An MS Words, PDF, or plain text file that contains:

1) Instructions on how to compile and run your program, and
Open a terminal and cd into the project source file.
Then run the following commands:
javac GameMain.java
java GameMain
Then, play the game.
The status information will be shown on the terminal.

2) A high-level description of your program design.
We adopt Object Oriented Design, and separate the functions of the game into the following different modules.
GameMain.java is the main and starting class, which contains the entry point - main method.
In AIPlayerMinimaxCutoff.java, we implemented the required heuristic eval function.
In AIPlayerBetterHuristic.java, we implemented a better heuristic eval function
AbstractAIPlayer.java AIPlayerBetterHuristic.java AIPlayerMinimaxCutoff.java Board.java Cell.java GameMain.java GameState.java Seed.java

Project: 4X 4 Tic-Tac-Toe Game

Set a fixed depth limit so that CUTOFF-TEST(state, depth) returns true for all depth greater than some fixed depth d. (It must also return true for all terminal states, just as TERMINAL-TEST did.) The depth d is chosen so that a move is selected within the allocated time.

Project: clarification on Eval function
In the Project description, X3 is the same as X3(s) and O3 is the same as O3(s) in the equation for the evaluation function Eval(s).

Project: a couple of details
A couple of details for the project:

* Let the computer (or the max player) have the X symbol and the human player have the O symbol.

* When the game starts, let the human player choose whether to go first or second.

AI Project

I simplified the project a little to make it easier to implement -- now you do not have to determine which player has the most number of 3 symbols that line up when the 4 x 4 grid is filled up (as I described in Sat 3/25 lecture.) It will simply be a draw when the grid is filled up. The evaluation function is also easier to compute now. For the diagonals, you only have to consider the middle diagonals with 4 squares.

Project:
4*4 tic tac game

Changes in Alpha-Beta Algorithm:

Min_utility, max_utility should be changed to min_eval, max_eval to start

Insert cutoff search line

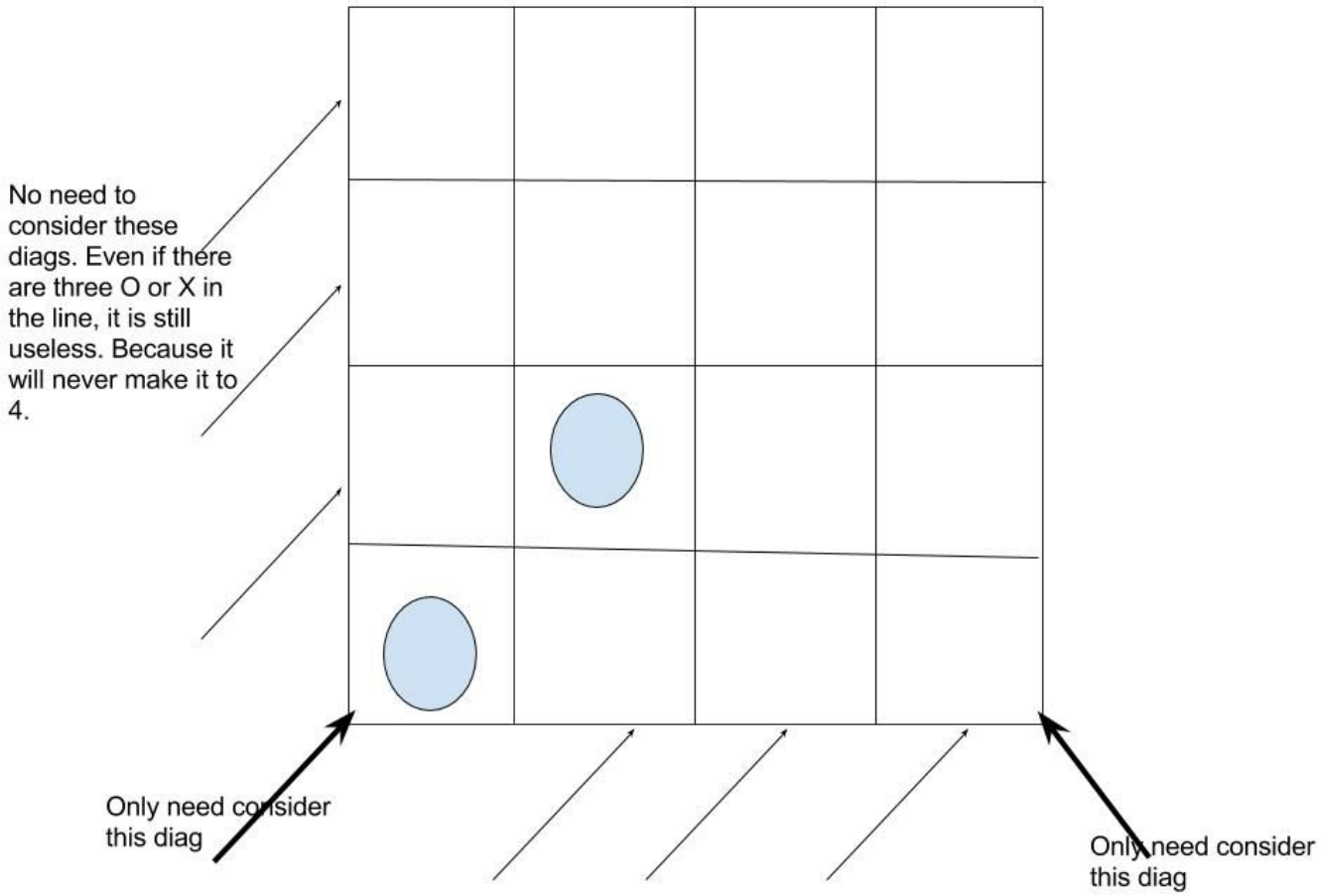
Terminal_test changed to Eval_test

There are three types of terminal states: X win (return min_eval), O win (return max_eval), draw

Simplification: (March 30)
Only when there are four O or X in a line can be taken as winning. Thus, only need to consider two diag
So only consider four rows, four columns, two diags.

Changes in the alg (the same in two places):
If CutOff(s) then Eval(s)

Simple: **If reach level 10, then return. Or if reach 10 s computation by a normal PC, then return**.


No need to consider these diags. Even if there are three O or X in the line, it is still useless. Because it will never make it to 4.
Only need consider this diag
Only need consider this diag