

INTELIGENCIA ARTIFICIAL

CURSO 2024-25

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

APELLIDOS Y NOMBRE	Sallami Moreno Ismael		
GRUPO TEORÍA	DG	GRUPO PRÁCTICAS	DG1

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. Ejemplos de imágenes se pueden encontrar en [Imagen1](#) y en [Imagen2](#).

Consideraciones importantes:

- Antes de empezar a llenar 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).

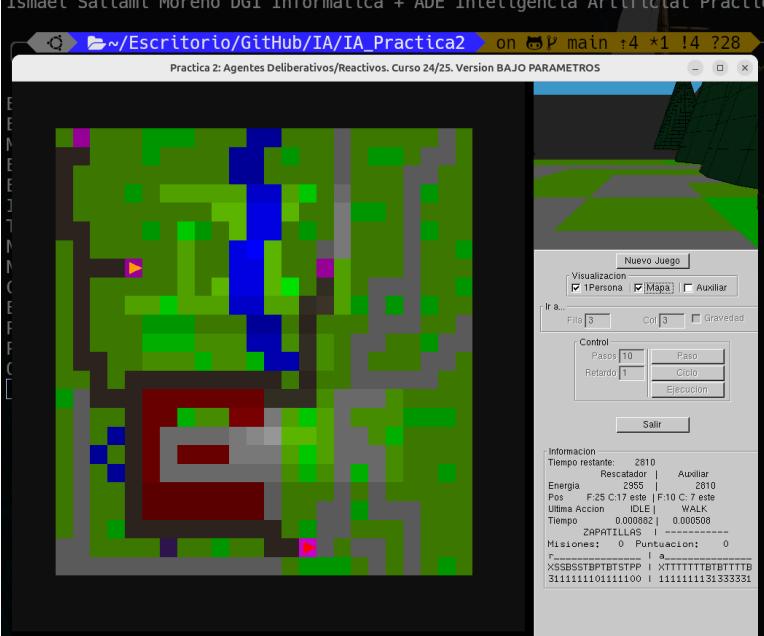
Enumera los niveles presentados en su práctica (Nivel 0, Nivel 1, Nivel 2, Nivel 3, Nivel 4):

Nivel 0, Nivel 1, Nivel 2, Nivel 3, Nivel 4

Nivel 0-El Despertar Reactivo

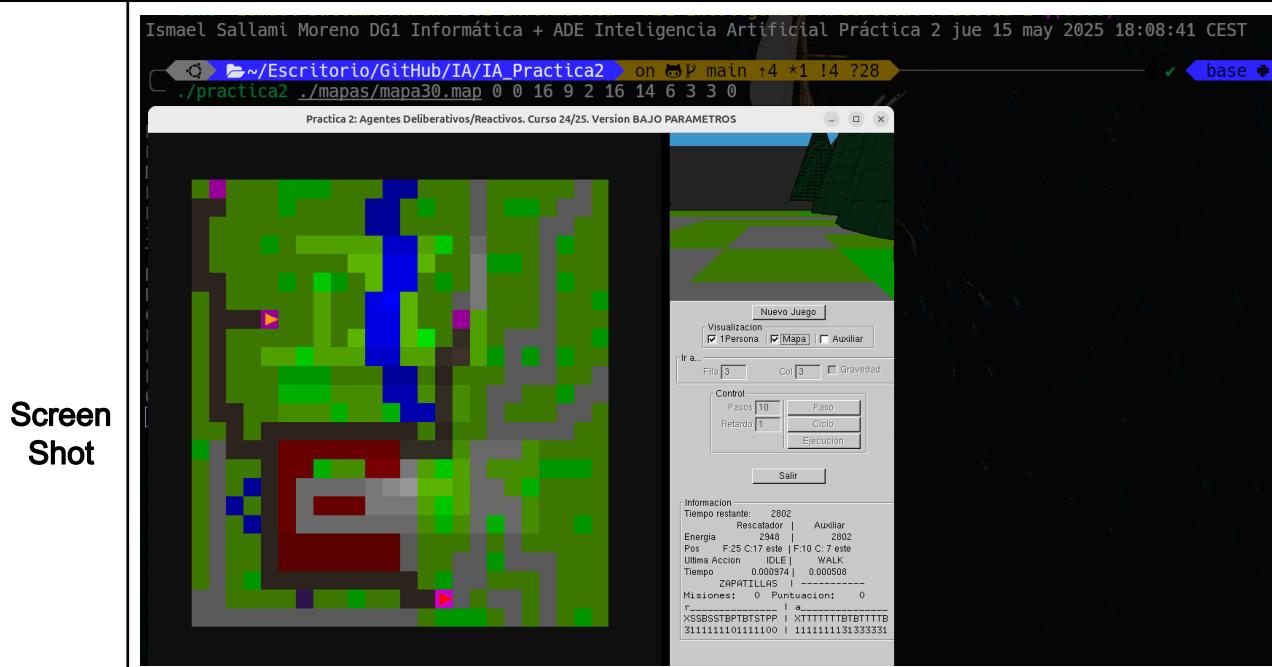
(a) Rellena los datos de la tabla con el resultado de aplicar

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

Screen Shot	
Instantes de simulación no consumidos	2810
Coste de Energía (Rescatador)	45
Coste de Energía (Auxiliar)	190

