ST1507 DATA STRUCTURES AND ALGORITHMS (DSAA)

CA1

Name	Emily Lim Xiang Qin
Class	DIT/FT/2B/13
Admin Number	1936207

1. Operation of application

a. Description:

This application is about using python's turtle to draw a maze and using algorithm to make the drone get out of the maze. Two algorithms were used to get the turtle out of the maze. These two algorithms are Left-hand Rule and Breadth-first search algorithm. The application allows user to choose different path finding algorithms to get out of the maze. Whenever the drone takes a step, the number of steps would be incremented in the title bar. After reaching the end point, the total number of steps would be shown. The application allows the user to change between the 2 different algorithms by using 'tab' key. The user is also able to repeat the path taken by the drone using the 'r' key. The user is able to pause the drone for 2 seconds by pressing the 'p' key.

b. User guidelines

i. Start the application by running the code in your anaconda prompt. Type python <path directory that you stored your code file in>. The text file of the maze needs to be in the same directory as the python code file. A new window showing the maze, name, title and icons in the maze should come out. The drone has different colours representing each algorithm. The start and end point also has different shapes and colours representing them.

ii. Press the 'tab' key to start the first algorithm, Left-Hand Rule.

iii. After reaching the end of the maze, press the 'tab' key if you want to change algorithm to Breadth-first search. The 'tab' key allows user to change between the 2 algorithms.

iv. If you want the drone to repeat the path again after reaching the end of the maze, press the 'r' key to repeat the path.

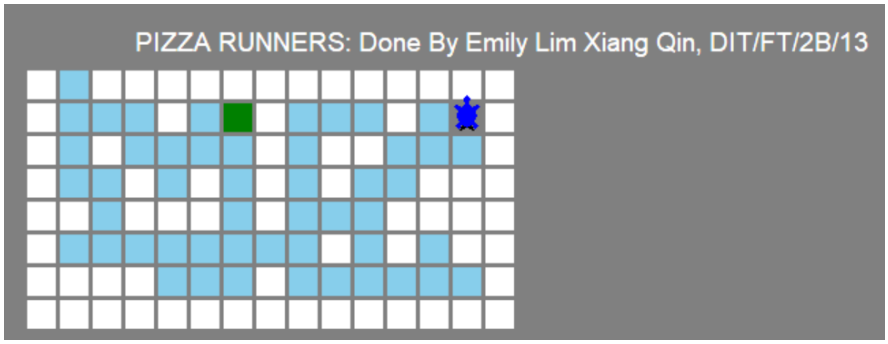
v. The dots appearing in the square boxes after the drone moved are the trails left behind by the drone.

vi. To pause the drone, press the 'p' key. Each time you press the 'p' key, the drone will stop moving for 2 seconds.

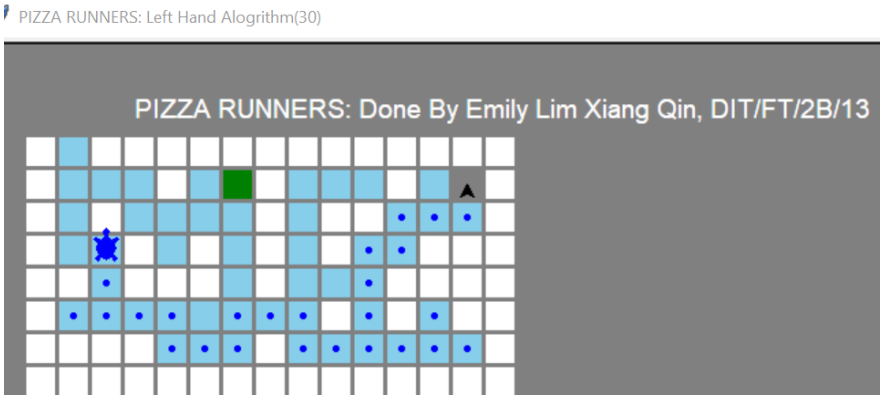
vii. To close the application, the user just needs to click on the screen of the window.

c. Screenshots

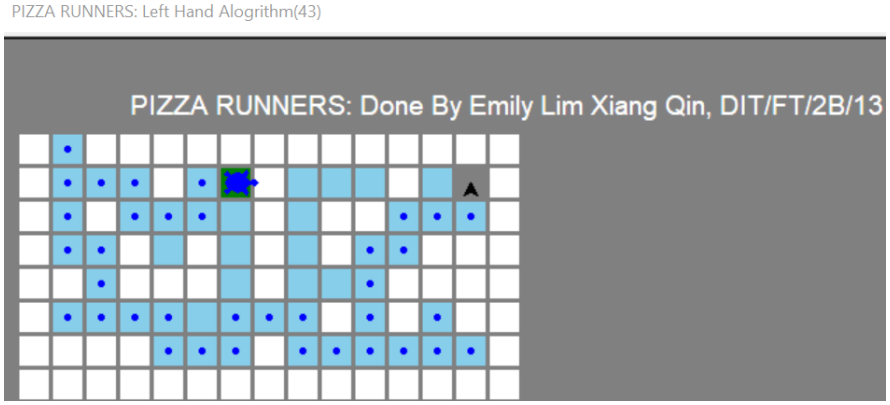
i. The start of the application:



