Lecture 26 Minimax strategy

Tic-tac-toe

We represent a Tic-tac-toe board with a $3 \times 3$ matrix. The user plays a position by entering the position on the keyboard numpad. Positions already played are marked X or O. Here is a typical board:

<table>
<thead>
<tr>
<th></th>
<th>O</th>
<th></th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>numpad</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>positions</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

Classwork 26.1(6) tictactoe.f95 Write a program which allows two players to play Tic-tac-toe. X always starts.

!c26_1_6tictactoe.f95

program tic_tac_toe
integer :: move !numpad entry 7,8,9; 4,5,6; 1,2,3.
character(1) :: b(3,3)="-"
character(1),parameter :: undecided=""
5 format((3(1x,a1)))
print*,"Enter 0 to quit."
print 5,(b(i,:),i=1,3)
do
    print*,"Enter numpad position for ", X_O(b)
    read*,move
    if(move==0) _____
    if(move<1 .or. move>9) _____
    i = 3-(move-1)/3 !get i coordinate of move
    j= ___________ !get j coordinate
    if(b(i,j)/="-" ) _______ !occupied position
    b(i,j) = X_O(b) !=X or O depends on turn
    print 5,(b(i,:),i=1,3);
    if(winner(b)/=undecided) then
        print*,winner(b)," wins"; _____
    endif
enddo

contains
corer(6) function winner(b)
corer(1):::v(8,3),b(3,3)
winner=undecided
v(1,:)=b(1,:);v(2,:)=b(2,:);v(3,:)=b(3,:)
v(4,:)=b(:,1);v(5,:)=b(:,2)
    !replace with cases for v(6,:) ... v(8,:)
do  i=1,8
if(all(v(i,:)=="X") )then

Winning a two-person game such as TicTacToe is harder than solving a one-person puzzle such as the 8-puzzle.
MiniMax strategy

Player I and player II alternately take turns making moves, player I moves first. If I wins, he wins $1.00, II loses $1.00. If II wins, he gets $1.00 and I loses $1.00. In case of a draw, both get $0.00. The payoff is the amount I wins. It is also what II loses. Hence I wants to maximize the payoff (his winnings) and II wants to minimize the payoff (his losses). A negative payoff is a loss for I and a gain for II. I likes positive payoffs, II likes negative payoffs (a negative loss is a positive win).

Possible moves are pictured as a downward choices in a tree. The nodes of the tree are the game’s possible states (configurations, boards). The start node (player I’s turn) is at the root at the top. Nodes at odd levels are boards for which it is I’s turn to move; nodes at even levels are for II. For each move a play can make, there is a downward edge to the next level. The game ends at nodes at the bottom of the tree. We use the minimax strategy to calculate the payoff (I’s winnings) for each node of the tree when both players play optimal games. Nodes at the bottom of the tree (leaves) are configurations where the game ends. They are given payoffs 1, -1, 0 if the configuration they represent are a win for I,
a win for $II$ or a draw. Given a node, suppose, by recursion, that payoffs have been assigned to lower nodes. At nodes in levels for $I$, (alpha stages) the payoff is the maximum of the payoffs on nodes below it. $I$ will pick a move which leads to a lower node of maximum payoff since he wants to maximize his winnings. If it is $II$'s turn (beta stages) the payoff is the minimum of the payoffs of nodes below it. $II$ will pick a move which leads to a lower node of minimum payoff since he wants to minimize his loss. In the picture, calculate the payoffs for the nodes of the tree below and which moves each player should take.
For TicTacToe, the nodes are boards for the game with the empty board at the root. Nodes where I (i.e., player X) wins have payoffs 1. Nodes where O wins have payoffs -1. Draws have payoff 0.

**Alpha-beta pruning**

At a player I stage (alpha stage) you search for a maximum; at a II stage (beta stage) you search for a minimum. The standard way to pick a maximum is to search the whole vector and then pick the largest. But since 1 is the largest possible value for minimax vectors, you may stop an alpha search whenever you hit a 1 (there is no larger value). When searching [0,1,-1,0,-1,1,1] at a maximum (alpha) stage you can stop at position 2 (there is no larger value). At a minimum (beta) stage, you can stop at position 3 (there is no smaller value).

