_csv(self.path,delimiter="\t", names=['actor','film'])
```

2- Assign id to each actor and to each movie

```
            actorsDict=dict(map(reversed,enumerate(df.actor.unique())))
            movieDict=dict(map(reversed,enumerate(df.film.unique(),start=len(actorsDict))))
```

3-Extract year from list of unique movie titles, create dict {movie title: year}

```
            yearDict= dict(map(self.get_year,df.film.unique()))
```

4-Add id_actor,id_film,year columns in df

```
            df['id_actor']=df['actor'].map(actorsDict),df['id_film']=df['film'].map(movieDict)
            df['year']=df['film'].map(yearDict)
```

5-Build the graph from df using id_actor, id_film columns

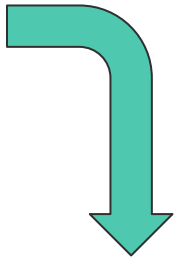
```
            self.g=nx.from_pandas_edgelist(df,"id_actor","id_film")
```

6-Build reversed dictionaries for nodes attributes setting :

```
            self.reversedAcrtorsDict=dict(map(lambda x:(x[1],{'label':x[0],'type':0}),actorsDict.items()))
            …..
            nx.set_node_attributes(g,self.reversedAcrtorsDict)
            nx.set_node_attributes(g,self.reversedMovieDict)
```

Construct Graph using pandas

	actor	film
0	$, Homo	Nykytaiteen museo (1986)
1	$, Homo	Suuri illusioni (1985)
2	$, Steve	E.R. Sluts (2003) (V)
3	$hort, Too	American Pimp (1999)
4	$hort, Too	Beats, Rhymes & Life: The Travels of a Tribe C...
5	$hort, Too	Gangsta Rap: The Glockumentary (2007)
6	$hort, Too	Get It Where You Fit in 1 (2003) (V)
7	$hort, Too	Ghetto Physics (2010)
8	$hort, Too	Ghostride the Whip (2008) (V)
9	$hort, Too	Hip Hop Uncensored Vol. 4: Miami Vice (2002) (V)



	actor	film	id_actor	id_film	year
0	$, Homo	Nykytaiteen museo (1986)	0	2364796	1986
1	$, Homo	Suuri illusioni (1985)	0	2364797	1985
2	$, Steve	E.R. Sluts (2003) (V)	1	2364798	2003
3	$hort, Too	American Pimp (1999)	2	2364799	1999
4	$hort, Too	Beats, Rhymes & Life: The Travels of a Tribe C...	2	2364800	2011
5	$hort, Too	Gangsta Rap: The Glockumentary (2007)	2	2364801	2007
6	$hort, Too	Get It Where You Fit in 1 (2003) (V)	2	2364802	2003
7	$hort, Too	Ghetto Physics (2010)	2	2364803	2010
8	$hort, Too	Ghostride the Whip (2008) (V)	2	2364804	2008
9	$hort, Too	Hip Hop Uncensored Vol. 4: Miami Vice (2002) (V)	2	2364805	2002

Question 1.E : (Execution time for year : 30 seconds)

Which one is the actor who worked for the longest period considering only the movies up to year x?

```
def find_actor_with_longestPeriod(self,x: int)-> tuple:
    longestPeriod=0, longestActorId=0
```

1-Iterate over actors nodes defined by type 0

```
    for j,d in self.g.nodes(data=True):
        if d['type']==0:
```

2-Get the movie list of the actor j, and create set of unique years

```
            movies=list(self.g.neighbors(j))
            years=set()
            if(len(movies)!=0):
```

3-Iterate movies,add years of movie inferior of the given year x

```
                for m in movies:
                    if(self.g.nodes[m]['year']<=x):
                        years.add(self.g.nodes[m]['year'])
```

4- Calculate the period of work of actor j

```
            period=(lambda x,y: 0 if x==0 else
                                ( 1 if x==1 else
                                max (y)-min(y)))(len(years),years)
```

5-Check the max score after each actor score

```
            if(period>longestPeriod):
                longestPeriod=period,  longestActorId=j
```

Year	Actor	score
1930	Corbett, James J.	36
1940	D?az, Porfirio (I)	44
1950	Cody, William F.	56
1960	King Haakon VII	63
1970	Kaiser Wilhelm II	74
1980	Emperor Franz Josef	84
1990	King Edward VII	96
2000	King Edward VII	108
2010	Cody, William F	114
2020	King Edward VII	120

Question 2.1

Considering the movies up to year x, restricting to the largest CC of the graph. Compute exactly the diameter of G.

