Universidad del País Vasco

Facultad de Ciencia y Tecnología

Diseño de Algoritmos
Práctica III

Grupo 11

Ander Cano

Mikel Lamela

Aitor Larrinoa

Abril 2021

Índice

1. Introducción

En el presente documento se mostrará el diseño de un algoritmo para simular un juego entre la computadora y un usuario. Para ello, hemos hecho uso del algoritmo minimax visto en clase, con su correspondiente poda alfa beta. Este consiste en la búsqueda del mejor movimiento para uno mismo, suponiendo que el rival escogerá el peor para ti (mejor para él).

Para desarrollar esta idea, hemos decidido implementar el llamado *Juego del Oso*. Este es un juego parecido al tres en raya, pero con unas variantes interesantes que creemos que lo hace más atractivo. Más adelante se expondrá el funcionamiento del juego y sus reglas.

Empezaremos nuestro documento definiendo el tablero de juego y reglamento, seguido de una descripción de diferentes estrategias posibles a llevar a cabo, que haremos que la CPU utilice mediante diferentes funciones de evaluación. Con esto conseguiremos que nuestro juego conste de tres niveles de dificultad: fácil, medio y difícil. Más adelante, mostraremos el algoritmo minimax utilizado, el cual es parecido al visto en clase pero con varios matices diferentes necesarios a la hora de desarrollar nuestro ejemplo. Seguiremos con un estudio de las complejidades temporales de nuestros algoritmos, para después finalizar viendo cómo es nuestro algoritmo principal y unas cuantas conclusiones de las diferentes estrategias utilizadas y del trabajo en general.

Para ver el código Python correspondiente a cada algoritmo ver 8.

2. Modo de juego y reglamento

La dinámica del juego del oso es muy parecida al del conocido tres en raya. El objetivo de este consiste en conseguir escribir la palabra "OSO" más veces que el adversario. Esta puede formarse en vertical, horizontal o diagonal. En cada turno, sólo se podrá escribir una letra, es decir, en cada turno podremos colocar una "O" o una "S" en una única casilla vacía. Si alguien forma la palabra "OSO", se anotará un punto en el marcador y repetirá turno. Mostremos una imagen que representa el juego de manera clara y sencilla.

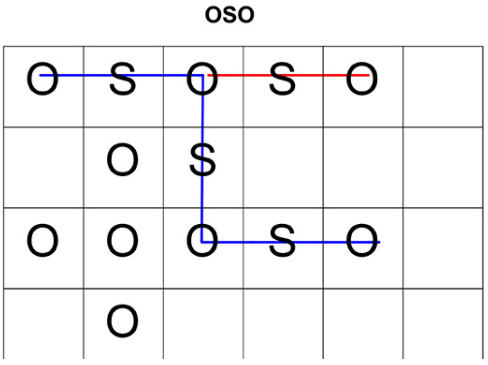

Figura 1: Imagen Oso

El juego finalizará cuando todas las casillas estén llenas. En ese instante se comprobará el resultado del marcador y se decidirá el ganador del juego. El marcador está compuesto por dos cifras, una de ellas indicará el número de "OSO" que ha conseguido hacer el jugador 1, y la otra indicará el número de "OSO" que ha conseguido hacer el jugador 2. A continuación mostramos las reglas principales del juego.

En el juego del oso hay tres reglas fundamentales y son las siguientes:

- En cada turno solamente se puede escribir una letra.
- Si se completa la palabra "OSO", se repite turno.
- No se puede cambiar o sobreescribir letras del tablero.

En conclusión, es un juego sencillo y divertido para jugar con amigos.

2.1. Tablero

Normalmente, se juega en tableros de tamaño 6x6 u 8x8, aunque nosotros implementaremos el código para un tablero 6x6. De esta manera, el juego se hace más ameno ya que si consideramos un tablero de tamaño 8x8, la partida se hace demasiada larga.

3. Estrategias - Funciones de evaluación

En esta sección abordaremos las diferentes estrategias que puede desarrollar la CPU. En este caso se han propuesto tres estrategias bastante diferentes, podríamos decir que representan a un jugador ofensivo (que desea hacer gran cantidad de OSO, arriesgando en cada jugada), un jugador que tiene ciertos concocimientos previos del juego y un jugador experto que aboga por la prudencia esperando el fallo del rival.

