

# 2D Array Operation

Rotate and Mirror

# Content

- Rotate 90 degrees clockwise
- Rotate 90 degrees counterclockwise
  - Mirror upside down
  - Mirror left and right

# 1-1 Rotate 90 degrees clockwise

Given a 2 X 3 (row X column) 2D Array:  $A[2][3]$

Original number of Row: 2  
Original number of Column: 3

Number of Row after rotation: 3  
Number of Column after rotation: 2

$A_{00} = 1$	$A_{01} = 2$	$A_{02} = 3$
$A_{10} = 4$	$A_{11} = 5$	$A_{12} = 6$



Rotate 90 degrees clockwise

$A_{10} = 4$	$A_{00} = 1$
$A_{11} = 5$	$A_{01} = 2$
$A_{12} = 6$	$A_{02} = 3$

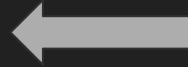
Original number of Row: 2  
Original number of Column: 3

## 1-2 Rotate 90 degrees clockwise

Number of Row after rotation: 3  
Number of Column after rotation: 2

Use an empty array T to store the array after rotation; this allows us to print out the array after rotation using two for loops.

$T_{00} = 4$	$T_{01} = 1$
$T_{10} = 5$	$T_{11} = 2$
$T_{20} = 6$	$T_{21} = 3$



$A_{10} = 4$	$A_{00} = 1$
$A_{11} = 5$	$A_{01} = 2$
$A_{12} = 6$	$A_{02} = 3$

\*Row indices of the array T correspond to the column indices of array A after rotation; the column indices of T are derived by subtracting the row index of A from the original number of row minus one.

Pseudo code:  $T[\text{row}][\text{column}] = A[\text{original number of row} - 1 - \text{column}][\text{row}]$

# 1-3 Rotate 90 degrees clockwise

Code in C++:

```
void clockwise(int Arr[100][100], int r, int c)
{
    int T[100][100]; //to store the array after rotation

    for(int i=0; i<c; i++){
        for(int j=0; j<r; j++){
            T[i][j] = Arr[r-1-j][i];
        }
    }

    for(int i=0; i<c; i++){
        for(int j=0; j<r; j++){
            printf("%2d ", T[i][j]);
        }
        cout << endl;
    }
}
```

Original Array:

1	2	3	4
5	6	7	8
9	10	11	12

Rotate 90 degrees clockwise:

9	5	1
10	6	2
11	7	3
12	8	4

## 2-1 Rotate 90 degrees counterclockwise

Original number of Row: 2  
Original number of Column: 3

Given a 2 X 3 (row X column) 2D Array:  $A[2][3]$

Number of Row after rotation: 3  
Number of Column after rotation: 2

$A_{00} = 1$	$A_{01} = 2$	$A_{02} = 3$
$A_{10} = 4$	$A_{11} = 5$	$A_{12} = 6$



Rotate 90 degrees counterclockwise

$A_{02} = 3$	$A_{12} = 6$
$A_{01} = 2$	$A_{11} = 5$
$A_{00} = 1$	$A_{10} = 4$

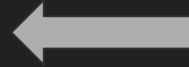
Original number of Row: 2  
Original number of Column: 3

## 2-2 Rotate 90 degrees counterclockwise

Number of Row after rotation: 3  
Number of Column after rotation: 2

Use an empty array T to store the array after rotation; this allows us to print out the array after rotation using two for loops.

$T_{00} = 3$	$T_{01} = 6$
$T_{10} = 2$	$T_{11} = 5$
$T_{20} = 1$	$T_{21} = 4$



$A_{02} = 3$	$A_{12} = 6$
$A_{01} = 2$	$A_{11} = 5$
$A_{00} = 1$	$A_{10} = 4$

\*Column indices of the array T correspond to the row indices of array A after rotation; the row indices of T are derived by subtracting the column index of A from the original number of row minus one.

Pseudo code:  $T[\text{row}][\text{column}] = A[\text{column}][\text{original number of column} - 1 - \text{row}]$

## 2-3 Rotate 90 degrees counterclockwise

Code in C++:

```
void counterclockwise(int Arr[100][100], int r, int c)
{
    int T[100][100]; // to store the array after rotation

    for(int i=0; i<c; i++){
        for(int j=0; j<r; j++){
            T[i][j] = Arr[j][c-1-i];
        }
    }

    for(int i=0; i<c; i++){
        for(int j=0; j<r; j++){
            printf("%2d ", T[i][j]);
        }
        cout << endl;
    }
}
```

Original Array:

1	2	3	4
5	6	7	8
9	10	11	12

Rotate 90 degrees counterclockwise:

4	8	12
3	7	11
2	6	10
1	5	9



# Key for array rotation

No matter the type of rotation, we're always trying to find the pattern of indices of row and index of the array after rotation and reconfigure that to an empty array.

