Algorithms and Data Structures

Ahad Malik
40122791@napier.ac.uk
Edinburgh Napier University - Algorithms and Data Structures (SET08122)

1 Introduction

This report provides an insight into the planning and development of a Tic-tac-toe game developed as part of Coursework for the SET08122 Algorithms and Data Structures module. The main objective of this coursework was to allow two players to play a text based version of the game against one another by placing pieces on a board.

2 Design

Initial design of the software began with creating the board itself. For loops were considered to print segments of the grid but I decided to go with simple print statements instead, when the board function is called the command prompt is cleared which makes the screen less cluttered. An array was initialised to represent each cell in the 3x3 grid, this stores all the current board states.

A status function was added, this is used to check the status of the game, whether a winning combo has been achieved or a draw. If statements and equality operators were used to compare the data of the cells and check for winning combos or draws. If a win was achieved the function returns 1, if a draw it returns 0 and -1 if the game has not finished.

User input and input validation was added to the game function. This function checks if the data entered by the user is valid by calling the validEntry function, which returns true if an entry is between 1-10. If entered data is valid the function will compare the entered number with the grid cell numbers (1-9), if the grid cell has not yet been marked it is marked with the current players marker. Markers are determined by the Player number, Player 1 will always have marker X.

After the foundation of the game had been laid down, I started to add further functionality. An undo function was added, this function used the stack data structure to store all choices(moves) made by the player. The Stack data structure allows elements to be added and extracted in a LIFO manner (Last in First out). This proved ideal for storing the moves made by a player as the latest move would be first move available to extract from the Stack.

To implement a replay system for the game I looked at using a linked list but due to not having used them a huge amount I decided implement a 2D Array, this array stores the undo stacks from previous games. When the replay function is called a for loop will go through the undo stack of a game and replay all moves that were made during that game.

A menu was added to the game to allow users to access the two function, this was done using switch statements letting users choose whether to Play a game, Replay a Game or Exit.

3 Enhancements

If time allowed, I would have like to make quite a few enhancements to this game. A proper replay capability would be implemented for a start, one that tracked every user move better rather than just showing the history of the game board, tracking each invalid move and undoes.

I would have also liked to have added the redo feature, this would have worked the same as the undo stack, as anything that was popped off the undo stack would be added to the redo stack. Another useful feature to implement would be to have an AI for the user to play against, the AI would check for possible moves to make in a game and decide.

Another feature that can be added would be to have different sizes and types of boards. This would mean some of the evaluation functions and the data structures they use would have to be changed, larger boards would require the use of 2D arrays.

4 Critical Evaluation

The game manages to meet the basic criteria that it was designed for. The user interface is concise and simple. The board grid is well spaced out and clearly labelled with cell numbers which makes it easier for users identify each cell to make the next move. Menu system is clear and has a default case for invalid inputs.

Input validation helps prevent the game from crashing and gives it structure. Replay functionality is really simple and can be vastly improved upon, as it shows the history of the game, this might not include all the moves in a game that were made(invalid or undone using undo). The undo feature should accompanied by a redo feature.

5 Personal Evaluation

The assignment broadened my understanding on the implementation of a variety of different data structures and how to manipulate them using different kinds of algorithms. This assignment not only required you to apply prior knowledge

gained from labs but also helped me break down a problem into smaller manageable chunks. Breaking down the code into smaller pieces, using various different functions helped with keeping track of whats going on.

Time allocation in an assignment like this should be thought out and followed rigidly otherwise one can lose traction and get sidetracked. A more detailed project plan for the future would help tremendously, one that allocates time to each activity properly. A more rigid project plan can allow one to focus on the more important tasks before others.