UNIVERSIDAD DE GRANADA

/ UGR / decsai

Departamento de Ciencias de la Computación e Inteligencia Artificial

Técnicas de los Sistemas Inteligentes

Curso Académico 2021-22

Práctica 1: Desarrollo de agentes basado en técnicas de búsqueda (heurística) dentro del entorno GVGAI

La Práctica 1 consiste en desarrollar cinco agentes, basados en cinco técnicas de búsqueda diferentes (incluyendo técnicas de búsqueda no informada y de búsqueda heurística), dentro del entorno GVGAI[1], que guíe a un avatar a resolver un juego en distintos niveles. El juego escogido es el juego con índice 58 en los tipos de juego "*singleplayer*", que se pueden encontrar en el fichero "examples/all_games_sp.csv" de la distribución de GVGAI, denominado *Labyrinth*.

1. Descripción general del juego

Labyrinth es un videojuego sencillo que consiste en encontrar un camino hasta una casilla objetivo, a través de un laberinto donde hay muros que el avatar no puede atravesar, y trampas que eliminan al avatar si éste pasa sobre ellas.

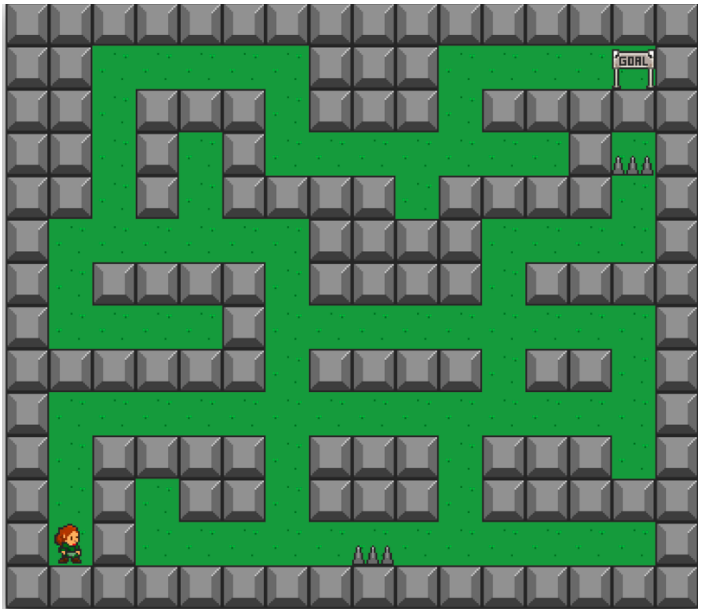

Figura 1: Mapa del primer nivel (levelIdx=0) de Labyrinth en GVGAI

Acciones: se pueden ejecutar 4 acciones de Movimiento (IZQUIERDA, DERECHA, ARRIBA, ABAJO). Adicionalmente, el agente también podría realizar la acción de

1 Se puede descargar desde https://github.com/GAIGResearch/GVGAI/archive/master.zip

USO (definida para otros juegos; en este juego no tiene ningún efecto) y la acción NIL (acción nula).

Para ver más información sobre cómo instalar el entorno y cómo desarrollar un controlador básico para el juego, se pueden consultar las transparencias de la presentación de la práctica y el documento Tutorial de GVGAI. En las transparencias de presentación de la práctica hay instrucciones sobre cómo instalarlo en Eclipse.

Este juego es completamente determinista ya que todos los elementos del mapa, excepto el avatar, son estáticos. Por tanto, el estado del mapa después de que el avatar ejecute cualquier acción es conocido a priori. Por ejemplo, si el avatar decide moverse hacia arriba, en función de la casilla superior se sabe que (i) el avatar se quedará en la casilla actual si hay un muro; (ii) el avatar será eliminado si hay una trampa; (iii) el avatar completará el nivel si hay una casilla objetivo; o (iv) el avatar se desplazará si es una casilla libre.

