línea horizontal

GUIÓN 4

Juegos

línea horizontal  
Inteligencia Artificial

# 

Manuel Ruiz Fernández **26518655T**

*mrf00020@red.ujaen.es*

Juan Bautista Muñoz Ruiz **26516720C**

*jbmr0001@red.ujaen.es*

**MINIMAX**

**Descripción de la estrategia:**

Llamamos a la función generaHijos2(). En esta función, si no se trata de un nodo final, un nodo que no permite más inserciones, llamamos a la función HijoEnColumna. Esta función añade a la Lista Enlazada de hijos del nodo actual un hijo por cada columna en la que se puede poner una ficha.

Posteriormente por cada hijo creado hacemos una llamada a la función generaHijos2() creando así el árbol minimax de forma completa.

La función HijoEnColumna() recorre todas las columnas del tablero y en caso de que se pueda realizar una inserción crearemos un hijo con la ficha insertada en dicha columna. Esta función será controlada por dos booleanos uno para impedir que se cree un hijo con más de una ficha añadida y otro para comprobar si hay espacio.

**MINIMAX RESTRINGIDO**

**Descripción de la estrategia:**

Es el mismo procedimiento solo que limitamos la creación de hijos a una profundidad mediante la variable generaciones, por lo que hará el siguiente procedimiento hasta la profundidad marcada:

Llamamos a la función generaHijos2(). En esta función, si no se trata de un nodo final (un nodo que no permite más inserciones o un nodo cuya variable profundidad haya llegado al límite ) llamamos a la función HijoEnColumna().

Esta función añade a la Lista Enlazada de hijos del nodo actual un hijo por cada columna en la que se puede poner una ficha.

Posteriormente por cada hijo creado hacemos una llamada a la función generaHijos2() creando así el árbol minimax de forma completa.

En caso contrario, se trata de un nodo hoja. Por lo que llamamos a la función heurística la cual calcula el valor del nodo en base a las siguientes funciones:

**valorVertical():** En un doble for recorremos todas las columnas. Por cada columna actualizamos los contadores de fichas seguidas. De esta manera por cada ficha de un mismo jugador en esa columna se incrementa las seguidas ese jugador. En el momento en el que se cambie de jugador se resetea el contador y seguimos con el contador del otro jugador. Al acabar la ejecución calculamos el valor multiplicando el jugador que tiene la última ficha por 10 a la potencia de sus seguidas y se lo sumamos al valor de nuestro nodo.

**valorHorizontal(int jug):** En primer lugar, se inicializan los contadores y un booleano que nos permitirá salir de la función en caso de que la fila actual esté vacía.

Recorreremos el tablero fila a fila, de abajo a arriba, primero recorreremos la fila actual buscando si hay alguna ficha del jugador actual para parar o no.

En caso de continuar, se recorrerá la fila en 4 tramos, del valor 0-4 del valor 1-5, del valor 2-6 y del 3-7, contando los huecos y fichas del jugador jug. En caso de encontrar una ficha del jugador contrario se parará de contar pues será imposible ganar en ese tramo. Finalmente se le sumará al valor val que devolveremos, el número de fichas del tramo recorrido elevado a 10.

**valorOblicuo():** Recorremos la matriz de la misma forma que el oblicuo del checkwin por cada columna. De izquierda a derecha y de derecha a izquierda. Posteriormente por cada fila obtenida en los dos recorridos diagonales en la matriz calculamos el valor en función de la suma de seguidos más huecos vacíos. Si esta suma es mayor o igual a cuatro y la última ficha no es un calculamos el valor multiplicando el jugador que tiene la última ficha por 10 a la potencia de sus seguidas y se lo sumamos al valor de nuestro nodo.

**Minimax**

Para la realización del minimax, en primer lugar se llamará a la función Minimax(Estado e), desde el método jugada con el estado actual. Este, devolverá la posición de la columna en la que se añadirá la ficha actual. Dentro del minimax, llamaremos a la función Max del estado de e. Esta devolverá un pair con el valor de la heurística mejor y la posición del estado hijo en el índice de la lista enlazada de estados de e. De estado hijo sacaremos la posición de la columna donde añadió la ficha al ser creado.

Max(estado e) y Mini(estado e): Devuelve un pair con la heurística mejor y el lugar donde está el estado hijo con esta heurística. Recorrerá todos sus estados hijo y si son nodos hoja se guardará su heurística en una lista enlazada. En caso de no ser nodo hoja, llamará a la función Mini o Max (la contraria a la actual) con el estado hijo actual guardando en la lista enlazada el valor heurístico devuelto por el pair. Tras esto, recorreremos la lista enlazada en busca del valor mayor/menor de esta y lo guardaremos y su posición dentro de la lista también. Estos serán devueltos en el pair.

**PODA ALFA BETA**

**Descripción de la estrategia:**

Seguimos la misma estrategia que en el minimax restringido solo que añadiendo las variables alfa y beta en las llamadas a la función. Por cada ejecución max igualamos la alfa al valor max, si la beta es menor o igual que la alfa realizamos un break. Por cada ejecución min igualamos la alfa al valor min, si la beta es menor o igual que la alfa realizamos un break.

**Ejecuciones 4 niveles de profundidad:**

| **Ejecución** | **Posición de casilla inicial** | **Movimientos** | **Ganador** |
| --- | --- | --- | --- |
| 1 | 0 | 20 | Humano |
| 2 | 3 | 21 | IA |
| 3 | 1 | 18 | Humano |

**Ejecuciones 5 niveles de profundidad:**

| **Ejecución** | **Posición de casilla inicial** | **Movimientos** | **Ganador** |
| --- | --- | --- | --- |
| 1 | 2 | 28 | Humano |
| 2 | 3 | 17 | IA |
| 3 | 1 |  | Empate |

**Ejecuciones 6 niveles de profundidad:**

| **Ejecución** | **Posición de casilla inicial** | **Movimientos** | **Ganador** |
| --- | --- | --- | --- |
| 1 | 2 | 19 | IA |
| 2 | 3 | 25 | IA |
| 3 | 4 | 14 | IA |

**Ejecuciones 7 niveles de profundidad:**

| **Ejecución** | **Posición de casilla inicial** | **Movimientos** | **Ganador** |
| --- | --- | --- | --- |
| 1 | 1 | 13 | IA |
| 2 | 6 | 18 | IA |
| 3 | 0 | 25 | Humano |

**Ejecuciones 8 niveles de profundidad:**

**Produce problema de: java.lang.OutOfMemoryError: Java heap space**

| **Ejecución** | **Posición de casilla inicial** | **Movimientos** | **Ganador** |
| --- | --- | --- | --- |
| 1 |  |  |  |
| 2 |  |  |  |
| 3 |  |  |  |