

Solution 1	Solution 2	Solution 3	Solution 4
<pre>1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved. 2 3 using namespace std; 4 5 ▾ struct InfoMatrixItem { 6 int numZeroesBelow; 7 int numZeroesRight; 8 }; 9 10 bool hasSquareOfZeroes(11 vector<vector<InfoMatrixItem>>& infoMatrix, 12 int r1, 13 int c1, 14 int r2, 15 int c2, 16 unordered_map<string, bool> &cache 17); 18 bool isSquareOfZeroes(19 vector<vector<InfoMatrixItem>>& infoMatrix, 20 int r1, 21 int c1, 22 int r2, 23 int c2 24); 25 vector<vector<InfoMatrixItem>> preComputedNumOfZeroes(vector<vector<int>> matrix); 26 27 // O(n^3) time O(n^3) space - where n is the height and width of the matrix 28 ▾ bool squareOfZeroes(vector<vector<int>> matrix) { 29 vector<vector<InfoMatrixItem>> infoMatrix = preComputedNumOfZeroes(matrix); 30 int lastIdx = matrix.size() - 1; 31 unordered_map<string, bool> cache; 32 return hasSquareOfZeroes(infoMatrix, 0, 0, lastIdx, lastIdx, cache); 33 } 34 35 // r1 is the top row, c1 is the left column 36 // r2 is the bottom row, c2 is the right column 37 bool hasSquareOfZeroes(38 vector<vector<InfoMatrixItem>>& infoMatrix, 39 int r1, 40 int c1, 41 int r2, 42 int c2, 43 unordered_map<string, bool> &cache 44 ▾) { 45 if (r1 >= r2 c1 >= c2) return false; 46 47 string key = to_string(r1) + '-' + to_string(c1) + '-' + to_string(r2) + '-' + to_string(c2); 48 if (cache.find(key) != cache.end()) return cache[key]; 49 50 cache[key] = 51 isSquareOfZeroes(infoMatrix, r1, c1, r2, c2) 52 hasSquareOfZeroes(infoMatrix, r1 + 1, c1 + 1, r2 - 1, c2 - 1, cache) 53 hasSquareOfZeroes(infoMatrix, r1, c1 + 1, r2 - 1, c2, cache) 54 hasSquareOfZeroes(infoMatrix, r1 + 1, c1, r2, c2 - 1, cache) 55 hasSquareOfZeroes(infoMatrix, r1 + 1, c1 + 1, r2, c2, cache) 56 hasSquareOfZeroes(infoMatrix, r1, c1, r2 - 1, c2 - 1, cache); 57 58 return cache[key]; 59 } 60 61 // r1 is the top row, c1 is the left column 62 // r2 is the bottom row, c2 is the right column 63 bool isSquareOfZeroes(64 vector<vector<InfoMatrixItem>>& infoMatrix, 65 int r1, 66 int c1, 67 int r2, 68 int c2 69 ▾) { 70 int squareLength = c2 - c1 + 1; 71 bool hasTopBorder = infoMatrix[r1][c1].numZeroesRight >= squareLength; 72 bool hasLeftBorder = infoMatrix[r1][c1].numZeroesBelow >= squareLength; 73 bool hasBottomBorder = infoMatrix[r2][c1].numZeroesRight >= squareLength; 74 bool hasRightBorder = infoMatrix[r1][c2].numZeroesBelow >= squareLength; 75 return hasTopBorder && hasLeftBorder && hasBottomBorder && hasRightBorder; 76 } 77 78 ▾ vector<vector<InfoMatrixItem>> preComputedNumOfZeroes(vector<vector<int>> matrix) { 79 vector<vector<InfoMatrixItem>> infoMatrix; 80 ▾ for (int i = 0; i < matrix.size(); i++) { 81 vector<InfoMatrixItem> inner; 82 ▾ for (int j = 0; j < matrix[i].size(); j++) { 83 int numZeroes = matrix[i][j] == 0 ? 1 : 0;</pre>			

```
84         inner.push_back(InfoMatrixItem {numZeroes, numZeroes});
85     }
86     infoMatrix.push_back(inner);
87 }
88
89     int lastIdx = matrix.size() - 1;
90     for (int row = lastIdx; row >= 0; row--) {
91         for (int col = lastIdx; col >= 0; col--) {
92             if (matrix[row][col] == 1) continue;
93             if (row < lastIdx) {
94                 infoMatrix[row][col].numZeroesBelow += infoMatrix[row + 1][col].numZeroesBelow;
95             }
96             if (col < lastIdx) {
97                 infoMatrix[row][col].numZeroesRight += infoMatrix[row][col + 1].numZeroesRight;
98             }
99         }
100     }
101
102     return infoMatrix;
103 }
```