Examen de Teoría de la Programación

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

Final Junio – Curso 2007-2008 — 12 de junio de 2008

DNI _____ Nombre _____ Apellidos _____

Titulación: ☐ Gestión ☐ Sistemas

3. (1,5 puntos) El problema del salto del caballo consiste en recorrer un tablero de ajedrez completo (8x8) a base de movimientos como los que realiza el caballo. Se quiere buscar al menos una forma de recorrer el tablero utilizando la técnica de backtracking (vuelta atrás). Completar el código Java para dar solución a este problema (escribirlo en los recuadros preparados a tal efecto).

```
public class Caballo
{
   private int n;                                  // lado del tablero
   private int[][] tablero;                        // representación tablero
   private final int NUM_MOVIMIENTOS= 8;           // Num movimientos posibles
   private int[] h= {1, 2, 2, 1, -1, -2, -2, -1};  // Mov horizontal
   private int[] v= {2, 1, -1, -2, -2, -1, 1, 2};  // Mov vertical

   public Caballo(int tamTablero)      { …    }     // constructor

      public boolean buscar1Solucion(int i, int x, int y)
      {
            boolean haySolucion= false;
            int a;
            int b;

            for (int k= 0; k<NUM_MOVIMIENTOS &&
!haySolucion; k++)
            {
                  a= x+h[k];
                  b= y+v[k];
                  if (a>=0 && a<n && b>=0 && b<n &&
tablero[a][b]==0)
                  {
                        tablero[a][b]= i;
                        if (i<n*n)
                        {
                              haySolucion= buscar1Solucion(i+1,a,b);
                              if (!haySolucion) tablero[a][b]= 0;
                        }
                        else
                              haySolucion= true;
                  }
            }

            return haySolucion;
      }

      public static void main(String[] args)
      {
            Caballo cab= new Caballo(8);
            cab.situar(0, 0, 1);// caballo en la posición 0,0 del tablero, movi. 1
            if (cab.buscar1Solucion(2, 0, 0))
                  cab.mostrarTablero();
            else
                  System.out.println("NO hay solución");

      }

}
```

4. (1 punto) Sea la siguiente función que obtiene el valor máximo de los *n* primeros elementos de un vector, especificada utilizando la técnica de precondición - postcondición:

- **{Q ≡ 1≤n ≤ 1000}**
- **Tipo vector= array [1..1000] de entero;**
- **fun *maximoValor*(*a:vector;n:entero*) dev (*x:entero*)**
- ***{R* ≡ (∀i ∈ {1..n}.x≥a[i]) ∧ (∃j ∈{1..n}.x=a[j])}**

a) **Escribir en notación formal la postcondición de esta función.**
b) **Codificar en Java el esqueleto de esta función utilizando un método público: sólo cabecera y precondición, no hace falta la postcondición (no hace falta codificar la funcionalidad del método en sí).**

```
public int maximo(int[] a, int n) throws IllegalArgumentException
  {
    // Precondición
    if (n<0 || n>1000)
       throw new IllegalArgumentException();
```

…

5. (1,5 puntos) Plantear el algoritmo Divide y Vencerás que resuelva un determinante por adjuntos. Sabiendo que la fórmula es:

$$|A| = \sum_{i=1}^{n} a_{1i} A_{1i}$$

Donde A_{ij} es el *adjunto* de a_{ij} , y es igual a $(-1)^{i+j}$ $|M_{ij}|$.

a) **Escribir el pseudocódigo del método para resolverlo.**

```
private long calcularDeterminante(int orden)
{
       long sumatorio= 0;
       if (orden==1)
              return (long)getElemento(0,0);
       else
       {
              for (int i= 0; i<orden; i++)
              {
                     Matriz m= calcularComplementaria(0,i);
                     if ((i+1)%2!=0)  // si es impar
                            sumatorio+= matriz[0][i] *
       m.calcularDeterminante(orden-1);
                     else
                            sumatorio-= matriz[0][i] *
       m.calcularDeterminante(orden-1);
              }
       }
       return sumatorio;
}
```

b) **¿De qué orden es la complejidad de este algoritmo?**

$O(n! \cdot n^2) \rightarrow O(n!)$

5. (1,5 puntos) Pau Gasol con los Lakers es el primer jugador español que juega una final de la NBA. La final se juega a un máximo de 7 partidos y a día de hoy los Celtics vencen a los Lakers por 2 – 1. Preguntado a un experto la probabilidad de que los Lakers ganen cada uno de los siguientes partidos es del 60%.
Dada la función P(i,j) que se define como la probabilidad de que el equipo A gane la eliminatoria cuando le quedan por ganar i partidos frente a los j partidos que le quedan por ganar al equipo B de la siguiente forma:

