



Universidad Autónoma de Nuevo León

FCFM

Inteligencia Artificial

Tarea 5 – Python

Equipo:

Alexis Emiliano Eguia Sifuentes	1861817
Leonardo Martin Vázquez Cruz	1679807
Bryan Aguirre Tudon	1735839

Definición de la tarea:

```
1 """
2 Pseudocódigo
3 funcion Minimax(S, max=true)
4     if s es terminal:
5         return f(s)
6     if Max:
7         v = -infinito
8         for each hijo en S:
9             v = max(v, Minimax(hijo, False))
10        return v
11    else:
12        v = infinito
13        for each hijo en S
14            v = min(v, MiniMax(hijo, True))
15        return v
16 #Evaluar primero el nodo(Hoja,Hijo,Inicial)
17 """
```

Programar el algoritmo minimax para el problema tic-tac-toe (El juego del gato). Como les expliqué en el salón, pueden hacerlo de diferentes maneras:

- 1) Como un algoritmo que sugiere al jugador que movimiento realizar.
- 2) Un algoritmo con el que puedan jugar, es decir, ustedes hacen un movimiento y el algoritmo usando minimax, decide el suyo.

Para cualquiera de los dos casos tienen dos formas también de resolverlo:

- 1) Expanden todo el árbol y le asignan 1 si es un estado ganador, 0 si es un empate, -1 si es un estado perdedor.
- 2) Pueden expandir solo cierta cantidad de estados y evaluar como está el tablero.

La tarea es en equipo de no más de 3 personas, de igual forma se va a entregar un reporte explicando su algoritmo, pseudocódigos y experimentos realizados, así como una conclusión de cuál fue mejor.

Programa(Interfaz Grafica)

```
1 from tkinter import *
2 from frames import *
3 from tkinter import messagebox
4
5 ventana = Tk() # VENTANA RAIZ
6 ventana.geometry("700x600") # DIMENCIONES
7 ventana.title("Tres en raya") # TITULO
8 ventana.resizable(0,0) # NO REDIMENSIONABLE
9
10 cargar_ventana(ventana) # CARGAMOS LOS OBJETOS
11
12 ventana.mainloop() # MOSTRAMOS LA PANTALLA
```

Nosotros agregamos como extra, el colocarle una interfaz grafica para que sea mas cómodo el manejo del TABLERO del juego TIC TAC TOE



Clase Nodo

Clase NODO controla nuestro árbol de posibilidades aplicando el algoritmo MINMAX

```
1 from random import randint
2 from principales import evaluar_tablero
3 class Nodo():
4     def __init__(self, tablero, turno):
5         self.tablero = tablero
6         self.turno = turno
7         self.casillas_libres = []
8         self.lista_relaciones = []
9         self.estado = evaluar_tablero(self.tablero)
10        self.valor = ""
11        if turno == 'O':
12            self.maximo = True
13        else:
14            self.maximo = False
15
16        indice = 0
17        for casilla in tablero:
18            if casilla == '.':
19                self.casillas_libres.append(indice)
20            indice += 1
21
22        if self.estado == -2:
23            for indice in self.casillas_libres:
24                tablero_aux = []
25                for casilla in tablero:
26                    tablero_aux.append(casilla)
27                    tablero_aux.insert(indice, turno)
28                    tablero_aux.pop(indice+1)
29                    if turno == "X":
30                        turno_aux = 'O'
31                    else:
32                        turno_aux = 'X'
33
34            self.lista_relaciones.append(Nodo(tablero_aux, turno_aux))
35            if len(self.lista_relaciones) != 0:
36                if self.maximo:
37                    self.nodo_optimo = self.max()
38                    self.valor = self.nodo_optimo.getValor()
39                else:
40                    self.nodo_optimo = self.min()
41                    self.valor = self.nodo_optimo.getValor()
42            else:
43                self.valor = self.getEstado()
44
45        #Metodos Get
46        def getTablero(self):
47            return self.tablero
48        def getListRelaciones(self):
49            return self.lista_relaciones
50        def getTurno(self):
51            return self.turno
52        def getCasillasLibres(self):
53            return self.casillas_libres
54        def getEstado(self):
55            return self.estado
56        def getValor(self):
57            return self.valor
58        def getNodoOptimo(self):
59            return self.nodo_optimo
60
61        #Metodos Set
62        def setTablero(self, tablero):
63            self.tablero = tablero
64        def setListaRelaciones(self, lista):
65            self.lista_relaciones = lista
66        def setTurno(self, turno):
67            self.turno = turno
68
69        #Metodos
70        def max(self):
71            long = len(self.lista_relaciones)-1
72            maximo = self.lista_relaciones[randint(0, long)]
73            for nodo_hijo in self.lista_relaciones:
74                if nodo_hijo.valor > maximo.valor:
75                    maximo = nodo_hijo
76            return maximo
77        def min(self):
78            long = len(self.lista_relaciones)-1
79            minimo = self.lista_relaciones[randint(0, long)]
80            for nodo_hijo in self.lista_relaciones:
81                if nodo_hijo.valor < minimo.valor:
82                    minimo = nodo_hijo
83            return minimo
84
85
```

