HOMEWORK 4

JULIEN BLANCHET

OCTOBER 22, 2019 • COSC 181 • PROF. LI

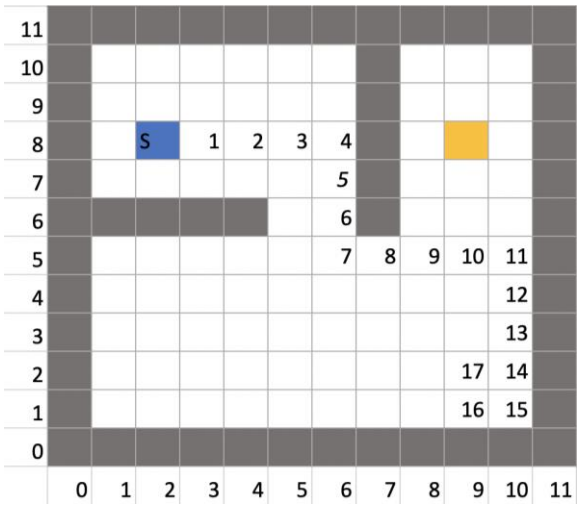
THE TASK

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 blue cell and wants to get to the yellow cell.

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, drawing 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.

ILLUSTRATION (QUESTION 1)



Note: number in the cell marks both the cost and the sequence in which the nodes have been expanded (they're equivalent, since the cost increases by 1 each time).

EXECUTION TABLE (QUESTION 1)

- Green indicates chosen (next expanded) node
- Red indicates previous node (disallowed)
- Gray indicates a wall
- Note that the order of expansion will be from the left to the right, skipping over walls and the previous node

Cost	Current	Right	Down	Up	Left
0	2,8	3,8	2,7	2,9	1,8
1	3,8	4,8	3,7	3,9	2,8
2	4,8	5,8	4,7	4,9	3,8
3	5,8	6,8	5,7	5,9	4,8
4	6,8		6,7	6,9	5,8
5	6,7		6,6	6,8	5,7
6	6,6		6,5	6,7	5,6
7	6,5	7,5	6,4	6,6	5,5
8	7,5	8,5	7,4		6,5

EXPLANATION

For this problem, I simulated a DFS algorithm which has does not keep track of which nodes have been previously visited except for the direction it just came in. For example, if we start at (2,8) and visit (3,8), then going immediately back to (2,8) is disallowed. This slight improvement does not impact time or space complexity and felt worth it.

NODE TREE

QUESTION 2

Does the DFS terminate? If not, please show a way to solve the problem by modifying the DFS algorithm.

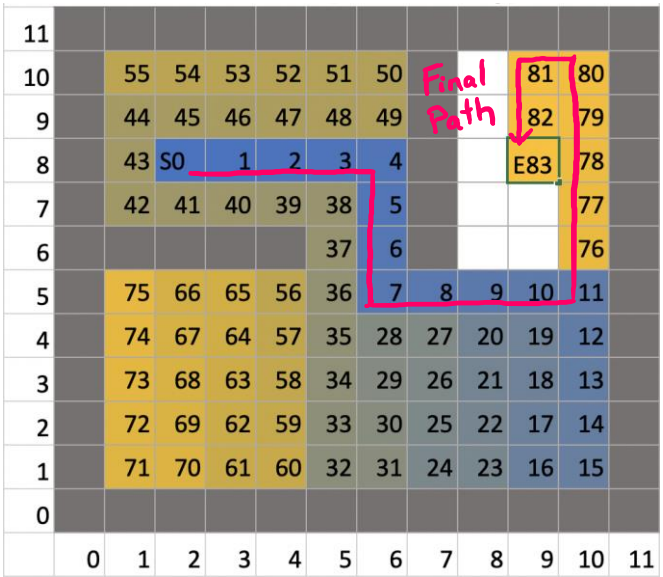
EXPLANATION

Despite the above improvement, the algorithm does not terminate. In the diagram above, you can see that when the search has reached the 17th node at (9,2), the next node to search would be (10,2) based off the motion preference to move rightwards. At this point it would continue to search in an infinite loop in the order:
(9,2)→(10,2)→(10,1)→(9,1)→(9,2).

To solve this behavior, I'd modify the algorithm to keep track of previously visited nodes and disallow visiting them again. This increases the space complexity of algorithm to O(V) instead of O(1), where V is the number of cells / vertices in the grid. However, this modification prevents infinite loops and guarantees completeness (however, it does not guarantee optimality).

As applied to this problem, the search algorithm will expand nodes 0 (located at 2,8) through 17 (located at 9,2) in the same fashion as before. However, since (10,2) was already expanded as node 14, the search continues upwards to (9,3) and continues in the order you see in the diagram below.

ILLUSTRATION



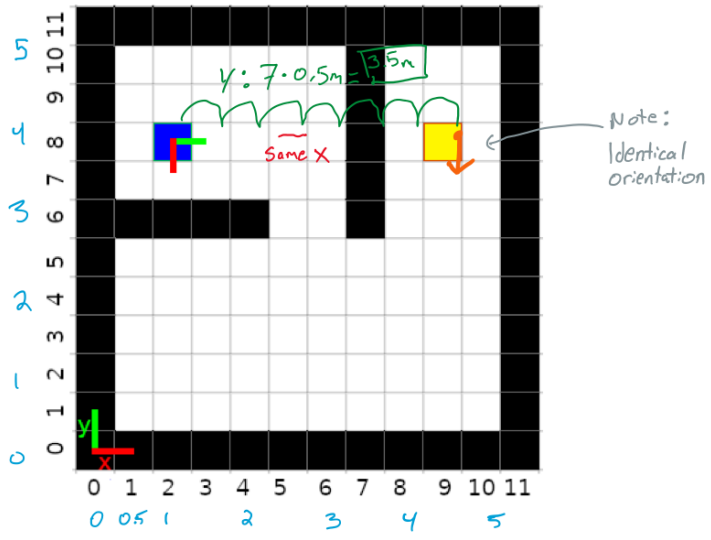
FINAL PATH & COST

Node	Cost
S0	0
1	1
2-10	…
11	11
76	12
77	13
78-81	…
82	18
E83	19

The number in each cell indicates the order in which the nodes are expanded. For nodes 0-55, it also indicates the cost to get to that point.

QUESTION 3

Assuming that the length of each side of the cell is 0.5m, what is the pose of the robot and the goal 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? Show also general formulas that for an arbitrary pose will transform coordinates in the grid reference frame to the odom reference frame, and viceversa. Note that the image reference frame is where the red and green lines are at the bottom left on the grid.



Note: $\theta_{odom} = -90° = -\frac{\pi}{2}$

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

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

$$ {}^{map}\theta = \theta_{odom} + \frac{\pi}{2} $$

$$ {}^{odom}T_{map} = {}^{map}T_{odom}^{-1} = \begin{bmatrix} 0 & -1 & 4 \\ 1 & 0 & -1 \\ 0 & 0 & 1 \end{bmatrix} $$

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

$$ {}^{odom}\theta = \theta_{map} - \frac{\pi}{2} $$

Robot Pose @ Start *odom_frame*

x	0
y	0
orientation (yaw)	0

Robot Pose @ Goal *odom_frame*

x	0
y	3.5m
orientation (yaw)	0