

ESTRUCTURAS DE DATOS Y ALGORITMOS

En este documento describimos las estructuras de datos más importantes, junto con el funcionamiento de algunos algoritmos relevantes.

Representación del estado: Para la creación del algoritmo de la inteligencia artificial hemos utilizado la clase *Tablero*, ya que este representaba todas las cualidades que creíamos necesarias para representar el estado de una partida.

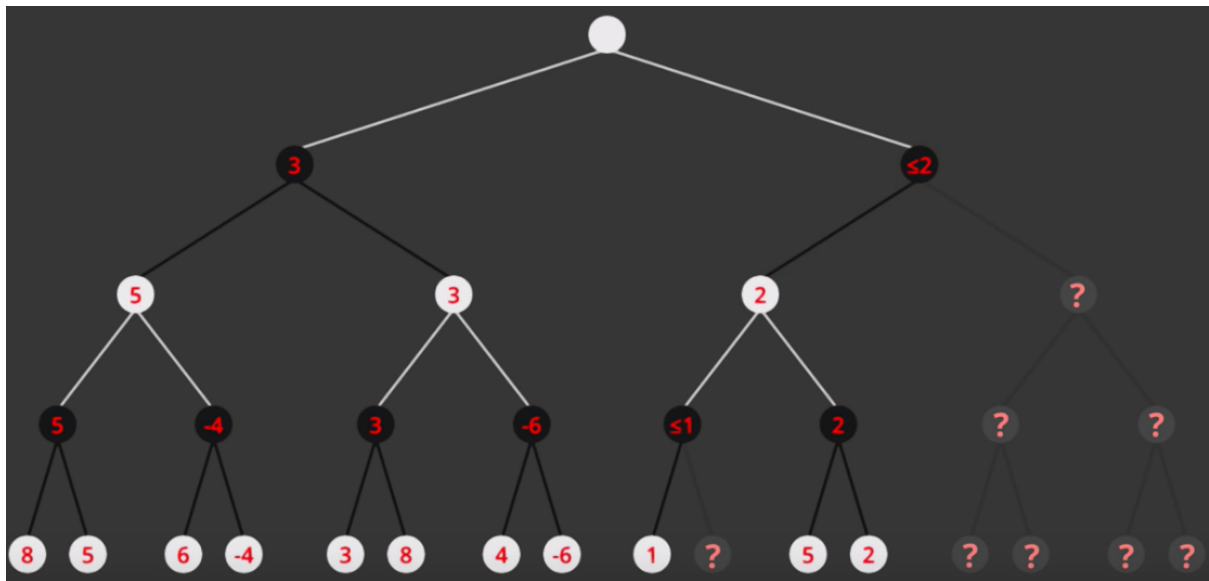
Generación de sucesores: Consecuentemente, para la generación de los estados hemos usado un Set de *Tablero*, donde cada miembro de ese Set representa un estado hijo(el *Tablero* resultante de cada uno de los posibles movimientos que tenemos).

Algoritmo de la IA: Para determinar qué movimiento debería hacer la máquina hemos utilizado el algoritmo *MiniMax*, usando la clase *Tablero* como estado tal y como hemos descrito anteriormente. Este algoritmo recibe la entrada:

```
valorMax(Tablero t, int turno, int alpha, int beta, int depth,  
int[] reglas)
```

Donde *t* representa el estado inicial, *turno* representa el turno de la partida en este estado, *alpha* y *beta* sirven para acotar el árbol de ejecución del algoritmo, *depth* representa la profundidad hasta la cual se quiere bajar y *reglas* representa el conjunto de reglas disponibles del modo de juego de partida(tirar/comer horizontalmente, verticalmente y diagonalmente).

Para una ejecución del algoritmo nos quedará un árbol de ejecución como el siguiente:



La única diferencia de nuestro árbol con el de la figura es que no necesariamente han de salir dos hijos de cada estado. Estos los generamos a partir de la lista de fichas disponibles del estado inicial, y no tienen porque ser exactamente dos posiciones.

Si deducimos que hay una rama que nunca vamos a explorar este algoritmo acotará la rama y no la vamos a leer, tal y como se muestra en la figura.

Por último, el algoritmo retornará el estado resultante de ejecutar el movimiento que *depth* pasos después nos dé la máxima puntuación posible.

El tablero:

El tablero es una estructura que nos da idea de cómo están las fichas en cada turno. Es una matriz de casillas básicamente donde cada casilla tiene propio tipo; vacío, disponible, blancas, negras.

Algoritmo Bfs:

Usamos bfs para dos cosas diferentes, una para encontrar las fichas disponibles (casillas que son válidas para colocar las fichas) y otra para modificar las fichas que se ven afectadas cuando colocamos una ficha en una casilla válida.

Como tenemos opciones para jugar el juego de forma horizontal, vertical, diagonal o mezclando una con otra aquí, dependiendo de qué conjunto de reglas cogemos hacemos los bfs horizontalmente , verticalmente , diagonalmente o mezcla de uno con otra.

Antes de empezar cada bfs de calcular Casillas disponibles inicializamos cada grafo(matriz de int) con el estado actual del tablero y a medida que se vaya ejecutando el bfs las gráficas irán cambiando lo que se verá reflejado en el tablero actual ya que es con el que trabajamos en otras clases.

Ranking: Esta clase contiene la información de cada jugador respecto a su posición dentro del ranking global del juego (representada por un Arraylist de Elementos de tipo *ElementoRanking*), además de contener también en el interior de esta clase lo necesario para todo el sistema de logros.

ElementoRanking: Contiene en su interior toda la información necesaria sobre un Jugador dentro de nuestro ranking (su identificador único “ID”, “Nickname”, además de diferentes números enteros para representar sus estadísticas respecto a las partidas jugadas por este mismo [TOTAL DE PARTIDAS JUGADAS, PARTIDAS GANADAS y PARTIDAS PERDIDAS]).

Logros: Contiene toda la información necesaria para determinar los logros obtenidos por los jugadores dentro del juego. Es una clase relativamente simple donde se almacenan números enteros y Strings (para determinar ID de los jugadores, Nicknames y algún número entero: suele indicar el máximo/mínimo de un tipo de entero en concreto [p.e.j. MÁXIMO DEL TOTAL DE PARTIDAS GANADAS POR UN MISMO JUGADOR]). Se disponen de hasta 5 logros diferentes:

- Partida más corta (respecto a Turnos jugados)
- Máximo de capturas totales dentro de una única partida
- Jugador que más ha jugado (Mayor número de partidas totales)
- Jugador que más ha ganado (Mayor número de partidas ganadas)
- Jugador que más ha perdido (Mayor número de partidas perdidas)