

(γ)aths : Video -

44



∞ → codestorywithMIK

→ CSwithMIK

→ codestorywithMIK

WeekendWithMIK

Weekend WITH MIK

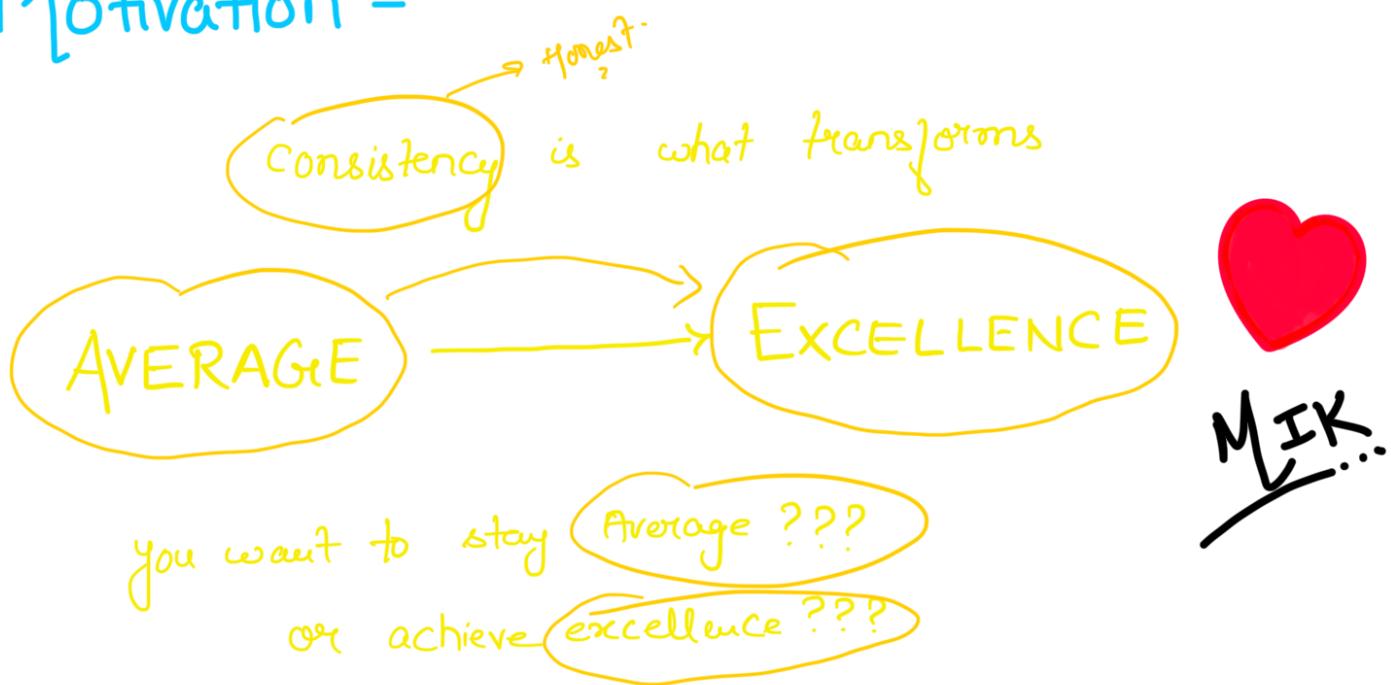
@WeekendWithMIK

Welcome to WeekendWithMIK

MIK.

Trey this channel to
see "Life behind the Scenes + Tech News"

Motivation -



1895. Largest Magic Square

Medium Topics Companies Hint

A $k \times k$ **magic square** is a $k \times k$ grid filled with integers such that every row sum, every column sum, and both diagonal sums are all equal. The integers in the magic square do not have to be distinct. Every 1×1 grid is trivially a **magic square**.

Given an $m \times n$ integer grid, return the **size** (i.e., the side length k) of the **largest magic square** that can be found within this grid.

Example 1:

7	1	4	5	6
2	5	1	6	4
1	5	4	3	2
1	2	7	3	4



Input: grid = [[7,1,4,5,6], [2,5,1,6,4], [1,5,4,3,2], [1,2,7,3,4]]

Output: 3 ↗

Explanation: The largest magic square has a size of 3.

