Data Structures – CST 201
Module - 2

Syllabus

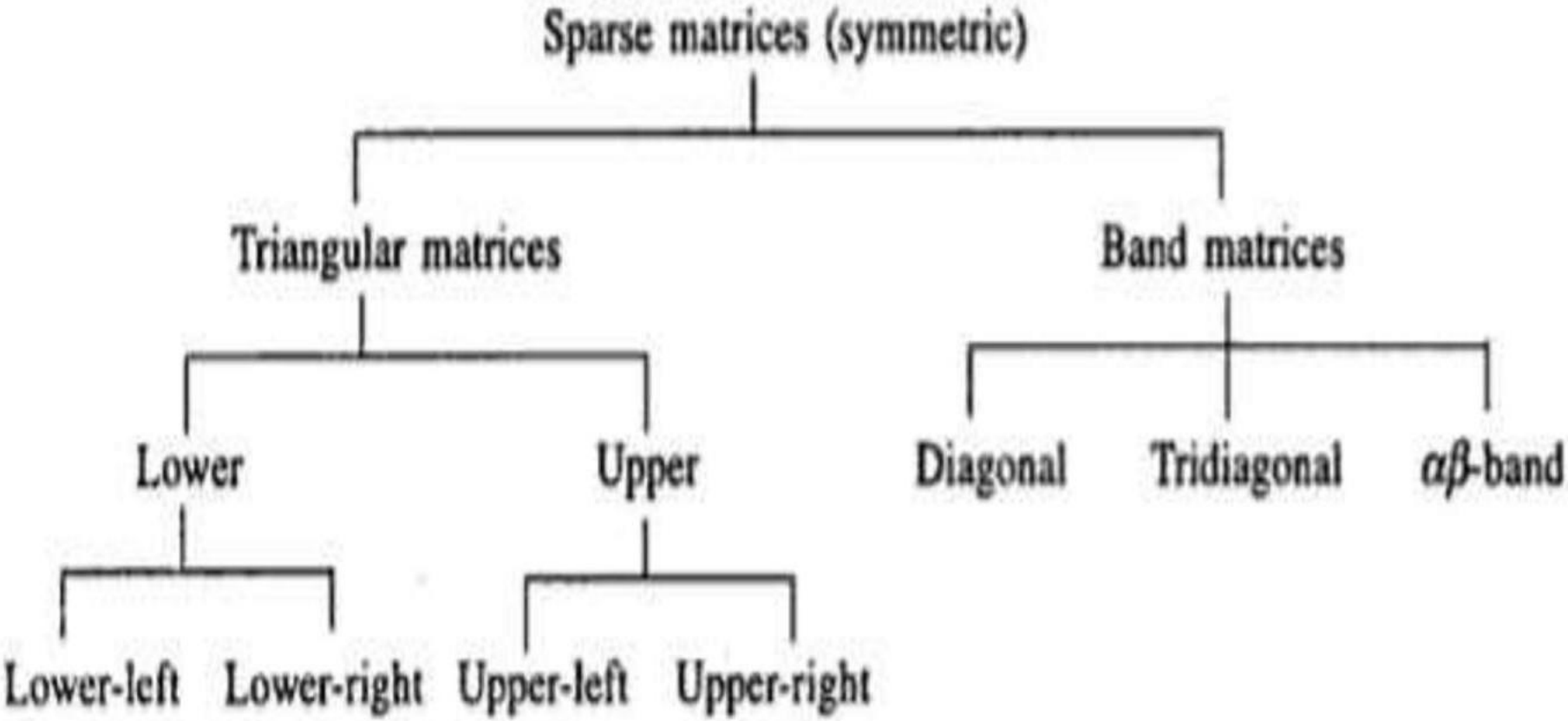
- Searching
  - Linear Search
  - Binary Search
- Polynomial representation using Arrays
- **Sparse matrix**
- Stacks
- Queues
  - Circular Queues
  - Double Ended Queues
  - Priority Queues
- Evaluation of Expressions

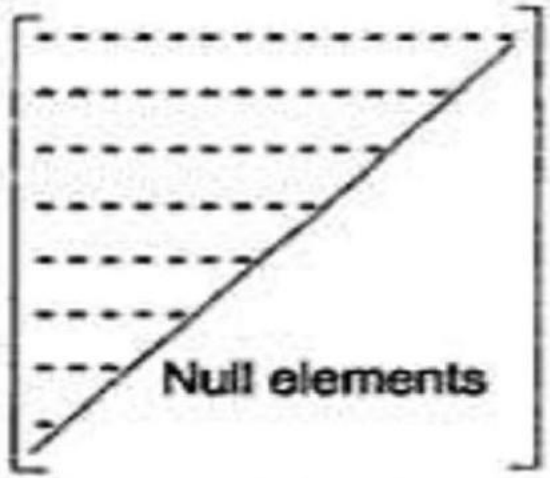
SPARSE MATRCES

- **Matrix:** It is a two-dimensional data object made of m rows and n columns, therefore having total m x n values.

