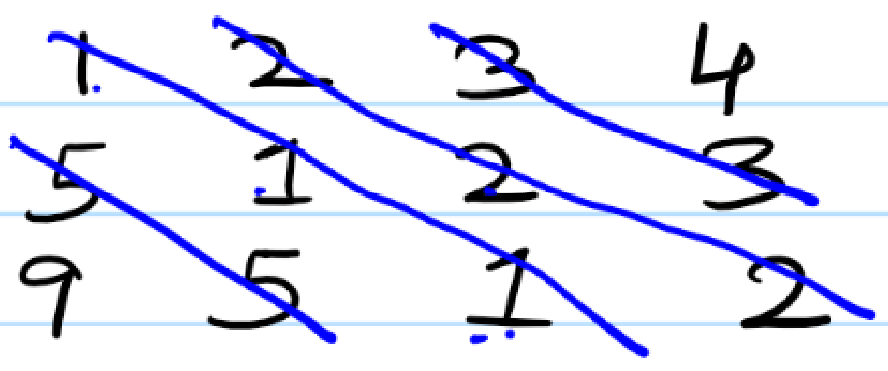
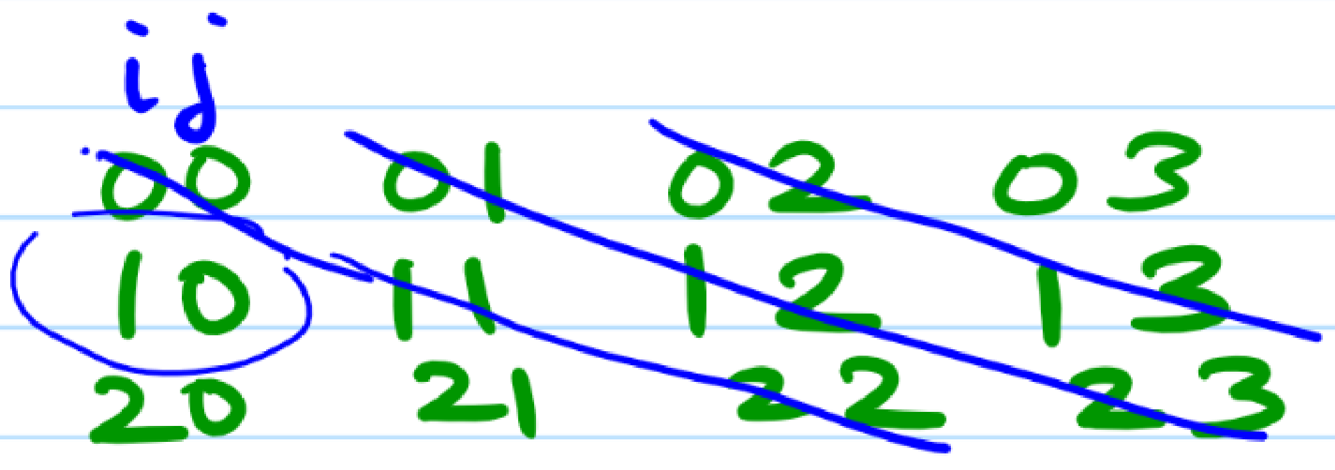
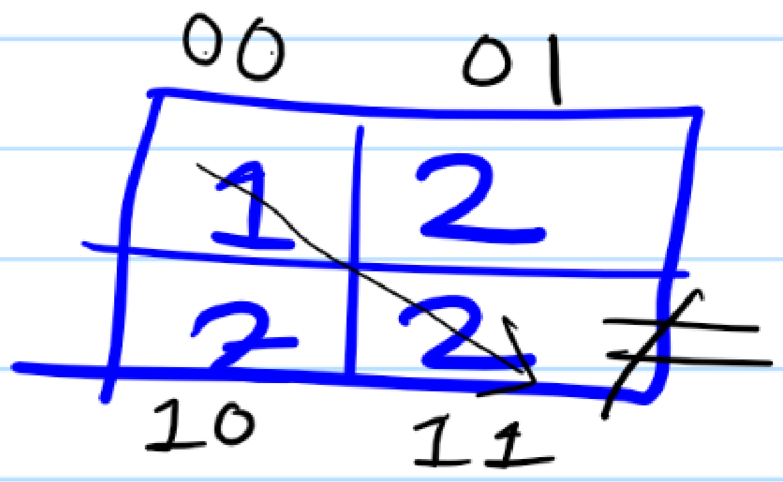
2D arrays class 2

Toeplitz Matrix





$(i-j)$ = same value



$1 \neq 2$ on diagonal

Not a Toeplitz matrix

HashMap

0	1
-1	2
1	2

False

1st approach:

HashMap

Key i,j difference	Value Element value
0	1 2
1	3
2	
3	4

Algorithm:

1) Maintain a hashmap of i,j difference and corresponding value of element

2) Keep on comparing value of elements who have the same i,j difference

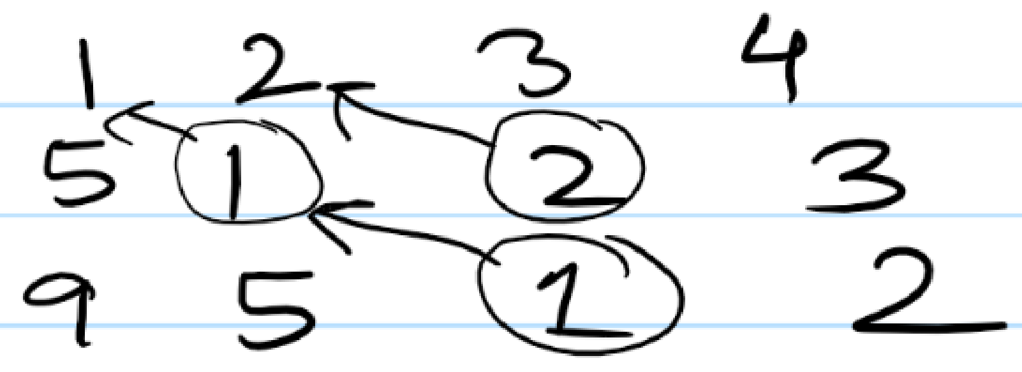
If not equal simply return false.

3) Return true

TC: O(M*N)

SC: O(M+N)

Approach 2



a[i][j] =? a[i-1][j-1]

Algorithm: For every element that belongs to a diagonal, compare it with the previous element (if it exists) on the same diagonal.

TC: O(MXN) SC: O(1)

Q2 Transpose of a Matrix

$$\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix} \xleftrightarrow{\text{Transpose}} \begin{bmatrix}1&4&7\\2&5&8\\3&6&9\end{bmatrix}$$

$a[i][j] \longleftrightarrow a[j][i]$.

TC: $O(m \times n)$

SC: $O(m * n)$

Q3: Rotate image by 90° clockwise

$$\begin{matrix}1&2&3\\4&5&6\\ \cancel{7}&\cancel{8}&\cancel{9}\end{matrix} \longrightarrow \begin{matrix}1&4&7\\2&5&8\\3&6&9\end{matrix} \rightarrow \begin{matrix}7&4&1\\8&5&2\\9&6&3\end{matrix}$$

$$\begin{matrix}7\\8\\9\end{matrix}$$

$$\boxed{90^\circ \text{ rotation} = \text{Transpose} + \text{Reverse}}$$

Answer

$$\begin{matrix}5&1&9&11\\2&4&8&10\\13&3&6&7\\15&14&12&16\end{matrix} \longrightarrow \begin{matrix}15&13&2&5\\14&3&4&1\\12&6&8&9\\16&7&10&11\end{matrix}$$

$$\begin{matrix}5&2&13&15\\1&4&3&14\\9&8&6&12\\11&10&7&16\end{matrix} \longrightarrow \begin{array}{ccccc} &00&01&02&03\\ &15&13&2&5\\ &14&3&4&1\\ 20&\boxed{12}&6&8&\boxed{9}\ 23\\ &16&7&10&11\end{array}$$

$4-3-1 = 0$

TC: $O(m * n)$

SC: $O(1)$

Transpose: $a[i][j] = a[j][i]$

Reverse: $a[i][j] = a[i][n-j-1]$

n = no. of columns

$O(m \times n) + O(m \times n)$
$= O(m \times n)$

```
public void rotate ( int [][] matrix )
     {
          transpose ( matrix);
          reverse ( matrix);
     }
void transpose ( int [][] matrix)
     {
      int n = matrix.length;
 for ( int i=0; i<n; i++)
     {
     for ( int j = i+1; j<n; j++)
          {
               int tmp = matrix[i][j];
               matrix[i][j] = matrix [j][i];
               matrix (j] [i] = temp;
          }
     }
```

```
void reverse ( int [][] matrix)
   {
   int n = matrix.length;
     for ( i=0; i<n ; i++)
       {
       for ( j=0; j< n/2 ; j++)
         {    int temp = matrix [i][j];
              matrix [i][j] = matrix[i] [n-j-1];
              matrix [i] [n-j-1] = temp;
         }
       }
   }
```

Q4: Degree of an array

0 1 2 3 4

[1, 2, 2, 3, 1]

5

Degree:

1 → 2

2 → 2

3 → 1

Left of 2 = 1
Right of 2 = 2
↓
2-1+1
= 2

Left of 1 = 0
Right " = 4
4-0+1
= 5

Array size 5 has degree 2

[1,2,2] = 2

[2,2,3] = 2

= 1 ×

[2,3,1]

✓ [2,2] = 2

size = 2

Ans = 6

0 1 2 3 4 5 6

[1, 2, 2, 3, 1, 4, 2]

degree

minimum subarray with degree 3

1 → 2	
2 → 3	
3 → 1	
4 → 1	

Left occurence
Right occurence
count of an element

Left

1	0
2	1
3	3
4	5

Right

1	~~0~~ 4
2	~~1~~ ~~2~~ 6
3	3
4	5

count

1	~~1~~ 2
2	~~1~~ ~~2~~ 3
3	1
4	1

Right - left +1
6 - 1 + 1
= $\boxed{6}$

Max count = 3 of element 2

TC: O(N)

SC: 3 hashMaps O(N)