CS 3050: Group Project

CS 3050

November 8, 2016

DUE: 12/08/2016

Topic: Motion planning of robots

Description

This is the final project of the course. The project can be done alone or in a group of at most 3 people. The project contributes to 12% of your total grade. You should send an email to Rina Bao, Roshan Neupane, and myself by November 18 with the subject "CS3050 project" indicating whether you are choosing to do the project alone or in a group. If you choose to do the project in a group, please include the name of your teammates in the project (one email per team is fine).

The purpose of this project is to utilize and build upon your knowledge in graph searching algorithms. In robotics, it is common to traverse through rooms or buildings and gather data. To do this the robot traverses through, using graph searching, until it finds its path. In this project you will be implementing a motion planning algorithm which coordinates the motion of one robot in a room with 2 moving obstacles. The robot starts at starting point F and has to travel to exit location L. The room itself is in a square shape and for the purposes of the project, you may assume that the room in the shape of a $n \times n$ grid surrounded by walls all around. A location in the room will be denoted by a pair of coordinates (x, y) where x and y are positive integers between 1 and n. Figure gives a snapshot of the room with a robot and two obstacles. The coordinate $(1, 1)$ indicates the North-West corner, $(1, 5)$ indicates the North-East corner, $(5, 1)$ indicates the South-West corner, $(5, 5)$ indicates the South-East corner.

Robot movement. In the room, a robot can only move at most one grid square per time instant, which could be any of its 8 possible neighbors. Of course, if the robot is up against a wall, it cannot break the wall and so its available next moves will be limited! Also, the robot can't teleport to other parts of the room.

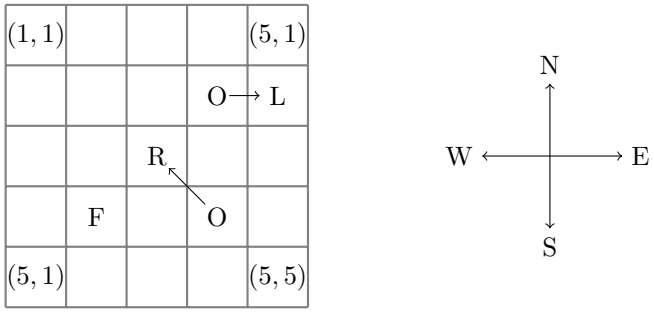

Figure 1: A snapshot of the room which is a 5×5 grid. The robot is at $(3, 3)$. One obstacle is at $(2, 4)$ and is moving eastwards. The second obstacle is at $(4, 4)$ and moving North-West. The starting position F is at $(4, 2)$ and the exit location L is at $(2, 5)$.

Obstacle movement. Each of the two moving obstacles themselves will have a speed and an initial direction. The speed is specified as the number of grid square movements per time instant. The initial direction of an obstacle is specified by a pair (a, b). a, b can be $+1, -1$ and 0 and correspond to the directions in the following way. The first component, a, indicates the direction along the North-South Axis with -1 meaning Northwards, $+1$ meaning Southwards and 0 meaning no movement along the North-South axis. The second component b indicates the direction along the East-West Axis with $+1$ meaning Eastwards, -1 meaning Westwards and 0 meaning no movement along the East-West axis. So, for example $(+1, +1)$ means that the initial direction of the obstacle is South-East, $(+1, -1)$ means that the initial direction of the obstacle is South-West, $(0, -1)$ means that the initial direction of the obstacle is Westwards and $(0, 0)$ means that the robot is stationary.

Obstacles hitting a wall. When the obstacle hits a wall then it changes direction as follows. If it hits either the Northern or the Southern wall, then the first component of the direction changes sign. In other words if a was $+1$ in the current time instant then it will be -1 in the next time instant, and if it was -1 in the current time instant then it will be $+1$ in the next time instant. Similarly if it hits the Eastern or Western wall then the first component of the direction changes sign. If it hits a corner, then both the components change sign.

