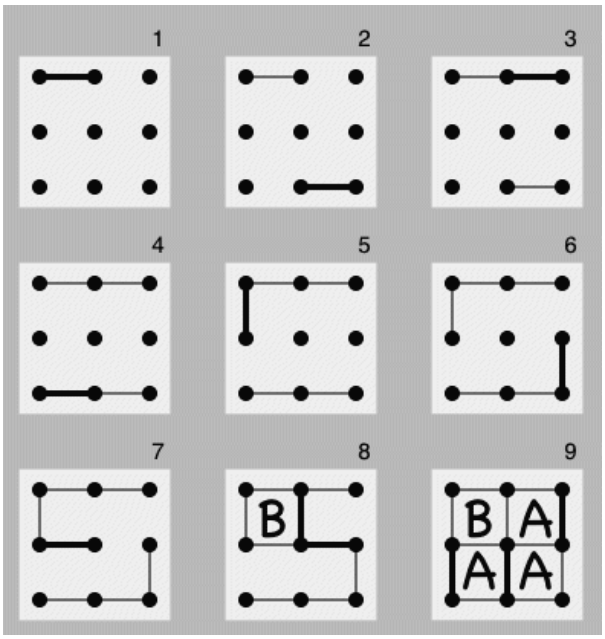
CS165A Programming Assignment 1: Boxes and Grids

In this lab you will write a computer program to play a game, dots and boxes. (http://en.wikipedia.org/wiki/Dots_and_boxes)



Starting with an empty grid of dots, players take turns, adding a single horizontal or vertical line between two unjoined adjacent dots. A player who completes the fourth side of a 1x1 box earns one point whereas a player who completes two grids of size 1x1 simultaneously earns 2 point . The game ends when no more lines can be placed. The winner of the game is the player with the most points.

Problem Statement.

1. You should implement the minimax search algorithm and alpha-beta pruning on a box grid of size 6x6, as shown in Figure 2.

2. Since the search tree can be very deep, you should implement the cut-off test strategy and go only till a certain depth. You need to design a evaluation function for a given configuration. (That's the key for your computer player's strength).

   Try different cut-off depths and set a depth so that each move takes proper time on your computer. Your code will be evaluated both on your final score as well as the time taken for making a move.

Implementing Minimax

Given below are the details that are required for implementing minimax

State of the game :- The entire grid structure is represented as one horizontal matrix ***h_matrix*** and one vertical matrix ***v_matrix.*** Both the matrices are Boolean in nature and are used for the storing the edges of the grids. The edge is set as True if a move has already been made.The horizontal edge between the *ith* row and the *jth* column is stored in the position ***h_matrix[i][j]***. Similarly, the vertical edge between the *ith* row and the *jth* column is stored at the position ***v_matrix[i][j]***.

The state will generally be used for determining the current state of the matrix as well as predicting the future states.

Move :- Each move is a vector with three entries, while the first two entries represent the coordinates or the row and column location , the third entry represents whether the move is horizontal or vertical. 0 is used for denoting the vertical edges while 1 is used for denoting the horizontal edges.

Example

[0,2,0]- The vertical edge between the first row and the third column. (Remember, that things in computer science start from 0.)

[3,4,1]- The horizontal edge between the fourth row and the fifth column.

Scoring:- For completing one block each player will be given one points whereas a simultaneous completion will result in two points.

