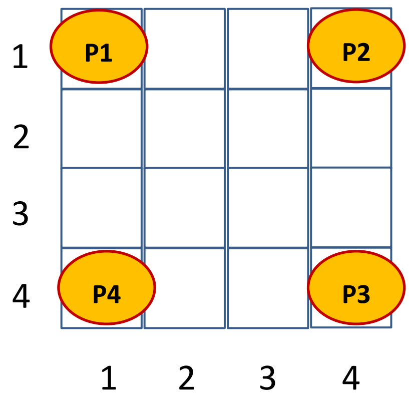
The University of Texas at Dallas
CS 6364
Artificial Intelligence
Fall 2020
Instructor: Dr. Sanda Harabagiu
Grader/ Teaching Assistant: Maxwell Weinzierl

Homework 2: 100 points (20 points extra-credit)

Issued September 22, 2020
Due October 7, 2020 before midnight
Submit only in eLearning

Name: Shankaranarayanan Kallidaikuruchi Ramakrishnan (sxk190109)

PROBLEM 1: Multi-player Games (**60 points**)
Four players are playing a game which is illustrated in the following Figure:



Player 1 is initially positioned in square [1,1], Player 2 is positioned in square [1,4], Player 3 is positioned in square [4,4] and Player 4 is positioned in square [1,4]. The player that will reach first the diagonally opposite position will win the game. Player 1 will start the game, followed by Player 2, Player 3 and Player 4.
All Players can move either up, down, left or right, if the square in that direction is available and not occupied by any other player.
For example, Player 1 initially can move only to the right or down.
Question 1: How do you represent each node of the game? (**3 points**) How is the initial node of the game represented? (**1 point**)

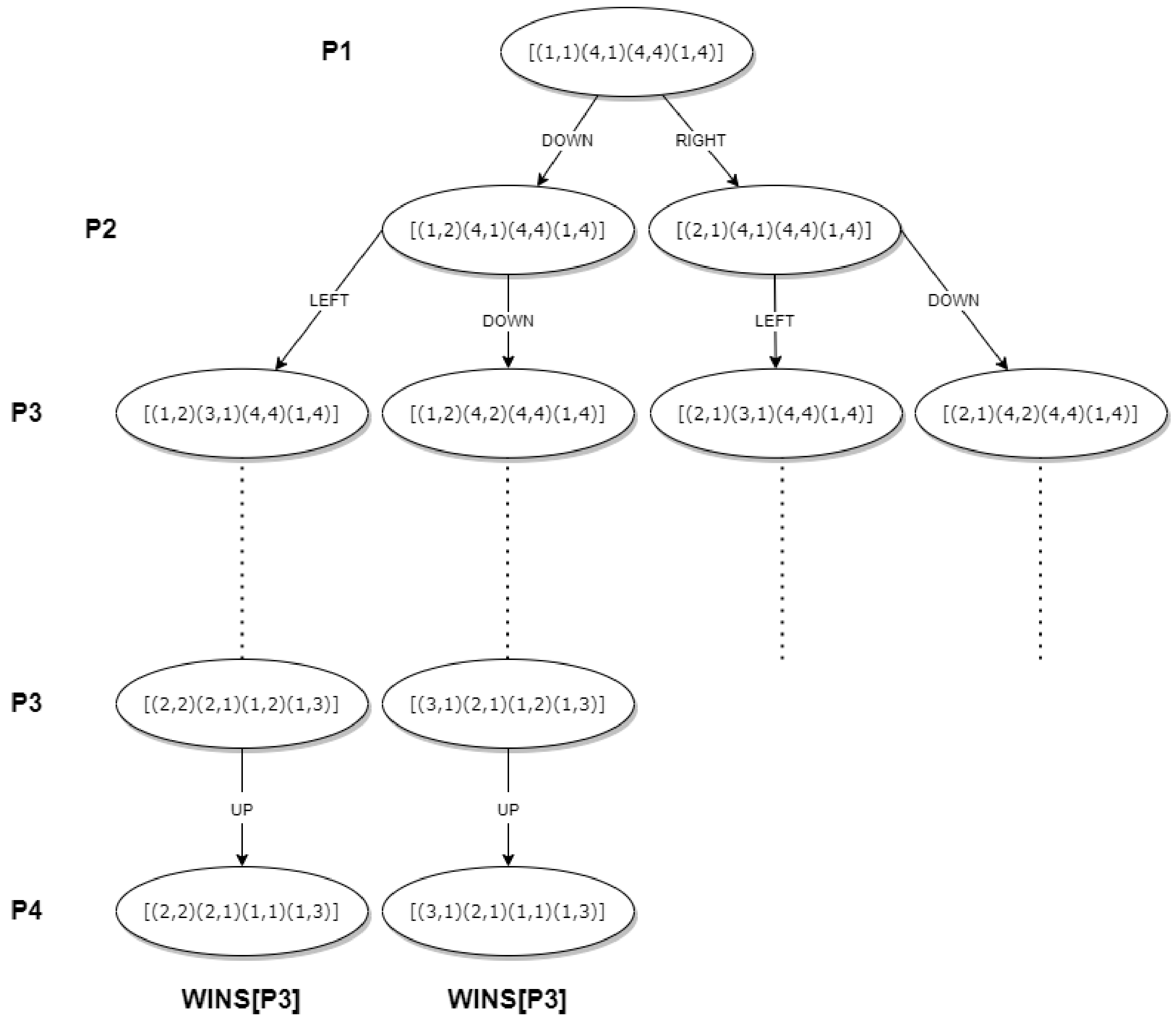
Solution:
Each node of the game can be represented as the coordinates (xi,yi) of position of players_i on the game board. Since we have 4 players we have = [(x1,y1),(x2,y2),(x3,y3),(x4,y4)]

