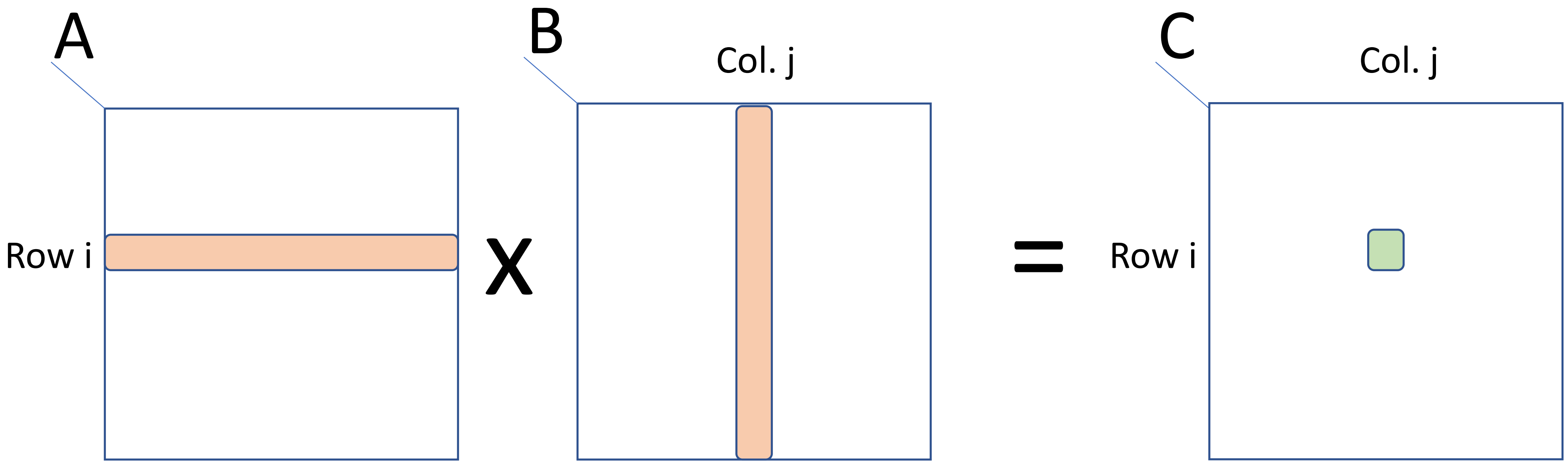
Cannon’s Algorithm for Matrix Multiplication

L. Cannon, 1969

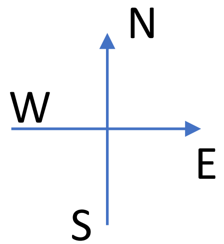
For ease of exposition, let us assume square matrices/

Input: A(nxn) x B(nxn) = C(nxn)



$$C[i][j] = \sum_{k=0}^{n-1} A[i][k] \times B[k][j]$$

Cannon's algorithm



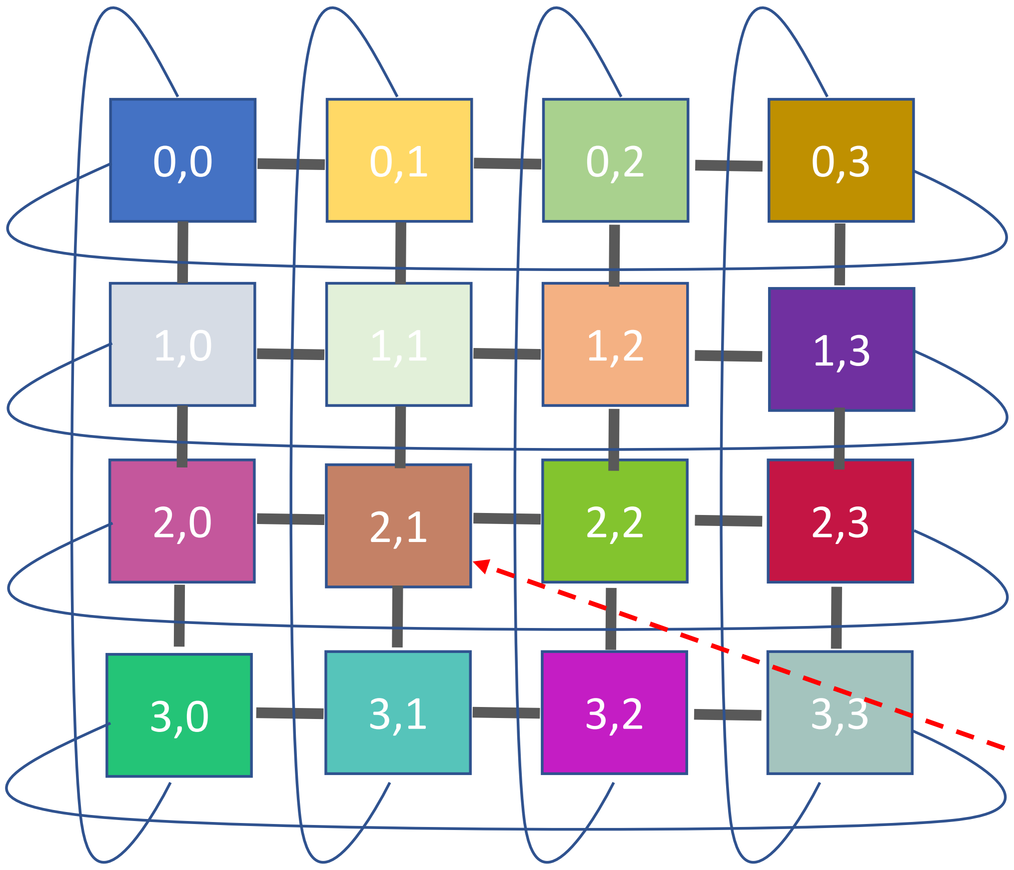
// Initialize matrices A and B on the processes:

- First load cells A[i,j] and B[i,j] on proc. [i,j]
- Circular left shift the i^{th} row of matrix A, i times
- Circular up shift the j^{th} column of matrix B, j times
- Set **a** and **b** to the corresponding values of A and B the proc. got after the shift.

// Main algorithm @ proc. [i,j]

- c=0 **// local c element which will be output by this proc.**
- For the length of a row:
  - ***c += a x b***
  - Move ***a*** element one step left (west)
  - Move ***b*** element one step up (north)
- Output ***c*** **// this will be the output cell C[i,j]**

Assume that the Processes are Logically Arranged as a Torus (wrap-around mesh)



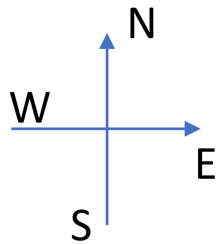
- Let us assume we have $n^2=p$
- Identify processes by their tile coordinate:
  - E.g., *(i,j)* is the label of the process in row *i* and column *j* on the Torus

---

- <u>Strategy:</u>
  - *C[i][j]* will be generated by process *(i,j)*

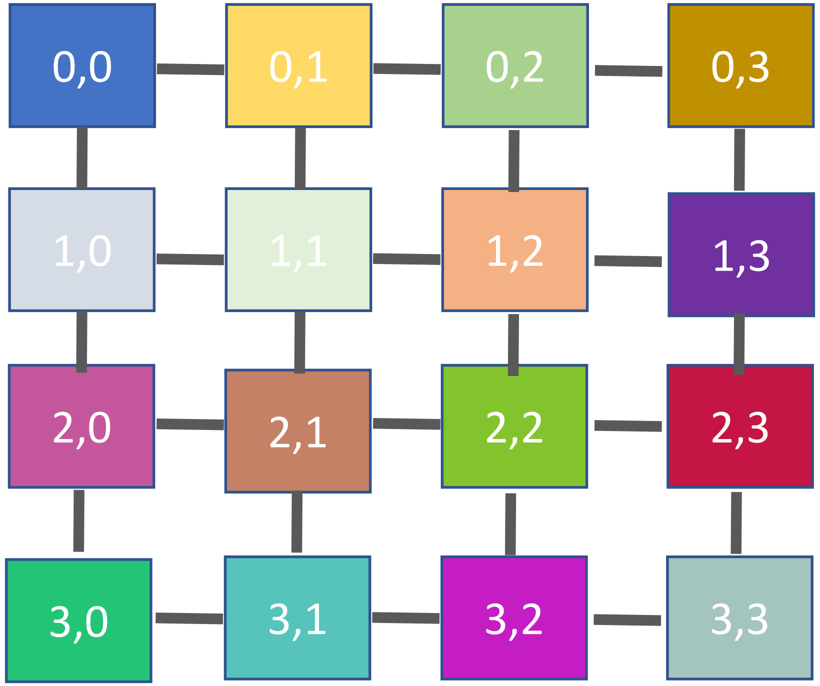
e.g.,
Proc. (2,1) will generate output value C[2][1]