2. Descripción de la tarea a realizar

El objetivo de la práctica es que los estudiantes se familiaricen con los **comportamientos deliberativos de las técnicas de búsqueda heurística**. Para ello, el estudiante deberá implementar los siguientes algoritmos de búsqueda:

- **Búsqueda en anchura** (**BFS**)
- **Búsqueda en profundidad** (**DFS**)
- **Búsqueda heurística** (offline) con **A***
- **Búsqueda heurística** (offline) **con memoria acotada** con **IDA***
- **Búsqueda heurística** (online) **en tiempo real** con **RTA***

Cada uno de estos algoritmos serán ejecutados en cuatro mapas de diferente tamaño proporcionados por los profesores, desde mapas extremadamente pequeños hasta mapas de dimensiones considerables (véase Figuras 2, 3, 4 y 5).

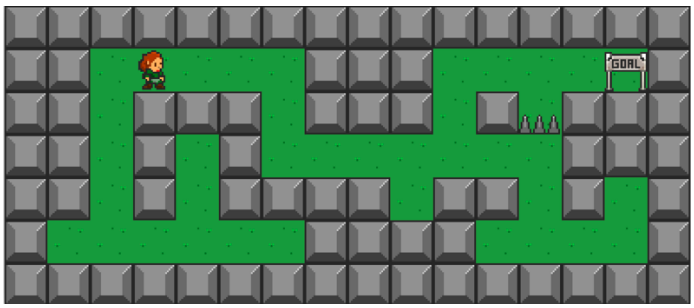

Figura 2: Ejemplo de mapa de tamaño muy pequeño

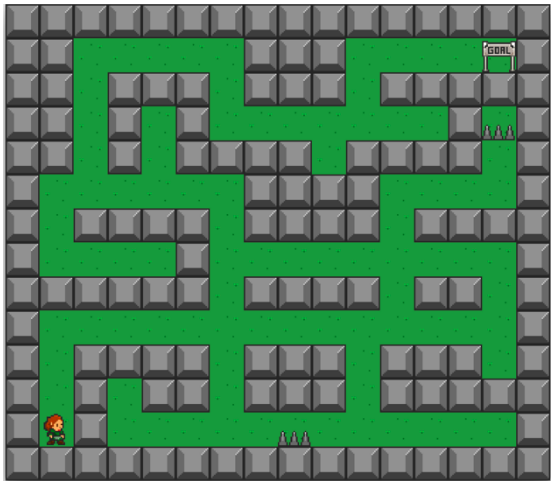

Figura 3: Ejemplo de mapa de tamaño pequeño

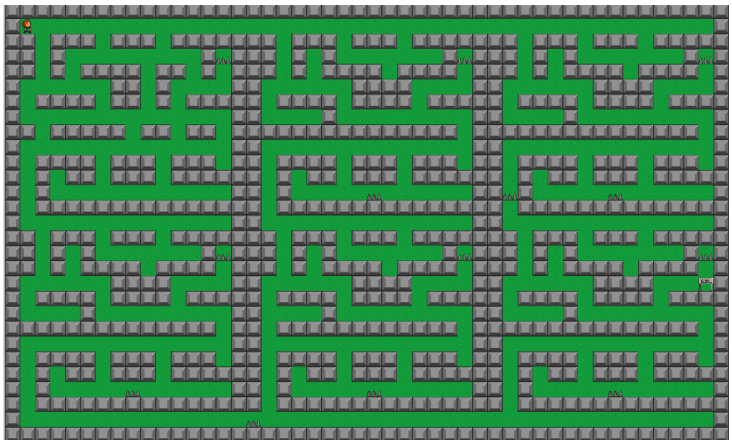

Figura 4: Ejemplo de mapa de tamaño mediano

Independientemente de los mapas suministrados, **se recomienda que los estudiantes creen sus propios mapas para verificar el correcto funcionamiento de sus algoritmos**. Las soluciones propuestas por los estudiantes deben ser generalistas, es decir, dentro de la definición dada de la tarea en cuestión, deben permitir resolver correctamente otros mapas similares.