In short:

1. Draw out the array after rotation on paper with indices if possible
2. Find the pattern between the indices of array after rotation
3. Use the pattern found to pass each index into an empty array

## 3-1 Mirror upside down

Given a 2 X 3 (row X column) 2D Array:  $A[2][3]$

$A_{00} = 1$	$A_{01} = 2$	$A_{02} = 3$
$A_{10} = 4$	$A_{11} = 5$	$A_{12} = 6$

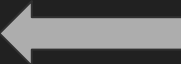


Mirror upside down

$A_{10} = 4$	$A_{11} = 5$	$A_{12} = 6$
$A_{00} = 1$	$A_{01} = 2$	$A_{02} = 3$

## 3-2 Mirror upside down

Use an empty array T to store the array after rotation; this allows us to print out the array after rotation using two for loops.

$T_{00} = 4$	$T_{01} = 5$	$T_{02} = 6$		$A_{10} = 4$	$A_{11} = 5$	$A_{12} = 6$
$T_{10} = 1$	$T_{11} = 2$	$T_{12} = 3$		$A_{00} = 1$	$A_{01} = 2$	$A_{02} = 3$

\*Column indices of the array T correspond to the column indices of array A after mirroring; the row indices of T are derived by subtracting the row index of A from the number of row minus one.

Pseudo code:  $T[\text{row}][\text{column}] = A[\text{number of row} - 1 - \text{row}][\text{column}]$

## 3-3 Mirror upside down

Code in C++:

```
void mirror_upside_down(int Arr[100][100], int r, int c)
{
    int T[100][100]; // to store the array after mirroring

    for(int i=0; i<r; i++){
        for(int j=0; j<c; j++){
            T[i][j] = Arr[r-1-i][j];
        }
    }

    for(int i=0; i<r; i++){
        for(int j=0; j<c; j++){
            printf("%2d ", T[i][j]);
            cout << " ";
        }
        cout << endl;
    }
}
```

Original Array:

1	2	3	4
5	6	7	8
9	10	11	12

Mirror upside down:

9	10	11	12
5	6	7	8
1	2	3	4

## 4-1 Mirror left and right

Given a 2 X 3 (row X column) 2D Array:  $A[2][3]$

$A_{00} = 1$	$A_{01} = 2$	$A_{02} = 3$
$A_{10} = 4$	$A_{11} = 5$	$A_{12} = 6$

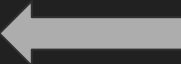


Mirror left and right

$A_{02} = 3$	$A_{01} = 2$	$A_{00} = 1$
$A_{12} = 6$	$A_{11} = 5$	$A_{10} = 4$

## 4-2 Mirror left and right

Use an empty array T to store the array after rotation; this allows us to print out the array after rotation using two for loops.

$T_{00} = 3$	$T_{01} = 2$	$T_{02} = 1$		$A_{02} = 3$	$A_{01} = 2$	$A_{00} = 1$
$T_{10} = 6$	$T_{11} = 5$	$T_{12} = 4$		$A_{12} = 6$	$A_{11} = 5$	$A_{10} = 4$

\*Row indices of the array T correspond to the row indices of array A after mirroring; the column indices of T are derived by subtracting the column index of A from the number of column minus one.

Pseudo code:  $T[\text{row}][\text{column}] = A[\text{row}][\text{number of column} - 1 - \text{column}]$

## 4-3 Mirror left and right

Code in C++:

```
void mirror_left_right(int Arr[100][100], int r, int c)
{
    int T[100][100]; // to store the array after mirroring

    for(int i=0; i<r; i++){
        for(int j=0; j<c; j++){
            T[i][j] = Arr[i][c-1-j];
        }
    }

    for(int i=0; i<r; i++){
        for(int j=0; j<c; j++){
            printf("%2d ", T[i][j]);
        }
        cout << endl;
    }
}
```

Original Array:

1	2	3	4
5	6	7	8
9	10	11	12

Mirror left and right:

4	3	2	1
8	7	6	5
12	11	10	9