

# Recuperación Primer Parcial de Estructuras de Datos y Algoritmos (EDA) – 25 de Junio de 2021

Duración: 1h. 30m.

|                   |       |
|-------------------|-------|
| APELLIDOS, NOMBRE | GRUPO |
|                   |       |

**1.- (3 puntos)** El siguiente método recibe dos **ListaConPI** genéricas y comprueba si ambas listas contienen los mismos elementos pero en orden inverso.

Por ejemplo, el método devolverá **true** si la primera lista es [1, 3, 5, 7, 9] y la segunda es [9, 7, 5, 3, 1]. Este método no debe modificar el contenido de las listas.

Escribe en cada recuadro el número de la opción (ver listado a la derecha) que le corresponde. Una opción puede no aparecer en ningún recuadro, o puede usarse para rellenar uno o varios recuadros.

```
public  boolean inversas(ListaConPI  l1,  
                                     ListaConPI  l2) {  
    if (l1. != l2.) return false;  
    l1.;  
    l2.;  
    return inversasRec(l1, l2);  
}  
  
private  boolean inversasRec(ListaConPI  l1,  
                                       ListaConPI  l2) {  
    if (l1.) return true;  
    E e1 = l1.;  
    l1.;  
    if (!inversasRec(l1, l2)) return ;  
    E e2 = l2.;  
    l2.;  
    return ;  
}
```

- ① <E>
- ② <E extends Comparable<E>>
- ③ static <E>
- ④ static <E extends Comparable<E>>
- ⑤ inicio()
- ⑥ siguiente()
- ⑦ recuperar()
- ⑧ esFin()
- ⑨ talla()
- ⑩ esVacía()
- ⑪ eliminar()
- ⑫ true
- ⑬ false
- ⑭ e1.compareTo(e2) == 0
- ⑮ e1.equals(e2)
- ⑯ e1 == e2

## 2.- Aplicación de la estrategia **Divide y Vencerás** (DyV).

**2.1.- (2,70 puntos).** Dado un array ordenado ascendentemente de **n** enteros, con valores en el rango **[1 .. n]**, se quiere encontrar, si existe, el único valor entero que aparece repetido. Si no existe ningún valor repetido, se indicará con el valor especial **-1**.

Implementa un método estático que, aplicando la estrategia **Divide y Vencerás**, resuelva el problema del modo más eficiente posible. Según la estrategia DyV, hay que implementar 2 métodos: uno lanzadera, otro recursivo.

La tabla de la derecha muestra algunos ejemplos.

| Array de enteros, <b>v</b>  | resultado |
|-----------------------------|-----------|
| [1, 2, 3, 4, 4, 5, 6, 7, 8] | 4         |
| [1, 2, 2, 3, 4, 5, 6, 7, 8] | 2         |
| [1, 2, 3, 4, 5, 6, 7, 8, 8] | 8         |
| [1, 2, 3, 4, 5, 6, 7, 8, 9] | -1        |
| [1, 1, 2, 3, 4]             | 1         |

```
public static int metodo2(int[] v) {  
    return metodo2(v, 0, v.length - 1);  
}  
  
public static int metodo2(int[] v, int ini, int fin) {  
    if (ini > fin) return -1; // subarray vacío, no existe elemento repetido  
    int mitad = (ini + fin) / 2;  
    if (v[mitad] != mitad + 1) { // comprobar si en mitad está el elemento repetido  
        if (mitad > 0 && v[mitad] == v[mitad - 1]) { return v[mitad]; }  
        // si el elemento central no está en su posición  
        // entonces el elemento repetido estará en una posición anterior  
        return metodo2(v, ini, mitad - 1);  
    }  
    // si el elemento central está en su posición  
    // entonces el elemento repetido estará en una posición posterior  
    return metodo2(v, mitad + 1, fin);  
}
```

### 2.2.- Estudia el coste Temporal del método recursivo que has implementado:

**a) (0,10 puntos).** Expresa la talla del problema **x** en función de los parámetros del método.

**x = fin - ini + 1** .

**b) (0,40 puntos).** Escribe la(s) Relación(es) de Recurrencia que expresa(n) el coste del método.

En el caso general, cuando **x > 0** ,

**$T^p(x) = T^p(x/2) + k$**

**c) (0,30 puntos).** Resuelve la(s) Relación(es) de Recurrencia del apartado **b)**, indicando el(los) Teoremas de Coste que usas y escribiendo el coste Temporal del método en notación asintótica (**O** y  **$\Omega$**  o bien  **$\Theta$** ).

