Robot Motion Planning
Activity Sheet

Please use this sheet to record your data from the Robot Motion activity session.

***Materials Needed**: Printed copy and pencil, or digital copy with drawing capabilities.*

Robot Motion Planning Introduction

If you had a robot, what would you want it to do?

_____

_____

_____

How would this robot move? What decisions would it need to make?

_____

_____

_____

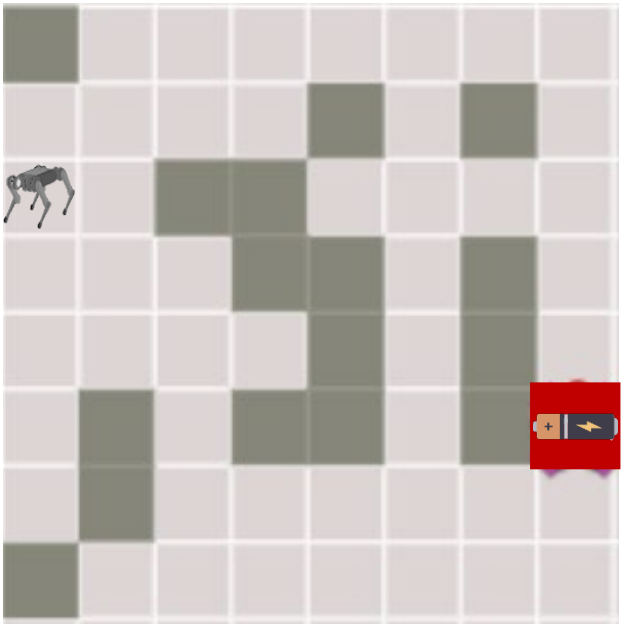
Robot Motion Planning Experiment 1 – Naïve Approach

Charlie the robot is hungry! Let's help Charlie find a path to some tasty batteries. Before we jump into the planning algorithm, let's see how well we can do just with our own intuition. Follow the steps below to plan some motions and measure how well you did.

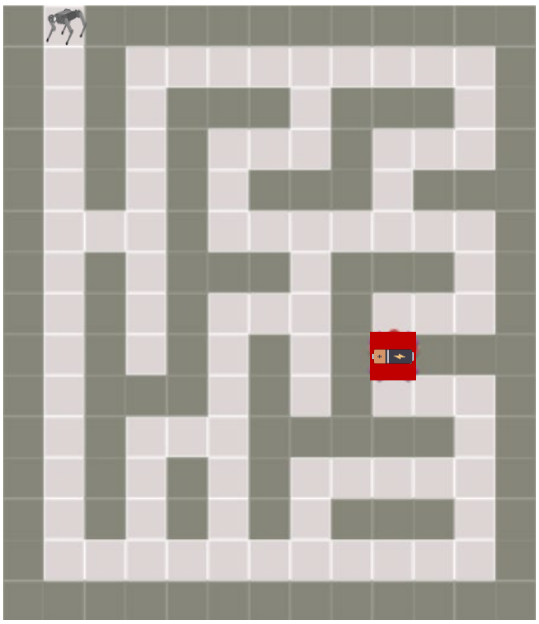
For each scene:

1. Start your path at the start location (*the cell with the robot*), either with a pencil or by drawing a line on your screen.
2. Without lifting your pencil/mouse, move up, down, left, or right to an adjacent cell.
   Remember that you cannot move into blocked (shaded) cells!
3. Every time you move, add one to your Total Move Cost (you can either keep track in your head or write it down).
4. Repeat steps 2-3 to move around the maze until you reach the goal (the cell with the battery). Try to do this in as few moves as possible, but never lift your pencil/mouse. It's okay if you get stuck and need to back up but keep adding those moves to your Total Move Cost.
5. Write down your Total Move Cost in the table below next to "Naïve Approach."

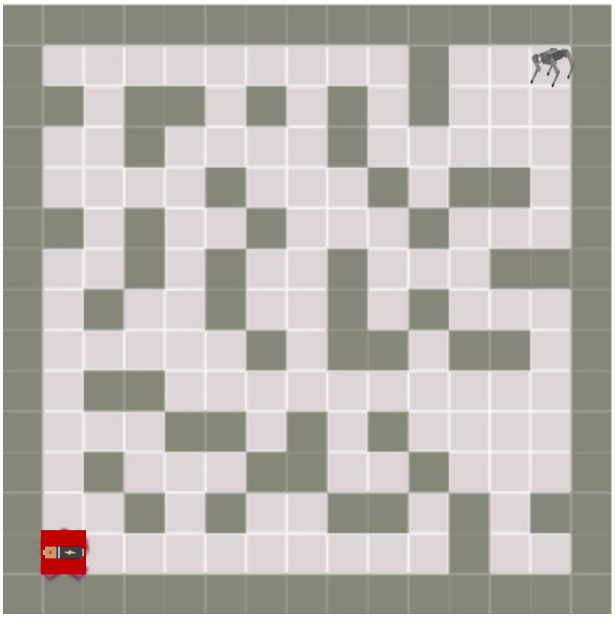
Scene 1:



Scene 2:



Scene 3:



	Scene 1 Total Move Cost	Scene 2 Total Move Cost	Scene 3 Total Move Cost
Naïve Approach			
Dijkstra's Algorithm			

How would you describe your general strategy (*think:* what decisions did you have to make)?