En el constructor de la clase damos como parámetro el tablero y el turno del jugador y definimos atributos del nodo

Observamos cuales son las casillas vacías

Cambiamos de turno y creamos los nodos hijos del tablero que se paso como parámetro en la clase

Métodos Get, los cuales nos facilitan manipular los atributos de la clase

Métodos SET, los cuales nos permiten actualizar los atributos importantes

Método **MINMAX**

MAX -> Calcula la mejor jugada de la IA

MIN -> Calcula la mejor jugada del PLAYER

Evaluar TABLERO

```
1 def evaluar_tablero(tablero):
2     #return -1 cuando ganas
3     #return 1 cuando pierdes
4     #return 0 cuando es empate
5     #return -2 NO es nodo terminal
6     c_central = tablero[4]
7
8     if c_central != '.':
9         #Verificamos si en las digonales se ha ganado
10        if tablero[0] == c_central and tablero[8] == c_central or tablero[6] == c_central and tablero[2] == c_central :
11            if c_central == 'X':
12                return -1
13            else:
14                return 1
15
16        #Verificamos si en la fila y columna que contenga c_central se ha ganado
17        if tablero[1] == c_central and tablero[7] == c_central or tablero[3] == c_central and tablero[5] == c_central:
18            if c_central == 'X':
19                return -1
20            else:
21                return 1
22
23        #Verificamos si en las filas y columnas restantes se ha ganado
24        c_esquina1 = tablero[0]
25        if c_esquina1 != '.':
26            if tablero[1] == c_esquina1 and tablero[2] == c_esquina1 or tablero[3] == c_esquina1 and tablero[6] == c_esquina1:
27                if c_esquina1 == 'X':
28                    return -1
29                else:
30                    return 1
31
32        c_esquina2 = tablero[8]
33        if c_esquina2 != '.':
34            if tablero[2] == c_esquina2 and tablero[5] == c_esquina2 or tablero[6] == c_esquina2 and tablero[7] == c_esquina2:
35                if c_esquina2 == 'X':
36                    return -1
37                else:
38                    return 1
39
40        #Verificamos si NO es un nodo terminal
41        for i in range(0,9):
42            if tablero[i] == '.':
43                return -2
44
45        #Si no se cumple ningun caso anterior entonces es Empate
46        return 0
```

La función retorna los 4 posibles casos:

- Ganar(Que el jugador gane)
- Perder(Que el jugador pierda)
- Empate(No gana la IA ni el jugador)
- En curso(La partida aun no concluye)

