Algorísmica Avançada

Parcial II de Pràctiques

Grup A - divendres de 10 a 12. Prof. Carles Franquesa

16 de desembre de 2011

Grau en Enginyeria Informàtica

Universitat de Barcelona

Per fer aquest parcial no heu de crear cap fitxer nou. Tant el codi com les instàncies de prova que es mostren al final de l'enunciat estan penjades al campus. Descarregueu-ho, i renomeneu l'arxiu font en python *codi.py* al format *P2_A_nom_cognoms.py*. El parcial es composa de dos exercicis que un cop resolts caldrà penjar de nou al campus, en el mateix únic fitxer, en el qual ha de figurar-hi també el vostre nom en la primera línia.

En els dos exercicis, l'entrada de dades és un graf, que es llegirà d'un fitxer de text amb la rutina `llegir_graf()` que també es dóna.

1. *Antenes.* Es disposa d'un graf que representa un territori geogràfic, de manera que cada node correspon a una població. Les arestes ens indiquen tan sols que dos pobles són veïns, no representen distàncies. Per donar cobertura d'un servei qualsevol (telefonia, televisió, ràdio...) al territori complet, cal que cada població o bé tingui una antena repetidora, o si no, que sigui veïna d'alguna població que sí que en tingui una.

   Implementeu un algorisme greedy en una funció, `antenes()`, per cobrir el graf sencer utilitzant el mínim nombre d'antenes. Analitzeu-ne l'eficiència. La resposta de la funció donat un graf és un vector de booleans indicant les poblacions on calgui col.locar-les. Com veieu, el graf no apareix a la capçalera de la funció. Això és així perquè el declarem com a variable global. 40 punts.

2. *Recobriment.* Un recobriment per vèrtexos d'un graf $G(V, E)$ és un subconjunt $S \subseteq V$, tal que qualsevol vèrtex del graf o bé és a S, o bé és veí d'algun vèrtex d'S. En altres paraules, un recobriment és un subconjunt de vèrtexos $S \subseteq V$ tals que totes les arestes del graf tenen algun extrem dins S.

   Ompliu el codi que es dóna per fer un algorisme que calculi el recobriment de cardinalitat mínima d'un graf. Aquest cop cal retornar la llista de vèrtexos que conté S. A més, en aquest cas sí que heu de donar la resposta òptima.

   Tal com es pot observar en el codi donat utilitzem backtracking en un graf implícit. Es tracta d'implementar un arbre d'exploració binari. És a dir, d'un node, només pengen dos successors. El que resulta d'afegir un vèrtex de G a S, i el que resulta de no afegir-li. Per tant, en la rutina d'exploració no caldrà cap bucle. L'estructura del node del graf d'exploració ve donada.

   Aneu alerta, es tracta d'un backtracking modificat, ja que no tota solució parcial és factible. Per això, posposeu la comprovació de factibilitat a tan sols quan es tenen solucions totals, és a dir, no comproveu la factibilitat de solucions parcials. 60 punts.

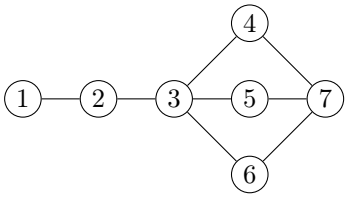
```
def llegir_graf():                                          #Θ(V + E)
    nom = raw_input("Dona'm un nom pel graf:  ")            #Θ(1)
    G = nx.read_adjlist(nom, nodetype = int)                #Θ(V + E)
    return G                                                #Θ(1)


# EXERCICI 1
def antenes():

            # AFEGIU CODI AQUI

    return x


# EXERCICI 2
class node:
    def __init__(self,n):
        self.i = 0
        self.x = [False]*(1+n)    # solucio en el node.

    def z(self):                  # retorna la cardinalitat d'S.
        q = 0
        for j in range(1,self.i+1):
            if self.x[j]:  q = q + 1
        return q

def cobert(i,c):                  # retorna si el vertex i queda
                                  # cobert amb la solucio c, o no.
            # AFEGIU CODI AQUI

    return False                  # (modifiqueu aquesta linia)

def factible(s):                  # retorna si el node d'exploracio s
                                  # representa una solucio factible, o no.
            # AFEGIU CODI AQUI

    return False                  # (modifiqueu aquesta linia)

def explora(s):                   # fa el backtracking a partir del node s.
    global z,x                    # valor de la millor solucio trobada.

            # AFEGIU CODI AQUI

    return                        # (elimineu aquesta linia)

def escriu():                     # escriu la solucio.
            # AFEGIU CODI AQUI

    return                        # (elimineu aquesta linia)

oo = 10000

def main():
    global G,n                    # el graf, i el nombre de vertexos.
    global z,x                    # valor de la millor solucio trobada.

    G = llegir_graf()
    n = G.number_of_nodes()
    z = oo
    x = [False]*n
    s = node(n)
    explora(s)
    escriu()
main()
```

codi.py

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 que hauria de donar l'exercici 2. Respecte l'exercici 1, la sortida depèn de l'astúcia del vostre codi.

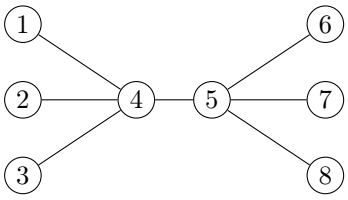
1.




```
1 2
2 3
3 4 5 6
7 4 5 6
```

```
[2,7]
```

2.




```
1 4
2 4
3 4
4 5
5 6 7 8
```

```
[4,5]
```