# CISC 440 - Artificial Intelligence & Robotics
**Spring 2020**

## Assignment 02: Due Tuesday, 2/25 by 11:59pm

For this assignment, you will finish the agents that use iterative deepening search and A* search to solve 8 puzzle. Begin by forking and cloning the provided assignment as described in the "Assignment Setup and Submission" document in Canvas. You will need to finish the implementations of the IterativeDeepeningSearchPlayer and AStarSearchPlayer classes.

## EightPuzzle

EightPuzzle PEAS task environment description:

**Performance measure**
- The number of empty cell moves until the puzzle is solved, which for us has the empty cell in the bottom right corner

**Environment**
- 3x3 (or 2x2) grid of cells, containing values 1-8 (or 1-3) and one empty cell

**Actuators**
- Move the empty cell up, down, left, or right one cell

**Sensors**
- Knowledge of entire board

## IterativeDeepeningSearchPlayer (eightpuzzle.player)

**getAction() : Action**

The IterativeDeepeningSearchPlayer class uses (you guessed it) iterative deepening search to find a solution to the 8 puzzle. This method operates on a copy of the board (created using inherited behavior). If the puzzle has yet to be solved:

- Reset the puzzle to its initial state (by calling the resetToInitialState method)
- Increase the depth limit and attempt to solve the puzzle

My solution calls a private recursive "solvePuzzle" method with parameters storing the current depth and the depth limit. This recursive method:

- Determines the next move of the empty cell* and stores it by calling "addPlannedAction"
- Moves the empty cell
- Checks to see if the puzzle can be solved within the current depth limit after the move
- Moves the empty cell back if the puzzle cannot be solved within the current depth limit

If the puzzle is solved, this method will return the next planned action in the player's list.

To state it differently, the first time this method is called, it will solve a copy of the puzzle, keep track of the necessary moves while doing so, and return the first move in the list. Subsequent calls to this method will return the subsequent moves in the list.

## AStarSearchPlayer (eightpuzzle.player)

**getAction() : Action**

The AStarSearchPlayer class implements uses iterative deepening search to find a solution to the 8 puzzle with the following heuristic:

- *h* = the sum of the distances of the tiles from their goal positions

The behavior of this method will be exactly like that in IterativeDeepeningSearchPlayer except for the following:

- When choosing the direction to move the empty cell, select the neighbor that results in a board with the smallest *h* value.

## Odds and Ends

- When choosing the direction to move the empty cell, the iterative deepening search algorithm should select neighbors (if they exist) in the following order: above, right, below, left.

- You cannot make any changes besides:
    - the getAction methods in IterativeDeepeningSearchPlayer and AStarSearchPlayer
    - any helper methods within those classes that you wish to add

Your IterativeDeepeningSearchPlayer.java and AStarSearchPlayer.java files will be used along with the other provided files when grading. Your grade be determined largely by the number of tests passed in IterativeDeepeningSearchPlayerTest.java and AStarSearchPlayerTest.java. While these tests are a good indicator of correctness, they do not guarantee a fully correct implementation. In other words, passing all tests may not translate to a 100% on the assignment.

Submit your assignment by pushing your code to your gitlab repository fork by the deadline.