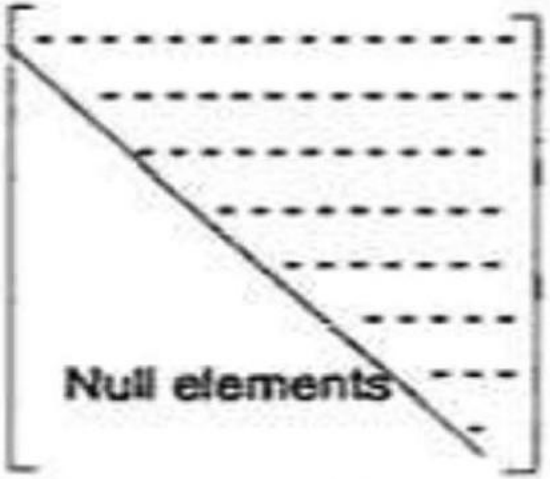
- **Sparse Matrix:** Most of the elements of the matrix are 0
- **Dense Matrix:** Most of the elements of the matrix are nonzero

SPARSE MATRCES

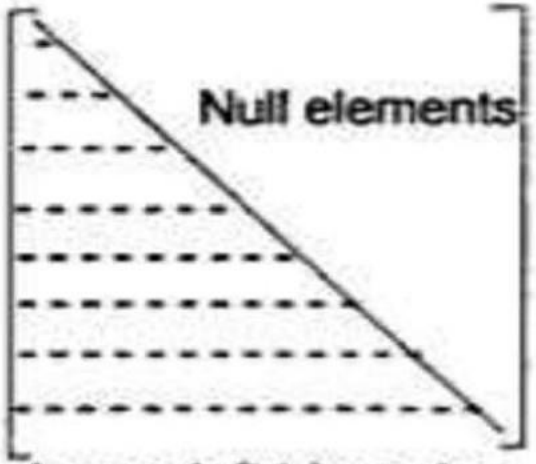




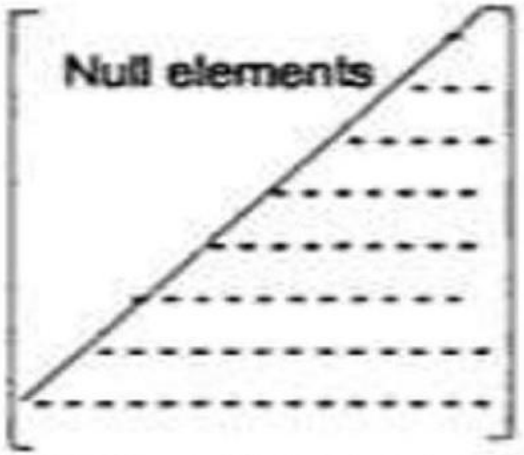
Upper-left triangular



Upper-right triangular

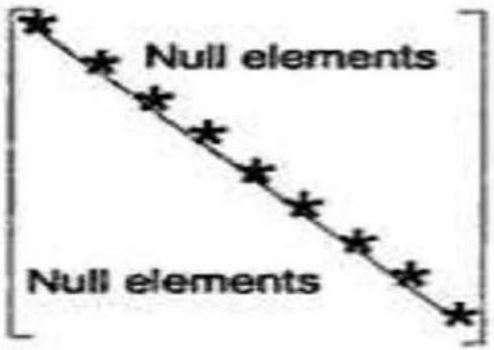


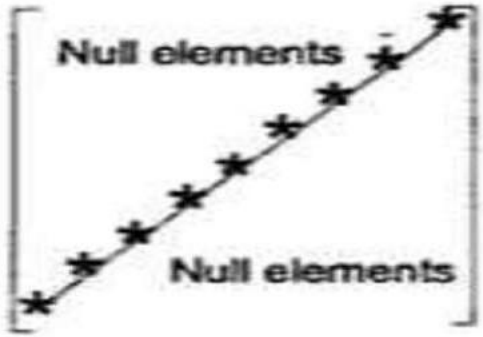
Lower-left triangular



Lower-right triangular

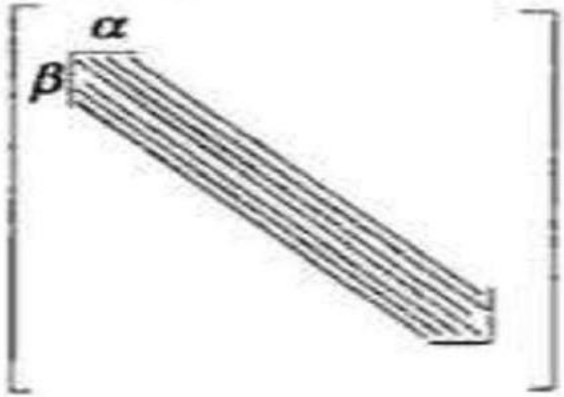
(a) Various triangular matrices





Diagonal matrices

Tridiagonal matrices

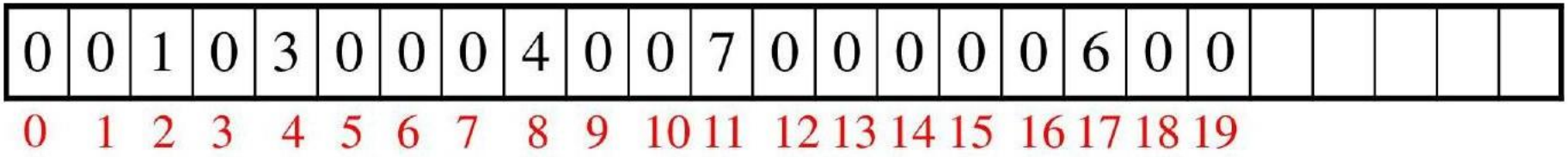


α β band Matrix

SPARSE MATRCES

- **Matrix Representation Using Array**

	0	1	2	3	4
0	0	0	1	0	3
1	0	0	0	4	0
2	0	7	0	0	0
3	0	0	6	0	0



- Storing of null element is wastage of memory

SPARSE MATRCES

