Intelligent Systems

Assignment 3: Adversary Search and CSPs

The assignment involves to construct a game, a game-playing agent, and an agent that can solve a specific constraint satisfaction problems. First, you have to implement the Python code to implement a game and construct a heuristic evaluation function to help your computer player decide its moves while playing the game by using adversarial search. Your function will participate in a tournament against the computer players of the other teams in the class. For the CSP case, you should formulate and implement your formulation as Python code to solve a specified CSP.

PROBLEM 1: Games and Adversarial Search

Game: Blob

Instructions:

Once formed as a team, **request the professor a team number**. The assignment consists of designing and implementing the game and an evaluation function for the game that you can find and play at http://www.mathsisfun.com/games/blob-game.html.

The Blob Game is a two-player game. The object of the game is to be the last player with a "Blob" remaining on the board. At the start of the game, the computer randomly place 12 red and 12 green Blobs on a 6x6 slots board and choose which player goes first (the red player or the green player). In each turn a player moves all his/her Blobs horizontally or vertically one position in one of four directions (left, right, up or down). A Blob dies if (1) an opponent's Blob lands on it, or (2) it falls off the edge of the board as a result of its movement. The board starts at 6x6 but will shrink if there are no Blobs on an outer row or column of the board.

Among our goals we can cite the following:
- Prove to what extent we can provide intelligence to a computer system that implements an strategic board game in which two opponents compete.
- Learn, at the same time, that we enjoy this experimentation.

The main questions that must be answered during the development of this activity are:
- What are the main features of the game that can make a player of any type win or lose?
- How easy is it to implement those features in computer systems?

- Are they worth considering according to the computational cost of their implementation?
- Is it possible to include strategies within an evaluation function? How?
- By what means is it possible to adjust the weights assigned to the features in the evaluation function?

Part 1: Game Programming

To program the Blob game you should create in the Python language a subclass of the Game class whose definition is in the file **games.py** of the online code provided as a resource of the textbook AIMA.

Part 2: Design and Program a Heuristic Evaluation Function

You should design an evaluation function for Blob and apply it to several predefined test cases presented below. This function should come from a thorough analysis of the game, and a first response to the questions set out above. In the final document, you MUST include concise answers to the questions asked and give the detailed description of the evaluation function and its manual application (not of its program) to the below proposed situations (assume that MAX is the red player).

SITUATION 1 (green turn)	SITUATION 2 (green turn)	SITUATION 3 (red turn)
		

You must also implement the functions in Python so that they can be used by the alphabeta_search function to decide the actions of your player during the tournament.
To avoid name matches between the name of your functions and the name of your opponent's functions, you must add your team number as a suffix to your evaluation function and all its auxiliary functions. The code you deliver must run smoothly.

Tasks that do not meet the above specifications will be penalized and will be excluded from the tournament, losing the corresponding grade. The source file should include the name and registration number of the team members as Python comments.

Part 3: Tournament

A round-robin game tournament will be held, confronting your evaluation functions and modifying the level of exploration in the alpha-beta MINIMAX algorithm. The team that obtains the highest score will win the tournament, which will be considered for the evaluation of the tournament. The score per game will be administered as a football match, that is, three points for the winner, one point for each team in case of a tie, and zero points in case of losing the game. Each match will be double-checked so that both teams have the opportunity to start the game.
It would be wise to investigate the game, to experience it in your own right, and to analyze your function's behavior during the game.

Evaluation criteria (50% of the assignment)

The weights assigned to the activities for the evaluation of this problem are:
- Game programming: 50%
- Design and programming of the evaluation function: 30%
- Tournament: 20%

Regarding the design of the evaluation function, the grade will be computed on the following criteria:
- Answer to the questions of the first part (20%)
- Quality of the proposed function, taking into account the validity of the criteria used in it and its ease of implementation (40%).
- Detailed results of applying the proposed evaluation function to test boards (30%).
- Quality of the delivered document (10%).

