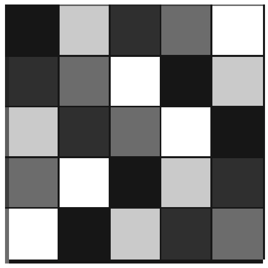
FEBRERO 2S 2014

Problema (4 puntos).

Se considera un tablero *n x n* y un conjunto de *n* colores. Cuando a cada casilla se le asigna un color de los *n* disponibles de tal manera que no se repiten colores en ninguna fila ni en ninguna columna, el tablero se conoce como *cuadrado latino*. La siguiente figura muestra un ejemplo para *n = 5*:



Se pide diseñar un algoritmo que dados *n* colores presente todos los cuadrados latinos para un tablero *nxn*.

Se pide:

1. Elección del esquema más apropiado, el esquema general y explicación de su aplicación al problema (0,5 puntos).
2. Descripción de las estructuras de datos necesarias (0,5 puntos solo si el punto 1 es correcto).
3. Algoritmo completo a partir del refinamiento del esquema general (2,5 puntos solo si el punto 1 es correcto).
4. Estudio del coste del algoritmo desarrollado (0,5 puntos solo si el punto 1 es correcto).

Elección del esquema

El esquema más apropiado es el de **vuelta atrás.** El esquema general se encuentra en otro PDF

Estructuras de datos

Una posibilidad es dar las soluciones en forma de tuplas de n^2 elementos: $(x_1, x_2, \dots, x_{n^2})$ donde x_i es el color asignado a la casilla i, siendo el conjunto de colores $\{1, \dots, n\}$. Para ello hay que numerar adecuadamente las casillas del tablero. Como no podemos usar dos veces el mismo color en la misma fila o columna, necesitamos marcadores para saber si un color aparece ya en una fila o columna. Es decir, necesitamos dos matrices de booleanos $F[1 \dots n, 1 \dots n]$ y $C[1 \dots n, 1 \dots n]$, tales que:

- $F[i, color] \Leftrightarrow$ en la fila i hemos usado el color $color$
- $C[j, color] \Leftrightarrow$ en la columna j hemos usado el color $color$

Algoritmo completo a partir del refinamiento del esquema general

El algoritmo recursivo que imprime todas las soluciones es el siguiente:

```
fun latino_va(sol: vector [1...n²] de colores, k: entero, F, C: MatrizB)
  fila ← fila(k); columna ← columna(k)
  para color = 1 hasta n hacer
    si ¬ F[fila, color] ∧ ¬ C[columna, color] entonces
      sol[k] ← color
      F[fila, color] ← cierto; C[columna, color] ← cierto {marcar}
      si k = n² entonces imprimir(sol)
      sino latino_va(sol, k + 1, F, C)
      fsi
      F[fila, color] ← falso; C[columna, color] ← falso {desmarcar}
    fsi
  fpara
ffun
```

```
fun fila(k, n: entero): entero
  dev ((k − 1) div n) + 1
ffun
```

```
fun columna(k, n: entero): entero
  dev ((k − 1) mod n) + 1
ffun
```

El algoritmo principal con la inicialización correspondiente es:

```
tipo MatrizB = matriz [1 ... n, 1 ... n] de booleano
fun latino1(n: natural)
  var
    sol: vector [1 ... n²] de entero
    F[1..n, 1..n], C[1..n, 1..n]: MatrizB
  fvar
  F[1..n, 1..n] ← [falso]; C[1..n, 1..n] ← [falso]
  latino_va(sol, 1, F, C)
ffun
```

Coste

El árbol de exploración tiene n^2 niveles y n hijos por nodo. Luego una cota para el coste es $\boldsymbol{O\left(n^{n^2}\right)}$.