Maze Solving Algorithms

April 20, 2020

- Ayush Agrawal cse180001011

- Harsh Chaurasia cse180001018

Overview

Maze solving algorithms refer to automated methods to solve a labyrinth, i.e, to find a route between the starting and ending points of a maze. Mazes containing no loops are known as "simply connected", or "perfect" mazes, and are equivalent to a tree in graph theory. Thus many maze solving algorithms are closely related to graph theory.

Several algorithms for solving mazes exist, each with its own set of benefits and demises. For eg. The simplest DFS (Depth First Search) has moderate space complexity, but exponential time complexity.

This project aims at implementing various algorithms and analyzing their Time and Space Complexities.

Specifications

Before research on solving mazes mathematically, several low-level techniques were implemented to solve mazes. These were not very effective, and often, could not even find a solution. Some trivial algos are discussed below.

Wall Follower - This is the simplest Maze solving algorithm. According to the rules, we start off at the junction and always turn right (or always turn left). If we keep marking each point we visit, then the points visited only once are the answer. It would be quite obvious that this algo wont function if there exists a closed boundary around our finish line (finish line is in the center of the maze).

Flood Fill - Flood Fill algorithm is one of the best maze solving algorithms. It involves assigning values to each of the cells in a maze where these values represent the distance from any cell on a maze to the destination cell, done in four parts: update walls, flood maze, turn determination and move to the next cell. Keep Flooding the passages sequentially, assigning distances of the next cells. If at the end, the end isn't filled, the Maze has no solution. For Mazes with an entrance and exit on the edges, FloodFill one wall, and the remaining edge marks out the solution. For Mazes with the start or goal inside the Maze, FloodFill the surrounding wall, and if the exit wall isn't erased, wall following won't work to solve it. This technique is used in several other Algos at some point.

Dijkstra Algorithm - In this algorithm we generate a Shortest Path Tree (SPT) with a given source as root. We maintain two sets, one set contains vertices included in the shortest path tree, the other set includes vertices not yet included in the shortest path tree. At every step of the algorithm, we find a vertex which is in the other set (set of not yet included) and has a minimum distance from the source.

Goals

1. To analyze the existing 2D Maze Solving Algorithms.
2. To implement these algorithms.
3. Optimizing the available techniques for better efficiency.

Applications of efficient MazeSolvers

Terrain Walking Robots , need to explore the environment efficiently. The speed of the robot to find its path, affected by the applied algorithm, acts the main part in the present projects. We need quick exploration algorithms.

Other areas include Navigation Systems - Intelligent traffic control that helps ambulances, fire fighters, or rescuing robots to find their shortest path to their destination.

Related Research Papers

1. Huijuan Wang, Yuan Yu and Quan Bo Yuan, "Application of Dijkstra algorithm in robot path-planning," 2011 Second International Conference on Mechanic Automation and Control Engineering, Hohhot, 2011, pp. 1067-1069.
2. Y. J. Zhang, Z. L. Zhang and Y. Deng, "An improved maze solving algorithm based on an amoeboid organism," 2011 Chinese Control and Decision Conference (CCDC), Mianyang, 2011, pp. 1440-1443.
3. B. Rahnama, M. C. Özdemir, Y. Kiran and A. Elçi, "Design and Implementation of a Novel Weighted Shortest Path Algorithm for Maze Solving Robots," 2013 IEEE 37th Annual Computer Software and Applications Conference Workshops, Japan, 2013, pp. 328-332.
4. B. Rahnama, A. Elçi and S. Metani, "An Image Processing Approach to Solve Labyrinth Discovery Robotics Problem," 2012 IEEE 36th Annual Computer Software and Applications Conference Workshops, Izmir, 2012, pp. 631-636.
5. D. Osmanković and A. Aćimović, "K Modal Logic Approach to Maze Solving," 2018 IEEE 12th International Symposium on Applied Computational Intelligence and Informatics (SACI), Timișoara, 2018, pp. 000125-000130.
6. A. M. J. Sadik, M. A. Dhali, H. M. A. B. Farid, T. U. Rashid and A. Syeed, "A Comprehensive and Comparative Study of Maze-Solving Techniques by Implementing Graph Theory," 2010 International Conference on Artificial Intelligence and Computational Intelligence, Sanya, 2010, pp. 52-56.
7. P. H. Kim and R. Crawfis, "The quest for the perfect perfect-maze," 2015 Computer Games: AI, Animation, Mobile, Multimedia, Educational and Serious Games (CGAMES), Louisville, KY, 2015, pp. 65-72.