6322. Check Knight Tour Configuration

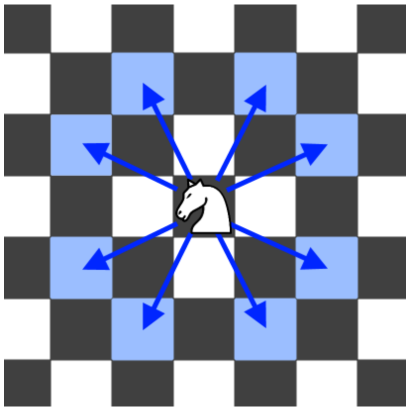
User Accepted:	1209
User Tried:	1646
Total Accepted:	1217
Total Submissions:	2181
Difficulty:	Medium

There is a knight on an `n x n` chessboard. In a valid configuration, the knight starts **at the top-left cell** of the board and visits every cell on the board **exactly once**.

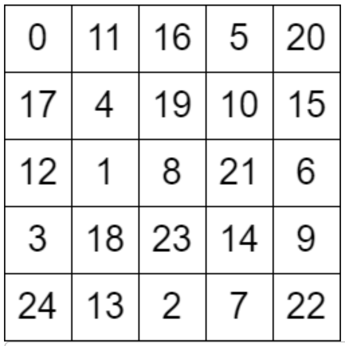
You are given an `n x n` integer matrix `grid` consisting of distinct integers from the range `[0, n * n - 1]` where `grid[row][col]` indicates that the cell `(row, col)` is the `grid[row][col]`th cell that the knight visited. The moves are **0-indexed**.

Return `true` *if* `grid` *represents a valid configuration of the knight's movements or* `false` *otherwise*.

Note that a valid knight move consists of moving two squares vertically and one square horizontally, or two squares horizontally and one square vertically. The figure below illustrates all the possible eight moves of a knight from some cell.



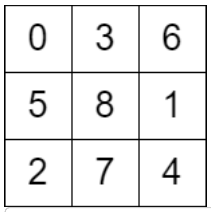
Example 1:



0	11	16	5	20
17	4	19	10	15
12	1	8	21	6
3	18	23	14	9
24	13	2	7	22

```
Input: grid = [[0,11,16,5,20],[17,4,19,10,15],[12,1,8,21,6],[3,18,23,14,9],[24,13,2,7,22]]
Output: true
Explanation: The above diagram represents the grid. It can be shown that it is a valid configuration.
```

Example 2:



0	3	6
5	8	1
2	7	4

```
Input: grid = [[0,3,6],[5,8,1],[2,7,4]]
Output: false
Explanation: The above diagram represents the grid. The 8th move of the knight is not valid considering its position after the
```

Constraints:

- `n == grid.length == grid[i].length`

- `3 <= n <= 7`
- `0 <= grid[row][col] < n * n`
- All integers in `grid` are **unique**.

JavaScript

```javascript
const checkValidGrid = (g) => {
    if (g[0][0] != 0) return false;
    let n = g.length, a = Array(n * n);
    for (let i = 0; i < n; i++) {
        for (let j = 0; j < n; j++) a[g[i][j]] = [i, j];
    }
    for (let i = 1; i < n * n; i++) {
        let [x1, y1] = a[i - 1], [x2, y2] = a[i];
        if (!ok(x1, y1, x2, y2)) return false;
    }
    return true;
};

const ok = (x1, y1, x2, y2) => {
    let dx = Math.abs(x1 - x2), dy = Math.abs(y1 - y2);
    return (dx == 1 && dy == 2) || (dx == 2 && dy == 1)
};
```

Custom Testcase   Use Example Testcases

Run   Submit

Submission Result: Accepted (/submissions/detail/917821240/)   More Details (/submissions/detail/917821240/)

Share your acceptance!