```
def get_largest_cc(self,year: int):
```

1- Create “h ” a subgraph filtering movie nodes greatest than year

2- Find the largest cc using max(nx.connected_components(h))

3-Find the node of highest degree in the largest CC

```
def get_hiestDegree_node(self,graph:nx.Graph) -> tuple:
    return max(graph.degree,key=lambda x: x[1])
```

4-Do a BFS in the LGG starting from the highest degree node

```
def breadth_first_search(self,cc:nx.Graph,start_node:int)->dict:
    visited = {}, queue = Queue(), queue.put(start_node),visited[start_node]=0
    while not queue.empty():
        current_node = queue.get()
        for next_node in cc.neighbors(current_node):
            if next_node not in visited:
                queue.put(next_node)
                visited[next_node]=visited[current_node]+1
    B_u = defaultdict(list)
    for key, value in visited.items():
        B_u[value].append(key)
    return B_u
```

5- Return a dictionary B_u

ex: year:1960,
u is 1023057 with
degree 924.
Doing BFS we
obtain a dict:

B_u = {24:[1,2],
23:[65,5],
....,
2:[8],
1:[99,55,66] }

Question 2.1

```
def calc_diameter(self,cc:nx.Graph,B_u:dict)->int:
```

1- Find ecc of u, assign lower bound and upper bound

2- Iterate and compare values.

```
    ecc=max(B_u), i=ecc,lb=ecc, up=2*ecc
    while up >lb:
        eccs=nx.eccentricity(cc, v=B_u[i])
        B_i=max(eccs.values())
        max_val=max(B_i,lb)
        if max_val > 2*(i-1):
            return max_val
        else:
            lb=max_val
            up=2*(i-1)
        i-=1
   return lb
```

Year	start node, degree	ecc	inspected levels	Diameter	time
1930	(1023057, 925)	24	7	**34**	5min
1940	(1023057, 925)	22	5	**34**	2.5min
1950	(1023057, 925)	26	8	**36**	3min
1960	(2368131, 1298)	18	1	**35**	2min
1970	(2368131, 1298)	15	2	**27**	3min
1980	(2368131, 1298)	15	3	**25**	5min
1990	(2368131, 1298)	15	2	**27**	4min
2000	(193178, 1581)	18	4	**28**	8min
2010	(193178, 1786)	17	3	**28**	15min
2020	(193178, 1882)	18	2	**32**	20min

Question3.IV

Who is the actor who participated in movies with largest number of actors?

```
def find_actor_with_largest_staff (self)-> tuple:
    largestscore=0
    largestActorId=0
```

1- Iterate over actors

```
    for j,d in self.g.nodes(data=True):
        if d['type']==0:
```

2-movie is the list of all movie actor j he participated in

```
            movies=list(self.g.neighbors(j))
```

3- For each movie find the number of actors who participated in and sum it up to to score

```
            score=sum(map(lambda x: len(list(self.g.neighbors(x))) ,movies))
            if(score>largestscore):
                largestscore=score
                largestActorId=j
    return (largestActorId,largestscore)
```

Name of the actor who participated with largest num of actors	Score (number of actors he collaborated with them)	Time
Flowers, Bess	**41245**	2min

Question 4

1- Build the actor graph, whose nodes are actors and two actors are connected if they did a movie together.

```
def create_actor_graph_from_dict(self,saveName="actors_graph.gpickle"):
        movie_list=self.reversedMovieDict.keys()
        self.A=nx.Graph()
```

1- Iterate over movie nodes

```
        for j in movie_list:
```

2-Take the list of actors who participate in the movie j

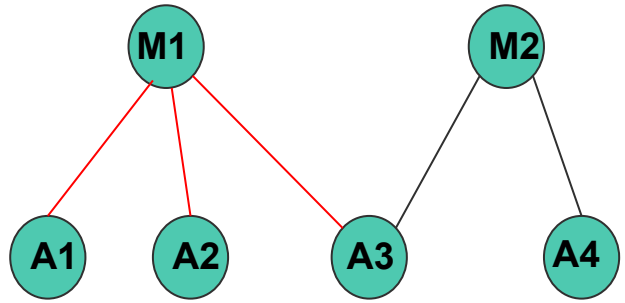
```
            actors=list(self.g.neighbors(j))
```

3- Iterate over all possible combination of actors

```
            for i in combinations(actors,2):
```

4- If edge exists in the graph then increase the weight by 1, otherwise add the edge with weight equal to 1

```
                if self.A.has_edge(*i):
                    self.A[i[0]][i[1]]['weight']+=1
                else:
                    self.A.add_edge(i[0],i[1],weight=1)
```



M1,M2 are movies nodes, Ai are actors nodes. The algorithm inspects the children of movie node . A movie will not be visited twice, while actors nodes does, it depending on the number of edges.

- Cost of the algorithm is O(E)
- Need 16 RAM of memory, on pc with 8 RAM run out of memory and fails.
- Execution time: 15min - 20min

Question 4

2-Which is the pair of actors who collaborated the most among themselves?

```
def find_most_actors(self):
        return max(self.A.edges(data=True),key=lambda x:x[2]['wieght'])
```

Pair of actors who collaborated the most among themselves	Number of collaborations	Time
North, Peter (I) and Byron, Tom (I)	**420**	18min

UNIVERSITÀ DEGLI STUDI FIRENZE

Thank you for your attention!