PRÁCTICA DE SISTEMAS INTELIGENTES

Autor: Jose Miguel Gómez Lozano
Correo: Josemygel@gmail.com
DNI: 48833504L

ÍNDICE

Autor: Jose Miguel Gómez Lozano
Correo: Josemygel@gmail.com
DNI: 48833504L
Fecha: 15/10/2018
Asignatura: Sistemas Inteligentes

PSEUDOCÓDIGO

```
/**
*       NOTA:
*       f() = g() + h(objetivo)
*       g() Es el coste desde el caballero hasta la posición seleccionada
*       h(objetivo) Es el coste desde la posición seleccionada, hasta el objetivo
*/

Asignamos coste en posición del caballero a 0
Creamos una lista interior
Creamos una lista frontera con únicamente el punto inicial (Nodo del Caballero)

WHILE ( La lista frontera no esté vacía )
        Elegimos un nodo en la lista frontera, el cual, su f() de menor valor
        Lo quitamos de la lista frontera y lo insertamos en la lista interior

        IF ( El nodo seleccionado es el objetivo )
                END WHILE
        END IF

        FOR ( Las casillas adyacentes (M) a la posición seleccionada (N) )
                El padre de M es N

                IF ( M es accesible (No es 'p' ni 'k' ni 'b') )
                        IF ( M está en lista interior o en lista frontera )
                                IF ( padre de M con g() mayor que el g() de N )
                                        padre de M es ahora N
                                END IF
                        ELSE IF ( M no está en lista frontera )
                                insertamos M en lista frontera
                        END IF
                END IF
        END FOR
END WHILE
```

• Pseudocódigo del algoritmo A* que se ha implementado. Explicando con ejemplos cada uno de los posibles casos que nos podemos encontrar a la hora de evaluar nuevos nodos.

Autor: Jose Miguel Gómez Lozano
Correo: Josemygel@gmail.com
DNI: 48833504L
Fecha: 15/10/2018
Asignatura: Sistemas Inteligentes

HEURÍSTICA

Cuando h se aleja de h* se expanden muchos nodos, como por ejemplo se podría usar h con valor 0, que expande una cantidad excesiva de nodos.

Cuando h se acerca a h* pasa lo opuesto, podemos observar que el valor de los Nodos disminuye bastante.

Para esta práctica hemos seleccionado 4 heurísticas:

h0: Esta heurística, según las pruebas realizadas encuentra el valor óptimo, por lo que es admisible en ese aspecto, el problema está en que realiza una expansión de nodos excesiva.

hManhattan: Esta heurística **NO** es admisible, ya que resuelve con un camino el cual no es el óptimo.

hEuclidea: Esta heurística es buena, pues encuentra el camino óptimo, pero sigue teniendo una cantidad de nodos excesiva.

hCustom: Esta heurística se basa en que, trabajando en un mapa hexagonal, realizamos un camino **lo más recto posible** hacia el objetivo, contando así la cantidad de casillas por recorrer hasta llegar al objetivo. Por las pruebas realizadas, y los resultados obtenidos, esta es la mejor heurística, expande bastantes nodos menos que las anteriores encontrando un camino óptimo.

hCube: Esta heurística se basa en calcular la distancia. Es tan válida como **hCustom** puesto que ofrece la **MISMA** solución al fin y al cabo, solo que con menor coste de cálculo (siempre y cuando no se guarde la distancia ya calculada). Pero obviando esto, a nivel de eficacia son igualmente válidas.

Conclusión: Entre todas las heurísticas analizadas, la candidata con mayor eficacia es la de **hCube**, pero personalmente, me parece mucho más díficil de entender que **hCustom**, la cual cabe destacar que una vez se ha calculado la distancia, no necesita realizar el cálculo ya que se guarda en una variable. En un entorno profesional, me decantaría por **hCustom**, por estas razones, ya que por lo que he leído e investigado, y **hCube** está demasiado enrevesado para hacer **lo mismo** que **hCustom** al fin y al cabo.

Autor: Jose Miguel Gómez Lozano
Correo: Josemygel@gmail.com
DNI: 48833504L
Fecha: 15/10/2018
Asignatura: Sistemas Inteligentes

EXPLICACIÓN

Supongamos que la posición superior-izquierda de este resumido mapa es la posición 0,0 y el mapa es de 6x7 como se puede observar. Entonces, las casillas adyacentes son 3 bloques (inaccesible), 1 zona con camino (coste 1), 1 zona con agua (coste 3) y 1 zona con hierba (coste 2).

En el algoritmo, se expande el nodo que hay en la lista frontera al comienzo, este es el nodo que contiene al caballero. Los nodos que pasan a formar parte de la lista son los adyacentes al nodo expandido y LOS CUALES son accesibles, es decir, las casillas que no contienen un bloque, una piedra o al caballero.
La lista frontera ahora contiene (siendo los parámetros X,Y y la letra que representa la zona) las coordenadas *A*(1,1,c), *B*(2,2,a) y *C*(3,1,h) (empezando las coordenadas por 0,0).

