CS 152 - Lab 001

TA: Kage Weiss
Office Hours: T 3:30-5:00 Cochiti [SUB], or by appointment.
Contact: mmweiss@unm.edu
Website: **http://cs.unm.edu/~kageweiss/TA/cs152.html -- SLIDES POSTED**

- Sign in sheet located at whiteboard
- Today we are working on Connect Four (Lab 6)

Connect Four

There are two major parts to Connect Four, each with its own two parts:

1. Win detection
    1. Check the whole board for a win
    2. Check in a line for a win
2. Minimax AI
    1. maxScoreForComputer (simulated computer turn)
    2. minScoreForHuman (simulated human turn)

Win Detection

The hard part of win detection is already broken down for you: knowing the goal and the steps to achieving it.

- To find a win on the board, we need to check each piece, in each direction.
  - This is made simpler by the other portion:
- To find a win in a direction, we just need to count the matching pieces in a line from our starting piece.
  - Make sure the pieces match, and we're not attempting to look off the board for pieces that don't exist.

Win Detection

Finding a win in a direction (findLocalWinner) takes two *sets* of information, the original (col, row) location on the board, and the (Δcol, Δrow) direction pair. From our starting point (c, r) we need to count steps in our direction to look for matching pieces. This can be done with a for(i < number for win) loop, or a while(true) loop, as long as we have an index to "step" with.
You would do this with the following format:

```
Loop(){
        if(location exists in the board){
                if(our piece doesn't match){
                        return no win;
                }
        }
}
return win;
```

Win Detection

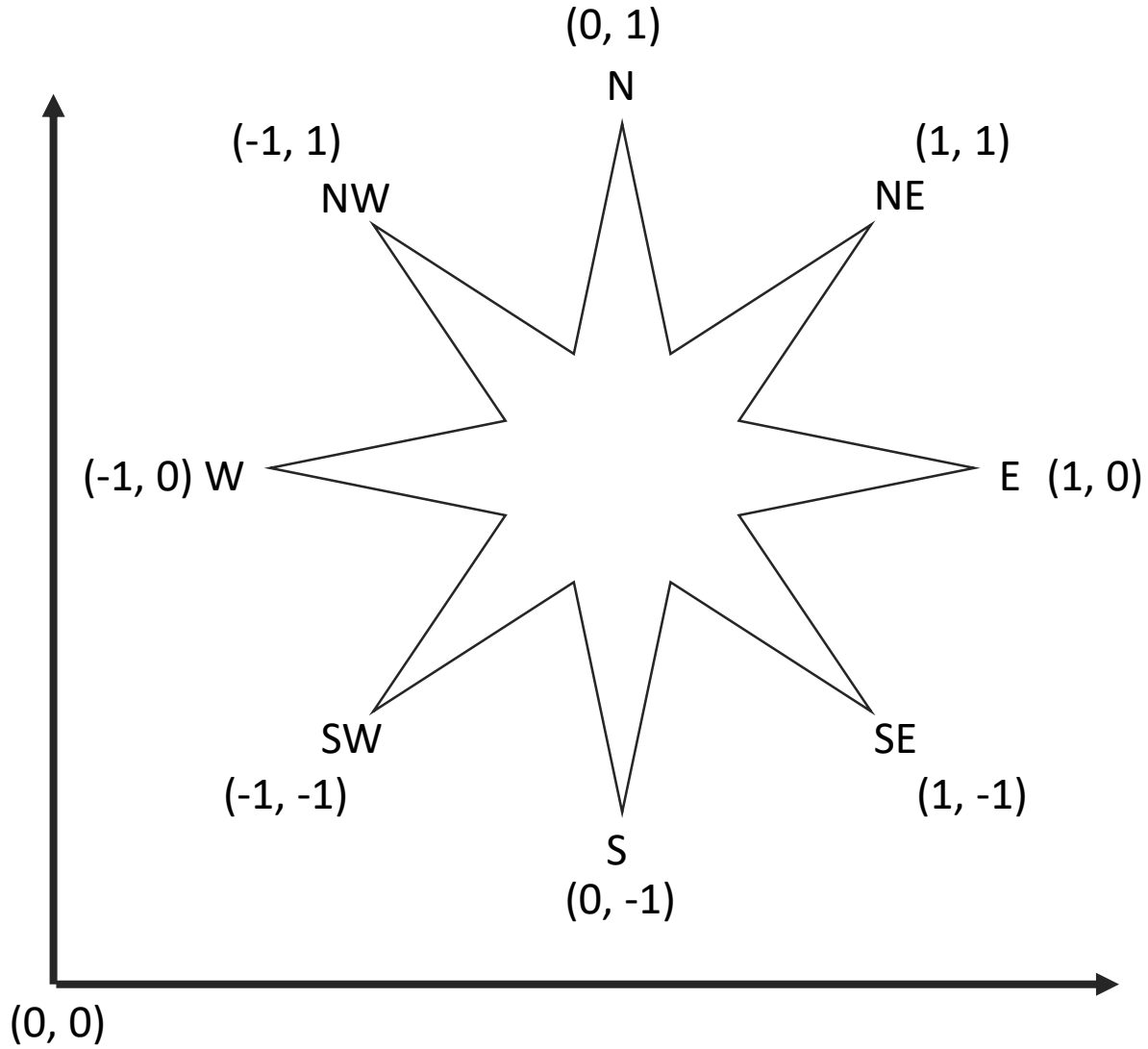
The way we step in a direction with an offset is rather simple,
We take our original location: (col, row)
We decide how many steps we're taking: i
We decide how long our steps are: (Δcol, Δrow)
And then we add the distance to our position: (col + i*Δcol, row + i*Δrow)