**Aplicando el teorema 3 con  $a=1$  y  $c=2$ ,  $T^p(x) \in \Theta(\log x)$ . Por tanto,  $T(x) \in \Omega(1)$  y  $T(x) \in O(\log x)$ .**

**3.- (3,50 puntos)** Implementa un método estático y genérico que, dado un map **m**, de tipo **Map<C,V>**, devuelva el valor, de tipo **V**, que sea el más frecuente en el map **m**. Se considera (precondición) que existe ese valor y que su frecuencia de aparición en el map **m** es, al menos, 2. Se debe implementar de modo eficiente, de modo que se acceda una sola vez a cada entrada del map **m**. Se puede usar otro map como estructura auxiliar.

Ejemplo. Sea **m** un **Map<String,Integer>**, donde las claves son nombres de jugadores de baloncesto y los valores son los puntos anotados. El resultado de invocar el método será **264**, los puntos anotados por más jugadores.

**m**

|                    |     |
|--------------------|-----|
| "Larry Bird"       | 258 |
| "Michael Jordan"   | 362 |
| "Magic Johnson"    | 264 |
| "Patrick Ewing"    | 258 |
| "Shaquille O'Neal" | 264 |
| "David Robinson"   | 243 |
| "Pau Gasol"        | 264 |

```
public static <C,V> V metodo3(Map<C,V> m) {
    Map<V,Integer> m2 = new TablaHash<>(m.talla());
    Integer max = 0; // cero o uno
    V vMax = null;
    ListaConPI<C> lc = m.claves();
    for (lc.inicio(); !lc.esFin(); lc.siguiente()) {
        C c = lc.recuperar();
        V v = m.recuperar(c);
        Integer n = m2.recuperar(v);
        if (n == null) m2.insertar(v, 1);
        else {
            n++;
            if (n > max) { max = n; vMax = v; }
            m2.insertar(v, n);
        }
    }
    return vMax;
}
```

## ANEXO

### Las interfaces Map y ListaConPI del paquete modelos.

```
public interface Map<C, V> {  
    V insertar(C c, V v);  
    V eliminar(C c);  
    V recuperar(C c);  
    boolean esVacio();  
    int talla();  
    ListaConPI<C> claves();  
}
```

```
public interface ListaConPI<E> {  
    void insertar(E e);  
    void eliminar();  
    void inicio();  
    void siguiente();  
    void fin();  
    E recuperar();  
    boolean esFin();  
    boolean esVacia();  
    int talla();  
}
```

### Teoremas de coste

#### Teorema 1:

$f(x) = a \cdot f(x - c) + b$ , con  $b \geq 1$

- si  $a=1$ ,  $f(x) \in \Theta(x)$ ;
- si  $a>1$ ,  $f(x) \in \Theta(a^{x/c})$ ;

#### Teorema 2:

$f(x) = a \cdot f(x - c) + b \cdot x + d$ , con  $b$  y  $d \geq 1$

- si  $a=1$ ,  $f(x) \in \Theta(x^2)$ ;
- si  $a>1$ ,  $f(x) \in \Theta(a^{x/c})$ ;

#### Teorema 3:

$f(x) = a \cdot f(x/c) + b$ , con  $b \geq 1$

- si  $a=1$ ,  $f(x) \in \Theta(\log_c x)$ ;
- si  $a>1$ ,  $f(x) \in \Theta(x^{\log_c a})$ ;

#### Teorema 4:

$f(x) = a \cdot f(x/c) + b \cdot x + d$ , con  $b$  y  $d \geq 1$

- si  $a < c$ ,  $f(x) \in \Theta(x)$ ;
- si  $a = c$ ,  $f(x) \in \Theta(x \cdot \log_c x)$ ;
- si  $a > c$ ,  $f(x) \in \Theta(x^{\log_c a})$ ;

### Teoremas maestros

**Teorema para recurrencia divisora:** la solución a la ecuación  $T(x) = a \cdot T(x/b) + \Theta(x^k)$ , con  $a \geq 1$  y  $b > 1$  es:

- $T(x) = O(x^{\log_b a})$  si  $a > b^k$ ;
- $T(x) = O(x^k \cdot \log x)$  si  $a = b^k$ ;
- $T(x) = O(x^k)$  si  $a < b^k$ ;

**Teorema para recurrencia sustractora:** la solución a la ecuación  $T(x) = a \cdot T(x-c) + \Theta(x^k)$  es:

- $T(x) = \Theta(x^k)$  si  $a < 1$ ;
- $T(x) = \Theta(x^{k+1})$  si  $a = 1$ ;
- $T(x) = \Theta(a^{x/c})$  si  $a > 1$ ;