31 Sustainable Batteries

30 points

Introduction

Welcome to the R&D laboratory! We discovered a new material that could open the door to more sustainable batteries because it has a very special property: it can be charged with electrical current and its molecules accumulate energy, expanding the material. Then, when the material is under a constant pressure along its surface, the molecules start moving, producing an electrical current when they collide with other molecules. However, since this new material is not completely perfect, the molecules may collide with "obstacles" due to the material's impurity.

We want to analyze the trajectory of the molecules placed at different positions. Could you help us by creating a simulator for the molecules' movement? The simulator must fulfill the following requirements:

- The simulation must generate a sequence of 2D grids, each grid representing the position of the molecules and the obstacles at the same time sample (the duration of the simulation is specified as part of the input).
- The molecules will be placed at different positions within a 2D grid, each one with its own initial direction, and all of them will move at the same time with the same speed.
- The obstacles are placed at arbitrary positions within a 2D grid and they do not move during the simulation.
- When a molecule collides with an obstacle or the edges of the 2D grid, the molecule bounces and its direction is inverted accordingly. Therefore, a molecule cannot be placed outside the 2D grid or in the same position than an obstacle.
- When a molecule collides with another molecule, the collision must be marked in the 2D grid. Therefore, different molecules may be placed in the same position of the 2D grid. After collision the molecules won't change their trajectory.

Important! Obstacles may be next to other obstacles (in adjacent cells), but they will never form gaps of only 1 cell between obstacles (they must be at least 2 cells away of other obstacles).
Use the web-based tool in **Guides & Tools tab** that we implemented to see the output in a more visual way. It will help you to implement the solution :)

The input describes the dimensions of the 2D grid of the simulation, the number of frames that the simulator must provide and the initial state of the molecules and obstacles. More precisely:

- The 1st line is the word "rows" followed by a positive integer higher or equal than 3, which indicates the number of rows of the 2D grid.
- The 2nd line is the word "cols" followed by a positive integer higher or equal than 3, which indicates the number of columns of the 2D grid.
- The 3rd line is the word "frames" followed by a positive integer higher or equal than 2, which indicates the number of frames of the 2D grid that the simulation must compute.
- The rest of the lines describe the initial state of the molecules and obstacles, until no more lines are provided:
  - If the line starts with the word "molecule", it is followed by 5 numbers:

    1st ID of the molecule, which will be used to represent the molecule in the 2D grid. Different molecules can have the same ID, it is only used to have different types of molecules.

    2nd an integer in the range [0, rows) which indicates the initial row of the molecule.

    3rd an integer in the range [0, cols) which indicates the initial column of the molecule.

    4th an integer in the range [-1, 1] which indicates the initial direction in the "rows" axis.

    5th an integer in the range [-1, 1] which indicates the initial direction in the "cols" axis.
  - If the line starts with the word "obstacle", it is followed by 2 numbers:

    1st an integer in the range [0, rows) which indicates the row of the obstacle.

    2nd an integer in the range [0, cols) which indicates the column of the obstacle.

Output

The output must be a sequence of 2D matrices following a JSON format for arrays (see the output to clarify this format), providing as many matrices as indicated in the "frames" number from the input, being the first matrix the initial state of the 2D grid, with all the particles and obstacles in their initial position.

Each matrix must have size "rows x cols" and must be filled according to the following rules:

- The position of the molecules in the matrix must be filled with their ID.

- The position of the obstacles in the matrix must be filled with value -1.

- If 2 or more molecules are in the same position of the 2D grid, that cell must be filled with value -2.

- Any other cell without molecules or obstacles must be filled with value 0.