P = 16 = 4x4

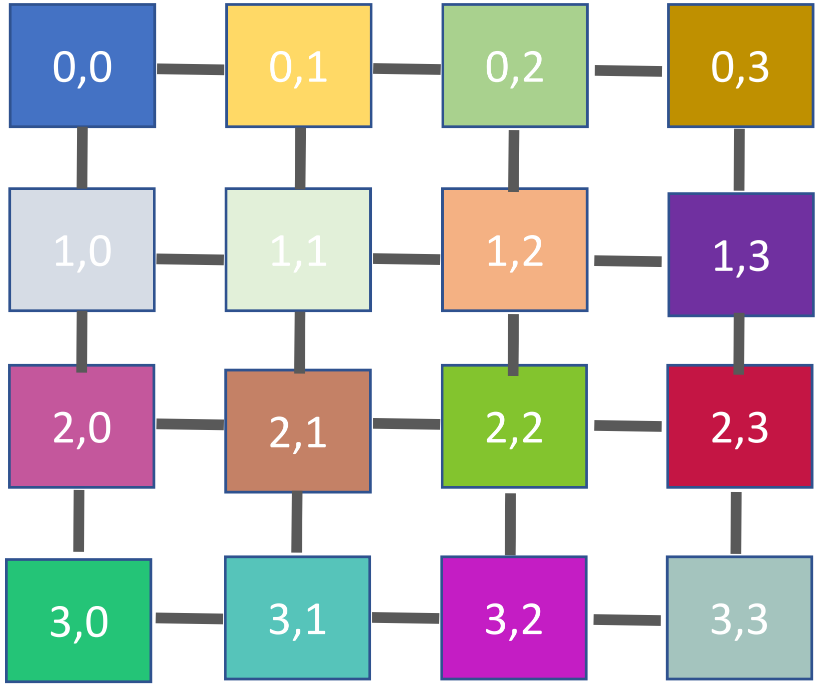


Load the matrices A and B are loaded such that cell (i,j) resides on proc. (i,j)

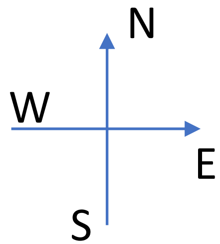
A

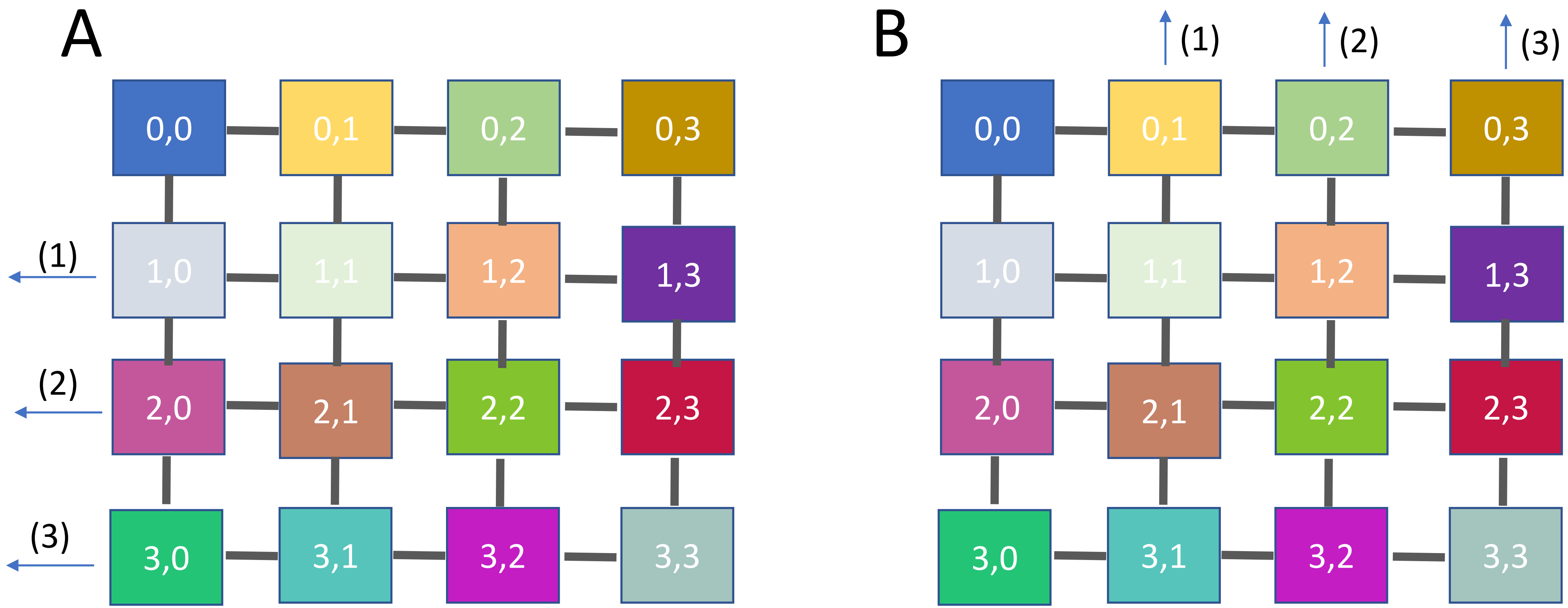


B



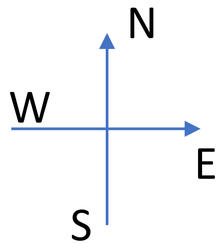
Move row i of A, i times West (circular left shift)
Move col. j of B, j times North (circular up shift)



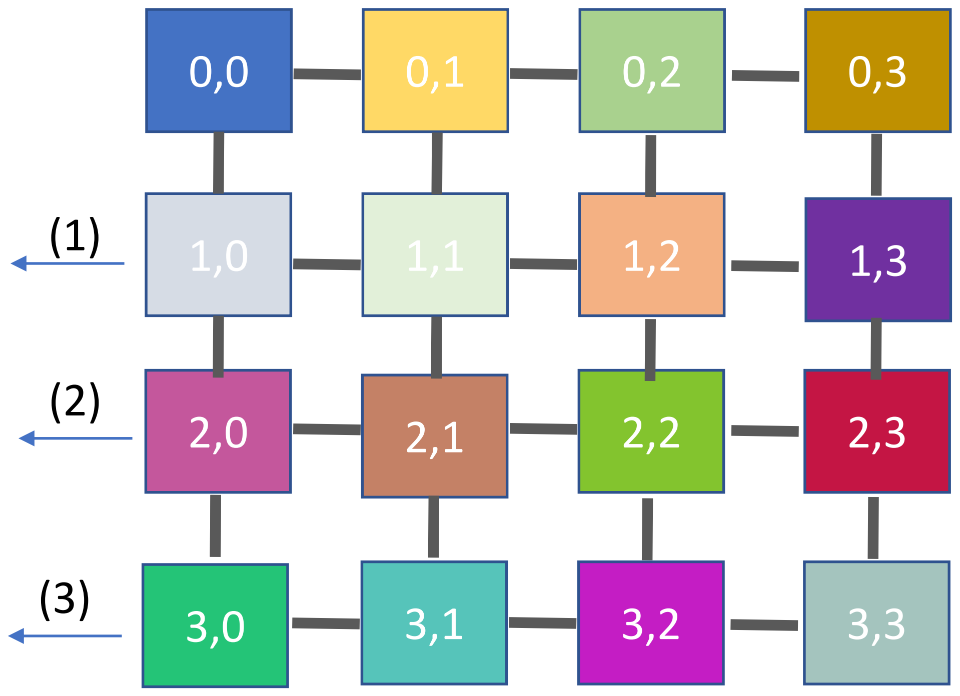


Move row i of A, i times West (circular left shift)

Move col. j of B, j times North (circular up shift)

