Prácticas de tablero – Sesión 4

EJERCICIO 1: Camino Simple Menor Coste

Se pide calcular el camino simple de **menor** coste entre un nodo origen y un nodo destino (origen <> destino) en un grafo dirigido de pesos positivos.

De los diferentes algoritmos existentes que resuelven este interesante problema (ver en Wikipedia "problema del camino más corto"), vamos aquí a ver dos: backtracking y Floyd (que pertenece a la técnica de "programación dinámica").

Backtracking: Estudiar y ejecutar las clases CaminoMejor.java y CaminoMejorPoda.java (son algoritmos no polinómicos (factoriales) o intratables).

Floyd: Estudiar y ejecutar las clases Floyd.java y FloydTiempos.java.

Traza de ejemplo:

```
n=5;
for (int i=0;i<w.length;i++)
for (int j=0;j<w.length;j++) w[i][j]=10000000;
w[0][1]=10;w[0][3]=30;w[0][4]=100;w[1][2]=50;
w[2][4]=10;w[3][2]=20;w[3][4]=60;
```

Resultados:

```
p.e. java s4.Floyd 0 4
COSTE MIN=60 ** CAMINO=NODO0–NODO3–NODO2–NODO4

p.e. java Floyd 1 2
COSTE MIN=50 ** CAMINO=NODO1–NODO2

p.e. java s4.Floyd 3 0
NO HAY CAMINO
```

La clase **s4.FloydTiempos** simula el crecimiento del problema (n = número de nodos el grafo) y mide tiempos, siendo claramente cúbicos.

EJERCICIO 2: Camino Simple Mayor Coste

El problema de calcular **el camino simple de mayor coste en un grafo no tiene** soluciones de complejidad temporal polinómica, por lo tanto es un problema NP.

Luego el «backtracking» tiene sentido, aunque lleve a tiempos intratables.

La clase CaminoPeor.java calcula ese camino de mayor coste. Estudiarla y ejecutarla (en este caso no podemos emplear la poda, al estar calculando un máximo).