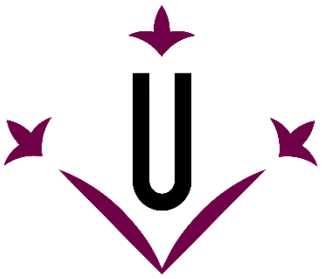
Pràctica 2: Team creator

Algorítmica i complexitat



Universitat de Lleida

Segura Paz, Aleix

Serrano Ortega, Aniol

Índex

Introducció

La present pràctica tracta d'implementar en el llenguatge de programació *Python* un programa que a partir d'una llista d'usuaris i els seus seguidors (*follows.txt*), que aquesta és, a la vegada, generada per un altre arxiu *Python* que se'ns proporciona (*follows_generator.py*), generi equips que sorgeixen a partir de les relacions que es troben en els seguidors de cadascun dels usuaris. Per a formar els equips, *Kosmos*, l'empresa que ens encarrega el *software*, ens estableix 3 normes que es detallen a continuació:

1. Els membres dels equips han d'estar vinculats. Dos jugadors estan vinculats si un segueix a l'altre o si es segueixen mútuament.
2. Han de cursar la mateixa titulació.
3. Un equip serà vàlid si te almenys 5 jugadors.

En els següents apartats, s'explica com s'ha implementat el programa i la seva funcionalitat.

Implementació del codi

Disseny i implementació

El disseny i implementació és pot dividir en dues fases: la fase de lectura i generació dels equips i la fase de representació, amb grafs, dels equips. Expliquem cada una d'elles:

Fase de lectura i generació

Primer de tot importem les llibreries a utilitzar: *matplotlib.pyplot* per a la representació i dibuix, *networkX* per als grafs *i random* per a generar colors aleatoris per a la representació dels grafs.

És defineix una variable global *teams_graph* que inicialitzem a un objecte de tipus *Graph*. Aquesta variable ens servirà per anar creant el graf i les seves diferents components conectades. En aquest punt ja podem començar a implementar la funció *teams_creator*. El que volem al començament es llegir el arxiu *follows.txt* que conté els usuaris d'Instagram generats i els seus seguits així que obrim l'arxiu i el tractem línia a línia. Cada línia té l'estructura:

Usuari: seguit1, seguit2, seguit3, ..., seguitN

On Usuari i $seguit_i$ són de l'estructura:

@GRAUnom_datanaixement, on:

- GRAU: és el grau del usuari i està compost per tres caràcters que identifiquen el grau en qüestió. Exemple: GEI per a Grau en Enginyeria Informàtica.
- nom: és el nom de l'usuari
- datanaixement: la data de naixement del usuari del tipus 1532005.

Per tant, tractem cada línia que hi ha en l'arxiu *follows* que hem obert. Per cada línia fem:

1. Obtenim l'usuari actual amb: *user = line[:line.find(':')]*.
2. Obtenim els seguits de l'usuari actual i els fiquem en una llista amb: *followeds = line[line.find(' ') + 1:].split()*

Un cop obtingut l'usuari i els seguits passem *user* i *followeds* a la funció *make_edges* per tal d'anar construint el graf. Dins de *make_edges*:

1. Afegim l'usuari com a un node del graf si aquest encara no hi era.
2. Obtenim el grau de l'usuari amb: *degree = user[1:4]*
3. Recorrem cada seguit en la llista *followeds*. Si aquell seguit no és encara un node del graf, l'afegim. Obtenim el grau del seguit i aquí es troba la clau de la formació dels equips: si el grau del usuari és el mateix que el grau del seguit actual i encara no hi ha una relació (aresta) entre ells, creem aquesta aresta.

Un cop hem recorregut cada línia de l'arxiu *follows.txt* podem tancar l'arxiu.

Seguidament podem obtenir els equips vàlids, on cada element de la llista serà un conjunt de nodes. Obtenim els equips vàlids amb:

teams = list(team for team in nx.connected_components(teams_graph) if len(team) >= 5)

Mirem les components del graf que hem anat generant i ens quedem amb aquelles que tenen almenys 5 nodes connectats.

Fase de representació

En aquesta fase mostrem els equips per pantalla i els grafs. Per a mostrar els equips per pantalla tenim la funció *show_teams* on li passem els equips vàlids obtinguts prèviament. En *show_teams* simplement:

1. Establim una variable *id* inicialitzada a 1 com a identificador d'equip.
2. Per a cada equip en els equips vàlids mostrem per pantalla el seu identificador i els jugadors que el conformen.

