

INTELIGENCIA ARTIFICIAL

CURSO 2024-25

PRACTICA 2: Repertorio de preguntas para la autoevaluación de la práctica 2.

APELLIDOS Y NOMBRE	Fata Hamza		
GRUPO TEORÍA		GRUPO PRÁCTICAS	

Instrucciones iniciales

En este formulario se proponen preguntas que tienen que ver con ejecuciones concretas del software desarrollado por los estudiantes. También aparecen preguntas que requieren breves explicaciones relativas a como el estudiante ha hecho algunas partes de esa implementación y que cosas ha tenido en cuenta.

En las preguntas relativas al funcionamiento del software del alumno, estas se expresan haciendo uso de la versión de invocación en línea de comandos cuya sintaxis se puede consultar en el guion de la práctica.

El estudiante debe poner en los recuadros la información que se solicita.

En los casos que se solicita una captura de pantalla (**ScreenShot**), extraer la imagen de la ejecución concreta pedida (sustituyendo la llamada practica2SG por practica2). En los **niveles 0 y 1**, esta captura será de la situación final de la simulación en el modo "mapa" en la que se ve lo que los agentes descubrieron. En los **niveles 2 y 3** donde aparezca la línea de puntos que marca el recorrido (justo en el instante en el que se construye obtiene el plan). Además, en dicha captura debe aparecer al menos el nombre del alumno. Un ejemplo de captura de imagen se puede encontrar en el apartado (a) del nivel 0.

Consideraciones importantes:

- Antes de empezar a rellenar el cuestionario, actualiza el código de la práctica con los cambios más recientes. Recuerda que puedes hacerlo o bien realizando **git pull upstream main** si has seguido las instrucciones para enlazar el repositorio con el de la asignatura, o bien descargando desde el enlace de GitHub el zip correspondiente, y sustituyendo los ficheros **rescatador.cpp**, **rescatador.hpp**, **auxiliar.cpp** y **auxiliar.hpp** por los vuestros.
- Si en alguna ejecución consideras que tu agente se ha visto perjudicado puedes añadirlo a los comentarios en el comentario final (al final del documento).


Enumera los niveles presentados en su práctica (Nivel 0, Nivel 1, Nivel 2, Nivel 3, Nivel 4):

Nivel 0, Nivel 1, Nivel 2, Nivel 3,

Nivel 0-El Despertar Reactivo

(a) Rellena los datos de la tabla con el resultado de aplicar

`./practica2SG ./mapas/mapa30.map 0 0 24 10 2 17 17 0 3 3 0`

<div> <div>ScreenShot</div>  </div>	
Instantes de simulación no consumidos	2933
Coste de Energía (Rescatador)	81 (3000 – 2919)
Coste de Energía (Auxiliar)	15

