CONECTA 4

Práctica Inteligencia Artificial

Julián Toledano Díaz

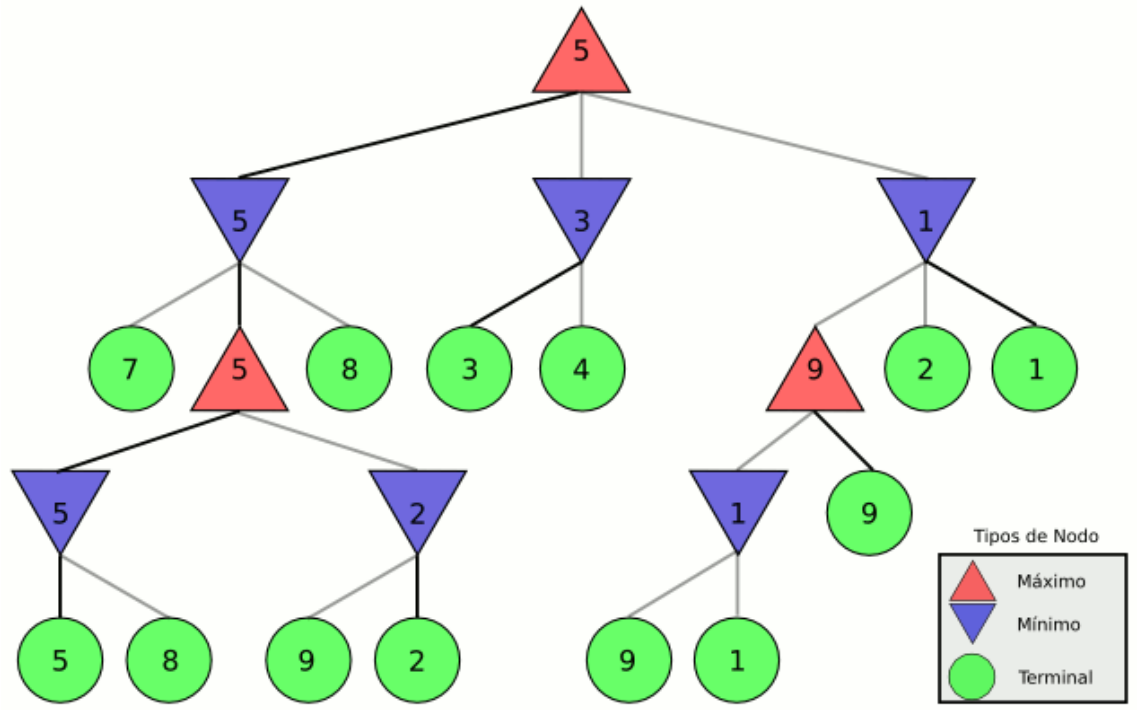
Índice

Minimax

El método utilizado para la inteligencia artificial del juego ha sido el algoritmo minimax. Aunque existen otros algoritmos más eficientes me decanté por él porque es el más sencillo de implementar.

¿Cómo funciona?

El algoritmo minimax devuelve la acción que corresponde al mejor movimiento posible, esto es, el movimiento que tiene la mejor utilidad, asumiendo que el otro jugador juega para minimizar la utilidad. Las funciones de maxValue y minValue crean el árbol de juego completo a través de sus llamadas recursivas y guardan la mayor y menor utilidad respectivamente de todas las hojas derivadas de ese estado. La utilidad se comprueba de abajo hacia arriba siendo la llegada a un estado terminal o alcanzar la profundidad máxima las condiciones de salida de las funciones.



Pseudocódigo

El siguiente pseudocódigo aparece en el libro Russell, S., (2014), *Artificial Intelligence a Modern Approach.*

```
function MINIMAX-DECISION(state) returns an action
  return arg max_{a ∈ ACTIONS(s)} MIN-VALUE(RESULT(state, a))

function MAX-VALUE(state) returns a utility value
  if TERMINAL-TEST(state) the return UTILITY(state)
  v ← −∞
  for each a in ACTIONS(state) do
      v ← MAX(v, MIN-VALUE(RESULT(s, a)))
  return v

function MIN-VALUE(state) returns a utility value
  if TERMINAL-TEST(state) the return UTILITY(state)
  v ← ∞
  for each a in ACTIONS(state) do
      v ← MIN(v, MAX-VALUE(RESULT(s, a)))
  return v
```

La notación arg max$_{a \in S}$ f(a) calcula el elemento a del grupo S que tiene el máximo valor de f(a).