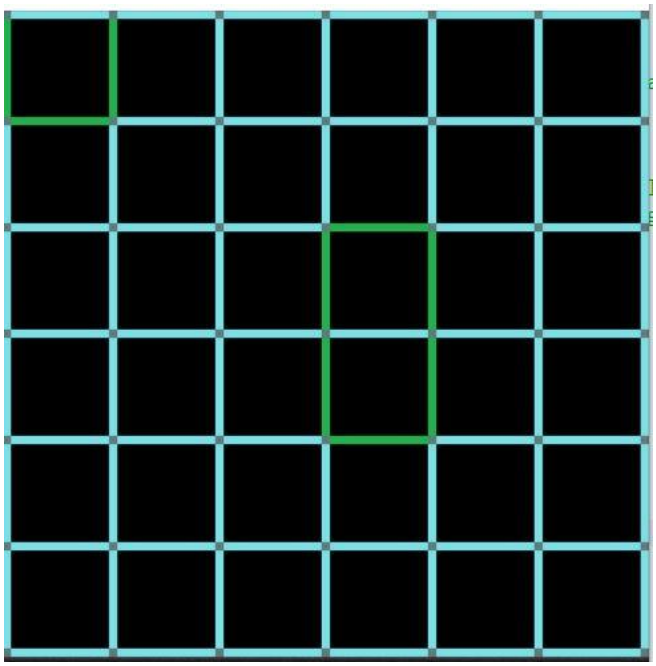


Figure: A sample grid comprising of horizontal and vertical edges. The ones displayed in the color green represent the moves that have been made. Competing the box on the upper left corner gives a score of 1 and completing the box in the middle will gives a score of 2.

Helper functions

Here is a list of functions that can be useful while implementing the minimax algorithm.

current_state():- The function returns the current state of the system in the form of ***h_matrix*** and ***v_matrix.***

game_ends(h_matrix,v_matrix):- Returns True if the game is over and False otherwise.

next_state(move,h_matrix,v_matrix):- Returns the future states and corresponding score given the present state and the move.

increment_score(move,h_matrix,v_matrix):- Given a move and the state of the system, the function returns the increment in the score because of the respective move. The increment can be 0,1, or 2.

make_move(self,move,player_id): The function is used for making a move. Given that there are two players, the id of the player will either be 0 or 1. (You can always set the id to 1 for this assignment.)

***list_possible_moves*(*h_matrix ,y_matrix*) :-** The function lists all the possible moves given the state of the game in terms of horizontal and vertical matrices.

Prerequisites

You will need to have Python 2.7. installed in your system. Apart from python you will be needing the the pygame package installed in your system

Pygame :- Pygame is a python based platform used for developing games. You need to have pygame installed before setting up the algorithm. Here is a link on installing pygame in your system. Here are few links that will help you in installing pygames

https://www.pygame.org/wiki/GettingStarted

File Details

Here are the files that you will be working on

1) **Practice. py :-** The file is to allow you to understand the basic working of the game and try the different functions that you will be using for implementing minimax and alpha beta pruning. Read the comments on top of file and uncomment some of the sample moves given right at the bottom.

2) **file1.GUI.py** :- This is where you will be writing your final code. You need to make changes to the following functions defined in the file. Your algorithm will play against a simple heuristic defined in player 1.

    a) ***minimax(self)*** : You will implement the minimax algorithm in this function. You can change the number of input parameters of the function and the output of the function must be the optimal move made by the function.

    ***Note* :-Please be careful of the depth and set an optimal depth of your minimax algorithm.**

    Hint :- You may have to implement two functions for calculating the minimum and maximum successor of a given node.

    b) **alphabetapruning(self)**:- You will implement the alpha beta pruning algorithm in this function. You can change the number of input parameters of the function and the output of the function must be the optimal move made by the function.

    Please report and print the values of alpha and beta.

    c) **evaluate(self):-** Come up with your own evaluation strategy for minimax . Try to make the best guess of the winner at a reasonable depth given the state of the system.

    d) **player2(self):**-Call the minimax and alpha beta pruning in this function to compete with the other heuristic. The function should return the optimal move.

    Note:- Please don't make any changes to any other functions apart from this function as we will be running these functions from your code onto our own files.

3) **sampleboxesandgrids.py**:- Implementation of the non GUI version of the game. You will use this if there is some problem with installing pygame. You will make changes to the following functions. All the functions mentioned above have also been implemented in this code and you need to make changes to these functions only.

Submission Details

Please submit the entire folder with the files. Your folder should have a doc file with your name and UCSB perm number. Describe your algorithm and state whether you have used file1.GUI.py or sampleboxesandgrids.py

You will be graded on the following parameters.

a) Time taken by your algorithm to make a move.

b) Strategic Efficiency of your algorithm.

Due date is **22 February 2018 (11:59 PST)**

Your program should run on CSIL.

All the best !!!!!!!