675. Cut Off Trees for Golf Event

Hard | Topics | Companies

You are asked to cut off all the trees in a forest for a golf event. The forest is represented as an `m x n` matrix. In this matrix:

- `0` means the cell cannot be walked through.
- `1` represents an empty cell that can be walked through.
- A number greater than `1` represents a tree in a cell that can be walked through, and this number is the tree's height.

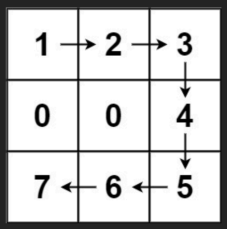
In one step, you can walk in any of the four directions: north, east, south, and west. If you are standing in a cell with a tree, you can choose whether to cut it off.

You must cut off the trees in order from shortest to tallest. When you cut off a tree, the value at its cell becomes `1` (an empty cell).

Starting from the point `(0, 0)`, return *the minimum steps you need to walk to cut off all the trees*. If you cannot cut off all the trees, return `-1`.

Note: The input is generated such that no two trees have the same height, and there is at least one tree needs to be cut off.

Example 1:



Input: forest = [[1,2,3],[0,0,4],[7,6,5]]
Output: 6
Explanation: Following the path above allows you to cut off the trees from shortest to tallest in 6 steps.

```python
class Solution:
    def cutOffTree(self, forest: List[List[int]]) -> int:
        if not forest or not forest[0]: return -1

        m, n = len(forest), len(forest[0])
        #collect trees
        trees = [(forest[i][j], i, j) for i in range (m) for j in range (n) if forest[i][j] > 1]
        trees.sort()
        #BFS
        def bfs(sx, sy, tx, ty):
            if sx == tx and sy == ty: return 0

            visited = [[False] * n for _ in range(m)]
            q = deque([(sx, sy, 0)])
            visited[sx][sy] = True
            directions = [(0, 1), (1, 0), (0, -1), (-1, 0)]
            while q:
                r, c, d = q.popleft()
                for dx, dy in directions:
                    nx, ny = dx + r, dy + c
                    if 0 <= nx < m and 0 <= ny < n and not visited[nx][ny] and forest[nx][ny]:
                        if nx == tx and ny == ty: return d+1
                        visited[nx][ny] = True
                        q.append((nx, ny, d+1))
            return -1

        res = 0
        sx, sy = 0, 0
        for h, i, j in trees:
            temp = bfs(sx, sy, i, j)
            if temp == -1:
                return -1

            res += temp
            sx, sy = i, j
            forest[i][j] = 1

        return res

```

collect trees

sort

BFS

must cut trees in order