Las estrategias vendrán representadas por el nivel de dificultad que se escoja, "Fácil", "Medio" o "Difícil", es obvio qué tipo de jugador representa cada una. Pero, ¿qué quiere decir que una estrategia represente un nivel de dificultad? Las estrategias, realmente, son funciones de evaluación, de las cuales el algoritmo minimax hará uso para dar un valor a cada una de las casillas. Luego, haremos uso de ese valor, para saber dónde y qué escribir en el tablero. Cuanto mejor sea la función de evaluación, la decisión de qué y dónde escribir será mas influyente en el marcador final en favor del ordenador. Con esto, venimos a decir, que cuanto más precisa sea la función de evaluación, más difícil será ganarle a la máquina.

Una vez se tiene la idea de qué representan las estrategias, a continuación vamos a describir cada una de ellas, añadiendo el algoritmo de la función de evaluación correspondiente. No obstante, cabe destacar que siempre que sea posible la maquina hará la palabra OSO, en este aspecto consideramos un nivel de error bajo ya que al no haber límite de tiempo, cada jugador puede estudiar concienzudamente el tablero previo a indicar su jugada.

3.1. Estrategia 1 - *Nivel Fácil*

La estrategia 2 es la que hemos denominado *nivel fácil*. Suponiendo que no existe la posibilidad de hacer OSO, la idea de esta segunda función de evaluación es escribir únicamente letras "O" en una casilla colocada en L (Salto del Caballo en Ajedrez) a una casilla en la que tengamos escrita una "O". Esto generará confusión en el usuario y provocará que se equivoque con mayor facilidad.

El algoritmo correspondiente a la función de evaluación se puede ver en Algoritmo (1)

La idea es simple, otorgar un valor alto a la jugada en la que se forma OSO con en casos anteriores, y sino estudiar si en algún salto del caballo hay una "O", en ese caso se asigna un valor positivo o negativo dependiendo del usuario, y como antes 0 en cualquier otro caso.

3.2. Estrategia 2 - *Nivel Medio*

La estrategia 1 es la que hemos denominado *nivel medio*. Suponiendo que no se puede escribir OSO, entonces entra en juego la función de evaluación. La idea de esta función de evaluación es comprobar el número de "O" y el número de "S" que rodean a cada casilla. La resta de ambas cifras determinará lo que vayamos a escribir en dicha casilla. Supongamos lo siguiente:

c = "Número de "O" que rodean a una casilla (a,b) del tablero"
d = "Número de "S" que rodean a una casilla (a,b) del tablero"

Si realizamos la resta de ambas cifras, tenemos tres posibilidades:

si c − d > 0 entonces escribimos una "O" en la casilla (a,b)
si c − d < 0 entonces escribimos una "S" en la casilla (a,b)
si c − d = 0 entonces escribimos una "O" o una "S" indistintamente en la casilla (a,b)

La idea de las decisiones que acabamos de señalar es la siguiente: si una casilla está rodeada de muchas "O" y pocas "S" entonces estamos en el primero de los casos, la resta es positiva, y eso quiere decir que si escribimos una "S", habría muchas opciones de hacer OSO por parte del oponente. Por ello decidimos escribir una "O" en dicha casilla. Lo mismo ocurre con el segundo caso, pero cambiando las "O" por "S", la idea es la misma. En el caso de que la resta de ambos sea 0, escribimos una letra indistintamente.

Hemos denominado a esta función de evaluación el nivel fácil porque en el caso de que la resta sea 0, la letra a escribir es aleatoria, y puede que el usuario pueda escribir "OSO" en la siguiente jugada ya que dejamos la elección de la letra a escribir en manos del azar. Además, jugando adecuadamente es bastante sencillo realizar la palabra OSO.

El algoritmo de la función de evaluación se puede ver en Algoritmo (2)

Como se ha comentado, primeramente se estudia cuantos OSO se pueden hacer con la jugada actual, que se hace mediante el Algoritmo (3), y se le otorga un valor de c*(100 + el número de OSO formados), donde c = 1 si es turno del ordenador y c = -1 en caso de turno del usuario.

A continuación, se mira los casos descritos anteriormente otorgando el valor c*(10 + $|c-d|$), y en caso de empate un valor aleatorio entre 1 y 5. Al resto de jugadas no deseadas se les otroga un 0.

3.3. Estrategia 3 - *Nivel Difícil*

La estrategia 3 corresponde al *nivel difícil*. Esta estrategia es muy sencilla como función de evaluación pero es muy precisa a la hora de determinar qué escribir y dónde hacerlo. A continuación definimos la función de evaluación:

$$\text{feval}_3 = \text{Puntuación CPU - Puntuación Usuario}$$

A medida que aumentamos el nivel de exploración se hace cada vez más difícil ganarle a esta función de evaluación, ya que estimará el hecho de que el usuario haga puntos tomando el valor minímo del valor de feval_3, y cuando el turno sea de la máquina, cogerá el máximo del valor de feval_3. Esto nos permite estimar la jugada del usuario de una manera muy acertada, anticipando movimientos y dejando pocas opciones de victoria.Sin embargo, un jugador experto tendría la posibilidad del empate asegurada si toma una estrategia defensiva o conservadora.