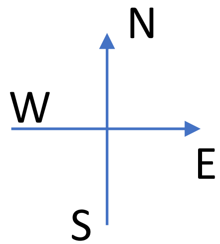


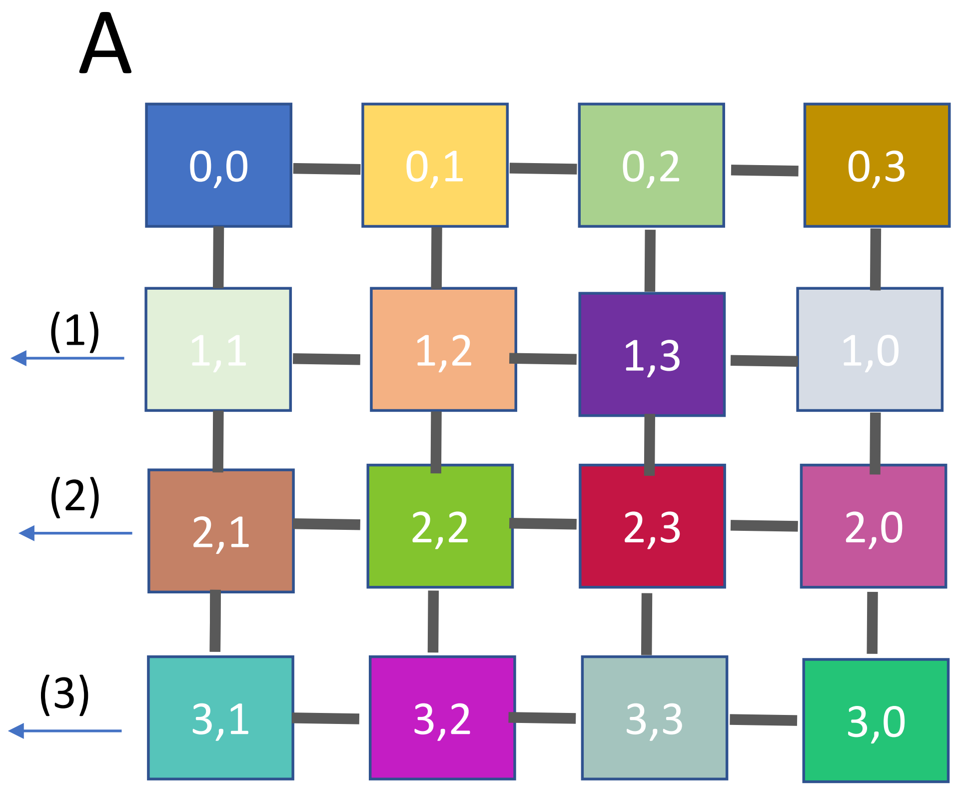
A



Move row i of A, i times West (circular left shift)

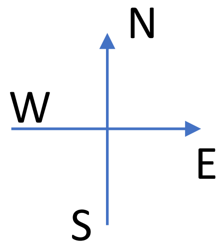
Move col. j of B, j times North (circular up shift)

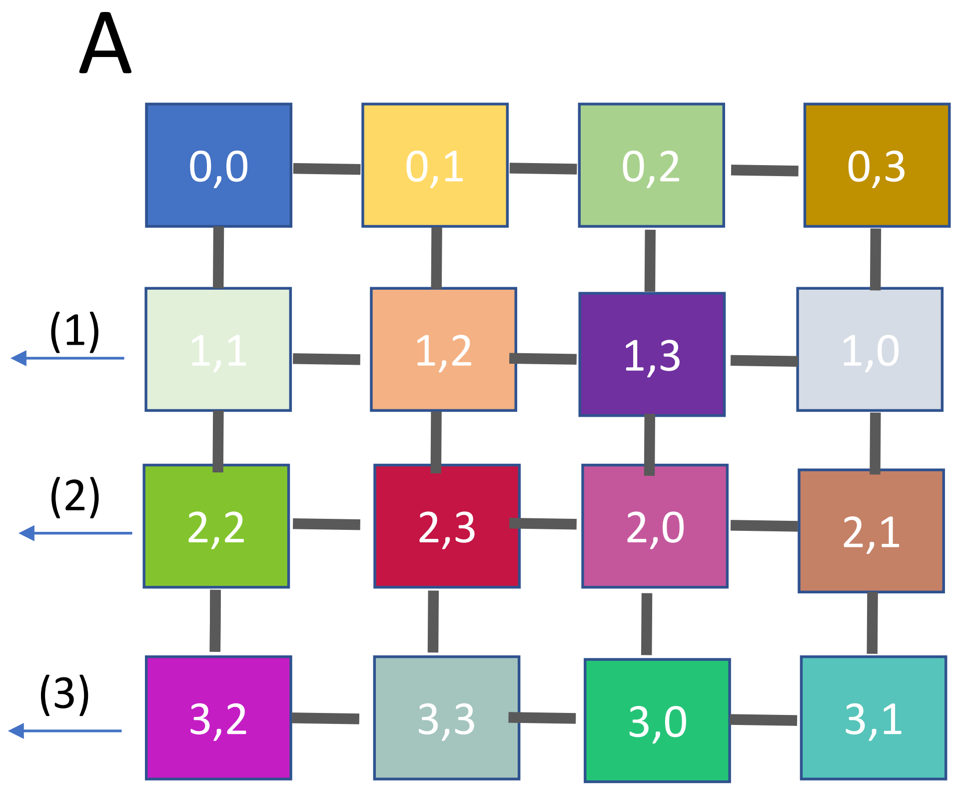




Move row i of A, i times West (circular left shift)

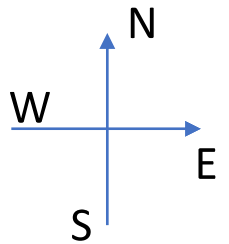
Move col. j of B, j times North (circular up shift)



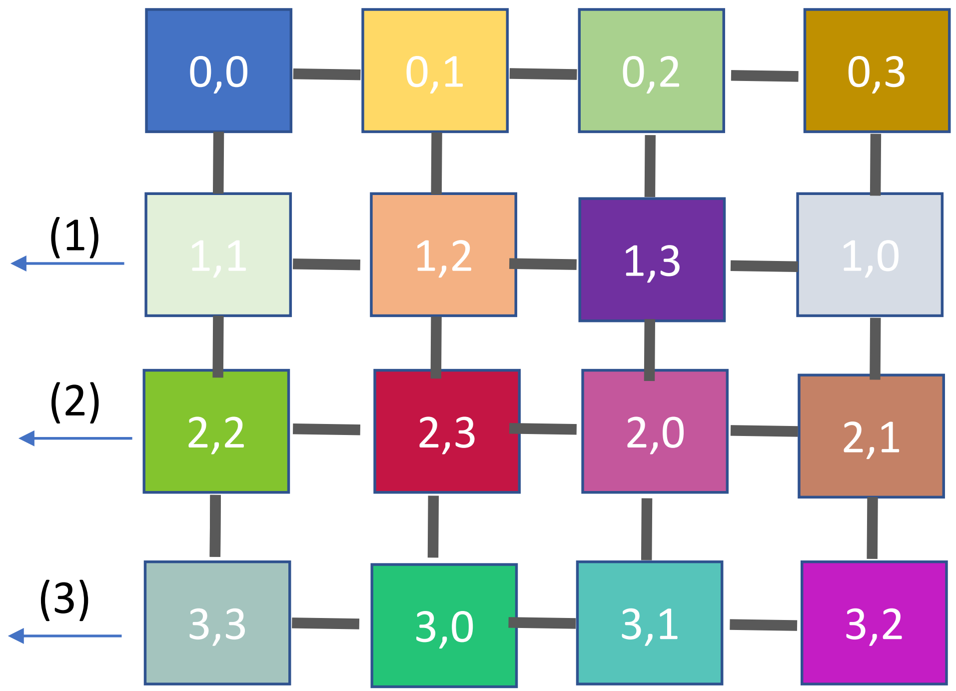


Move row i of A, i times West (circular left shift)

Move col. j of B, j times North (circular up shift)

