RoboMind Challenges

Maze solving

Always find the exit in any maze

Difficulty: ★★☆ (Medium), Expected duration: Couple of days

Description

In this activity you will use RoboMind, a robot simulation environment, to solve simply connected mazes. A simply connected maze is a maze has no loops, and no inaccessible areas. Robotics competitions often include maze solving challenges. This activity is a good introduction to the programming tasks one might face as part of a robotics club or team, preparing for competition! Besides, after completing this activity, you'll never again have to worry about being hopelessly lost in a maze.

In this activity, you will:

- Become familiar with RoboMind, a robot simulation environment
- Program the robot to solve a simply connected maze
- Develop an understanding of maze algorithms, and turn your algorithms into code for your robot

Note: there is programming involved in completing this activity. The RoboMind programming language is very simple, however this activity does not include a programming tutorial. If you don't have any programming experience, you might want to try the Getting Started activity first, as it includes step-by-step instructions for programming the RoboMind robot.

Make sure you have the latest software

You need the RoboMind software to complete this activity. If you have done other RoboMind activities, you should already have the software installed on your system. If not, download and install the software that is available on www.robomind.net.

Introduction

In this exercise, you will use the RoboMind robot simulation environment to program a robot to solve simply connected mazes. Maze solving challenges appear often in robotics club competitions, so this is good practice if you're involved with a robotics club or thinking about joining robotics contests. It's also a fun activity for anybody who is into puzzles.

By completing this exercise, you will gain experience with programming a robot to solve mazes, and also learn about maze solving algorithms. Algorithms are a description of the steps one takes to solve a problem; a maze solving algorithm is just the rules that the robot will follow to solve the maze, once you have translated it into code.

Getting Started

The first thing you will need is a maze to solve. We'll start out with a small maze, so as you work on your code, you can determine pretty quickly whether or not it is doing what you want it to do. Once you think you have your robot trained to solve mazes, you can turn it loose on a larger maze and see how it does.

The starter maze looks like this:

RoboMind Maze Solving Starter Map

Open the map smallmaze.map in RoboMind using the **File > Open map**. Once the map is loaded you will see the robot in its starting position near the lower left corner, and a beacon near the upper right corner. To solve the maze, the robot must find the beacon, and then pick up the beacon and walk onto the space previously occupied by the beacon.

Getting From Here to There, Simply

The simplest (and least interesting) way to solve the maze is by dead reckoning. With dead reckoning, the robot's position is calculated based on its previous position. So, for example, if you told the robot to move 4 spaces to the north, you know that the robots final position after movement will be 4 spaces north of its previous position. Nothing too tricky there! So, using dead reckoning, you could examine the maze, solve it yourself, and then write a simple program that tells the robot to move 4 spaces north, then 4 spaces east, then 2 spaces south, etc., until the robot arrives at the beacon to complete the task. Technically, that is a solution for this maze, but your program is not a general maze solver; give the robot a different maze, and, using the same program, it will fail to find the beacon. However, if you are new to RoboMind, you might consider "warming up" by writing a short program to solve the maze by dead reckoning. You can use Run > Remote control to drive the robot around and have it write the instructions for you, or you can look under Edit > Insert... for the RoboMind commands. And of course you can always consult the online RoboMind programming documentation.

Getting From Here to There, Intelligently

So, how do you go about writing a program that can solve any simply connected maze? Fortunately for us, the problem of solving mazes has been studied by a lot of smart people, and many algorithms have been developed to solve different types of mazes. Some of these algorithms focus on the robot (or person, or mouse, etc.) that is inside the maze and can only see its immediate surroundings. These types of algorithms are good for walking around inside a corn maze, or for a robot or (very smart) mouse inside of a maze. Other algorithms focus on the maze itself, and assume that the entire maze can be observed at once. These types of algorithm would be useful to a person solving a traditional paper-and-pencil maze. For this exercise, you will want to look for the first kind of algorithm (focus on the person/robot) rather than the second (focus on the whole maze).

Maze Solving Algorithms

Searching around on the Internet you can find all sorts of information about mazes, including descriptions of algorithms for solving them (as well as computer code for them!). One really great web page has a nice description of different kinds of mazes, algorithms for creating mazes, and also algorithms for solving mazes. About 3/4 of the way down the page, there is a description of several maze solving algorithms. After the descriptions, there is a table, classifying the algorithms. We're interested in the ones that say Inside/Above in the Human Doable? column. Those are the algorithms that are useful for a robot that is inside of a maze and trying to find its way out.

Random Mouse Algorithm

The Random Mouse algorithm is just what it sounds like: it's the way a mouse would solve a maze. And no, I don't mean by following its nose to the cheese. The Random Mouse algorithm is just wandering around the maze, hoping you find the end. Start your robot out by going straight, and keep going straight ahead (following any turns that the maze makes) until you come to any point where there is more than one way to go (an intersection or junction). At this point, make a random choice which way to go, and do it. Then go back to just following the maze until you come to the next junction. At the junctions, only include left, right, or straight in your random choice; never turn 180 degrees and go back the way you came. The only time you ever turn around and go back is when you come to a dead-end (front, left, and right are all blocked). That's all there is to it. If you do a Random Mouse for your robot, expect it to take a LONG time for large mazes. In

fact, it's kind of fun to start a Random Mouse run on a large maze before you go to sleep at night, and see if it has finished when you get up in the morning!