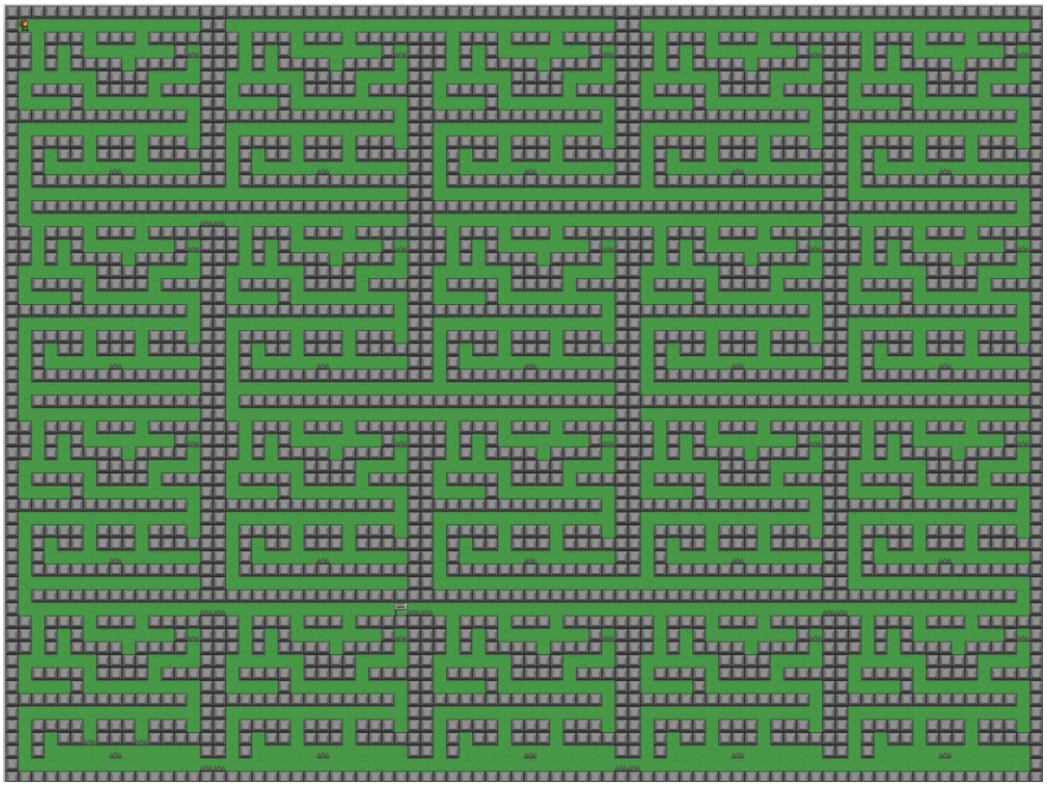

Figura 5: Ejemplo de mapa de tamaño grande

Toda la implementación debe desarrollarse dentro del siguiente **paquete Java** (que cada estudiante deberá crear al inicio de la práctica):

src/tracks/singlePlayer/evaluacion/src_APELLIDO1_APELLIDO2_NOMBRE

donde "nombre" y "apellidos" irán en **mayúscula** y sin guiones ni espacios (Java es case-sensitive). Dentro de este paquete, se definirán, al menos, las siguientes **clases Java** (case-sensitive):

- AgenteBFS
- AgenteDFS
- AgenteAStar
- AgenteIDAStar
- AgenteRTAStar

Adicionalmente, la implementación deberá respetar y/o considerar las siguientes restricciones, penalizando total o parcialmente la puntuación final obtenida en aquellos casos donde no se respeten o consideren.

Restricción de nomenclatura. La nomenclatura anterior (nombre de paquete y clases) debe ser respetada **estrictamente**.

Restricciones en la ejecución. Es importante tener en cuenta que cada ciclo de decisión está limitado de tal forma que la decisión de qué acción ejecutar en **cada *tick* debe tomarse en un tiempo no superior a 40ms**[2]. Además, **la ejecución está limitada a un máximo de 10.000 *ticks***. Por tanto, es importante que la implementación sea suficientemente eficiente, con una estructura de datos ligera y ágil en las operaciones requeridas, y que la definición del juego y de GVGAI permita la ejecución hasta 10.000 *ticks* (ver la presentación de la práctica para más detalles).