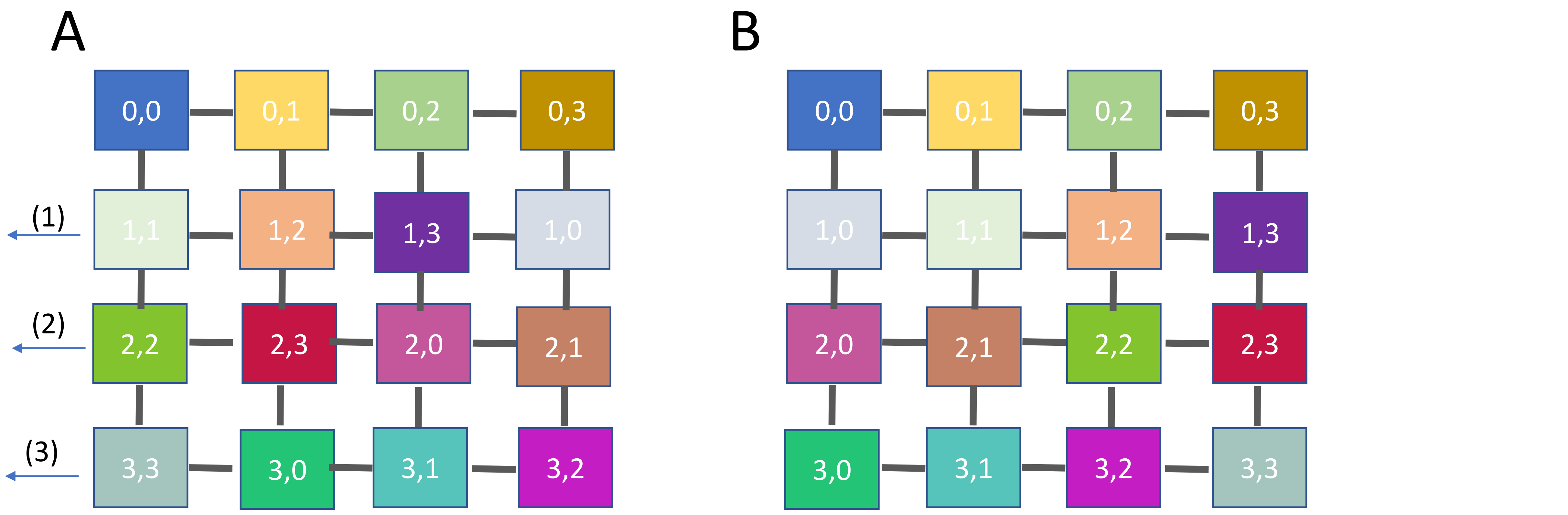


A



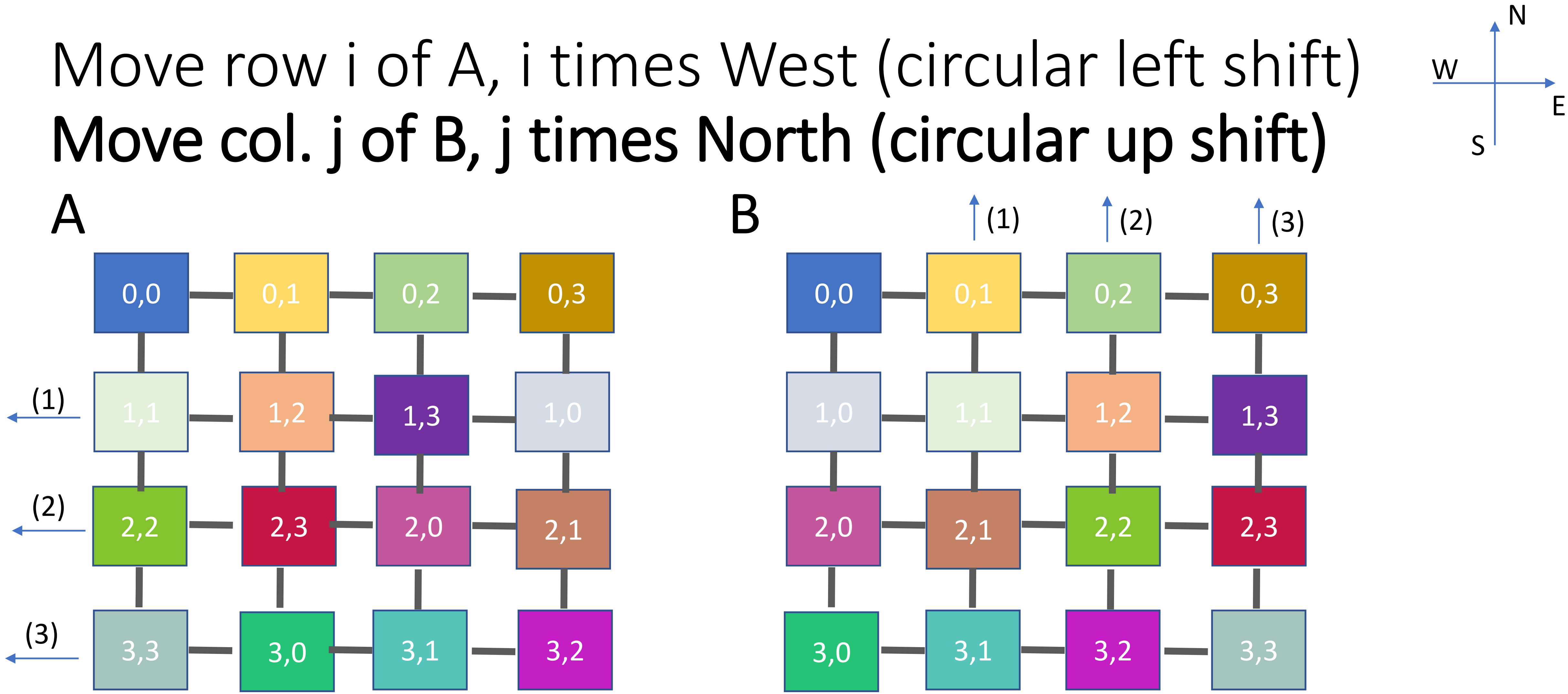
Move row i of A, i times West (circular left shift)

Move col. j of B, j times North (circular up shift)



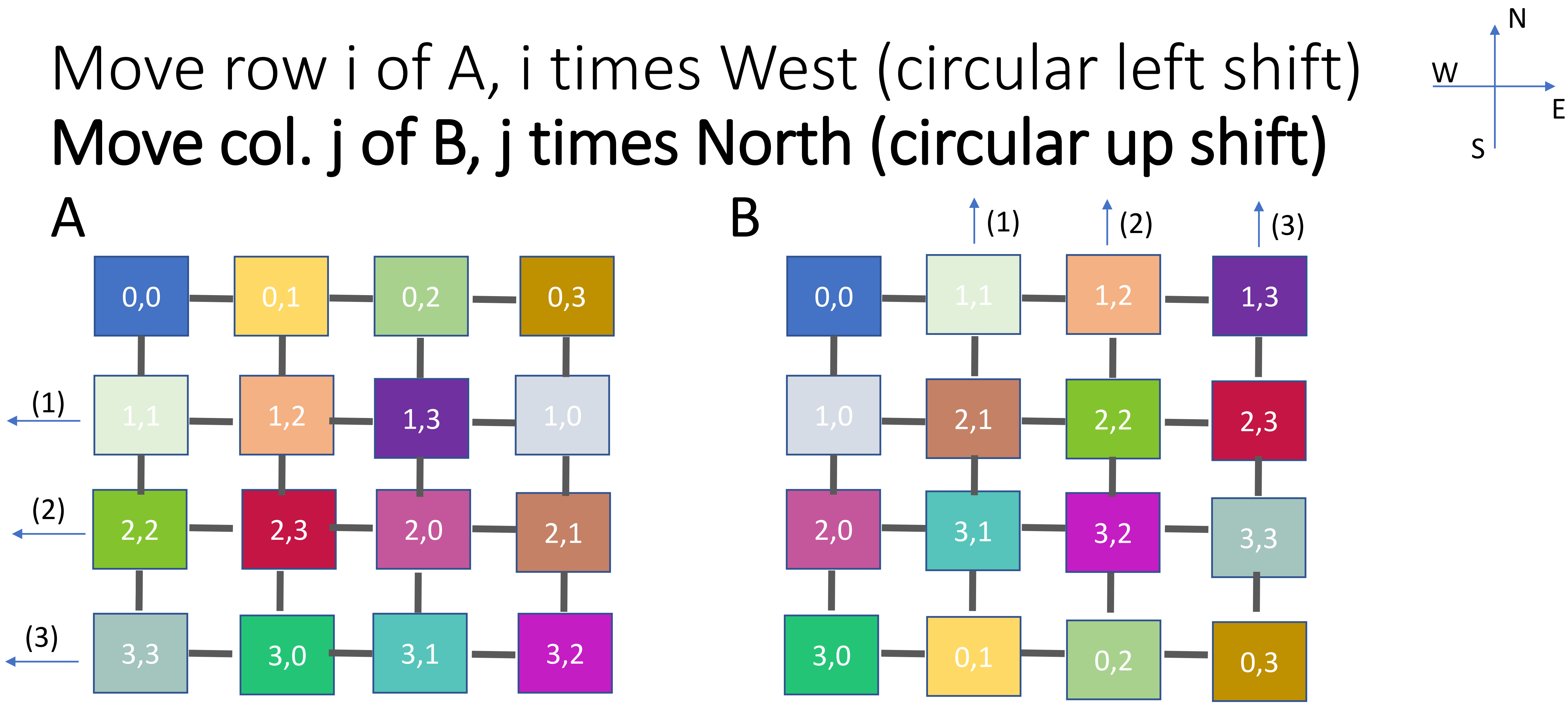
Move row i of A, i times West (circular left shift)

Move col. j of B, j times North (circular up shift)



