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

Control de TECPRO

29 de maig de 2012 — Enginyeria de Sistemes TIC — 40 MINUTS

COGNOMS:

NOM: GRUP de LAB:

Exercici 1. Considereu la funció polinòmica $P(x) = a_0 + a_1x + \cdots + a_nx^n$. Noteu que aquesta funció es pot expressar com $P(x) = a_0 + xP'(x)$ on $P'(x)$ és una altra funció polinòmica. En aquest cas $P'(x) = a_1 + a_2x + \cdots + a_nx^{x-1}$. Basant-se en aquest fer es demana que dissenyeu una funció recursiva tal que, donat un polinomi P i un valor x retorni $P(x)$, és a dir avaluï P en el punt x. A tal efecte representeu el polinomi com una llista de coeficients $[a_0, a_1, \ldots, a_n]$.

SOLUCIÓ:

La funció que se'ns demana té el següent aspecte:

```
def avalua(p,x):
    """Avalua 'p' en el punt 'x'"""
    pass
```

Hi ha dos casos possibles:

1. (base) El polinomi té un sol coeficient, que ha de ser per força el de grau 0. És a dir $P(x) = a_0$. Llavors avaluar el polinomi significa tornar directament aquest coeficient.
2. (recurrència) El polinomi té més d'un coeficient. Llavors apliquem la fórmula que dóna l'enunciat.

Expressant això amb `Python`, tenim:

```
def avalua(p,x):
    """Avalua 'p' en el punt 'x'"""
    if len(p) == 1:
        return p[0]
    else:
        return p[0] + avalua(p[1:], x)
```

Exercici 2. La fusió de llistes és una operació tal que donada dues llistes ordenades l_1 i l_2 calcula una tercera llista ordenada l_3 que conté els elements d'l_1 i l_2. Per exemple, si les llistes són ordenades creixents i d'enters, la fusió de [1,5,23,45] i [6,6,20,90] és la llista [1,5,6,6,20,23,45,90].

Es demana que dissenyeu una funció recursiva tal que, donada dues llistes d'enters ordenades de manera creixent, torni la seva fusió ordenada. Noteu que l'algorisme ha de tenir un cost temporal asimptòtic en cas pitjor $O(m+n)$, on n és la mida de la primera llista i m la mida de la segona.

SOLUCIÓ:

Apliquem el mètode de sempre: plantejem la funció, analitzem els casos i després els traslladem al llenguatge. En aquest cas la funció que es demana té la següent especificació:

```
def fusio(l1, l2):
    """Fusiona les llistes ordenades `l1` i `l2` """
```

L'estudi de casos resulta ser:

1. (base) Si l1 és buida, llavors la fusió es redueix a l2.

2. (base) De manera simètrica, i l2 és buida, llavors la fusió es redueix a l1.

3. (recurrència) Si tant l1 com l2 contenen elements, llavors la fusió depen de quin element és més petit, l1[0] o l2[0]:

   a) Si l1[0]<l2[0] llavors la llista resultant és [l1[0]]+fusio(l1[1:],l2).

   b) Si l1[0]>=l2[0] llavors la llista resultant és [l2[0]]+fusio(l1,l2[1:]).

Noteu que els dos primers casos ja inclouen la possibilitat de que tant l1 com l2 siguin buides.

Traslladant l'anàlisi a `Python` resulta:

```
def fusio(l1, l2):
    """Fusiona les llistes ordenades `l1` i `l2` """
    if not l1:
        return l2
    elif not l2:
        return l1
    else:
        if l1[0] < l2[0]:
            return [l1[0]] + fusio(l1[1:], l2)
        else:
            return [l2[0]] + fusio(l1, l2[1:])
```

Exercici 3. Considereu la següent funció en la que el paràmetre l és una llista de paraules.

```
def f(l):
    for p in l:
        if 'a' in p:
            return True
    return False
```

Si el cost asimptòtic temporal en cas pitjor de l'operació `'a'in`p és $O(m)$ essent m la longitud de la paraula p, quin és el cost asimptòtic temporal en cas pitjor de la funció f()?

SOLUCIÓ:

El cas pitjor es donarà quan la llista l no conté cap paraula p que contingui la lletra `'a'`. Si assumim que l té n paraules i que la paraula més llarga possible té m lletres, llavors el cost assimptòtic en cas pitjor és $O(mn)$.

Exercici 4. La següent funció determina si un mot m és palíndrom:

```
def p(m):
    l = m[:]
    while l:
        if l[0] != l[-1]:
            return False
        else
            l = l[1:-1]
    return True
```

Calculeu la funció de temps $T(n)$ en cas pitjor on n és la longitud de la paraula m. Assumiu que l'accés a qualsevol element d'un mot donat el seu índex triga un temps constant. De quin ordre asimptòtic seria $T(n)$?

SOLUCIÓ:

Per calcular el cost en temps assumirem els següents costos constants: una assignació t_a, l'accés al mot t_m, el cost d'una iteració t_i i el cost d'un condicional t_c.

Per la hipòtesi de cas pitjor, la iteració **while** itera $\frac{n}{2}$ essent n la mida de la paraula m. Això és així per que a cada iteració la paraula m disminueix en dues lletres la seva llargada. Així doncs el cost en funció d'n és:

$$\begin{aligned} T(n) &= (nt_m + t_a) + n(t_i + t_c + 4t_m) \\ &= (t_1 + t_c + 5t_m)n + t_a \end{aligned}$$

La funció, doncs, té un cost assimptòtic $O(n)$