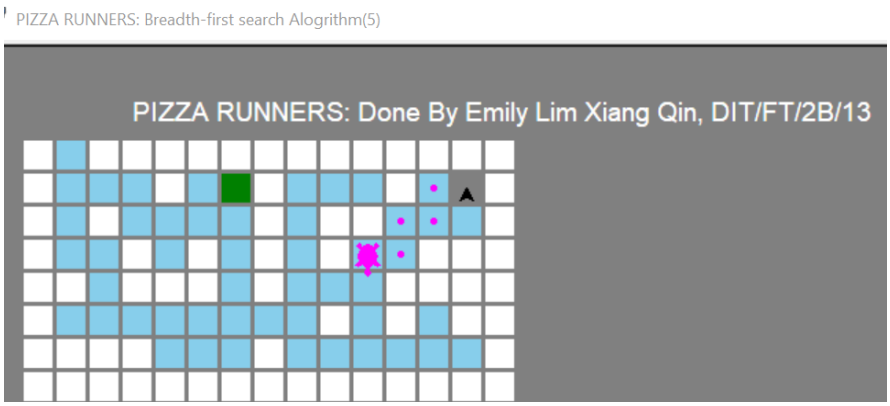
ii. When the left hand algorithm is implemented:



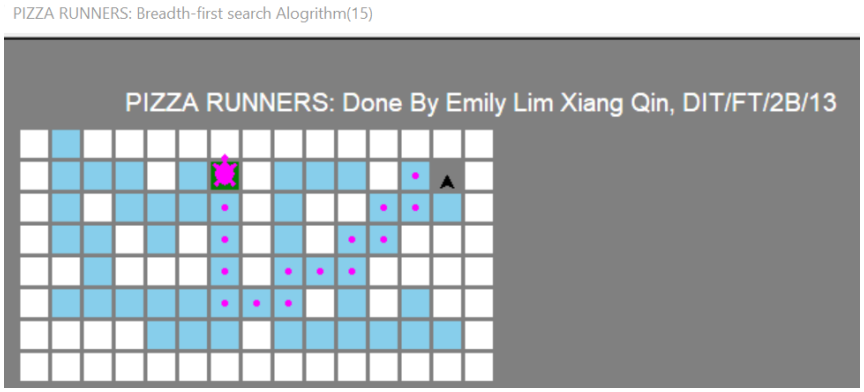
iii. After the left hand algorithm's drone has reached the end:



iv. The 'tab' key is pressed to change to the breadth-first search:



v. When the breadth-first search's drone has reached the end:



2. Algorithms implemented

a. Description

i. The left hand algorithm, is how a blind person(in this case the drone) would use their left hand to touch and follow the wall in the maze to find the exit. The drone will follow the path and travel accordingly to where the walls lead.

ii. The breadth-first search algorithm uses the shortest pathway to try and find a way out. The algorithm reaches to all parts of the maze and has to consider all the paths until reaching the end, causing the algorithm to stop the search. It also has to keep track of the paths visited in order to determine which is the fastest way out.

b. Comparison

i. The breadth-first search algorithm does not follow the wall like the left hand algorithm. It is a much better algorithm as it does not have to travel through long pathways to find the exit and it is able to get to the exit much quickly compared to the left hand algorithm. The left-hand rule is not the most efficient way as it does not consider the possible paths before reaching the exit.

ii. Using left-hand rule, user may sometimes find itself being trapped if the maze is in a loop. However, if user uses the breadth-first search they will not be trapped if the maze loops itself.

c. Challenges faced

i. It was really hard to use the python turtle as most of the time I had to search online and was not able to get what I needed. Sometimes after reading the documentation for the turtle graphics, I thought I understood what the code does. However, when I tried using it in my code, it does not seem to be the correct one. I felt that the left hand algorithm was very tricky as it might go into a loop and I had a hard time trying to think of the logic to make it work. When I changed some parts of the code, the previously working code might not work. Overall, I cannot say I enjoyed the process of doing this project but I

feel I have learnt a lot from this experience. It was a good learning process as I got to strethen my understanding and I felt a sense of satisfaction when my code works.

d. Performance issues

i. The turtle would hang if there was error in the code and it was very frustrating as I had to waste time waiting and trying to close the application.

ii. The error message was sometimes not very useful and it was hard to find solutions for it online. I had to go through the code again to see what I missed or changed.

iii. Even though I did penup, the maze initially did not load properly. Although I put the shape of the maze to be square, arrows and lines were drawn instead. I felt very confused as I did not put any arrow shape in my code.

e. Data structures

i. I used class for the program I created that requires many functions to work. I want to keep all the information under the same class for easier access and readability. For example, I have 4 functions for the tab class and I want them to be in the same class so that I can access them easily and I know that they belong together.

ii. Class was used for the 2 algorithms with functions that allows the solving of the maze. For the user to change algorithm and reset the maze, class was also used to store functions that had the logic to allow user to change and reset.

iii. 4 classes were used to create the shapes to make the walls, path, start and end of the maze. This allows them to be represented as an entity to create the map.

iv. The generatemap function does the reading of file and adding squares to make the maze using for loop.

v. The pause function was used alone and not put in a class as it only requires a simple time.sleep() to do the pause.

f. Limitations/constraints

i. In order for the maze to work properly, the user must wait patiently for the drone to reach the end of the maze before changing the algorithm or doing a reset.

ii. The user is not able to move the drone manually and can only press ‘tab’ for the drone to start moving.

iii. The application is not able to support very big dimensions of maze for example dimensions of 25 by 70.