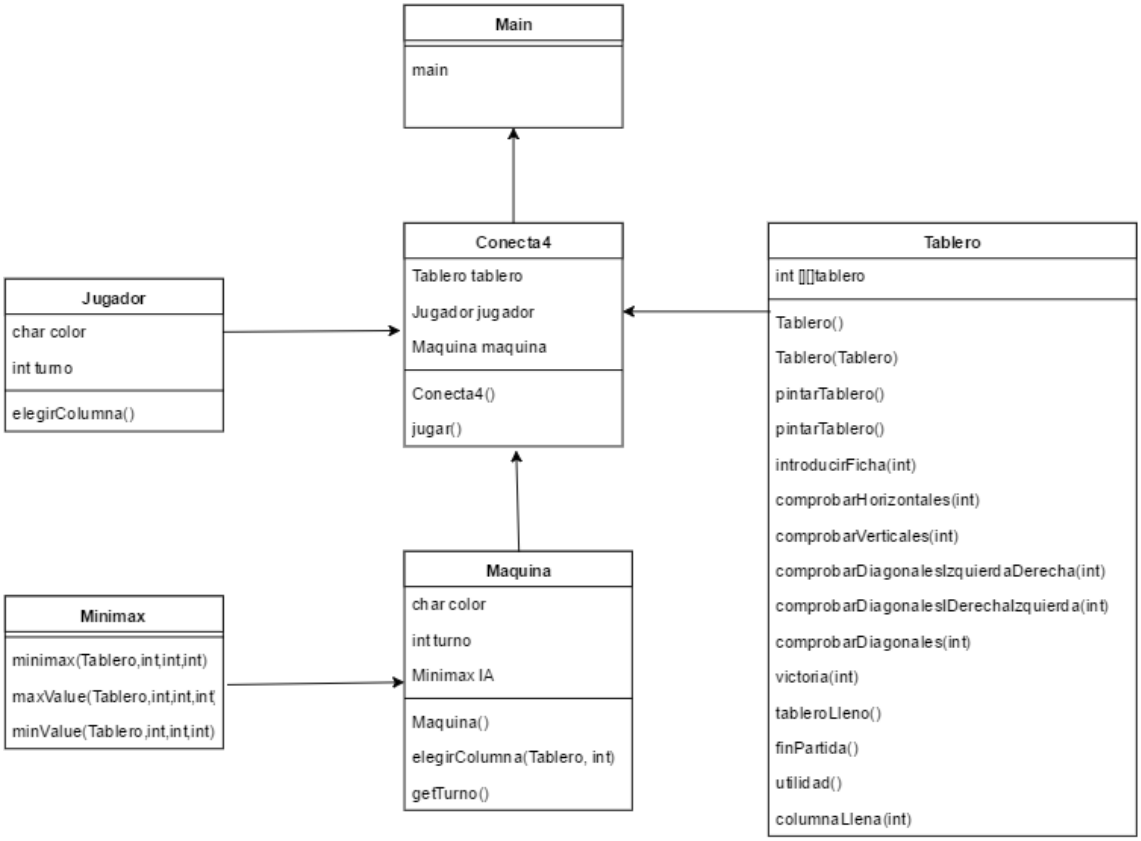
Complejidad

Si la profundidad máxima del árbol es *n* y existen *b* movimientos en cada estado, entonces la complejidad temporal del algoritmo minimax será de $O(b^n)$. La complejidad espacial será de $O(bm)$ si el algoritmo crea todo el árbol de decisiones de una vez.

Diseño e implementación

El programa consta de dos paquetes uno que contiene la representación del juego con toda su lógica: jugadores, tablero, métodos para comprobar si el juego termina, si alguien se proclama vencedor y otro paquete que contiene los algoritmos de inteligencia artificial.

A la hora de diseñarlo pensé que lo mejor sería crear una clase para modelar a cada uno de los agentes que intervendrían en el mundo real: dos jugadores y un tablero (clases Jugador, Máquina y Tablero) y utilizar estas tres en una clase que represente la partida. Esta clase es la clase Conecta4. La clase Maquina llama a la clase Minimax del paquete IA dentro de la función elegirColumna.



Debido a que las clases Jugador y Maquina son muy parecidas pensé en crear una interfaz que implementase el método elegirColumna, sin embargo, opté por no hacerlo debido a que me resulta más comprensible diferenciar a la persona física de la máquina.

A la hora de implementar el algoritmo minimax, a diferencia del pseudocódigo que utilicé como guía, decidí incluir una profundidad máxima hasta la que se puede crear el árbol de

estados. De esta manera el algoritmo es más rápido, pero menos eficaz cuanta menos profundidad pueda recorrer.

Cabe destacar que es necesario pasar una copia del tablero a la función minimax, no el tablero actual del juego.

Resultados obtenidos

Aunque por lo general los resultados son buenos existen algunos casos que hay que reportar.

1. El algoritmo puede llegar a obviar la columna 0.
2. Si la profundidad máxima es muy elevada el algoritmo responde con movimientos inesperados

Por lo demás el algoritmo se comporta de la forma esperada.