



UNIVERSIDADE DA CORUÑA

## Grado en Ingeniería Informática Sistemas inteligentes

### Práctica 1: Estrategias de búsqueda

*Curso 2021/2022*

En esta práctica desarrollarás en Java varias estrategias de búsqueda y las utilizarás para resolver un problema real.

Se te suministra un proyecto de IntelliJ IDEA con código de ejemplo que incluye una primera implementación: la de la Estrategia Básica 4 vista en teoría.

- **Ejercicio 1 (3 puntos): Haz las siguientes modificaciones sobre el código de ejemplo.**
  - a) Crea una clase Nodo según lo descrito en teoría (D45). Adapta Estrategia4 para que registre los estados explorados utilizando Nodos.  
Cambia el método *soluciona* de la clase EstrategiaBusqueda para que devuelva un array de Nodos (Nodo[]). Modifica la implementación de *soluciona* en Estrategia4 para que, cuando encuentre una solución, devuelva la lista de nodos que representan los estados recorridos desde el estado inicial hasta la meta encontrada. Implementa para ello en Estrategia4 un método *reconstruye\_sol* como el descrito en teoría.
  - b) Estrategia4 falla cuando llega a un estado sin sucesores. Sin embargo, es posible que estados por los que había transitado previamente todavía tengan sucesores sin explorar. La estrategia Busqueda\_Grafo vista en teoría soluciona esto añadiendo una frontera en la que almacena los estados sucesores encontrados mientras no son explorados. Implementa una clase EstrategiaBusquedaGrafo (que herede de Estrategia) que implemente dicha estrategia (te recomendamos utilizar Estrategia4 como plantilla).

- **Ejercicio 2 (7 puntos):** Buscamos solucionar el siguiente problema utilizando estrategias de búsqueda:

Un cuadrado mágico de  $N \times N$  es una matriz que contiene los números entre 1 y  $N^2$  dispuestos de tal manera que la suma de los elementos de cada una de sus filas (o de sus columnas o de sus diagonales principales) es siempre la misma:

es siempre la misma:  $\frac{N(N^2+1)}{2}$

2	7	6	→15
9	5	1	→15
4	3	8	→15
↙15	↓15	↓15	↓15

*Ejemplo de cuadrado mágico de 3x3*

Queremos desarrollar un agente capaz de, a partir de un cuadrado parcialmente relleno, completar el cuadrado mágico de  $N \times N$  (si es que es posible), respetando los valores suministrados originalmente.

**Ejemplos:**

**Estado inicial:**

4	9	2
3	5	
	1	

2		

**Estado meta:**

4	9	2
3	5	7
8	1	6

2	9	4
7	5	3
6	1	8

u otro cuadrado mágico de 3x3

2			
	1		

2	8	15	9
14	12	5	3
11	13	4	6
7	1	10	16

u otro cuadrado mágico de 4x4

APARTADO A (3 puntos):

- ☐ Formaliza el problema para que se pueda resolver utilizando estrategias de búsqueda. Utiliza acciones sencillas que solo modifiquen una casilla a la vez. Después impleméntala escribiendo una clase ProblemaCuadradoMagico que sea subclase de ProblemaBusqueda y define las subclases de Estado y Accion necesarias para representar el problema.
- ☐ Implementa las estrategias de búsqueda en profundidad y en anchura y utilízalas para resolver el problema (utiliza el primer ejemplo u otras variaciones sencillas).
- ☐ Adapta tus implementaciones para que lleven cuenta de:
  - ☐ Número de nodos expandidos
  - ☐ Número de nodos creados
- ☐ ¿Cuál de las dos estrategias es la más adecuada? Compruébalo mediante experimentos. ¿Cuál es la causa?

APARTADO B (3 puntos):

- ☐ Indica una heurística apropiada para el Cuadrado Mágico. Crea una implementación de la clase abstracta Heuristica que la calcule para este problema. ¿Es tu heurística admisible? ¿Y consistente? Justifica tu respuesta.
- ☐ Implementa el método de búsqueda A\*. Para ello, crea una subclase de EstrategiaBusquedaInformada y modifica la clase Nodo para que incluya el coste del camino, el valor de la función f y para que implemente el interfaz Comparable. Utiliza esta estrategia para resolver el segundo ejemplo.

APARTADO C (1 punto):

- ☐ ¿Puede tu programa resolver el tercer ejemplo en un tiempo razonable? ¿Qué mejoras harías para conseguir resolverlo más rápidamente? Descríbelas en detalle e impleméntalas. Si la implementación requiere muchos cambios, puedes hacer un nuevo paquete llamado es.udc.sistemasinteligentes.gA\_xy\_2C con las copias mejoradas de las clases.

- **Entrega**

El ejercicio se realizará en los grupos de dos personas establecidos en el Campus Virtual. En la entrega, que se realizará vía Campus Virtual, hay que incluir:

- una pequeña memoria explicativa que contenga las respuestas a las preguntas, incluyendo una descripción de las implementaciones hechas en los apartados correspondientes detallando, si los hay, los problemas encontrados y justificando las decisiones tomadas.
- el código completo debidamente documentado y comentado. Todas las clases deben ir en un paquete llamado `es.udc.sistemasinteligentes.ga_xy` donde `ga_xy` indicará el grupo al que se pertenece (renombrar el paquete `es.udc.sistemasinteligentes` provisto a `es.udc.sistemasinteligentes.ga_xy` y trabaja ahí). Se debe incluir una clase Main para cada ejercicio (MainEj1, MainEj2a y MainEj2b, cada una con el correspondiente método `main`).

La fecha límite de entrega de esta práctica es el **18 de marzo (23:55h)**.