



**UNIVERSIDAD  
DE GRANADA**

**E.T.S. DE INGENIERÍAS INFORMÁTICA y DE  
TELECOMUNICACIÓN**

**Departamento de Ciencias de la  
Computación e Inteligencia Artificial**

# **Algorítmica**

## **Guión de Prácticas**

### **Práctica 4: Programación Dinámica**

Curso 2018-2019

Doble Grado en Ingeniería Informática y Administración y Dirección de  
Empresas

# 1 Normativa

El objetivo de esta práctica es la puesta en práctica los conocimientos relacionados con la técnica de Programación Dinámica adquiridos en clase de teoría. Para ello cada equipo deberá realizar lo siguiente:

1. Seleccionar uno de los dos problemas de dificultad similar que se proponen.
2. La generación de datos para cada problema.

Cada equipo tendrá que entregar:

1. Una memoria en la que se describa la solución propuesta para el problema elegido. No puede obviarse la descripción de la ecuación recurrente definida para la resolución del problema por la técnica de Programación Dinámica.
2. En la memoria se debe añadir el pseudocódigo del algoritmo propuesto, y varios escenarios de ejecución.
3. El código elaborado para resolver ambos problemas. El código se debe entregar indicando cómo debe ser compilado. Se aconseja entregar el Makefile.

## 2 Viajante de comercio

En su formulación más sencilla, el problema del viajante de comercio (TSP, por sus siglas en inglés *Traveling Salesman Problem*) se define como sigue: dado un conjunto de ciudades y una matriz con las distancias entre todas ellas, un viajante debe recorrer todas las ciudades exactamente una vez, regresando al punto de partida, de forma tal que la distancia recorrida sea mínima. Mas formalmente, dado un grafo  $G$ , conexo y ponderado, se trata de hallar el ciclo hamiltoniano de mínimo peso de ese grafo.

### Tareas

1. Diseñar y desarrollar un algoritmo basado en Programación Dinámica para el problema del Viajante de Comercio.
2. Comparar la solución y eficiencia del algoritmos propuesto por Programación Dinámica con los elaborados siguiendo una estrategia voraz.

**Datos de prueba.** Los datos que se tienen que usar para evaluar esta tarea están disponibles en PRADO. Los datos se han obtenido de TSPLIB.<sup>1</sup> Cada problema está descrito por dos ficheros:

**Fichero .tsp.** Fichero en el que se indica tanto el número de ciudades (**DIMENSION**) como sus coordenadas  $x$  e  $y$ . Los datos de cada ciudad están en una nueva línea.

**Fichero .opt.tour.** Fichero en el se muestra el resultado que debe devolver el algoritmo.

Las distancias entre las ciudades se debe calcular mediante la distancia euclídea, es decir, dadas las coordenadas de dos ciudades  $((x_1, y_1), (x_2, y_2))$  la distancia entre ellas se calcularía de la siguiente manera:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (1)$$

La distancia  $d$  debe redondearse al entero más próximo.

**Visualización.** Se puede usar cualquier programa para la visualización de los ciudades (puntos) en el espacio. Si se usa **gnuplot**, tomando los ficheros de datos que se proporcionan se puede visualizar las ciudades ejecutando el siguiente comando:

```
gnuplot> plot "fichero.tsp" using 2:3 with points
```

Una vez que se haya obtenido la solución y se genera el fichero con las ciudades y los puntos se puede generar el recorrido del viajante de comercio usando **gnuplot** de la siguiente manera:

```
gnuplot> plot "ficheroreord.tsp" using 2:3 with lines
```

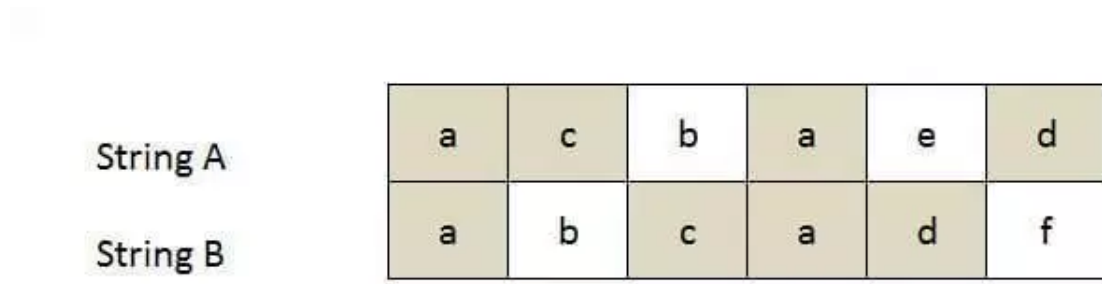
---

<sup>1</sup><https://wwwproxy.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp/>

### 3 Subsecuencia de caracteres más larga

Sean dos secuencias de caracteres  $X = (x_1, x_2, \dots, x_m)$  e  $Y = (y_1, y_2, \dots, y_n)$ , de longitudes  $m$  y  $n$  respectivamente. El problema consiste en encontrar la máxima subsecuencia de caracteres común que aparecen en ambas cadenas de izquierda a derecha (no necesariamente de forma contigua). Por ejemplo, para las cadenas  $S = \text{ABAZDC}$  y  $T = \text{BACBAD}$ , la máxima subsecuencia común tiene longitud 4 y es  $\text{ABAD}$ , siendo localizadas en  $S = \text{ABAZDC}$  y en  $T = \text{BACBAD}$  (ver Figura ??<sup>2</sup>).

Se pide: Implementar un algoritmo de programación dinámica que calcule la subsecuencia común más larga entre dos cadenas de caracteres de entrada.



String A	a	c	b	a	e	d
String B	a	b	c	a	d	f

Figure 1: Ejemplo de subsecuencia de caracteres más larga.

---

<sup>2</sup>Imagen tomada en: <https://algorithms.tutorialhorizon.com/dynamic-programming-longest-common-subsequence>