Confluence Log in Try Confluence for free

Robotic Lawnmower Homework - Orbitworks

By Sebastian Knak Bendtsen 2026-01-07

Please read through the full document before starting any work

Expectations

We expect you to spend roughly **4 hours** on this problem, but you can spend more time if you want to. Please see the Evaluation Criteria section for more information on what we are looking for.

Scenario

You work for *Robomower Inc.*, and your first product is an automated lawn mower. You can check out this video if you want to visualize what the end product might look like. Your manager has tasked you to start building a minimum viable product (MVP) for the robotic lawnmower simulator. The purpose of this simulator is to verify the output of paths from different algorithms for automated lawn mowing.

❓ Problem Statement

The Algorithms team is trying out many algorithms for automated lawnmowers. Since this is early stage, their system outputs a series of steps for the lawnmower. They want to use your simulator to simulate those steps and determine:

- Was all the grass cut?
- If all the grass was not cut, how much **uncut grass** remained?
- Did the lawnmower crash?

Lawn Definition

The company is looking for a relatively simple simulator:

- The lawn will be a **rectangular 2D grid** divided into **uniform-sized squares.**
- Each Square can be **one** of 3 types:
  - Uncut grass
  - Cut grass
  - Rock
- **The default type** of a square is **Uncut Grass**
- There is only **one** lawnmower, which will occupy exactly one square
- A lawnmower can move only along 4 directions:

POWERED BY

  - Left
  - Right

🫡 Rules

- A lawnmower will **always start** on the **top left corner [0,0]** of the grid
- The **top left corner [0,0]** is never a **Rock**
- A lawnmower can move exactly **one square** at a time
- If a lawnmower lands on a square with **Uncut grass**, it changes it to **Cut grass**
- If a lawnmower lands on **Cut grass**, nothing changes
- If a lawnmower lands on **Rock**, it **crashes** and can no longer do anything (termination)
- If a lawnmower goes off the grid, it **crashes** into the fence and can no longer do anything (termination)

📑 Requirements

Input

The inputs are to be provided by the Algorithms team. They have a strong preference for the GraphQL API. But, they are amenable to other common web-based APIs such as REST. You have the power to establish how this input is provided to work with your application. You have to come up with a design and communicate with the Algorithms team to get information about:

1. The dimensions of the lawn
2. Locations of grass and rock
3. The steps that the lawnmower takes from beginning to end
4. Anything else you deem necessary to build your application

Output

Again, the Algorithms team is flexible and wants to give you the power. But ultimately, they need to answer the three questions mentioned earlier. You have to come up with a design and communicate with the Algorithms team for your application to answer:

- Was all the grass cut?
- If all the grass was not cut, how much **uncut grass** remained?
- Did the lawnmower crash?

You are free to provide any additional information if you think it will be useful for the Algorithm team. However, they only require the above three pieces of information.

Environment

Since the team using your simulator is completely separate, they may or may not have all your dependencies installed. To simplify this, you are asked to isolate your environment using **Docker**. You can set this up any way you would like. You should:

- Have a `Dockerfile` that defines your environment
- Ability to build your environment in a Docker image using `docker build`
- Ability to run your image using `docker run`

Grid Conventions

The Algorithms team is not flexible on the lawn definition conventions. Below is a visual representation, with conventions on how the Algorithms team is using a **grid** representation for a lawn. Below is an example of a **5x4** grid, with square locations written in them:

[0,0]	[0,1]	[0,2]	**[0,3]**
[1,0]	[1,1]	[1,2]	[1,3]
[2,0]	[2,1]	[2,2]	[2,3]
[3,0]	[3,1]	[3,2]	[3,3]
[4,0]	[4,1]	[4,2]	**[4,3]**

Example Scenario

Input

Lawn: 3x2

Rocks: [1,1], [0,1]

Lawnmower path: [down, down, right]

By default, everywhere else is grass

Lawn Visualization

[0,0]	[0,1]
[1,0]	[1,1]
[2,0]	[2,1]

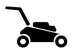
Simulation Steps

Below is a walkthrough of the simulation, given the input file above.