Did you get stuck? If so, how did you get unstuck?

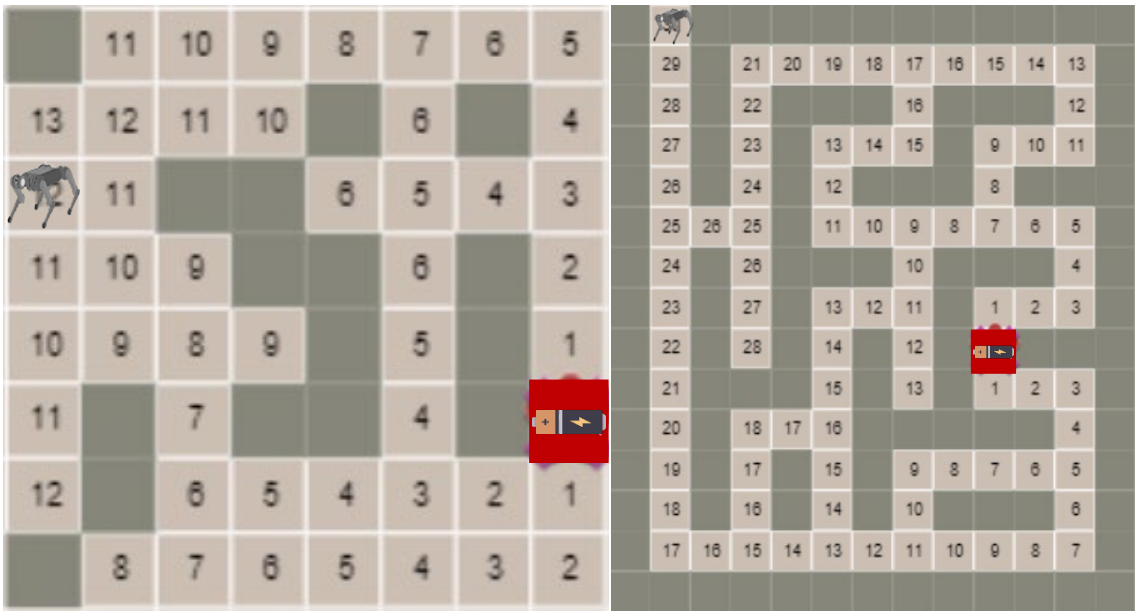
Robot Motion Planning Experiment 2 – Dijkstra's Algorithm (*with precomputed costs*)

Now that we have an idea of how to plan paths, let's use an algorithm to do it instead like we saw in the presentation. In this experiment we will use Dijkstra's Algorithm, which will give the shortest paths possible (Charlie is tired)! Follow the steps below to plan motions with Dijkstra's algorithm. For this experiment we've precomputed the costs of each cell to speed things up.

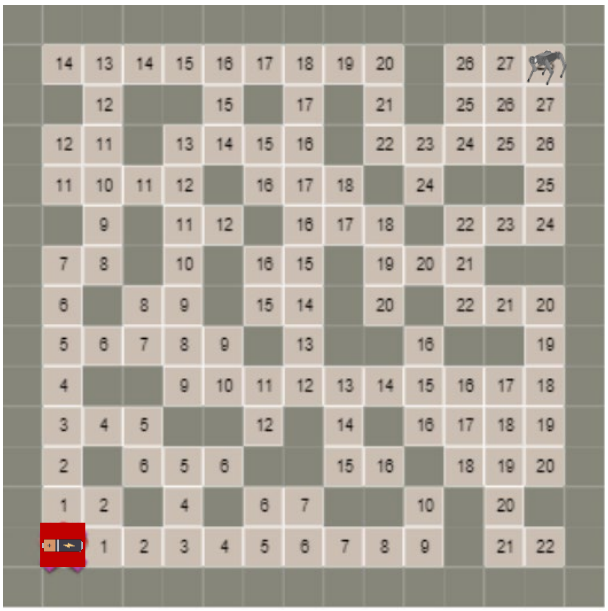
For each scene:

1. Repeat steps 1-4 from Experiment 1, but this time always move to the adjacent cell with the lowest cost.
2. Once you reach the goal, write down your Total Move Cost in the previous table next to "Dijkstra's Algorithm."

Scene 1: Scene 2:



Scene 3:



What decisions did you have to make this time?

