```
function resolve(N) {
    // 初始值构建
    // ES6语法创建
    // let triangle = Array.from({ length: N + 2 }, (_, row) =>
    //     Array(N + 2).fill(1)
    // )
    // for (let i = 1; i <= N; i++) {
    //     for (let j = 1; j <= N - i + 1; j++) {
    //         triangle[i][j] = 0
    //     }
    // }

    // ES5语法创建
    let triangle = []
    for (let i = 0; i < N + 2; i++) {
        triangle[i] = []
        for (let j = 0; j < N + 2; j++) {
            triangle[i][j] = 1
        }
    }
    for (let i = 1; i <= N; i++) {
        for (let j = 1; j <= N - i + 1; j++) {
            triangle[i][j] = 0
        }
    }

    // console.log(triangle)

    let row = 1, col = 1;
    let cur = 1;
    let total = N * (N + 1) / 2;
    let dir = 0;  // 共三种方向：0,1,2
    while (cur <= total) {
        triangle[row][col] = cur++
        // 判断下一个的值是否==0，若不是的话，证明这个方向已经到边界了
        switch (dir) {
            case 0:
                if (triangle[row + 1][col] == 0) {
                    row++
                } else {
                    dir = 1
                    row--
                    col++
                }
                break;
            case 1:
                if (triangle[row - 1][col + 1] == 0) {
                    row--
                    col++
                } else {
                    dir = 2
                    col--
                }
                break  //TODO:记得break
            case 2:
                if (triangle[row][col - 1] == 0) {
                    col--
                } else {
                    dir = 0
                    row++
                }
                break;  // TODO:记得break
        }
    }

    let res = []
    for (let i = 1; i <= N; i++) {
        let item = triangle[i].slice(1, N - i + 2)
        res.push(...item) // 此处若用concat，只是返回新数组  原来的数组并未改
变！！
    }

    console.log(res)
}

resolve(5)
```