2805ICT, Milestone 2: Progress Report

Natnicha Titiphanpong, s2940970

August 26, 2017

Contents

1 Milestone 2

1.1 Requirements

The current version of minesweeper successfully meets the requirements of the traditional minesweeper game however it is still yet to implement the hexagon and colour-based minesweeper modes successfully.

List of completed requirements

Functionality associated with icons — The program is able to meet this requirements by accomplishing the following actions:

- The program shows whether a cell is covered or uncovered by clearly displaying the state of the buttons. The buttons possess a 3D-like image when unclicked, the colour of the buttons change once it is sunken (or clicked) into a darker contrast of the same colour.
- When a cell with no adjacent mines is clicked, all adjacent cells alike are uncovered until it reaches a cell with adjacent mines. This task was done in a while loop, not recursively (as there was no need) as previously stated.
- The state of the board, once the buttons are pressed, are represented by a 'backing grid' which assigns all cells with a number or mine, according to the randomly generated mines and its coordinates.
- The program displays whether a cell has a mine or not by representing it with an image of a mine when pressed and also displaying a message that the game is over.

Displaying icons —

The program uses the tkinter module and its button component. This makes responding to left mouse click for icon display possible. The left click implementation for the flag is yet to be implemented.

Losing a game —

When the user clicks on a button which reveals a mine, a message is displayed in the window "Game Over!" and the state of the game is disabled.

Difficulty —

The user has the ability to select a difficulty: beginner, intermediate or advanced. When a difficulty is selected, it resets the board to that chosen difficulty.

Options —

The user is able to start the game by selecting a difficulty and not a game mode, the program will default to the basic minesweeper game. Selecting the cross at the top right of the window will exit the program and window. The pause button is identified by the universal word "pause" and will disable the game and stop the timer until pressed again and the game is resumed.

Prioritized list of on-going requirements

- The user is able to flag a cell that they believe is a mine.
- The scoring system of the game - the game timer.
- The different game modes

1.2 Product Use Cases

The following list itemizes the user requirements for the implementation of the Ladder-gram program

- The user should be able to interact with the program by specifying the dictionary file name (without the .txt extension).
- The user should be able to interact with the program by specifying the source and target word.
- The user should be able to supply a list of words that cannot be used in the path solution by specifying the file name and placing it in the same directory as the program file.
- The user will have the option of choosing to evaluate the shortest path from source word to target word as the solution.

1.3 Software Architecture

The following list itemizes the software requirements for the implementation of the Ladder-gram program

- The program will notify the user if the inputs are invalid and the program will exit.
- The program will notify the user if a solution was found.
- If a valid solution is found the program will print it to the console along with the length of the solution.
- If the user chooses to input a list of banned words, the program will remove those words from the working dictionary.
- The program will evaluate the shortest solution if the user requests.

1.4 Software Design

High Level Design - Logical Block Diagram

1.5 Testing

No.	Test Case	Expected Results	Actual Results
1.0	**Test User Input**		
1.1	Source word contains number.	Program displays error message: "Source or target input words cannot contain numbers or special characters" and system exits.	As expected.
1.2	Target word input contains special character.	Program displays error message: "Input words cannot contain numbers or special characters" and system exits.	As expected.
1.3	Source word input contains special character.	Program displays error message: "Input words cannot contain numbers or special characters" and system exits.	As expected.
1.4	Target word input contains number.	Program displays error message: "Input words cannot contain numbers or special characters and system exits."	As expected.
1.5	Source and target words are the same.	System prints error: "Start and target words cannot be the same."	As expected.
1.6	Target word input is of different length to source word.	System prints error message: "Start and target words must be the same length."	As expected.
1.7	Check whether init function returns source and target word after input is validated.	Function returns source word as the first element in the list and target as second.	As expected.
2.0	**Test Helper Functions**		
2.1	Test same function and whether it returns the correct number of identical letters.	Test case returns true if same function returns 4 for 'lead' & 'lead', 3 for 'lead' & 'mead' and false if 'hello' & 'there' equals 0.	As expected.

