MATEMATICAS DISCRETAS

GRUPO 3:

Ricardo Falcon

Juan Lopez


INICIO
Inicia agente1 o agente2
Centro ocupado
Juega centro
Juega diagonal
MINIMAX
Retorna 2 hojas o mas con valores iguales


Verifica en las validaciones
Juega con la de mayor valor
Solo un espacio disponible
juega
EMPATE
TRIUNFO
FIN

arbol.m | main.m | nodo.m | playAgentStudent.m

```
% Programa para el proyecto de Matemáticas Discretas
%
% Marco E. Benalcázar
% marco.benalcazar@epn.edu.ec
%
clc;
close all;
%clear all;
warning off all;

% Añade la librería para visualizar el tablero del 'tic-tac-toe'
addpath('Toolbox TicTacToe');

% Lista de opciones para el juego:
% -----------------------------------
%
% Opción 1: Agente de estudiantes vs. juego manual
% Opción 2: Agente del profesor vs. juego manual
% Opción 3: Agente de estudiantes vs. agente del profesor
%
% NOTA: En las opciones 2 y 3 se ha incluído como agente del profesor uno
%       que realiza movimientos aleatorios. El agente del profesor que realiza
%       movimientos "inteligentes" se pondrá a prueba el día de la defensa del
%       del presente proyecto.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           AQUÍ DEBE CAMBIAR EL VALOR DE LA VARIABLE "OPTION"
%
% IMPORTANTE: Colocar el valor de la variable option en 1 cuando
% quiera probar su agente con un usuario; caso contrario, coloque el valor
% en 3 cuando quiera verificar que su programa es capaz de interactuar con
% el agente el profesor. Recuerde que la versión del agente implementado por el
% profesor que viene con este código no es la que se usará el día de la defensa del
% presente proyecto.
option = 3;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tictactoe(option);
```

arbol.m | main.m | nodo.m | playAgentStudent.m

```
1  classdef nodo < handle
2
3      properties
4          contenido;
5          rama;
6          valor;
7      end
8
9      methods
10         function obj = nodo(board)
11             obj.contenido = board;
12             obj.rama(:) = [];
13             obj.valor = 0;
14         end
15
16         function setValor(obj, num)
17             obj.valor = num;
18         end
19
20         function root = getValor(obj)
21             root = obj.valor;
22         end
23
24         function setRama(obj, nodo, ind)
25             obj.rama(ind) = nodo;
26         end
27
28         function root = getNodoContent(obj)
29             root = obj.contenido;
30         end
31
32         function setNodoContent(obj, board)
33             obj.contenido = board;
34         end
35
36
37     end
```

arbol.m | main.m | nodo.m | playAgentStudent.m*

```matlab
pause(0.7);
board = board';
board = board(:)';
% Encuentra las posiciones que no han sido ocupadas por ningún jugador
vector = 1:9;
availablePositions = vector(board == 0);


%SE DECLARA LA MATRIZ COMO UN VECTOR DE 0 - 9 PARA LAS POSICIONES
mesa(1,1) = board(1);
mesa(1,2) = board(2);
mesa(1,3) = board(3);
mesa(2,1) = board(4);
mesa(2,2) = board(5);
mesa(2,3) = board(6);
mesa(3,1) = board(7);
mesa(3,2) = board(8);
mesa(3,3) = board(9);

a = arbol();
a.crearRaiz(mesa)
a.crearArbolNivel1(mesa)

for i=1:numel(a.raiz.rama)
    a.crearArbolNivel2(a.raiz.rama(i).contenido, i);
end


% CODIGO ALEATORIO SUMINISTRADO EN EL BASE
% % Selecciona aleatoriamente una posición vacía
% [dummy, idx] = sort(  rand( size(availablePositions) )  );
% [col, row] = ind2sub(  [3 3], availablePositions( idx(1) )  );
[row, col] = MejorJugada(a, availablePositions);
clear a;

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
return
```

arbol.m* | main.m | nodo.m | playAgentStudent.m

```
classdef arbol < handle

    properties
        raiz;

    end

    methods

        function obj = arbol()
            obj.raiz = nodo(zeros(3,3));  %la matriz o mesa de ceros
        end

        function raiz = getRaiz(arbol)
            raiz = arbol.raiz;
        end

        function crearRaiz(obj, board)
            obj.raiz = nodo(board);
            obj.raiz.rama = nodo([]);
        end

        function crearArbolNivel1(obj, board)
            contador = 1;
            for i=1:3
                for j=1:3
                    board2 = board;
                    if(board(i,j)==0)
                        board2(i,j) = 1;
                        obj.raiz.rama(contador) = nodo(board2);
                        obj.raiz.rama(contador).rama = nodo([]);
                        contador = contador + 1;
                    end
                end
            end
        end

