***Conecta-K***

Israel Estévez Hatchuell

78517399E

La representación del problema será resuelta a tres niveles:

**.-Conceptual**: En la matriz de 6x7 las fichas colocadas en su interior según empiece, avance y termine la partida. Como único operador será elegir una columna **válida** donde introducir la ficha. Es necesario asegurar la aplicabilidad del operador en el estado actual para generar otro nuevo admisible.

**.- Nivel lógico:** Elegimos una estructura de datos para los estados y elegimos formato de codificación para los operadores (en este caso sólo uno, que es elegir una columna válida, no llena, en la que se pueda insertar una ficha que caerá en el sitio siguiente al ocupado o en la primera posición).

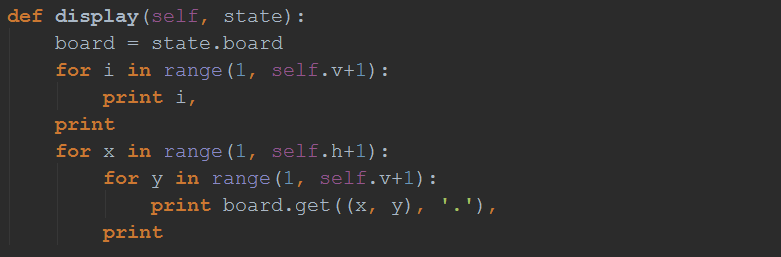
**.- Nivel físico:** implementación de estados y operadores en el lenguaje de programación elegido (Python en este caso).

***Analizando la clase Games.py***

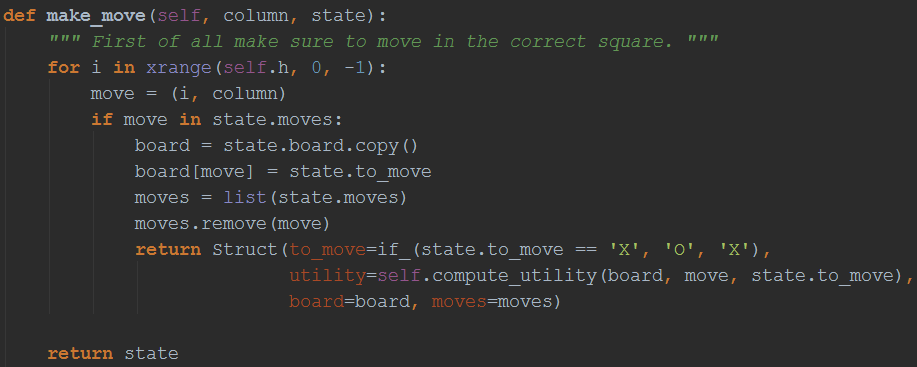
Al heredar de ella es necesario implementar las funciones: ***legal\_moves()***que devuelve una lista de los movimientos legales en ese momento,***make\_move()***devuelve el estado después de haber hecho el movimiento,***utility()***retorna el valor de este estado final al jugador *y* ***terminal\_test*()** que devuelve *verdadero* si la partida ha finalizado ya sea porque hay ganador o porque quede en tablas. Se puede sobrescribir *display* y *successors* o tratarlos como vienen por defecto. También es necesario establecer el atributo *initial* al estado inicial, en el constructor por ejemplo. Para ello heredamos de la clase TicTacToe, la cual hereda de Game y así podemos reutilizar bastante código.

***Sobrescribimos algunas funciones***

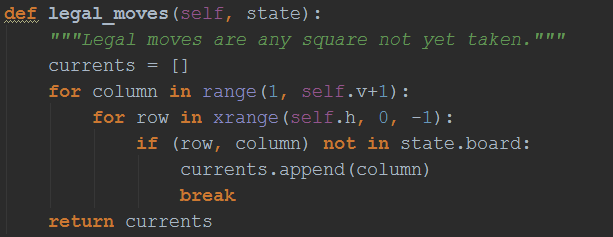
En primer lugar, la función *display* se encarga de imprimir el estado del tablero. Imprimirá una primera fila con el número de columnas y seguidamente la representación del tablero. Para ello modificamos la función *display* heredada.



A continuación nos fijamos en *make\_move,*  función que se encarga de realizar un movimiento legal sobre el tablero cambiando el estado a otro que cumpla las restricciones del juego o devolviendo el mismo estado en caso contrario. Esta función que heredamos de TicTacToe no es posible reutilizarla tal cual, ya que en ConnectFour causa efecto del factor gravedad, no nos podemos encontrar fichas en suspensión en el tablero. Como parámetros le pasamos un movimiento, en este caso una columna, y el estado en que se encuentra el juego para *mover ficha* cumpliendo las restricciones del juego.



Es el momento de prestar atención a *legal\_moves,* es la función encargada de proporcionar los sucesores a cada estado. Para ello, debemos conseguir que busque las posiciones legales del tablero dependiendo del estado actual del mismo. Le pasamos el estado actual como parámetro, hacemos un barrido del tablero y tratamos de buscar las columnas que no estén llenas.



