**Rotten Oranges :-**

Given a matrix of dimension **r**\***c** where each cell in the matrix can have values 0, 1 or 2 which has the following meaning:  
**0**: Empty cell  
**1** : Cells have fresh oranges  
**2** : Cells have rotten oranges

So, we have to determine what is the minimum time required to rot all oranges. A rotten orange at index [i,j] can **r**ot other fresh orange at indexes [i-1,j], [i+1,j], [i,j-1], [i,j+1] (**up**, **down**, **left** and **right**) in unit time. If it is impossible to rot every orange then simply return -1.

**Input:**  
The first line of input contains an integer T denoting the number of test cases. Each test case contains two integers r and c, where r is the number of rows and c is the number of columns in the array a[]. Next line contains space separated r\*c elements each in the array a[].

**Output:**  
Print an integer which denotes the minimum time taken to rot all the oranges (-1 if it is impossible).

**Constraints:**  
1 <= T <= 100  
1 <= r <= 100  
1 <= c <= 100  
0 <= a[i] <= 2

**Example:  
Input:**  
2  
3 5  
2 1 0 2 1 1 0 1 2 1 1 0 0 2 1  
3 5  
2 1 0 2 1 0 0 1 2 1 1 0 0 2 1  
  
**Output:**  
2  
-1

**Explanation:  
Testcase 1:**  
2 | 1 | 0 | 2 | 1  
1 | 0 | 1 | 2 | 1  
1 | 0 | 0 | 2 | 1

Oranges at positions {0,0}, {0, 3}, {1, 3} and {2, 3} will rot oranges at {0, 1}, {1, 0}, {0, 4}, {1, 2}, {1, 4}, {2, 4} during 1st unit time. And, during 2nd unit time, orange at {1, 0} got rotten and will rot orange at {2, 0}. Hence, total 2 unit of time is required to rot all oranges.