```

arbol.m | main.m | nodo.m | playAgentStudent.m

```
38        function crearArbolNivel2(obj, board, ind)
39            contador = 1;
40            for i=1:3
41                for j=1:3
42                    board2 = board;
43                    if(board(i,j)==0)
44                        board2(i,j) = 2;
45                        obj.raiz.rama(ind).rama(contador) = nodo(board2);
46                        contador = contador + 1;
47                    end
48                end
49            end
50        end
51
52        function [row, col] = MejorJugada(arbol, avalaiblePositions)
53            fd = 0;
54            jugadaMinim = zeros(3,3);
55            jugadaMaxim = zeros(3,3);
56            min = 111;
57            max = -111;
58            MAX = 0;
59            MIN = 0;
60
61
62            board = arbol.raiz.contenido;
63
64                 if((board(2,2)==2 && board(1,1)==1) && board(3,3)==2 && board(1,2)==0 && board(1,3)==0 && board(2,1)==0 && board(2,3)==0 && board(3,2)==0)
65                       row=3;
66                       col=1;
67                       return
68                 end
69
70
71
72            for i=1:numel(arbol.raiz.rama)
73          %       Ocupar el centro de ser posible
74                for m=1:numel(avalaiblePositions)
```

arbol / MejorJugada

FILE NAVIGATE EDIT BREAKPOINTS RUN

arbol.m | main.m | nodo.m | playAgentStudent.m | +

```
62 -            board = arbol.raiz.contenido;
63
64 -                if((board(2,2)==2 && board(1,1)==1) && board(3,3)==2 && board(1,2)==0 && board(1,3)==0 && board(2,1)==0 && board(2,3)==0 && board(3,2)==0)
65 -                    row=3;
66 -                    col=1;
67 -                    return
68 -                end
69
70
71
72 -          for i=1:numel(arbol.raiz.rama)
73 -        %       Ocupar el centro de ser posible
74 -              for m=1:numel(avalaiblePositions)
75 -                 if(avalaiblePositions(m)==5)
76 -                    row = 2;
77 -                    col = 2;
78 -                    return
79 -                 end
80 -              end
81
82 -           %     ocupa la ultima posicion de la mesa
83 -              if(numel(avalaiblePositions) == 1)
84 -                  for m=1:3
85 -                      for n=1:3
86 -                          if(arbol.raiz.contenido(m,n)==0)
87 -                              row = m;
88 -                              col = n;
89 -                              return
90 -                          end
91 -                      end
92 -                  end
93 -              end
94
95
```

