CALIFORNIA STATE SCIENCE FAIR
2017 PROJECT SUMMARY

Name(s)

Jonathan Z. Xu

Project Number

J1508

Project Title

Analysis of Maze Solving Algorithms

Abstract

Objectives/Goals

Find the most efficient maze-solving algorithm among three different algorithms: the #right-hand rule#, #dead reckoning#, and #dead end remembering#.

Methods/Materials

The programming language I used is called Java. I developed all the algorithms in an Integrated Development Environment (IDE) called Processing. I downloaded open-source code from GitHub for random maze generation. I programmed different maze-solving algorithms. Then I ran 8 tests, each measuring how many moves taken by each algorithm on the same maze. The maximum number of moves, minimum number of moves, and average number of moves by each algorithm among all 8 tests are recorded. The code generates a random maze each test so that a variety of mazes with different possibilities are covered.

Results

Right-hand rule kept a consistent range (178~397), as well as having the least average (293.87). Dead end remembering was better than dead reckoning, with less major outliers.In test #2, the maze generated had many forks, which resulted in the two random algorithms (#dead reckoning# and #dead end remembering#) having many more moves. However, in test #8, dead end remembering takes only 43 steps to reach the center comparing to more than 200 steps by the other algorithms.

Conclusions/Discussion

I believe dead end remembering was still the most efficient out of the three. While the maze generation code was programmed not to generate loops, the right-hand rule would be stuck in an infinite loop, but eventually the random algorithms would be able to solve the maze. While dead reckoning is too pointless, getting stuck in long loops over and over, dead end remembering was made to improve on this. It will not be stuck in a loop for too many times due to its mental wall creation technique, and will efficiently solve any maze.

Summary Statement

I programmed 3 different algorithms in Java and I tested them on 8 randomly generated mazes to determine the most efficient algorithm.

Help Received

I downloaded open-source code from GitHub for random maze generation.