Programación II - 2012/2013 - Universitat Jaume I
Evaluación continua - Módulo 1 - Teoría

25 de marzo de 2013

La duración máxima de esta prueba es de 25 minutos. No puedes consultar libros ni apuntes.

APELLIDOS:	Marca tu grupo de teoría
NOMBRE:	☐ TE1 ☐ TE2 ☐TE3

Ejercicio 1 (1,5 puntos)

Expresa, empleando la notación O, el tiempo de ejecución en el peor de los casos de los algoritmos indicados a continuación.

Búsqueda secuencial de un elemento en un vector: $O(\quad)$

Búsqueda binaria de un elemento en un vector: $O(\quad)$

Ordenación por selección de un vector: $O(\quad)$

Ordenación por fusión o mezcla (merge sort) de un vector: $O(\quad)$

Ejercicio 2 (4,5 puntos)

Los siguiente métodos tienen como parámetros dos vectores en los que no existen elementos repetidos y calculan si todos los elementos del primer vector están contenidos en el segundo vector. Expresa, empleando la notación O, el tiempo de ejecución en el peor de los casos de cada método. Indica también qué vectores deben estar ordenados para que los códigos proporcionados sean válidos.

(a) **[1,5 puntos]**

```
public static boolean contieneA(int[] v1, int[] v2) {
    for (int i = 0; i < v1.length; i++) {
        boolean encontrado = false;
        for (int j = 0; j < v2.length && !encontrado; j++)
            if (v1[i] == v2[j])
                encontrado = true;
        if (!encontrado)
            return false;
    }
    return true;
}
```

El coste temporal del método `contieneA` es: $O(\quad)$

El vector `v1` debe estar ordenado: ☐ Sí ☐ No

El vector `v2` debe estar ordenado: ☐ Sí ☐ No

(b) **[1,5 puntos]**

```
public static boolean contieneB(int[] v1, int[] v2) {
    for (int i = 0; i < v1.length; i++) {
        boolean encontrado = false;
        int inicio = 0;
        int fin = v2.length - 1;
        while (inicio <= fin && !encontrado) {
            int medio = (inicio + fin) / 2;
            if (v1[i] == v2[medio])
                encontrado = true;
            else if (v1[i] < v2[medio])
                fin = medio - 1;
            else
                inicio = medio + 1;
        }
        if (!encontrado)
            return false;
    }
    return true;
}
```

El coste temporal del método `contieneB` es: $O($ $)$

El vector `v1` debe estar ordenado: ☐ Sí ☐ No

El vector `v2` debe estar ordenado: ☐ Sí ☐ No

(c) **[1,5 puntos]**

```
public static boolean contieneC(int[] v1, int[] v2) {
    int i1 = 0;
    int i2 = 0;
    while (i1 < v1.length && i2 < v2.length) {
        if (v1[i1] < v2[i2])
            return false;
        else if (v2[i2] < v1[i1])
            i2++;
        else {
            i1++;
            i2++;
        }
    }
    return i1 == v1.length;
}
```

El coste temporal del método `contieneC` es: $O($ $)$

El vector `v1` debe estar ordenado: ☐ Sí ☐ No

El vector `v2` debe estar ordenado: ☐ Sí ☐ No

Ejercicio 3 (4 puntos)

Indica lo que se escribe en la salida estándar al ejecutar cada uno de los siguientes programas. No es necesario que expliques por qué. Escribe tu respuesta a la derecha de cada `println`.

(a) **[2 puntos]**

```
public class Traza1 {
    public static void main(String[] args) {
        System.out.println(misterio(3));
        System.out.println(misterio(-54));
        System.out.println(misterio(100));
        System.out.println(misterio(123));
    }
    public static int misterio(int n) {
        if (n < 0)
            return misterio(-n);
        else if (n < 10)
            return n;
        else
            return misterio(n / 10) + n % 10;
    }
}
```

(b) **[2 puntos]**

```
public class Traza2 {
    public static void main(String[] args) {
        int[] v = { 30, 50, 70, 150, 170, 210, 250 };

        System.out.println(misterio(v, 40, 60));
        System.out.println(misterio(v, 80, 100));
        System.out.println(misterio(v, 200, 220));
        System.out.println(misterio(v, 220, 240));
    }
    public static int misterio(int[] v, int menor, int mayor) {
        return misterio(v, menor, mayor, 0, v.length - 1);
    }
    public static int misterio(int v[], int menor, int mayor,
                                        int inicio, int fin) {
        if (inicio > fin)
            return -1;
        int mitad = (inicio + fin) / 2;
        int valor = v[mitad];
        if (valor >= menor && valor <= mayor)
            return mitad;
        else if (valor > mayor)
            return misterio(v, menor, mayor, inicio, mitad - 1);
        else
            return misterio(v, menor, mayor, mitad + 1, fin);
    }
}
```