Problem 329: Longest Increasing Path in a Matrix

Problem Information

Difficulty: Hard
Acceptance Rate: 56.00%
Paid Only: No
Tags: Array, Dynamic Programming, Depth-First Search, Breadth-First Search, Graph, Topological Sort, Memoization, Matrix

Problem Description

Given an `m x n` integers `matrix`, return _the length of the longest increasing path in_`matrix`.

From each cell, you can either move in four directions: left, right, up, or down. You **may not** move **diagonally** or move **outside the boundary** (i.e., wrap-around is not allowed).

Example 1:

![](https://assets.leetcode.com/uploads/2021/01/05/grid1.jpg)

Input: matrix = [[9,9,4],[6,6,8],[2,1,1]] **Output:** 4 **Explanation:** The longest increasing path is [1, 2, 6, 9].

Example 2:

![](https://assets.leetcode.com/uploads/2021/01/27/tmp-grid.jpg)

Input: matrix = [[3,4,5],[3,2,6],[2,2,1]] **Output:** 4 **Explanation:** The longest increasing path is [3, 4, 5, 6]. Moving diagonally is not allowed.

Example 3:

Input: matrix = [[1]] **Output:** 1

Constraints:

* `m == matrix.length` * `n == matrix[i].length` * `1 <= m, n <= 200` * `0 <= matrix[i][j] <= 231 - 1`

Code Snippets

C++:

```
class Solution {
public:
int longestIncreasingPath(vector<vector<int>>& matrix) {

}
};
```

Java:

```
class Solution {
public int longestIncreasingPath(int[][] matrix) {

}
}
```

Python3:

```
class Solution:
def longestIncreasingPath(self, matrix: List[List[int]]) -> int:
```