Codigo Intefaz Grafica

```
1 from tkinter import *
2 from tkinter import messagebox
3 from principales import evaluar_tablero, imprimir_tablero
4 from nodo import Nodo
5
6 class Marcador():
7
8     def __init__(self):
9         self.victorias=0
10        self.derrotas=0
11        self.empates=0
12
13    def getVic(self):
14        return self.victorias
15    def getDer(self):
16        return self.derrotas
17    def getEmp(self):
18        return self.empates
19
20    def incVic(self):
21        self.victorias += 1
22    def incDer(self):
23        self.derrotas += 1
24    def incEmp(self):
25        self.empates += 1
26
27    def incVic(texto):
28        marc.incVic()
29        texto.config(text=f"Victorias: {marc.getVic()}")
30    def incDer(texto):
31        marc.incDer()
32        texto.config(text=f"Derrotas: {marc.getDer()}")
33    def incEmp(texto):
34        marc.incEmp()
35        texto.config(text=f"Empates: {marc.getEmp()}")
36
37    marc=Marcador()
38    v=0
39    e=0
40    d=0
41    lista=[]
42
43    def cargar_ventana(ventana):
44        # CREAMOS EL CONTENEDOR
45        encabezado = Frame(ventana, width=700, height=120)
46
47        # AGREGAMOS ESTILOS AL CONTENEDOR
48        encabezado.config(bg="Black")
49
50        # OBJETOS DEL CONTENEDOR
51        titulo = Label(encabezado,width=12)
52        titulo.config(
53            text="Tres en raya",
54            bg="Black",
55            fg="Yellow",
56            font=('Stencil',40)
57        )
58        titulo.pack()
59
60        # EMPAQUETAMOS CONTENEDOR
61        encabezado.pack(fill=X, expand=True, side=TOP, anchor=N)
62        encabezado.pack_propagate(False)
63
64        #FUENTE Y TAMAÑO DE LA LETRA
65        fuente = ('Fixedsys',18)
66        fondo = "Black"
67        color = "lime"
68
69        # CREAMOS EL CONTENEDOR
70        marcador = Frame(encabezado, width=700, height=50)
71
72        # AGREMOS ESTILO AL CONTENEDOR
73        marcador.config(bg=fondo)
74
75        # OBJETOS DEL CONTENEDOR
76        victorias = Label(marcador)
77        empates = Label(marcador)
78        derrotas = Label(marcador)
79
80        # AGREGAMOS ESTILOS A LOS OBJETOS
81        victorias.config(
82            text=f"Victoria: {marc.getVic()}",
83            width=10,
84            bg=fondo,
85            fg=color,
86            font=fuente
87        )
88        empates.config(
89            text=f"Empates: {marc.getEmp()}",
90            width=10,
91            bg=fondo,
92            fg=color,
93            font=fuente
94        )
95        derrotas.config(
96            text=f"Derrotas: {marc.getDer()}",
97            width=10,
98            bg=fondo,
99            fg=color,
100            font=fuente
101        )
```

```

1 # EMPAQUETAMOS LOS OBJETOS AL CONTENEDOR
2 victorias.pack(side=LEFT, fill=X, expand=True)
3 empates.pack(side=LEFT, fill=X, expand=True)
4 derrotas.pack(side=LEFT, fill=X, expand=True)
5
6 # EMPAQUETAMOS EL CONTENEDOR
7 marcador.pack(fill=X, expand=True, side=TOP, anchor=N)
8 marcador.pack_propagate(False)
9
10
11 # FONDO DEL CONTENEDOR(menu)
12 fondo="lightBlue"
13 # FUENTE
14 fuente = "Fixedsys"
15
16 # CREAMOS EL CONTENEDOR
17 menu = Frame(ventana)
18
19 # AGREGAMOS ESTILO AL CONTENEDOR
20 menu.config(
21     width=200,
22     height=480,
23     bg=fondo
24 )
25
26 # OBJETOS DEL CONTENEDOR
27 titulo = Label(menu)
28 empezar = Button(menu)
29 reiniciar = Button(menu)
30 pregunta = Label(menu)
31 respuesta1 = Radiobutton(menu)
32 respuesta2 = Radiobutton(menu)
33 void = Label(menu)
34 void1 = Label(menu)
35 void2 = Label(menu)
36 opcion = StringVar()
37 opcion.set(None)
38
39 def f_reiniciar(tablero):
40     for casilla in tablero:
41         casilla.config(
42             text=" ",
43             fg="Black"
44         )
45
46 # AGREGAR ESTILOS A LOS OBJETOS
47 titulo.config(
48     text="Menu Principal",
49     bg=fondo,
50     font=(fuente,17)
51 )
52 empezar.config(
53     text="Empezar",
54     font=(fuente,16),
55     padx=10,
56     pady=10,
57 )
58 reiniciar.config(
59     text="Reiniciar",
60     font=(fuente,16),
61     command=lambda:f_reiniciar(lista),
62     padx=10,
63     pady=10
64 )
65 pregunta.config(
66     text="¿Quien iniciara\nla partida?",
67     font=(fuente,16),
68     padx=10,
69     bg=fondo
70 )
71 respuesta1.config(
72     text="Player",
73     value="Player",
74     variable=opcion,
75     font=(fuente,16),
76     bg=fondo
77 )
78 respuesta2.config(
79     text="Maquina",
80     value="Maquina",
81     variable=opcion,
82     font=(fuente,16),
83     bg=fondo
84 )
85 void.config(
86     text=" ",
87     font=('Mistral',1),
88     height=1,
89     bg=fondo
90 )
91 void1.config(
92     text=" ",
93     font=('Mistral',1),
94     height=1,
95     bg=fondo
96 )
97 void2.config(
98     text=" ",
99     font=('Mistral',12),
100     height=1,
101     bg=fondo
102 )