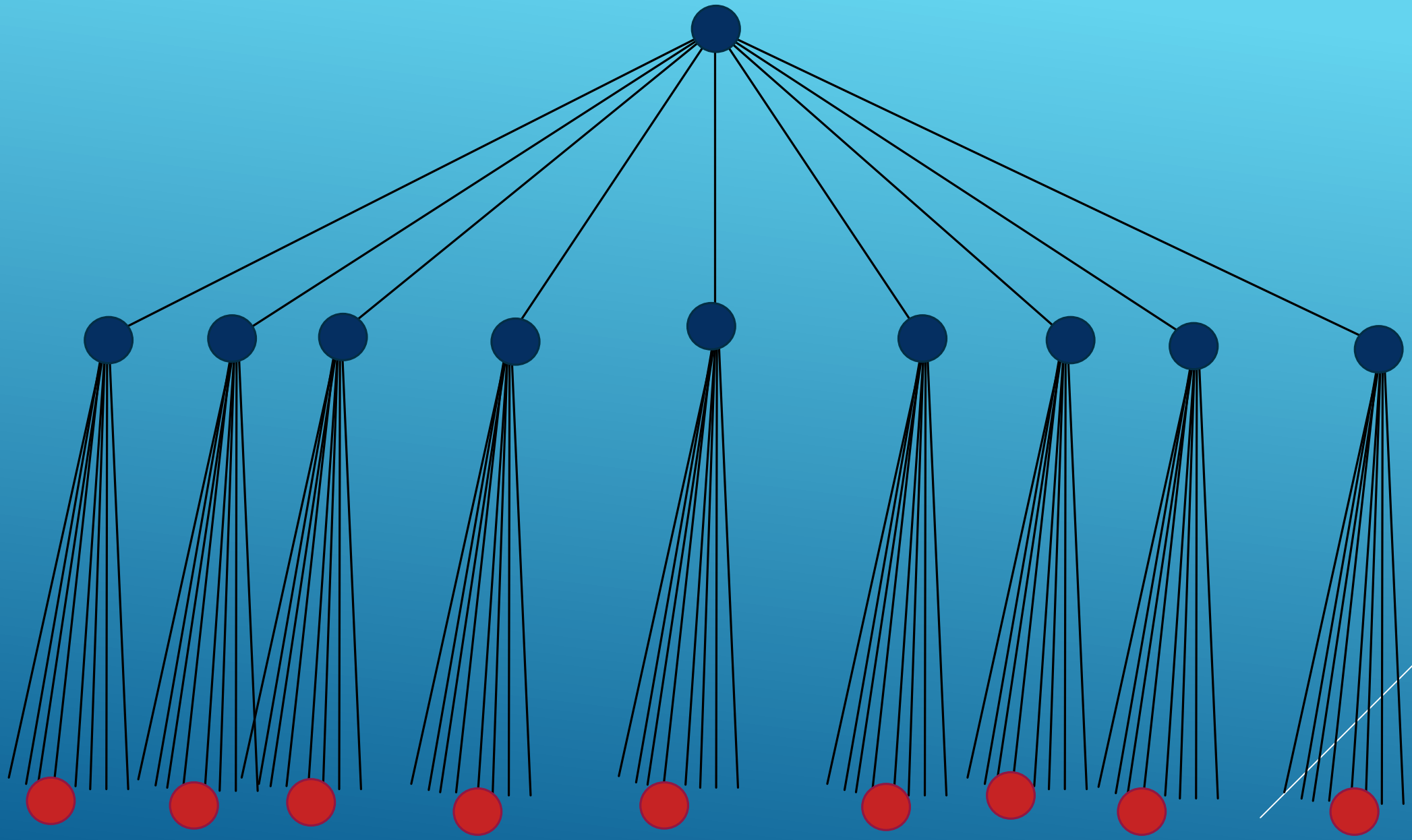
arbol.m* | main.m | nodo.m | playAgentStudent.m

```
95                  %MINIMAX
96                 for j=1:numel(arbol.raiz.rama(i).rama)
97                     jugadaMinim = arbol.raiz.rama(i).rama(j).contenido;
98                     for k=1:3
99                        %Se calcula el numero de filas y columnas que MIN
100                       %puede llegar a ocupar para ganar
101                       %---------------
102                       % en filas
103                       if((jugadaMinim(k,1)==2||jugadaMinim(k,1)==0) && (jugadaMinim(k,2)==2||jugadaMinim(k,2)==0) && (jugadaMinim(k,3)==2||jugadaMinim(k,3)==0))
104                           MIN = MIN + 1;
105
106                       end
107
108                       if((jugadaMinim(1,k)==2||jugadaMinim(1,k)==0) && (jugadaMinim(2,k)==2||jugadaMinim(2,k)==0) && (jugadaMinim(3,k)==2||jugadaMinim(3,k)==0))
109                           MIN = MIN + 1;
110
111                       end
112                       %Se calcula el numero de filas y columnas que MAX
113                       %puede llegar a ocupar para ganar
114                       %-----------------
115                       %en columnas
116                       if((jugadaMinim(k,1)==1||jugadaMinim(k,1)==0) && (jugadaMinim(k,2)==1||jugadaMinim(k,2)==0) && (jugadaMinim(k,3)==1||jugadaMinim(k,3)==0))
117                           MAX = MAX + 1;
118
119                       end
120                       if((jugadaMinim(1,k)==1||jugadaMinim(1,k)==0) && (jugadaMinim(2,k)==1||jugadaMinim(2,k)==0) && (jugadaMinim(3,k)==1||jugadaMinim(3,k)==0))
121                           MAX = MAX + 1;
122
123                       end
124                       if(jugadaMinim(k,1)==1 && jugadaMinim(k,2)==1 && jugadaMinim(k,3)==1)
125                           MAX = MAX + 111;
126
127                       end
128                       if(jugadaMinim(1,k)==1 && jugadaMinim(2,k)==1 && jugadaMinim(3,k)==1)
129                           MAX = MAX + 111;
130
131                       end
```