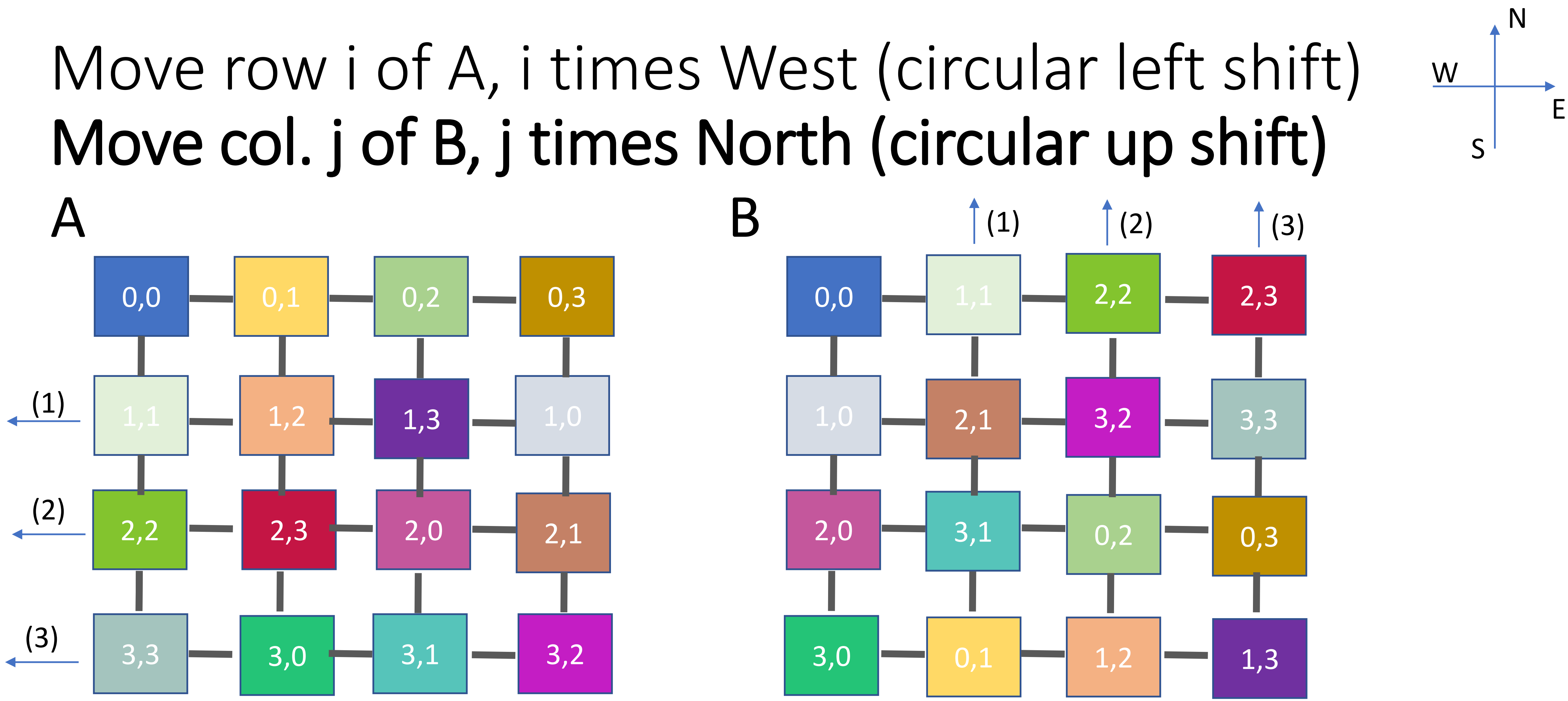
Move row i of A, i times West (circular left shift)

Move col. j of B, j times North (circular up shift)



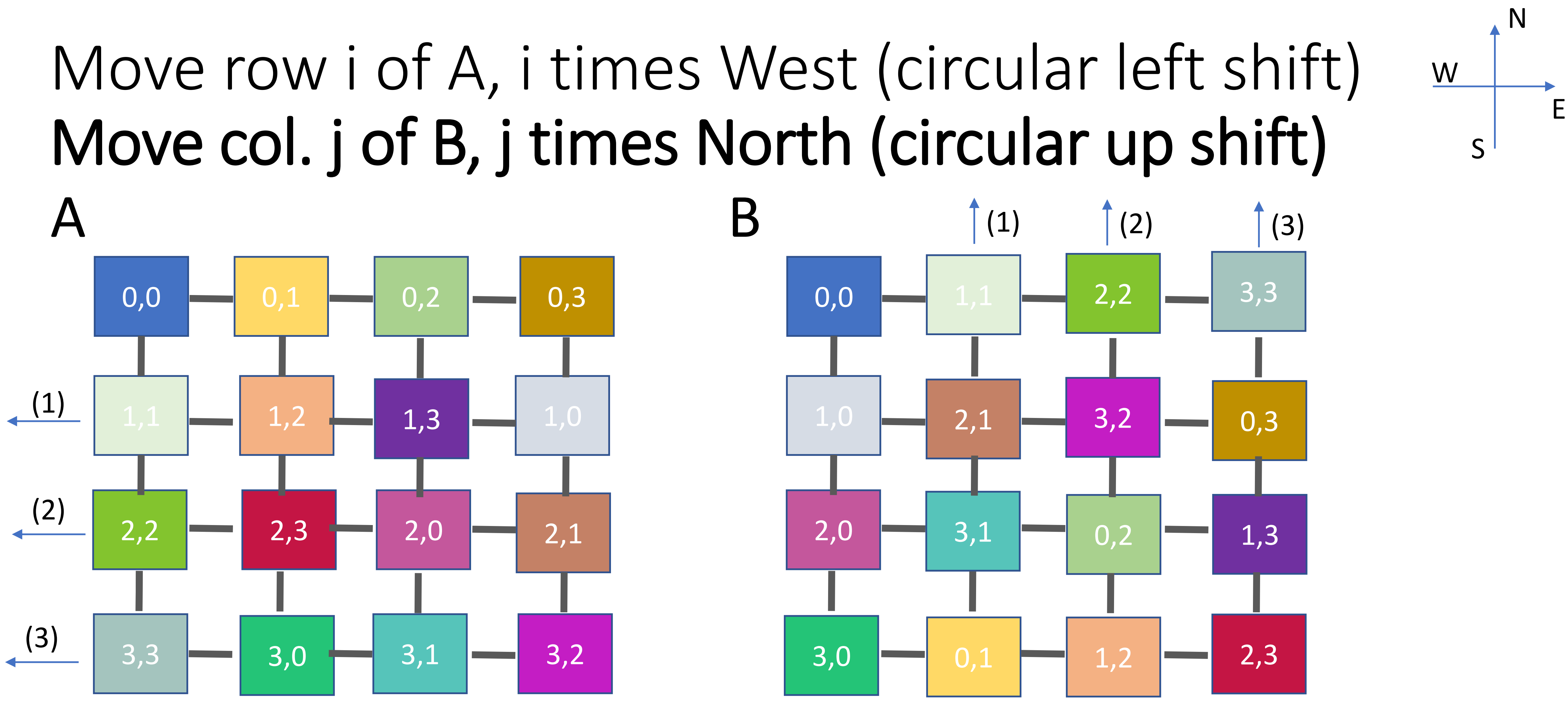
Move row i of A, i times West (circular left shift)

Move col. j of B, j times North (circular up shift)



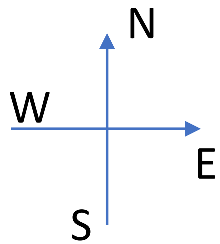
Move row i of A, i times West (circular left shift)

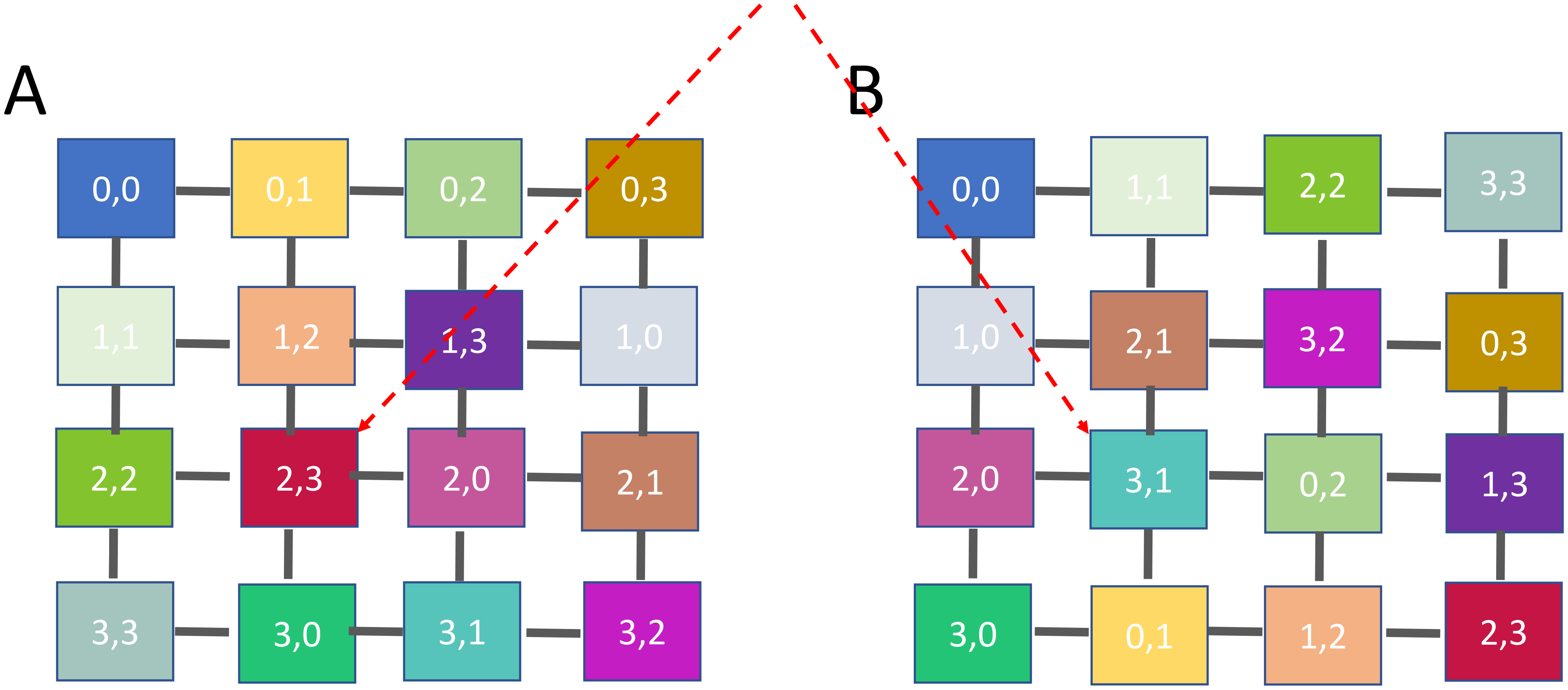
Move col. j of B, j times North (circular up shift)



Main Algorithm

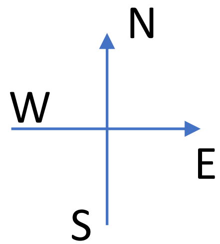
Let us focus on the computation of C[2][1].
Tile (2,1) will compute it.

