Escuela de Ingeniería Informática
Universidad de Oviedo

Algoritmia
2014-2015

Control 1 – 9 de marzo de 2015

Apellidos, nombre _____ NIF: _____

Pregunta 1 (1,5 p.)

Responde a las siguientes preguntas:

a) Si la complejidad de un algoritmo es O(nlogn), y dicho algoritmo toma 5 segundos para n=4, calcula el tiempo que tardará para n=16.

b) Considere ahora un algoritmo con complejidad $O(n^2)$. Si para t = 5 segundos el método pudiera resolver un problema con un tamaño de n = 1000, ¿cuál podría ser el tamaño del problema si dispusiéramos de un tiempo de 45 segundos?

Pregunta 2 (1,5 p.)

Indica la complejidad temporal de los siguientes fragmentos de código:

a)

```
public static void test(int n) {
    for (int i=0; i<n; i++) {
        for (int j=i; j<n; j++) {
            System.out.println("Values: i:"+i+ " j:"+j);
        }
    }
    test(n-1);
    test(n-1);
    test(n-1);
}
```

b)

```
//This is a code snippet created with the PHP language
function test($n) {
    for ($i = 1; $i < $n; $i *= 3) {
        for ($j = $n; $j >= 0; $j /= 4) {
            for ($k = 10; $k < $n*$n; $k++) {
                echo "i:$i j:$j k:$k"; //O(1)
            }
        }
    }
}
```

c)

```
    public static boolean metodo5 (int n)     {
 if (n<=5)
     cont++;
 else
 {
     for (int i=0; i<n; i+= 2)
         cont++ ;
     metodo5(n/2);
     metodo5(n/2);
 }
 return true;
}
```

Pregunta 3 (1 p.)

Teniendo en cuenta el algoritmo de ordenación Quicksort, responde a las siguientes preguntas:

a) Indica la complejidad del algoritmo en los casos mejor, medio y peor cuando se elige un buen pivote como elemento a particionar y el número de elementos es grande

b) Indica la complejidad del algoritmo en los casos mejor, medio y peor cuando se elige un mal pivote como elemento a particionar y el número de elementos es grande
c) Si la lista de elementos que queremos ordenar ya viene ordenada y elegimos como pivote el último elemento de la lista en cada iteración, ¿cuál es la complejidad? ¿por qué? ¿sería mejor o peor que la complejidad del método de Inserción directa en este caso?
d) Si la lista de elementos que queremos ordenar ya viene ordenada y elegimos como pivote el elemento central de la lista en cada iteración, ¿cuál es la complejidad? ¿por qué? ¿cambiaría la complejidad si en lugar del elemento central elegimos la mediana como pivote?

Pregunta 4 (2 p.)

Partiendo de la siguiente secuencia de enteros `(7, 1, 4, 5, 6, 2, 3)` realizar la traza para ordenar la secuencia utilizando el algoritmo Rápido (Quicksort) con el elemento central como pivote. Indicando claramente en cada paso, cuales son el pivote (circulo) y las particiones obtenidas (subrayando cada parte del vector).

Pregunta 5 (2,5 p.)

Dado un vector de n enteros (a_1, a_2, a_3, …, a_n) obtener la suma máxima de un conjunto de elementos consecutivos. Por ejemplo, dado el vector (-2, 11, -4, 13, -5, -2) la solución sería 20, desde a_2, hasta a_4.

Resolver el problema mediante la técnica DV:

a) Escribir el código del método principal recursivo
b) Escribir el código del método que combina las soluciones parciales, para que no supere una complejidad O(n).
c) (0,5 p.) Qué complejidad el algoritmo diseñado

Pregunta 6 (1,5 p.)

Convertir la implementación del algoritmo ordenación por mezcla o mergesort DV recursiva a una implementación que utilice hilos para ejecutar cada una de las llamadas de forma paralela.

```
public class MezclaParalelo
{
	static int []v;

	public static void mezcla(int[]v) {
		mezclarec (v,0,v.length-1);
	}

	private static void mezclarec(int[] v, int iz,int de) {
		if (de>iz)
		{
			int m=(iz+de)/2;
			mezclarec(v,iz,m);
			mezclarec(v,m+1,de);
			combina(v,iz,m,m+1,de);
		}
	}

	public static void main (String arg [] ) {
		int n=10;
		v = generarVector(n);
		mezcla(v);
		imprimirVector(v);
	}
...
}
```