Nótese que en este caso no es necesario estudiar si se forma OSO, ya que para decidir el turno en la llamada recursiva se realiza dicho estudio que actualiza el marcador, luego simplemente con eso ya es conocido si formamos OSO en algun nivel, dado que el turno se mantiene.

4. Algoritmo Minimax

Presentamos ahora el algoritmo Minimax utilizado en el juego, las modificaciones realizadas sobre el algoritmo principal han sido realizadas a conveniencia del juego. La principal novedad que presenta el algortimo es que en cada casilla es posible escribir dos letras, lo cual induce a dificultades. Para tener ordenados las casillas y libres, y recorrer las posibilidades, hemos optado por doblar las casillas en la lista de casillas vacias. Consideraremos que en las casillas en posición par, indexando desde el 0, se escribirá una "S", mientras que en las impares escribiremos una "O".

Variables

- Tablero: Tablero 6x6 que representa la situación actual del juego.
- V: Conjunto de casillas libres dentro de Tablero.
- Casilla: Casilla actual analizada de Tablero.
- n: Nivel de exploración.
- turno: Variable que representa el turno. Si turno = True juega la máquina, si turno = False juega el usuario.
- f: Función de evaluación.
- alpha: Cota inferior inicializada a -1000.
- beta: Cota superior inicializada a 1000.

El algoritmo minimax se puede observar en Algoritmo (4).

La función auxiliar "Escribir" unicamente escribe una "S" o una "O" en la casilla de parámetro, mirando si el índice pasado es par o impar. La funcion auxiliar"OSO", determina si se ha hecho OSO actualizando tanto el marcador como el turno del juego. De esta forma, se respeta la norma de en caso de formar OSO mantener el turno. El algoritmo es sencillo ya que es mirar las casillas algo trivial, por lo que no se expone el algoritmo.

5. Complejidades temporales y espaciales

La complejidad temporal tiene una dependencia tanto del tamaño del tablero, como del nivel de dificultad al que se este jugando, como de los niveles de exploración escogidos así como de la destreza del usuario.

Primero comentar que las funciones de evaluación, tienen un coste temporal constante ya que realizan un número finito no muy grande de comparaciones, o simplemente una resta. Por otro lado, las funciones auxiliares son de coste temporal constante también por la misma razon.

Nivel fácil: tiempo medio de jugada: 9.989738464355469e-05 segundos.

Nivel medio: tiempo medio de jugada: 0.0003135831732498972 segundos.

Nivel difícil: tiempo medio de jugada: 1.6943857272466023 segundos.

Teniendo en cuenta todo esto, sean "m" niveles de exploración y un tablero nxn, entonces el algoritmo tienen un peor caso de t(n) $\in \boldsymbol{O}(m * (n^2))$, en el que la exploración es máxima suponiendo que el rival rellena el menor número de casillas. Aun así, es difícil de estimar ya que influye demasiado la jugada del Usuario, ya que puede porvocar una concatenación de OSO's que produzca jugadas rápidas, lo cúal produce un tiempo de ejecución mínimo.

En el mejor caso es un caso hipotético en el que la máquina cometa errores claros, lo cual porvocaría repetidos turnos del usuario complentando gran parte del tablero, reduciendo la zona de exploración.

La complejidad espacial en este caso es fija, dado que el almacenamiento se conoce una vez determinado el tamaño del tablero n x n. Una estimación prescisa sería S(n) $\in \boldsymbol{O}(3n^2)$, ya que es el coste de almacenar el tablero y la lista de casillas vacias, donde las casillas están dobladas.

6. Algoritmo Principal

El algoritmo principal de juego es bastante simple consta de un ciclo principal que representa el inicio del programa general en el que se puede realizar la consulta del reglamento y la configuración. Una vez realizado esto, se realiza un sorteo para decidir quien empieza escribiendo, y comienza el juego, es decir un nuevo ciclo, dependiendo de la variable turno se pide la jugada al usuario o se hace uso de la función Minimax para determinar la jugada del ordenador. Así hasta completar el tablero, y determinar el ganador. Una vez terminada la partida se puede jugar la revancha eligiendo el nivel de dificultad deseado.

El algoritmo principal se puede ver en (5) de la sección de Algoritmos.