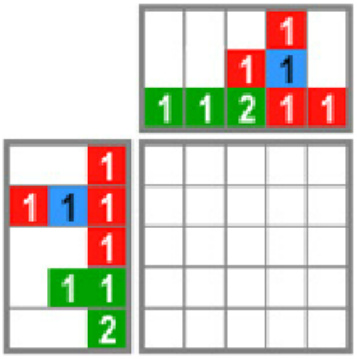
The implementation of the code will be graded according to the following rules:
- If the program does not run but gives a "readable" code and makes sense, the note in this part is 50.
- If the program runs but does not work according to the specifications of the evaluation function, and a "readable" code is given and makes sense, the grade is 75.
- If the program runs and works according to the specifications of the evaluation function, the note is 100.

The points of the tournament will be assigned according to the proportion obtained between the points obtained by your evaluation function, as a result of the games, and the score obtained by the best team of the tournament.

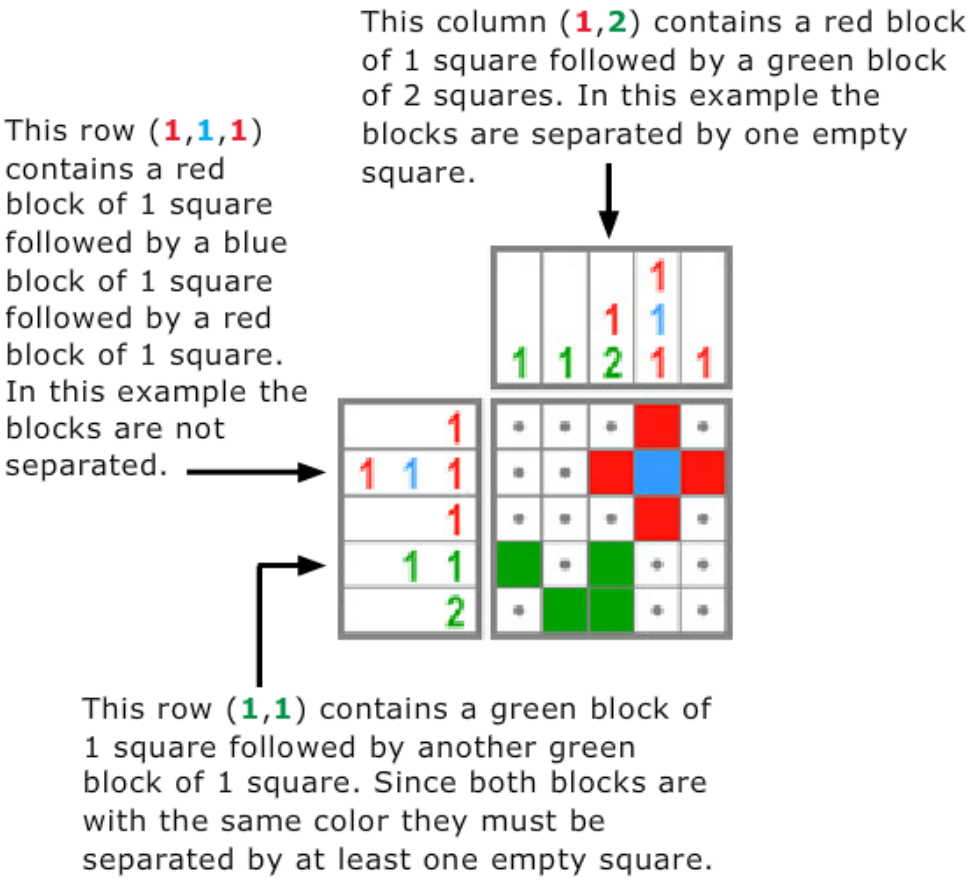
PROBLEM 2: Constraint Satisfaction Problem (CSP)

Puzzle: Pic-a-Pix

Each puzzle consists of a blank grid with colored clues on the left of every row and on the top of every column.