Every row sum, column sum, and diagonal sum of this magic square is equal to 12.

- Row sums: $5+1+6 = 5+4+3 = 2+7+3 = 12$
- Column sums: $5+5+2 = 1+4+7 = 6+3+3 = 12$
- Diagonal sums: $5+4+3 = 6+4+2 = 12$

Constraints:

constraint analysis:-

- $m == \text{grid.length}$ ↗
- $n == \text{grid}[i].length$ ↗
- $1 \leq m, n \leq 50$ ↗
- $1 \leq \text{grid}[i][j] \leq 10^6$ ↗

Thought Process

$(j + \text{side} - 1)$

$\text{side} = 3$

$$\begin{aligned} & 2 + 3 - 1 \\ & = 4 \quad j \end{aligned}$$

7	1	4	5	6
2	5	1	6	4
1	5	4	3	2
1	2	7	3	4

\rightarrow^1

$k=0, i=0, j=0$

$m = 4, n = 5$

Grid

$(i+2) \cap (j+2) \quad k++ \quad n = 5$

$k=0, i=1, j=1$

$m = 4, n = 5$

$k=0, i=2, j=2$

$m = 4, n = 5$

$k=0, i=3, j=3$

$m = 4, n = 5$

$k=0, i=4, j=4$

$m = 4, n = 5$

$k=0, i=5, j=5$

$m = 4, n = 5$

$k=0, i=6, j=6$

$m = 4, n = 5$

$k=0, i=7, j=7$

$m = 4, n = 5$

$k=0, i=8, j=8$

$m = 4, n = 5$

$k=0, i=9, j=9$

$m = 4, n = 5$

$k=0, i=10, j=10$

$m = 4, n = 5$

$k=0, i=11, j=11$

$m = 4, n = 5$

$k=0, i=12, j=12$

$m = 4, n = 5$

$k=0, i=13, j=13$

$m = 4, n = 5$

$k=0, i=14, j=14$

$m = 4, n = 5$

$k=0, i=15, j=15$

$m = 4, n = 5$

$k=0, i=16, j=16$

$m = 4, n = 5$

$k=0, i=17, j=17$

$m = 4, n = 5$

$k=0, i=18, j=18$

$m = 4, n = 5$

$k=0, i=19, j=19$

$m = 4, n = 5$

$k=0, i=20, j=20$

$m = 4, n = 5$

$k=0, i=21, j=21$

$m = 4, n = 5$

$k=0, i=22, j=22$

$m = 4, n = 5$

$k=0, i=23, j=23$

$m = 4, n = 5$

$k=0, i=24, j=24$

$m = 4, n = 5$

$k=0, i=25, j=25$

$m = 4, n = 5$

$k=0, i=26, j=26$

$m = 4, n = 5$

$k=0, i=27, j=27$

$m = 4, n = 5$

$k=0, i=28, j=28$

$m = 4, n = 5$

$k=0, i=29, j=29$

$m = 4, n = 5$

$k=0, i=30, j=30$

$m = 4, n = 5$

$k=0, i=31, j=31$

$m = 4, n = 5$

$k=0, i=32, j=32$

$m = 4, n = 5$

$k=0, i=33, j=33$

$m = 4, n = 5$

$k=0, i=34, j=34$

$m = 4, n = 5$

$k=0, i=35, j=35$

$m = 4, n = 5$

$k=0, i=36, j=36$

$m = 4, n = 5$

$k=0, i=37, j=37$

$m = 4, n = 5$

$k=0, i=38, j=38$

$m = 4, n = 5$

$k=0, i=39, j=39$

$m = 4, n = 5$

$k=0, i=40, j=40$

$m = 4, n = 5$

$k=0, i=41, j=41$

$m = 4, n = 5$

$k=0, i=42, j=42$

$m = 4, n = 5$

$k=0, i=43, j=43$

$m = 4, n = 5$

$k=0, i=44, j=44$

$m = 4, n = 5$

$k=0, i=45, j=45$

$m = 4, n = 5$

$k=0, i=46, j=46$

$m = 4, n = 5$