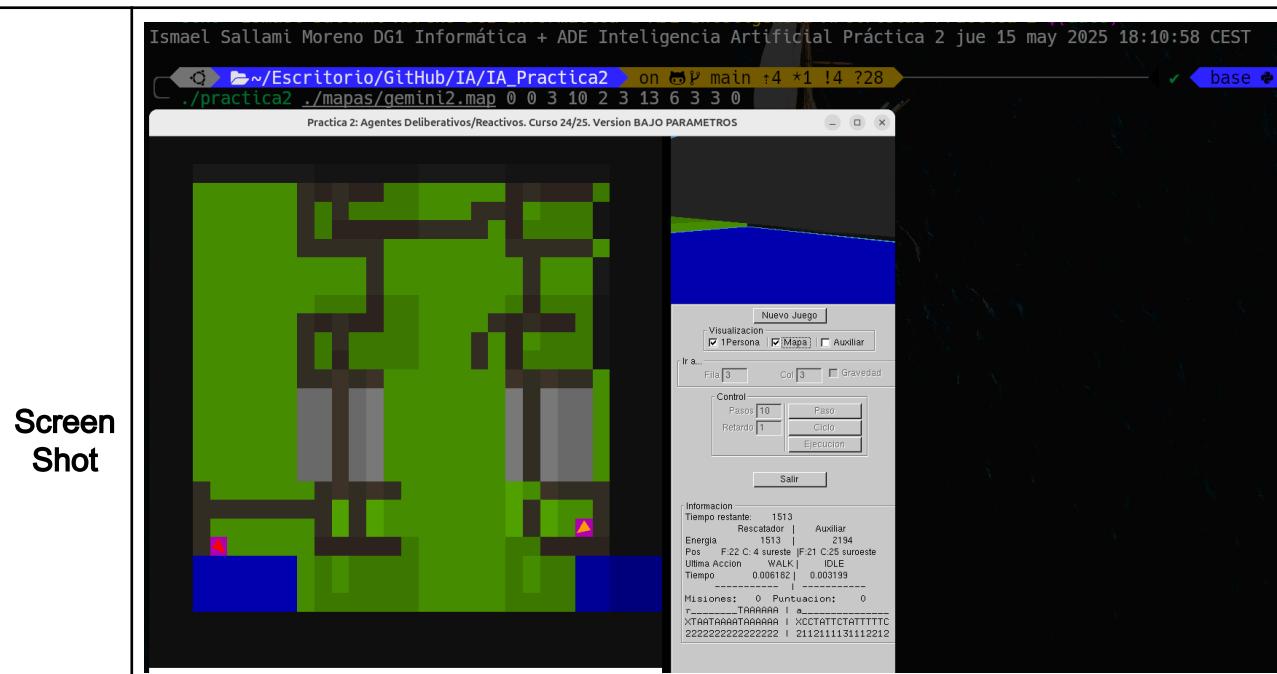
(b) Rellena los datos de la tabla con el resultado de aplicar

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



Instantes de simulación no consumidos	2802
Coste de Energía (Rescatador)	52
Coste de Energía (Auxiliar)	198

(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`



Instantes de simulación no consumidos	1513
Coste de Energía (Rescatador)	1487
Coste de Energía (Auxiliar)	806

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)

En mi caso he usado ambos agentes.

Rescatador: En este evitamos en un primer momento la colisión con el otro agente, cogemos las zapatillas si podemos.

Si la casilla actual la hemos visitado más de 6 veces, escogemos una acción priorizando las casillas de interrogación, si no, pasamos a ver si la casilla de la semiderecha es válida, si lo es avanzamos hacia ahí, y así con las demás.

Luego, uso un contador que cuente cada acción, simulando los turnos que llevamos, de manera que si llegamos al turno 2200 cogemos una casilla válida en el orden semiderecha, delante e izquierda.

Luego deshacemos los giros, si los movimientos posibles no están vacíos lo devolvemos. Luego vemos la casilla que más interesante es. Si devuelve 0 la casilla interesante cogemos una aleatoria.

Auxiliar: Primero este agente obtiene los movimientos válidos en función de la posición en la que se encuentre, evitamos la colisión con el rescatador, si la casilla actual es la de zapatillas las cogemos, si estamos en una casilla, la cual hemos visitado más de 30 veces, vemos si la de la semiderecha es válida vamos devolvemos esa dirección, luego lo mismo con delante y por último hacia la semizquierda, si ninguna es válida giramos hacia la izquierda.

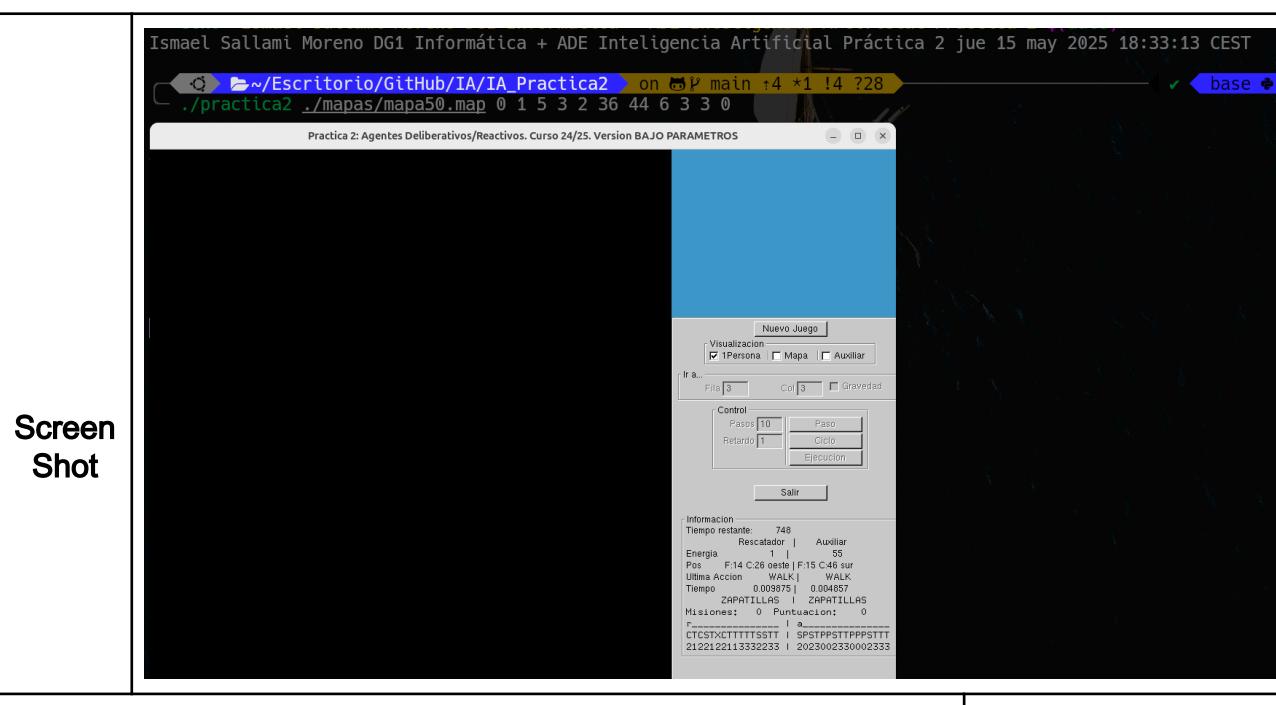
Si la estructura que almacena los movimientos válidos no está vacía, devolvemos el primero de ahí.

A continuación, pasamos a deshacer los giros45 que tengamos y calculamos cual es la casilla interesante.

Por último, para evitar bucles, cada x número de acciones elegimos una aleatoria.

