Citrine Informatics Technical Challenge

Front End Engineer

Version: 6bc92357bbeeaf077ca8284b651c1d9287069f17

This challenge should take about five hours, both parts included. If you find yourself spending substantially more time than that, consider reducing the complexity of your implementation.

writing

Communication to both technical and non-technical coworkers is critical. For this part of the challenge, pretend that you are living in the hypothetical world described below:

You are an engineer working on the Data Ingestion pipeline. You see developers struggle to release their code, as they sometimes produce bugs or mess up the deployment scripts. You believe that a continuous integration and deployment system would dramatically improve the quality and cadence of engineering work, but the VP of Engineering is skeptical when you bring it up:

- We have too many customers to configure CI/CD for - at Citrine every customer gets their own AWS account
- We should release our code at the end of every sprint
- Each team uses different technologies
- We can’t afford to take engineers off of product work to work on it

Write an e-mail to our VP of Engineering to convince him that implementing a CI/CD workflow would be a worthy investment.

software

Pokepaths

Pokemon World is plagued by a strange malady. Pokemon are somehow unable to find their way home. Trees and rocks block our friends' way and prevent them from reaching their goals.

Fortunately, a wildly successful start-up, Citrine Informatics is developing a tool to help the wayward souls of these stumbling pokemon. They have already implemented an API which lets a wayward pokemon find instructions for how to reach their destination. Unfortunately, pokemon are technically laymen and aren't able to interact with the API.

Your task is to implement a graphical interface using the React framework for the tool which should enable the following functionality:

- A user can select the map size (an n x n square)
- A user can mark any number of squares as impassable
- A user can mark a start square, from which a route to the end square will be calculated
- A user can mark the end square
- A user can take some action which uses the provided API endpoint to calculate the path from the start square to the end square and see the results in some way

NOTE: Please use React to implement this UI

Feel free to use the tiles provided to build the visualizer.

API Details

The API is hosted here

The API endpoint accepts the following information:

- the side length of a square map expressed by a grid of square cells
- array of x,y coordinates indicating the cells on the map which are impassable by a pokemon
- the x,y coordinates at which the pokemon begins her journey
- the x,y coordinates she is attempting to reach

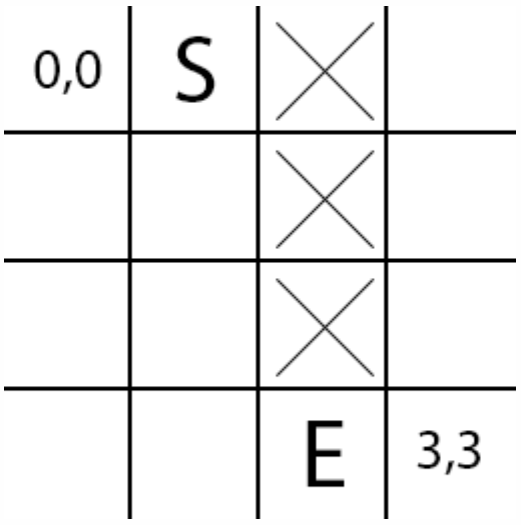
The API endpoint returns the shortest path that the pokemon should take to avoid the obstacles on the map and reach her goal. Each move is expressed by one of the following letters:

- "U" - move one cell upwards
- "D" - move one cell downwards
- "R" - move one cell right
- "L" - move one cell left

See the api-spec.yml file for details about the body of the requests to this API. You can visualize the specfile by pasting the spec document into the left side of the swagger API doc editor:

https://editor.swagger.io/

Example



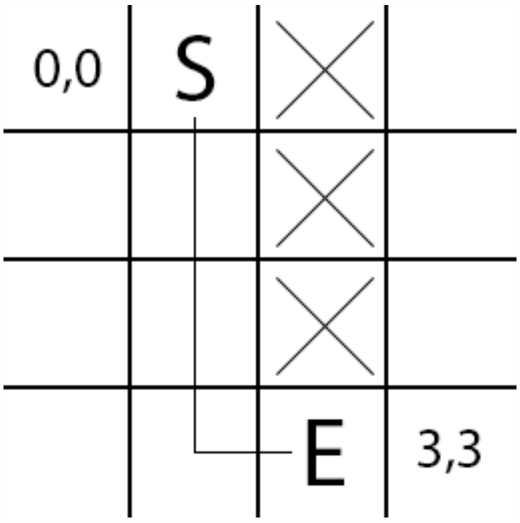
In this map, `0,0` and `3,3` simply indicate the coordinate system (positive x is toward the right and positive y is downward). Additionally:

- `S` is the starting location of the pokemon
- `E` is the target ending location for the pokemon
- `X` are impassable objects (cells which the pokemon may not occupy)

To express this map to the API, the following body would be used:

```
{
  "sideLength": 4,
  "impassables": [
    {
      "x": 2,
      "y": 0
    },
    {
      "x": 2,
      "y": 1
    },
    {
      "x": 2,
      "y": 2
    }
  ],
  "startingLoc": {
    "x": 1,
    "y": 0
  },
  "endingLoc": {
    "x": 2,
    "y": 3
  }
}
```

To express the following path as the solution:



The API might return the following response body:

```
{
  "moves": [
    'D',
    'D',
    'D',
    'R'
  ]
}
```