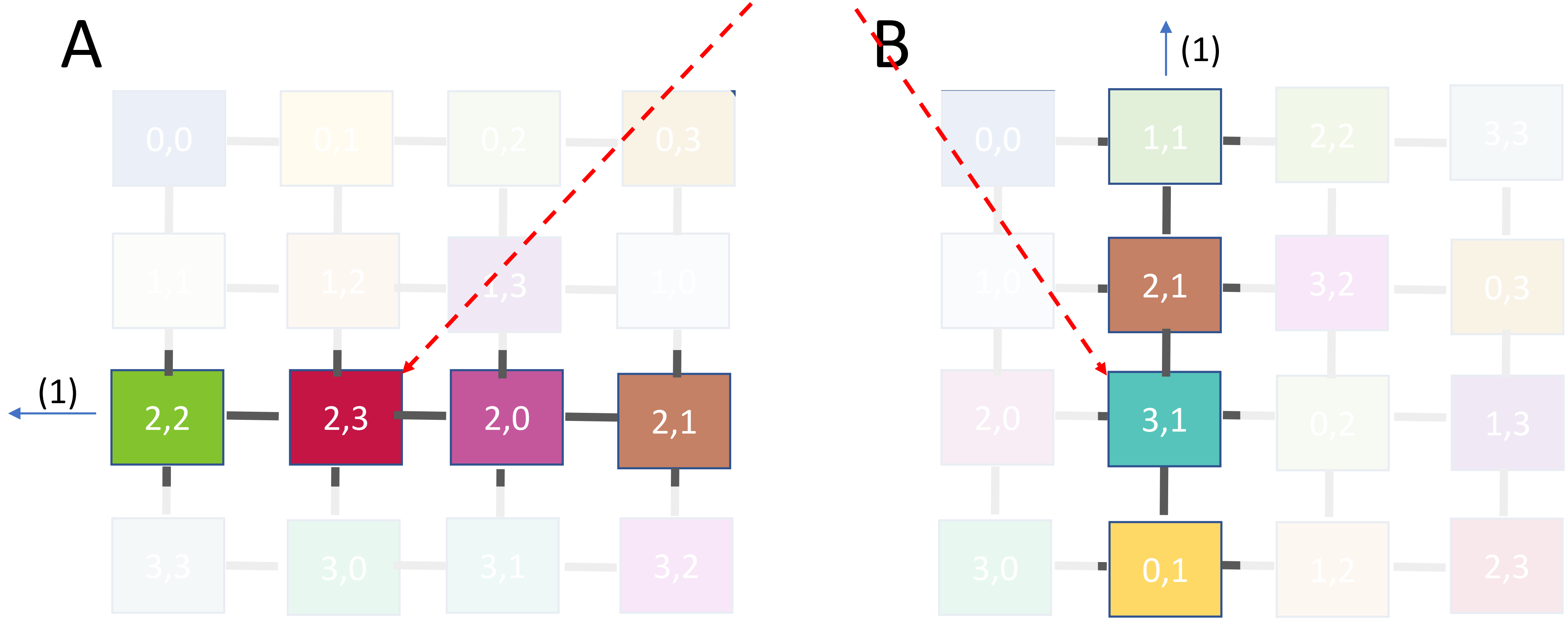




Main Algorithm

Let us focus on the computation of C[2][1].
Tile (2,1) will compute it.

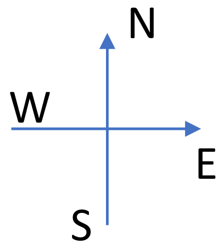




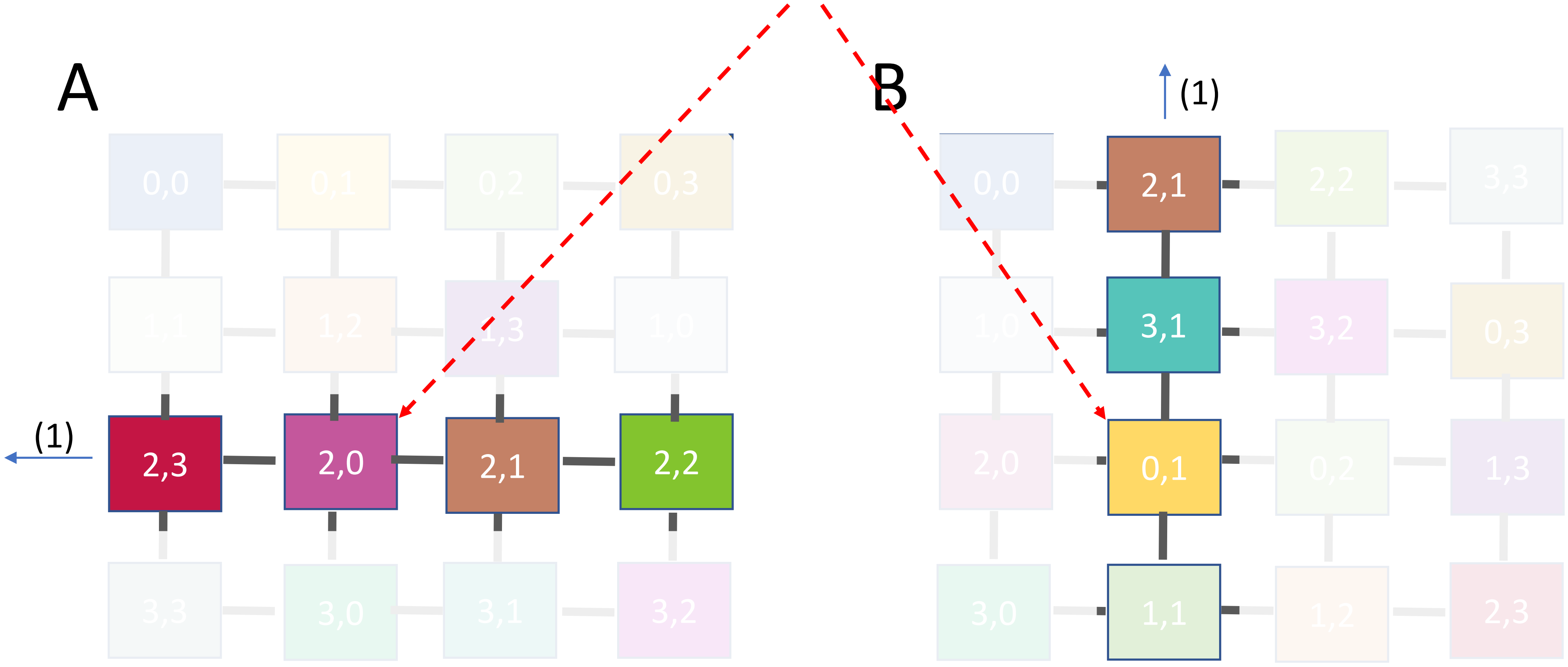
$$C[2][1] = A[2][3] \times B[3][1]$$

STEP 1

Main Algorithm



Let us focus on the computation of C[2][1].
Tile (2,1) will compute it.