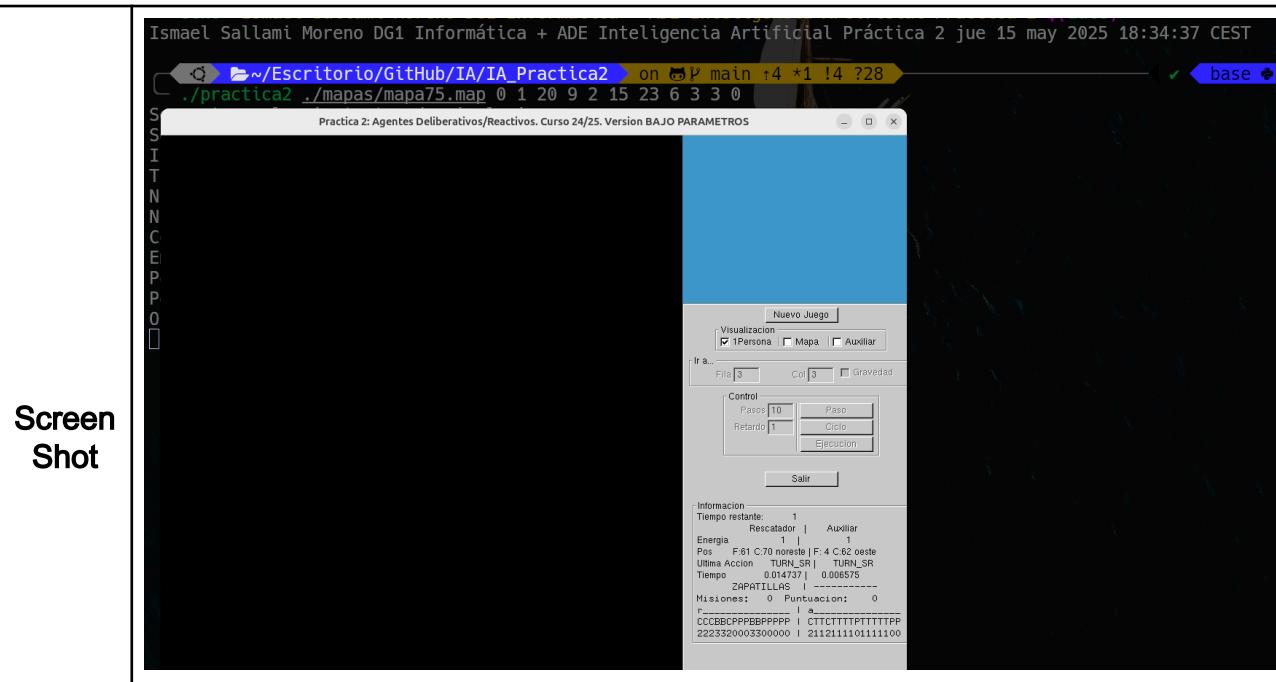
- (b) Rellena los datos de la tabla con el resultado de aplicar

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



Porcentaje descubierto de caminos y senderos | 100

(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`



Porcentaje descubierto de caminos y senderos | 63.2833

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

```
./practica2SG ./mapas/parchis.map 1 1 32 68 6 17 68 4 3 3 0
```

The screenshot shows a terminal window with the following details:

- Terminal title: ~ / Escritorio / GitHub / IA / IA_Practica2
- Terminal command: ./practica2 ./mapas/parchis.map 1 1 32 68 6 17 68 4 3 3 0
- Terminal output: Práctica 2: Agentes Deliberativos/Reactivos. Curso 24/25. Versión BAJO PARAMETROS
- Graphical User Interface (right side):
 - Nuevo Juego** button
 - Visualización** dropdown with options: 1 Persona, Mapa, Auxiliar
 - Ir a...** button
 - Fila**: 3 | **Columna**: 3 | **Gravedad**
 - Control** section:
 - Pasos**: 10 | **Paso**
 - Retardo**: 1 | **Ciclo**
 - Ejecución**
 - Salir** button
- Information panel (bottom right):
 - Tiempo restante: 463
 - Energía: 1 | Rescata: 393
 - Pos: F-39 C-59 noroeste | F-80 C-80 este
 - Última Acción: TURN_SR | TURN_SR
 - Tiempo: 0.011916 | 0.005471
 - ZAPATILLAS: 0 | ZAPATILLAS: 0
 - Misiones: 0 | Puntuación: 0
 - Mapa representation: CCSSSSSSSSSSSSS | 000000000DPPPPP
3411411114111111 | 2222222200000000

Nivel 2-La Búsqueda del Camino de Dijkstra

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

En cuanto al rescatador la definición de estado ha sido la siguiente:

- f y c: representa la fila y la columna para saber la posición en la que se encuentra dicho estado.
- brújula: representa hacia donde está orientado el estado (noroeste, suroeste, ...).
- zapatillas: indica si se tiene o no las zapatillas.
- operador ==: ver si dos estados son iguales.
- operador <: ayuda a ver si un estado es menor a otro.

Se ha implementado de esta manera debido a que es necesario representar cada variable de estado y poder compararlos entre sí.

(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. En mi caso si el nodo actual se encuentra en una casilla que da las zapatillas que ponga a true esta variable, y como los hijos que se generan es en base a este nodo actual, se entiende que los que se generan a partir de este ya tienen en cuenta esto.

(c) En el algoritmo de búsqueda en anchura, el primer nodo que se genera compatible con la solución es la solución óptima y se puede detener la búsqueda en ese punto. Esto no se verifica en el algoritmo de Dijkstra. ¿Tuviste en cuenta esto en tu implementación? Describe brevemente como lo tuviste en cuenta.

En mi algoritmo hago uso de una matriz que me ayuda para tener esto en cuenta. En el algoritmo de Dijkstra lo que hacemos es buscar el camino de coste mínimo, por ende, mi matriz compuesta por las filas, las columnas y la orientación (inicializada con INT_MAX), almacena las casillas y el coste de estas si se explora. De manera que al generar cada hijo, vemos si el coste del hijo que hemos generado es menor que el coste que tenemos en la matriz, que se supone que debe de ser el mínimo, lo actualizamos y lo añadimos a la cola de abiertos para explorarlo, ya que el coste sería menor.

(d) Incluye en el siguiente recuadro de texto el trozo de código de tu implementación del algoritmo de Dijkstra que genera el nodo

descendiente de aplicar la acción RUN sobre el estado actual. Si tu proceso de generación de descendientes es genérico, pon como trozo de código todo el ciclo donde esté incluida esa generación de los descendientes.

NOTA: Adjunto también imagen por si se ve mejor, más claro y ordenado.

