CS 245 - Lab 8 Fall, 2021

Lab 8 - Zombieverse

The goal of this assignment is to demonstrate your mastery of matrix, array and searching(BFS, DFS) . Download your code here.

Background

In one of the parallel universes, earth was infected with a zombie virus. In different regions, you are given a ***m*** x ***n*** **grid**, where each cell can have one of the three values:

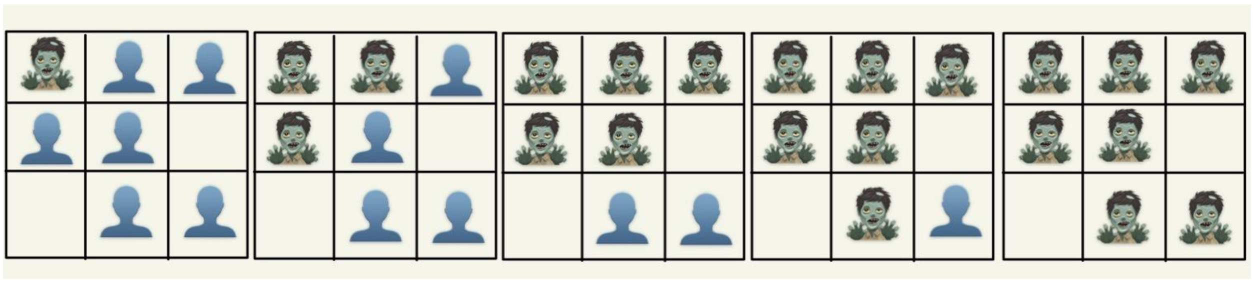
- 0 representing empty space
- 1 representing human
- 2 representing zombie

Every day, any human that is **4-directional adjacent** to the zombie will be infected.
Your job is to figure out whether any human survives or how many days it takes for all humans to be completely infected.
Return **-1** if any human survived, return **0** if there are no humans in the grid at the first place, else return total number of **days** it took to infect all humans.

Constraints:

- 1 <= m, n <= 10
- You are guaranteed to have values of 0, 1, 2 in the matrix
- m = grid.length, n = grid[i].length

Examples:

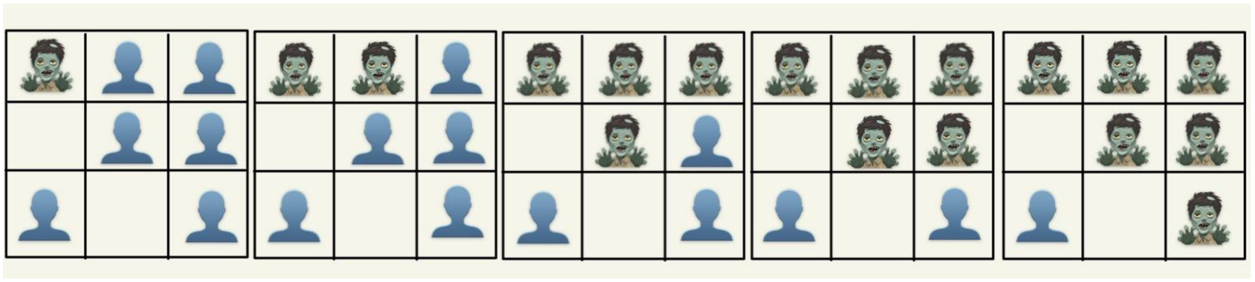


```
Day1          Day2          Day3          Day4          Day5
```

Input: `grid = [[2,1,1],`
`[1,1,0],`
`[0,1,1]]`

Output: `4`

Explanation: `It took 4 days to completely infect all humans in the grid`

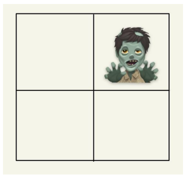


```
Day1          Day2          Day3          Day4          Day5
```

Input: `grid = [[2,1,1],`
`[0,1,1],`
`[1,0,1]]`

Output: `-1`

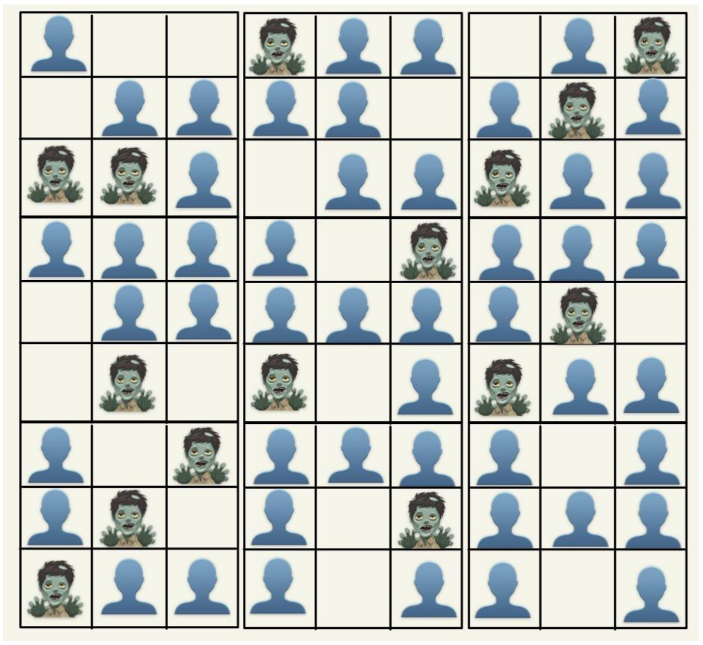
Explanation: `The human in row 3 column 1 was never infected`