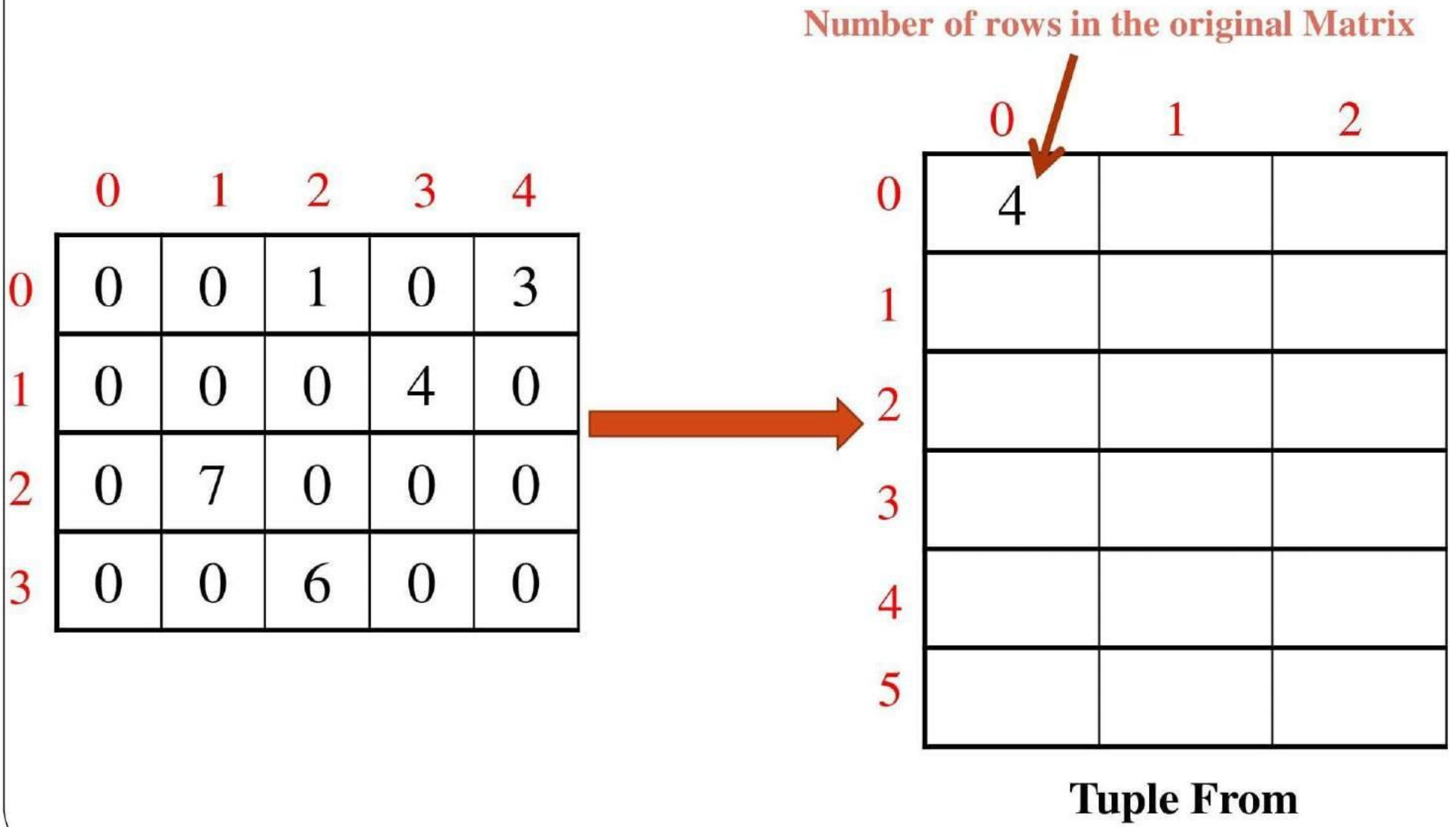
- **Sparse Matrix Representation(as Tuple)**
  - There are n+1 number of rows and 3 columns, where n is the total number of non-zero elements in the given matrix.
  - The first row is a triple- **(number of rows, number of columns, total number of nonzero elements)**
  - The remaining n rows are used to store all nonzero elements. The non-zero elements are stored as a **triples- (row, column, value)**
    - **row**- The row in which the element is reside
    - **column**- The column in which the element is reside
    - **value**- The element itself

SPARSE MATRCES

- **Sparse Matrix Representation(as Tuple)**
  - It is stored in the **row major form**.
    - In the ascending order of rows.
    - Inside each row, store elements in the ascending order of columns

