Escuela de Ingeniería Informática
Universidad de Oviedo

Algoritmia
2014-2015

Examen final – Conv. Extraordinaria - 25 de junio de 2015

Apellidos, nombre _____ NIF: _____

Pregunta 1 (1 p.)

Responde a las siguientes preguntas:

a) (0,5 puntos) Si la complejidad de un algoritmo es $O(3^n)$, y dicho algoritmo toma 2 segundos para n=2, calcula el tiempo que tardará para n=6.

b) (0,5 puntos) Considere de nuevo un algoritmo con complejidad $O(3^n)$. Si para t = 2 segundos el método pudiera resolver un problema con un tamaño de n = 100, ¿cuál podría ser el tamaño del problema si dispusiéramos de un tiempo de 54 segundos?

Pregunta 2 (1 p.)

Indica la complejidad temporal de los siguientes fragmentos de código:

a)
```
public void method2(int n) {
        if (n <= 0) System.out.println("Hello");
        else {
              method2(n/2);
              method2(n/2);
              for (int i = 3; i <= n; i++)
                    for (int j = n-2; j >= 0; j++) {
                          System.out.println(i);
                          System.out.println(j);
                    }
              method2(n/2);
              method2(n/2);
        }
}
```

b)
```
public static void foo(int n, int m)
{
   int i = m;
   while (i > 100)
      i = i / 3;

   for (int k = i;  k >= 0; k--)
   {
        For (int j = 1; j < n; j *= 2)
              System.out.print(k + "/t" + j);
        System.out.println();
   }
}
```

Pregunta 3 (2 p.)

Tenemos un laberinto representado por una matriz cuadrada (n x n), de enteros donde 0 (camino) significa que se puede pasar y 1 (muro) que no se puede pasar.

El punto de *inicio* en el laberinto estará situado siempre en una de las filas de la primera columna y estará representado por una coordenada (Ini_x, Ini_y); y el punto de destino estará situado en la última columna en una de sus filas y estará representado por (des_x, des_y).

Los movimientos posibles son: arriba, abajo, izquierda y derecha. Debemos marcar el camino que se sigue desde el inicio al destino con el número 2 en cada casilla. Al final debe aparecer si se encontró solución o no, y si se encuentra mostrar el número de pasos realizados desde origen a destino. Completar el programa en Java, que implementa mediante backtracking la solución al problema.

```
public class LaberintoUna {
    static int n;                //tamaño del laberinto (n*n)
    static int[][] lab;          //representación de caminos y muros
    static boolean haySolucion;      //se encontró una solución
       // arriba, abajo, izquierda, derecha
    static int[] movx= {0,  0,  -1, 1};     // desplazamientos x
    static int[] movy= {-1, 1,  0,  0};     // desplazamientos y

    static int inix;             //coordenada x de la posición inicial
    static int iniy;             //coordenada y de la posición inicial

    static int desx;             //coordenada x de la posici□estino
    static int desy;             //coordenada y de la posici□estino

    static void backtracking(____________________________________) {
        if (____________________________________) {
             ____________________________________
            ____________________________________
            ____________________________________

        }
        else
        {
            for (____________________________________)  {
                int u= x+movx[k];
                int v= y+movy[k];

                if (____________________________________
                        ____________________________________)  {
                    ____________________________________
                    backtracking(____________________________________);
                    ____________________________________
                }
            }
        }
    }

    public static void main(String arg[]) {
      …
        ____________________________________
        ____________________________________

        backtracking(____________________________________);
        if (!haySolucion) System.out.println("NO HAY SOLUCIÓN");     }
```

Pregunta 4 (2 p.)

Teniendo en cuenta la siguiente secuencia de números: 5, 2, 1, 6, 9, 8, 7.

a) (0,75 puntos) Ordénalos por el algoritmo de ordenación llamado Rápido utilizando como pivote el **elemento central**. Indica la traza para ver la solución paso por paso.
b) (0,25 puntos) Justifica y explica la complejidad del algoritmo que has aplicado para ordenar los números indicados.
c) (0,75 puntos) Ordénalos por el algoritmo de ordenación llamado Inserción directa. Indica la traza para ver la solución paso por paso.
d) (0,25 puntos) Justifica y explica la complejidad del algoritmo anterior para el caso mejor, peor y medio.

Pregunta 5 (2 p.)

Sea el esquema general de la técnica de ramificación y poda:

```
public void ramificaYPoda(Nodo nodoRaiz)
{
    cola.insertar(nodoRaiz);
    cotaPoda = nodoRaiz.valorInicialPoda();
    while (!cola.vacia() && cola.estimacionMejor()<cotaPoda) {
        Nodo actual = cola.sacarMejorNodo();

        ArrayList<Nodo> hijos = actual.expandir();
        for (Nodo estadoHijo : hijos) {
            if (estadoHijo.getHeuristico() < cotaPoda)
                if (estadoHijo.solucion()) {
                    mejorSolucion = estadoHijo;
                    cotaPoda= estadoHijo.getHeuristico();
                    }
                }
                else
                    cola.insertar(estadoHijo);
        }
    } //while
}
```

(a)

Explicar (respuesta corta) qué ocurre en términos de funcionamiento y tiempo de ejecución si realizamos los siguientes cambios en este esquema:

a) Qué modificaciones habría que realizar para que este esquema buscase la primera solución (tacha lo que sobre y escribe lo que falta sobre el código anterior).
b) Si la clase cola guardase los elementos en una pila LIFO en vez de una cola de prioridad.
c) Si el método `getHeuristico()` devolviera un valor constante para todos los estados del problema.
d) Si eliminamos el fragmento de código marcado como (a)

Pregunta 6 (2 p.)

Supongamos que se desea resolver el siguiente problema: a partir de la tabla de alimentos, que se proporciona a continuación, elaborar un menú en el que tomemos la mayor cantidad de calorías posible, sin gastar más de una cantidad dada. Cada alimento escogido puede aparecer en el menú en una cantidad máxima de 100 gr.; pero podría aparecer en una cantidad menor.

Alimento	Kilocalorías (100gr.)	Precio céntimos de € (100gr.)
Sopa	336	14
Lentejas	325	12
Arroz	362	14
Aceite de oliva	900	34
Filete ternera	92	90
Manzana	45	6
Leche	63	5
Pan	270	12
Huevos	280	17

a) (0,75 p.) Plantear un heurístico para resolver este problema mediante un algoritmo voraz.

b) (1,25 p.) Dar una solución al problema, con la tabla dada, aplicando el heurístico, suponiendo que como mucho podemos gastar 1 €.