De las tres casillas, ahora toca seleccionar cual expandir, se selecciona la que tiene un menor f() = g() + h(). *A* tiene una g() de 1 y un h() de 3, es decir, un f() de 4. *B* tiene un f() de 5(3+2) y *C* tiene un f() de 4(2+2).

El algoritmo está diseñado para coger el primero que se insertó en la lista frontera, por lo que cogeríamos el *A* para expandir. Ahora, vamos a expandir el nodo *A*, por lo que añadiremos los nodos adyacentes a este, quedando en la lista interior el nodo que contenía el caballero y el nodo *A*.

Procedemos a añadir solo el nodo *D*(1,2,c), ya que el resto está ya añadido o no es accesible. *D* contiene un f() de 5 (2+3).

Tenemos en la frontera ahora a *B*, *C*, y *D*.
Expandimos ahora *C,* cuyo único nodo accesible es *E*(1,4,a) con un f() de 7(5+2).
Tenemos en la frontera ahora a *B*, *D* y *E*, con f() de 5, 5 y 7 respectivamente.
Expandimos ahora *B*, cuyo único nodo accesible y que NO está en la lista frontera ni en la interior es *F*(3,2,c) con f() 6 (4+2).

Expandimos ahora *D*, y en esta iteración topamos con *F*, nodo de la lista frontera, pero que cuyo f(), si se le accediera desde *D*, ahora sería 5(3+2), por lo que se actualiza su nodo padre.

Ahora tenemos en la lista frontera *E* y *F*, y en la lista interior tenemos *A, B, C* y *D*.
En la siguiente comprobación, entre *E* y *F* seleccionamos la que menor f(), que es *F* con 5.

Los nodos que son accesibles, y no están en ninguna lista son *G*(3,1,c) y *H*(5,3,c) con f() de 7(4+3) y 6(4+2) respectivamente.

Ahora en la lista frontera tenemos *E, G* y *H*, de los cuales el menor f() pertenece a *H*.
Procedemos a expandir *H*, obteniendo dos nodos no explorados y accesibles, los cuales serían *I*(1,4,c) y *J*(3,4,c) sin actualizar ningún nodo, cuya f() valen 8(5+3) y 7(5+2).

En la lista interior (nodos ya expandidos) contenemos *A, B, C, D, F* y *H*, y en la lista frontera (nodos adyacentes a nodos expandidos) contenemos *E, G, I* y *J*, de los cuales el menor f() pertenece a *E* con f() de 7.
Tras expandir el nodo *E,* solo obtenemos un nodo a añadir a la lista frontera, el cual es *K*(1,5,h) con f() de 9(7+2).

El siguiente a expandir es *G*, pero no se obtiene nada significativo, seguidamente es *J* cuyo f() es 7. Solamente se añade un nodo a la lista frontera, el cual es *L*(4,4,c) cuyo f() es 7(6+1).

El siguiente nodo a expandir es *L*, cuyos nodos adyacentes son M(3,5,h), N(4,5,c) y O(3,4,d), siendo este último el objetivo, y tienen un f() de 10(7+3), 9(8+1) y 7(7+0).

En la lista frontera, ahora contenemos *I(8), K(9), M(10), N(9)* y *O(7)*, de estos, el que tiene menor f() y se introdujo antes a la lista frontera es O, pero como es el objetivo finalizamos el algoritmo *A Estrella*.

Podemos observar viajando entre los nodos de la lista interior, comenzando por el nodo que está en la coordenada del objetivo, y su padre, y el padre de su padre, y así consecutivamente hasta llegar al caballero, que queda el camino que podemos observar en la imagen.

Autor: Jose Miguel Gómez Lozano
Correo: Josemygel@gmail.com
DNI: 48833504L

Fecha: 15/10/2018
Asignatura: Sistemas Inteligentes

PRUEBAS

Hemos realizado varios mapas para pruebas, desde mapas en miniatura hasta mapas normales. Podemos observar que el algoritmo funciona correctamente, y que los casos los resuelve sin problema alguno.

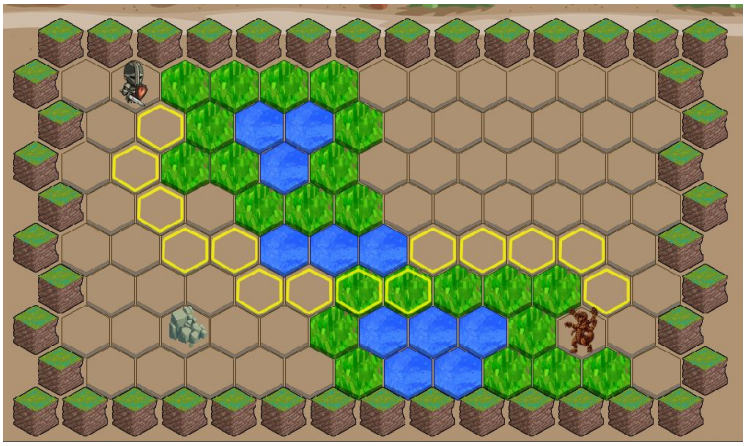
Alguno de los mapas son los siguientes.