SPARSE MATRCES

- **Sparse Matrix Representation(as Tuple)**



SPARSE MATRCES

- **Sparse Matrix Representation(as Tuple)**

	0	1	2	3	4
0	0	0	1	0	3
1	0	0	0	4	0
2	0	7	0	0	0
3	0	0	6	0	0

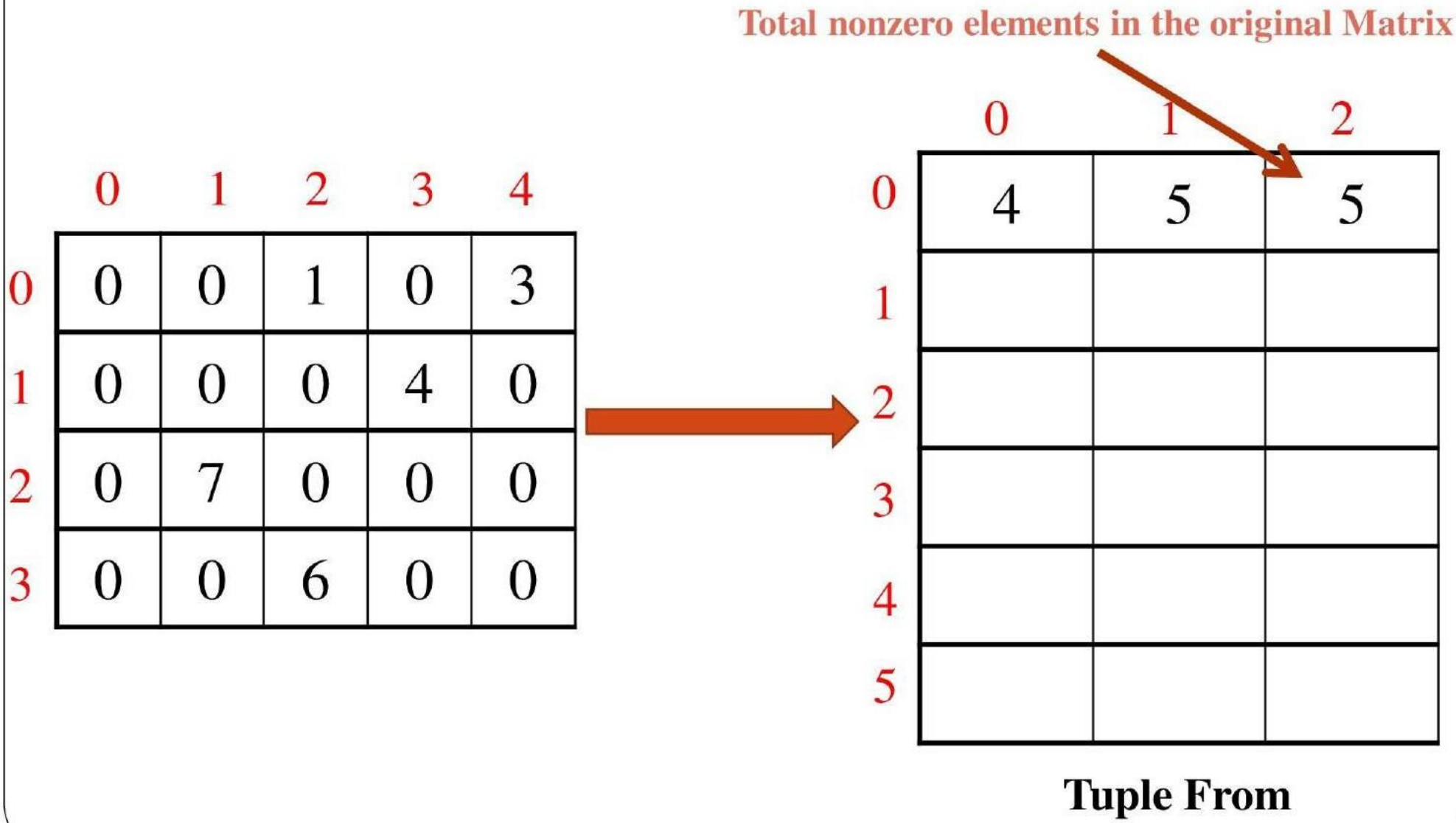
Number of columns in the original Matrix

