Algorísmica Avançada

Parcial I de Pràctiques

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

2 de novembre de 2011

Grau en Enginyeria Informàtica

Universitat de Barcelona

Per fer aquest parcial no heu de crear cap fitxer nou. Tant les rutines auxiliars com les instàncies de prova que es mostren al final de l'enunciat estan penjades al campus. Descarregueu-les, i renomeneu l'arxiu font en python al format *P1_ C_ nom_ cognoms.py*. El parcial es composa de tres 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 tots tres, l'entrada de dades és un graf, que es llegirà d'un fitxer de text amb la rutina `llegir_graf()` que com podeu observar, també la teniu disponible en les rutines auxiliars al final de l'enunciat.

1. *Graf Biconnexe.* Un graf és *biconnexe* si no té punts d'articulació. Un punt d'articulació és un node tal que si es fa desaparèixer (juntament amb les arestes de la seva adjacència), el nombre de components connexes del graf augmenta. Feu una funció `biconnexe(G)`, que retorni un booleà indicant si el graf G és biconnexe. Analitzeu l'eficiència de la funció. (10 punts)

2. *Camins.* En un graf biconnexe, entre qualsevol parella de nodes hi ha dos camins disjunts. Feu una funcio `camins(G,u,v)` que donat un graf i dos nodes ens dongui les seqüències de nodes que representen cada un dels dos camins entre els dos nodes donats. Analitzeu l'eficiència de la funció. (30 punts)

3. *Biconnexions.* Quan en un graf hi ha un punt d'articulació, que en definitiva ve a ser un punt crític en la xarxa, podeu assegurar les connexions afegint una aresta entre les dues components que en resulten de fer desaparèixer l'articulació. Feu un programa que donat un graf, si no té punts d'articulació es pari. I si en té, asseguri les connexions eliminant-los. La sortida del programa, doncs, pot ser de dos tipus. En un cas tan sols treurà un missatge que dirà "el graf es biconnexe". En l'altre cas ens dirà quin és el node punt d'articulació, quina aresta ha afegit de totes les possibles per biconnectar el graf, i també els dos camins resultants entre els vèrtexos d'aquesta aresta. (60 punts)

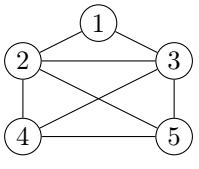
```
def llegir_graf():                                          #Θ(V + E)
    nom = input("Dona'm un nom pel graf:  ")                #Θ(1)
    G = nx.read_adjlist(nom)                                #Θ(V + E)
    return G                                                #Θ(1)

def dfs_visit(G,v):                                         #Θ(E/V)
    global vistos
    vistos.append(v)                                        #Θ(logV)
    for u in G.neighbors(v):                                #Θ(E/V)
        if u not in vistos:                                 #Θ(logV)
            dfs_visit(G,u)                                  #Θ(E/V)

def dfs(G):                                                 #Θ(V + E)
    global vistos
    q = 0                                                   #Θ(1)
    vistos = []                                             #Θ(1)
    for v in G:                                             #Θ(V)
        if v not in vistos:                                 #Θ(logV)
            q = q + 1                                       #Θ(1)
            dfs_visit(G,v)                                  #Θ(E/V)
    return q                                                #Θ(1)

def numero_de_components(G):                                #Θ(V + E)
    return dfs(G)                                           #Θ(1)

def punt_d_articulacio(G):                                  #Θ(V(V + E))
    p = numero_de_components(G)                             #Θ(V + E)
    for u in G:                                             #Θ(V)
        G2 = G.copy()                                       #Θ(V + E)
        G2.remove_node(u)                                   #Θ(V)
        q = numero_de_components(G2)                        #Θ(V + E)
        if q == p+1:                                        #Θ(1)
            return u                                        #Θ(1)
    return -1                                               #Θ(1)
```

Rutines auxiliars.

entrades i sortides

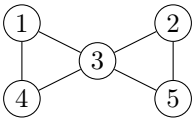
Seguidament es mostren dos exemples. A la primera columna hi ha el graf, a la segona el contingut del fitxer de text que el representa, i en la tercera, la sortida que hauria de donar el vostre programa.

1.




```
1 2 3
2 3 4 5
3 4 5
4 5
```

```
el graf es biconnexe
```

2. 

1 3 4
2 3 5
3 4 5

El node 3 es un punt d'articulacio.
He afegit aresta (1 , 2) per biconnectar el graf
$[1, 2]$
$[1, 3, 2]$