Edinburgh Napier UNIVERSITY

Checkers Game
Coursework Report

Igors Ahmetovs
40125689@napier.ac.uk
Edinburgh Napier University - Algorithms and Data Structures (SET09117)

Abstract

This report describes the checkers game produced for Algorithms and Data Structures module assessment. It summarises the main functionality of the application and gives justification to the design decisions behind it. It also looks at potential improvements that could be made in the future.

Keywords – Checkers, Draughts, Python, Game, AI, Data Structures

1 Introduction

The students were tasked with implementing a checkers game in a programming language of choice.

The produced game should be able to present the user with a representation of a game board and allow piece movement according to the rules of the game.

This includes "kinging" a piece when it hits the last opposite row, which then allows a give piece to move both backwards and forwards. Players can also capture the opponents pieces, again, according to the rules.

The game is visually represented to the user after each move. User input is always validated to ensure move legality and that the game only processes the data it expects.

There are three game modes: Player vs Player, Player vs Computer and Computer vs Computer. The last two include utilising an algorithm to determine which move should be taken by the computer.

All movement is recorded throughout the game, and can be replayed upon user's request.

When the game includes human players, there is an option to undo and redo the moves.

The users has the freedom to choose the size of the game board.

2 Design

The game board is stored as a multi-dimensional list of a combination of empty squares, the user's and the opponent's pieces. The list consists of several lists, which themselves represent a game board row and consist of a number of game board squares of the particular row.

Every time a move has been made, the list is updated. Because the size is fixed, and the program does not have to loop through it, the update happens in constant time.

Following every move, the state of the game board is saved by adding the list into another list(of lists of lists). This second list is an ordered representation of the game boards throughout the duration of the game. This can be used to achieve replaying functionality, by iterating through each of the list's elements in the given order, which allows the user to see each move step-by-step.

This also means that the stored states of the game boards can be removed from the higher-level list, or added back in again. This allows the user to discard move(s), and alter the game, or repeat the made moves. The movements that the user decides to undo are not discarded straight away, they are rather overwritten when the player takes another approach that is different from the original move.

However, there is also an option to redo a movement that was previously undone.

The legality of the move is decided by a separate function that checks whether the piece can take the action against the current state of the board. This is achieved by checking values of the multi-dimensional list that represent the game board.

This same function is used to check the validity of the computer player's movements. The initial move is randomly generated and fed into the legality checking function, which then indicates whether this procedure needs to be re-attempted or not.

3 Enhancements

As with any application, the functionality and logic can be further extended and improved.

The algorithm behind the AI is fairly straightforward and not always sophisticated enough to defeat a human player. A better approach would be to utilise a minimax rule and score each possible move based on the potential outcome. This would allow to program the AI so that it would proceed with the move that has the highest score, thus maximising the chances of winning the game.

4 Critical Evaluation

As mentioned in the previous paragraph, the part of the game that is played by the computer needs to be improved to be fully functional.

5 Personal Evaluation

This task has pushed the student to think about a program and its element in terms of Data Structures, and the executed procedures as Algorithms.

Another beneficial aspect would be learning Python as a programming language and getting familiar with the Data Structures of the language and their unique traits, e.g. immutability of certain datatypes.