INTELIGENCIA ARTIFICIAL

CURSO 2019-20

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

APELLIDOS Y NOMBRE	Nikita Stetskiy		
GRUPO TEORÍA	B	**GRUPO PRÁCTICAS**	B 3

Instrucciones iniciales

Poner en los recuadros la información que se solicita. En los casos en los que se solicita una captura de pantalla (ScreenShot) [apartado (a) del Anchura y apartado (c) del Costo Uniforme], extraer esa imagen de la ejecución concreta pedida, donde aparezca la línea de puntos que marca el camino. Además, en dicha captura debe aparecer al menos el nombre del alumno. Ejemplos de imágenes se pueden encontrar en Imagen1 y en Imagen2.

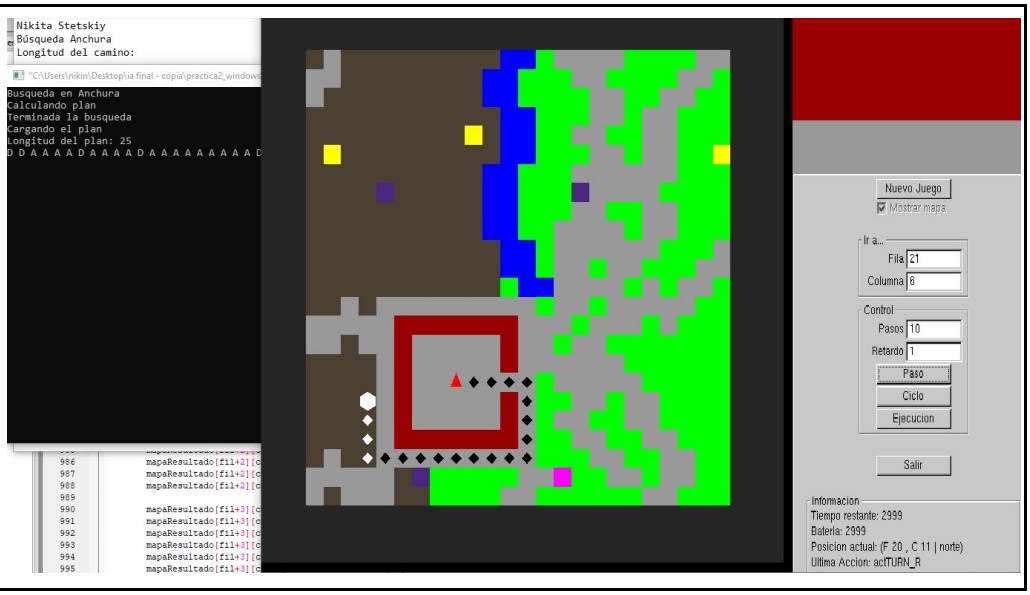
Indica el nivel máximo abordado (Nivel 1-Anchura, Nivel 1-Coste o Nivel 2):

Nivel 2 - He explicado algunas mejoras de la versión que he mandado por correo. La única mejora diferencial es la planificación (La he marcado con un *).

Nivel 1-Anchura: Usa tu implementación del algoritmo de búsqueda en anchura en el mapa30 y dinos qué planes obtiene ante esta situación:

(a) Posición Inicial del agente: Fila 20, Columna 11, Orientación oeste y Posición Objetivo de Fila 21 y Columna 6.

Longitud del Plan	25
Plan	D D A A A A D A A A A D A A A A A A A A A D A A A

ScreenShot	

Nivel 1-Coste: Usa tu implementación del algoritmo de búsqueda de coste uniforme en el mapa30 y dinos qué planes obtienes ante estas tres situaciones:

(a) Posición Inicial del agente: Fila 20, Columna 11, Orientación oeste y Posición Objetivo de Fila 21 y Columna 6.

