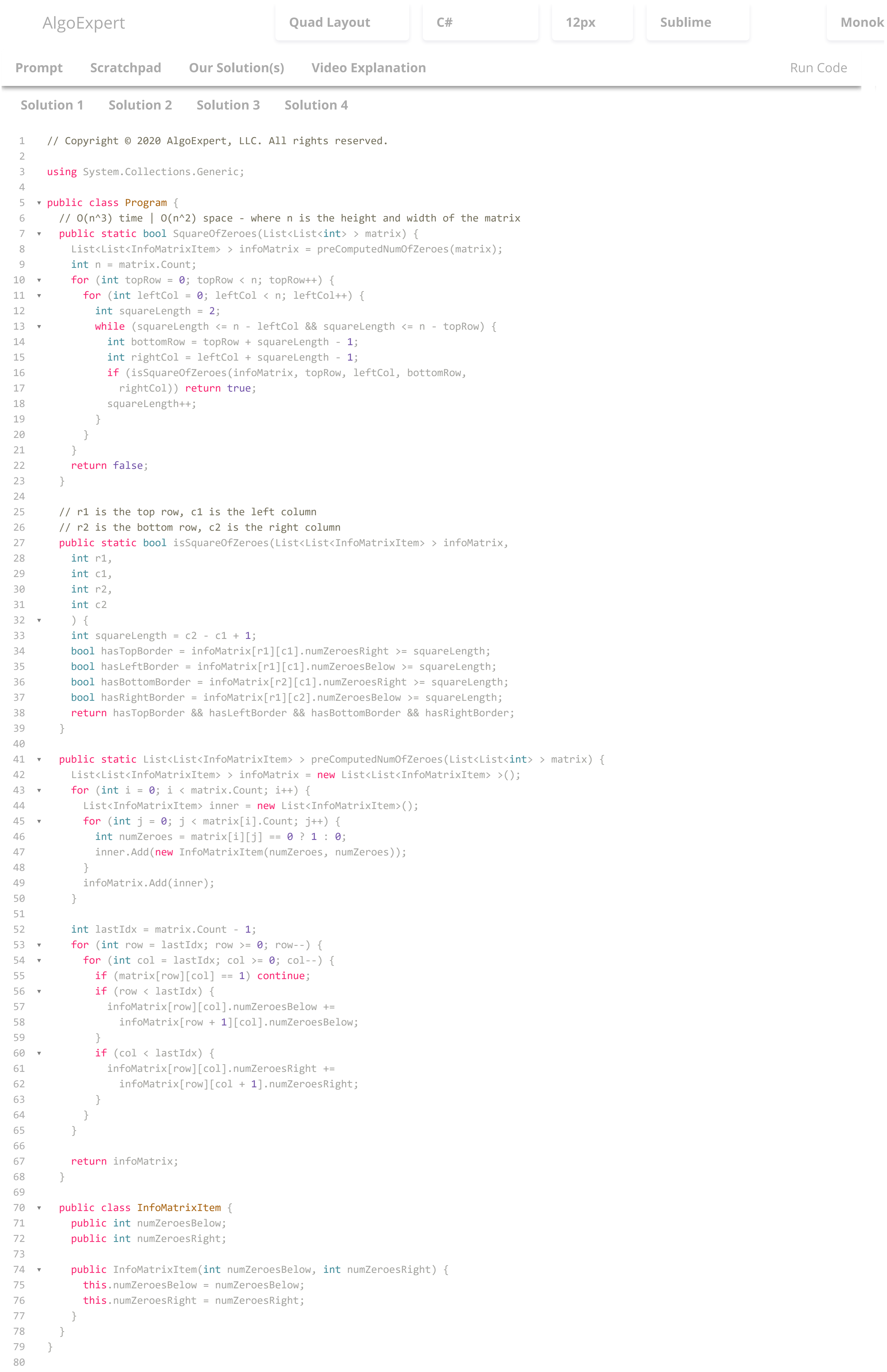

```csharp
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

using System.Collections.Generic;

public class Program {
  // O(n^3) time | O(n^2) space - where n is the height and width of the matrix
  public static bool SquareOfZeroes(List<List<int> > matrix) {
    List<List<InfoMatrixItem> > infoMatrix = preComputedNumOfZeroes(matrix);
    int n = matrix.Count;
    for (int topRow = 0; topRow < n; topRow++) {
      for (int leftCol = 0; leftCol < n; leftCol++) {
        int squareLength = 2;
        while (squareLength <= n - leftCol && squareLength <= n - topRow) {
          int bottomRow = topRow + squareLength - 1;
          int rightCol = leftCol + squareLength - 1;
          if (isSquareOfZeroes(infoMatrix, topRow, leftCol, bottomRow,
            rightCol)) return true;
          squareLength++;
        }
      }
    }
    return false;
  }

  // r1 is the top row, c1 is the left column
  // r2 is the bottom row, c2 is the right column
  public static bool isSquareOfZeroes(List<List<InfoMatrixItem> > infoMatrix,
    int r1,
    int c1,
    int r2,
    int c2
    ) {
    int squareLength = c2 - c1 + 1;
    bool hasTopBorder = infoMatrix[r1][c1].numZeroesRight >= squareLength;
    bool hasLeftBorder = infoMatrix[r1][c1].numZeroesBelow >= squareLength;
    bool hasBottomBorder = infoMatrix[r2][c1].numZeroesRight >= squareLength;
    bool hasRightBorder = infoMatrix[r1][c2].numZeroesBelow >= squareLength;
    return hasTopBorder && hasLeftBorder && hasBottomBorder && hasRightBorder;
  }

  public static List<List<InfoMatrixItem> > preComputedNumOfZeroes(List<List<int> > matrix) {
    List<List<InfoMatrixItem> > infoMatrix = new List<List<InfoMatrixItem> >();
    for (int i = 0; i < matrix.Count; i++) {
      List<InfoMatrixItem> inner = new List<InfoMatrixItem>();
      for (int j = 0; j < matrix[i].Count; j++) {
        int numZeroes = matrix[i][j] == 0 ? 1 : 0;
        inner.Add(new InfoMatrixItem(numZeroes, numZeroes));
      }
      infoMatrix.Add(inner);
    }

    int lastIdx = matrix.Count - 1;
    for (int row = lastIdx; row >= 0; row--) {
      for (int col = lastIdx; col >= 0; col--) {
        if (matrix[row][col] == 1) continue;
        if (row < lastIdx) {
          infoMatrix[row][col].numZeroesBelow +=
            infoMatrix[row + 1][col].numZeroesBelow;
        }
        if (col < lastIdx) {
          infoMatrix[row][col].numZeroesRight +=
            infoMatrix[row][col + 1].numZeroesRight;
        }
      }
    }

    return infoMatrix;
  }

  public class InfoMatrixItem {
    public int numZeroesBelow;
    public int numZeroesRight;

    public InfoMatrixItem(int numZeroesBelow, int numZeroesRight) {
      this.numZeroesBelow = numZeroesBelow;
      this.numZeroesRight = numZeroesRight;
    }
  }
}
```