The object is to reveal a hidden picture by painting blocks in each row and column so their length, color and sequence corresponds to the clues, and there is at least one empty square between adjacent same-color blocks. It is allowed to have no empty square between adjacent different-color blocks.



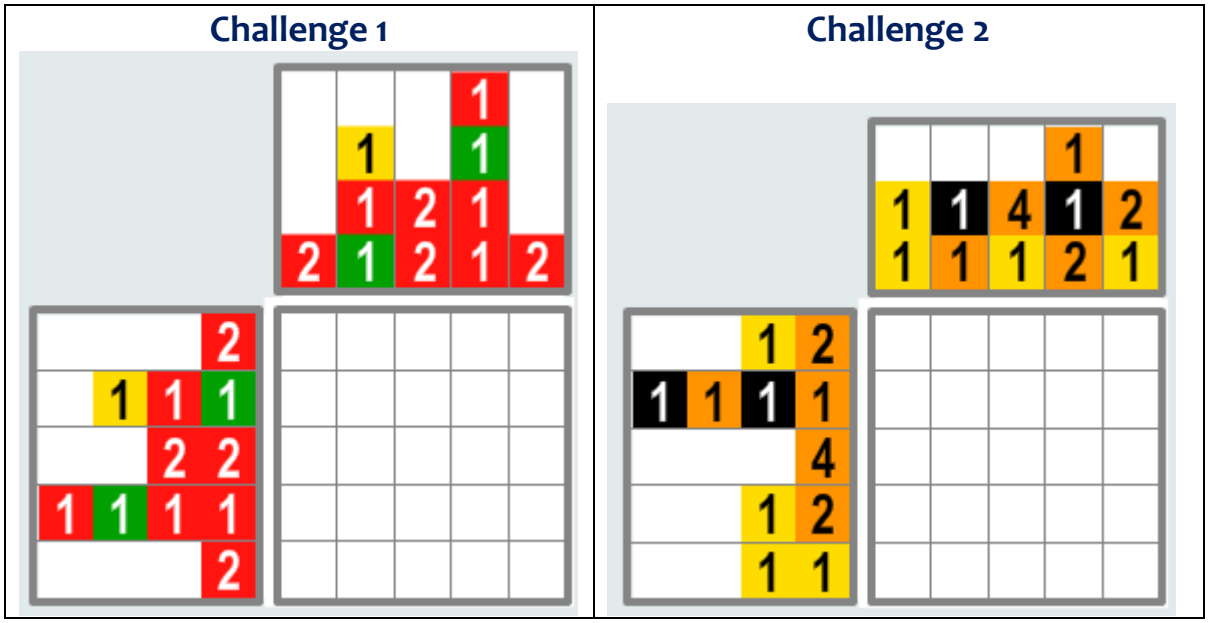
You can play it on the page http://mypuzzle.org/pic-a-pix.

ASSIGNMENT:

Part 1: Problem Documentation as CSP

a) Describe how these types of problems can be formulated as a CSP, that is, describe the variables, their domains, and the constraints they must met.
b) Illustrate this formulation for the 2 challenges illustrated below.

CHALLENGES:



Part 2: Scheduling the solution

To program Pic-a-Pix, you must create a subclass of the CSP class whose definition is in the **csp.py** file of the online code provided as a resource of the AIMA textbook in the Python language.

Evaluation Criteria (50% of activity):

Word or Pdf document that includes (40 points):

- The general formulation as a CSP (60%)
- Specific formulations of challenge problems (40%)

Python source program and its execution (60 points):

- If the program does not run but a "readable" code is given for all required functions (50%).
- If the program runs, but does not solve the problems and gives a code "readable" and that makes sense according to the formulation of the challenges (75%).
- If the program runs, the code matches the formulation of the problems and gets the correct answers for the challenges (100%).