**Classwork 26.2(6) tictactoe_ai.f95** Write a program which plays an optimal Tic-tac-toe game for player O against a human player X. X always starts.
program tictactoe_ai
integer:: move, payoff, payoff2
integer, parameter:: inf=10**6
character(1):: b(3,3), a(3,3)
character, parameter:: undecided=""
b(1,:)=(/"-","-","-"/)
b(2,:)=(/"-","X","-"/)
b(3,:)=(/"-","-","-"/)
print *,"Enter 0 to quit."
5 format(3(1x,a1))
print 5,(b(k,:),k=1,3) ; print *
do

  !Human move
  print *,"Enter numpad position for ",X_O(b)
  read*,move
  if(move==0) exit
  if(move<1 .or. move>9) cycle
  i=3-(move-1)/3 ! get i coordinate of move
  j=__________ ! get j coordinate
if(b(i, j)/="-") ______ ! occupied position
b(i, j)= X_O(b);  !=X or O depends on turn
print 5,(b(k,:),k=1,3) ; print*
if(winner(b)/=undecided) then
  print*,"Winner is ", winner(b);_____
endif

!Computer move

call get_payoff(b,payoff,inxt,jnxt)

b(inxt, jnxt)=X_O(b)
print 5,(b(k,:),k=1,3) ; print*
if(winner(b)/=undecided) then
  print*,"Winner is ", winner(b);_____
endif
enddo

contains

recursive subroutine get_payoff(b,payoff,inxt,jnxt)
!payoff = best payoff for X, inxt, jnxt = best position
character(1)::b(3,3),a(3,3)
integer::i,j,payoff,payoff2,inxt,jnxt,inxt2,jnxt2
if(winner(b)/=undecided)then;
select case(winner(b))
    case("X"); payoff= 1
    case("O"); payoff= ____________
    case("Nobody"); payoff= ____________
endselect
return;
endif
select case(X_O(b))
    case("X"); payoff=-inf
    case("O"); payoff=inf
endselect
do i=1,3; do j=1,3
    if(b(i,j)/=' ')cycle
    a=b;
    a(i,j)=X_O(b)
call get_payoff(a,payoff2,inxt2,jnxt2)
if(X_O(b)=="X")then
    if(payoff<payoff2)then
        payoff=payoff2; inxt=i; jnxt=j;
        !alpha prune position, may exit if payoff=1
endif
else
  if(payoff>payoff2) then
    payoff=payoff2; inxt=i; jnxt=j;
    ! beta prune position, may exit if payoff = -1
  endif
endif
enddo; enddo
endsubroutine

character(6) function winner(b)
character(1):: v(8,3), b(3,3)
winner=undecided
v(1,:)=b(1,:); v(2,:)=b(2,:); v(3,:)=b(3,:)
v(4,:)=b(:,1); v(5,:)=b(:,2)
  ! replace with cases for v(6,:), ..., v(8,:)
do i=1,8
  if(all(v(i,:)=="X")) then
    winner="X"; return; endif
if(all( ___________ )) then
  winner= ___________ ; return; endif
enddo
if(all(b/="-")) then; winner="nobody"; return; endif
endfunction

character(1) function X_O(b)
character(1):: b(3,3)
If(mod(count(b/="-"),2)==0) then; X_O="X"
else; X_O="O"
endif
endfunction
endprogram

CLASSWORK  26.1(6)tictactoe  26.2(6)tictactoe_ai
email: dale@math.hawaii.edu  subject line: 190 c26(12)
Quizzed on the connect_three.f95 program and
cnect_three_ai.f95  get_payoff subroutine.
**CONNECT-THREE**

Connect-three is like Tic-tac-toe. You must get three-in-a-row. For a similar game see Connect-four:

https://www.mathsisfun.com/games/connect4.html

At each move the player chooses a column and an X or O is deposited at the lowest unoccupied place in the column.

**HOMEWORK 26.1(5) connect_four.f95**  Write a program which allows two players to play Connect-three.

!connect_three.f95  subject line: 190 h26.1(5)

program connect_three
integer :: move
character(1) :: b(5,3)="-"
character(1),parameter :: undecided=""
5 format(3(1x,a1))
print*,"Enter 0 to quit."
print 5,(b(i,:),i=1,5)
do
  print*,"Enter column #, 1, 2, 3 for ",x_0(b)
read*, move
if (move == 0) __________
If (move < 1 .or. move > 3) __________
j = move
i = 5 - count (b(:, j) /= "-") ! i = lowest unoccupied row
if (i == 0) ______ ! column full
b(i, j) = X_O(b)! X or O depends on turn
print 5, (b(i, :), i = 1, 5);
if (winner(b) /= undecided) then
    print*, "Winner is ", winner(b); exit
endif
enddo
contains
character(6) function winner(b)
character(1): : b(5, 3), v(20, 3)
winner = undecided
do i = 1, 5; v(i, :) = b(i, :); enddo
do j = 1, 3
    v(5 + j, :) = b(1:3, j); v(8 + j, :) = b(2:4, j);
v(11 + j, :) = __________
enddo

do  i=1,3
    v(14+i,:)=(/b(i,1),b(i+1,2),b(i+2,3)/)
    v(17+i,:)=(/b(i,3),b(i+1,2),b(i+2,1)/)
endo

do  i=1,20
    if(all(v(i,:)=="X"))then; winner="X";return;endif
    if(all(__________ ))then; ________;return;endif
endo

if(all(b/="-"))then; winner="nobody";return;endif
endfunction

character(1) function X_O(b)
character(1)::b(5,3)
If(mod(count(b/="-"),2)==0)then; X_O="X"
else; X_O=__________
endif
endfunction
endfunction
endprogram