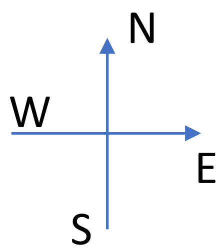


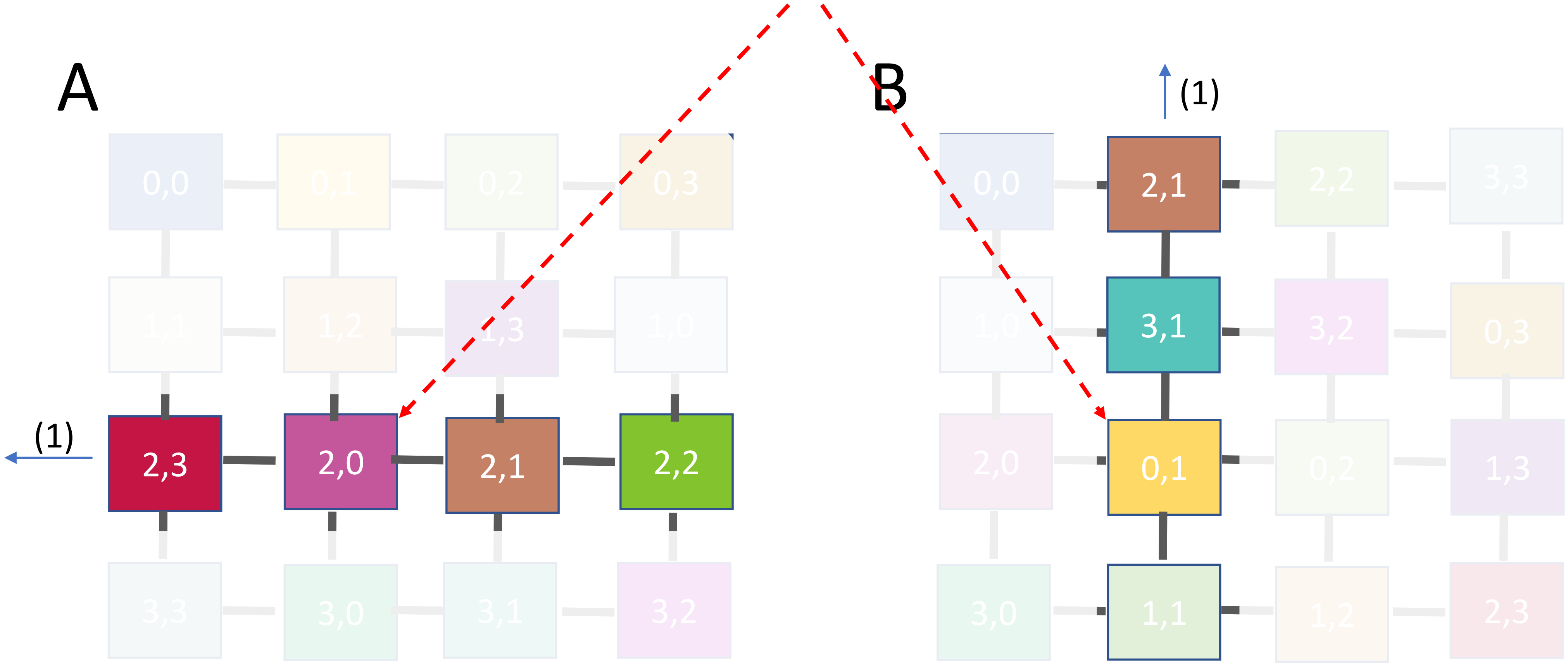
$$C[2][1] = A[2][3] \times B[3][1]$$

STEP 2

Main Algorithm

Let us focus on the computation of C[2][1].
Tile (2,1) will compute it.



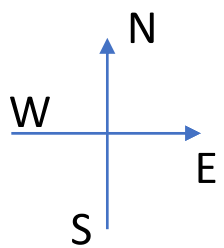


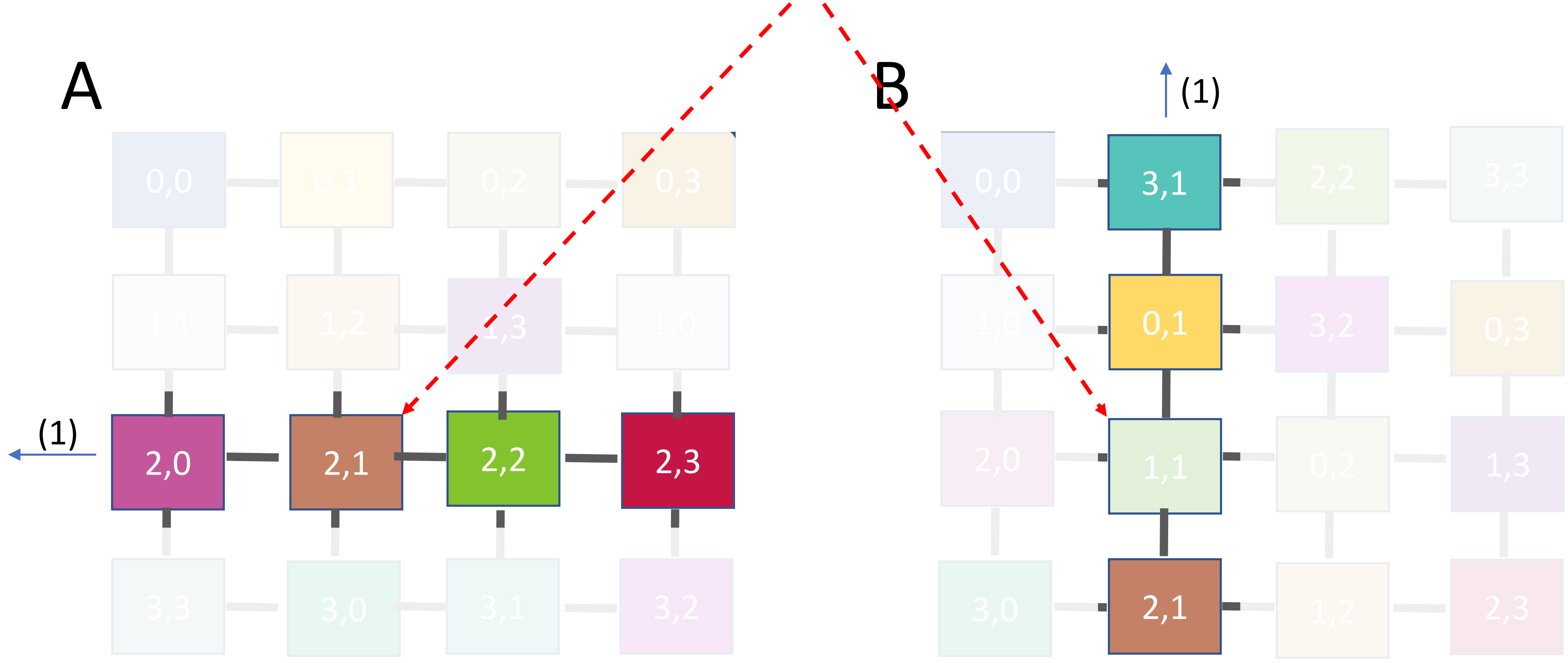
$$C[2][1] = A[2][3] \times B[3][1] + A[2][0] \times B[0][1]$$

STEP 2

Main Algorithm

Let us focus on the computation of C[2][1].
Tile (2,1) will compute it.



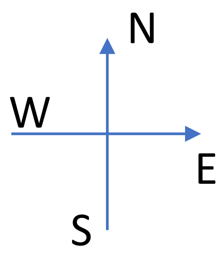


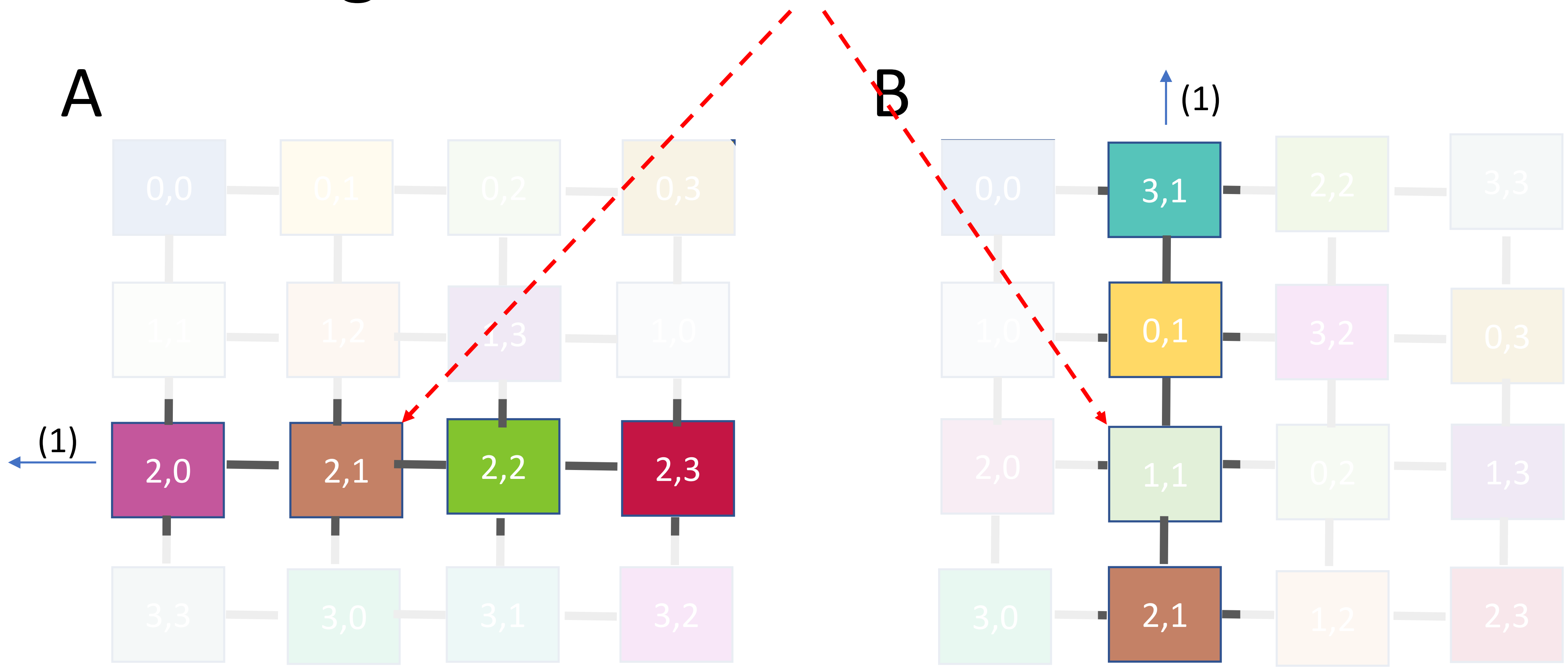
$$C[2][1] = A[2][3] \times B[3][1] + A[2][0] \times B[0][1]$$

STEP 3

Main Algorithm

Let us focus on the computation of C[2][1].
Tile (2,1) will compute it.



