Examen de Teoría de la Programación

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

Final Septiembre – Curso 2008-2009 9 de septiembre de 2009

DNI _____ Nombre _____ Apellidos _____

Titulación: ☐ Gestión ☐ Sistemas

3. **(2 puntos) El problema de las n reinas consiste en colocar n reinas en un tablero de ajedrez sin que ninguna reina pueda comer a otra. Pretendemos buscar la primera solución a este problema con un tablero de 4 x 4. El problema se ha resuelto utilizando la técnica de Backtracking.**

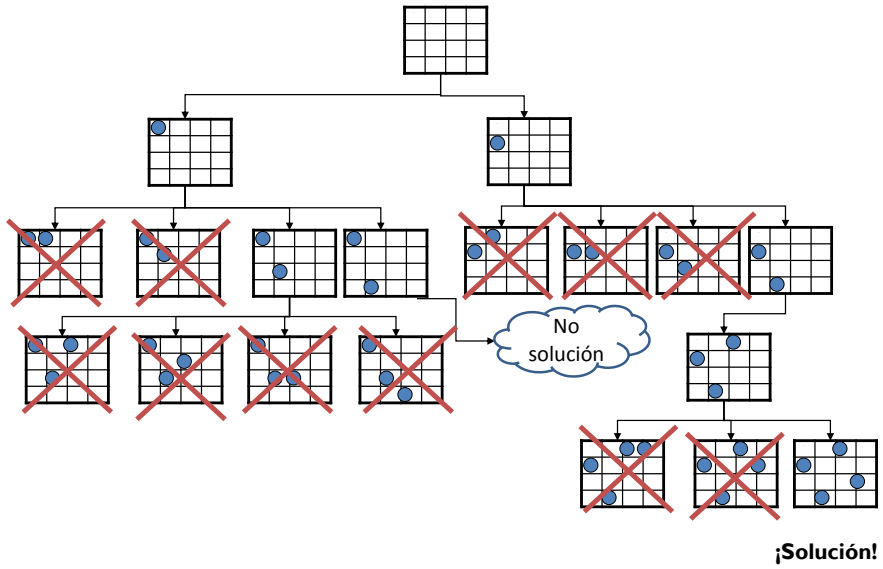
b) (1,5 puntos) Completar el código Java para dar solución a este problema (escribirlo en los recuadros preparados a tal efecto).

```
/** Clase que implementa el problema de las n reinas */
public class Reinas
{
private int n;      /* Será inicializado en el constructor con el número de reinas a colocar */
private int[] tablero; /* Tablero donde vamos almacenando la solución */
private boolean haySolucion= false; /* Indica si hemos encontrado una solución */
/* Vectores auxiliares para comprobar si un estado es válido */
private boolean[] filaLibre;
private boolean[] diagonal1Libre;  // Para la diagonal /
private boolean[] diagonal2Libre;  // Para la diagonal \

public void buscar1SolucionReinas(int col)
{
        // Bucle que permite probar todas las posibilidades
        for (int fila= 0; fila<n && !haySolucion; fila++)
        {
                // Comprueba si el estado es válido
                if (filaLibre[fila] && diagonal1Libre[col+fila]
                        && diagonal2Libre[col-fila+n])
                {
                        // Añadimos nueva reina a la solución
                        tablero[col]= fila;
                        // Actualizamos vectores para comprobar si estado es válido
                        filaLibre[fila]= false;
                        diagonal1Libre[col+fila]= false;
                        diagonal2Libre[col-fila+n]= false;
                        // si colocamos la última reina hemos llegado a una solucion
                        if (col==n-1)
                                haySolucion= true;
                        else
                        {
                                // Llamada de recursiva para explorar en profundidad
                                buscarSolucion(col+1);
                                if (!haySolucion)
                                {
                                        // Borramos vectores comprobar estado válido
                                        filaLibre[fila]= true;
                                        diagonal1Libre[col+fila]= true;
                                        diagonal2Libre[col-fila+n]= true;
                                }
                        }
                }
        }
}

}
```

a) (0,5 puntos) Dibujar el árbol completo que genera la técnica de backtracking para este problema.



4. (0,5 puntos) Sea la función:
Fun raiz_cuadrada(a: entero) dev (r: real)
que realiza la raíz cuadrada el número *a* y lo devuelve en *r*.

a) Realizar la especificación formal de la función con la técnica pre/post.

a) Especificación formal de la función raíz cuadrada.

Se debe emplear notación formal.
{Q ≡ a≥0}
fun raiz_cuadrada (a: entero) **dev** (r: real)
{R ≡ (a=r*r) ∧ (r≥0)}

5. (1,5 puntos) Sea la función de Fibonacci:

$$f(n)=\begin{cases} 0 \text{ si } n=0 \\ 1 \text{ si } n=1 \\ f(n-1)+f(n-2) \text{ si } n>1 \end{cases}$$

a) Escribir el código Java para resolver esta función por la técnica de divide y vencerás.
b) Qué complejidad presenta esta función. Proporcionar al menos una aproximación y justificar la respuesta.
c) Escribir el código Java para resolver esta función por la técnica de programación dinámica.

Ver cuaderno didáctico:
- Solución DV y programación dinámica: páginas 51 y siguientes.
- Complejidad páginas 25 y 26.