```

```

1 # EMPAQUETAMOS LOS OBJETOS
2 titulo.pack()
3 void.pack()
4 empoar.pack()
5 void1.pack()
6 reiniciar.pack()
7 void2.pack()
8 pregunta.pack()
9 respuesta1.pack()
10 respuesta2.pack()
11
12 # EMPAQUETAMOS EL CONTENEDOR
13 menu.place(x=0,y=120)
14 menu.pack_propagate(False)
15
16 #FUENTE Y TAMAÑO DE LA LETRA(tablero)
17 fuente=('Fixedsys',50)
18
19 # FONDO DEL CONTENEDOR(tablero)
20 fondo="#31FA31"
21
22 # CREAMOS EL CONTENEDOR
23
24 tablero = Frame(ventana)
25
26 # AGREGAMOS LOS ESTILOS AL CONTENEDOR
27 tablero.configure(
28     width=500,
29     height=400,
30     bd=15,
31     bg=fondo
32 )
33
34 # OBJETOS DEL CONTENEDOR
35 fila1 = Frame(tablero)
36 fila2 = Frame(tablero)
37 fila3 = Frame(tablero)
38 casilla1 = Button(fila1)
39 casilla2 = Button(fila1)
40 casilla3 = Button(fila1)
41 casilla4 = Button(fila2)
42 casilla5 = Button(fila2)
43 casilla6 = Button(fila2)
44 casilla7 = Button(fila3)
45 casilla8 = Button(fila3)
46 casilla9 = Button(fila3)
47 void = Label(tablero)
48
49 # AGREGAMOS LOS ESTILOS A LOS OBJETOS
50 casilla1.config(
51     text=" ",
52     width=3,
53     height=1,
54     padx=10,
55     pady=10,
56     font=fuente,
57     relief=SOLID,
58     bg=fondo,
59     fg="Black",
60     command=lambda: colocar_ficha(casilla1,opcion,victorias,derrotas,empates)
61 )
62 casilla2.config(
63     text=" ",
64     width=3,
65     height=1,
66     padx=10,
67     pady=10,
68     font=fuente,
69     relief=SOLID,
70     bg=fondo,
71     fg="Black",
72     command=lambda: colocar_ficha(casilla2,opcion,victorias,derrotas,empates)
73 )
74 casilla3.config(
75     text=" ",
76     width=3,
77     height=1,
78     padx=10,
79     pady=10,
80     font=fuente,
81     relief=SOLID,
82     bg=fondo,
83     fg="Black",
84     command=lambda: colocar_ficha(casilla3,opcion,victorias,derrotas,empates)
85 )
86 casilla4.config(
87     text=" ",
88     width=3,
89     height=1,
90     padx=10,
91     pady=10,
92     font=fuente,
93     relief=SOLID,
94     bg=fondo,
95     fg="Black",
96     command=lambda: colocar_ficha(casilla4,opcion,victorias,derrotas,empates)
97 )
98 casilla5.config(
99     text=" ",
100     width=3,
101     height=1,
102     padx=10,
103     pady=10,
104     font=fuente,
105     relief=SOLID,
106     bg=fondo,
107     fg="Black",
108     command=lambda: colocar_ficha(casilla5,opcion,victorias,derrotas,empates)
109 )
110 casilla6.config(
111     text=" ",
112     width=3,
113     height=1,
114     padx=10,
115     pady=10,
116     font=fuente,
117     relief=SOLID,
118     bg=fondo,
119     fg="Black",
120     command=lambda: colocar_ficha(casilla6,opcion,victorias,derrotas,empates)
121 )
122 casilla7.config(
123     text=" ",
124     width=3,
125     height=1,
126     padx=10,
127     pady=10,
128     font=fuente,
129     relief=SOLID,
130     bg=fondo,
131     fg="Black",
132     command=lambda: colocar_ficha(casilla7,opcion,victorias,derrotas,empates)
133 )
134 casilla8.config(
135     text=" ",
136     width=3,
137     height=1,
138     padx=10,
139     pady=10,
140     font=fuente,
141     relief=SOLID,
142     bg=fondo,
143     fg="Black",
144     command=lambda: colocar_ficha(casilla8,opcion,victorias,derrotas,empates)
145 )
146 casilla9.config(
147     text=" ",
148     width=3,
149     height=1,
150     padx=10,
151     pady=10,
152     font=fuente,
153     relief=SOLID,
154     bg=fondo,
155     fg="Black",
156     command=lambda: colocar_ficha(casilla9,opcion,victorias,derrotas,empates)
157 )
158 void.config(
159     text=" ",
160     font=('Mistral',12),
161     width=1,
162     height=60,
163     bg=fondo
164 )