No.	Test Case	Expected Results	Actual Results
2.2	Test build basic function	All words that are adjacent to the start word and not visited in dictionary, are in the list returned by the function. Checks correctness of function with and without keys in visited dictionary.	As expected.
2.3	Test build function.	Builds the list of neighboring elements correctly by returning a list of words that satisfy the regex pattern. The function will also not return words that have been seen or are in the current path.	As expected.
2.4	Test populate dictionary function takes a dictionary name.	Reads in dictionary filename and checks the correct number of words present for each word length.	As expected.
2.5	Test populate dictionary function takes list of banned words.	Reads in banned words list, and asserts that banned words are not present in the populated lists.	As expected.
3.0	**System tests**		
3.1	Test full system using recursive algorithm with mock input and output.	Reads in input and outputs a valid path from start to target word	As expected.
3.2	Test full system using breadth-first search with mock input and output.	Reads in required input and outputs the shortest path from start to target word.	As expected.

Version Control History / Log

User	Activity
ZaymonFC	Finished testfile
ZaymonFC	Merge branch 'master' of github.com/ZaymonFC/2810ICT_Assignment
ZaymonFC	Added diagrams and function description
Natnicha	Added modifications to full system tests.
Natnicha	Added more unit tests.
ZaymonFC	Added system tests and put the main method contents of word ladder into a run function
ZaymonFC	Added build test
ZaymonFC	Merge conflict resolved
ZaymonFC	Finished adding data structures and list of functions.
Natnicha	Edited acceptance test and unit test in document.
Natnicha	Merge branch 'master' of github.com/ZaymonFC/2810ICT_Assignment
Natnicha	Added unit tests to report.
ZaymonFC	Added tests and added the banned words implementation to the word ladder
ZaymonFC	Removed TestCase
ZaymonFC	Added comments for some test cases in testfile.py
ZaymonFC	Added multiple unit tests for different functions in word_ladder
ZaymonFC	Added tests for user input in testfile.py
ZaymonFC	Logged work
ZaymonFC	Fixed BFS and implemented a queue data type
ZaymonFC	Added breadth first search for shortest path
Natnicha	Resolved merge conflict
Natnicha	Modified unit test section.
ZaymonFC	Resolved merge conflict
ZaymonFC	Added test skeleton for input validation
Natnicha	Fixed spelling and name errors

Natnicha	Merge branch 'master' of github.com/ZaymonFC/2810ICT_Assignment
Natnicha	Added requirement acceptance test & user instructions.
ZaymonFC	Added section describing all functions in the program
ZaymonFC	Created UML and logical block diagrams, added to report
ZaymonFC	Merge branch 'master' of github.com/ZaymonFC/2810ICT_Assignment
ZaymonFC	Modified test file and fixed formatting in report
Natnicha	Merge branch 'master' of github.com/ZaymonFC/2810ICT_Assignment
Natnicha	Modified test table
ZaymonFC	Fixed algorithm to find short paths by limiting reverse progress
ZaymonFC	Merge branch 'master' of github.com/ZaymonFC/2810ICT_Assignment
ZaymonFC	Improved performance by reversing sorting direction of fitness function
Natnicha	Added version control section & Unit test tables
Natnicha	Merge branch 'master' of github.com/ZaymonFC/2810ICT_Assignment
Natnicha	Added unittest cases
ZaymonFC	Refactored word_ladder.py to validate user input
Natnicha	Modified listing for user requirements
ZaymonFC	Added unittest file and added test ideas to README.md
ZaymonFC	Added .gitignore file and removed postcompile latex objects
ZaymonFC	Merge branch 'master' of github.com/ZaymonFC/2810ICT_Assignment
ZaymonFC	Modified report preamble to remove redundant packages
Natnicha	Added headings and user requirements
ZaymonFC	Added LaTeX template for assignment documentation
ZaymonFC	Added program files and created project directory structure
ZaymonFC	Initial Commit

Note: Repository is private to prevent plagiarism. | This log was created by using the command

```
git log --pretty=format:'%h;%an;%s' > ./log.csv
```