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

import java.util.*;

class Program {
  // O(n^3) time | O(n^3) space - where n is the height and width of the matrix
  public static boolean squareOfZeroes(List<List<Integer>> matrix) {
    List<List<InfoMatrixItem>> infoMatrix = preComputedNumOfZeroes(matrix);
    int lastIdx = matrix.size() - 1;
    Map<String, Boolean> cache = new HashMap<String, Boolean>();
    return hasSquareOfZeroes(infoMatrix, 0, 0, lastIdx, lastIdx, cache);
  }

  // r1 is the top row, c1 is the left column
  // r2 is the bottom row, c2 is the right column
  public static boolean hasSquareOfZeroes(
    List<List<InfoMatrixItem>> matrix,
    int r1,
    int c1,
    int r2,
    int c2,
    Map<String, Boolean> cache
  ) {
    if (r1 >= r2 || c1 >= c2) return false;

    String key = String.valueOf(r1) + '-' + String.valueOf(c1) + '-' + String.valueOf(r2) + '-' + String.valueOf(c2);
    if (cache.containsKey(key)) return cache.get(key);

    cache.put(key,
      isSquareOfZeroes(matrix, r1, c1, r2, c2) ||
      hasSquareOfZeroes(matrix, r1 + 1, c1 + 1, r2 - 1, c2 - 1, cache) ||
      hasSquareOfZeroes(matrix, r1, c1 + 1, r2 - 1, c2, cache) ||
      hasSquareOfZeroes(matrix, r1 + 1, c1, r2, c2 - 1, cache) ||
      hasSquareOfZeroes(matrix, r1 + 1, c1 + 1, r2, c2, cache) ||
      hasSquareOfZeroes(matrix, r1, c1, r2 - 1, c2 - 1, cache));

    return cache.get(key);
  }

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

  public static List<List<InfoMatrixItem>> preComputedNumOfZeroes(List<List<Integer>> matrix) {
    List<List<InfoMatrixItem>> infoMatrix = new ArrayList<List<InfoMatrixItem>>();
    for (int i = 0; i < matrix.size(); i++) {
      List<InfoMatrixItem> inner = new ArrayList<InfoMatrixItem>();
      for (int j = 0; j < matrix.get(i).size(); j++) {
        int numZeroes = matrix.get(i).get(j) == 0 ? 1 : 0;
        inner.add(new InfoMatrixItem(numZeroes, numZeroes));
      }
      infoMatrix.add(inner);
    }

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

    return infoMatrix;
  }

  static class InfoMatrixItem {
```

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