Primer Parcial de PRG - ETSInf
Data: 21 de març de 2011. Duració: 1 hora i 30 minuts

NOTA: Cal contestar en fulles apart. No és necessari entregar aquesta fulla.

1. (3 punts) El següent mètode **iteratiu** calcula el producte d'un vector fila `x` per una matriu quadrada `a`, obtenint com a resultat un altre vector fila. Per tractar-se d'una matriu quadrada tant el vector `x` com el vector resultat tindran la mateixa dimensió, és a dir, el nombre de files de la matriu.

```
/**  x.length = a.length  i  a  és una matriu quadrada */
static double [] vectorPerMatriu( double x[], double a[][] )
{
    double r[] = new double [ a.length ];

    for( int i=0; i < r.length; i++ ) {
        r[i] = 0.0;
        for( int j=0; j < x.length; j++ )
            r[i] += x[j] * a[j][i];
    }
    return r;
}
```

Per exemple, per $x = (2, 1, 3)$, un vector fila de dimensió 1×3, i $a = \begin{pmatrix} 4 & 3 & 2 \\ 2 & 5 & 1 \\ 1 & 0 & 3 \end{pmatrix}$, una matriu de dimensió 3×3, el producte $x \times a$ és $r = (13, 11, 14)$, un vector fila de dimensió 1×3.

Es demana:

a) Indicar quina és la grandària o talla del problema, així com l'expressió que la representa.

b) Identificar, en cas que n'hi hagués, les instàncies del problema que representen el cas millor i pitjor de l'algorisme.

c) Triar una unitat de mesura per a l'estimació del cost (passos de programa, instrucció crítica) i d'acord amb ella, obtenir una expressió matemàtica, el més precisa possible, del cost temporal del programa, a nivell global o en les instàncies més significatives si les hi ha.

d) Expressar el resultat anterior utilitzant notació asimptòtica.

Solució:

1. La talla o grandària del problema és el nombre de files de la matriu `a.length`, que al coincideix amb el de columnes de la propia matriu i amb el nombre de components del vector `x`.D'ara endavant, anomenarem a aquest nombre n. És a dir, $n =$ `a.length`.

2. Es tracta d'un problema de recorregut, per tant, per a una mateixa talla del problema no hi ha instàncies significatives.

3. Si es pren com a unitat de mesura el pas de programa, es pot expressar el cost de l'algorisme de la manera següent: $T(n) = 1 + \sum_{i=0}^{n-1}(1 + \sum_{j=0}^{n-1} 1) = 1 + \sum_{i=0}^{n-1}(1 + n) = 1 + n + n^2$.

   Si es tria com a unitat de mesura per a l'estimació del cost una instrucció crítica, en concret, seleccionant l'assignació més interna `r[i] += x[j] * a[j][i]`, obtenim la següent funció de cost temporal: $T(n) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} 1 = n^2$.

4. En notació asimptòtica: $T(n) \in \Theta(n^2)$.

2. (3.5 punts) Per obtenir la posició de l'últim element senar d'un array `v` d'enters, es proposa el següent mètode **iteratiu**:

```
static int posUltimSenar( int[] v )
{
    int i = v.length-1;
    while( i >= 0 && v[i]%2 == 0 ) i--;
    return i;
}
```

Es demana:

a) Indicar quina és la grandària o talla del problema, així com l'expressió que la representa.

b) Identificar, en cas que n'hi hagués, les instàncies del problema que representen el cas millor i pitjor de l'algorisme.

c) Triar una unitat de mesura per a l'estimació del cost (passos de programa, instrucció crítica) i d'acord amb ella, obtenir una expressió matemàtica, el més precisa possible, del cost temporal del programa, a nivell global o en les instàncies més significatives si les hi ha.

d) Expressar el resultat anterior utilitzant notació asimptòtica.

Solució:

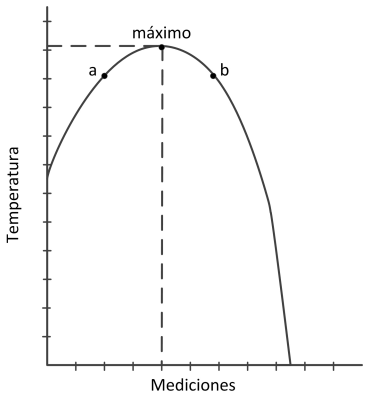
a) La grandària o talla del problema és el nombre d'elements de l'array `v` i l'expressió que la representa és `v.length`. D'ara endavant, anomenarem a aquest nombre n. És a dir, $n =$ `v.length`.

b) Es tracta d'un problema de cerca (seqüencial iterativa descendent) i, per tant, per a una mateixa talla si que presenta instàncies diferents. El *cas millor* es dóna quan l'últim element de l'array és senar (`v[v.length-1]%2==1`). El *cas pitjor* passa quan tots els elements de l'array són parells ($\forall$`i`, `0`$\leq$`i`$<$`v.length`, `v[i]%2==0`).