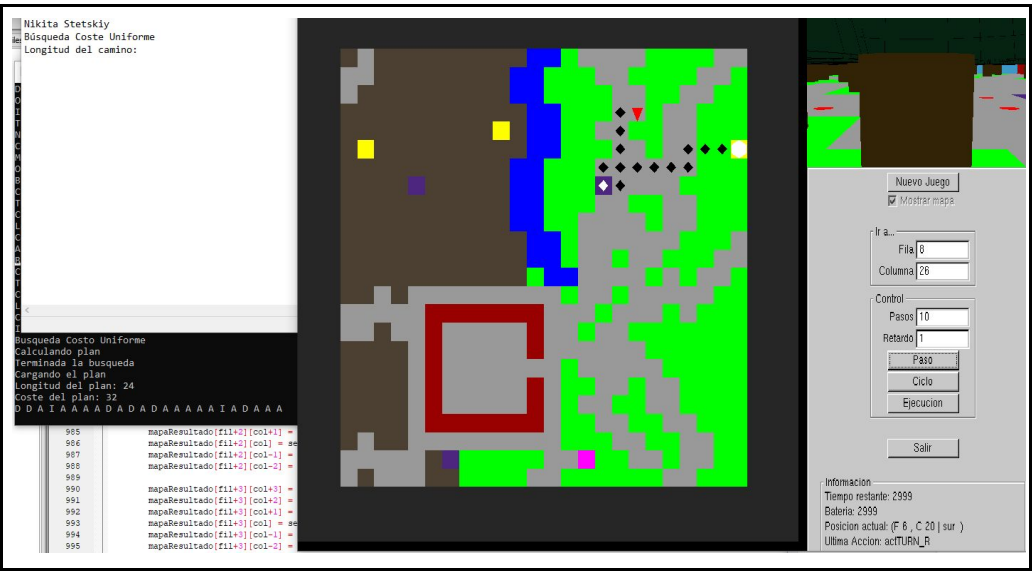
Longitud del Plan	26
Coste del Plan	26
Plan	D D A A A A D A A A A D A A A A A A A A D A A A I A

(b) Posición Inicial del agente: Fila 6, Columna 11, Orientación este y Posición Objetivo de Fila 6 y Columna 15.

Longitud del Plan	10
Coste del Plan	25
Plan	A D A D D A D A A A

(c) Posición Inicial del agente: Fila 6, Columna 20, Orientación este y Posición Objetivo de Fila 8 y Columna 26.

Longitud del Plan	24
Coste del Plan	32
Plan	D D A I A A A A D A D A D A A A A A I A D A A A

ScreenShot	

Nivel 2-Reto: Responde con brevedad a las siguientes preguntas de acuerdo a como lo hayas hecho en la implementación de tu práctica:

(a) ¿Qué algoritmo de búsqueda usas en el nivel 2-Reto?

He utilizado el algoritmo A estrella, ya que al utilizar una heurística donde se usa una combinación entre búsquedas del tipo primero en anchura ‘h(n)’ con primero en profundidad ‘g(n)’ se consigue una gran mejora de tiempo conforme al algoritmo Coste Uniforme.

A* mantiene dos estructuras de datos auxiliares, que podemos denominar abiertos, implementado como una cola de prioridad y cerrados, donde se guarda la información de los nodos que ya han sido visitados. En cada paso del algoritmo, se expande el nodo que esté primero en abiertos, y en caso de que no sea un nodo objetivo, calcula la f(n) de todos sus hijos, los inserta en abiertos, y pasa el nodo evaluado a cerrados.

(b) ¿Has incluido dentro del algoritmo de búsqueda usado en el nivel 2-Reto que si pasas por una casilla que da las zapatillas o el bikini, considere en todos los estados descendientes de él que tiene las zapatillas y/o el bikini?

Por supuesto, utilizo varias estructuras de booleanos donde se compara el estado de los nodos. El struct de ComparaEstadosAbsolutos tiene la función de establecer un orden de prioridad. Aquí se añade la comprobación de bikini y zapatillas al anterior struct ComparaEstados.