Initial Node: [(1,1), (4,1), (4,4), (1,4)]

Programming Assignment for Problem 1:
A. Write code that generates the game tree for this multi-player game. (**15 points**) The output should use the following format:
[*Current player* =???| Father node (if not initial node) =???| Action= ????| Current game node =???| if the game node is repeated, write REPEATED; if the current game node corresponds to a winning situation for one of the players, write WINS[PLAYER???]]
Hint: Successors of repeated game nodes should not be considered!

Question 2: Draw the game tree based on the output of your code. (**11 points**) Note: You can draw the game tree on a piece of paper, take a picture and attach it to your answers, which will be returned in a PDF file.

Solution:



Question 3: If Player 1 wins, the utility should be 200. If Player 2 wins, the utility should be 300. If Player 3 wins, the utility should be 400 and if Player 4 wins, the Utility should be 500. In any case, because this is not a zero-sum game, the other players also receive utility values:

- When Player 1 wins, Player 2 receives utility=10, Player 3 receives utility=30, and Player 4 receives utility=10;
- When Player 2 wins, Player 1 receives utility=100, Player 3 receives utility=150, and Player 4 receives utility=200;
- When Player 3 wins, Player 1 receives utility=150, Player 2 receives utility=200, and Player 4 receives utility=300;
- When Player 4 wins, Player 1 receives utility=220, Player 2 receives utility=330, and Player 3 receives utility=440;

If the minimax values of the repeated states shall be ignored, compute the MINIMAX values in all other states of the game, using the following program assignment.