Mechanics. In each time instant, both obstacles will move according to their respective current directions and current speed. That is if its speed is r grid squares per movement, it will move r grid squares (it may have to change direction!). Also at the same time instant, you may move your robot to any of its 8 possible neighbors or choose to remain at your current location. Both

the obstacles can be in the same location in the grid. But if your robot and an obstacle land in the same square then your robot dies!

Goal. Your goal is to plan a path for the robot that takes it from the F to L.

Constraints

The most natural way to implement the project is to view the room as a graph with each square of the grid representing a vertex (what are the edges?) If you choose to use this representation, you will develop a graph traversal algorithm which finds a path for the first robot from (F) to (L) in presence of moving obstacles.
The constraints of the program include:

1. The program can be written in C, C++ or JAVA. If you want to use a different language, please check with us.
2. Either use a Netbeans project or use a Makefile to compile and run the program.
3. Include a README file to tell the TA's how to run your program and how to interpret your output.
4. You should include a project report in which you detail your efforts, the algorithms used (along with their complexity) and the contribution of your team members. We shall use this report primarily to give you partial credit if your program does not work.
5. You may not use any graph theory libraries to solve this program.
6. All graphing algorithms should be implemented.
7. A moderate amount of error checking and resource management is required. Your application should check that each line from the input file is properly formatted, that the file is successfully opened, the file is successfully closed upon reading of the file and that all allocated space is deallocated at the exit. If the input is not properly formatted, you should report the error on stdout and exit the program.
8. You may use standard input/output libraries, arrays, vectors, stacks, heaps, lists and queues. If using other libraries, please please check with us.
9. A moderate amount of formatting and documentation is required. Comments should be descriptive and used to illustrate the purpose and inner workings of an algorithm or function; they should not be used to annotate each line or self-evident logic.

Input

Your program should take at least one arguments on the command line. Thus is the input file to the program. The input file itself is formatted as follows.

1. The first line contains a non-negative integer which is the length of the room, n.
2. The second line will be the starting location of the robot F. The location F will be written as (x, y) where x and y are positive integers between 1 and n.
3. The third line will be the exit location of the robot L. The location L will be written as (x, y) where x and y are positive integers between 1 and n.
4. The fourth line will be the starting location of the first obstacle. The location will be written as (x, y) where x and y are positive integers between 1 and n.
5. The fifth line will be a non-negative integer, which will be the speed of the first obstacle.
6. The sixth line will be the direction of the first obstacle. The direction will be written as (a, b) where a, b can be $+1$, 0 or -1.
7. The seventh line will be the starting location of the second obstacle. The location will be written as (x, y) where x and y are positive integers between 1 and n.
8. The eight line will be a non-negative integer, which will be the speed of the second obstacle.
9. The ninth line will be the direction of the second obstacle. The direction will be written as (a, b) where a, b can be $+1$, 0 or -1.

Sample Input File

```
10
(1,3)
(10,9)
(10,8)
1
(0,+1)
(2,2)
1
(-1,+1)
```

Due Dates

- November 18 is the date that you must inform us about the composition of your team.
- During the week of November 28-December 2, please try to meet one of the TAs or myself during office hours and show the progress of the project. If the times do not work, please send us an email and we will try to find additional time to check the progress.
- The final submission is due December 8 at 11:59:59 pm.

Submission

You should place all your submission material into a folder named as follows: *pawprint1_pawprint2_pawprint3_project* where the *pawprint1_pawprint2_pawprint3* is the list of you and your group member pawprints. You will submit one file, a compressed directory containing the project report, all the source code and appropriate Makefile(s). The naming convention should be: *pawprint1_pawprint2_pawprint3_project.tgz* or *pawprint1_pawprint2_pawprint3_project.zip*.

Grading

There are 120 points possible for this assignment. The grade breakdown is as follows:

- 15 points for README, error checking and resource management.
- 20 points for general programming style and adherence to the constraints.
- 20 points for the project report.
- 15 points for correctly inputting the starting configuration.
- 50 points for finding and displaying the paths taken by the robots.