Departament de Disseny i
Programació de Sistemes Electrònics
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Prova final d'INFORMÀTICA

24 de gener de 2011 — Enginyeria de Sistemes TIC — 90 MINUTS

Exercici 1 [5 punts]. Una llista l conté una sèrie de cadenes de la mateixa longitud fomades pels caràcters `'0'` i `'1'`. La longitud de les cadenes no és afitada i, per tant, poden ser arbitràriament llargues. Es vol que implementeu una funció tal que, donada una llista com la descrita, interpreti cada element com una paraula binària, en calculi l'`AND` lògic de totes elles i retorni el nombre de 1's del resultat.

SOLUCIÓ:

Definim primer una funció que calculi l'AND de dues paraules i després fem un recorregut de la llista multiplicant cada paraula.

La funció andp es pot implementar de manera senzilla usant l'operador **and** de Python. Alternativament es podria fer un tractament per casos:

```
def andp(p,q):
    r = ''
    for i in range(len(p)):
        r += str(int(p[i]) and int(q[i]))
    return r
```

La solució del problema és ara senzilla. Només cal recórrer els elements de la llista i calcular l'AND:

```
def n_uns(l):
    prod = l[0]
    for e in l:
        prod = andp(prod,e)
    return prod.count('1')
```

Exercici 2 [3 punts]. Sigui M una matriu quadrada. Direm que M és triangular superior ssi tots els elements per sota de la diagonal principal són zero, és a dir $\forall i, j : i > j : M_{ij} = 0$.

1. Dissenyeu i implementeu la funció es_triangular(m) tal que donada una matriu m representada com a llista de llistes —llista de files— retorna True ssi m és triangular superior. Per exemple, per a la llista [[1,1,1], [0,1,1], [0,0,1]] la funció hauria de tornar True.

2. Dissenyeu i implementeu una funció tal que, donada una matriu com l'anterior, retorna l'índex de la columna les components de la qual sumen el valor més gran. Per exemple, per a la llista [[1,1,1], [0,1,1], [0,0,1]] hauria de tornar 2, atès que la columna 2 suma 3, la 1 suma 2 i la 0 suma 1.

SOLUCIÓ:

En el primer apartat cal aplicar un esquema de cerca sobre el triangle inferior de la matriu:

```
def es_triangular(m):
    for i in range(1,len(m)):
        for j in range(i):
            if m[i][j] == 0.0: return False
    return True
```

El segon apartat és, en essència, un càlcul del màxim. Per simplificar la feina primer dissenyem una funció que calcula la suma dels elements d'una columna. Aquesta funció tindrà en compte que la matriu és triangular superior:

```
def suma(m,c):
    """ retorna la suma de la columna c de la matriu m"""
    s = 0.0
    for f in range(c+1):
        s += m[f][c]
    return s
```

A continuació calculem el màxim fent un recorregut com és habitual:

```
def maxim(m):
    k = 0 # k es la columna de pes maxim
    for c in range(1,len(m)):
        if suma(m,k) < suma(m,c): k = c
    return k
```

Exercici 3 [2 punts]. Un antropòleg està estudiant la tribu dels Mackluskis. En aquest estudi ha preguntat a cada membre de la tribu a quins altres membres obeïa. Ha codificat les respostes en un diccionari com aquest: {`'Edgar':'Jaume'`, `'Jaume': 'Marta'`, `'Pere':'Jaume'`, `'Tomeu':'Marta'`}.

En aquest diccionari cada entrada correspon a una parella de noms. Una parella (A, B) en que A és la clau i B el valor significa que A obeeix directament a B i ho escriurem $A > B$. Per exemple, el Pere obeeix directament el Jaume. No succeirà mai que $A > B$ i $A > C$ essent B i C membres diferents. És a dir tot membre obeeix a una única persona. Tampoc es dona mai que $A > A$.

A banda de l'obediència directa es pot donar l'obediència transitiva. Fixeu-vos que en l'exemple, Pere > Jaume > Marta. Per tant el Pere obeeix (transitivament) la Marta. Quan $A > B > \cdots > S$ ho escriurem com $A >^* S$. Fixeu-vos que si $A >^* B$ i $B >^* C$, llavors $A >^* C$.

Es demana que implementeu la funció un_sol_cap(t) que retorna True ssi a la tribu t té un membre a qui tothom obeeix.

SOLUCIÓ:

En aquest problema cal primer adonar-se'n de dues coses. La primera és que un cap no obeeix a ningú. La segona és que pot ser que hi hagi més d'un cap.

Si A és un cap, llavors per força ha de formar part dels valors dels diccionari però no de les claus. Per tant, podem calcular els caps buscant aquests individus. Fem-ho:

```
def cerca_caps(t):
    lc = []
    for m in t.values():
        if m not in t and m not in lc: lc.append(m)
    return lc
```

Ara només resta determinar si la tribu té un sol cap:

```
def un_sol_cap(t):
    return len(cerca_caps(t)) == 1
```