$k=0, i=47, j=47$

$m = 4, n = 5$

$k=0, i=48, j=48$

$m = 4, n = 5$

$k=0, i=49, j=49$

$m = 4, n = 5$

$k=0, i=50, j=50$

$m = 4, n = 5$

$k=0, i=51, j=51$

$m = 4, n = 5$

$k=0, i=52, j=52$

$m = 4, n = 5$

$k=0, i=53, j=53$

$m = 4, n = 5$

$k=0, i=54, j=54$

$m = 4, n = 5$

$k=0, i=55, j=55$

$m = 4, n = 5$

$k=0, i=56, j=56$

$m = 4, n = 5$

$k=0, i=57, j=57$

$m = 4, n = 5$

$k=0, i=58, j=58$

$m = 4, n = 5$

$k=0, i=59, j=59$

$m = 4, n = 5$

$k=0, i=60, j=60$

$m = 4, n = 5$

$k=0, i=61, j=61$

$m = 4, n = 5$

$k=0, i=62, j=62$

$m = 4, n = 5$

$k=0, i=63, j=63$

$m = 4, n = 5$

$k=0, i=64, j=64$

$m = 4, n = 5$

$k=0, i=65, j=65$

$m = 4, n = 5$

$k=0, i=66, j=66$

$m = 4, n = 5$

$k=0, i=67, j=67$

$m = 4, n = 5$

$k=0, i=68, j=68$

$m = 4, n = 5$

$k=0, i=69, j=69$

$m = 4, n = 5$

$k=0, i=70, j=70$

$m = 4, n = 5$

$k=0, i=71, j=71$

$m = 4, n = 5$

$k=0, i=72, j=72$

$m = 4, n = 5$

$k=0, i=73, j=73$

$m = 4, n = 5$

$k=0, i=74, j=74$

$m = 4, n = 5$

$k=0, i=75, j=75$

$m = 4, n = 5$

$k=0, i=76, j=76$

$m = 4, n = 5$

$k=0, i=77, j=77$

$m = 4, n = 5$

$k=0, i=78, j=78$

$m = 4, n = 5$

$k=0, i=79, j=79$

$m = 4, n = 5$

$k=0, i=80, j=80$

$m = 4, n = 5$

$k=0, i=81, j=81$

$m = 4, n = 5$

$k=0, i=82, j=82$

$m = 4, n = 5$

$k=0, i=83, j=83$

$m = 4, n = 5$

$k=0, i=84, j=84$

$m = 4, n = 5$

$k=0, i=85, j=85$

$m = 4, n = 5$

$k=0, i=86, j=86$

$m = 4, n = 5$

$k=0, i=87, j=87$

$m = 4, n = 5$

$k=0, i=88, j=88$

$m = 4, n = 5$

$k=0, i=89, j=89$

$m = 4, n = 5$

$k=0, i=90, j=90$

$m = 4, n = 5$

$k=0, i=91, j=91$

$m = 4, n = 5$

$k=0, i=92, j=92$

$m = 4, n = 5$

$k=0, i=93, j=93$

$m = 4, n = 5$

$k=0, i=94, j=94$

$m = 4, n = 5$

$k=0, i=95, j=95$

$m = 4, n = 5$

$k=0, i=96, j=96$

$m = 4, n = 5$

$k=0, i=97, j=97$

$m = 4, n = 5$

$k=0, i=98, j=98$

$m = 4, n = 5$

$k=0, i=99, j=99$

$m = 4, n = 5$

$k=0, i=100, j=100$

$m = 4, n = 5$

$k=0, i=101, j=101$

$m = 4, n = 5$

$k=0, i=102, j=102$

$m = 4, n = 5$

$k=0, i=103, j=103$

$m = 4, n = 5$

$k=0, i=104, j=104$

$m = 4, n = 5$

$k=0, i=105, j=105$

$$5+4+1 = \min(m,n)$$

Check every possible square.

$$k=1$$

$\text{grid}[i+k][j+side-1-k]$

$\text{grid}[2][1+3-1-1]$ Square = side = 1×1

$[2][2]$

Ans = 1

2x2
3x3 ←
4x4
5x5
...