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

Donde *p* es la probabilidad de que A gane un partido y *q* la probabilidad de que gane B.

a) Representar gráficamente el array necesario para implementar el algoritmo, señalando el significado de filas y columnas. Minimizar el espacio del array para hacer el cálculo del apartado b).
b) Calcular la probabilidad que tiene Pau Gasol de ponerse el anillo de campeón partiendo de la situación actual. Recuadrar el resultado final para dejar claro cuál es.
c) Calcular la complejidad de la función con programación dinámica.

a) y b)

A los Celtics les quedan por ganar 2 partidos, mientras que a los Lakers les quedan 3.
Por tanto, lo que necesitamos calcular es P(3,2)

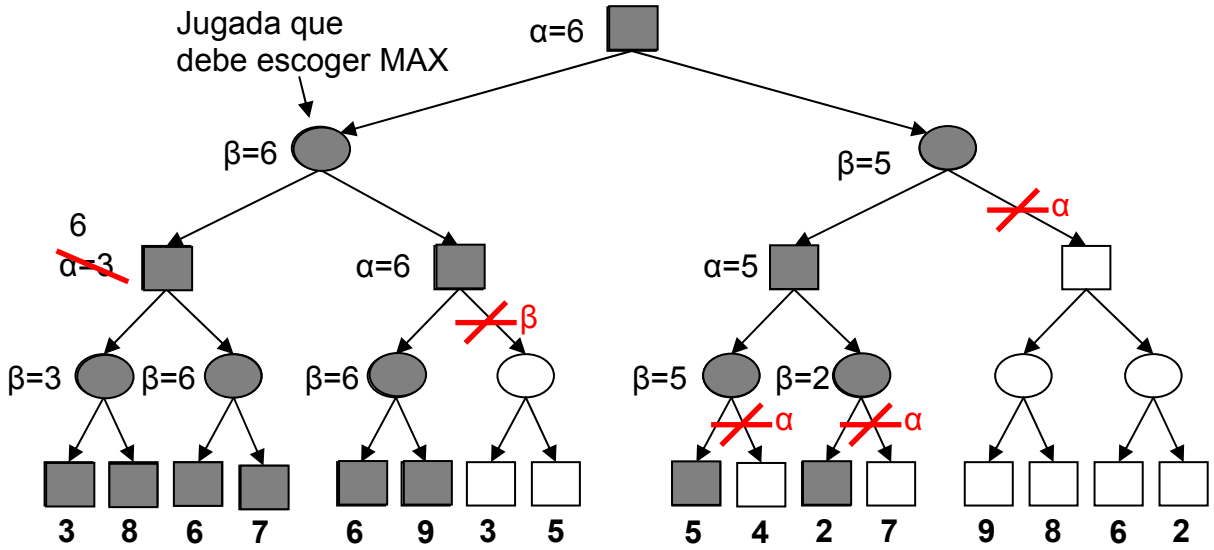
i \ j	0	1	2	
0		1		1
1	0	0.6	1·0,6+0,6·0,4=	0,84
2	0	0,36	0,84·0,6+0,36·0,4=	0,648
3	0	0,216	0,648·0,6+0,216·0,4=	0,4752

La probabilidad actual para Pau de enfundarse el anillo es 47,52%.

c) La complejidad es la de rellenar la tabla de dos dimensiones $O(n^2)$

(Desgraciadamente para Pau Gasol los Lakers han perdido en el 6º partido)

7. (1,5 puntos) Desarrollar la poda α-β para conocer que jugada debe realizar el jugador MAX, sobre el siguiente árbol:



a) Sombrear los nodos que haya que desarrollar
b) Escribir las cotas α y β,
c) Marcar los cortes e indicar si son de tipo α o β,
d) Por último, indicar que jugada debe elegir MAX para situarse en la mejor posición posible.

Notas: El jugador que realiza el primer movimiento en el árbol es MAX. Los nodos del árbol se desarrollan de izquierda a derecha.