Algorísmica Avançada

Parcial II de Pràctiques

Grup C - dimecres de 8 a 10. Prof. Carles Franquesa

21 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_ C_ 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. *Mapamundi.* Els cartògrafs tenen el problema de pintar mapamundis polítics. Volen que cada país quedi d'un color diferent al dels seus països veïns. És ben clar que una manera de fer-ho és pintant cada país d'un color diferent. Però es tracta d'estalviar tinta, i cada color nou representa un cost addicional per imprimir els mapes.

   Donat un graf $G(V, E)$ en el que els vèrtexos de V representen països, i entre dos vèrtexos hi ha una aresta si els dos països corresponents són veïns, implementeu un algorisme greedy en una funció, `mapamundi()`, per pintar el graf sencer sense que vèrtexos veïns comparteixin color, utilitzant el mínim nombre de colors. Analitzeu-ne l'eficiència. La resposta de la funció donat un graf ha de ser un vector d'enters indicant el color assignat a cada país. Com veieu, el graf no apareix a la capçalera de la funció. Això és així perquè el declarem com a variable global. Per fer aquest exercici, considereu enters tant els colors, com els països. Una selecció voraç raonable és assignar el primer color diponible a cada node, i només crear un color nou quan no hi hagi més remei.(40 punts).

2. *Nombre cromàtic.* El nombre cromàtic d'un graf és el nombre mínim de colors necessaris per *pintar* un graf, tenint en compte que vèrtexos veïns han de tenir colors diferents. Encara que s'utilitzi el terme "pintar", és ben clar que el problema consisteix en assignar un valor natural a cada vèrtex, que representa el color.

   Resoleu òptimament el nombre cromàtic d'un graf. Per això us cal implementar un algorisme enumeratiu en el que el nivell de profunditat de l'exploració coincideix amb el nombre de vèrtexos amb color assignat. La sortida d'aquest segón exercici ha de ser com en el cas anterior el vector amb l'assignació dels colors als vèrtexos.

   Tal com es pot observar en el codi donat utilitzem backtracking en un graf implícit. Es tracta d'implementar un arbre d'exploració n-ari. És a dir, d'un node, pengen n successors corresponents a pintar un vèrtex de G amb un color entre 1 i n. Per tant, en el cas no trivial de la rutina d'exploració caldrà un bucle. En el cas trivial, només caldrà guardar el resultat a les variables globals x i z. L'estructura del node del graf d'exploració ve donada. (60 punts).

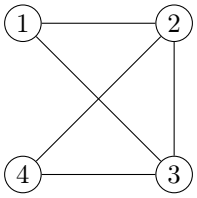
```
def llegir_graf():                                          #Θ(V + E)
    nom = raw_input("Dona'm un nom pel graf:  ")            #Θ(1)
    G = nx.read_adjlist(nom,nodetype = int)                 #Θ(V + E)
    return G                                                #Θ(1)

# EXERCICI 1:
def mapamundi():
            # AFEGIU CODI AQUI
    return x

# EXERCICI 2:
class node:
    def __init__(self):
        self.i = 0
        self.x = [0]*(1+n)          # per poder indexar els colors
                                    # a partir de l'1.
    def z(self):                    # retorna el nombre de colors
                                    # de la solucio self.x.
        q = 0
        for i in range(1,n+1):      # per cada color mirem
            nou = True              # si es nou a self.x...
            for j in range(1,i):
                if (self.x[i] == self.x[j]):
                    nou = False
                    break
            if nou:  q = q + 1      # ...i si ho es, el contem.
        return q

def vertex_factible(v,c):           # retorna si el color
                                    # assignat a v en el vector
                                    # c es diferent al dels
                                    # seus veins, o no.
            # AFEGIU CODI AQUI
    return True                     # (modifiqueu aquesta linia)

def factible(s):                    # retorna si la solucio repre-
                                    # sentada pel node d'exploracio
                                    # s es factible, o no.
            # AFEGIU CODI AQUI
    return True                     # (modifiqueu aquesta linia)

def explora(s):                     # fa el backtracking a partir
                                    # del node d'exploracio s.
    global z,x                      # millor solucio obtinguda.
            # AFEGIU CODI AQUI
    return                          # (elimineu aquesta linia)

def escriu():                       # escriu la solucio.
            # AFEGIU CODI AQUI
    return                          # (elimineu aquesta linia)

def main():
    global G,n                      # el graf i el nombre de nodes.
    global z,x                      # millor solucio obtinguda.

    G = llegir_graf()
    n = G.number_of_nodes()
    z = 10000                       # inicialitem un minim a oo.
    x = [0]*(1+n)
    s = node()
    explora(s)
    escriu()
```

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.

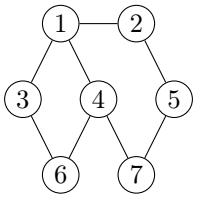
1.




```
1 2 3
2 3 4
3 4
```

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

2.




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

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