A los nodos se les ha añadido el atributo de coste y el bool de zapatillas y bikini. El bool se inicia al principio conforme a los atributos que tiene declarados en las variables de estado.
Al comprobar un nodo que pasa por zapatillas o bikini, se declara como true el bool correspondiente en el estado y en el atributo del nodo. Cuando el agente pase realmente por la casilla (este cambia el atributo privado mediante la función correspondiente de think).

Los bools de struct de estado existe para CompararEstados, como anteriormente se ha explicado. Es decir, que si se llegan a cambiar los estados, se le asignará diferente orden de prioridad.

(c) ¿Qué estructura de datos has utilizado para implementar la lista de abiertos y por qué?

priority_queue<nodo3, vector<nodo3>, CompararCosteAbsoluto> cola;

// Lista de Abiertos

Como podemos comprobar, para la cola de abiertos he utilizado una cola con prioridad para que se ordene automáticamente la cola de manera eficiente.

(d) ¿Qué estructura de datos has utilizado para implementar la lista de cerrados y por qué?

set<estado,ComparaEstadosAbsoluto> generados;

// Lista de Cerrados

Como se puede ver he utilizado un set para la lista de los cerrados, como se ha enseñado en el de profundidad.

(e) ¿Bajo qué condiciones has planteado replanificar?

He probado distintos métodos de replanificacion. Primero he pensado en replanificar todas las veces cuando el agente vea una casilla de Agua o Bosque y tenga que pasar por ahí, pero me he dado cuenta de que se vuelve muy ineficiente ya que tarda mucho en mapas grandes en replanificarlo todo. Por lo que después he pensado hacerlo cada dos turnos mediante un contador o que a los 1000 turnos deje de replanificar. Estos eran mejoras visibles, aunque no del todo eficiente.

* La mejora más visible ha sido cuando he implementado que recalcule cuando vea que la casilla [2] sea Agua o Bosque y que las casillas alrededor sean diferentes y las que están delante de los sensores sean ‘?’, es decir, que no están exploradas. Por lo que no recalcula si está en el océano o cuando ya está explorada la zona. Solo recalcula cuando la zona de delante tiene agua o bosque, las casillas de alrededor son diferentes y las de delante están sin explorar.

(f) ¿Qué valor le has dado a la casilla desconocida en la construcción de planes que se enfrentaban a mapas con casillas aún sin conocer?

Al principio he pensado que el valor fuera 3, ya que tiene que ser mayor que el valor por defecto y menor que las casillas grandes, como el agua y los bosques (con bikini y zapatillas). Finalmente lo cambié a 4 para que sea más propenso que pasara por casillas que ya conoce como los caminos.

(g) ¿Has tenido en cuenta la recarga de batería? En caso afirmativo, describe de qué manera la tienes en cuenta.

Por supuesto, esto es algo que he tenido muy en cuenta para poder hacer +25 objetivos en los mapas grandes. Paso por batería en tres ocasiones.

Cuando estoy cerca (mediante la H de A*) veo si la batería es menor que el tiempo, pues recargo hasta que sea el doble que el tiempo. Con cerca me refiero a un radio de 4.

Cuando me pilla de camino (mediante la G de A*), es decir, cuando el camino de la batería hacia el objetivo es menor que el mio, puedo pasar por la batería y recargar. Esto también lo hago cuando tengo la batería menor que el tiempo. Además para que sea más eficiente solo lo calculo en un radio de 25 de la batería y de mi, ya que sino los cálculos se hacen muy ineficientes.

Finalmente cuando tengo la bateria por debajo del coste del camino hacia el destino + el coste del camino hacia la batería. Me es preferible ir directamente hacia la bateria y recargar y esto lo hago claramente cuando me es preferible ir calculandolo con el tiempo, es decir, que si me es preferible ir al objetivo porque voy a morir si voy a la batería, pues irá al objetivo.

Esto lo hago cada dos turnos ya que calcular muchos algoritmos A * se vuelve muy ineficiente.