	0	1	2
0	4	5	
1			
2			
3			
4			
5			

Tuple From

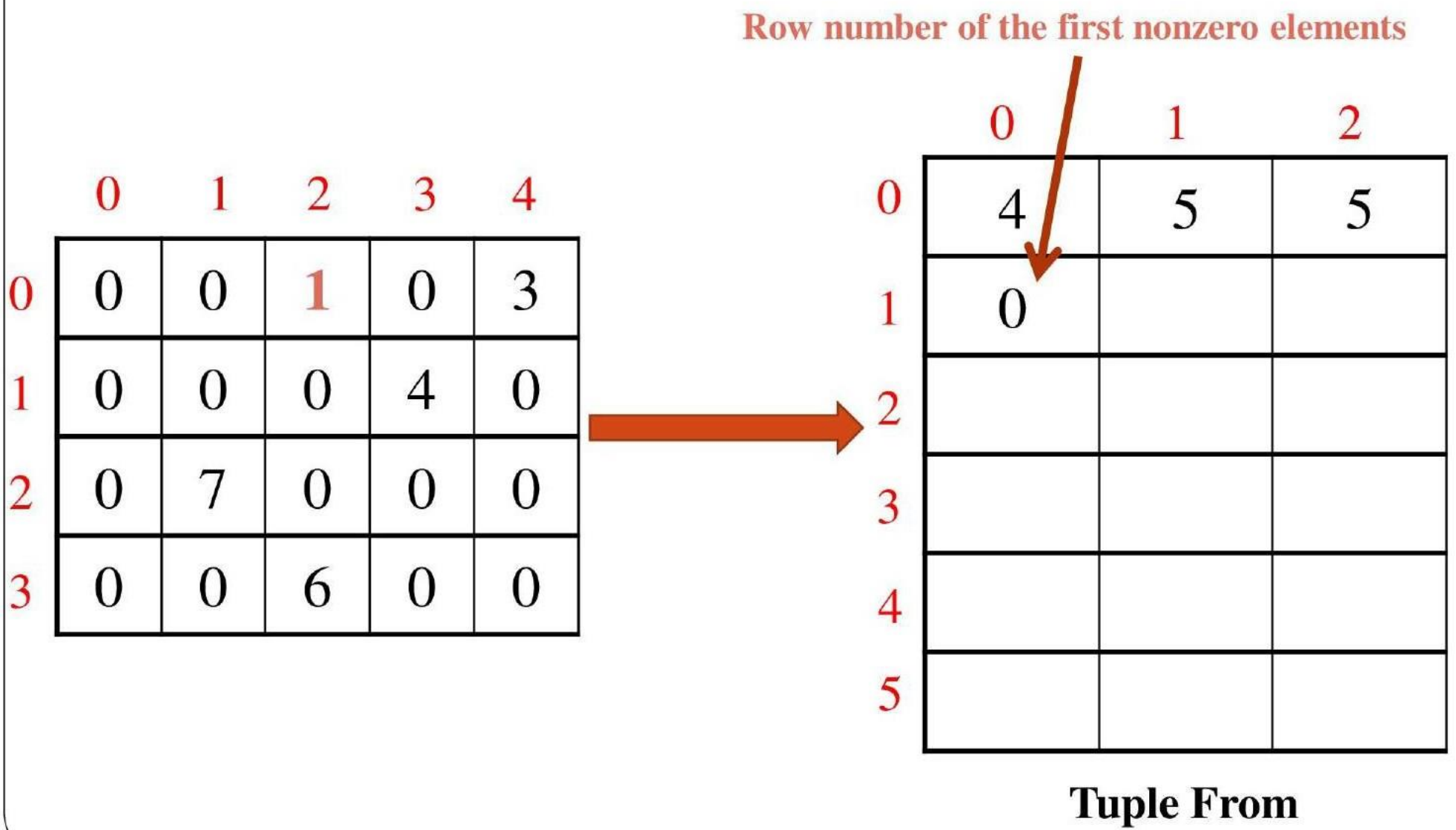
SPARSE MATRCES

- **Sparse Matrix