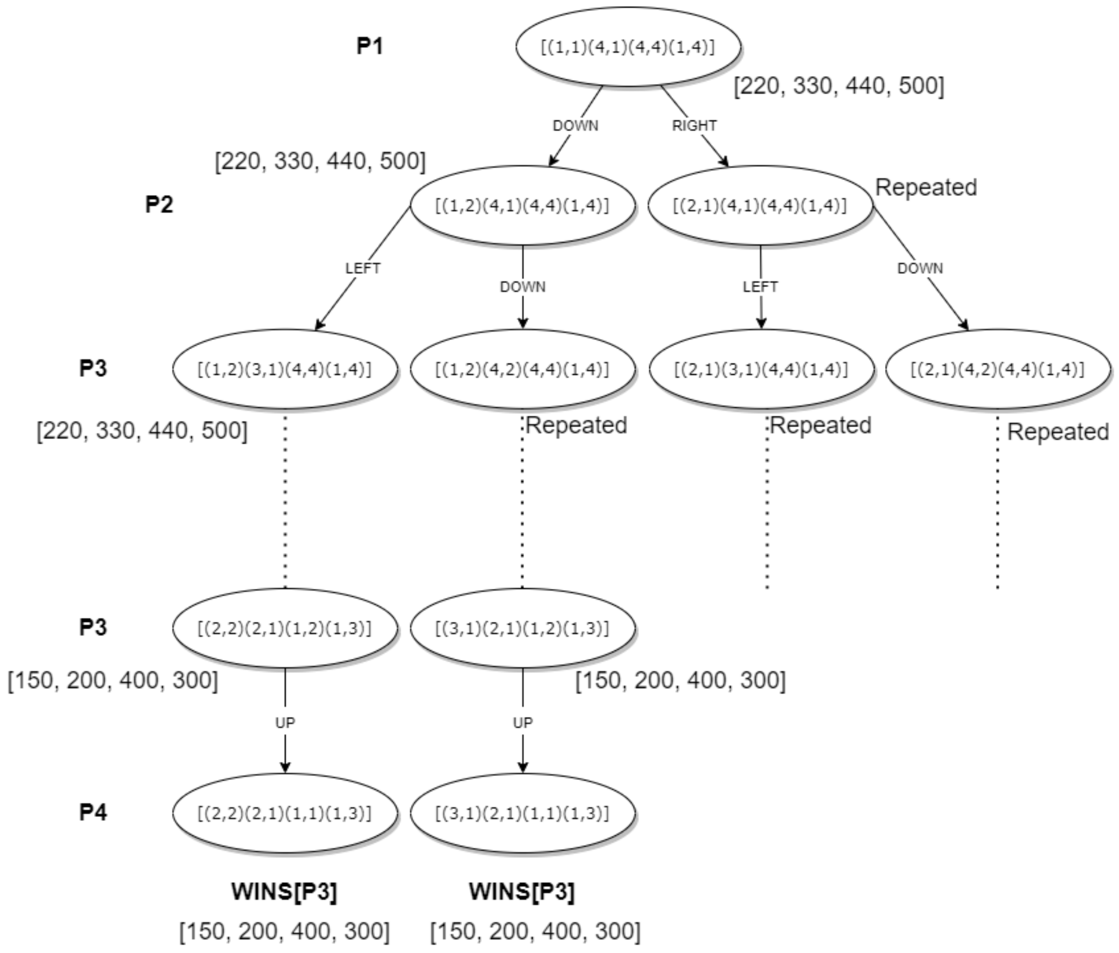
Programming Assignment for Problem 1: (**15 points**)

B. Write code that computes the MINIMAX values for all non-repeated game nodes and display the values of MINIMAX in the following format:

[*Current player* = …| Father node (if not initial node) = …| Action= ????| Current game node =…| if the current game node corresponds to a winning situation for one of the players, write WINS[PLAYER???]| MINIMAX = ?????]

Question 4: Place the MINIMAX values for each non-repeated game state in the drawing of the game tree based on the output of your code. (**15 points**)

Solution:



Extra-credit:

Programming Assignment for Problem 1:

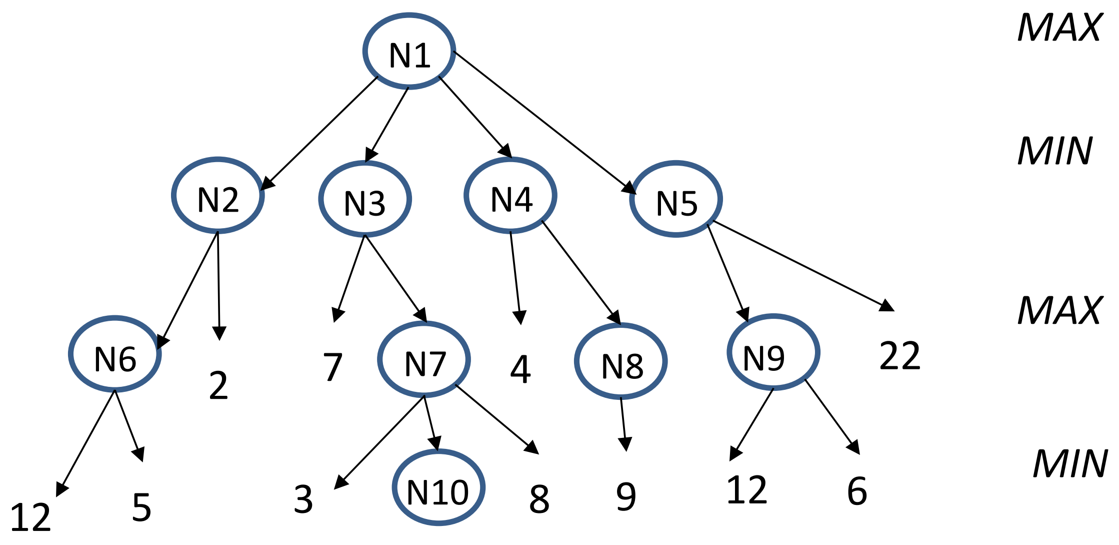
C. If you allow an additional action for the game, namely that any player can jump over an occupied square, and represent this new action as : Jump+Down, Jump+Left, Jump+Right or Jump+Down, modify your program to generate the new game tree. Produce the output in the same format as when doing assignment A. (**10 points**) Comment on the difference between the game trees: Which player won faster in which variant of the game??? Are the repeated states any different? (**10 points**)