Per altra banda, per a mostrar els grafs generats tenim la funció *draw* on també passem els equips vàlids *teams*. En *draw:*

1. Inicialitzem un diccionari buit, on guardarem parelles node-color. La intenció és que tots els nodes d'una component siguin del mateix color.
2. Recorrem cada equip. Escollim un color aleatori per a aquell equip i assignem a cada node del equip el color generat.
3. Establim els atributs dels nodes del graf, en aquest cas el color, amb els generats i assignats al diccionari *color_dict*.

4. Creem una variable amb tots els colors del graf, on tenim els generats aleatòriament per als equips vàlids i el color negre per a aquelles components que no arriben a 5 jugadors.
5. Canviem la disposició del graf per a millorar la visualització. Concretament s'utilitza la representació *spring_layout*.
6. Finalment dibuixem el graf generat amb totes les seves components i tenim els equips vàlids amb colors diferents a negre i aquelles components que no poden formar equip amb color negre.

Pseudo-codi i cost empíric

Pel que fa al pseudo-codi i al cost empíric s'analitza fins a l'acabament de la fase de lectura i generació, és a dir, obviant les funcions *show_teams* i *draw* que només serveixen per a mostrar els equips per pantalla i graf i no formen part de l'algorisme de generació d'equips.

Tenim els següents pseudo-codis per a les funcions *teams_creator* i *make_edges:*

```
def teams_creator():
    obre follows.txt com usuaris_seguidors:
        per linia en usuaris_seguidors:
            usuari = linia[inici : primer ':']
            seguits = linia[primer espai en blanc + 1 : final]
            make_edges(usuari, seguits)
    usuaris_seguidors.tanca
    equips = llista(equip per equip en components_conectades(graf) si
                equip >= 5)
```

```
def make_edges(usuari, seguits):
	global graf
	si !graf_té_node(usuari):
		graf.afegir_node(usuari)
	grau = usuari[1 : 4]
	per seguit en seguits:
		si !graf_té_node(seguit):
			graf.afegir_node(seguit)
		grau_seguit = seguit[1 : 4]
		si grau == grau_seguit i !graf_té_aresta(usuari, seguit)
			graf.afegir_aresta(usuari, seguit)
```

I el següent cost empíric analitzat:

Funció	Codi	Cost - Cops executat
teams_creator	teams_graph = nx.Graph()	O(1)
	with open("../material/follows.txt") as users_follows	O(1)
	for line in users_follows	O(n)
	user = line[:line.find(':')]	O(1*n) = O(n)
	followeds = line[line.find(' ') + 1:].split()	O(1*n) = O(n)
make_edges	global teams_graph	O(1*n) = O(n)
	if not teams_graph.has_node(user)	O(1*n) = O(n)
	teams_graph.add_node(user)	O(1*varia segons construcció graf)
	degree = user[1:4]	O(1*n) = O(n)
	for followed in followeds	O(n*n) = O(n^2)
	if not teams_graph.has_node(followed)	O(n*n) = O(n^2)
	teams_graph.add_node(followed)	O(n*varia segons construcció graf)
	followed_degree = followed[1:4]	O(n*n) = O(n^2)
	if degree == followed_degree and not teams_graph.has_edge(user, followed)	O(n*n) = O(n^2)
	teams_graph.add_edge(user, followed)	O(n*varia segons construcció graf)
teams_creator	users_follows.close()	O(1)
	teams = list(team for team in nx.connected_components(teams_graph) if len(team) >= 5)	O(n)

En quant al cost que podem apreciar en la taula tenim un cost final generalitzat de $O(n^2)$ ja que la funció *teams_creator* està fent un bucle per recórrer cada línia de l'arxiu *follows* i per tractar cada línia s'invoca la funció *make_edges* que també fa un bucle per a recórrer els seguits d'un usuari. Per tant tenim un bucle que recorre 'verticalment' les línies de *follows.txt* i un bucle que recorre 'horitzontalment' els seguits de cada usuari.

Com a fet que cal explicar tenim les instruccions situades sota un bloc *if* on tenim el cost O(1*varia segons construcció graf) o O(n*varia segons construcció graf) on aquest 'varia segons construcció' depèn de la 'sort' i la forma aleatòria amb la que s'ha generat l'arxiu 'follows.txt' és a dir ens podem trobar en més o menys ocasions que tinguem que afegir un node o no depenent si l'hem afegit abans en un altre bloc *if* o no. Igualment això no acaba influint en el cost final que com s'ha comentat resulta de $O(n^2)$.

Conclusions

La present pràctica ha servit per a acabar d'assimilar els coneixements que es volen tractar en l'assignatura d'Algorítmica i Complexitat. Es torna a fer èmfasi en el cost dels algorismes que generem i a més a més es treballen aspectes que han sortit en altres assignatures com és la utilització de la llibreria *NetworkX* dedicada a grafs, que han estat estudiats en l'assignatura de Matemàtica Discreta.