arbol.m | main.m | nodo.m | playAgentStudent.m

```
                    end
                end
                    %si MIN puede llegar ocupar las diagonales (calculada por minimax)
                    if((jugadaMinim(1,1)==2||jugadaMinim(1,1)==0) && (jugadaMinim(2,2)==2||jugadaMinim(2,2)==0) && (jugadaMinim(3,3)==2||jugadaMinim(3,3)==0))
                        MIN = MIN + 1;
                    end
                    if((jugadaMinim(1,3)==2||jugadaMinim(1,3)==0) && (jugadaMinim(2,2)==2||jugadaMinim(2,2)==0) && (jugadaMinim(3,1)==2||jugadaMinim(3,1)==0))
                        MIN = MIN + 1;
                    end
                    %si MAX puede llegar ocupar las diagonales (calculada por minimax)
                    if((jugadaMinim(1,1)==1||jugadaMinim(1,1)==0) && (jugadaMinim(2,2)==1||jugadaMinim(2,2)==0) && (jugadaMinim(3,3)==1||jugadaMinim(3,3)==0))
                        MAX = MAX + 1;
                    end
                    if((jugadaMinim(1,3)==1||jugadaMinim(1,3)==0) && (jugadaMinim(2,2)==1||jugadaMinim(2,2)==0) && (jugadaMinim(3,1)==1||jugadaMinim(3,1)==0))
                        MAX = MAX + 1;
                    end
                    if(jugadaMinim(1,1)==1 && jugadaMinim(2,2)==1 && jugadaMinim(3,3)==1)
                        MAX = MAX + 111;
                    end
                    if(jugadaMinim(1,3)==1 && jugadaMinim(2,2)==1 && jugadaMinim(3,1)==1)
                        MAX = MAX + 111;
                    end
                    fd = MAX - MIN;
                    arbol.raiz.rama(i).rama(j).setValor(fd);
                    fd = 0;
                    MAX = 0;
                    MIN = 0;
              end
              %%%%%%%%%%%%%%%%%%%%%%%%%%%


            %  SI MINIMAX DEVUELVE DOS VALORES DE HOJAS IGUALES ENTRA A
            %  VALIDAR
%             SI HAY DOS EN FILA Y ELIGE LA MEJOR
             for k=1:3
              if((board(k,1)==0 && board(k,2)==1) &&  board(k,3)==1 )
                 row = k;
```



arbol.m | main.m | nodo.m | playAgentStudent.m

```
61
62                  %  SI MINIMAX DEVUELVE DOS VALORES DE HOJAS IGUALES ENTRA A
63                  %  VALIDAR
64      %             SI HAY DOS EN FILA Y ELIGE LA MEJOR
65 -                 for k=1:3
66 -                  if((board(k,1)==0 && board(k,2)==1) &&  board(k,3)==1 )
67 -                     row = k;
68 -                     col = 1;
69 -                     return
70 -                     end
71 -                 end
72 -                  for k=1:3
73 -                   if((board(k,1)==0 && board(k,2)==2) &&  board(k,3)==2 )
74 -                     row = k;
75 -                     col = 1;
76 -                     return
77 -                     end
78 -                  end
79 -                  for k=1:3
80 -                   if((board(k,1)==1 && board(k,2)==0) &&  board(k,3)==1 )
81 -                     row = k;
82 -                     col = 2;
83 -                     return
84 -                     end
85 -                  end
86 -                 for k=1:3
87 -                   if((board(k,1)==2 && board(k,2)==0) &&  board(k,3)==2 )
88 -                     row = k;
89 -                     col = 2;
90 -                     return
91 -                     end
92 -                  end
93
94 -                  for k=1:3
95 -                   if((board(k,1)==1 && board(k,2)==1) &&  board(k,3)==0 )
96 -                     row = k;
97 -                     col = 3;
```

```
330 -                   for m=1:numel(arbol.raiz.rama(i).rama)
331 -                       if(arbol.raiz.rama(i).rama(m).getValor < min)
332 -                           min = arbol.raiz.rama(i).rama(m).getValor;
333 -                       end
334 -                   end
335 -                   arbol.raiz.rama(i).setValor(min);
336 -           end
337
338 -               for i=1:numel(arbol.raiz.rama)
339 -                   if(arbol.raiz.rama(i).getValor >= max)
340 -                       max = arbol.raiz.rama(i).getValor;
341 -                       jugadaMaxim = arbol.raiz.rama(i).contenido;
342 -                   end
343 -               end
344
345 -               arbol.raiz.setValor(max);
346
347 -               for i=1:3
348 -                   for j=1:3
349 -                       if(arbol.raiz.contenido(i,j) ~= jugadaMaxim(i,j))
350 -                           row = i;
351 -                           col = j;
352 -                           return
353 -                       end
354 -                   end
355 -               end
356 -           end
357 -       end
358 -   end
359
```

GRACIAS POR SU ATENCION