7. Conclusiones

En conclusión, hemos comprobado ideas que podíamos pensar antes de implementar el juego, como por ejemplo cuánto más compleja sea la táctica más tiempo tardará el ordenador en pensar, o que hay una influencia clara en la estrategia seguida por el usuario. Además, hemos mostrado como el algoritmo Minimax con poda alpha-beta es extremadamente eficiente en este tipo de juegos 1 contra 1 donde no influye el azar.

8. Código Python

Las siguientes funciones escritas en código Python son las correspondientes a la estructura gráfica del juego:

```
1
2  ''' ESTRUCTURA GR FICA DEL JUEGO '''
3  import random
4
5  def Reglas():
6      print('\n Para leer las instrucciones Pulse CUALQUIER letra y ENTER. ...
           Para jugar pulse solamente ENTER.')
7      aa = input()
8      if aa != '':
9          fp = open('reglas_oso.txt')
10         print(fp.read())
11         fp.close()
12         print('  Jugamos ? Click ENTER ')
13         bb = input()
14         while bb != '':
15             print('  Jugamos ? Click ENTER ')
16             bb = input()
17
18 def dibujarTablero(tablero):
19
20     print('\n')
21     print('\t\t '  + '1' + '   ' + '2' + '   ' + '3' + '   ' + '4' + '   ' + ...
           '5' + '   ' + '6'+'\n')
22     print('\t\t   |   |   |   |   |   ')
23     print('\t     '  + '1' + '   '  + tablero[0][0] + ' | ' + tablero[0][1] ...
           + ' | ' + tablero[0][2] + ' | ' + tablero[0][3] + ' | ' + ...
           tablero[0][4] + ' | ' + tablero[0][5])
24     print('\t\t   |   |   |   |   |   ')
25     print('\t\t-----------------------')
26     print('\t\t   |   |   |   |   |   ')
27     print('\t     '  + '2' + '   '  + tablero[1][0] + ' | ' + tablero[1][1] ...
           + ' | ' + tablero[1][2] + ' | ' + tablero[1][3] + ' | ' + ...
           tablero[1][4] + ' | ' + tablero[1][5])
28     print('\t\t   |   |   |   |   |   ')
29     print('\t\t-----------------------')
30     print('\t\t   |   |   |   |   |   ')
31     print('\t     '  + '3' + '   '  + tablero[2][0] + ' | ' + tablero[2][1] ...
           + ' | ' + tablero[2][2] + ' | ' + tablero[2][3] + ' | ' + ...
           tablero[2][4] + ' | ' + tablero[2][5])
32     print('\t\t   |   |   |   |   |   ')
33     print('\t\t-----------------------')
34     print('\t\t   |   |   |   |   |   ')
35     print('\t     '  + '4' + '   '  + tablero[3][0] + ' | ' + tablero[3][1] ...
           + ' | ' + tablero[3][2] + ' | ' + tablero[3][3] + ' | ' + ...
           tablero[3][4] + ' | ' + tablero[3][5])
36     print('\t\t   |   |   |   |   |   ')
37     print('\t\t-----------------------')
38     print('\t\t   |   |   |   |   |   ')
39     print('\t     '  + '5' + '   '  + tablero[4][0] + ' | ' + tablero[4][1] ...
           + ' | ' + tablero[4][2] + ' | ' + tablero[4][3] + ' | ' + ...
           tablero[4][4] + ' | ' + tablero[4][5])
40     print('\t\t   |   |   |   |   |   ')
41     print('\t\t-----------------------')
42     print('\t\t   |   |   |   |   |   ')
```

```
43      print('\t      '  + '6' + '    '  + tablero[5][0] + ' | ' + tablero[5][1] ...
            + ' | ' + tablero[5][2] + ' | ' + tablero[5][3] + ' | ' + ...
            tablero[5][4] + ' | ' + tablero[5][5])
44      print('\t\t    |   |   |   |   |   ')
45      print('\n')
46      print('CPU: '+ str(tablero[-1][0]))
47      print('USUARIO: '+ str(tablero[-1][1]))
48
49
50  def Nivel():
51      print('\n Elegir nivel de dificultad: \n')
52      print('    1 - F cil')
53      print('    2 - Medio')
54      print('    3 - Dificil')
55      while True:
56          f = input('\n Introducir  N   del nivel de dificultad: ')
57          if f in ['1','2','3']:
58              f = int(f)
59              break
60
61      if f == 3:
62          n = int(input('\n    Introducir  N   de niveles de exploraci n: '))
63      else:
64          n = 1
65
66      return f,n
67
68
69  def JugadaUsuario(tablero,V):
70      pos = (' ',' ')
71
72      while True:
73          print('\n   Cul   es la posici n en la que quiere escribir? fila ...
                columna')
74          strpos = input()
75          if strpos == 'EXIT':
76              return 0
77          entrada = strpos.split()
78          if len(entrada) == 2:
79              if entrada[0] in ['1','2','3','4','5','6'] and entrada[1] in ...
                    ['1','2','3','4','5','6']:
80                  pos = (int(entrada[0])-1,int(entrada[1])-1)
81                  if pos  in V:
82                      break
83                  else:
84                      print( '\n    AVISO ! Casilla ocupada' )
85              else:
86                  print( '\n    AVISO ! Coordenadas fuera del tablero' )
87          else:
88              print( '\n    AVISO ! Formato erroneo' )
89
90      letra = ''
91      while True:
92          print('\n   Qu   letra desea escribir? S | O')
93          letra = input().upper()
94          if letra == 'S' or letra == 'O':
95              break
96          else:
97               print( '\n    AVISO ! Escriba S,s,O,o' )
98
99      return [letra,pos]
```

```
100
101 def Escribir(tablero, V, Mov):
102     V.remove(Mov[1])
103     V.remove(Mov[1])
104     (a,b) = Mov[1]
105     tablero[a][b] = Mov[0]
106
107 def Final(Tablero):
108     if Tablero[-1][0] < Tablero[-1][1]:
109         print(' \t        VICTORIA !!!')
110     elif Tablero[-1][0] > Tablero[-1][1]:
111         print(' \t DERROTA...')
112     else:
113         print(' Empate')
114
115 def revancha():
116     Decision = ''
117     while Decision not in ['SI','NO']:
118         print('  Desea  volver a jugar (SI o NO)?')
119         Decision = input()
120     return Decision == 'SI'
121
122
123 def QuienVaPrimero():
124     return random.choice([True,False])
```

