

Diagonal Traverse

Given a matrix of M x N elements (M rows, N columns), return all elements of the matrix in diagonal order as shown in the below image.

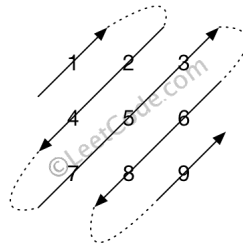
Example:

Input:

```
[  
  [ 1, 2, 3 ],  
  [ 4, 5, 6 ],  
  [ 7, 8, 9 ]  
]
```

Output: [1,2,4,7,5,3,6,8,9]

Explanation:



Note:

1. The total number of elements of the given matrix will not exceed 10,000.

Solution 1

I don't think this is a hard problem. It is easy to figure out the walk pattern.

Anyway...

Walk patterns:

- If out of **bottom border** ($\text{row} \geq m$) then $\text{row} = m - 1$; $\text{col} += 2$; change walk direction.
- if out of **right border** ($\text{col} \geq n$) then $\text{col} = n - 1$; $\text{row} += 2$; change walk direction.
- if out of **top border** ($\text{row} < 0$) then $\text{row} = 0$; change walk direction.
- if out of **left border** ($\text{col} < 0$) then $\text{col} = 0$; change walk direction.
- Otherwise, just go along with the current direction.

Time complexity: $O(m * n)$, m = number of rows, n = number of columns.

Space complexity: $O(1)$.

```
public class Solution {
    public int[] findDiagonalOrder(int[][] matrix) {
        if (matrix == null || matrix.length == 0) return new int[0];
        int m = matrix.length, n = matrix[0].length;

        int[] result = new int[m * n];
        int row = 0, col = 0, d = 0;
        int[][] dirs = {{-1, 1}, {1, -1}};

        for (int i = 0; i < m * n; i++) {
            result[i] = matrix[row][col];
            row += dirs[d][0];
            col += dirs[d][1];

            if (row >= m) { row = m - 1; col += 2; d = 1 - d;}
            if (col >= n) { col = n - 1; row += 2; d = 1 - d;}
            if (row < 0) { row = 0; d = 1 - d;}
            if (col < 0) { col = 0; d = 1 - d;}
        }

        return result;
    }
}
```

written by [shawngao](#) original link [here](#)

Solution 2

Within diagonal row+col is same, so we first sort index pairs by row+col, and within diagonal sort them either by row or by column index depending if row+col is odd/even.

```
def findDiagonalOrder(self, matrix):  
    l = [[i,j] for i in range(len(matrix)) for j in range(len(matrix[0]))]  
    l.sort(key=lambda x: float(x[0]+x[1]) - float(x[(x[0]+x[1])%2])*0.00000001  
    )  
    return [matrix[x][y] for [x,y] in l]
```

written by [_aig_](#) original link [here](#)

Solution 3

Put all diagonal sequences from top-right to bottom-left to an array and then combine all sequence together by reversing odd sequences.

```
class Solution {
public:
    vector<int> findDiagonalOrder(vector<vector<int>>& matrix) {
        int m = matrix.size();
        if (m == 0) return vector<int>();
        int n = matrix[0].size();
        vector<vector<int>> tmp (m+n-1);
        for (int i = 0; i < m+n-1 ; i++) {
            int row = max(0, i-n+1);
            int col = min(i, n-1);
            for (; col >= 0 && row < m; row++, col--) {
                tmp[i].push_back(matrix[row][col]);
            }
        }
        vector<int> res;
        for (int i = 0; i < tmp.size(); i++) {
            if (i % 2) res.insert(res.end(), tmp[i].begin(), tmp[i].end());
            else res.insert(res.end(), tmp[i].rbegin(), tmp[i].rend());
        }
        return res;
    }
};
```

written by [Hcisly](#) original link [here](#)

From [LeetCoder](#).