Wall Follower Algorithm

The Wall Follower algorithm is very simple to understand. At the start of the maze, put your left (or right) hand on the wall, and then start walking. Never take your hand off the wall. For a simply connected maze, you will find the exit. That's all there is to it. The picture above shows a maze solved using the Wall Follower algorithm; the red path shows the solution, the gray path shows the other parts of the maze that were visited while following the wall. Look carefully at the maze, and see if you can figure out if this was left- or right-hand wall following.

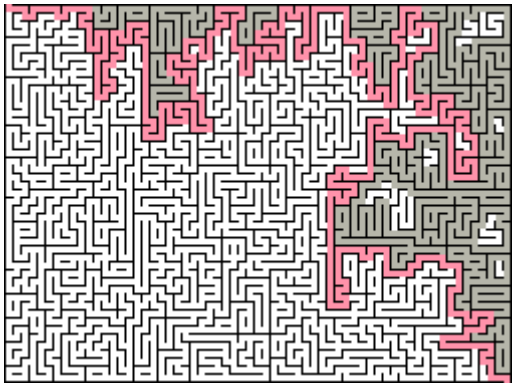

Figure 1 Wall Follower Maze Solution

Pledge Algorithm

The Pledge algorithm is similar to the Wall Follower algorithm, and it includes wall following as part of its solution. The Pledge algorithm is more powerful than Wall Follower, and can solve some mazes that Wall Follower can't. However, the Pledge algorithm points out a limitation of the RoboMind programming language: RoboMind doesn't support storage of values in variables. If you read the detailed description of the Pledge algorithm on the maze web page posted earlier, you'll see that the robot would need to count the number of turns it has made, and with RoboMind, there is no way to do this. Sorry, you can't do the Pledge algorithm for this exercise, but keep it in mind for real-life robot competitions.

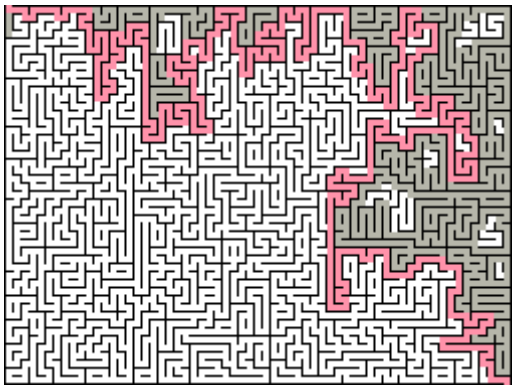

Figure 2 Pledge Algorithm Maze Solution

Tremaux's Algorithm

Ok, this one is built for RoboMind! Read the description of the algorithm on the web page. Note that you will be marking the maze while you walk it. You finally get to use the robot's paint brush! Note that the robot can paint white and black, and you need to be able to mark paths the first time you've seen them, and the second time you've seen them. Pick one color for your original marking, and the other color for your secondary marking. When you're done, not only will your robot be at the beacon position, but you should be able to see the solution marked out in your primary color! This one may be a bit challenging, but give it a try and show off your

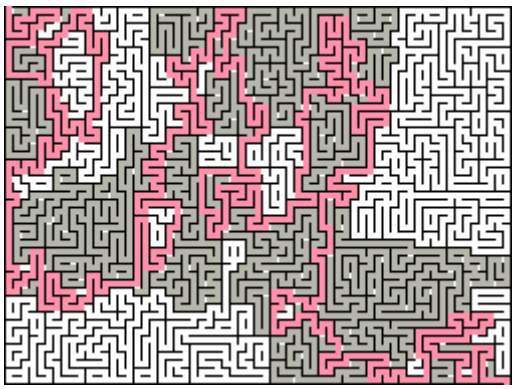

Figure 3 Tremaux's Algorithm Maze Solution

robotics and programming skills!

Your Task

So now that you've read all about mazes and algorithms, here's the task. Implement at least one of the algorithms, save the program, and submit it to the activity. The activity will require 1 file submission, but will allow for additional optional uploads. There are 3 algorithms that you should be able to implement with RoboMind: Random Mouse, Wall Follower, and Tremaux's. Submit a program for at least 1 of the algorithms, and if you want, submit programs for any of the other ones that you do. And if you do Tremaux's, it would be cool if you could grab a screenshot of your solved maze, with the solution painted on the map, and submit that, too.

Oh, and remember how that first maze was just a starter maze, to use while you're getting your program to work? Open the map bigmaze.map to use for testing your program. Your Random Mouse may need a day or two to solve this one!