Solution:

When "Jump" action was not considered, total number of unique states in the game tree is 19068. However when with "Jump" actions, the total number of unique game states in the game tree increases to 35342.
In the first variant of the game (without jumps), Player 3 won faster and number of repeated states is 24929. In the second variant of the game (with jumps), Player 2 won faster and number of repeated states is 51000.

PROBLEM 2: Alpha-Beta Search (**26 points**)

Find the move ordering (killer move) that allows you to prune the largest number of subtrees when using alpha-beta pruning on the following game tree:



You will receive **10 points** if you find the killer move and explain why it is a killer move. You should produce a trace of alpha-beta pruning using the format showing how the killer move operates (**10 points)**:

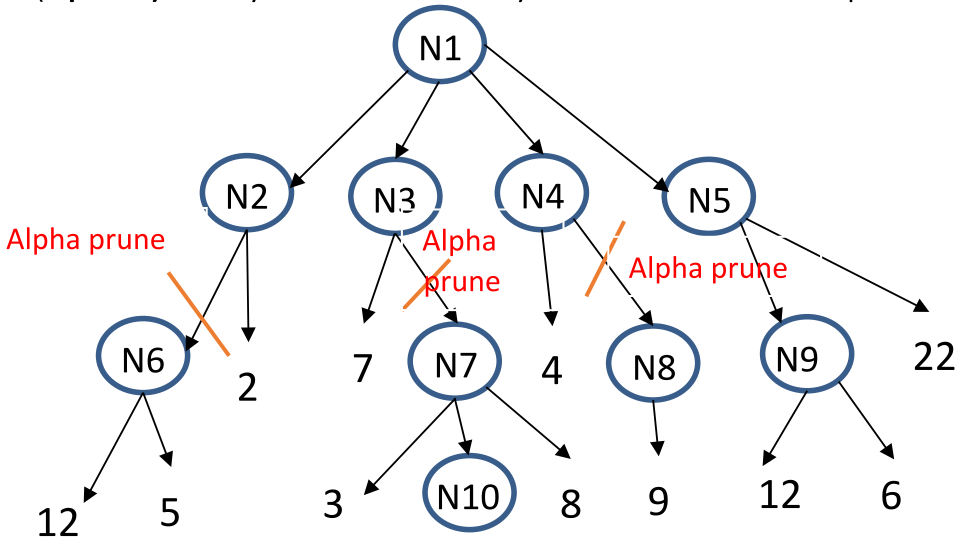
Solution:

The killer move is moving from **N1 to N5.**
When we take action to move from N1 to N5, we evaluate the node N5 as Min function. Looking at the successors of N5, we obtain N9 as max function. The max value at N9 becomes 12 and Min value at N5 also becomes 12. Therefore, we know that utility value at N1 should be greater than 12 as it is a max function.