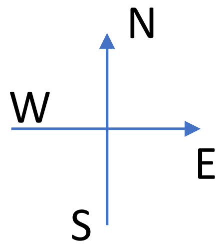


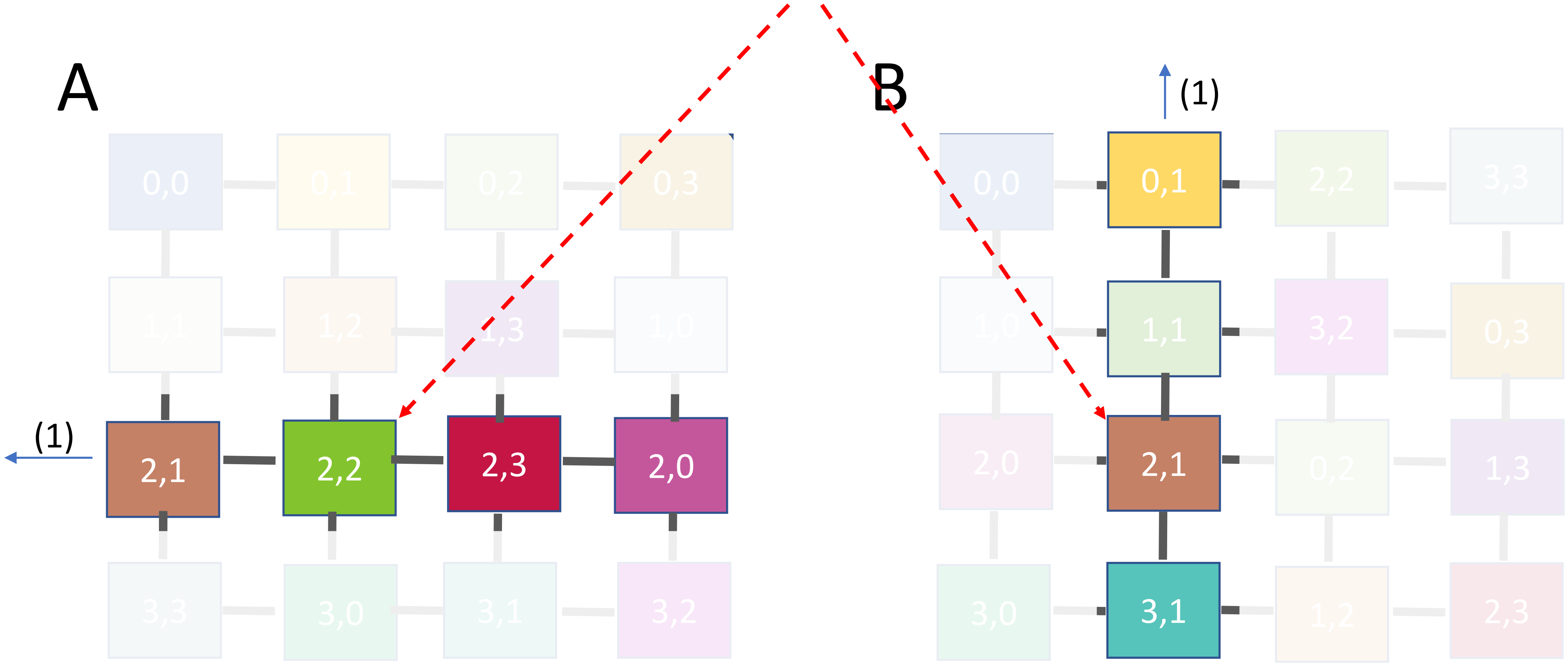
$$C[2][1] = A[2][3] \times B[3][1] + A[2][0] \times B[0][1] + A[2][1] \times B[1][1]$$

STEP 3

Main Algorithm

Let us focus on the computation of C[2][1].
Tile (2,1) will compute it.



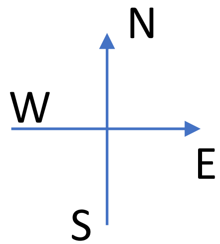


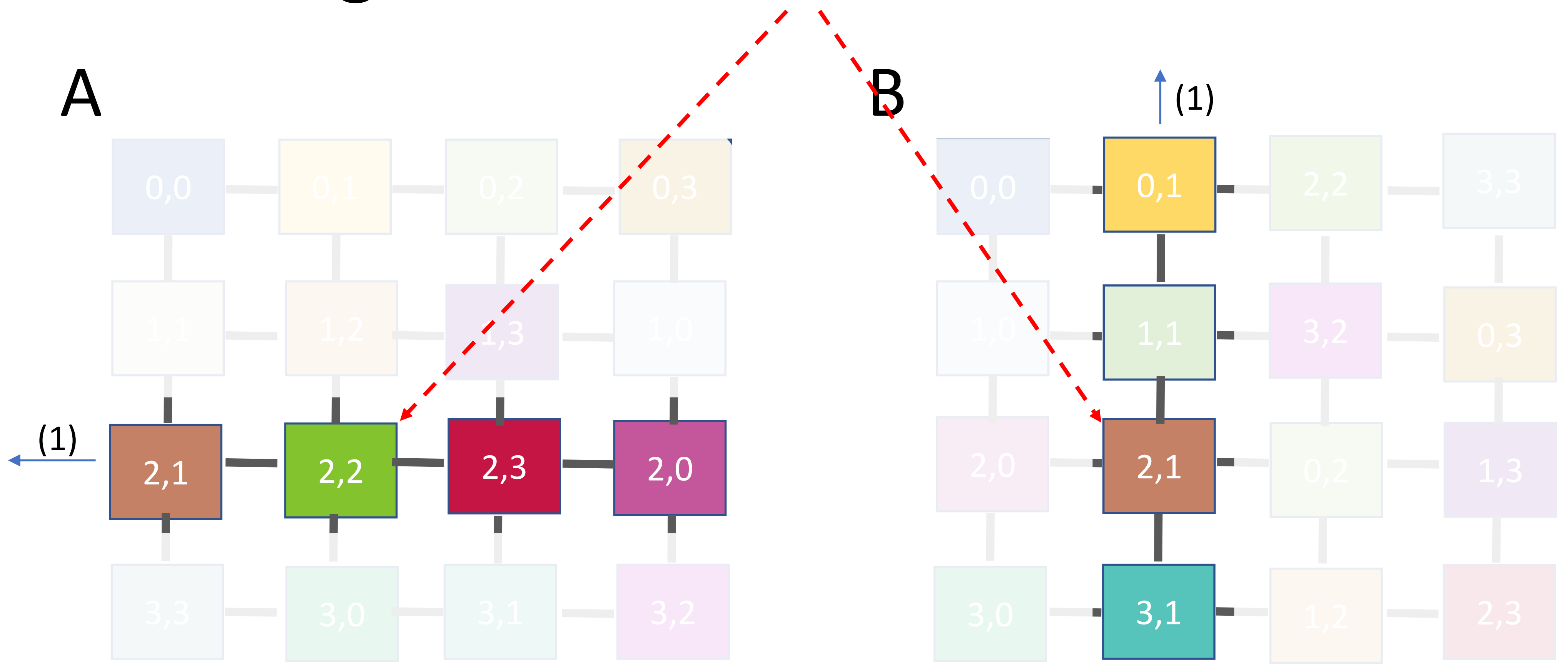
$$C[2][1] = A[2][3] \times B[3][1] + A[2][0] \times B[0][1] + A[2][1] \times B[1][1]$$

STEP 4

Main Algorithm

Let us focus on the computation of C[2][1].
Tile (2,1) will compute it.





$$C[2][1] = A[2][3] \times B[3][1] + A[2][0] \times B[0][1] + A[2][1] \times B[1][1] + A[2][2] \times B[2][1]$$

Proc (2,1) outputs C[2][1]

STEP 4