CS 81 - Principles of Robot Design & Programming

Carter Kruse

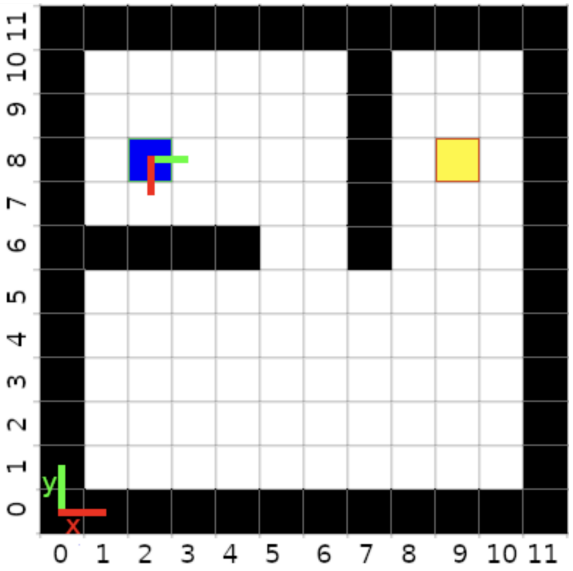
Homework 4 - Planning

Purpose

In this assignment, you will put in practice some of the planning concepts learned in class.

General Instructions

Given the following environment represented as a grid (black cells: obstacles, white cells: free space; each cell denoted with coordinates x, y), where a robot, able to move up/down/left/right with cost 1, starts from the the blue cell and wants to get to the yellow cell.



Answer to the following questions, showing all the relevant drawings and intermediate math work by uploading a PDF file containing the work. You can include code written by yourself if used for doing some of the repeated calculations, but write down the formulas used on paper. Feel free to add comments in the text box.

