Functional Testing Plan and Test Case Descriptions

For

Labyrinth of Algorithms

Version 1.3

Prepared by Will Franzen and Lukas Livengood

Case Western Reserve University,
CSDS 393 - Software Engineering

December 9, 2022

Table of Contents

1. Introduction

This document outlines the testing strategy for our application Labyrinth of Algorithms and was made to assist in making production efficiently work towards achieving our goal of a functioning game that is accessible to people interested in learning about algorithms. Included in this document are the explanations of the tests to be run on our application that will indicate whether it is able to satisfy the necessary requirements for showcasing algorithms. The developers and testers of this application are to use this document in evaluating the performance of the application.

2. Scope

The scope of the Labyrinth of Algorithms testing is further detailed throughout this document but a basis shall be described here. The backend of the application, coded primarily in Java, is tested using JUnit 5.0 in eclipse on a windows device. These tests ensure the proper creation of labyrinths and their passing to the frontend, accurate pathfinding from user movements and specified search algorithms, and faultless altering of the labyrinth data structure as per the user defined edits. The frontend of the application will be coded using React and the helper languages required for its implementation. Given the frontend's event-driven design, the testing tools Jest and the React Testing Library will be implemented to ensure the correct visualizations of backend components and proper variable updating per user selections and actions.

3. Test Design Specification

3.1 Specification Format

As the application's classes have heavily varied functionality, each function will be given specific tests. Tests will be deemed as successful if and only if they produce the expected result for a given action. Functions shall have multiple use case tests to ensure they work in a variety of conditions.

3.2 Presumption and Suspension Criteria

Failure to produce expected results for a test shall be deemed as a complete failure for a test and will require developers to attend to any errors in the code that produce unexpected results.

4. Test Specifications

I. Tile Editing

A. Create Obstacle

1. Player will be able to alter an empty tile's state into an impassable obstacle
2. Labyrinth index for the tile will be changed and the visualization will update to that of a WallTile

B. Remove Obstacle

1. Player will be able to alter an obstacle tile's state into an passable empty tile
2. Labyrinth index for the tile will be changed and the visualization will update to that of an EmptyTile

C. Define Initial State

1. Player will be able to select a define a tile as the initial state for the program
2. Labyrinth index for the tile will be changed and the visualization will update to that of a StartTile
3. Only one initial state may be defined at a time

D. Define Goal State

1. Player will be able to select a define a tile as the goal state for the program
2. Labyrinth index for the tile will be changed and the visualization will update to that of a GoalTile
3. Only one goal state may be defined at a time

II. Labyrinth Solving

A. Select Search Algorithm

1. Player will be able to choose from a list of defined search algorithms to visualize traversal through the labyrinth with the possible options as:
    a) Breadth-First
    b) Depth-First
    c) Greedy-Best-First
    d) Dijkstra's Algorithm
    e) A* Search
    f) Random Walk
2. After the desired selection is made, a String is passed to the backend updating the program with which algorithm to use during in its execution

3. During the pathfinding of these algorithms, only empty tiles are to be considered possible moves during its traversal from the initial to goal state tiles

B. Run Algorithm
   1. Player will be able to execute the program and run a visualization of the user selected algorithm
   2. Program will check which algorithm has been selected as described in Section 4.II.A.
   3. Search visualization will begin at the location of the initial state, traveling through empty tiles, completing at the location of the goal state
   4. As the visualization passes through empty tiles, those tiles will be updated with a to-be-determined color or icon to represent the algorithm's traversal through them
   5. The program's execution will terminate once the goal state has been reached during the algorithm's visualization

III. Labyrinth Traversal
   A. Iterative Stepping
      1. Player will be able to click/press a button to advance the algorithm's traversal of the labyrinth by one step. This should end when the algorithm pointer attempts to move into the goal tile.
   B. User Traversal
      1. Player can control and traverse a user pointer through the use of the W, A, S, and D keys corresponding to "Up", "Left", "Down", and "Right"
      2. This pointer begins at the location of the goal state
      3. The pointer will only be able to travel along empty tiles in their traversal from the initial to goal state tiles
      4. Once the goal state tile is reached, the program's execution will terminate as the function's purpose has been met