Representation(as Tuple)**



SPARSE MATRCES

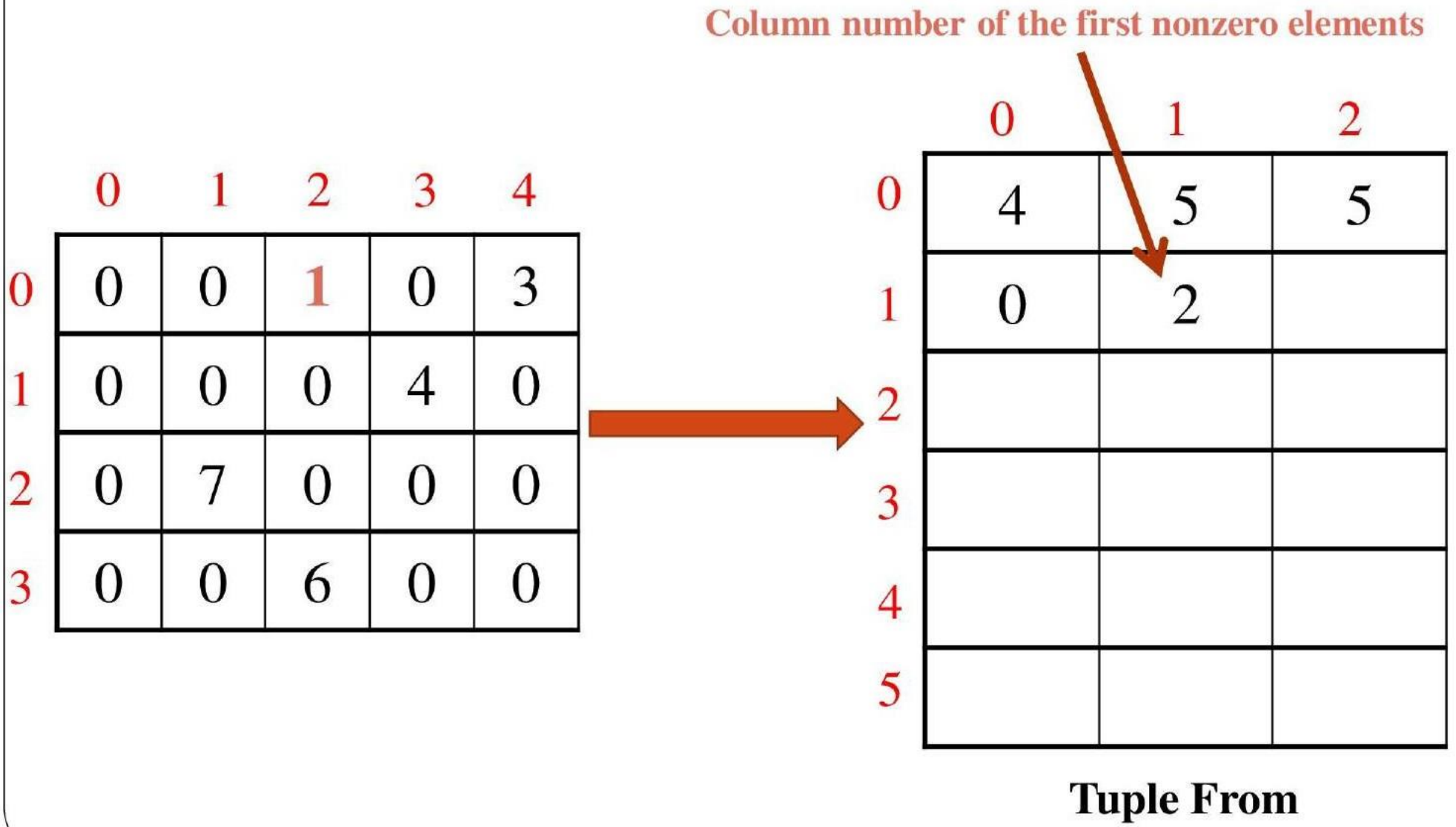
- **Sparse Matrix Representation(as Tuple)**