Parte de Dijkstra:

```
for (Action accion : acciones)
    { // Hemos decidido usar un bucle para evitar repetir código
        NodoR hijo = current_nodo;
        hijo.estado = applyR(accion, current_nodo.estado, terreno, altura);
        int costeTerreno =
            CosteBaseTerreno(terreno[current_nodo.estado.f][current_nodo.estado.c], accion);
        int costeAltura = (accion == WALK || accion == RUN) ? CosteCambioAltura(
            altura[current_nodo.estado.f][current_nodo.estado.c],
            altura[hijo.estado.f][hijo.estado.c],
            terreno[current_nodo.estado.f][current_nodo.estado.c],
            accion)
            : 0;

        hijo.coste = current_nodo.coste + costeTerreno + costeAltura;
        hijo.secuencia = current_nodo.secuencia;
        hijo.secuencia.push_back(accion);

        int f = hijo.estado.f;
        int c = hijo.estado.c;
        int ori = hijo.estado.brujula;
        int zap = hijo.estado.zapatillas ? 1 : 0;

        if (f >= 0 && f < filas && c >= 0 && c < columnas)
        {
            if (hijo.coste < costeMinimo[f][c][ori][zap])
            {
                costeMinimo[f][c][ori][zap] = hijo.coste;
                frontier.push(hijo);
            }
        }
    }
```

```

● ● ●
1 for (Action accion : acciones)
2 { // Hemos decidido usar un bucle para evitar repetir código
3     NodoR hijo = current_nodo;
4     hijo.estado = applyR(accion, current_nodo.estado, terreno, altura);
5     int costeTerreno = CosteBaseTerreno[terreno[current_nodo.estado.f][current_nodo.estado.c], accion];
6     int costeAltura = (accion == WALK || accion == RUN) ? CosteCambioAltura(
7                                     altura[current_nodo.estado.f][current_nodo.estado.c],
8                                     altura[hijo.estado.f][hijo.estado.c],
9                                     terreno[current_nodo.estado.f][current_nodo.estado.c],
10                                    accion)
11                                    : 0;
12
13     hijo.coste = current_nodo.coste + costeTerreno + costeAltura;
14     hijo.secuencia = current_nodo.secuencia;
15     hijo.secuencia.push_back(accion);
16
17     int f = hijo.estado.f;
18     int c = hijo.estado.c;
19     int ori = hijo.estado.brujula;
20     int zap = hijo.estado.zapatillas ? 1 : 0;
21
22     if (f >= 0 && f < filas && c >= 0 && c < columnas)
23     {
24         if (hijo.coste < costeMinimo[f][c][ori][zap])
25         {
26             costeMinimo[f][c][ori][zap] = hijo.coste;
27             frontier.push(hijo);
28         }
29     }
30 }

```

EstadoR ComportamientoRescatador::applyR(Action accion, const EstadoR &st, const vector<vector<unsigned char>> &terreno, const vector<vector<unsigned char>> &altura)

{

EstadoR next = st;

switch (accion)

{

case WALK:

if (CasillaAccesibleRescatador(st, terreno, altura))

{

next = NextCasillaRescatador(st);

}

break;

case TURN_SR:

next.brujula = (next.brujula + 1) % 8;

break;

case TURN_L:

next.brujula = (next.brujula + 6) % 8;

break;

case RUN:

next = NextCasillaRescatador(st);

bool check1 = false, check2 = false, check3 = false, check4 = false, check5 = false;

check1 = terreno[next.f][next.c] != 'P' and terreno[next.f][next.c] != 'M'; // Nos aseguramos que no es un precipicio ni una montaña

```

check2 = terreno[next.f][next.c] != 'B'; // Sabemos que el
rescatador no puede pasar por casillas de tipo bosque
if (next.f == posicionAuxiliar.first && next.c == posicionAuxiliar.second)
{
    // cout << "No se puede pasar por la casilla del auxiliar" << endl;
    // cout << "F " << next.f << " C " << next.c << endl;
    return st; // No se puede pasar por la casilla del auxiliar
}
next = NextCasillaRescatador(next);
check3 = abs(altura[next.f][next.c] - altura[st.f][st.c]) <= (!st.zapatillas ? 1 : 2); // Comprobamos que la diferencia de altura es menor o igual a 1 si no tiene zapatillas, o menor o igual a 2 si tiene zapatillas
check4 = terreno[next.f][next.c] != 'P' and terreno[next.f][next.c] != 'M'; // Nos aseguramos que no es un precipicio ni una montaña
check5 = terreno[next.f][next.c] != 'B'; // Sabemos que el rescatador no puede pasar por casillas de tipo bosque
if (next.f == posicionAuxiliar.first && next.c == posicionAuxiliar.second)
{
    // cout << "No se puede pasar por la casilla del auxiliar" << endl;
    // cout << "F " << next.f << " C " << next.c << endl;
    return st; // No se puede pasar por la casilla del auxiliar
}

if (check1 and check2 and check3 and check4 and check5)
{
    // cout << "La casilla es accesible DENTRO DE RESCATADOR" << endl;
    return next;
}
else
{
    return st; // Si no es accesible, volvemos al estado original
}
break;
}
// vemos si estamos en una casilla que da las zapatillas
if (terreno[next.f][next.c] == 'D')
{
    next.zapatillas = true;
}
return next;
}

