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

using namespace std;

bool hasSquareOfZeroes(
  vector<vector<int>>& matrix,
  int r1,
  int c1,
  int r2,
  int c2,
  unordered_map<string, bool> &cache
);
bool isSquareOfZeroes(vector<vector<int>>& matrix, int r1, int c1, int r2, int c2);

// O(n^4) time | O(n^3) space - where n is the height and width of the matrix
bool squareOfZeroes(vector<vector<int>> matrix) {
  int lastIdx = matrix.size() - 1;
  unordered_map<string, bool> cache;
  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
bool hasSquareOfZeroes(
  vector<vector<int>>& matrix,
  int r1,
  int c1,
  int r2,
  int c2,
  unordered_map<string, bool> &cache
) {
  if (r1 >= r2 || c1 >= c2) return false;

  string key = to_string(r1) + '-' + to_string(c1) + '-' + to_string(r2) + '-' + to_string(c2);
  if (cache.find(key) != cache.end()) return cache[key];

  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
bool isSquareOfZeroes(vector<vector<int>>& matrix, int r1, int c1, int r2, int c2) {
  for (int row = r1; row < r2 + 1; row++) {
    if (matrix[row][c1] != 0 || matrix[row][c2] != 0) return false;
  }
  for (int col = c1; col < c2 + 1; col++) {
    if (matrix[r1][col] != 0 || matrix[r2][col] != 0) return false;
  }
  return true;
}
```