```



```

1  # ENPAQUETAMOS LOS OBJETOS
2  void.pack(side=LEFT, anchor=W)
3  fila1.pack()
4  fila2.pack()
5  fila3.pack()
6  casilla1.pack(side=LEFT, anchor=W)
7  casilla2.pack(side=LEFT, anchor=W)
8  casilla3.pack(side=LEFT, anchor=W)
9  casilla4.pack(side=LEFT, anchor=W)
10 casilla5.pack(side=LEFT, anchor=W)
11 casilla6.pack(side=LEFT, anchor=W)
12 casilla7.pack(side=LEFT, anchor=W)
13 casilla8.pack(side=LEFT, anchor=W)
14 casilla9.pack(side=LEFT, anchor=W)
15
16 #LISTA(tablero)
17 lista.append(casilla1)
18 lista.append(casilla2)
19 lista.append(casilla3)
20 lista.append(casilla4)
21 lista.append(casilla5)
22 lista.append(casilla6)
23 lista.append(casilla7)
24 lista.append(casilla8)
25 lista.append(casilla9)
26
27 # ENPAQUETAMOS EL CONTENEDOR
28 tablero.place(x=200, y=120)
29 tablero.pack_propagate(False)
30
31 def leer_tablero(lista):
32     tablero=[]
33     for cas in lista:
34         c=cas['text']
35         c=c.upper()
36         if c != '.':
37             tablero.append(c)
38         else:
39             tablero.append('.')
40     return tablero
41
42 def
colocar_ficha(casilla,opcion,victorias,derrotas,empates):
43     if opcion.get() != 'None':
44         tablero = leer_tablero(lista)
45         vacio=True
46         for cas in tablero:
47             if cas != '.':
48                 vacio = False
49                 break
50
51     #COLOCAMOS FICHA
52     if vacio and opcion.get()=='Maquina':
53         if evaluar_tablero(tablero) == -2:
54             nodo_inicial = Nodo(tablero, 0)
55             nodo_optimo = nodo_inicial.nodo_optimo
56             tablero.nodo_optimo =
57             nodo_optimo.getTablero()
58             casilla_optima = 0
59             for casilla_i in tablero:
60                 if tablero[casilla_optima] !=
61                     tablero_nodo_optimo[casilla_optima]:
62                         break
63                 casilla_optima += 1
64                 casilla = lista[casilla_optima]
65                 casilla.config(
66                     text='o',
67                     fg='blue'
68                 )
69                 tablero = leer_tablero(lista)
70                 imprimir_tablero(tablero)
71             else:
72                 if evaluar_tablero(tablero) == -2 and
73                     casilla['text']=='.':
74                         casilla.config(
75                             text='x',
76                             fg='red'
77                         )
78                 tablero=leer_tablero(lista)
79                 resultado=evaluar_tablero(tablero)
80
81     #Creamos el arbol y buscamos la mejor
82     jugada para la IA
83     if resultado == -2:
84         nodo_inicial = Nodo(tablero, 0)
85         nodo_optimo = nodo_inicial.nodo_optimo
86         tablero.nodo_optimo =
87         nodo_optimo.getTablero()
88         casilla_optima = 0
89         for casilla_i in tablero:
90             if tablero[casilla_optima] !=
91                 tablero_nodo_optimo[casilla_optima]:
92                     break
93             casilla_optima += 1
94             casilla = lista[casilla_optima]
95             casilla.config(
96                 text='o',
97                 fg='blue'
98             )
99             else:
100                 messagebox.showinfo("Partida", "Selecciona quien
101                     iniciara la partida:")
102                 tablero_l = leer_tablero(lista)
103                 resultado=evaluar_tablero(tablero_l)
104                 if resultado==1: #gana
105                     messagebox.showinfo("Resultado de la
106                         partida", "Felicitades has ganado!")
107                     incvic(victorias)
108                 elif resultado==1: #perdiste
109                     messagebox.showinfo("Resultado de la
110                         partida", "Suerte para la proxima!\nPerdiste :(")
111                     incder(derrotas)
112                 elif resultado==0: #empate
113                     messagebox.showinfo("Resultado de la
114                         partida", "Nadie gano\nsera la proxima ves!")
115                     incEmp(empates)
116
117     a1 = 'X':
118         return -1
119     else:
120         return 1
121
122     c_esquina2 = tablero[8]
123     if c_esquina2 != '.':
124         if tablero[2] == c_esquina2 and tablero[5] ==
125             c_esquina2 or tablero[6] == c_esquina2 and tablero[7] ==
126             c_esquina2:
127                 if c_esquina2 == 'X':
128                     return -1
129                 else:
130                     return 1
131
132     #Verificamos si NO es un nodo terminal
133     for i in range(0,9):
134         if tablero[i] == '.':
135             return -2
136
137     #Si no se cumple ningun caso anterior entonces es
138     Empate
139     return 0

```