Examen. 7 de febrero de 2.006

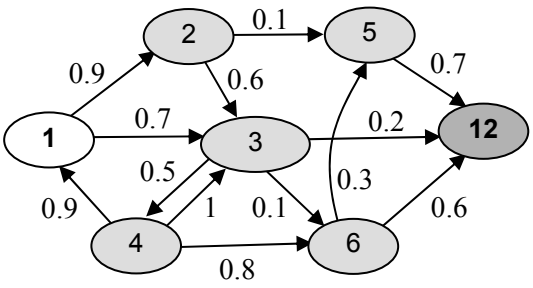
Resolver cada pregunta en una hoja distinta.
No hay que entregar esta hoja con el examen.

1. (2,5 puntos) Sobre una zona del plano se define una cuadrícula de 1000x1000 posiciones. Dentro de ella, se eligen ciertos puntos al azar y se toman medidas en pequeñas regiones contiguas de tamaño cuadrado. Dichas medidas se han de almacenar en una tabla de dispersión. Disponemos de una tabla de tamaño 2000. Se pide:
   a) Si usamos dispersión cerrada, discutir sobre cuál sería el tamaño máximo del número de muestras a tomar de forma que el comportamiento de la tabla fuera razonablemente eficiente. ¿Y si usamos dispersión abierta? Razonar brevemente las respuestas.
   b) Diseñar una buena función de dispersión para esta aplicación.
   c) Obtener el número promedio de colisiones en la inserción, con dispersión abierta, si tenemos 40375 muestras en la tabla.
   d) Diseñar una buena estrategia de redispersión en caso de utilizar dispersión cerrada.

2. (2,5 puntos) Programar una operación de unión entre dos árboles trie. Dados dos árboles trie, la operación debe añadir a uno de ellos todas las palabras que aparecen en el otro. Se deben utilizar las operaciones genéricas sobre nodos trie: **Consulta (n: trie, c: carácter): trie** (devuelve el hijo del nodo trie **n** para el carácter **c**, o NULO si no existe), **Inserta (n: trie, c: carácter, m: trie)** (añadir al nodo trie **n** el hijo **m** asociado al carácter **c**), **NuevoTrie: trie** (devuelve un nuevo nodo vacío) y un iterador del tipo **para cada carácter c hijo del nodo n hacer**.

3. (3 puntos) ¡El mundial se acerca y la selección española te necesita! Tenemos un esquema del equipo con 12 posiciones, una por cada jugador (el portero es el número 1) más una para la portería contraria (que es el número 12). Para cada par de jugadores (**a**, **b**), tenemos la probabilidad, **P[a, b]** $\in$ (0, 1), de que un pase desde **a** hasta **b** salga bien, es decir, que no lo intercepte el equipo contrario. La matriz no es simétrica, y **P[a, 12]** indica la probabilidad de que **a** marque un gol al chutar. A partir de esas probabilidades básicas, se puede calcular la probabilidad de una secuencia de pases, mediante el producto. La probabilidad de que la secuencia de pases **a** → **b** → **c** salga bien será: **P[a, b]*P[b, c]**, y así sucesivamente. Una estrategia de juego es una secuencia de pases que empieza en nuestro portero y acaba en gol en la portería contraria.

   Escribir un algoritmo eficiente que encuentre la estrategia de juego óptima, es decir, la secuencia de pases entre 1 y 12 que maximice la probabilidad de salir bien. Aplicar el algoritmo al ejemplo de abajo, indicando la estrategia óptima y la probabilidad asociada.



Esquema del equipo con probabilidades **P[a, b]** de que los pases salgan bien. Si no aparece una flecha es que la probabilidad es 0

Nota: La pregunta 3 no deben hacerla los alumnos que lograron el Reto del Viajante.

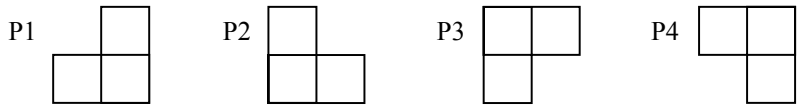
4. (2 puntos) Partiendo de la especificación del tipo **Natural** vista en clase, escribir una especificación formal del tipo **Matriz_de_naturales** con las siguientes operaciones: **crear** (crea una matriz, inicializada a 0, de tamaño UxV; este tamaño es fijo, por lo que no hace falta incluirlo en la especificación), **asignar** (sustituye el valor de una celda, dada la fila, la columna y el nuevo valor), **obtener** (obtiene el valor de una posición dada de la matriz), **sumar** (suma dos matrices de dimensión UxV), **produtoEscalar** (multiplica cada elemento de la matriz por un natural dado) y **máximoHistórico** (devuelve el máximo valor asignado en algún momento a cualquier posición de la matriz). Se puede suponer la existencia de las operaciones **sumarN** y **multiplicarN** sobre los naturales.

Nota: La pregunta 4 no deben hacerla los alumnos que tengan aprobada la práctica de especificaciones formales.

Resolver cada pregunta en una hoja distinta.
No hay que entregar esta hoja con el examen.

1. (3 puntos) Un problema se puede resolver usando divide y vencerás de dos formas distintas. En la primera forma se divide el problema de tamaño **n** en dos subproblemas de tamaño **n**/2, y la combinación tiene un coste **n**. Por la segunda forma se divide el problema en tres subproblemas de tamaño **n**/3, y la combinación tiene un coste 2**n**. En ambos métodos el tamaño del caso base es 1, y el coste de resolver el caso base y de comprobar si el problema es suficientemente pequeño y de dividir el problema también tiene coste 1.
   a) Calcular el tiempo de ejecución de los dos algoritmos, y determinar cuál de los dos es preferible en cuanto al tiempo de ejecución. ¿Y en cuanto al orden de complejidad?
   b) Estudiar la ocupación de memoria de los dos algoritmos, e indicar cuál es preferible en cuanto a ocupación de memoria.

2. (3 puntos) Tenemos un puzzle consistente en un tablero de **n**x**m** casillas y piezas de cuatro tipos distintos:



   con un determinado número de piezas de cada uno de los tipos, **NumP**[1..4]. Las piezas se deben poner en el tablero, sin solaparse, sin salirse y ocupándolo de forma completa. Programar la solución del puzzle por backtracking. Se supone que se busca sólo una solución, la cual no se puede garantizar que exista. Habrá que explicar el tipo de árbol que se utiliza, cómo se representan las soluciones, las estructuras que se utilizan en la solución, y programar las funciones, usando los esquemas vistos en clase.

3. (2 puntos) En cierto teclado, asignamos las teclas especiales a cadenas de más o menos tamaño, (por ejemplo: F1→abc, F2→ca, F3→ab, F4→bcc). Dada otra cadena más larga (por ejemplo: abccabcabb) se quiere escribirla pulsando el menor número de teclas. Se pueden usar las teclas normales o las especiales. Explicar cómo se resuelve el problema por programación dinámica: hay que dar la fórmula de recursión, el valor de los casos base, y las tablas que se usan en la solución del problema. Explicar el funcionamiento con el ejemplo dado. **Sugerencia**: estudiar la descomposición recurrente de una función **MinTeclas (i: entero): entero**, que devuelve el menor número de pulsaciones para escribir los **i** primeros caracteres de la cadena larga.

4. (2 puntos) Programar por un método de avance rápido la solución del problema anterior. Explicar su funcionamiento con el ejemplo. Justificar si el método encuentra siempre la solución óptima.

Nota: La pregunta 4 no deben hacerla los alumnos que tengan convalidada la Práctica 4, o los que tengan aprobada la asignatura de plan antiguo "Laboratorio de Programación".