```

```

1 EstadoR ComportamientoRescatador::applyR(Action accion, const EstadoR &st, const vector<vector<unsigned char>> &terreno, const vector<vector<unsigned char>> &altura)
2 {
3     EstadoR next = st;
4     switch (accion)
5     {
6         case WALK:
7             if (CasillaAccesibleRescatador(st, terreno, altura))
8             {
9                 next = NextCasillaRescatador(st);
10            }
11            break;
12        case TURN_N:
13            next.brujula = (next.brujula + 1) % 8;
14            break;
15        case TURN_L:
16            next.brujula = (next.brujula + 6) % 8;
17            break;
18        case RUN:
19            next = NewCasillaRescatador(st);
20            if (check1 == false, check2 == false, check3 == false, check4 == false, check5 == false;
21                check1 = terreno[next.f][next.c] != 'P' and terreno[next.f][next.c] != 'M'; // Nos aseguramos que no es un precipicio ni una montaña
22                check2 = terreno[next.f][next.c] != 'B'; // Sabemos que el rescatador no puede pasar por casillas de tipo bosque
23                if (next.f == posicionAuxiliar.first && next.c == posicionAuxiliar.second)
24                {
25                    // cout << "No se puede pasar por la casilla del auxiliar" << endl;
26                    // cout << "# " << next.f << " " << next.c << endl;
27                    return st; // No se puede pasar por la casilla del auxiliar
28                }
29            next = NextCasillaRescatador(next);
30            check3 = abs(altura[next.f][next.c] - altura[st.f][st.c]) <= (!st.zapatillas ? 1 : 2); // Comprobamos que la diferencia de altura es menor o igual a 1 si no tiene zapatillas, o menor o igual a 2 si tiene zapatillas
31            check4 = terreno[next.f][next.c] != 'P' and terreno[next.f][next.c] != 'M'; // Nos aseguramos que no es un precipicio ni una montaña
32            check5 = terreno[next.f][next.c] != 'B'; // Sabemos que el rescatador no puede pasar por casillas de tipo bosque
33            if (next.f == posicionAuxiliar.first && next.c == posicionAuxiliar.second)
34            {
35                // cout << "No se puede pasar por la casilla del auxiliar" << endl;
36                // cout << "# " << next.f << " " << next.c << endl;
37                return st; // No se puede pasar por la casilla del auxiliar
38            }
39            if (check1 and check2 and check3 and check4 and check5)
40            {
41                // cout << "La casilla es accesible DENTRO DE RESCATADOR" << endl;
42                return next;
43            }
44        else
45        {
46            return st; // Si no es accesible, volvemos al estado original
47        }
48    }
49    break;
50 }
51 // Vemos si estamos en una casilla que da las zapatillas
52 if (terreno[next.f][next.c] == 'D')
53 {
54     next.zapatillas = true;
55 }
56 return next;
57 }
```

```

bool ComportamientoRescatador::CasillaAccesibleRescatador(const EstadoR &st, const
vector<vector<unsigned char>> &terreno, const vector<vector<unsigned char>> &altura)
{
    EstadoR next = NextCasillaRescatador(st);
    // cout << "La casilla accesible en casillaAccesibleRescatador es: " << next.f << " " <<
    next.c << endl;
    bool check1 = false, check2 = false, check3 = false;
    check1 = terreno[next.f][next.c] != 'P' and terreno[next.f][next.c] != 'M'; // Nos
aseguramos que no es un precipicio ni una montaña
    check2 = terreno[next.f][next.c] != 'B'; // Sabemos que el
rescatador no puede pasar por casillas de tipo bosque
    check3 = abs(altura[next.f][next.c] - altura[st.f][st.c]) <= (!st.zapatillas ? 1 : 2); // Comprobamos que la diferencia de altura es menor o igual a 1 si no tiene zapatillas, o
menor o igual a 2 si tiene zapatillas
    bool check4 = false; // Comprobamos que las
casillas esten dentro del mapa
    if (next.f >= 0 && next.f < terreno.size() && next.c >= 0 && next.c < terreno[0].size())
    {
        check4 = true;
    }
    if (terreno[next.f][next.c] == '?')
    {
        check3 = true;
    }
    if (next.f == posicionAuxiliar.first && next.c == posicionAuxiliar.second)
    {
        // cout << "No se puede pasar por la casilla del auxiliar" << endl;
```

```

    // cout << "F " << next.f << " C " << next.c << endl;
    return false; // No se puede pasar por la casilla del auxiliar
}

// cout << "check1: " << check1 << " check2: " << check2 << " check3: " << check3 <<
endl;
return check1 and check2 and check3 and check4; // Comprobamos que la casilla es
accesible
}

```

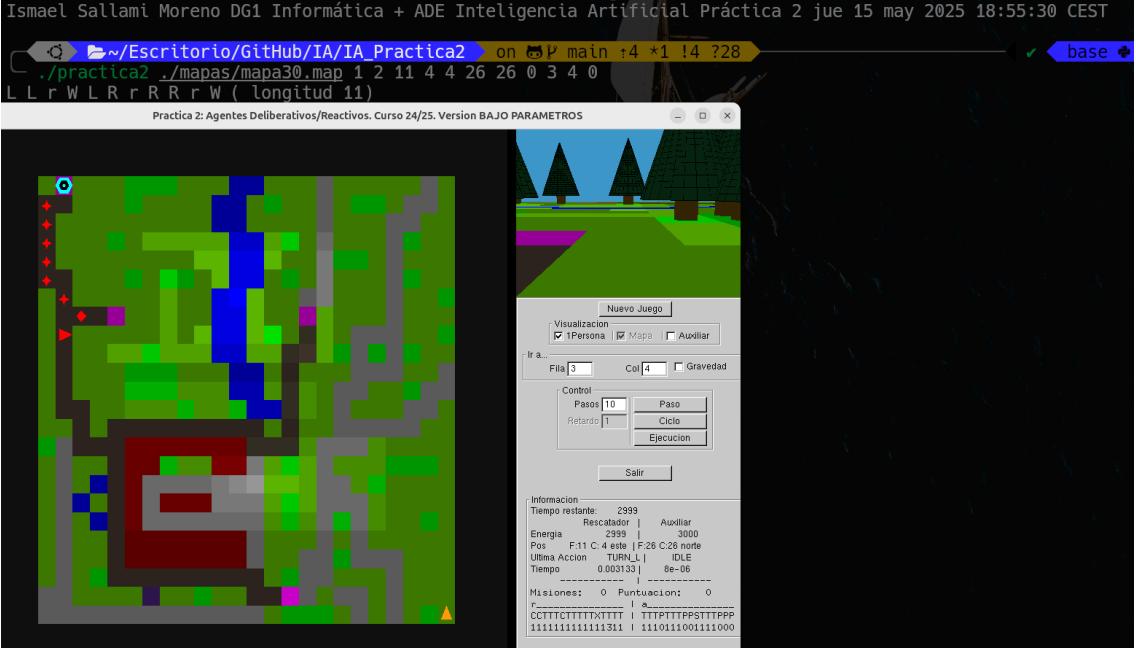
```

1 bool ComportamientoRescatador::CasillaAccesibleRescatador(const EstadoR &st, const vector<vector<unsigned char>> &terreno, const vector<vector<unsigned char>> &altura)
2 {
3     EstadoR next = NextCasillaRescatador(st);
4     // cout << "La casilla accesible en casillaAccesibleRescatador es: " << next.f << " " << next.c << endl;
5     bool check1 = false, check2 = false, check3 = false;
6     check1 = terreno[next.f][next.c] != 'P' and terreno[next.f][next.c] != 'M'; // Nos aseguramos que no es un precipicio ni una montaña
7     check2 = abs(altura[next.f][next.c] - altura[st.f][st.c]) <= (st.zapatillas ? 1 : 2); // Sabemos que el rescatador no puede pasar por casillas de tipo bosque
8     check3 = abs(altura[next.f][next.c] - altura[st.f][st.c]) <= (st.zapatillas ? 1 : 2); // Comprobamos que la diferencia de altura es menor o igual a 1 si no tiene zapatillas, o menor o igual a 2 si tiene zapatillas
9     bool check4 = false;
10    if (next.f >= 0 && next.f < terreno.size() && next.c >= 0 && next.c < terreno[0].size())
11    {
12        check4 = true;
13    }
14    if (terreno[next.f][next.c] == '?')
15    {
16        check4 = true;
17    }
18    if (next.f == posicionAuxiliar.first && next.c == posicionAuxiliar.second)
19    {
20        // cout << "No se puede pasar por la casilla del auxiliar" << endl;
21        // cout << "F " << next.f << " C " << next.c << endl;
22        return false; // No se puede pasar por la casilla del auxiliar
23    }
24
25    // cout << "check1: " << check1 << " check2: " << check2 << " check3: " << check3 << endl;
26    return check1 and check2 and check3 and check4; // comprobamos que la casilla es accesible
27 }