Lo siguiente corresponde a las diferentes funciones de evaluación y al algoritmo minimax descritos anteriormente.

```
1
2 import random
3
4 ''' FUNCIONES AUXILIARES '''
5 def haceroso(casilla,Tablero):
6     (a,b)=casilla
7     n = len(Tablero[0])
8     Num_osos = 0
9     if Tablero[a][b] == "O":
10         X = [-2,-1,-2,-1,-2,-1,0,0,0,0,2,1,2,1,2,1]
11         Y = [2,1,0,0,-2,-1,2,1,-2,-1,2,1,0,0,-2,-1]
12         for i in range(0,15,2):
13             if 0≤ a + X[i] < n and 0≤ a + X[i+1] < n and 0≤ b + Y[i] < n and ...
                   0≤ b + Y[i+1] and (Tablero[a + X[i]][b + Y[i]],Tablero[a + ...
                   X[i+1]][ b + Y[i+1]]) == ('O','S'):
14                 Num_osos += 1
15     else:
16         X = [-1,1,0,0,-1,1,-1,1]
17         Y = [0,0,-1,1,-1,1,1,-1]
18         for i in range(0,7,2):
19             if 0≤ a + X[i] < n and 0≤ a + X[i+1] < n and 0≤ b + Y[i] < n and ...
                   0≤ b + Y[i+1] < n and (Tablero[a + X[i]][b + Y[i]],Tablero[a ...
                   + X[i+1]][ b + Y[i+1]]) == ('O','O'):
20                 Num_osos += 1
21     return Num_osos
22
23 def OSO_O(casilla,Tablero,turno):
24     X = [-2,-1,-2,-1,-2,-1,0,0,0,0,2,1,2,1,2,1]
25     Y = [2,1,0,0,-2,-1,2,1,-2,-1,2,1,0,0,-2,-1]
26     (a,b) = casilla
```

```
27      n = len(Tablero[0])
28      Num_osos = 0
29      T = not turno
30      for i in range(0,15,2):
31          if 0≤ a + X[i] < n and 0≤ a + X[i+1] < n and 0≤ b + Y[i] < n and 0≤ ...
                b + Y[i+1] and (Tablero[a + X[i]][b + Y[i]],Tablero[a + X[i+1]][ ...
                b + Y[i+1]]) == ('O','S'):
32              Num_osos += 1
33
34      if Num_osos > 0:
35          T = not T
36
37      if turno:
38          Tablero[-1] = (Tablero[-1][0] + Num_osos, Tablero[-1][1])
39      else:
40          Tablero[-1] = (Tablero[-1][0], Tablero[-1][1] + Num_osos)
41      return T
42
43  def OSO_S(casilla,Tablero,turno):
44      (a,b) = casilla
45      X = [-1,1,0,0,-1,1,-1,1]
46      Y = [0,0,-1,1,-1,1,1,-1]
47      n = len(Tablero[0])
48      Num_osos = 0
49      T = not turno
50      for i in range(0,7,2):
51          if 0≤ a + X[i] < n and 0≤ a + X[i+1] < n and 0≤ b + Y[i] < n and 0≤ ...
                b + Y[i+1] < n and (Tablero[a + X[i]][b + Y[i]],Tablero[a + ...
                X[i+1]][ b + Y[i+1]]) == ('O','O'):
52              Num_osos += 1
53      if Num_osos > 0:
54          T = not T
55      if turno:
56          Tablero[-1] = (Tablero[-1][0] + Num_osos, Tablero[-1][1])
57      else:
58          Tablero[-1] = (Tablero[-1][0], Tablero[-1][1] + Num_osos)
59      return T
60
61  ''' FUNCIONES DE EVALUACI N '''
62  def eval1(casilla,Tablero,V,turno):
63      return Tablero[-1][0] - Tablero[-1][1]
64
65  def eval2(casilla,Tablero,V,turno):
66      if not turno:
67          c=1
68      else:
69          c=-1
70      (a,b)=casilla
71
72      Num_osos = haceroso(casilla,Tablero)
73      if haceroso(casilla,Tablero) > 0:
74          return (-c)*(100 + Num_osos)
75      o=0
76      s=0
