Edinburgh Napier UNIVERSITY

Algorithms and Data Structures

Ryan Hunter
40206280@live.napier.ac.uk
Edinburgh Napier University - Module Title (SET09117)

1 Introduction

The objective of this task was to implement a checkers game in a language of the developers choosing. The game should be able to played by two people, 1 person and a computer or two computers using a simple artificial intelligence algorithm allowing a computer player to make it's own moves. The implemented program should record each game so it can be replayed and so moves can be undone and redone.

2 Design

The checkers game board was designed using Python to be simple and easy for users to operate while using a command line interface. A list of list was used to draw the board and map coordinates to the board as this was the most simple and effective data structure for this purpose.

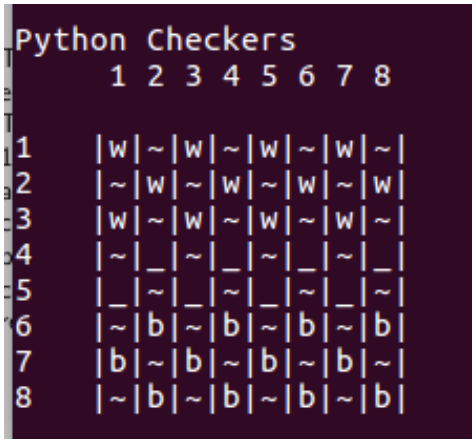

Figure 1: **Checkers Board** - The command line interface checkers board

The game board used a lower case letter to represent either white pawn or black pawn, the letters were capitalized to represent kings, empty cells are shown with an underscore and a tilde was used to show unplayable cells. This gave users a clear indication of what each cell contained and what the meaning of that cell was.

Inputting data was designed around the rules of checkers. Users are asked to input x and y coordinates for both the cell they want to move a piece from and the cell they want to move that piece to. A series of checks were ran on those inputs. All user inputs were first checked for being a number in range of one to nine. The inputs were then checked against the rules of the game. If the coordinates given by the user broke the rules of the game then an exception was thrown and users then had to try again.

Once inputs had passed the checks for the game rules, the movement algorithm is then run. Movement is done by allocating the source and destination cell to the users inputs. The source cell is then changed to empty and the destination cell is changed to show the piece that had moved. Using these user inputs allowed a middle cell to be found. If a middle cell did not contain a piece, it was declared as invalid move but if there was an opponents piece in the middle cell, that cell would be changed to empty as would to source cell and the destination cell would contain the current players piece.

Listing 1: Algorithm for taking pieces

```
if board[mid_y][mid_x] == WP or board[mid_y][mid_x] == ↩
  WQ:
    if board[src_y][src_x] == BP:
        if dst_y == 0:
            White_Pieces = White_Pieces - 1
            board[src_y][src_x] = EC
            board[mid_y][mid_x] = EC
            board[dst_y][dst_x] = BQ
        else:
            board[src_y][src_x] = EC
            board[mid_y][mid_x] = EC
            board[dst_y][dst_x] = BP
            White_Pieces = White_Pieces - 1
    elif board[src_y][src_x] == BQ:
        board[src_y][src_x] = EC
        board[mid_y][mid_x] = EC
        board[dst_y][dst_x] = BQ
        White_Pieces = White_Pieces -1
    print_board(board)
    if White_Pieces == 0:
        print('Black Wins') #win condition
        exit()
```

As of the rules of checkers, if a player takes a piece then if any other pieces can be taken, they must be taken. An algorithm was developed with this in mind. Once a player has taken a piece, the cells diagonally adjacent from the moved piece are checked again for opponents pieces. If an opponents piece is found then the cells adjacent from that are checked for being empty. If these two conditions are met then opponents pieces are automatically taken.

Implementing kings into the game was done by checking the y coordinates of the piece. If a black piece reached the y coordinate of one, then the piece was capitalized and could move diagonally in any direction.

3 Enhancements

The Python Checkers game is a very basic checkers game with not much more functionality than playing the game it's self with another human player. One of the enchantments that could have been made would be implementing an artificially intelligent computer player. Developing and implementing an algorithm that allowed computer verses human play would have greatly increased the functionality of the game. It would be a choice for the user, continuing to allow two human players but greatly increasing the depth of the game.

Another feature the would enhance the ability to undo and redo moves. This could have been done but allowing the user to type undo or redo into the terminal and reversing or redoing their previous move.

Furthermore a replay feature would allow users to watch back their games, increasing the functionality of the game. It would allow users to study their games and see where they made mistakes but also what they done well.

A graphical user interface would have made the game more user friendly. The command line interface requires users to study coordinates on a small board. Implementing a GUI that allows users to drag and drop pieces would appeal to a much larger group of users.

4 Critical Evaluation

Once of the features that worked well in Python Checkers was the ability to take pieces. Taking pieces was always consistent, never showing the wrong information in a cell. The algorithm was well developed and robust.

Another feature that worked well was user input. Inputs were always within range and within the set rules. The program cannot be crashed with wrong user inputs thus is quite stable.

However one of the features that did not work well was the automatic taking of pieces. The developed algorithm would only take one other piece automatically. This could have fixed by using a loop but trying to implement this would crash the program or only take the piece as set by the user.

5 Personal Evaluation

During my time spent on this coursework i have learned everything I know about python. I had previously never written a line of python so that in itself was a challenge. I over came this by doing work with textbooks online and looking a programs that other people had written and trying to understand them. I broke down problems into smaller pieces and wrote some psuedocode to break the problem down further and attempted to translate that into python which worked well. At the start of this year I would not have been confident in developing checkers in any language, so for that reason I am happy with the final output and my performance during this task.