```

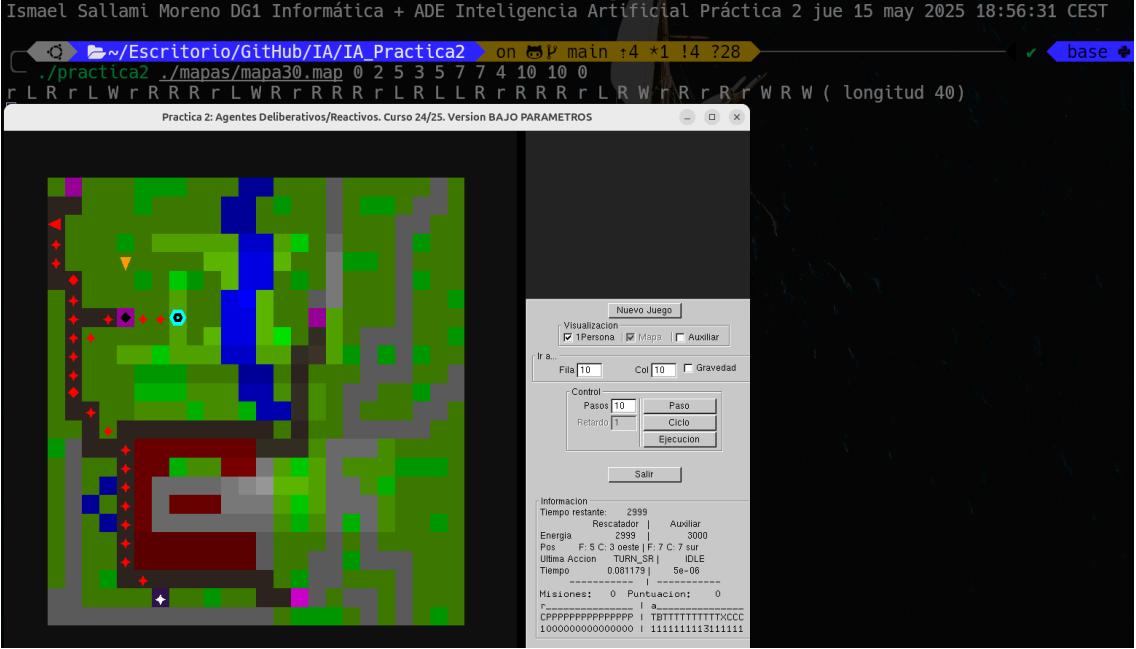
(e) Rellena los datos de la tabla con el resultado de aplicar

./practica2SG ./mapas/mapa30.map 1 2 11 4 4 26 26 0 3 4 0

Screen Shot	 <p>Ismael Sallami Moreno DG1 Informática + ADE Inteligencia Artificial Práctica 2 jue 15 may 2025 18:55:30 CEST ./practica2 ./mapas/mapa30.map 1 2 11 4 4 26 26 0 3 4 0 L L r W L R r R R r W (longitud 11)</p> <p>Practica 2: Agentes Deliberativos/Reactivos. Curso 24/25. Version BAJO PARAMETROS</p> <p>Nuevo Juego</p> <p>Visualización <input checked="" type="checkbox"/> 1Persona <input type="checkbox"/> Mapa <input type="checkbox"/> Auxiliar</p> <p>Ir a... Fila: 3 Col: 4 Gravedad</p> <p>Control</p> <table border="1"><tr><td>Pasos: 10</td><td>Paso</td></tr><tr><td>Retardo: 1</td><td>Ciclo</td></tr><tr><td colspan="2">Ejecucion</td></tr></table> <p>Salir</p> <p>Información</p> <table><tr><td>Tiempo restante:</td><td>2999</td><td>Rescatador </td><td>Auxiliar</td></tr><tr><td>Energía:</td><td>2999</td><td colspan="2">3000</td></tr><tr><td>Pos:</td><td>F:11 C:4 este</td><td> F:26 C:28 norte</td><td></td></tr><tr><td>Última Acción:</td><td>TURN_L</td><td> </td><td>IDLE</td></tr><tr><td>Tiempo:</td><td>0.003133</td><td> </td><td>6e-06</td></tr></table> <p>Misiones: 0 Puntuación: 0</p> <p>R: CTTCTCTTTTXXTTT TTTPTTTPPSTTTPPP 111111111111311 1110111001111000</p>	Pasos: 10	Paso	Retardo: 1	Ciclo	Ejecucion		Tiempo restante:	2999	Rescatador	Auxiliar	Energía:	2999	3000		Pos:	F:11 C:4 este	F:26 C:28 norte		Última Acción:	TURN_L		IDLE	Tiempo:	0.003133		6e-06
Pasos: 10	Paso																										
Retardo: 1	Ciclo																										
Ejecucion																											
Tiempo restante:	2999	Rescatador	Auxiliar																								
Energía:	2999	3000																									
Pos:	F:11 C:4 este	F:26 C:28 norte																									
Última Acción:	TURN_L		IDLE																								
Tiempo:	0.003133		6e-06																								
Longitud del camino (Rescatador)	11																										
Coste de Energía (Rescatador)	11																										

