Escuela de Ingeniería Informática
Universidad de Oviedo

Algoritmia
2015-2016

Control 1 – 29 de febrero de 2016

Apellidos, nombre _____ NIF: _____

Pregunta 1 (1 p.)

Responde a las siguientes preguntas:

a) (0,5p.) Si la complejidad de un algoritmo es $O(3^n)$, y dicho algoritmo toma 10 segundos para n=10, calcula el tiempo que tardará para n=14.

b) (0,5p.) Considere ahora un algoritmo con complejidad $O(\log_2 n)$. Si para t = 5 segundos el método pudiera resolver un problema con un tamaño de n = 2, ¿cuál podría ser el tamaño del problema si dispusiéramos de un tiempo de 50 segundos?

Pregunta 2 (2 p.)

Indica la complejidad temporal de los siguientes fragmentos de código:

a)
```
public void method1(int n) {
        int t = 200;
        for (int i = 2*n; i>=0; i -= 3) {
                for (int j = i; j <= n*n*n; j*=4) {
                        System.out.println("Hello");
                        t++;
                        for (int k=0; k<j; k += 2) {
                                System.out.println(t);
                        }
                }
        }
  }
```

b)
```
public void method2(int n, int p) {

        if (n < 0)
                System.out.println("Bye");
        else {
                int sum = 0;
                method2(n/2, p);
                for (int i = 0; i<n; i++) {
                        for (int j = 0; j<n; j++) {
                                sum++;
                        }
                }
                System.out.println("sum:" + sum);
        }
}
```

c)
```
public void method3(int n) {
        for (int i=0; i<=n; i++) {
                method3(n/2);
        }
}
```

d)
```
public int method4(int n) {

        if (n <= 0)
                return 0;
        else if (n==1)
                return 1;
        else return method4(n-1) + method4(n-2);
}
```

Pregunta 3 (3 p.)

Considerando la siguiente secuencia de números: 3, 5, 1, 6, 9, 2, 7, 8, 4, ordénalos utilizando los métodos indicados a continuación e indica claramente los movimientos de números que realizas paso a paso:

a) Inserción directa.
b) Rápido utilizando como pivote la mediana a tres o elemento central.
c) ¿Cuáles son las complejidades temporales de los métodos anteriores en los casos mejor y peor? ¿cuándo se dan?

Pregunta 4 (2 p.)

En la paralelización de un algoritmo DV conseguimos que la ejecución secuencial de las llamadas recursivas se realice ahora en distintos núcleos / procesadores.

a) (0,5 p.) Indica si se ganaría tiempo si paralelizamos la búsqueda binaria (justifica la respuesta).
b) (1,5 p.) Convertir la implementación del algoritmo ordenación quicksort DV recursivo a una implementación que utilice hilos para ejecutar cada una de las llamadas de forma paralela.

```
public class Rapido
{
    static int []v;     // vector sobre el que trabajamos

    public static void main (String arg [] )
    {
        int n= 10000;

        v = new int [n];

        Vector.aleatorio (v);
        Vector.mostrar (v); // antes de ordenar

        rapido(v,0,n-1);

        Vector.mostrar (v); // ordenado

    } // fin de main

    public static void quicksort (int[] v, int iz, int de)
    {
        int m;
        if (de>iz)
        {
            m=particion(v,iz,de);
            quicksort(v,iz,m-1);
            quicksort(v,m+1,de);
         }
     }
}
```

Pregunta 5 (2 p.)

Queremos diseñar un algoritmo de búsqueda "ternaria". Partiendo de un vector ordenado, primero compara con el elemento en posición n/3 del vector, si éste es menor que el elemento x a buscar entonces compara con el elemento en posición 2n/3, y si no coincide con x busca recursivamente en el correspondiente subvector de tamaño 1/3 del original. Este algoritmo devolverá la posición del elemento x buscado y -1 si no existe.

a) (0,5 p.) ¿Cuál es la altura máxima del árbol de llamadas?
b) (0,5 p.) ¿Conseguimos así un algoritmo más eficiente que el de búsqueda binaria?
c) Escribir el código Java del método que implemente este algoritmo.