Tuple From

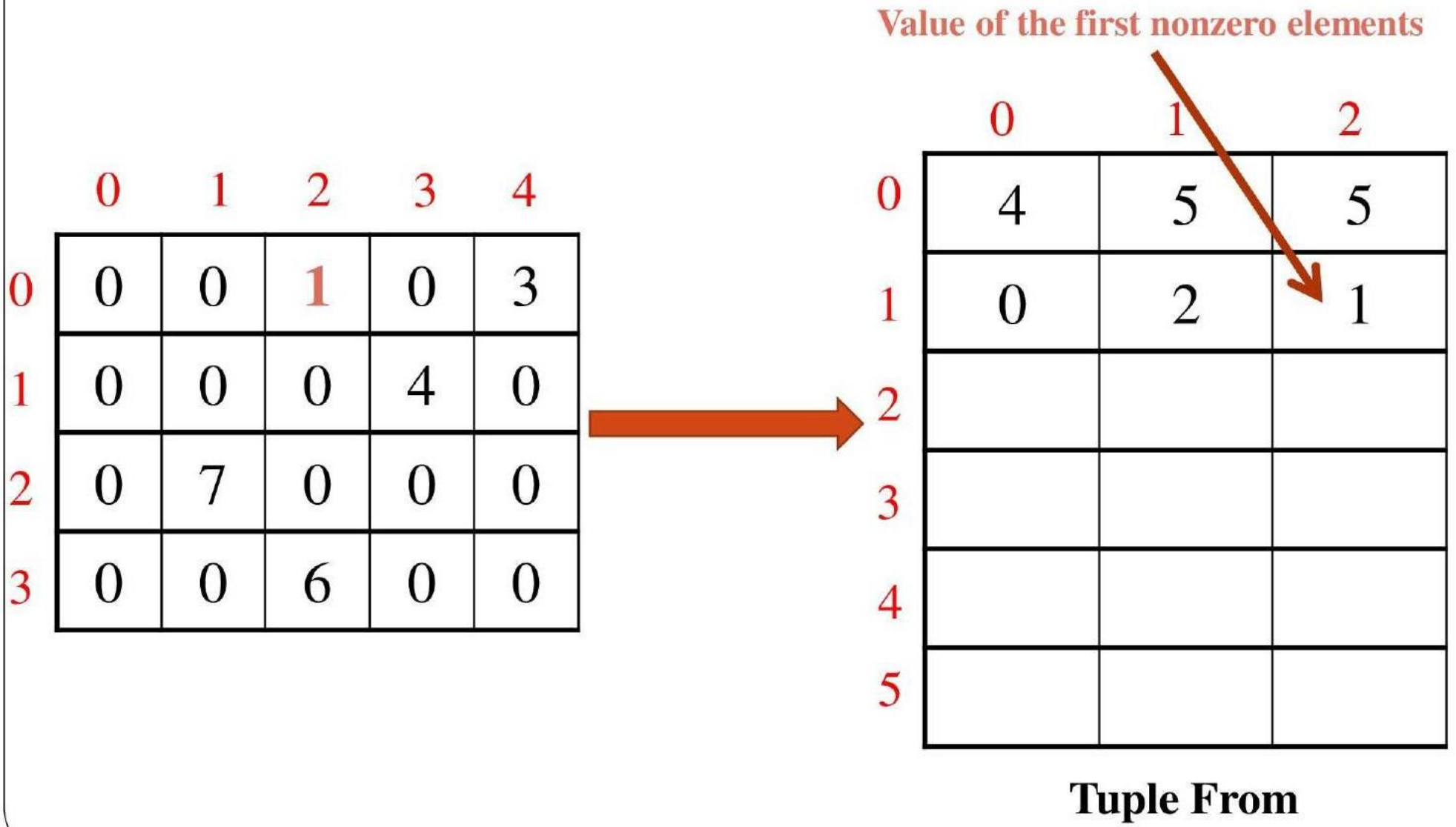
SPARSE MATRCES

- **Sparse Matrix Representation(as Tuple)**



SPARSE MATRCES

- **Sparse Matrix Representation(as Tuple)**



SPARSE MATRCES

- **Sparse Matrix Representation(as Tuple)**

	0	1	2	3	4
0	0	0	1	0	3
1	0	0	0	4	0
2	0	7	0	0	0
3	0	0	6	0	0

	0	1	2
0	4	5	5
1	0	2	1
2	0	4	3
3	1	3	4
4	2	1	7
5	3	2	6

Tuple From

SPARSE MATRCES

- **Why to use Sparse Matrix instead of simple matrix ?**
  - **Storage:** There are lesser non-zero elements than zeros and thus lesser memory can be used to store only those elements.
  - **Computing time:** Computing time can be saved by logically designing a data structure traversing only non-zero elements..

Sparse Matrix Representation as Tuple

Algorithm Sparse(A,m,n)

```
{       S[0][0]=m;          S[0][1]=n;          k=1;
        for i=0 to m-1 do
        {       for j=0 to n-1 do
                {       if  A[i][j] != 0 then
                        {       S[k][0] = i;            S[k][1] = j;
                                S[k][2] = A[i][j]
                                k=k+1
                        }
                }
        }
        S[0][2] = k-1;
}
```

SPARSE MATRIX TUPLE FORMATION - PROGRAM

```
#include<stdio.h>
