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

import java.util.*;

class Program {
  // O(n^4) time | O(n^3) space - where n is the height and width of the matrix
  public static boolean squareOfZeroes(List<List<Integer>> matrix) {
    int lastIdx = matrix.size() - 1;
    Map<String, Boolean> cache = new HashMap<String, Boolean>();
    return hasSquareOfZeroes(matrix, 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<Integer>> 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<Integer>> matrix,
    int r1,
    int c1,
    int r2,
    int c2
  ) {
    for (int row = r1; row < r2 + 1; row++) {
      if (matrix.get(row).get(c1) != 0 || matrix.get(row).get(c2) != 0) return false;
    }
    for (int col = c1; col < c2 + 1; col++) {
      if (matrix.get(r1).get(col) != 0 || matrix.get(r2).get(col) != 0) return false;
    }
    return true;
  }
}
```