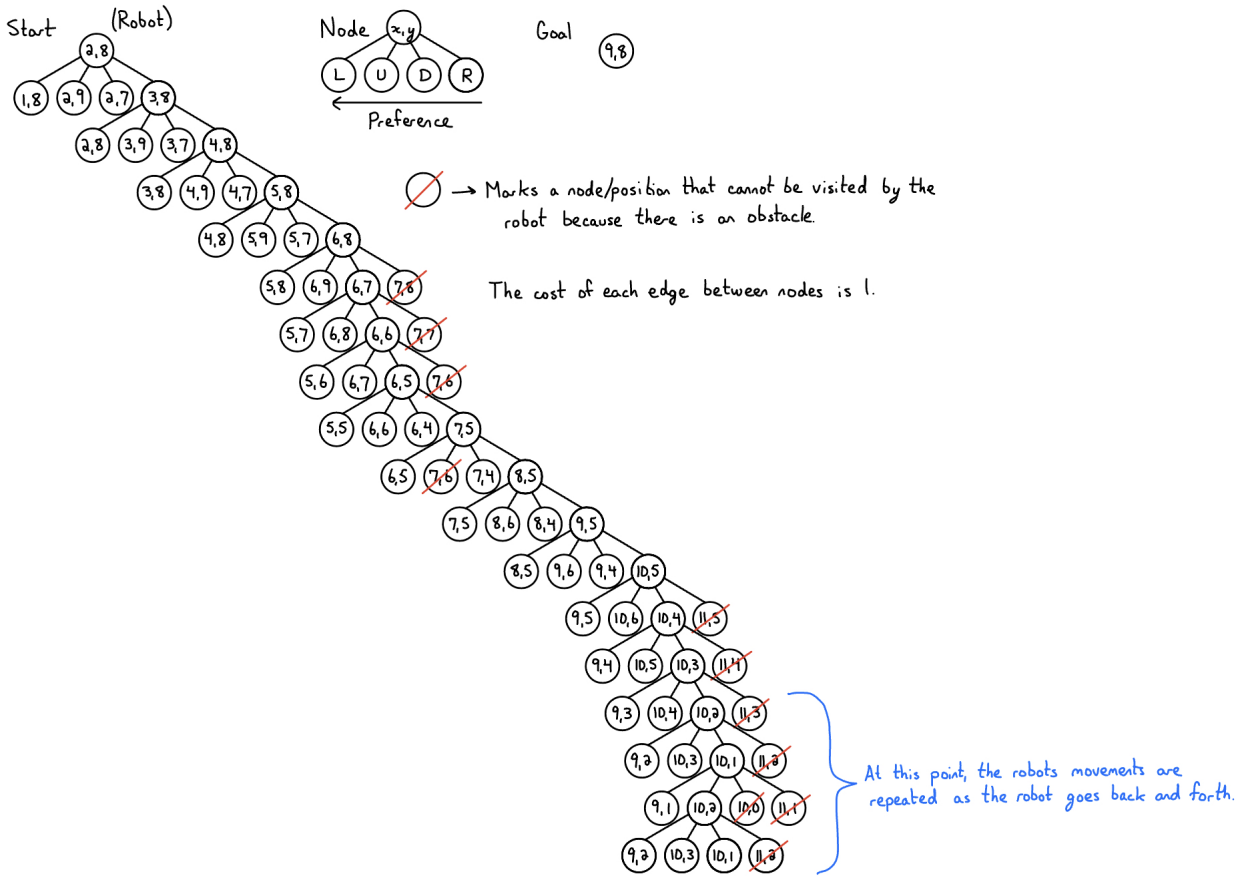
Question 1

Apply a depth first search (DFS) algorithm (tree search, without history), where the order of preference for motion is right, down, up, left (with respect to the global map frame located at the bottom left, i.e., East-Right, North-Up,...).

Draw the draw the expanded tree, marking the related cell and cost, for each node, and explicitly showing the order in which nodes are expanded, and the final path (if any), found by the algorithm.

When applying a depth first search (DFS) algorithm (tree search, without history), where the order of preference for motion is right, down, up left (with respect to the global map frame located at the bottom left), a stack implementation is used.

This stack implementation may be expressed as an expanded tree, with related nodes, as demonstrated below.



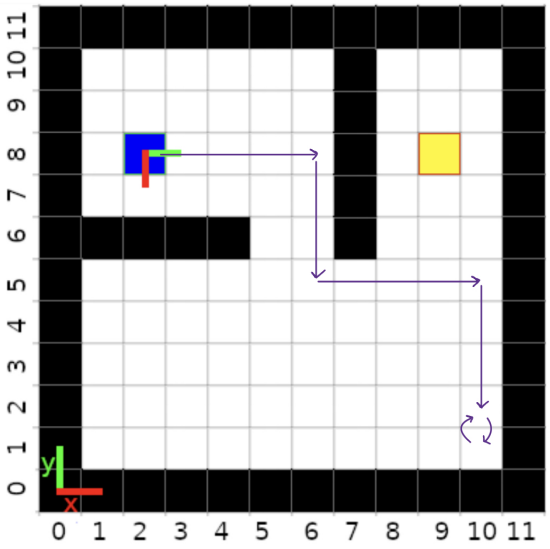
The robot starts at the node $(2, 8)$ with the goal of reaching the node $(9, 8)$. As depicted, the order of preference is right, down, up, left, so adjacent nodes are added to the stack in reverse order. In this way, the rightmost (available) node is always selected (for the robot's movement).

The terms node, cell, and location are used interchangeably from this point forward.

To mark nodes that cannot be visited by the robot (i.e. there is an obstacle preventing movement), a red line is used to "cross-out" the node. In this way, we are able to quickly determine the action of the robot, according to the related cells/nodes, explicitly showing the order in which nodes are expanded.

As indicated in the drawing, the cost of each edge between nodes is 1, so the cost of each node is simply the cost of the parent node plus one.

To show the final path found by the algorithm, consider the following:



Comment on whether the DFS applied in this question terminates or not. If not, please describe how to solve the non-termination problem.