_____

_____

_____

How do the performances of the naïve approach and Dijkstra's Algorithm compare?

Robot Motion Planning Experiment 3 – Dijkstra's Algorithm (*from scratch*)

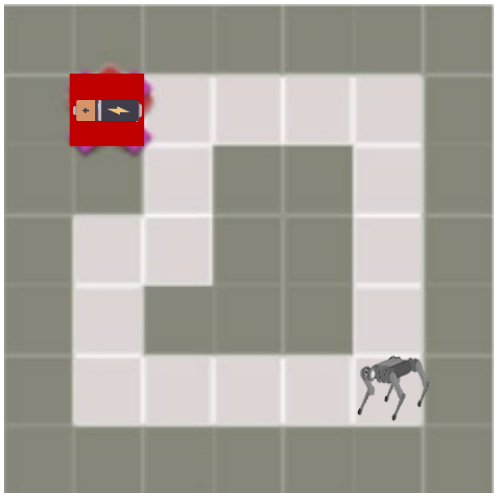
Now let's see if we can run Dijkstra's Algorithm on our own (both computing costs and finding the path). This may take a while on some of the trickier scenes – don't worry if you do not have time to complete them all.

For each scene:

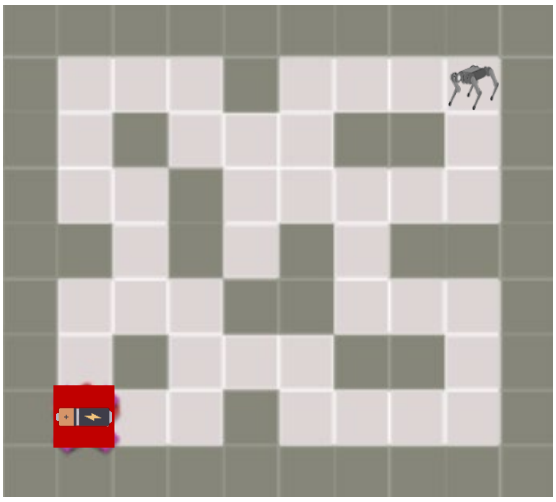
1. Compute the cost to reach the goal from each cell by following these steps:
    a. Write a zero in the goal cell.
    b. Find the cell with the lowest number that has no dot in the corner. We'll call this the "current cell."
    c. Fill in each neighbor of the current cell with the cost of the current cell plus one (since the move cost is one).
    d. If a neighbor already has a cost but the new cost would be lower, replace the old cost with this new one (otherwise leave it unchanged).
    e. Put a dot in the corner of the current cell.
    f. Repeat steps b – e until the start has a dot in its corner.
2. Follow the steps in Experiment 2 to trace the path from the start and the goal!

Scene 1:

Scene 2:



Scene 3:



Robot Motion Planning Experiment 4 – Path Following

Next we are going to put ourselves in Charlie's shoes to explore how and why these motion plans are used for guidance. We'll use a handy webpage for this section to view the world like robots and try to find some goals with and without paths (https://dhruvmisra.github.io/Pathfinding-Visualizer-ThreeJS/). You'll also need a stopwatch – if you do not have one handy, you can use this one (https://www.online-stopwatch.com/large-stopwatch/).

1. Let's get some obstacles: click on "Maze Algorithms" near the top and select "Random Maze." Get a good view of the map, the start location (green square), and goal location (red square). At this point you should not have any paths drawn on the map.
2. Find the goal without a path:
    a. Switch to first-person view: click on "First Person" near the right. You should be able to move around with arrow keys OR W-A-S-D keys and look around with the mouse (you might have to click on the screen to enable this after entering First Person mode). Hit escape to show your cursor after using the mouse to look around.
    b. Start your timer and try to find the goal! Once you find it (or if more than two minutes pass), record your time in the table below.
3. Find the goal with a path:
    a. Return to the top-down view: click the "Perspective" button
    b. Create the path: click "Visualize!" near the top. Watch as the optimal path is computed! This should help you find the goal.

c. Switch to first-person view: click on “First Person” near the right. Make sure you can look and move around as before.
d. Start your timer and try to find the goal (using the path for guidance)! Once you find it (or if more than two minutes pass), record your time in the table below.

4. Repeat the experiment for a different map:
    a. Generate new obstacles: click “Perspective” Mode, click “Clear Path” and “Clear Walls”, then select “Recursive Division” from the Maze Algorithm drop-down.
    b. Repeat steps 2&3 for this new scene.

	Random Maze Solve Time	**Recursive Division Solve Time**
Without Path		
With Path		

How did you do without a path? How did this compare to path following?

_____

_____

_____

Conclusions

Why would motion planning help a robot move?

_____

_____

_____

What would be different if we used algorithms like this on a real robot?

_____

_____

_____

_____

Bonus Scenes (try at your own risk!):