Example 1

Input

```
rows 8
cols 12
frames 6
molecule 1 0 0 1 1
molecule 2 3 4 1 1
molecule 1 7 8 -1 -1
obstacle 2 2
obstacle 2 8
obstacle 3 8
obstacle 5 3
obstacle 5 4
obstacle 5 5
```

Output

```
[
[
[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, -1, 0, 0, 0, 0, 0, -1, 0, 0, 0],
[0, 0, 0, 0, 2, 0, 0, 0, -1, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, -1, -1, -1, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0]
],
[
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, -1, 0, 0, 0, 0, 0, -1, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0],
[0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0],
[0, 0, 0, -1, -1, -1, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
],
[
[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, -1, 0, 0, 0, 0, 0, -1, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 2, 0, -1, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, -1, -1, -1, 1, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
],
[
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, -1, 0, 0, 0, 0, 2, -1, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
[0, 0, 0, -1, -1, -1, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
],
[
[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0],
[0, 0, -1, 0, 0, 0, 0, 0, -1, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, -1, -1, -1, 1, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
],
[
[0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0],
[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, -1, 0, 0, 0, 0, 0, -1, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, -1, -1, -1, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
]
]
```

Example 2

Input

```
rows 10
cols 23
frames 3
molecule 1 0 2 1 1
molecule 2 5 11 -1 1
molecule 3 6 17 1 1
obstacle 2 3
obstacle 2 4
obstacle 2 5
obstacle 2 10
obstacle 2 11
obstacle 2 12
obstacle 2 13
obstacle 2 16
obstacle 3 2
obstacle 3 6
obstacle 3 9
obstacle 3 16
obstacle 4 2
obstacle 4 6
obstacle 4 9
obstacle 4 16
obstacle 5 2
obstacle 5 3
obstacle 5 4
obstacle 5 5
obstacle 5 6
obstacle 5 9
obstacle 5 16
obstacle 6 2
obstacle 6 6
obstacle 6 9
obstacle 6 16
obstacle 7 2
obstacle 7 6
obstacle 7 10
obstacle 7 11
obstacle 7 12
obstacle 7 13
obstacle 7 16
obstacle 7 17
obstacle 7 18
obstacle 7 19
obstacle 7 20
```

Output

```
[
[
[0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, -1, -1, -1, 0, 0, 0, 0, -1, -1, -1, -1, 0, 0, -1, 0, 0, 0, 0, 0,
0],
[0, 0, -1, 0, 0, 0, -1, 0, 0, -1, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0],
[0, 0, -1, 0, 0, 0, -1, 0, 0, -1, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0],
[0, 0, -1, -1, -1, -1, -1, 0, 0, -1, 0, 2, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0],
[0, 0, -1, 0, 0, 0, -1, 0, 0, -1, 0, 0, 0, 0, 0, 0, -1, 3, 0, 0, 0, 0, 0],
[0, 0, -1, 0, 0, 0, -1, 0, 0, 0, -1, -1, -1, -1, 0, 0, -1, -1, -1, -1, -1, 0,
0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
],
[
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, -1, -1, -1, 0, 0, 0, 0, -1, -1, -1, -1, 0, 0, -1, 0, 0, 0, 0, 0,
0],
[0, 0, -1, 0, 0, 0, -1, 0, 0, -1, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0],
[0, 0, -1, 0, 0, 0, -1, 0, 0, -1, 0, 0, 2, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0],
[0, 0, -1, -1, -1, -1, -1, 0, 0, -1, 0, 0, 0, 0, 0, 0, -1, 0, 3, 0, 0, 0, 0],
[0, 0, -1, 0, 0, 0, -1, 0, 0, -1, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0],
[0, 0, -1, 0, 0, 0, -1, 0, 0, 0, -1, -1, -1, -1, 0, 0, -1, -1, -1, -1, -1, 0,
0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
],
[
[0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, -1, -1, -1, 0, 0, 0, 0, -1, -1, -1, -1, 0, 0, -1, 0, 0, 0, 0, 0,
0],
[0, 0, -1, 0, 0, 0, -1, 0, 0, -1, 0, 0, 0, 2, 0, 0, -1, 0, 0, 0, 0, 0, 0],
[0, 0, -1, 0, 0, 0, -1, 0, 0, -1, 0, 0, 0, 0, 0, 0, -1, 0, 0, 3, 0, 0, 0],
[0, 0, -1, -1, -1, -1, -1, 0, 0, -1, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0],
[0, 0, -1, 0, 0, 0, -1, 0, 0, -1, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0],
[0, 0, -1, 0, 0, 0, -1, 0, 0, 0, -1, -1, -1, -1, 0, 0, -1, -1, -1, -1, -1, 0,
0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
]
]
```