77      X=[-1,-1,-1,0,0,1,1,1]
78      Y=[1,0,-1,1,-1,1,0,-1]
79      n = len(Tablero[0])
80      for i in range(8):
81          if 0≤ a + X[i] < n and 0≤ b + Y[i] < n:
82              if Tablero[a + X[i]][b + Y[i]] == 'O':
83                  o+=1
```

```
84              elif Tablero[a + X[i]][b + Y[i]] == 'S':
85                  s+=1
86      if Tablero[a][b] == 'S' and o-s < 0:
87          return c*(10 + abs(o-s))
88      elif Tablero[a][b] == 'O' and o-s > 0:
89          return c*(10 + abs(o-s))
90      elif o-s == 0:
91          return random.randint(1,5)
92      else:
93          return 0
94
95  def eval3(casilla,Tablero,V,turno):
96      if not turno:
97          c=1
98      else:
99          c=-1
100     (a,b)=casilla
101     X=[2,2,-1,-1,1,1,-2,-2]
102     Y=[-1,1,-2,2,-2,2,-1,1]
103     Num_osos = haceroso(casilla,Tablero)
104     if haceroso(casilla,Tablero) > 0:
105         return (-c)*(10 + Num_osos)
106     n = len(Tablero[0])
107     for i in range(8):
108         if 0≤ a + X[i] < n and 0≤ b + Y[i] < n and Tablero[a + X[i]][b + ...
                Y[i]] == 'O':
109             return 5*c
110     return 0
111
112
113
114 ''' ALGORITMO MINIMAX '''
115 def minimax(casilla,tablero,V,turno,n,f,alpha=-1000,beta=1000):
116     letra = ''
117     pos = ''
118     if n==0 or not V:
119         return (f(casilla,tablero,V,turno),[letra,pos])
120     elif turno:
121         for i in range(len(V)):
122             (a,b) = V[i]
123             marcador = tablero[-1]
124             if i%2 == 0:
125                 tablero[a][b] = 'O'
126                 T = OSO_O(V[i],tablero,turno)
127                 VV = V[:i] + V[i+2:]
128             else:
129                 tablero[a][b] = 'S'
130                 T = OSO_S(V[i],tablero,turno)
131                 VV = V[:i-1] + V[i+1:]
132             Eval=minimax(V[i],tablero,VV,T,n-1,f,alpha,beta)[0]
133             if alpha < Eval:
134                 alpha = Eval
135                 pos = V[i]
136                 letra = tablero[a][b]
137             tablero[a][b] = ' '
138             tablero[-1] = marcador
139             if beta≤alpha:
140                 break
141         return (alpha,[letra,pos])
142     else:
143         for i in range(len(V)):
```

```
144             (a,b) = V[i]
145             marcador = tablero[-1]
146             if i%2 == 0:
147                 tablero[a][b] = 'S'
148                 T = OSO_O(V[i],tablero,turno)
149                 VV = V[:i] + V[i+2:]
150             else:
151                 tablero[a][b] = 'O'
152                 T = OSO_S(V[i],tablero,turno)
153                 VV = V[:i-1] + V[i+1:]
154             Eval=minimax(V[i],tablero,VV,T,n-1,f,alpha,beta)[0]
155             if beta > Eval:
156                 beta = Eval
157                 pos = V[i]
158                 letra = tablero[a][b]
159             tablero[a][b] = ' '
160             tablero[-1]  = marcador
161             if beta≤alpha:
162                 break
163         return (beta,[letra,pos])
```

