

GUIÓN DE PRÁCTICAS 7

BÚSQUEDA ENTRE ADVERSARIOS: MINIMAX

El código facilitado corresponde a una implementación (aún incompleta) de una versión del 3 en Raya, el juego TicTacToe, donde el objetivo también es conseguir 3 fichas iguales en la misma fila, columna o diagonal. Sin embargo, a diferencia del 3 en Raya, las fichas se van colocando en el tablero, pero no se desplazan dentro de él.

El juego finaliza cuando un jugador ha conseguido 3 fichas iguales en las posiciones correctas o cuando el tablero se completa sin que ningún jugador haya ganado (caso de empate).

Para completar la implementación de este juego dispones de 3 archivos con código fuente: `tictactoe.c` con la formulación del juego, `minimax` con funciones para implementar la estrategia minimax y recibir la jugada del jugador humano, y el archivo `main.c`, con el código para comenzar la ejecución.

En la formulación del problema, aunque se trate de un tablero de 3x3 celdas, se puede realizar la representación interna mediante un vector, y acceder a cada ficha con la posición que ocupa en este vector. Para ello cada posición se numera de 0 a 8 como se indica en la siguiente tabla:

Numeración de cada posición:

0	1	2
3	4	5
6	7	8

Representación Interna:

0	1	2	3	4	5	6	7	8

Ejemplo de un tablero:

X	X	
O	O	
	X	

Representación Interna:

0	1	2	3	4	5	6	7	8
1	1	0	-1	-1	0	0	1	0

X=Max (+1) O=Min (-1)

Tipos de datos para la representación del tipo Nodo:

```
typedef struct tNodo {
    int celdas[N]; // cada posición de celdas está asociada a una posición en una matriz 3x3
    int vacias;    // contador de las posiciones vacías en el tablero
} tNodo;
```

Se representaría en la implementación como:

```
tNodo juego;
juego = { 1, 1, 0, -1, -1, 0, 0, 1, 0}
```

Las jugadas realizadas por Max (X) se representan por +1, y las de Min (O) con -1. Así, por ejemplo, el acceso a la posición 0 del vector `juego.celdas[0]` daría como resultado 1, indicando que es una posición ocupada por el jugador MAX (el agente inteligente que estamos programando).

1. Observa las funciones `jugadaAdversario`, que corresponde al jugador humano, y `PSEUDOMinimax`, que correspondería al agente inteligente, y cómo se implementa el ciclo del juego completo.
2. Completa la especificación de cada función en los correspondientes ficheros `.h`
3. Ejecuta el código y familiarízate con los tipos de datos y las funciones que se utilizan.

GUIÓN DE PRÁCTICAS 7

4. Cambia los estados iniciales, y ve mostrando las posibles jugadas que podría realizar MAX junto con los valores de la función utilidad.
5. Por ahora la función PSEUDOMinimax, no realiza ningún movimiento de forma inteligente, para ello deberás implementar la estrategia minimax. A continuación se muestra el pseudocódigo de la estrategia minimax que te puede ayudar para realizar la implementación completa en C de esta estrategia:

```
tNodo: función minimax(E tNodo: nodo, E entero: jugador)
var
  entero: max,max_actual, jugada, mejorJugada
  tNodo: intento
inicio
  max ← -10000
  desde jugada ← 1 hasta N hacer
    si esValida(nodo, jugada) entonces
      intento ← aplicaJugada(nodo,jugador,jugada)
      max_actual ← valorMin(intento)
      si max_actual > max entonces
        max ← max_actual
        mejorJugada ← jugada
      fin_si
    fin_desde
  nodo=aplicaJugada(nodo,jugador,mejorJugada);
  devolver nodo
fin_función
```

GUIÓN DE PRÁCTICAS 7

```
entero: función valorMin(E tNodo: nodo)
var
  entero: valor_min, jugada, jugador=-1
inicio
  si terminal(nodo) entonces
    valor_min← utilidad(nodo)
  si_no
    valor_min ← +100000
    desde jugada ← 1 hasta N hacer
      si esValido(nodo, jugada) entonces
        valor_min ← min(valor_min, valorMax(aplicaJugada(nodo, jugador, jugada)));
      fin_si
    fin_desde
  fin_si
  devuelve valor_min
fin_función
```

```
entero: función valorMax(E tNodo: nodo)
var
  entero: valor_max, jugada, jugador=1
inicio
  si terminal(nodo) entonces
    valor_max← utilidad(nodo)
  si_no
    valor_max ← -100000
    desde jugada ← 1 hasta N hacer
      si esValida(nodo, jugada) entonces
        valor_max ← max(valor_max, valorMin(aplicaJugada(nodo, jugador, jugada)));
      fin_si
    fin_desde
  fin_si
  devuelve valor_max
fin_función
```

Resto de funciones:

- **terminal**(nodo): función que comprueba si es un nodo terminal
- **esValida**(nodo,jugada): comprueba si es válida aplicar la jugada al nodo actual
- **aplicaJugada**(nodo,jugador,jugada): aplica una jugada al nodo actual
- **min**(valor1,valor2): devuelve el mínimo de 2 valores
- **max**(valor1,valor2): devuelve el máximo de 2 valores