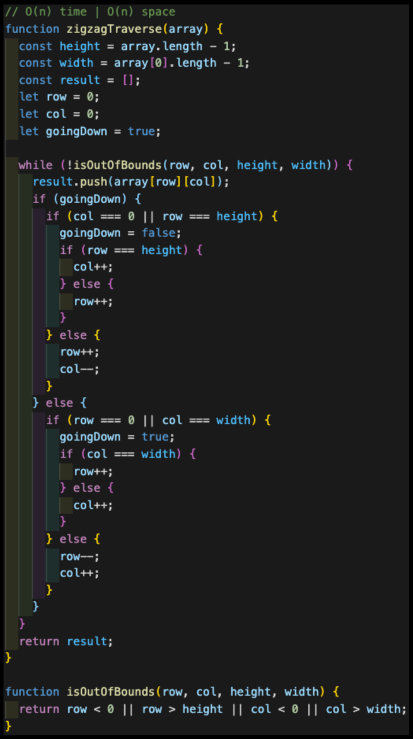
Input: n x m 2D matrix (can be square shaped when n == m)

```
array = [
  [1   3   4   10],
  [2   5   9   11],
  [6   8   12  15],
  [7   13  14  16],
]
```

Output: 1D array of all the element's in zig zag order

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]

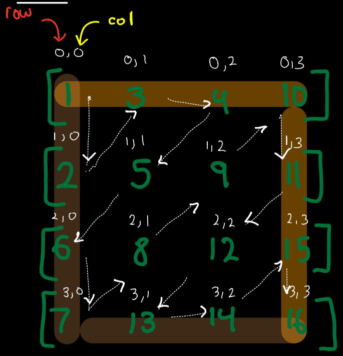


```javascript
// O(n) time | O(n) space
function zigzagTraverse(array) {
  const height = array.length - 1;
  const width = array[0].length - 1;
  const result = [];
  let row = 0;
  let col = 0;
  let goingDown = true;

  while (!isOutOfBounds(row, col, height, width)) {
    result.push(array[row][col]);
    if (goingDown) {
      if (col === 0 || row === height) {
        goingDown = false;
        if (row === height) {
          col++;
        } else {
          row++;
        }
      } else {
        row++;
        col--;
      }
    } else {
      if (row === 0 || col === width) {
        goingDown = true;
        if (col === width) {
          row++;
        } else {
          col++;
        }
      } else {
        row--;
        col++;
      }
    }
  }
  return result;
}

function isOutOfBounds(row, col, height, width) {
  return row < 0 || row > height || col < 0 || col > width;
}
```

Idea:



- Set height; width, row, col, goingDown and result array
- Check if we're goingDown or not or if we're on the perimeter and traverse accordingly

Time: O(n) (where n is the # of elements in the input array) since we are traversing the entire array

Space: O(n) because of our result matrix