Description | Hints | Submissions | Discussions | Notes

The Witcher II

[illegible] Difficulty | 2 sec Time Limit | 256000KB Memory | 100 Score

80/80 XP | 30/30

Description

Gerald of Rivia also known as *The Witcher* wants to reach the end of the dungeon. The dungeon consists of $n \times m$ rooms laid out in a $2D$ grid. *Gerald* was initially positioned in the top-left room and must fight his way through, to the bottom-right room of the dungeon, where the exit is located.

Gerald has an initial health point represented by a positive integer. If at any point his health point drops to 0 or below, he dies immediately.

Some of the rooms are guarded by demons (represented by negative integers), so *Gerald* loses health upon entering these rooms; other rooms are either empty (represented as 0) or contain magic orbs that increase Gerald's health (represented by positive integers).

To reach the exit as quickly as possible, Gerald decides to move only rightward or downward in each step.

Return Gerald's minimum initial health so that he can exit the dungeon.

Note that any room can contain threats or power-ups, even the first room Gerald enters and the bottom-right room where the exit is located.

Input Format

First-line contains T - the number of test cases.
First-line of each test case contains 2 integers n and m.
Each of the next n lines of each test case contains m integers, denoting the values of $n \times m$ dungeon's rooms.

Output Format

For each test case, output Gerald's minimum initial health so that he can exit the dungeon.

C++14 | 00:00:00 | 12 px

```cpp
#include <bits/stdc++.h>
using namespace std;

#define ll int64_t
#define endl '\n'

int n, m;
vector<vector<int>> dungeon, dp;

int rec(int i, int j) {
    // If the position is out of bounds, return a large value indicating an invalid path
    if (i >= n || j >= m) return 1e9;

    // Base case: When we reach the bottom-right corner of the dungeon
    if (i == n - 1 && j == m - 1) {
        return max(1, 1 - dungeon[i][j]);
    }

    // If the subproblem has already been solved, return the stored result
    if (dp[i][j] != -1) return dp[i][j];

    // Calculate the minimum health needed from the right and down cells
    int right = rec(i, j + 1);
```

Sample Tests | Manual Tests

Test Case 1

ACCEPTED

Input

```
2
3 3
```

Console | Run on Sample