Side = $\min(m,n)$; side >= 2

→ Slight improvement

~~for (side = min(m,n) ; side >= 2 ; side --) {~~

\Rightarrow $\text{for } (i = 0 ; i + side - 1 < m ;) \{$

\Rightarrow $\text{for } (j = 0 ; j + side - 1 < n ; j++)$

// (i,j)

// side

O(1)

int targetSum = rowCumSum[i][j+side-1] - (j > 0 ?
rowCumSum[i][j-1] : 0)

bool allSame = True;

// Check all remain rows

$\text{for } (int r = i+1 ; r < i + side ; r++) \{$
int sum = rowCumSum[r][j+side-1] - (j > 0 ?
rowCumSum[r][j-1] : 0);
if (sum != targetSum) {

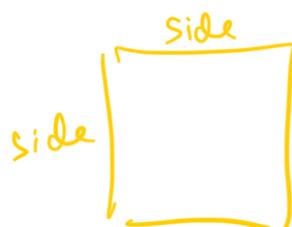
```
allSame = False;  
break;
```

```
if (allSame == False)  
    continue;
```

X

// Check all columns.

```
for (c = j ; c < j+side ; c++) {  
    int sum = colColumn[i+side-1][c]  
            - colColumn[i-1][c];  
    if (sum != target)  
        allSame = False;  
    break;
```



```
if (!allSame) continue;
```

```
diag = 0; antidi = 0;
```

```
for (int k=0; k < side; k++) {  
    diag += grid[i+k][j+k];  
    antidi += grid[i+k][j+side-1-k];
```

```
if (diag == targetSum && antidi == targetSum)  
    return side;
```

}

1.

Time Complexity:

~ | ~ Com ~

```
class Solution {
public:
    int largestMagicSquare(vector<vector<int>>& grid) {
        int rows = grid.size(); //m
        int cols = grid[0].size(); //n

        //Row wise prefix sum
        vector<vector<int>> rowCumSum(rows, vector<int>(cols));
        for(int i = 0; i < rows; i++) {
            rowCumSum[i][0] = grid[i][0];
            for(int j = 1; j < cols; j++) {
                rowCumSum[i][j] = rowCumSum[i][j-1] + grid[i][j];
            }
        }
    }
```

$O(m \times n)$

```
//Column wise prefix sum
vector<vector<int>> colCumSum(rows, vector<int>(cols));
for(int j = 0; j < cols; j++) {
    colCumSum[0][j] = grid[0][j];
    for(int i = 1; i < rows; i++) {
        colCumSum[i][j] = colCumSum[i-1][j] + grid[i][j];
    }
}
```

$O(m \times n)$

```
//try all possible side squares from each cell
for(int side = min(rows, cols); side >= 2; side--) {
    for(int i = 0; i + side - 1 < rows; i++) {
        for(int j = 0; j + side - 1 < cols; j++) {
```

$\text{side} = 2 \rightarrow m \cdot n \cdot 2$

$3 \rightarrow m \cdot n \cdot 3$

$4 \rightarrow m \cdot n \cdot 4$

$5 \rightarrow m \cdot n \cdot 5$

.

i

```
    int targetSum = rowCumSum[i][j+side-1] - (j > 0 ? rowCumSum[i][j-1] : 0);
    bool allSame = true;

    //check rows
    for(int r = i+1; r < i + side; r++) {
        int rowSum = rowCumSum[r][j+side-1] - (j > 0 ? rowCumSum[r][j-1] : 0);

        if(rowSum != targetSum) {
            allSame = false;
            break;
        }
    }

    if(!allSame)
        continue;
```

$O(\text{side})$

$m \cdot n (1+2+3+4+5+\dots)$

$m \cdot n \left(\frac{s(s+1)}{2} \right)$

$m \cdot n \cdot s^2$

```
    //check columns
    for(int c = j; c < j + side; c++) {
        int colSum = colCumSum[i + side - 1][c] - (i > 0 ? colCumSum[i-1][c] : 0);

        if(colSum != targetSum) {
            allSame = false;
        }
    }

    if(!allSame)
        continue;
```

$O(\text{side})$

$O(m \cdot n \cdot s^2)$

```
    //check diag and antiDiag
    int diag = 0;
    int antiDiag = 0;
    for(int k = 0; k < side; k++) {
        diag += grid[i+k][j+k];
        antiDiag += grid[i+k][j + side - 1 - k];
    }

    if(diag == targetSum && antiDiag == targetSum) {
        return side;
    }
}
```

$T.C. O(m \cdot n \cdot (\min(m, n))^2)$

\approx