Restricciones en el constructor de los agentes. Las operaciones realizadas en el constructor de los agentes se limitarán a acciones menores de inicialización. **Toda acción relativa al método de búsqueda deberá realizarse dentro del método "act"**. En concreto, todos los agentes (excepto RTA*) calcularán la ruta desde la posición actual del avatar hasta la casilla objetivo en la primera ejecución del método "act", eligiendo en esta primera acción la primera casilla a la que desplazarse, y en las llamadas sucesivas simplemente seguirán ejecutando los movimientos para desplazarse por la ruta ya calculada.

Restricciones de implementación de los algoritmos de búsqueda. La **versión exacta** de cada uno de los algoritmos de búsqueda que se deben implementar para esta práctica serán **proporcionados en las sesiones de práctica** y deben ser éstas, y no otras, las implementadas por los estudiantes. Nótese que cualquier discrepancia en la implementación puede dar lugar a resultados diferentes de los esperados (por ejemplo, en el número de *ticks* que el agente requiere para resolver un mapa), con la consecuente penalización en la nota final.

Restricciones de la heurística. La función heurística usada por todos los métodos de búsqueda será siempre la **distancia Manhattan**.

Restricciones de expansión de vecinos. El **orden de expansión** de los nodos **vecinos** de un nodo actual será siempre el mismo: **Arriba, Abajo, Izquierda, Derecha**. En los algoritmos de búsqueda no informada, este orden se usará directamente para visitar los siguientes nodos. En los algoritmos de búsqueda heurística, este orden se usará como criterio último para realizar desempates. Por ejemplo, en A* los nodos se visitan según el valor de f(n) y, en caso de empate, según el valor de g(n). Si tras estas dos comparaciones sigue existiendo un empate, los nuevos nodos se ordenarán siguiendo el orden anterior a modo de cola FIFO.

Restricciones de mensajes. Los agentes deberán **imprimir por pantalla únicamente** la información relativa a los algoritmos de búsqueda que se describe en la Tabla 1.

[2] Si la acción seleccionada en el método *act* se decide entre 40-50ms GVGAI devuelve acción nula; si el tiempo es mayor a 50ms se descalifica al agente.

Una vez realizada la implementación de los distintos agentes deliberativos, el estudiante deberá **ejecutarlos en los mapas proporcionados, de forma progresiva** (de menor a mayor tamaño) hasta que el agente no sea capaz de resolver cierto nivel, reportando los resultados hasta dicho nivel. Por ejemplo, si el agente A* hace *timeout* en el mapa mediano, se reportarán los resultados de este agente sobre el mapa muy pequeño, pequeño y mediano (ignorando la ejecución en el mapa grande y, por tanto, dejando en blanco la fila de la tabla correspondiente a ese método y a ese mapa).

Nota: el hecho de que un algoritmo haga *timeout* en cierto mapa, no impide ejecutar el correspondiente algoritmo de búsqueda en dicho mapa. En concreto, GVGAI detectará si se ha superado el tiempo límite después de que el agente devuelva una acción en el método "act". En ese momento, el algoritmo de búsqueda habrá calculado el plan completo (excepto en RTA*), por lo que se pueden reportar todos los valores pedidos en la tabla. Además, en la columna "*runtime*" se especificará que el agente fue descalificado por *timeout* (TO).

En base a estas ejecuciones, cada estudiante deberá desarrollar una memoria donde se complete la siguiente **tabla de resultados**:

Alg.	Mapa	Runtime (acumulado)	Tamaño de la ruta calculada	Número de nodos expandidos	Máximo número de nodos en memoria
BFS	Muy pequeño				
	Pequeño				
	Mediano				
	Grande				
DFS	Muy pequeño				
	Pequeño				
	Mediano				
	Grande				
A*	Muy pequeño				
	Pequeño				
	Mediano				
	Grande				
IDA*	Muy pequeño				
	Pequeño				
	Mediano				
	Grande				
RTA*	Muy pequeño				
	Pequeño				
	Mediano				
	Grande				

Tabla 1: Tabla de resultados

Para rellenar la tabla anterior se deben tener en cuenta las siguientes consideraciones:

- La columna "*runtime*" hace referencia únicamente al tiempo total utilizado por el algoritmo de búsqueda para calcular el plan (sin contar el tiempo que cada agente invertirá en ejecutar los desplazamientos). Nótese que BFS, DFS, A* e IDA* únicamente realizan una llamada al algoritmo de búsqueda para calcular el plan hasta el objetivo, mientras que RTA* realiza múltiples llamadas. Para medir el tiempo de cada una de estas llamadas, basta con usar estas líneas de código:

```
long tInicio = System.nanoTime();
// Código para el que se mide el tiempo
//  Ej: llamada A* que devuelve la ruta hasta objetivo
long tFin = System.nanoTime();
long tiempoTotalEnSegundos = (tFin - tInicio)/1000000;
```

- La columna "tamaño de la ruta calculada" hace referencia al número de casillas transitadas por el agente durante la ejecución.

- La columna "número de nodos expandidos" hace referencia al número de nodos para los que se ha comprobado si son nodos objetivo durante la ejecución del algoritmo (nótese que en algunos algoritmos, como en IDA*, esta comprobación se puede realizar en múltiples ocasiones para el mismo nodo durante la ejecución del algoritmo).

- La columna "máximo número de nodos en memoria" hace referencia al número máximo de nodos pendientes de visitar durante la ejecución del algoritmo, es decir, nodos que podrán ser expandidos en el futuro en caso de no encontrar un nodo solución previamente. Por ejemplo, en A* este número corresponde a la suma del tamaño de las listas de nodos abiertos y de nodos cerrados.

Nota: cada agente **solo puede, y debe, imprimir por pantalla** la información que se pide para rellenar la Tabla 1, es decir, *runtime* (acumulado), tamaño de la ruta calculada y número de nodos expandidos.

Adicionalmente, en la memoria se debe desarrollar la **respuesta justificada** de las siguientes preguntas:

1. Entre BFS y DFS, ¿qué algoritmo puede ser considerado más eficiente de cara a encontrar el camino óptimo?
2. ¿Se podría decir que A* es más eficiente que DFS?
3. ¿Cuáles son las principales diferencias entre A* e IDA*? ¿En qué contextos es más conveniente usar uno u otro?
4. ¿Se podría decir que RTA* es más eficiente que A*?

La respuesta a estas preguntas debe estar basada tanto en las características teóricas de los algoritmos como en los resultados experimentales obtenidos. **Aquellas respuestas que no estén adecuadamente justificadas o desarrolladas podrán invalidar total o parcialmente aquellos ejercicios a los que afecte.**

3. Material a entregar

El material a entregar será un fichero ZIP con el siguiente contenido:

- La carpeta del paquete Java, que tendrá el mismo nombre que el definido anteriormente. Dentro de esta carpeta estarán todas las clases Java correspondientes a los cinco agentes deliberativos (y otras clases auxiliares), con las siguientes consideraciones:
  a) El **código** debe estar adecuadamente presentado y **comentado**, explicando los distintos aspectos de la solución propuesta por el estudiante de una forma clara: qué se hace, cómo se hace y por qué. **Un código insuficientemente comentado o con comentarios extremadamente deficientes podrá invalidar total o parcialmente el ejercicio al que afecte**.
  b) El código **no puede imprimir nada**, salvo lo especificado con anterioridad, correspondiente a la información para cubrir la Tabla 1.
- El fichero PDF de la memoria, con una **extensión máxima de 3 páginas**, donde se incluyan los **datos del estudiante**, la **tabla de resultados** y las **respuestas razonadas** a las cuestiones planteadas. **Aquellas respuestas que no estén adecuadamente justificadas podrán invalidar total o parcialmente aquellos ejercicios a los que afecten**.

4. Criterios de evaluación

La evaluación consistirá en la ejecución del **software entregado** para comprobar su efectividad y eficiencia en la resolución de distintos mapas. Durante esta ejecución, los profesores podrán usar mapas diferentes a los proporcionados, para comprobar que la implementación es correcta. La calificación de la práctica se realizará teniendo en cuenta las siguientes puntuaciones:

- Implementación de BFS: 1 punto
- Implementación de DFS: 1 punto
- Implementación de A*: 2 puntos
- Implementación de IDA*: 2 puntos
- Implementación de RTA*: 2 puntos
- Memoria: 2 puntos