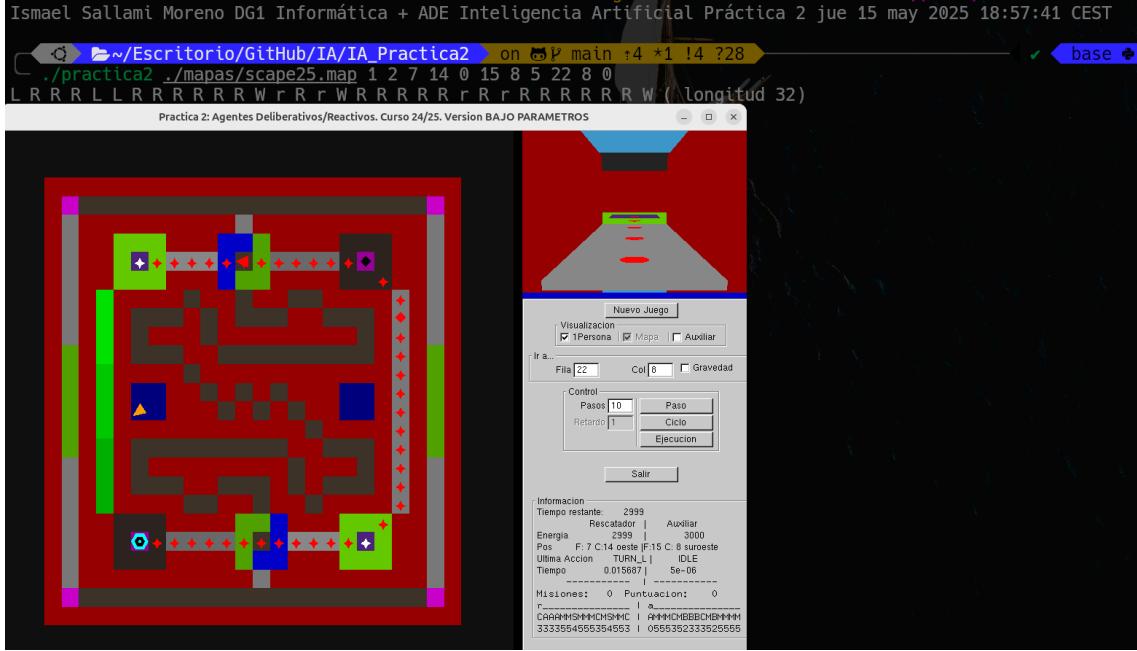
(f) Rellena los datos de la tabla con el resultado de aplicar

./practica2SG ./mapas/mapa30.map 0 2 5 3 5 7 7 4 10 10 0

Screen Shot	
Longitud del camino (Rescatador)	40
Coste de Energía (Rescatador)	64

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

`./practica2SG ./mapas/scape25.map 1 2 7 14 0 15 8 5 22 8 0`

Screen Shot	
Longitud del camino (Rescatador)	32
Coste de Energía (Rescatador)	61

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 es que en el algoritmo A estrella se ordena en vez del coste acumulado, en base a la heurística que se decida, en mi caso fue la distancia del máximo. Por lo que podemos mencionar que el A estrella prioriza los nodos que parecen más prometedores hacia el destino. En este usamos la heurística de Chebyshev (distancia máxima entre filas y columnas para estimar la cercanía al destino). En el Dijkstra se escoge el estado con un coste acumulado menor, mientras que en el de A estrella se coge el de coste Estimado (heurística + coste).

- (b) Copia y pega en el siguiente recuadro de texto la heurística seleccionada. Además, describela y justifica la razón que hace que sea admisible para este problema.

```
● ● ●  
1 int heuristic = max(abs(destino.f - hijo.estado.f), abs(destino.c - hijo.estado.c));
```

Esta heurística calcula la distancia Chebyshev, la cual podemos considerar adecuada para este escenario. Estima el costo restante al destino tomando el máximo entre la diferencia en filas y columnas. Es admisible porque nunca sobreestima el costo real: siempre da el número mínimo de movimientos necesarios. También es consistente(monótona), ya que la diferencia entre heurísticas de nodos vecinos no supera el costo del movimiento (1), cumpliendo la condición $h(n) \leq \text{cost}(n, n') + h(n')$.

- (c) Rellena los datos de la tabla con el resultado de aplicar
./practica2SG ./mapas/mapa75.map 1 3 5 5 2 8 5 2 31 54 0

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

```
./practica2SG ./mapas/2ez.map 0 3 7 7 4 14 16 4 16 16 0
```

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

```
./practica2SG ./mapas/paldea25.map 0 3 82 17 4 16 23 6 34 34 0
```

Nivel 4-Misión de Rescate

(a) Haz una descripción general de tu estrategia general con la que has abordado este nivel. Explica brevemente las razones de esos criterios.

En este caso he hecho uso de ambos agentes.

Rescatador: Si el mapa tiene grandes dimensiones recargo antes ya que es probable que me quede sin energía si decido ir a recargar con la misma que en mapas pequeños. Deshago los giros y cogo las zapatillas si puedo. A continuación añado la parte de recargar energía, que básicamente en mi caso la lógica que he implementado ha sido que si necesita recargar recorra el mapa para ver si ha encontrado casillas 'X' y entonces calcule el camino a la más cercana usando para ello la distancia de Manhattan. A continuación, si no hayPlan se genera usando para ello el algoritmo de A estrella ya que se vió en clase que este era más óptimo debido al uso de heurística. Si en este punto no hay plan, se investiga el caso de trazar una ruta hacia la casilla '?' más cercana. Si nada de esto funciona, vemos que si estamos en la misma casilla durante 3 turnos y no hay gravedad pasamos a buscar de nuevo la casilla '?'. Por último, comprobamos si las acciones son RUN o WALK si estas son válidas, de la misma manera veo si tengo en el siguiente estado al auxiliar y si es así lo tengo en cuenta para elaborar el plan y no chocar con él. Además compruebo si estoy en una casilla donde hay gravedad, si es así la acción que devuelvo es CALL_ON, pero si la última acción es CALL_ON, devuelvo IDLE ya que se supone que estoy esperando a que llegue el auxiliar.

Auxiliar: En cuanto a la recarga, decidí aplicar la misma lógica que en el rescatador. En este caso, para este agente actuamos si el sensor.venpaca es true de manera que es cuando nos ha llamado el rescatador. Pensé que sería buena idea que si el mapa era de pequeñas dimensiones que pruebe a hacer una búsqueda rápida de las zapatillas (una de las opciones es que pase por el centro). Otra lógica implementada que cabe destacar es que si la energía es menor que 500, el mapa es grande y hayPlan, abortamos el plan ya que de esta manera nos aseguramos recargar y que no finalice porque el auxiliar se haya quedado sin energía.

(b) ¿Qué algoritmo o algoritmos de búsqueda usas en el nivel 4? Explica brevemente la razón de tu elección.

Uso el algoritmo A* en el nivel 4 porque permite encontrar caminos óptimos de forma eficiente combinando el costo acumulado y una heurística. Utilizo varias heurísticas según el entorno, pero la más destacada es la distancia Chebyshev (el máximo entre diferencias de filas y columnas), ya que es admisible y consistente cuando se permiten movimientos en ocho direcciones con el mismo costo, lo que garantiza optimalidad.

(c) ¿Bajo qué condiciones replanifica tu agente?

