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

```
84    public class InfoMatrixItem {
85      public int numZeroesBelow;
86      public int numZeroesRight;
87
88      public InfoMatrixItem(int numZeroesBelow, int numZeroesRight) {
89        this.numZeroesBelow = numZeroesBelow;
90        this.numZeroesRight = numZeroesRight;
91      }
92    }
93  }
94
```