6. (1,5 puntos) Después de la derrota de España por una canasta en el Eurobasket 07 se realizó una petición popular para que la final se jugase estilo play-off a un máximo de 5 partidos. ¿Qué hubiese pasado si esto se hubiera realizado así en la final del Eurobasket de España? ¿Qué probabilidad tendría entonces de ganar España?

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.

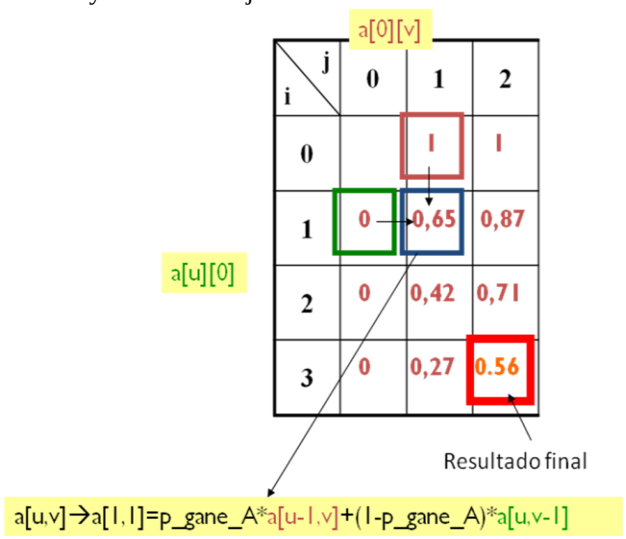
a) Representar gráficamente el array necesario para implementar el algoritmo mediante programación dinámica, señalando el significado de filas y columnas. Minimizar el espacio del array para hacer el cálculo del apartado b).

Teniendo en cuenta que la función es P(i,j) tendremos un array de dos dimensiones.
Nos indican que tenemos que crear un array del mínimo tamaño posible. Teniendo en cuenta que Rusia (equipo B) ha ganado ya el primer partido (apartado b) a este equipo le quedan 2 partidos para ser campeón, mientras que a España le quedan 3 partidos por ganar, por tanto la función concreta a resolver es P(3,2), la idea es tener las celdas justas para resolver esta función concreta.
Otra cuestión es si eliminamos la primera fila y columna que contienen los valores directos. Admito las dos soluciones tanto con valores triviales como sin ellos ya que no se utilizan demasiadas celdas.

i \ j	1	2
1		
2		
3		

b) Calcular la probabilidad que hubiese tenido España de ganar el Eurobasket 07, partiendo de que se ha jugado el primer partido y lo ha perdido España y la probabilidad de que España ganase cada uno de los siguientes partidos era del 65%. Resolverlo rellenando el array anterior. Dejar claro el resultado final recuadrándolo.

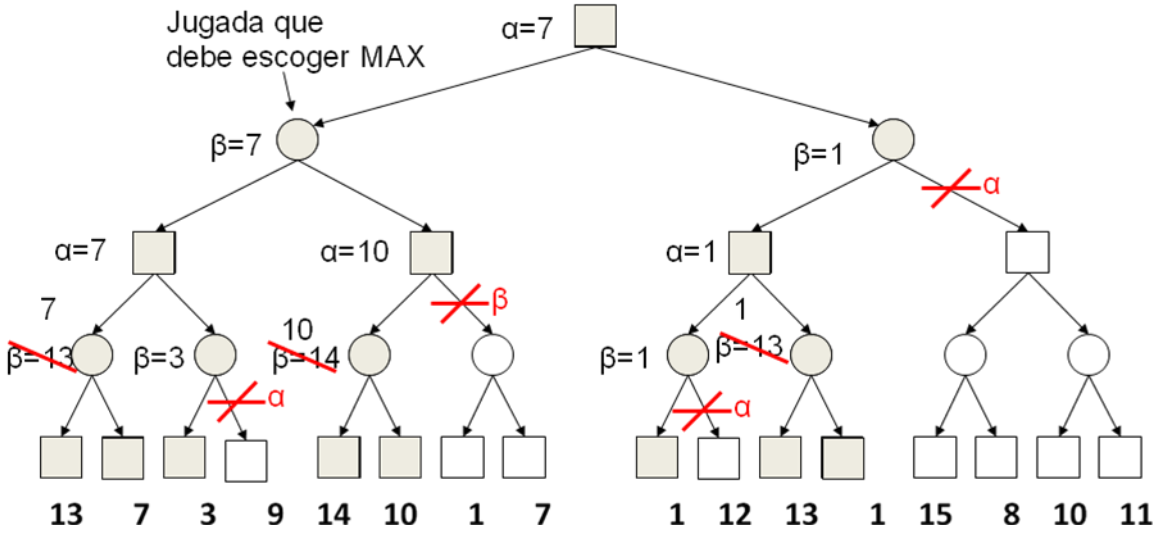
Volvemos a incluir los valores directos para mayor claridad.
Como ya dijimos antes tenemos que calcular P(3,2) y la probabilidad de que gane A (p_gane_A) es 0,65, simplemente se trata de aplicar la parte recursiva de la función sobre el array para ir calculando de izquierda a derecha y de arriba abajo los valores no triviales.



c) Calcular la complejidad de la función con programación dinámica.

La función se codifica con dos bucles anidados así que es **$O(n^2)$**.

6. (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.