***Heurística***

La heurística debe ser un proceso ligero, sencillo de calcular y que disponga de alguna información sobre lo cerca que se encuentra cada estado del objetivo, ventaja sobre el adversario, etc. Nos permitirá explorar los caminos más prometedores. Lo que haremos será devolver la diferencia que tiene el jugador máquina X sobre el jugador O. Para ello, calculamos para cada ficha el valor que tiene según el número de fichas consecutivas potenciales a ser 4-en-raya y lo prometedor que sea mover ficha en caso de que encuentre espacios vacíos. Para ello desarrollamos una heurística que para cada posición ocupada en el tablero examinamos su vertical, horizontal y ambas diagonales para incrementar en uno su valor por cada posición ocupada por él mismo o estando libre. **La condición de parada en el incremento, sería llegar antes de los extremos del tablero, columna 1 o 7, para así darle más importancia a la parte central, ó encontrarnos una posición ocupada por el adversario**. Pongamos un ejemplo para verlo mejor. Supongamos que el método de búsqueda del jugador máquina llega a un nodo terminal en el árbol de expansión, y se dispone a calcular su heurística teniendo el tablero la siguiente disposición de fichas:

1 2 3 4 5 6 7

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  | X |  |  |  |
|  |  | X | X |  | O | O |

1

6

La ventaja de X sobre O se calcularía de la siguiente manera: para la ficha (6,3): tendría un valor en su horizontal de **4** (dos fichas consecutivas y los espacios en blanco [6,2] y [6,5]), un valor vertical de **6** (ella misma + todos los espacios en blanco sobre ella), en la diagonal hacia el este un total de **4** (ella misma + la ficha [5,4] + 2 espacios en blanco) y en su diagonal hacia el oeste un total de **2** (ella misma + 1 espacio en blanco), todo ello hace un total de **16**. Pasamos a la ficha (6,4): valor de **4** en su horizontal (ella misma + la ficha [6,3] + los dos espacios en blanco [6,2] y [6,5]), en su vertical **6** (ella misma + la ficha [5,4] + los 4 espacios en sobre ellas), en su diagonal hacia el este **3** (ella misma + los 2 espacios en blanco [5,5] y [4,6]) y en su diagonal hacia el oeste otros **3** (ella misma + los 2 espacios en blanco [5,3] y [4,2]) haciendo un total de **16**.Por último, la ficha (5,4): en su horizontal un valor de **5** (ella misma + los dos espacios en blanco a cada lado), en su vertical **6** (ella misma + la ficha de abajo + las 4 fichas que tiene por encima de ella)**,** su diagonalhacia el este **4** (ella misma + la ficha [6,3] + los 2 espacios en blanco [4,5] y [3,6]) y por último la diagonal hacia el oeste **4** (ella misma + 2 espacios en blanco hacia el oeste [4,3] y [3,2] + el espacio en blanco hacia el este [6,5]) hace un total de **19**. Luego, para la X su valor sería de 16 + 16 + 19 = **51** **puntos**.

Calculamos la ventaja del círculo, para la ficha (6,6): en su horizontal tiene un valor un **3** (ella misma + la ficha [6, 7] + el espacio en blanco que tiene al lado [6,5]), en su vertical hace un total de **6** (ella misma + los 5 espacios en blanco situados en la columna), en la diagonal hacia el este **2** (ella misma + ficha [5, 7]) y por último en la diagonal hacia el oeste **5** (ella misma + 4 espacios en blanco que ocupan la diagonal entera), que hace un total de **16.** Para la ficha (6,7): en su horizontal **3** (ella misma + la ficha [6, 6] + el espacio en blanco [6,5]), en la vertical **6** (ella misma + 5 espacios en blanco hasta llegar a la cima de la columna), en la diagonal hacia el este **1** (ella misma) y en la diagonal hacia el oeste **6** (ella misma + 5 espacios en blanco que hacen la diagonal hasta la casilla [1,2]) haciendo un total de **16**. Luego, para el adversario O su disposición de fichas en el tablero hace un total de 16 + 16 = **32 puntos.**

La heurística será la encargada de devolver un valor heurístico de 51 – 32 = 19. **h() = 19.**

***Extensiones***

El juego dispone de las siguientes extensiones:

.- Permite jugar jugador-máquina ó jugador con otro compañero.

.- Permite diferentes niveles al jugar contra la máquina (fácil, medio y difícil).

Queda mucho por avanzar y perfeccionar en el juego. Me quedo con haberme dado cuenta que el desarrollo de la heurística, considerando que debe ser un proceso ligero y sencillo de calcular, es una delicada decisión a tomar. Básicamente, es con lo que el juego toma las decisiones más acertadas. Una buena heurística es equivalente a éxito.