Universitat de Barcelona

Grau en Enginyeria Informàtica

Tardor del Curs 2015-2016

Problemes d'Aparellament

Segona pràctica d'Algorísmica Avançada

- greedy -

1 Problemes d'Aparellament o Matching

A partir d'un conjunt d'individus i de coeficients que indiquen la seva propensió a ser aparellats, el problema de matching es planteja quines són les parelles òptimes per tal que tothom quedi content. No és difícil imaginar el plantejament del problema en un graf amb pesos. Llavors, la solució és un subconjunt d'arestes disjuntes, o sigui que no comparteixen cap vèrtex.

Qualsevol subconjunt d'arestes disjuntes es diu que és un matching d'un graf. I es diu també que es tracta d'un matching perfecte si recobreix tots els nodes, que per tant n'hi ha d'haver un número parell. Aquest problema és intensament utilitzat per euleritzar, és a dir, aconseguir que tots els nodes siguin parells, al calcular rutes en grafs connexes. Les solucions als problemes d'enrutament són cicles, i per tant, en elles, tots els nodes tenen grau parell.

Afortunadament, el problema de matching té una solució polinòmica des de 1965, introduïda per Jack Edmonds.[1] De fet, gràcies a això, el problema del carter xinès es resol polinòmicament. Ho fa plantejant un matching entre els nodes senars, que com va dir Leonhard Euler el 1736, a l'obra *Solutio problematis ad geometriam situs pertinentis*, en són un número parell.

2 Agència de Contactes

El problema de les agències de contactes consisteix en trobar la manera més bona d'aparellar persones entre totes aquelles que s'hagin inscrit a l'agència, que direm que són n, número parell. Per cada parella possible tenim un índex de rebuig, que és simètric. Es tracta de minimitzar el rebuig total entre les parelles formades.

Feu un algorisme voraç, `agencia_de_contactes(G)`, per calcular alguna solució al problema de l'agència de contactes, encara que no sigui una solució òptima. De fet, no es demana ni tan sols que el matching sigui perfecte.

L'entrada consisteix en un graf amb pesos. És a dir, les persones van de 1 a n i es corresponen amb els nodes del graf. Els pesos són els índexos de rebuig. Si entre dues persones no hi ha aresta significa que són incompatibles, que el rebuig és infinit. En la sortida cal escriure la llista de parelles sel·leccionades. Analitzeu-ne l'eficiència.

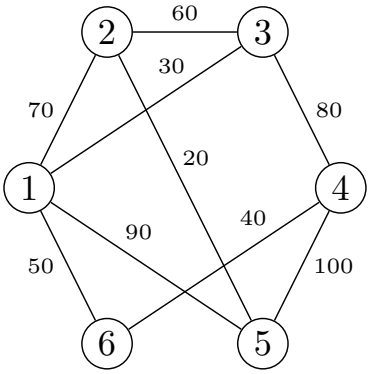
```
def llegir_graf():                                                    #Θ(V + E)
    nom = raw_input("Dona'm un nom pel graf:  ")                      #Θ(1)
    G = nx.read_edgelist(nom, nodetype=int, data=(('weight',float),)) #Θ(V + E)
    return G                                                          #Θ(1)
```

Rutina auxiliar.

[1] `http://cms.math.ca/openaccess/cjm/v17/cjm1965v17.0449-0467.pdf`

Entrades i sortides

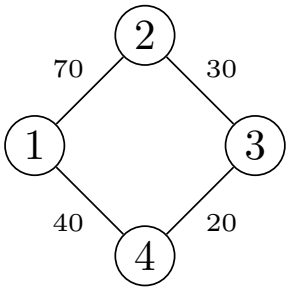
Seguidament es mostren dos exemples. A la primera columna hi ha el graf, a la segona el contingut del fitxer de text que el representa, i en la tercera, la sortida òptima que s'hauria de donar, si el programa fos capaç.




```
1 2 70.0
1 3 30.0
1 5 90.0
1 6 50.0
2 3 60.0
2 5 20.0
3 4 80.0
4 5 100.0
4 6 40.0
```

```
[[1,3], [2,5], [4,6]]
```

Figura 1: Graf, entrada i sortida del primer exemple




```
1 2 70.0
1 4 40.0
2 3 30.0
3 4 20.0
```

```
[[1,4], [2,3]]
```

Figura 2: Graf, entrada i sortida del segon exemple

3 Desenvolupament

Heu d'implementar la pràctica en Python, en un únic fitxer que contingui tot el conjunt de funcions que heu implementat. Al campus virtual teniu un arxiu `plantilla.py` que heu d'utilitzar com a base per la vostra implementació, seguint les següents normes:

- El vostre fitxer s'ha d'anomenar

  `greedy_nom_cognom1_cognom2_G.py` ,

  on "G" és la lletra del grup de pràctiques al que pertanyeu.

- Assegureu-vos de posar el vostre nom al principi de l'arxiu, on està indicat que ho feu.

- Configureu el vostre editor per utilitzar codificació "UTF-8". D'aquesta manera podreu passar la pràctica entre Windows, Mac i Linux sense problemes amb els caràcters especials (accents, apòstrofs, etc.).

- Configureu el vostre editor per utilitzar 4 espais per al sagnat del codi. Així treballarem tots amb la mateixa configuració i evitarem barrejar espais amb tabuladors a l'hora de codificar.

- L'arxiu de plantilla està preparat per executar la pràctica amb una entrada per defecte, però podeu executar la pràctica amb un altre arxiu d'entrada també:

```
python greedy_nom_cognom1_cognom2_G.py <arxiu>
```

La presentació de les pràctiques d'aquesta assignatura consta tant del codi que implementa el problema, com de l'anàlisi d'eficiència. L'anàlisi el podeu efectuar comentant cada línia amb la seva eficiència, i/o comentant com l'heu calculada al comentari de cada funció implementada.

Podeu utilitzar la llibreria `networkx`[2] per definir els vostres grafs. Tot i així, **no es permet utilitzar funcions d'alt nivell que implementin recorreguts complets**, ni estructures de dades sofisticades de les que no en coneixem la implementació ni, per tant, la seva eficiència.

4 Lliurament

El lliurament s'ha de fer via campus virtual abans de la data límit d'entrega, el diumenge 15 de novembre de 2015, a les 23:55. Heu d'enviar únicament l'arxiu de la vostra pràctica `greedy_nom_cognom1_cognom2_G.py`.

5 Criteris d'avaluació

- El programa dóna un resultat correcte per totes les entrades possibles: 50% de la nota
- Ús adequat del llenguatge i bon estil de programació: 25% de la nota
- Anàlisi d'eficiència: 25% de la nota

[2] `https://networkx.github.io/`