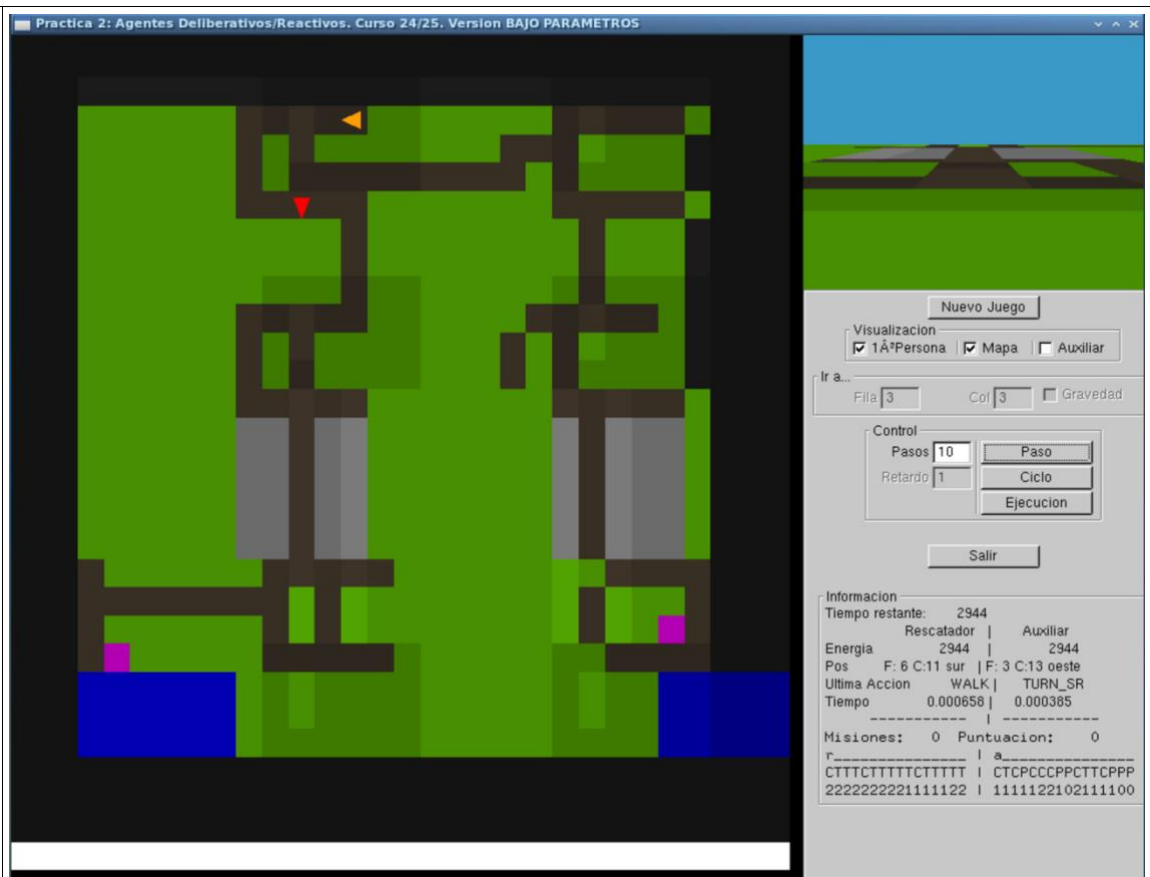
(b) Rellena los datos de la tabla con el resultado de aplicar

`./practica2SG ./mapas/mapa30.map 0 0 16 9 2 16 14 6 3 3 0`

Screenshot		
Instantes de simulación no consumidos		
Coste de Energía (Rescatador)		
Coste de Energía (Auxiliar)		

- (c) Rellena los datos de la tabla con el resultado de aplicar
- `./practica2SG mapas/gemini2.map 0 0 3 10 2 3 13 6 3 3 0`**
- Se quedan atascados en un bucle

ScreenShot



Instantes de simulación no consumidos

Coste de Energía (Rescatador)

Coste de Energía (Auxiliar)

Nivel 1-La cartografía de lo Desconocido

- (a) Describe brevemente cuál es el comportamiento que has implementado en los agentes para explorar el mundo. Indica si has usado los dos. Si has usado los dos indica describe las diferencias que hubiera entre ellos. (enumera los cambios y describe brevemente cada uno de ellos)

Se han utilizado ambos agentes (Rescatador y Auxiliar) para la exploración ,con la estrategia siguiente:

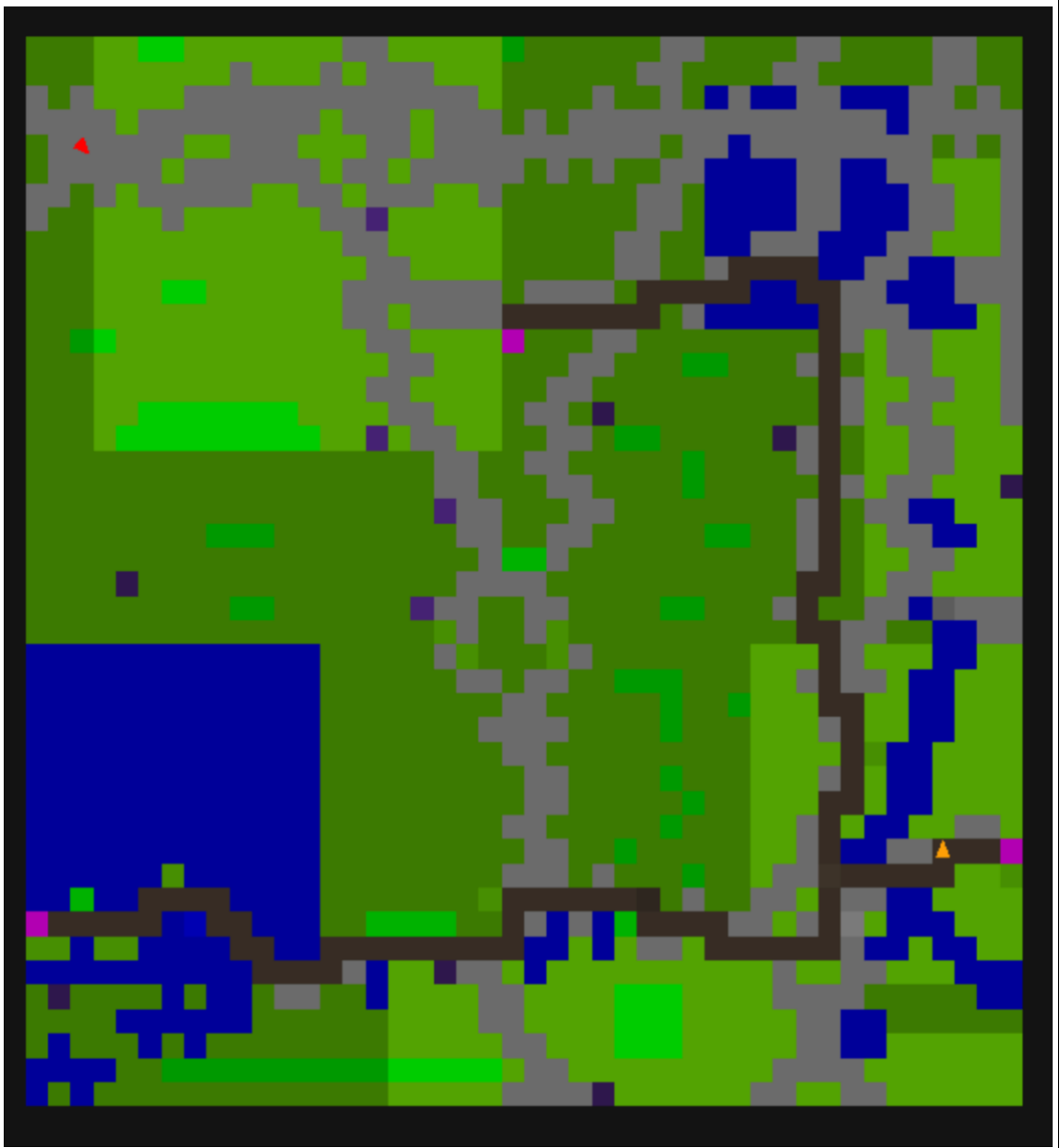
- **Prioridad Principal: Explorar lo Desconocido.** Ambos agentes priorizan el avance hacia casillas desconocidas (marcadas con '?') que estén en su campo de visión frontal. El orden de preferencia es: adelante (`sensores.superficie[2]`), luego 45° a la izquierda (`sensores.superficie[1]`) y finalmente 45° a la derecha (`sensores.superficie[3]`). Solo se moverán a una casilla '?' si es de tipo 'C' (Camino) o 'S' (Sendero).
- **Comportamiento de Respaldo: Seguir Caminos.** Si no hay casillas desconocidas a la vista, los agentes recurren a un comportamiento de seguimiento de caminos similar al del Nivel 0, pero esta vez aceptando tanto 'C' como 'S' como rutas válidas.
- **Diferencias entre Agentes:** La principal diferencia radica en su conjunto de acciones. El Rescatador puede usar `TURN_L` para giros eficientes a la izquierda, mientras que el Auxiliar debe componer varios `TURN_SR` para lograr el mismo efecto, lo que lo hace menos ágil en la exploración.
- **Mapeo:** Ambos agentes utilizan la función `SituarSensorEnMapa` en cada ciclo para actualizar `mapaResultado` y `mapaCotas` con la información de sus sensores.

