Examen de Teoría de la Programación

E. U. ING. TEC. EN INFORMÁTICA DE OVIEDO

Final Febrero – Curso 2005-2006 14 de febrero de 2006

DNI _____ Nombre _____ Apellidos _____

Titulación: ☐ Gestión ☐ Sistemas

Grupo al que asistes en teoría: _____

3. (1,25 puntos) La multiplicación de matrices cuadradas se puede resolver siguiendo la técnica divide y vencerás.

a) **En esta técnica, se diferencian tres pasos: descomposición del problema, resolución de los problemas elementales, combinación de los resultados para obtener la solución al problema inicial. Especificar que hay que hacer en cada uno de estos tres pasos en la resolución de la multiplicación de matrices cuadradas.**
b) **Cuál es la complejidad de esta solución por Divide y Vencerás.**

(Problema expuesto en el grupo A)

4. (1,25 puntos) El problema del *play-off* de baloncesto consiste en dos equipos *A* y *B* que disputan una eliminatoria de la fase final de la liga de baloncesto. Jugarán 2*n*-1 partidos y el ganador será el primer equipo que consiga *n* victorias. El equipo *A* tiene una probabilidad constante *p* de ganar un partido, mientras que *B* tiene una probabilidad *q* de ganar (*q*= 1 – *p*). Queremos conocer la probabilidad de que el equipo *A* gane el *play-off*, y que esto pueda calcularse tanto antes de empezar la eliminatoria, como una vez jugados varios partidos.

La función que proporciona la solución al problema es la siguiente:

$$P(i,j)=\begin{cases} 1 \quad si \quad i=0 \wedge j>0 \\ 0 \quad si \quad i>0 \wedge j=0 \\ p*P(i-1,j)+q*P(i,j-1).si.i>0 \wedge j>0 \end{cases}$$

Donde *P(i,j)* se define, como la probabilidad de que *A* gane la eliminatoria cuando le quedan por ganar *i* partidos a *A* y *j* partidos a *B*.

Utilizaremos la técnica de Programación Dinámica para obtener la solución al problema. Suponemos que para ganar la eliminatoria un equipo debe de ganar tres partidos.

a) **Justificar, en base a la complejidad temporal, por qué es conveniente resolver este problema por *Programación Dinámica* en vez de por *Divide y Vencerás*.**
b) **Representar gráficamente el array necesario para implementar el algoritmo, señalando el significado de filas y columnas (no hace falta rellenarlo). Tener en cuenta que queremos calcular la probabilidad antes de comenzar la eliminatoria.**
c) **Representar gráficamente que celdas son necesarias para calcular el valor de la celda (2,1).**
d) **Explicar que valores podemos rellenar de forma directa en la tabla.**
e) **Plantear el orden correcto para rellenar la tabla.**

5. (1,75 puntos) El día de San Valentín, Oscar quiere regalarle a su novia el CD de música perfecto. Para ello ha estado recopilando durante los 15 años que llevan juntos, todas las canciones que le gustan a su novia y les ha asignado una puntuación, con lo que dispone de una lista de ¡1300 canciones! Además, Oscar dispone del tamaño del archivo correspondiente a cada canción. Sólo quiere realizar un único CD y, lógicamente, no caben todas las canciones, para crear el mejor disco debe maximizar la puntuación total de las canciones que contiene, teniendo en cuenta que los ficheros de canciones no se pueden partir. Como Oscar tenía que estudiar para diversos exámenes de la convocatoria de febrero, lo ha dejado para última hora y debe utilizar un algoritmo que calcule las canciones que debe incluir de forma inmediata.

a) **Justificar por qué debe aplicar un algoritmo voraz para resolver este problema.**
b) **Explicar el heurístico más adecuado para este caso.**
c) **Escribir el código Java que permita obtener la solución para este problema mediante un algoritmo voraz.**

6. (1,25 puntos) El problema de la asignación de tareas consiste en asignar *n* tareas a *n* agentes, de tal forma que cada agente realiza una única tarea y cada tarea sólo puede ser realizada por un único agente. A cada agente le cuesta realizar más unas tareas que otras, luego para cada pareja *(agente, tarea)* tendremos asociado un coste. Esto se representa en una matriz. El objetivo del problema es buscar la asignación que minimice el coste total de realizar todas las tareas.

a) Completar el código Java para dar solución a este problema con la técnica de Backtracking (escribirlo en los recuadros preparados a tal efecto).

```
public class Main
{
    private int costes[][],  // almacena lo que de cuesta a cada agente realizar una tarea
            almacenEstados[],// estado actual del problema
            mejorSolucion[]; // contiene la mejor solución en cada momento
    private boolean tareasAsignadas[];// tareas ya asignadas (las que están a true)

    private int n, // Tamaño del problema.
            coste, // Almacena el coste acumulado en el cálculo de una solución.
            mejorCoste; // El mejor coste hasta el momento de una solución.

    […]

    public void ensaya(_____________________________)
    {
        for (_____________________________________________________________)
        {
            if (_____________________________________________________)
            {
                almacenEstados[agente] = tarea;
                coste += costes[tarea][agente];
                tareasAsignadas[tarea] = true;


                if (_____________________________________)     // no es solución
                {
                    ensaya(_____________________________);
                 }
                else         // es una solución posible
                    if (coste < mejorCoste)
                    {
                        mejorCoste = coste;
                        System.arraycopy(almacenEstados, 0, mejorSolucion, 0, n);
                    }

                coste -= costes[tarea][agente];
                tareasAsignadas[tarea] = false;
            }
        }
     } // Fin método
```

(Problema expuesto en el grupo B)

7. (1,5 puntos) Un campo está dividido en *m* secciones horizontales (filas) por *n* secciones verticales (columnas). Los saltos del *superconejo* de una sección a otra de este campo se pueden ver como un primer movimiento de *p* secciones en dirección horizontal o vertical, seguida de un segundo movimiento de *q* secciones en dirección perpendicular a la seleccionada anteriormente. (Cuando se mueve horizontalmente lo puede hacer hacia la izquierda o hacia la derecha y cuando se mueve verticalmente lo puede hacer hacia arriba o hacia abajo).
Colocamos al *superconejo* en la sección (*x,y*) y queremos que llegue a otra sección distinta (*u,v*) realizando los saltos como se describió anteriormente. Se debe buscar el camino que conlleve la cantidad mínima de saltos para lograrlo.

a) El grupo que ha resuelto este problema ha utilizado un desarrollo en anchura puro, sin utilizar heurísticos. Explicar por qué este método es mejor que otras técnicas vistas: backtracking o devorador.
b) Escribir el esquema del recorrido en anchura que se debe emplear para encontrar la solución a este problema.

(Problema expuesto en el grupo C)