Mini-mapa

```
b b b b b b b b b b b b b b
b c c h h h h c c c c c c b
b c c h a a h c c c c c c b
b c c h h a h c b b b b b b
b c c c h h h c b c c c c b
b c c c c a a c b k a c c b
b c c c c c h c b h p p c b
b c c p c c h c b a p d c b
b c c c c c h c b h h h c b
b b b b b b b b b b b b b b
```

Mapa de ejemplo en la documentación

```
b b b b b b b b b b b b b b
b c k h h h h c c c c c c b
b c c h a a h c c c c c c b
b c c h h a h c c c c c c b
b c c c h h h c c c c c c b
b c c c c a a a c c c c c b
b c c c c c h h h h h c c b
b c c p c c h a a a h d c b
b c c c c c h a a h h h c b
b b b b b b b b b b b b b b
```

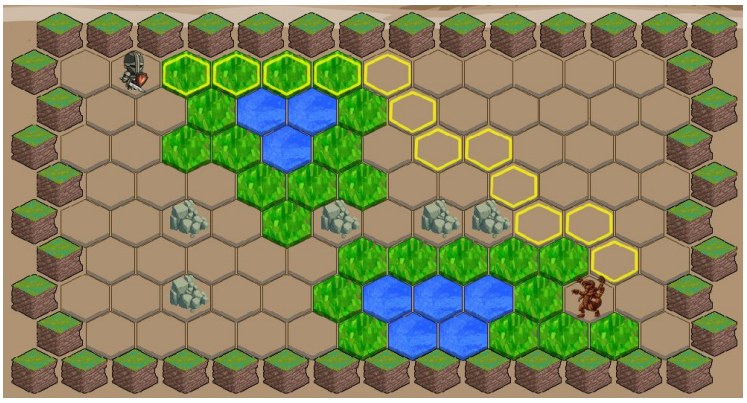


Autor: Jose Miguel Gómez Lozano
Correo: Josemygel@gmail.com
DNI: 48833504L

Fecha: 15/10/2018
Asignatura: Sistemas Inteligentes

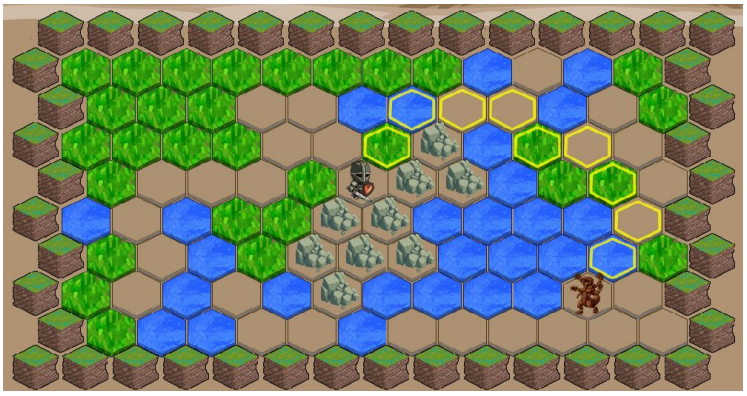
Mapa de prueba adicional 1

b b b b b b b b b b b b b b
b c k h h h h c c c c c c b
b c c h a a h c c c c c c b
b c c h h a h c c c c c c b
b c c c h h h c c c c c c b
b c c p c h p c p p c c c b
b c c c c c h h h h h c c b
b c c p c c h a a a h d c b
b c c c c c h a a h h h c b
b b b b b b b b b b b b b b



Mapa de prueba adicional 2

b b b b b b b b b b b b b b
b c k h h h h c c c c c c b
b c c h a a h c c c c c c b
b c c h h a h c c c c c c b
b c c c h h h c c c c c c b
b c c p c h p c p p c c c b
b c c c c c h h h h h c c b
b c c p c c h a a a h d c b
b c c c c c h a a h h h c b
b b b b b b b b b b b b b b



Autor: Jose Miguel Gómez Lozano
Correo: Josemygel@gmail.com
DNI: 48833504L
Fecha: 15/10/2018
Asignatura: Sistemas Inteligentes

CONCLUSIÓN

Como conclusión final, podemos decir que el algoritmo A* es muy útil para evitar explorar una cantidad excesiva de nodos, obteniendo un algoritmo liviano a la par que eficaz.
Sin duda un algoritmo basado únicamente en *backtracking* sería mucho más costoso y lento.

A nivel personal, lo más complicado de la práctica ha sido comprender los cambios en las casillas por el uso de un mapa hexagonal, ya que nunca me topé con un problema de dicho ámbito.

Autor: Jose Miguel Gómez Lozano
Correo: Josemygel@gmail.com
DNI: 48833504L

BIBLIOGRAFÍA

Enlaces:
StackOverflow (Para entender el funcionamiento en hCube)
RedBlobGames (Investigación sobre la heurística hCube)
Wikipedia (Investigación sobre las casillas hexagonales)
Investigación sobre Algoritmo Estrella

Además de la documentación ofrecida en clase.