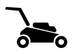
Initialization

[0,0]	[0,1]
[1,0]	[1,1]
[2,0]	[2,1]

Step 1 - down

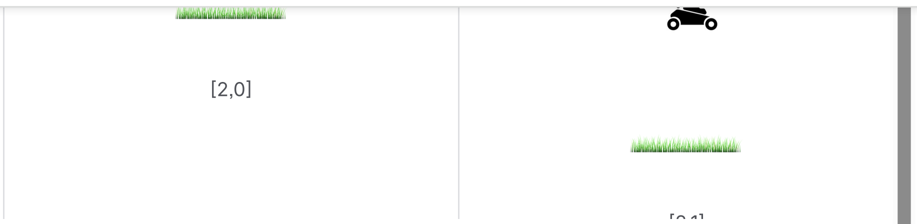
[0,0]	[0,1]
 [1,0]	[1,1]
[2,0]	[2,1]

Step 2 - down

[0,0]	[0,1]
[1,0]	[1,1]
 [2,0]	[2,1]

Step 3 - Right

[0,0]	[0,1]
[1,0]	[1,1]



Program Output

The output (in whatever format you design in your application) should clearly indicate that all the grass has been cut in this simulated lawn.

Other Scenarios

Taking the exact same example above, below are the expected program outputs, if the lawnmower path were different:

1. Crash Into Fence

If the input is:

Lawnmower path: [down, down, right, right]

The output (in whatever format you design in your application) should clearly communicate that the lawnmower has crashed into the fence

2. Crash Into Rock

If the input is:

Lawnmower path: [down, right, down]

The output (in whatever format you design in your application) should clearly communicate that the lawnmower has crashed into a rock

3. Incomplete

If the input is:

Lawnmower path: [down]

The output (in whatever format you design in your application) should clearly communicate that there are still **2** grid cells with **Uncut grass.**

Submission Requirements

You will submit your work through any cloud-based Git providers (GitHub, GitLab, etc.). Please add the following accounts to that **private** repository:

1. remy@loftorbital.com
2. paul.faugeras@loftorbital.com

Please also remember to send a link to your repository to the Recruiting Coordinator who sent you this, most commonly Sebastian.bendtsen@loftorbital.com

⚠ For a GitLab repository, please remember to add the accounts who will evaluate with the role of "reporter" so they can view the code, etc.

We will evaluate your submission on the criteria mentioned in the next section. But we will **heavily** rely on your documentation for:

1. Usage of your program
2. Design of your program
3. Inline logic

⚠ Since we offer some flexibility in how you set up your inputs and outputs, it is highly recommended that you provide a test "client" to use your service. This client can be modified as needed to run specific scenarios by us.

✅ Evaluation Criteria

We will be evaluating your submission based on the following criteria:

- Compliance with the requirements mentioned in this document
- Design and design documentation for breaking down the problem
- Ease of following your documentation on how to use your simulator application
- User experience of your simulator. The baseline assumption is that this is targeting someone comfortable with a command line interface and Linux
- General code readability
- Code Documentation
- Ability to work with **Python >=3.9**
- Ability to use and run your application within a **Docker** environment
- Ability to write **Python tests** and a way to easily run them
- Ease of scaling your application in the future

We are **NOT** looking for optimized software, 100% test coverage, or Python one-liners.

⭐ Extensions

If you want to go above and beyond, some extensions for you to consider are:

- Scale the application to serve multiple requests (100+) simultaneously on a single machine. Provide a way to deploy the application on a **Kubernetes** cluster where we have the ability to:
  - Scale **up** to have a single pod service with more concurrent requests
  - Scale **out** to service more concurrent requests
- Add a **CI/CD pipeline** that automates linting, type checking (with mypy), and runs the tests you wrote.

💡 Tips

- The problem is vague/open on purpose. Remember, your goal is to create an MVP
- Provide documentation and a reasonably simple application, keeping the user experience in mind
- Simplicity is underrated. How can someone outside of your team easily use your application? If you had another person join your team, how easily could they understand and contribute to your application?