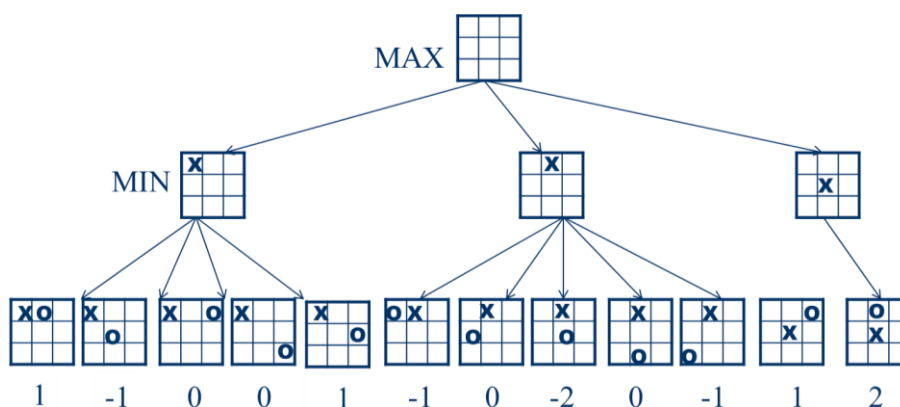


GUION DE PRÁCTICAS

BÚSQUEDA ENTRE ADVERSARIOS: FUNCIONES HEURÍSTICAS

Continuando con el juego TicTacToe:

1. Implementa una función heurística adecuada para este problema.
2. Las funciones heurísticas constituyen la alternativa a la generación del árbol completo Minimax. Crea nuevas funciones o **modifica las existentes** para que el código:
 - a. Incluya un límite de profundidad dado por la constante LIMITE
 - b. Se actualice la profundidad del árbol en cada llamada recursiva.
 - c. Si se llega a este límite y no es un estado terminal, entonces que aplique la función heurística al nodo y devuelva este valor como valor minimax del nodo al límite de profundidad
 - d. Si en cualquier momento antes de llegar al límite de profundidad, se llega a un nodo terminal, que devuelva el valor de la función de utilidad (Valor alto, por ejemplo 100 si es MAX quien gana la partida, valor bajo, por ejemplo -100 si es MIN quien gana la partida)



Ejemplo cuando la profundidad LÍMITE ha sido establecida en 2 (más el nodo inicial). En este caso no habrá ningún nodo terminal, en todos ellos se deberá aplicar la función heurística a los nodos del último nivel y mediante minimax estos valores serán propagados hacia los nodos superiores.

PODA ALFA-BETA

3. Implementa la poda alfa-beta para que funcione tanto si se desarrolla el árbol completo de búsqueda como si se establece un límite de profundidad y se aplican funciones heurísticas en los nodos finales.

```
Nodo: función poda_ab(E/S Nodo: nodo)
var
    entero: jugador, prof, v
    Jugada: jugada, mejorJugada
    Nodo: intento
inicio
    jugador ← 1  alfa ←  $-\infty$   beta ←  $+\infty$   prof ← 0
    mejorJugada ← jugadas[0]
    para cada jugada en jugadas hacer
        si esValida(nodo, jugada) entonces
            intento ← aplicaJugada(nodo, jugada, jugador)
            v ← valorMin_ab(intento, prof+1, alfa, beta)
            si v > alfa entonces
                alfa ← v
                mejorJugada ← jugada
            fin_si
        fin_si
    fin_para
    nodo=aplicaJugada(nodo, mejorJugada, jugador)
    devuelve nodo
fin_función
```

```
entero: función valorMin_ab(E Nodo: nodo, E entero: prof,
                           E entero: alfa, E entero: beta)
var
    entero: vmin, i
    Jugada: jugada
    Nodo: intento
inicio
    si terminal(nodo) entonces
        vmin ← utilidad(nodo)
    si_no
        si prof=LIMITE entonces
            vmin ← heuristica(nodo)
        si_no
            i ← 0
            mientras i<len(jugadas) Y alfa<beta hacer
                jugada ← jugadas[i]
                si esValida(nodo, jugada) entonces
                    intento ← aplicaJugada(nodo, jugada, -1)
                    beta ← minimo(beta, valorMax_ab(intento, prof+1,
                                                    alfa, beta))
                fin_si
                i ← i+1
            fin_mientras
            vmin ← beta
        fin_si
    fin_si
    devuelve vmin
fin_función
```

```
entero: función valorMax_ab(E Nodo: nodo, E entero: prof,  
                           E entero: alfa, E entero: beta)  
  
var  
    entero: vmax, i  
    Jugada: jugada  
    Nodo: intento  
inicio  
    si terminal(nodo) entonces  
        vmax ← utilidad(nodo)  
    si_no  
        si prof=LIMITE entonces  
            vmax ← heuristica(nodo)  
        si_no  
            i ← 0  
            mientras i<len(jugadas) Y alfa<beta hacer  
                jugada ← jugadas[i]  
                si esValida(nodo, jugada) entonces  
                    intento ← aplicaJugada(nodo, jugada, 1)  
                    alfa ← maximo(alfa, valorMin_ab(intento, prof+1,  
                                                    alfa, beta)  
                fin_si  
                i ← i+1  
            fin_mientras  
            vmax ← alfa  
        fin_si  
    fin_si  
    devuelve vmax  
fin_función
```