Rescatador: cuando necesita recargar, cuando no hay plan hacia el destino y tiene casillas '?' al alcance, si estamos 10 veces en la misma posición.

Auxiliar: las mismas casuísticas que el rescatador, además de que si tiene poca energía, el mapa es grande y hay plan, entonces decidimos recalcular el camino para recargar.

- (d) Explica el valor de coste que le has dado a la casilla desconocida en la construcción de planes cuando el mapa contiene casillas aun sin conocer. Justifica ese valor.

En mi caso tengo funciones que me dan el coste en base al terreno y a la altura, al ser la casilla ‘?’ distinta a las que tengo se asigna el coste de default que es 1, de esta manera en el primer plan se calcula un plan sin conocer nada del mapa (normalmente en línea recta), cuando va hacia esa casilla si ve que no puede se recalcula el camino. Gracias a esta manera siempre va a intentar ir de manera directa al destino.

- (e) ¿Has tenido en cuenta la recarga de energía? En caso afirmativo, describe la política usada por los agentes.

Sí la he tenido en cuenta, en ambos aplico la misma lógica, solo varía el tema en el que el mapa es grande, hayPlan y la energía es menor a una cierta cantidad en el auxiliar.

La lógica que se ha seguido ha sido la siguiente: he usado dos variables: “recargaEnergia” que indica la necesidad de recargar y “estoyRecargando” que indica si estoy ya recargando. Así que lo que hago es que si la energía es el máximo y tenía que recargar energía, ponlo la variable recarEnergia a false. Si la energía es menor que cuando debo de recargar pues pongo la variable recarga energía a true y recorro el mapa para ver si he encontrado casillas de recarga, si es así lo añado a una estructura que almacena esa casilla. Si la casilla actual es ‘X’ pongo la variable estoyRecargando a true, si no miro si hay casillas de recarga en dicha estructura que las almacena y trazo un camino hacia la más cercana usando para ello la distancia de Manhattan.

- (f) Añade aquí todos los comentarios que deseas sobre el trabajo que has desarrollado sobre este nivel, qué consideras son importantes para evaluar el grado en el que te has implicado en la práctica y que no se puede deducir de la contestación a las preguntas anteriores.

Bajo mi punto de vista, el grado de implicación hacia la práctica ha sido elevado de ahí a pensar diversas ideas y limitar el uso de IA generativa para poder idear una solución por mi propia cuenta. Además cabe destacar que:

- He dedicado tiempo a probar distintas heurísticas y analizar su impacto en el rendimiento del algoritmo, no solo en cuanto a optimalidad, sino también en velocidad de búsqueda.
- Me he esforzado por entender bien cómo afectan las propiedades de las heurísticas (admisibilidad, consistencia) al comportamiento del algoritmo A*, y he ajustado mi implementación en función de ello.
- He realizado múltiples pruebas con distintos escenarios y configuraciones del mapa para asegurar que la solución sea robusta y se adapte bien a diferentes

tipos de niveles.

- He trabajado en mantener el código limpio y modular, lo que me ha permitido probar cambios más fácilmente y entender mejor el funcionamiento del sistema.
- Aunque algunas decisiones de diseño no se ven directamente en las respuestas anteriores, han requerido tiempo de análisis y prueba-error para llegar a una solución eficaz.

En conclusión, creo que me he esforzado bastante en la práctica intentando entender cada parte del código que añadía.

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

```
./practica2SG ./mapas/mapa50.map 1 4 28 25 4 28 20 2 36 23 0 39 8 0  
46 26 1 39 34 0 26 37 0 18 46 0 3 46 0 3 3 0 10 17 1 39 45 0 9 16 0 38  
13 0 27 23 0 31 18 0 45 31 0 35 7 0 12 6 1 40 7 0 20 6 1 10 25 1 41 30  
0 14 31 0 26 24 1 38 26 1 38 20 1 44 14 0 17 40 0 45 3 1 4 9 0 33 44 0  
17 3 1 3 11 0 42 13 1 26 18 1 38 25 1 33 26 0 46 46 1 36 14 0 36 31 1  
17 34 0 8 22 1 44 41 1 16 11 0 44 17 0 29 32 0 42 21 0 46 19 1 40 34 0  
45 24 0 46 7 0 44 32 1 21 30 1 14 39 1 15 22 1 11 9 0 13 27 1 20 8 1  
45 5 0
```

Instantes de simulación no consumidos: 2599

Tiempo Consumido: 0.242248

Nivel Final de Energía (Rescatador): 1239

Nivel Final de Energía (Auxiliar): 0

Objetivos encontrados: (8) 21

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

```
./practica2SG ./mapas/mapa100.map 1 4 63 31 6 63 32 2 66 40 0 75 24  
0 85 36 1 83 6 0 60 10 0 33 11 0 84 7 0 86 40 0 68 77 1 79 91 0 19 33  
0 76 25 0 55 47 0 62 36 0 51 95 0 91 63 0 71 14 1 24 13 0 80 15 1 21  
51 1 83 61 0 29 63 0 52 49 1 78 52 1 76 40 1 90 28 0 39 80 0 91 6 1 94  
52 0 8 19 0 66 89 1 34 6 0 6 23 1 85 26 1 53 37 1 79 51 0 70 53 1 3 43  
0
```

Instantes de simulación no consumidos	1016
Tiempo Consumido	0.305928
Nivel Final de Energía (Rescatador)	0
Nivel Final de Energía (Auxiliar)	1379
Objetivos encontrados	(15) 40

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

```
./practica2SG ./mapas/F_islas.map 1 4 47 53 2 49 53 2 41 56 0 52 53 0
74 54 1 74 47 0 46 42 0 71 56 0 83 52 0 58 65 0 85 43 1 92 39 0 81 68
0 91 48 0 21 95 0 92 14 0 88 64 0 43 61 0 28 78 1 30 44 0 22 18 1 27
55 1 41 16 0 90 10 0 12 49 1 76 68 1 38 74 1
```

Instantes de simulación no consumidos	0
Tiempo Consumido	6.22396
Nivel Final de Energía (Rescatador)	1777
Nivel Final de Energía (Auxiliar)	2473
Objetivos encontrados	(35) 145

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.

Quiero agradecer al profesor de prácticas su implicación y disponibilidad, especialmente a través de Telegram, donde siempre respondió cuando le fue posible. Su apoyo ha hecho que el desarrollo de la práctica fuera más ameno, enriquecedor y accesible. También me gustaría destacar que esta primera toma de contacto con la práctica me ha resultado muy interesante, ya que se ha planteado a través de algoritmos sencillos y comprensibles, lo que facilita mucho la introducción a los conceptos clave de la Inteligencia Artificial y despierta el interés por la asignatura.