So, i steps from (col, row) in (Δcol, Δrow) direction means we will be at the location ((col + i*Δcol), (row + i*Δrow))

It's up to you whether you want to check if you took enough steps, or if you found enough pieces, just make sure you're not exiting the board, and you're checking in the proper 8 directions (or 4 depending on how you code it).

Win Detection

Remember, our board looks like a graph, so (0, 0) is the bottom left corner.



Minimax AI

Minimax allows the computer to “think” via 2-step recursion. The computer simulates possible plays by creating a tree of its possible moves and simulating the human player doing the same.
This is accomplished with the two methods:

- maxScoreForComputer (simulated computer turn)
- minScoreForHuman (simulated human turn)

Each method is basically the same, with the exception of the piece type dropped and the outcome if a win is found, the computer is trying to **max**imize its chances, so it returns high if the simulated computer can win, vice versa if the simulated human succeeds.
These methods call each other (recursion) until they reach their predetermined max depth, or find a win.

Minimax AI

The following is a demonstration of how minimax works. While technically an algorithm, I would use the term simulation to describe minimax, as instead of working out probabilities, it simple emulates how the game would play out.

There are three parts to the AI, the main caller “bestMoveForComputer”, and the algorithm pair, min and max. BestMove is tasked with finding the best column for the computer to play in, and this is done by simulating the game for each possible play, and comparing the results. Minimax returns one result for each call, but each call is a tree of possibilities:

Minimax AI

bestMoveForComputer

We want to maximize the computer score, so we'll call max for each column.

[Column 0] [Column 1] [Column 2]

Minimax AI

bestMoveForComputer
[Column 0] [Column 1] [Column 2]

Max (maxDepth: 4 , currentDepth: 0)
Play in column 0
Computer is not winning
Human is not winning
Board is not full
currentDepth < maxDepth
So check if the human could win
[Column 0] [Column 1] [Column 2]
Undo play

Minimax AI

bestMoveForComputer
[Column 0] [Column 1] [Column 2]

Max (maxDepth: 4 , currentDepth: 0)
Play in column 0
Computer is not winning
Human is not winning
Board is not full
currentDepth < maxDepth
So check if the human could win
[Column 0] [Column 1] [Column 2]
Undo play