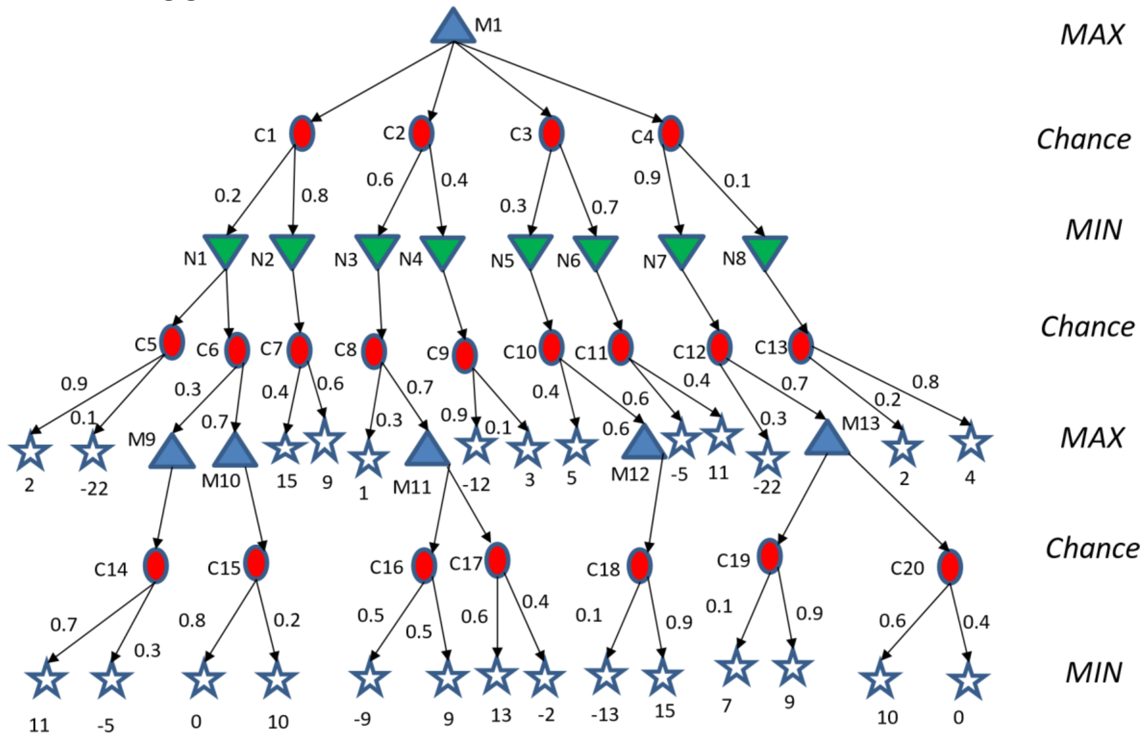
When we take action to move from N1 to N4, we evaluate the node N4 as min function. N4 has a terminal node with value of 4 as its successor. Since alpha = 12, we can prune the subtree based on the alpha value.
Similarly, when we take action to move from N1 to N3, we evaluate the node N3 as min function. N3 has terminal node with value of 7 as its successor. Since alpha = 12, we can prune the subtree based on the alpha value.
Finally, when we take action to move from N1 to N2, we evaluate the node N2 as min function. N2 has terminal node with value of 2 as its successor. Since alpha = 12, we can prune the subtree based on the alpha value.

N1:	alpha = -∞	beta = +∞	max_value = -∞
N5:	alpha = -∞	beta = +∞	min_value = +∞
N9:	alpha = -∞	beta = +∞	max_value = -∞
12:	return = 12		
N9:	alpha = 12	beta = +∞	max_value = 12
6:	return = 6		
N9:	alpha = 12	beta = +∞	max_value = 12
N5:	alpha = -∞	beta = 12	min_value = 12
22:	return = 22		
N5:	alpha = -∞	beta = 12	min_value = 12
N1:	alpha = 12	beta = +∞	max_value = 12
N4:	alpha = 12	beta = +∞	min_value = +∞
4:	return = 4		
N4 alpha cut			
N1:	alpha = 12	beta = +∞	max_value = 12
N3:	alpha = 12	beta = +∞	min_value = +∞
7:	return = 7		
N3 alpha cut			
N1:	alpha = 12	beta = +∞	max_value = 12
N2:	alpha = 12	beta = +∞	min_value = +∞
N6:	alpha = 12	beta = +∞	max_value = -∞
12:	return = 12		
N6:	alpha = 12	beta = +∞	max_value = 12
5:	return = 5		
N6:	alpha = 12	beta = +∞	max_value = 12
N2 alpha cut			
N1:	alpha = 12	beta = +∞	max_value = 12

Indicate on the Search Tree which subtrees shall be pruned and if it is alpha or beta pruning (**6 points)** when you indicate correctly which subtrees shall be pruned.