5. Functional Tests

5.1 Tile Editing Tests

Test	Description	Setup	Expected Result
TET-T1	Set tile to obstacle tile	Change character in Labyrinth text file at index of tile to "w"	Character at appropriate index now corresponds to obstacle tile, visual updates to reflect

			obstacle
TET-T2	Set tile to empty tile	Change character in Labyrinth text file at index of tile to “e”	Character at appropriate index now corresponds to empty tile, visual updates to reflect empty tile
TET-T3	Set to initial state tile	Change character in Labyrinth at index of tile text file to “s”	Character at appropriate index now corresponds to initial state tile, visual updates to reflect start tile
TET-T4	Set to goal state tile	Change character in Labyrinth at index of tile text file to “g”	Character at appropriate index now corresponds to goal state tile, visual updates to reflect goal tile
TET-T5	Try to create second goal or initial tile	Change character in Labyrinth at index of non-initial/goal tile to “s” and “g” in a Labyrinth with pre-defined goal and initial states.	Tile reverts to empty tile, message appears: “Initial/Goal tille already defined”
TET-T6	Try to overwrite goal or initial tile with another tile type	Change character in Labyrinth at index of initial/goal tile to different type.	The change does not take place and nothing occurs.

5.2 Algorithm Tests

Test	Description	Setup	Expected Result
AT-A1	Search Algorithm set to Breadth-First	Input String in algorithm selection dropdown reading “Breadth-First”	User defined algorithm to be used during execution set to “Breadth-First”
AT-A2	Search Algorithm set to Depth-First	Input String in algorithm selection dropdown reading	User defined algorithm to be used during execution set

		“Depth-First”	to “Depth-First”
AT-A3	Search Algorithm set to Greedy Best-First	Input String in algorithm selection dropdown reading “Greedy Best-First”	User defined algorithm to be used during execution set to “Greedy Best-First”
AT-A4	Search Algorithm set to Dijkstra’s Algorithm	Input String in algorithm selection dropdown reading “Dijkstra’s Algorithm”	User defined algorithm to be used during execution set to “Dijkstra’s Algorithm”
AT-A5	Search Algorithm set to A* Search	Input String in algorithm selection dropdown reading “A* Search”	User defined algorithm to be used during execution set to “A* Search”
AT-A6	Search Algorithm set to Random Walk	Input String in algorithm selection dropdown reading “Random Walk”	User defined algorithm to be used during execution set to “Random Walk”
AT-A7	Invalid Algorithm Type Selected	Input String reading anything other than the defined above	Message reading “Invalid Search Algorithm Type” is displayed to user
AT-R1	Program executed with existing solution	Run button corresponding to execution of search algorithm in a solvable Labyrinth	Path produced by the search algorithm is visualized to the user in the Labyrinth
AT-R2	Program executed with no existing solution	Run button corresponding to execution of search algorithm in an unsolvable Labyrinth	Visualization is still produced until all possible tiles have been explorer; Messages reads “No possible solution” and is displayed to user

5.3 Player Movement Tests

Test	Description	Setup	Expected Result
PMT-1	Moving the player in all valid directions	W, A, S, and D keys will be pressed in series when the	The player will move in the direction of the pressed key

		player is in a space where doing so will not collide with a obstacle	
PMT-2	Moving the player towards an obstacle	The player will be moved by the W,A,S, and D keys when surrounded by obstacles in every direction	The player will remain in their position
PMT-3	Attempting to move via non-WASD keys	Non-WASD keys will be pressed with the player not surrounded by collidable objects	The player will remain in their position

6. Defect Resolution

6.1 Issue Tracking

As the project is hosted on Github, its built-in issue tracking shall be utilized as a tool for project members to track problems in the repository.

6.2 Issue Resolution

Both developers have an obligation to work towards resolving issues present in the repository. Any revisions shall be accompanied with a brief explanation of changes made and any known remaining issues.

7. Applicable Documents

1. *Software Requirements Specification For Labyrinth of Algorithms (Version 1.2)*, September 20, 2022.
2. *Software Design Document For Labyrinth of Algorithms (Version 1.2)*, October 4, 2022.

Appendix: Revision History

Date	Version	Author	Review Outcome
10/26/22	1.0	WF and LL	*Initial Draft*
10/27/22	1.1	LL	Added depth of explanation to test specifications
10/27/22	1.2	WF	Finished functional test descriptions
12/4/22	1.3	WF	Removed Labyrinth file handling, users edit the map in the session.