- (b) Rellena los datos de la tabla con el resultado de aplicar

./practica2SG mapas/mapa50.map 0 1 5 3 2 36 44 6 3 3 0

- Se quedan atascados

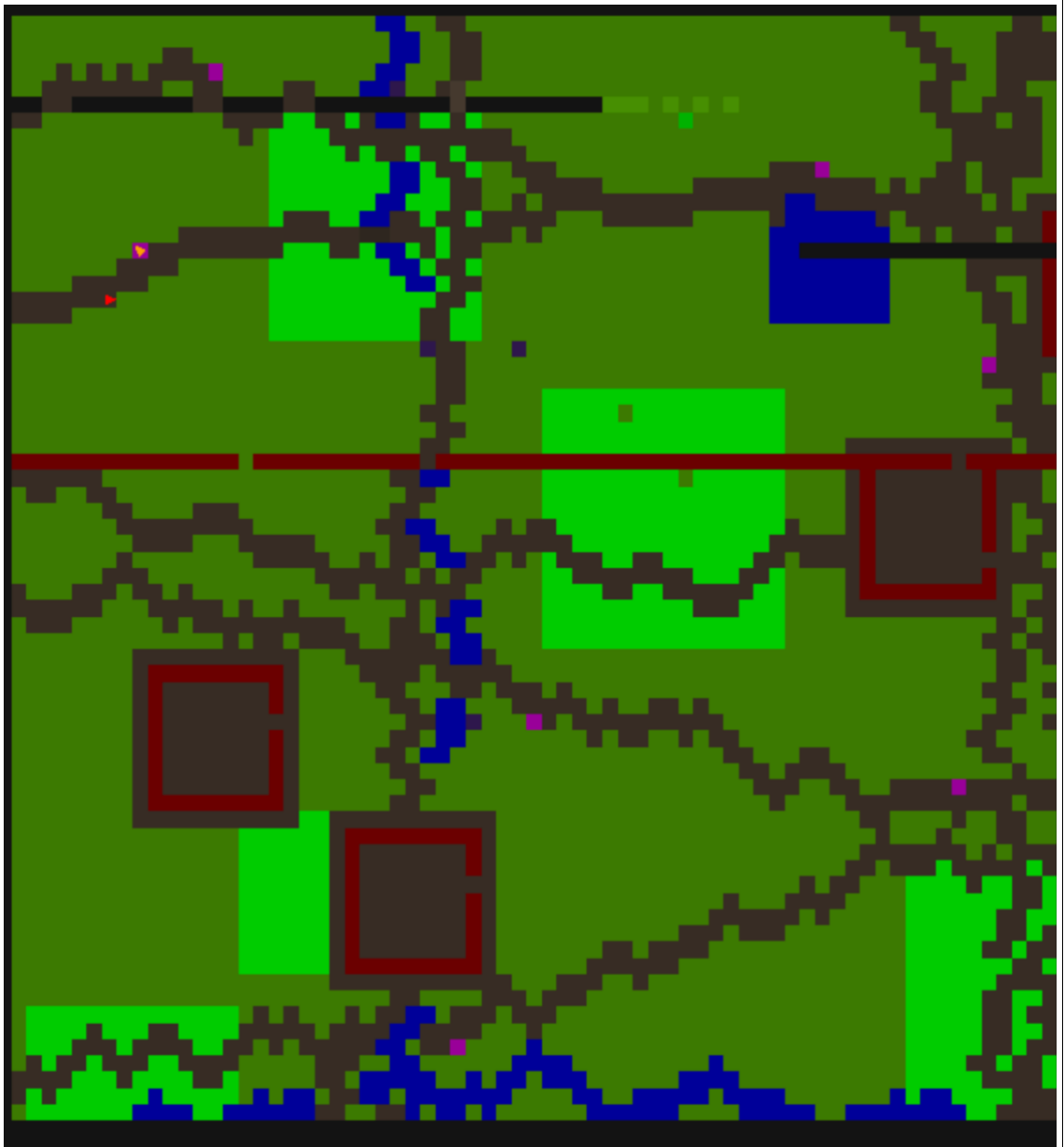
ScreenShot



Porcentaje descubierto de caminos y senderos

- (c) Rellena los datos de la tabla con el resultado de aplicar
`./practica2SG mapas/mapa75.map 0 1 20 9 2 15 23 6 3 3 0`

ScreenShot



Porcentaje descubierto de caminos y senderos

Nivel 2-La Búsqueda del Camino de Dijkstra

- (a) Indica cuál ha sido la definición de estado para resolver este problema. Justifica la definición.