Min (maxDepth: 4 , currentDepth: 1)
Play in column 0
Human is not winning
Computer is not winning
Board is not full
currentDepth < maxDepth
So check if the computer could win
[Column 0] [Column 1] [Column 2]
Undo play

Minimax AI

bestMoveForComputer
[Column 0] [Column 1] [Column 2]

Max (maxDepth: 4 , currentDepth: 0)

Max (maxDepth: 4 , currentDepth: 2)
Play in column 0
Computer is not winning
Human is not winning
Board is not full
currentDepth < maxDepth
So check if the human could win
[Column 0] [Column 1] [Column 2]
Undo play

Min (maxDepth: 4 , currentDepth: 1)
Play in column 0
Human is not winning
Computer is not winning
Board is not full
currentDepth < maxDepth
So check if the computer could win
[Column 0] [Column 1] [Column 2]
Undo play

Minimax AI

bestMoveForComputer
[Column 0] [Column 1] [Column 2]

Max (maxDepth: 4 , currentDepth: 0)

Max (maxDepth: 4 , currentDepth: 2)
Play in column 0
Computer is not winning
Human is not winning
Board is not full
currentDepth < maxDepth
So check if the human could win
[Column 0] [Column 1] [Column 2]
Undo play

Min (maxDepth: 4 , currentDepth: 1)

Min (maxDepth: 4 , currentDepth: 3)
Play in column 0
Human is not winning
Computer is not winning
Board is not full
currentDepth < maxDepth
So check if the computer could win
[Column 0] [Column 1] [Column 2]
Undo play

Minimax AI

bestMoveForComputer
[Column 0] [Column 1] [Column 2]

Max (maxDepth: 4 , currentDepth: 0)

Max (maxDepth: 4 , currentDepth: 2)

Max (maxDepth: 4 , currentDepth: 4)
Play in column 0
Computer is not winning
Human is not winning
Board is not full
currentDepth = maxDepth
So this play/branch is a draw
Undo Play
Return 0;

Min (maxDepth: 4 , currentDepth: 1)

Min (maxDepth: 4 , currentDepth: 3)
Play in column 0
Human is not winning
Computer is not winning
Board is not full
currentDepth < maxDepth
So check if the computer could win
[Column 0] [Column 1] [Column 2]
Undo play

Minimax AI

bestMoveForComputer
[Column 0] [Column 1] [Column 2]

Max (maxDepth: 4 , currentDepth: 0)
Undo play

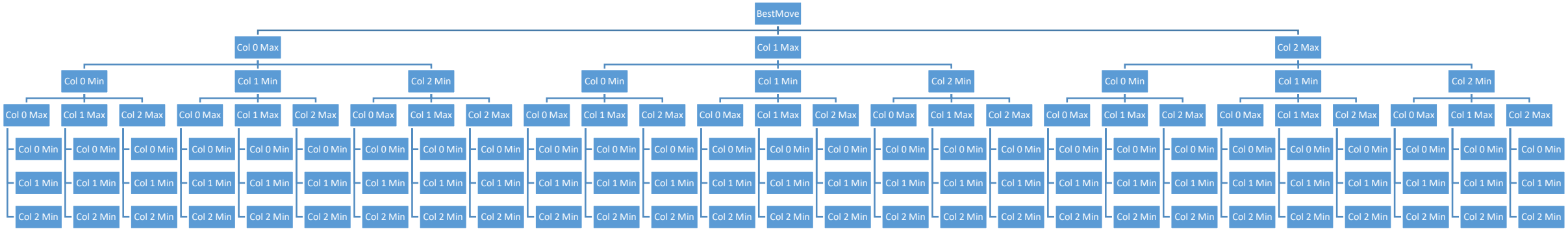
Max (maxDepth: 4 , currentDepth: 2)
Undo play