Y por último, el código que ejecutaremos a la hora de jugar al juego:

```
1
2
3  from Estructura_grafica import *
4  from OSO_Minimax import *
5  import random
6  import time
7
8  print( '- JUEGO DEL OSO - DISEÑO DE ALGORITMOS 2021 - GRUPO 11  - \n')
9  print( ' Activar: Bloq Mayus \n')
10 print( ' Para detener el juego escribir: EXIT \n')
11
12 # Reglamento
13 Reglas()
14
15
16 while True:
17
18     # Nivel de dificultad
19     f,n = Nivel()
20     Nombres_Funciones = [eval3,eval2,eval1]
21     f_eval = Nombres_Funciones[f-1]
22
23     # Creamos el tablero vacio
24     tablero = [[' 'for i in range(6)]for j in range(6)]
25     tablero.append((0,0))
26     dibujarTablero(tablero)
27
28     tiempos = list()
29     tiemp_med = 0
30     num = 0
31
32     # Casillas libres minimax
33     V = []
34     for i in range(6):
35         for j in range(6):
36             V.append((i,j))
```

```
37              V.append((i,j))
38
39      # Turno Inicial
40      if QuienVaPrimero():
41          turno = True
42          print("\n Ups! Comienza jugando la CPU\n")
43      else:
44          turno = False
45          print("\n Enhorabuena!! Te toca empezar!\n")
46
47      seEstajugando = True
48
49      while seEstajugando:
50
51          if not turno: # Turno jugador
52
53              Mov = JugadaUsuario(tablero,V)
54
55              if Mov == 0: # Detener el juego
56                  break
57
58              Escribir(tablero,V,Mov)
59
60
61              # An lisis del turno tras jugada
62              if Mov[0] == 'O':
63                  turno = OSO_O(Mov[1],tablero,turno)
64              else:
65                  turno = OSO_S(Mov[1],tablero,turno)
66
67              dibujarTablero(tablero)
68
69              # Estudio de Fin juego
70              if not V:
71                  seEstajugando = False
72                  print('\n\nTiempos de ejecuci n por jugada de CPU: \n')
73                  print(tiempos)
74                  print('\n Tiempo medio de jugada: ', tiemp_med/num)
75                  Final(tablero)
76                  seEstajugando = False
77                  Final(tablero)
78
79
80          else: # Turno del ordenador
81
82              t1 = time.time()
83              Mov = minimax(' ',tablero,V,True,n,f_eval)[1]
84              t2 = time.time()
85              tiempos.append(str(t2-t1))
86              tiemp_med += (t2-t1)
87              num += 1
88
89              Escribir(tablero,V,Mov)
90
91              # An lisis del turno tras jugada
92              if Mov[0] == 'O':
93                  turno = OSO_O(Mov[1],tablero,turno)
94              else:
95                  turno = OSO_S(Mov[1],tablero,turno)
96
97              dibujarTablero(tablero)
```

```
98
99              # Estudio de Fin juego
100             if not V:
101                 seEstajugando = False
102                 print('\n\nTiempos de ejecuci n por jugada de CPU: \n')
103                 print(tiempos)
104                 print('\n Tiempo medio de jugada: ', tiemp_med/num)
105                 Final(tablero)
106
107
108
109     if not revancha():
110         print( '\n \t FIN DEL JUEGO \n')
111         break
```