c) Optem per escollir la instrucció crítica com a unitat de mesura per a l'estimació del cost. L'única instrucció que podem considerar com a crítica és la guarda del bucle: `i>=0 && v[i]%2==0` (s'executa sempre una vegada més que qualsevol de les de dins del bucle). En el *cas millor* només s'executarà una vegada i la funció de cost temporal en aquest cas serà $T^m(n) = 1$. En el *cas pitjor* es repetirà tantes vegades com elements tinga la matriu més una (quan es fa falsa) i la funció de cost serà $T^p(n) = \sum_{i=0}^{n} 1 = n + 1$.

d) En notació asimptòtica: $T^m(n) \in \Theta(1)$ i $T^p(n) \in \Theta(n)$. Per tant, $T(n) \in \Omega(1)$ i $T(n) \in O(n)$, és a dir, el cost temporal està acotat inferiorment per una funció constant i superiorment per una funció lineal amb la talla del problema.

3. (3.5 punts) Es disposa d'una sèrie de mesures de temperatura que formen part d'una corba parabòlica (positiva) en un array `v` de reals. El següent mètode **recursiu** obté el màxim valor de la corba representada per l'array `v`.



Per a aquest tipus de corbes, donat un valor se sap que si l'anterior i el posterior són menors que ell, ens trobem en el punt màxim mesurat. Si per contra l'anterior és més gran i el posterior és menor, estem a la dreta del màxim, i si l'anterior és menor i el posterior gran, estem a l'esquerra del màxim. Un altra situació és impossible en aquest tipus de corba. A la corba de la figura, es pot observar que els punts `a` i `b` (anterior i posterior al punt màxim, respectivament) són menors que el punt `máximo`.

```
static double maximaTemp( double[] v, int ini, int fin )
{
    if ( ini == fin )
        return v[ini];
    else if ( ini == fin-1 )
        return Math.max( v[ini], v[fin] );
    else {
        int meitat = (fin+ini)/2;
        if ( v[meitat] > v[meitat-1] && v[meitat] > v[meitat+1] )
            return v[meitat];
        else if ( v[meitat] < v[meitat-1] && v[meitat] > v[meitat+1] )
            return maximaTemp( v, ini, meitat );
        else
            return maximaTemp( v, meitat, fin );
    }
}
```

La crida inicial serà: `double maxim = maximaTemp( v, 0, v.length-1 );`

Es demana:

a) Indicar quina és la talla del problema i quina expressió la defineix.

b) Determinar si hi ha instàncies significatives. Si n'hi ha, identificar les que representen els casos millor i pitjor de l'algorisme.

c) Escriure l'equació de recurrència del cost temporal en funció de la talla per a cada un dels casos si n'hi ha diversos, o una única equació si només hi hagués un cas. Resoldre per substitució.

d) Expressar el resultat anterior fent servir notació asimptòtica.

Solució:

1. La talla del problema és el nombre d'elements de l'array `v` entre les posicions `ini` i `fin`. És a dir, l'expressió de la talla és $n =$ `fin-ini+1`.

2. Es tracta d'un problema de cerca (binària), de manera que per a una mateixa talla si es distingeixen instàncies. El *cas millor* es dóna quan el punt màxim es troba en `v[(fin+ini)/2]` sent `ini=0` i `fin=v.length-1`. El *cas pitjor* es dóna quan per tal de trobar el punt màxim es fan el major nombre de crides possible, és a dir, quan la talla del problema es redueix fins aplegar a la talla del cas base.

3. En el *cas millor*, el cost és constant, $T^m(n) = c$. Per obtenir el cost en el *cas pitjor*, plantegem l'equació de recurrència: $T^p(n) = \begin{cases} T^p(n/2) + k & \text{si } n > 2 \\ k' & \text{si } n = 1 \text{ o } n = 2 \end{cases}$

   Resolent per substitució: $T^p(n) = T^p(n/2) + k = T^p(n/2^2) + 2k = \ldots = T^p(n/2^i) + ik$. En arribar al cas base, per talla 1, $n/2^i = 1 \rightarrow i = log_2 n$; i per talla 2, $n/2^i = 2 \rightarrow n = 2^i * 2 \rightarrow i = log_2 n + log_2 2 = log_2 n + 1$. Amb el que $T^p(n) \approx k' + k log_2 n$.

4. En notació asimptòtica: $T^m(n) \in \Theta(1)$ i $T^p(n) \in \Theta(log_2 n)$. Per tant, $T(n) \in \Omega(1)$ i $T(n) \in O(log_2 n)$, és a dir, el cost temporal està acotat inferiorment per una funció constant i superiorment per una funció logarítmica amb la talla del problema.