Problem Link:

<https://leetcode.com/problems/trapping-rain-water-ii/description/?envType=daily-question&envId=2025-10-03>

Solution:

class Solution {

public:

int trapRainWater(vector<vector<int>>& heightMap) {

int m = heightMap.size();

int n = heightMap[0].size();

if(m <= 2 || n <= 2)

return 0;

priority\_queue<pair<int, int>, vector<pair<int, int>>, greater<>> h;

vector<vector<bool>> v(m, vector<bool>(n, false));

for(int i = 0; i < m; ++i)

{

h.emplace(heightMap[i][0], i \* n + 0);

h.emplace(heightMap[i][n - 1], i \* n + n - 1);

v[i][0] = v[i][n - 1] = true;

}

for(int j = 1; j < n - 1; ++j)

{

h.emplace(heightMap[0][j], j);

h.emplace(heightMap[m - 1][j], (m - 1) \* n + j);

v[0][j] = v[m - 1][j] = true;

}

vector<pair<int, int>> d = {{-1,0},{1,0},{0,-1},{0,1}};

int w = 0;

while(!h.empty())

{

auto [height, pos] = h.top(); h.pop();

int x = pos / n, y = pos % n;

for(auto [dx, dy] : d)

{

int nx = x + dx, ny = y + dy;

if(nx < 0 || nx >= m || ny < 0 || ny >= n || v[nx][ny])

continue;

v[nx][ny] = true;

int nh = heightMap[nx][ny];

w += max(0, height - nh);

h.emplace(max(height, nh), nx \* n + ny);

}

}

return w;

}

};