9. Algoritmos

Entrada: Tablero, Casilla y Turno

Salida: Valor de la evaluación

si *turno CPU* **entonces**
  c ← 1
en otro caso
  c ← −1
fin

(a,b) ← casilla

Número de osos ← haceroso(casilla, Tablero)

si *Número de osos > 0* **entonces**
  devolver (-c)*(100 + Número de osos)
fin

X ← [2,2,-1,-1,1,1,-2,-2]

Y ← [-1,1,-2,2,-2,2,-1,1]

para cada $i < 7$ **hacer**
  si *Posible(a + X[i], b + Y[i]) y Tablero(a + X[i], b + Y[i]) = "O"* **entonces**
    devolver 5*c
  fin
fin

devolver 0

Algoritmo 1: Nivel Fácil

Entrada: Tablero, Casilla y Turno

Salida: Valor de la evaluación

si *turno CPU* **entonces**
  c ← 1
en otro caso
  c ← -1
fin

(a,b) ← casilla

Número de osos ← haceroso(casilla, Tablero)

si *Número de osos > 0* **entonces**
  devolver (-c)*(100 + Número de osos)
fin

o ← 0

s ← 0

X ← [-1,-1,-1,0,0,1,1,1]

Y ← [1,0,-1,1,-1,1,0,-1]

n ← Long(Tablero[0])
para cada $i < 7$ **hacer**
  si *posible(casilla,Tablero)* **entonces**
    si *Tablero[a + X[i]][b + Y[i]] == "O"* **entonces**
      o ← o + 1
    fin
    si *Tablero[a + X[i]][b + Y[i]] == "S"* **entonces**
      s ← s + 1
    fin
  fin
fin

si *Tablero[a][b] == "S" y o - s < 0* **entonces**
  devolver c*(10 + abs(o-s))
fin

si *Tablero[a][b] == "O" y o - s > 0* **entonces**
  devolver c*(10 + abs(o-s))
fin

si *o - s == 0* **entonces**
  devolver Valor aleatorio entre 1 y 5
en otro caso
  devolver 0
fin

Algoritmo 2: Alg nivel medio

```
(a,b) ← casilla
n ← Long(Tablero[0])
Número de osos ← 0
si Tablero[a][b] == "O" entonces
    X ← [-2,-1,-2,-1,-2,-1,0,0,0,0,2,1,2,1,2,1]
    Y ← [2,1,0,0,-2,-1,2,1,-2,-1,2,1,0,0,-2,-1]
    para cada i < 15 de 2 en 2 hacer
        si PosibleOso(casilla,Tablero) entonces
            Número de osos ← Número de osos + 1
        fin
    fin
en otro caso
    X ← [-1,1,0,0,-1,1,-1,1]
    Y ← [0,0,-1,1,-1,1,1,-1]
    para cada i < 7 de 2 en 2 hacer
        si PosibleOso(casilla,Tablero) entonces
            Número de osos ← Número de osos + 1
        fin
    fin
fin
devolver Número de osos
```

Algoritmo 3: Función haceroso

```
Letra, Pos ← ∅, ∅
si  n = 0 ó V vacio entonces
    devolver f(casilla,tablero,turno), Letra, Pos
en otro caso
    si TurnoCPU entonces
        para cada  i < Longitud(V) hacer
            (a,b) ← V(i)
            Tablero(a,b) ← Escribir(V(i), i)
            T ← OSO(V(i), Tablero, turno, i)
            Actualizar(V)
            Eval ← minimax(V(i), tablero, V, T, n − 1, f, alpha, beta)
            si alpha < Eval entonces
                alpha ← Eval
                Letra, Pos ← Tablero(a,b), V(i)
            fin
            Restablercer(tablero)
            si beta ≤ alpha entonces
                Parar
            fin
        fin
        devolver alpha, Letra, Pos
    en otro caso
        para cada  i < Longitud(V) hacer
            (a,b) ← V(i)
            Tablero(a,b) ← Escribir(V(i), i)
            T ← OSO(V(i), Tablero, turno, i)
            Actualizar(V)
            Eval ← minimax(V(i), tablero, V, T, n − 1, f, alpha, beta)
            si beta > Eval entonces
                beta ← Eval
                Letra, Pos ← Tablero(a,b), V(i)
            fin
            Restablercer(tablero)
            si beta ≤ alpha entonces
                Parar
            fin
        fin
        devolver beta, Letra, Pos
    fin
fin
```

Algoritmo 4: Algoritmo Minimax

```
mientras No Parar hacer
    Reglas()
    n,f ← NiveldeJuego()
    Tablero,V ← IniciarTablero()
    Turno ← QuienVaPrimero()
    mientras seEstaJugando hacer
        si TurnoUsuario entonces
            Movimiento ← JugadaUsuario(Tablero, V)
            si Movimiento = Exit entonces
                Parar
            fin
            Turno ← OSO(Movimiento, Tablero, turno)
            DibujarTablero()
            si V vacio entonces
                FinalJuego()
            fin
        en otro caso
            Movimiento ← Minimax(∅, Tablero, V, Turno, n, f, alpha, beta)
            si Movimiento = Exit entonces
                Parar
            fin
            Turno ← OSO(Movimiento, Tablero, turno)
            DibujarTablero()
            si V vacio entonces
                FinalJuego()
            fin
        fin
    fin
fin
```

Algoritmo 5: Algoritmo Principal