PRG - ETSInf. PRÀCTIQUES. Curs 2013-14. Parcial 1.
14 d'abril de 2014. Duració: 1 hora

1. 2.5 punts La següent implementació de l'algorisme recursiu del problema de les torres d'Hanoi es compila correctament i no es produeix cap error en la seua execució.

```
public static void hanoi(int discos, String origen, String destino, String auxiliar) {
  if (discos==1)
     moverDisco(origen, destino);
  else {
      hanoi(discos-1, origen, auxiliar, destino);
      moverDisco(origen, destino);
      hanoi(discos-2, origen, destino, auxiliar);
  }
}
```

No obstant, conté errors lògics, és a dir, el resultat de la seua execució és incorrecte respecte al vist en pràctiques.

Es demana: localitzar els errors, explicar-los i corregir el codi per tal que la solució siga la vista en pràctiques.

Solució: La segona crida recursiva, `hanoi(discos-2, origen, destino, auxiliar)`, és incorrecta per les següents raons:

- Després de desplaçar un disc, encara queden `discos-1` en la torre auxiliar, no `discos-2`.
- En aquesta segona crida recursiva, la torre `auxiliar` ha de fer la funció de `origen`, i la torre `origen` la funció de `auxiliar`.

En conseqüència, l'error es corregeix canviant la instrucció per la següent:

```
hanoi(discos-1, auxiliar, destino, origen);
```

2. 2.5 punts Es disposa d'un mètode `boolean esSufixe(String a, String b)` que torna `true` si la cadena `a` és un sufixe de `b` i `false` en cas contrari.

Es demana: implementar un mètode recursiu que, fent ús del mètode anterior, torne si una cadena donada `s` és subcadena d'un altra `t`. La capçalera del mètode ha de ser necessàriament:

```
public static boolean esSubcadena(String s, String t)
```

Recorda que `s.substring(i,j)` és un mètode disponible en la llibreria de Java que torna un objecte `String` que representa la substring de `s` formada per els caràcters compresos entre el `i` i el `j-1`.

Solució:

```
public static boolean esSubcadena(String s, String t) {
   if (s.length()>t.length()) return false;
   else if (esSufixe(s, t)) return true;
   else return esSubcadena(s, t.substring(0, t.length()-1));
}
```

3. 2.5 punts El mètode estàtic `ordena(int[])` de la classe `AlgorismesMesurables` té, en el cas promedi, un cost quadràtic, prenent com talla la grandària de l'array que se li passa com paràmetre.

Es demana: completar el codi del següent mètode per tal d'estudiar el seu cost empíric en el cas promedi, realitzant 50 repeticions per a cada talla `t`. Es poden utilitzar els següents mètodes:

- `public static int[] crearArrayAleatori(int talla)`, que torna un array de grandària `talla` amb els seus elements generats de forma aleatòria.
- `public static long nanoTime()`, de la classe `java.lang.System`, que torna el valor actual del temporitzador més precis del sistema en nano-segons.

```
public static void mesuraOrdenacio() {
    System.out.printf("# Talla    Temps\n");
    System.out.printf("#----------------\n");
    long t1 = 0, t2 = 0, tt = 0; double tmedt = 0;   // Temps
    for (int t=10000; t<=100000; t+=10000) {

      //       COMPLETAR

      System.out.printf("%8d  %8d\n", t, tmedt/1000);
    }
}
```

Solució:

```
public static void mesuraOrdenacio() {
    System.out.printf("# Talla    Temps\n");
    System.out.printf("#----------------\n");
    long t1 = 0, t2 = 0, tt = 0; double tmedt = 0;   // Temps
    for (int t=10000; t<=100000; t+=10000) {
      tt = 0;                          // Temps acumulat inicial a 0
      for (int r=0; r<50; r++) {
        int[] a = crearArrayAleatori(t);
        t1 = System.nanoTime();        // Temps inicial
        AlgorismesMesurables.ordena(a);
        t2 = System.nanoTime();        // Temps final
        tt += (t2-t1);                 // Actualitzar temps acumulat
      }
      tmedt = (double)tt/50;           // Temps promedi
      System.out.printf("%8d  %8d\n", t, tmedt/1000);
    }
}
```

4. 2.5 punts Suposant que els resultats de l'estudi empíric anterior s'han emmagatzemat al fitxer `resultats` en dues columnes (talla i temps), **es demana:**

a) (1.75 punts) Definir la funció a ajustar a la curva definida per l'estudi del cost teòric i realitzar l'ajust d'aquesta funció mitjançant l'ordre `fit` de Gnuplot, la sintaxi de la qual és la següent:

`fit` *funció fitxer* `using` i:j `via` *paràmetres*

on:

- *funció*: indica el nom de la funció a ajustar.

- *fitxer*: indica el nom del fitxer amb les dades a ajustar (s'especifica entre cometes dobles).
- `using` i`:`j: especifica les columnes del fitxer de dades que s'usaran (i per a l'eix X i j per a l'eix Y).
- `via` *paràmetres*: especifica els paràmetres (separats per comes) de la funció a ajustar.

b) (0.75 punts) Una vegada fet l'ajust, com calcularies el temps del mètode per a una talla igual a 25000?

Solució:

a) `f(x) = a*x*x + b*x + c;` `fit f(x) "resultats" using 1:2 via a,b,c`

b) Una vegada coneguts `a`, `b` i `c` (resultat d'haver realitzat l'ordre `fit`), calcular `f(25000)`