Applied Artificial Intelligence
EDA132
2016-02-10
Johanna Petersson (adi10jpe@student.lu.se)

Assignment 1 – Search

By: Johanna Petersson (adi10jpe@student.lu.se)

Getting the program

The code and an executable .jar file can be found at https://github.com/JohannaMoose/eda132 in the folder Assignment 1 – Search. To download the code and program onto one's computer go to the URL above and click the button "Download ZIP" that can be found just above the file list. This will download a zip file to the computer with all the files; go to where the file downloaded and into the folder Assignment 1 – Search to see the files pertaining to this assignment.

The file Othello.jar is runnable from the terminal on Linux/Mac computers with the command `java -jar Othello.jar` (tested on Mac) if the terminal is pointed to the folder. This will start the program, and instructions on how the game works will follow in the terminal window.

Of cause, the code in the folder should be buildable as well and should run the program without a hiccup, provided that Java SDK8 is installed on the computer. Nothing else is tested.

The program

The program for playing Othello is implemented with the help of four different classes and one enum. For a "real" program outside of a school or learning setting this would not be a good design and would need to be split into smaller parts for easier maintainability. In other words, this is dirty and easy solution to programming the task.

The game is played in the terminal as a text based game.

The game itself is implemented using an integer two-dimensional vector with a number representing white and one representing black. Throughout the program black and white is represented as a number. All this is done in the `OthelloGame` class

OthelloGame

The `OthelloGame` class keeps track of the board, and can determine if a move is legal in the game, and make that move, flipping all the appropriate disks on the board. It also is responsible for calculating scores, what color won the game and whose turn it is, based on the color. What it does not know is who is playing the color in the actual game. Inside the `OthelloGame` class there is another class, `GameMove`, which is a helper class to make it easy to transfer coordinates for a move in the game.

The most important and remarkable part of the code in OthelloGame is the `findOneVariableDirection` method. It is responsible for seeing if there is at least one direction that will flip disks if a move is made. It searches out from a set of

coordinates in all directions, on at a time, until one viable flip is found or all directions are searched and no flips can be found. If this method doesn't work as expected, nothing in the program will work as expected and the search will fail miserably. It is also a tricky class because its problems can manifest in many parts of the code without any clear connection to the method.

Program

The Program class is responsible for showing the game in the terminal window and running the game, who it is that should make a move and pass the moves on to the `OthelloClass`. This is the central class for the game, but without much interesting logic, with one exception. It is responsible for how it is handled when the computer can't find a move to make and make sure that there is a legal move for the human player. This is all done in the method `gameContinues(String, OthelloGame)`.

Algorithm

The program's AI is designed and implemented to use a mini-max adversarial search with the alpha-beta pruning. This is all done in a separate class, Algorithm.java that has only the responsibility of running that search. It is a fairly standard implementation of the algorithm, making heavy use of the `getNextMove` method, which is responsible for first checking to see how much time is spent and break it off if all the allotted time is spent. If there is more time, it goes on to get the next possible legal move in the game by incrementing the last move through the game grid until a new move has been found.

This class is all the magic as far as this assignment goes. But it is important to remember that if the `OthelloClass` doesn't work as expected, the search will never work.

Some comments

The program should satisfy the assignment specification, but most likely still contains bugs. As for a few things I have noticed but decided to overlook…

The time the AI spends finding a solution is just a tad longer than what the user specifies because the program doesn't stop at the moment the time is up. It stops going down the tree at that point, but needs to go back up and thus needs a bit more time.

As an implementation detail I have decided to analyze the game after each round to make sure that the user and/or computer have a possible legal move and for continence this analysis has the same amount of time as the user specified for the AI, with the same behavior of taking a bit longer than that.

If the AI is given to little time, say 2 seconds, everything will work fine up until the end part of the game where the time can be a bit to short and in a few instances of test running the computer didn't find a legal move even though there was one. I could have set a lower limit to how little time the AI can get, but this also depends on computer power and ease of testing so no such limit is imposed, but can cause problems. It follows from the comments above that this also goes for analyzing the game.