<u>PROBLEM 3:</u> Chance Games (**14 points**) Given the following game tree:



Compute the *Expectiminimax* value in the following nodes, making sure you show how you computed the value:
(**1 point**) In M1: **9.04**
(**1 point**) In M9: **6.2**
(**1 point**) In M10: **2**
(**1 point**) In M11: **7**
(**1 point**) In M12: **12.2**
(**1 point**) In M13: **8.8**
(**1 point**) In N1: **-0.4**
(**1 point**) In N2: **11.4**
(**1 point**) In N3: **5.2**
(**1 point)** In N4: **-10.5**
(**1 point**) In N5: **9.32**
(**1 point**) In N6: **1.4**
(**1 point**) In N7: **-0.44**
(**1 point**) In N8: **3.6**

Solution:

To compute M1 we have to start from bottom leaf nodes and work our way to the root.

In the last level, we have CHANCE nodes

C14:	0.7(11)+0.3(-5) = 7.7-1.5 = **6.2**
C15:	0.8(0)+0.2(10) = **2**
C16:	0.5(-9)+0.5(9) = **0**
C17:	0.6(13)+0.4(-2) = 7.8-0.8 = **7**
C18:	0.1(-13)+0.9(15)= **12.2**
C19:	0.1(7)+0.9(9)= **8.8**
C20:	0.6(10)+0.4(0)= **6**

Moving one level above, we have MAX nodes

M9:	max(C14) = **6.2**
M10:	max(C15) =**2**
M11:	max(C16,C17)= **7**
M12:	max(C18) =**12.2**
M13:	max(C19,C20)= **8.8**

Moving one level above, we have CHANCE nodes

C5:	0.9(2)+0.1(-22)= 1.8-2.2 =**-0.4**
C6:	0.3(M9)+0.7(M10)= 1.86+1.4 = **3.26**

C7: 0.4(15)+0.6(9) =6+5.4=**11.4**
C8: 0.3(1)+0.7(M11)=0.3+4.9=**5.2**
C9: 0.9(-12)+0.1(3) =**-10.5**
C10: 0.4(5)+0.6(M12)=**9.32**
C11: 0.6(-5)+0.4(11)=**1.4**
C12: 0.3(-22)+0.7(M13)=**-0.44**
C13: 0.2(2)+0.8(4)=**3.6**

Moving one level above, we have MIN nodes
N1: min(C5,C6) =min(-0.4,3.26) =**-0.4**
N2: min(C7) =**11.4**
N3: min(C8)=**5.2**
N4: min(C3)=**-10.5**
N5: min(C10)=**9.32**
N6: min(C11)=**1.4**
N7: min(C12)=**-0.44**
N8: min(C13)=**3.6**

Moving one level above, we have CHANCE nodes
C1: 0.2(N1)+0.8(N2)=-0.08+9.12=**9.04**
C2: 0.6(N3)+0.4(N4)=0.6(5.2)+0.4(-10.5)=**-1.08**
C3: 0.3(N5)+0.7(N6) = 2.796+0.98 =**3.776**
C4: 0.9(N7)+0.1(N8) = 0.9(-0.44)+0.1(3.6) = -0.396+0.36 =**-0.036**

Moving one level above, we have MAX node
M1: max(C1,C2,C3,C4) =max(9.04,-1.08,3.776,-0.036) =**9.04**

Software Engineering (includes documentation for your programming assignments)

Your README file must include the following:

- Your name and email address.
- *Homework number* for this class (AI CS6364), and the *number of the problem* it solves.
- A description of every file for your solution, the programming language used, supporting files from the aima code used, etc.
- How your code operates, in detail.
- A description of special features (or limitations) of your code.

Within Code Documentation:

- Methods/functions/procedures should be documented in a meaningful way. This can mean expressive function/variable names as well as explicit documentation.
- informative method/procedure/function/variable names.
- Efficient implementation
- Don't hardcode variable values, etc