La definición del estado para el agente Rescatador (EstadoR en `rescatador.hpp`)

```
es: struct EstadoR { int fila; int columna; int orientacion; bool
tiene_zapatillas; }
```

Justificación de cada componente:

- `fila, columna`: Son esenciales para definir la posición del agente en el mapa.
- `orientacion`: Es crucial porque las acciones de giro (`TURN_L`, `TURN_SR`) cambian este valor y tienen un coste energético. Sin la orientación, no se podría calcular correctamente el coste total de un plan que incluya giros.
- `tiene_zapatillas`: Este booleano es fundamental porque modifica las reglas de movimiento del agente. Con las zapatillas, el Rescatador puede superar diferencias de altura de hasta 2 unidades. Esto cambia qué transiciones son posibles en el grafo de búsqueda y, por tanto, debe formar parte del estado.

(b) ¿Has incluido dentro del algoritmo de búsqueda que si pasas por una casilla que da las zapatillas, considere en todos los estados descendientes de él, se está en posesión de las zapatillas? En caso afirmativo, explicar brevemente cómo.

Sí, se ha incluido. Dentro de la función `applyR` (en `rescatador.cpp`), que se usa para generar los estados sucesores en el algoritmo de Dijkstra, se comprueba el tipo de terreno de la casilla de destino. Si la nueva posición (`next.fila, next.columna`) corresponde a una casilla de tipo 'D', la variable booleana `next.tiene_zapatillas` del nuevo estado se establece en `true`. Este estado modificado es el que se añade a la frontera de búsqueda, por lo que todos los nodos que descendan de él heredarán la posesión de las zapatillas.

Nivel 3-El Ascenso del A*uxiliar

(a) ¿Qué diferencia este algoritmo del de Dijkstra que tuviste que implementar en el nivel anterior? (enumera los cambios y describe brevemente cada uno de ellos y que han implicado en la implementación)

La diferencia fundamental es que A* es un algoritmo de **búsqueda informada**, mientras que Dijkstra es de **búsqueda no informada** (o de coste uniforme). Esto implica dos cambios clave en la implementación:

1. **Uso de una Heurística**: A* utiliza una función heurística, $h(n)$, para estimar el coste desde el nodo actual hasta el objetivo. Esta estimación guía la búsqueda de manera más directa hacia el destino, expandiendo menos nodos que Dijkstra, que explora en todas las direcciones de manera uniforme.
2. **Criterio de Ordenación en la Cola de Prioridad**: Dijkstra ordena los nodos en la frontera únicamente por su coste acumulado desde el inicio, $g(n)$. En cambio, A* los ordena por la suma del coste acumulado más el coste heurístico, $f(n) = g(n) + h(n)$. Esto se refleja en la sobrecarga del `operator<` en la estructura del nodo (`NodoA_AStar`), que ahora considera `g_cost + h_cost`.

(b) Describe la heurística utilizada para resolver el problema

- (c) La heurística utilizada es la **Distancia de Manhattan**. Se calcula como la suma de las diferencias absolutas entre las coordenadas del estado actual ($st.f$, $st.c$) y las del estado final ($final.f$, $final.c$).
- (d) Fórmula: $h(n) = \text{abs}(st.f - final.f) + \text{abs}(st.c - final.c)$
- (e) Esta heurística es **admisibile** porque nunca sobreestima el coste real para llegar al objetivo, ya que representa el camino más corto en una cuadrícula sin obstáculos y sin considerar costes diagonales. El uso de una heurística admisible es un requisito para que A* garantice encontrar la solución óptima

Comentario final

Consigna aquí cualquier tema que creas, que es de relevancia para la evaluación de tu práctica o que quieras hacer saber al profesor.

Quería añadir un comentario sobre el estado de esta entrega. El documento está incompleto debido a una serie de problemas que ocurrieron al intentar subir los últimos cambios a mi repositorio de GitHub.

Para intentar solucionar dificultades con la actualización de los ficheros, utilicé un bot de IA para que me asistiera con el proceso del `commit` y `push`, pero desafortunadamente, esto complicó la situación. Durante la operación, algunos archivos de configuración del proyecto fueron eliminados o alterados accidentalmente.

Esto ha provocado un problema crítico en la ejecución del entorno, que ya de por sí es complicado de configurar y mantener en mi MacBook. Debido a este cambio de última hora, el programa ya no funciona como antes y no he podido generar las capturas para completar la autoevaluación.

Soy consciente de que faltan muchas cosas en el proyecto y que, a nivel de ejecución, varias partes no funcionan como deberían. Espero que puedan tener en cuenta estas circunstancias, así como las dificultades y la presión de tiempo a las que me he enfrentado para poder entregar la práctica.