Input: `[[0,0][0,2]]`

Output: `0`

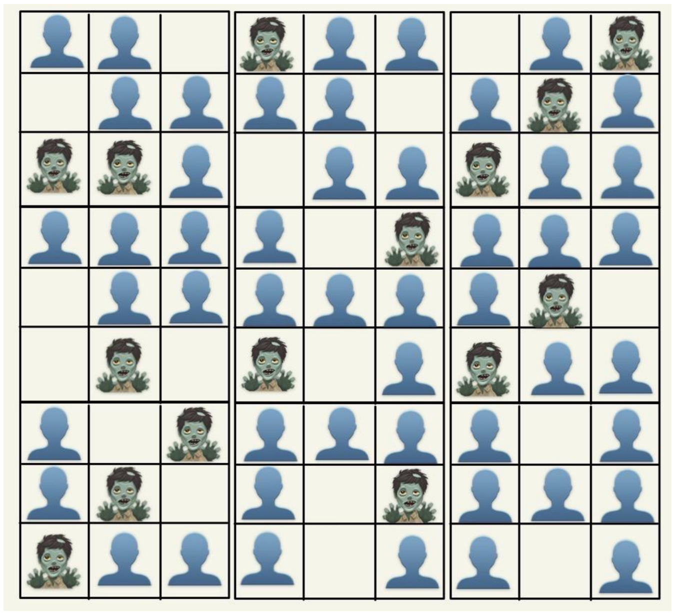
Explanation: `There are no human in day 1, so return 0`



Input: grid = [[1,0,0,2,1,1,0,1,2],
                [0,1,1,1,1,0,1,2,1],
                [2,2,1,0,1,1,2,1,1],
                [1,1,1,1,0,2,1,1,1],
                [0,1,1,1,1,1,1,2,0],
                [0,2,0,2,0,1,2,1,1],
                [1,0,2,1,1,1,1,0,1],
                [1,2,0,1,0,2,1,1,1],
                [2,1,1,1,0,1,1,0,1]]

Output: -1

Explanation: Human at row 1 column 1 was never infected



```
Input: grid = [[1,1,0,2,1,1,0,1,2],
               [0,1,1,1,1,0,1,2,1],
               [2,2,1,0,1,1,2,1,1],
               [1,1,1,1,0,2,1,1,1],
               [0,1,1,1,1,1,1,2,0],
               [0,2,0,2,0,1,2,1,1],
               [1,0,2,1,1,1,1,0,1],
               [1,2,0,1,0,2,1,1,1],
               [2,1,1,1,0,1,1,0,1]]
Output: 4

Input: grid =  [[1,1,1,1,1,1,1,1,1],
                [1,1,1,1,1,1,1,1,1],
                [1,1,1,1,1,1,1,1,1],
                [1,1,1,1,1,1,1,1,1],
                [1,1,1,1,2,1,1,1,1],
                [1,1,1,1,1,1,1,1,1],
                [1,1,1,1,1,1,1,1,1],
                [1,1,1,1,1,1,1,1,1],
                [1,1,1,1,1,1,1,1,1]]
Output: 8


Input: grid =  [[1,1,0,1,1,1,0,1,1],
               [0,1,1,1,1,0,1,1,1],
               [1,1,1,0,1,1,1,1,1],
               [1,1,1,1,0,1,1,1,1],
               [0,1,1,1,1,1,1,1,0],
               [0,1,0,1,0,1,1,1,1],
               [1,1,1,1,1,1,1,0,1],
               [1,0,0,1,0,1,1,1,1],
               [1,1,1,1,0,1,1,0,2]]
Output: 16
```

Requirements

- You must submit a single Java file containing the class named **Zombieverse**. You class must contain the function named "`infection`" with the following header:

```
public int infection(int[][] grid)
```

- Hints: (These hints are optional to follow)
    - 4-directional adjacent means (up, down, left, right) which corresponds to the grid[i][j] that grid[i+1][j], grid[i-1][j], grid[i][j+1], grid[i][j-1].
    - Track the number of humans (see if it has no humans at day 1), and track the position of the zombie and store it to some data structure, and perform **BFS**/DFS.

Submission

- Download the source code for your implementation from Github Classroom. If you don't have a Github account, create one via github.com > Sign up.
- Select your name from the list of names available, which will link your GitHub ID. If you do not see your name on the list, speak with the instructor or one of the teaching assistants.
- Submit the repository link on Canvas. You may also add any comments in a README file (text, PDF or Word document) to help the grader understand or execute your implementation.

Grading

Your grade for this assignment will be determined as follows:

- 75% = Implementation: your class implementations must run successfully with the source files and data provided. It must produce the expected results, a sample of which appears in the Implementation section above. Any deviation from the expected results results in 0 credit for implementation.
- 15% = Decomposition: in the eyes of the grader, the degree to which your solution follows the suggestions above or otherwise represents a reasonable object-oriented and procedural decomposition to this problem.
- 10% = Style: your code must be readable to the point of being self-documenting; in other words, it must have consistent comments describing the purpose of each class and the purpose of each function within a class. Names for variables and functions must be descriptive, and any code which is not straightforward or is in any way difficult to understand must be described with comments.