The DFS applied in this question does NOT terminate, as indicated by the "looping/oscillating" nature of the robot's movements. This is due to the order of preference for motion of the robot, which results in going from node $(10, 1)$ to $(10, 2)$ and back. To this end, the robot never reaches the goal node.

To solve the non-termination issue, it is not as simple as using a different ordering of preference for the motion of the robot. This would result in the same "looping/oscillating" effect in a different location, regardless of the ordering.

Rather, by considering the history of the robot's positions, we are able to solve the non-termination issue. By preventing the robot from re-visiting a node it has already visited, we may ensure that the robot does not become "stuck" in a given position. For this particular application, however, this is not enough, as we "cut off" any potential paths to the *goal* node in our initial pass, as evidenced by the drawing. To resolve this issue, a different ordering must be used.

Alternatively, we may use a different path planning/finding algorithm, such as breadth-first search (BFS), Dijkstra's algorithm, greedy best search, or A* to determine the path from the robot's current location to the goal location.

Question 2

Assume that the length of each side of the cell is $0.5\,m$.

Write what the poses of the robot and the goal are in the 'odom' reference frame, marked at the center of the blue cell with the red and green vectors corresponding to x and y, respectively.

In the ‘odom’ reference frame, marked at the center of the blue cell with the red and green vectors corresponding to x and y, respectively, the poses of the robot and goal are relatively simple.

$$\text{Robot} \longrightarrow \begin{bmatrix} x \\ y \\ z \\ \theta \end{bmatrix} = \begin{bmatrix} 0\,m \\ 0\,m \\ 0\,m \\ 0\,rad \end{bmatrix}$$

$$\text{Goal} \longrightarrow \begin{bmatrix} x \\ y \\ z \\ \theta \end{bmatrix} = \begin{bmatrix} 0\,m \\ 3.5\,m \\ 0\,m \\ 0\,rad \end{bmatrix}$$

Write general formulas that for an arbitrary pose will transform coordinates in the ‘grid’ reference frame to the ‘odom’ reference frame, and vice versa.

In the image, the ‘grid’ reference frame is where the red and green lines are at the bottom left on the grid.

The general formulas that for an arbitrary pose will transform coordinates in the ‘grid’ reference frame to the ‘odom’ reference frame, and vice versa, incorporate transformation matrices (with rotation and translation) as follows:

$$^{grid}p = {}^{grid}T_{odom}\,{}^{odom}p$$

$$^{odom}p = {}^{odom}T_{grid}\,{}^{grid}p$$

The ^{grid}p and ^{odom}p give the pose in the ‘grid’ and ‘odom’ reference frames, respectively, written as follows:

$$p = \begin{bmatrix} x \\ y \\ z \\ \theta \end{bmatrix}$$

The transformation matrices are as follows:

$$^{grid}T_{odom} = \begin{bmatrix} \cos\left(-\frac{\pi}{2}\right) & -\sin\left(-\frac{\pi}{2}\right) & 0 & 1 \\ \sin\left(-\frac{\pi}{2}\right) & \cos\left(-\frac{\pi}{2}\right) & 0 & 4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 1 \\ -1 & 0 & 0 & 4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{odom}T_{grid} = \begin{bmatrix} \cos\left(\frac{\pi}{2}\right) & -\sin\left(\frac{\pi}{2}\right) & 0 & 4 \\ \sin\left(\frac{\pi}{2}\right) & \cos\left(\frac{\pi}{2}\right) & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 & 4 \\ 1 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

In this circumstance, the “coordinates” are interpreted to be the actual x, y locations/positions, rather than simply the labels used for the cells. This is to allow the robot’s motion to be continuous, rather than discrete (according to strictly the use of the cells).

This is why the values of 1 and 4 are used for the translation, rather than 2 and 8 (respectively), as the length of each side of the cell is $0.5\,m$.