CPS 112: Homework 4
Determine Analysis of MazeSolver's SolveMaze method.

Analysis of solveMaze method

Let n denote the number of Locations in the maze (width*height).

The solve Maze method starts by initialising instance variables (Line 34 and 35) which takes O(1) time each. The solveMaze method then calls the clear() method (Line 36) from the Agenda class which clears the agenda through ArrayList's clear() method that takes O(n) time. The solveMaze method then uses addLocation() (Line 37) method from the Agenda class that only adds a Location to the agenda, which takes O(1) time. The while-loop (Lines 39-116) contains an assignment of a Location variable (Line 41) which takes O(1) time, and then calls the pause() from MazeGUI which takes O(1) time since this method only does an Interrupted Exception check. solveMaze then uses the visitLoc() methods (Line 45-41) from MazeGUI which each take O(1) time since it only does a comparison and changes the color of the square.

The code block inside the first if-statement contains a while-loop (lines 51-56) that has assignments of Locations, adding/removing of Locations from an ArrayLists, each takes O(1) time. The block also calls the addLocToPath() method from MazeGUI that only does a comparison and changes the color of the square, so this portion only takes O(1) time. Since everything in the block took O(1) time, the overall block takes O(1) time. The while-loop itself iteration depends on the length from start to goal, however this while loop only occurs once during the whole solveMaze method so this while loop takes O(1) amortized time. The if-statement block ends with a return of an ArrayList, already created, so this portion only takes O(1) time. Since everything in the if-statement takes O(1), then the overall while-loop takes O(1) time.

The else-statement (Lines 60-115) contains if-statements that only performs comparisons and assignments of Locations and adding/removing of Locations from ArrayLists or Agendas. Therefore this block is only O(1).

The outer while-loop itself iterates until the agenda is empty, in the worst case scenario looping through the whole maze, therefore the outer while-loop takes O(n) time. After the while loop, the solveMaze method returns an ArrayList (Line 117), therefore only take takes O(1) times. The solveMaze method as a whole takes O(n) time, since the most time consumption took O(n) time.

Discussion

The properties of mazes important in determining which one is more efficient is how open the maze is, or in other words the lack of walls the maze contained. *faststack.txt* , large amount of white space, had more squares visited when using the queue agenda because the queue agenda fans out and therefore produces more gray area, while stack moves in a direct direction and therefore visits up half the amount of space queue does in an open maze. Similarly, *fastqueue.txt,* large amount of walls, had more squares visited when using the stack agenda because this agenda moved in a straight direction and therefore missed the direction the goal was in, while the queue agenda hit the goal by fanning out at all directions.

Since the queue-base agenda fans out when looking for the path, then the type of maze in which the queue-based agenda is more efficient would be the maze with a large amount of walls. This is because

queue fans out towards all directions and hits the target at about the same possibility no matter the direction the goal was from the start.
Since the stack-based agenda sticks to a specific route until it reaches a boundary, then the type of maze in which the stack-based agenda is more efficient would be the maze with a large amount of white space. This is because the stack agenda will not try to take up as much space as possible to reach the goal, but rather cover half the area by pushing towards one coordinate direction.

Based on how my stack and queue act on the *blank.txt* maze, the queue agenda creates the shortest path from start to goal. Since the stack agenda goes straight through every other line, then the path given is the whole zig-zag line. However, the queue agenda fans out so this agenda covers more squares on the way to the goal which allows the agenda to provide the shortest path from start to goal, instead of the long path it covered.