Min (maxDepth: 4 , currentDepth: 1)
Undo play

Min (maxDepth: 4 , currentDepth: 3)
Undo play

Minimax AI

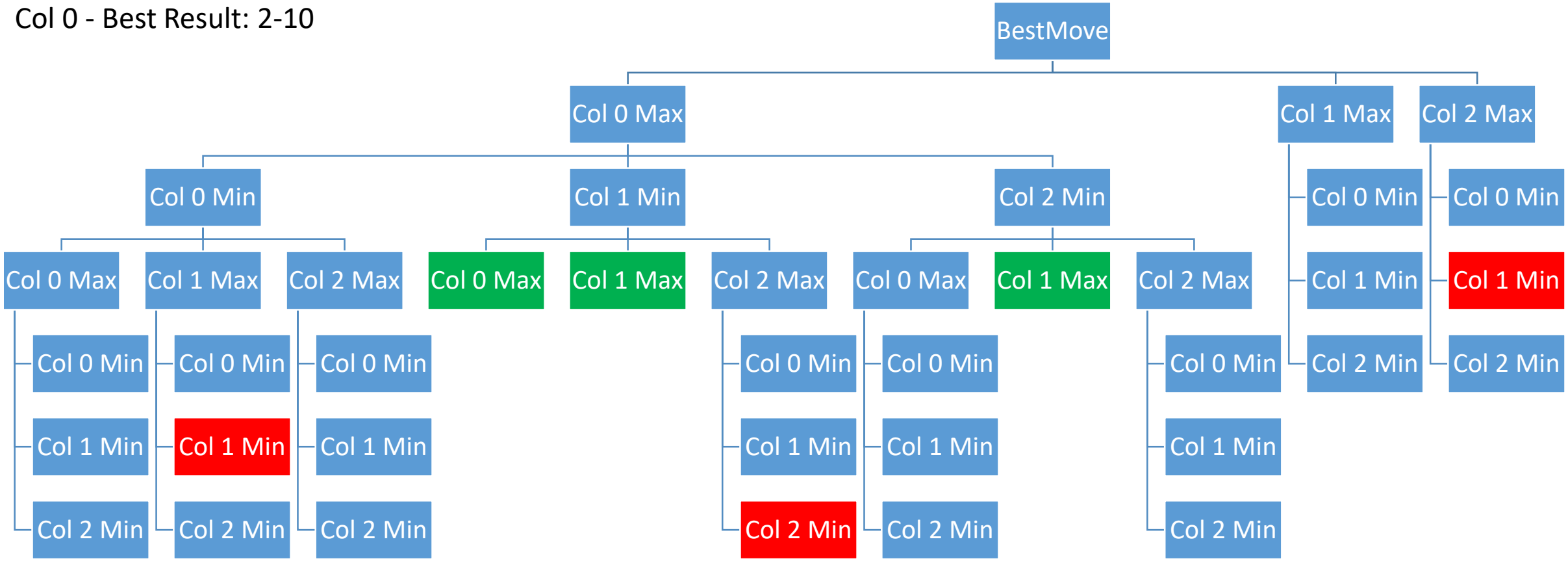
Max Depth: 4



Minimax AI

Max Depth: 4

Col 0 - Best Result: 2-10



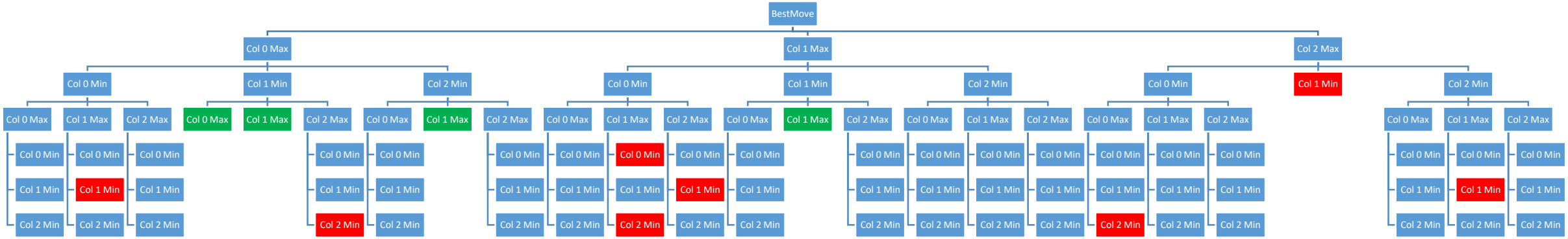
Col 0:0 - Best Result: 2-0

Col 0:1 - Best Result: 1-10

Col 0:2 - Best Result: 1-10

Minimax AI

Max Depth: 4



Col 0:0:0 - Best Result: 2-0
Col 0:0:1 - Best Result: 2-0
Col 0:0:2 - Best Result: 2-0

Col 0:1:0 - Best Result: 10
Col 0:1:1 - Best Result: 10
Col 0:1:2 - Best Result: 1-0

Col 0:2:0 - Best Result: 2-0
Col 0:2:1 - Best Result: 10
Col 0:2:2 - Best Result: 2-0

Col 0:0 - Best Result: 0
Col 0:1 - Best Result: 10
Col 0:2 - Best Result: 10

Col 1:0:0 - Best Result: 2-0
Col 1:0:1 - Best Result: 1-0
Col 1:0:2 - Best Result: 2-0

Col 1:1:0 - Best Result: 2-0
Col 1:1:1 - Best Result: 10
Col 1:1:2 - Best Result: 2-0

Col 1:2:0 - Best Result: 2-0
