Câu hỏi 4

Chính xác

Chấm điểm của 2,00

Given a n*m grid where each cell in the grid can have a value of 0, 1 or 2, which has the following meaning:
1. Empty cell
2. This cell contains a fresh apple
3. This cell contains a rotten apple
After 1 second, the cell with rotten apple will rot all fresh apples in all the cells adjacent to it (i.e the cells (x+1, y), (x-1, y), (x, y+1), (x, y-1))
Determine the minimum time (in seconds) required to rot all apples. If this cannot be done, return -1.
Note: iostream, vector, and queue are already included.

Constraint:
1 <= n, m <= 500

Hint: Have you ever heard about breadth-first-search?

Example 1:
Input: grid = {{2,2,0,1}}
Output: -1
Explanation:
The grid is
2 2 0 1
The apple at (0, 3) cannot be rotten

Example 2:
Input: grid = {{0,1,2},{0,1,2},{2,1,1}}
Output: 1
Explanation:
The grid is
0 1 2
0 1 2
2 1 1
Apples at positions (0,2), (1,2), (2,0)
will rot apples at (0,1), (1,1), (2,2) and (2,1) after 1 second.

For example:

Test	Input	Result
`int rows, cols;` `cin >> rows >> cols;` `vector<vector<int>> grid(rows, vector<int>(cols));` `for(int i = 0; i < rows; i++) {` `for(int j = 0; j < cols; j++) cin >> grid[i][j];` `}` `cout << secondsToBeRotten(grid);`	1 4 2 2 0 1	-1
`int rows, cols;` `cin >> rows >> cols;` `vector<vector<int>> grid(rows, vector<int>(cols));` `for(int i = 0; i < rows; i++) {` `for(int j = 0; j < cols; j++) cin >> grid[i][j];` `}` `cout << secondsToBeRotten(grid);`	3 3 0 1 2 0 1 2 2 1 1	1

Answer: (penalty regime: 0 %)

Reset answer

```cpp
// iostream, vector and queue are included
// Hint: use breadth-first-search

int secondsToBeRotten(vector<vector<int>>& grid) {
    int n = grid.size();
    int m = grid[0].size();
    queue<pair<int, int>> q;
    int freshCount = 0;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            if (grid[i][j] == 2) {
                q.push({i, j});
            }
            else if (grid[i][j] == 1) {
                freshCount++;
            }
        }
    }
    int time = 0;
    while (!q.empty()) {
        int size = q.size();
        for (int i = 0; i < size; i++) {
            int x = q.front().first;
            int y = q.front().second;
            q.pop();
            if (x > 0 && grid[x - 1][y] == 1) {
                grid[x - 1][y] = 2;
                q.push({x - 1, y});
                freshCount--;
            }
            if (x < n - 1 && grid[x + 1][y] == 1) {
                grid[x + 1][y] = 2;
                q.push({x + 1, y});
                freshCount--;
            }
            if (y > 0 && grid[x][y - 1] == 1) {
                grid[x][y - 1] = 2;
                q.push({x, y - 1});
                freshCount--;
```

Precheck Kiểm tra

	Test	Input	Expected	Got	

✓	`int rows, cols;`
	`cin >> rows >> cols;`
	`vector<vector<int>> grid(rows, vector<int>(cols));`
	`for(int i = 0; i < rows; i++) {`
	`    for(int j = 0; j < cols; j++) cin >> grid[i][j];`
	`}`
	`cout << secondsToBeRotten(grid);`
		1 4
		2 2 0 1
			-1	-1	✓

✓	`int rows, cols;`
	`cin >> rows >> cols;`
	`vector<vector<int>> grid(rows, vector<int>(cols));`
	`for(int i = 0; i < rows; i++) {`
	`    for(int j = 0; j < cols; j++) cin >> grid[i][j];`
	`}`
	`cout << secondsToBeRotten(grid);`
		3 3
		0 1 2
		0 1 2
		2 1 1
			1	1	✓

Passed all tests! ✓