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

package main

import "fmt"

// O(n^4) time | O(n^3) space - where n is the height and width of the matrix
func SquareOfZeroes(matrix [][]int) bool {
  lastIdx := len(matrix) - 1
  return hasSquareOfZeroes(matrix, 0, 0, lastIdx, lastIdx, map[string]bool{})
}

// r1 is the top row, c1 is the left column
// r2 is the bottom row, c2 is the right column
func hasSquareOfZeroes(matrix [][]int, r1, c1, r2, c2 int, cache map[string]bool) bool {
  if r1 >= r2 || c1 >= c2 {
    return false
  }

  key := fmt.Sprintf("%d-%d-%d-%d", r1, c1, r2, c2)
  if out, found := cache[key]; found {
    return out
  }

  cache[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[key]
}

// r1 is the top row, c1 is the left column
// r2 is the bottom row, c2 is the right column
func isSquareOfZeroes(matrix [][]int, r1, c1, r2, c2 int) bool {
  for row := r1; row < r2+1; row++ {
    if matrix[row][c1] != 0 || matrix[row][c2] != 0 {
      return false
    }
  }

  for col := c1; col < c2+1; col++ {
    if matrix[r1][col] != 0 || matrix[r2][col] != 0 {
      return false
    }
  }
  return true
}

```