PROGRAMACIÓ CIENTÍFICA. CURS 2019-20. PRIMAVERA. PRÀCTIQUES DE LABORATORI D'ORDINADORS

LLISTA 4. FUNCIONS RECURSIVES

1 Es vol calcular el determinant d'una matriu $n \times n$ de tipus *tridiagonal*, $A_n = (a_{ij})$, els elements no nuls de la qual són $a_{i,i} = a$ i $a_{i,i+1} = a_{i+1,i} = b$ ($\forall i$ possibles), on a i b són dos paràmetres reals.

És pot fer mitjançant una recurrència lineal i homogènia, de segon ordre, a coeficients constants. Concretament, si s'anomena $d_n \equiv det(A_n)$ ($\forall n \geq 1$), deduïu que es verifica

$$d_n = a\, d_{n-1} - b^2\, d_{n-2}\ ,\ \forall n \geq 3\ ;$$

i d_1 i d_2 es poden calcular directament.

Feu un programa que llegeixi els valors de les dades n, a i b; i calculi d_n de dues maneres:

- una, implementant la recurrència en una funció recursiva;

- l'altra, implementant la recurrència en una funció que no sigui recursiva, sinó que es faci mitjançant una iteració.

Compteu els temps d'execució en els dos casos i compareu-los (useu valors de n a partir de 30). Podeu explicar aquests resultats?

2 Es considera la *recurrència de segon ordre*

$$x_n = 0.4(x_{n-1}^2 + x_{n-2})\ ,\ \ \forall n \geq 2\ .$$

Donats x_0 i $x_1 \in \mathbb{R}$, la recurrència anterior defineix una successió $(x_n)_{n\geq 0}$.

Feu un programa que llegeixi x_0, x_1 i n, i calculi l'element x_n de la successió de dues maneres:

- Usant una funció de capçalera

  ```
  double recurs(int n, double x0, double x1)
  ```

  que sigui recursiva. O sigui, cal programar directament la recurrència.

- Usant una funció de capçalera

  ```
  double iterat(int n, double x0, double x1)
  ```

  que no sigui recursiva, sinó que implementi un càlcul iteratiu usant una instrucció `for` i 3 variables de tipus `double`, on es van guardant i actualitzant 3 termes consecutius.

Programeu també el càlcul del temps d'execució de cadascun dels dos mètodes. Comproveu que el mètode recursiu és molt menys eficient i feu-ne una anàlisi que ho expliqui.

3 Es considera el model de Nicholson-Bayley per a estudiar com evolucionen les poblacions de dues especies: hostes i paràsits. És una *recurrència de primer ordre amb dues variables* (H, P):

$$\left\{ \begin{array}{rcl} H_n &=& kH_{n-1}\exp(-aP_{n-1}) \\ P_n &=& cH_{n-1}(1-\exp(-aP_{n-1})) \end{array} \right. ,\forall n > 0\ ,$$

on $k > 1$, $c > 0$ i $a > 0$ són paràmetres fixats, i $H_0 > 0$ i $P_0 > 0$ són les poblacions inicials de cada espècie.

Feu un programa on es fixen els valors dels paràmetres (per exemple $k = 2$, $c = a = 1$); es llegeixen les dades H_0, P_0 i n; i es calculen H_n i P_n de dues maneres:

- Usant una funció recursiva, on s'implementi directament la recurrència.
- Usant una funció no recursiva, on es programa un mètode iteratiu.

En tots dos casos, la funció ha de tenir prototipus

```
void nomfun(int n, double h0, double p0, double *hn, double *pn)
```

on cal posar un `nomfun` diferent en cada cas.

Per tal que els resultats siguin interessants, cal que doneu valors inicials aproximats a *un punt d'equilibri inestable* $H_0 \approx \frac{k\log(k)}{(k-1)ac}$ i $P_0 \approx \frac{\log(k)}{a}$.

Nota. És fàcil analitzar que els dos mètodes tenen un cost lineal respecte n. Si ho intenteu comprovar calculant els temps d'execució, és complicat de veure-ho, ja que són molt ràpids.

4 Donat un vector de nombres reals, es vol calcular el seu valor màxim i el seu valor mínim.

La manera natural *no recursiva* és la següent. Inicialment es considera que tant el màxim com el mínim són el primer element. Llavors es fa un recorregut lineal del vector, cada element es va comparant amb el mínim i el màxim actuals, i, si cal, algun d'aquest es va actualitzant.

Una altra manera de fer-ho és *recursiva*, usant el principi anomenat *divideix i venceràs*. La idea, per al màxim, és la següent (al mateix temps es pot fer també el mínim, de manera anàloga):

El valor màxim del vector $v[\,]$ entre les posicions p i q, on $p \leq q$, es pot calcular així:

- Si $p = q$ llavors el màxim és $v[p]$. En cas contrari, es continua:
- Sigui r una posició qualsevol entre p i $q-1$ (per exemple, $\lfloor (p+q)/2 \rfloor$).
- Sigui $max1$ el valor màxim del vector $v[\,]$ entre les posicions p i r.
- Sigui $max2$ el valor màxim del vector $v[\,]$ entre les posicions $r+1$ i q.
- El valor buscat és el màxim entre $max1$ i $max2$.

Feu un programa que llegeixi un valor n gran, generi aleatòriament un vector de n valors reals, tots ells a l'interval $[-100, +100]$, i després busqui l'element màxim i l'element mínim del vector de les dues maneres anteriors.

Per a cada cas, cal fer una funció de prototipus

```
void nom(float *v, int p, int q, float *max, float *min)
```

les quals s'hauran de cridar des de la funció `main` amb els valors $p = 0$ i $q = n-1$.

Respecte l'eficiència, és el cas recursiu molt pitjor que el no recursiu? Depèn això de com es tria r cada vegada?