Do only one of 26.2A(6), 26.2B(8), or 26.2C(11).
Homework 26.2a(6) connect_three_ai_a.f95  Write a program which plays an optimal Connect_three game for player O against a human player X.

!connect_three_ai_a.f95  subject line:  190 h26.2a(6)
program connect_three_ai
integer::move, payoff, payoff2
integer, parameter::inf=10**6
character(1)::b(5,3)="-", a(5,3)
character, parameter::undecided=""
5  format(3(1x,a1))
print*,"Enter 0 to quit."
print 5,(b(k,:),k=1,5)
do

 !Human move
print*;print*,"Enter column #, 1, 2, 3 for ",X_0(b)
read*,move
if(move==0) ____________
if(move<1 .or. move>3) ____________
j=move
i=5-count(b(:,j)/="-")
if(i==0) _______ !column full
b(i,j)=X_0(b)  !=X or O depends on turn
print 5,(b(k,:),k=1,5)
if(winner(b)/=undecided) then
  print*,"Winner is ", winner(b); exit
!Computer move
Print*,"Computer play"
call get_payoff(b,payoff,inxt,jnxt)
b(inxt,jnxt)=X_O(b)
print 5,(b(k,:),k=1,5)
if(winner(b)/=undecided) then
  print*,"Winner is ", winner(b); exit
endif
do
contains
recursive subroutine get_payoff(b,payoff,inxt,jnxt)
!payoff = best payoff for X, inxt, jnxt = best position
character(1)::b(5,3),a(5,3)
integer::i,j,payoff,payoff2,inxt,jnxt,inxt2,jnxt2
if(winner(b)/=undecided)then;
  select case(winner(b))
    case("X"); payoff=1
    case("O"); payoff=-1
    case("nobody");payoff=_____
  endselect
  return;
endif
select case(X_O(b))
  case("X");payoff=-inf
  case("O");payoff=inf
endselect
do  j=1,3
   i=5-count(b(:,j)/="-")
   if(i==0) ______ !column full, cycle
      a=b;
      a(i,j)=X_0(b)
   call get_payoff(a,payoff2,inxt2,jnxt2)
enddo

endif
else
   if(payoff<payoff2)then
      payoff=payoff2; inxt=i; jnxt=j
      !alpha prune position, may exit if payoff=1
   endif
endif

endsubroutine

character(6) function winner(b)
character(1):: b(5,3), v(20,3)
winner=undecided
do i=1,5; v(i,:)=b(i,:); enddo
do j=1,3
   v(5+j,:)=b(1:3,j); v(8+j,:)=b(2:4,j);
v(11+j,:) = __________
enddo

do i = 1, 3
  v(14+i,:) = (/b(i,1), b(i+1,2), b(i+2,3)/)
  v(17+i,:) = (/b(i,3), b(i+1,2), b(i+2,1)/)
endo

do i = 1, 20
  if(all(v(i,:))=="X") then; winner = "X"; return; endif
  if(all(_________)) then; _________; return; endif
endo

if(all(b/="-")) then; winner = "nobody"; return; endif
endfunction

character(1) function X_O(b)
character(1): : b(5,3)
If(mod(count(b/="-"),2)==0) then; X_O = "X"
else; X_O = __________
endif
endfunction
endprogram

Do only one of 26.2A(6), 26.2B(8), or 26.2C(11).

**Homework 26.2B(8)** connect_three_ai_b.f95 Write a program which plays an optimal Connect-three game for player X against a human player O. X always starts first.

!connect_three_ai_b.f95 subject line: 190 h26.2b(8)
Do only one of \(26.2_A(6),\) \(26.2_B(8),\) or \(26.2_C(11)\).  

**Homework 26.2C(11)** connect_three_ai_c.f95  
Same as  
26.2A(6) - write a program which plays an optimal game for player \(O\) against a human player \(X\) but with Alpha-Beta pruning enabled. At alpha stages, exit if payoff=1. At beta stages, exit if payoff=-1.

!connect_three_ai_c.f95  subject line: 190 26.2C(11)