Col 1:2:1 - Best Result: 2-0
Col 1:2:2 - Best Result: 2-0

Col 1:0 - Best Result: 0
Col 1:1 - Best Result: 10
Col 1:2 - Best Result: 0

Col 2:0:0 - Best Result: 1-0
Col 2:0:1 - Best Result: 2-0
Col 2:0:2 - Best Result: 2-0

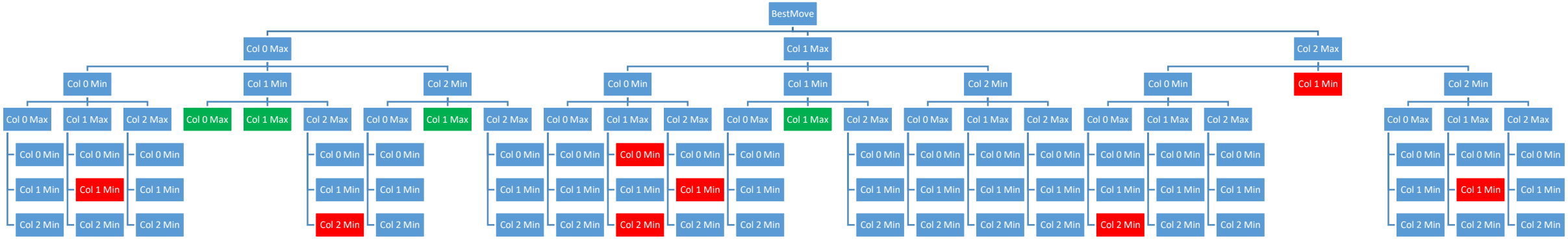
Col 2:1 - Best Result: -10

Col 2:2:0 - Best Result: 2-0
Col 2:2:1 - Best Result: 2-0
Col 2:2:2 - Best Result: 2-0

Col 2:0 - Best Result: 0
Col 2:1 - Best Result: -10
Col 2:2 - Best Result: 0

Minimax AI

Max Depth: 4



Col 0:0 - Best Result: 0
Col 0:1 - Best Result: 10
Col 0:2 - Best Result: 10

Minimax:
Col 0 - Best Result: 10

Col 1:0 - Best Result: 0
Col 1:1 - Best Result: 10
Col 1:2 - Best Result: 0

Minimax:
Col 1 - Best Result: 10

BestMove:
Best Result: Col 1 (Or column 0, depending on how you write it.)

Col 2:0 - Best Result: 0
Col 2:1 - Best Result: -10
Col 2:2 - Best Result: 0

Minimax:
Col 2 - Best Result: -10