void Sparse(int A[10][10],int m,int n)
{  int i, j, k=1, S[10][10];
   S[0][0]=m;   S[0][1]=n;
   for (i=0; i<m; i++)
   {      for (j=0; j<n; j++)
          {          if (A[i][j] != 0)
                     {          S[k][0] = i;
                                S[k][1] = j;
                                S[k][2] = A[i][j];
                                k++;
                     }
          }
   }
   S[0][2] = k-1;
   printf("\nTuple Form:\n");
   for (i=0; i<=S[0][2]; i++)
   {      for (j=0; j<3; j++)
          {
                printf("%d\t",S[i][j]);
          }
          printf("\n");
   }
}
```

```
void main()
{  int i,j,A[10][10],m,n;
   printf("Enter the number of rows and columns of the matrix: ");
   scanf("%d%d",&m,&n);
   printf("Enter the matrix:");
   for(i=0;i<m;i++)
         for(j=0;j<n;j++)
               scanf("%d",&A[i][j]);
   printf("The given matrix is:\n");
   for (i=0; i<m; i++)
         for (j=0; j<n; j++)
               printf("%d\t",A[i][j]);
         printf("\n");
   Sparse(A,m,n);
}
```

TRANSPOSE OF A SPARSE MATRCES

	0	1	2	3	4
0	0	0	1	0	3
1	0	0	0	4	0
2	0	7	0	0	0
3	0	0	6	0	0

	0	1	2	3
0	0	0	0	0
1	0	0	7	0
2	1	0	0	6
3	0	4	0	0
4	3	0	0	0

TRANSPOSE OF A SPARSE MATRCES

	0	1	2
0	**4**	**5**	**5**
1	0	2	1
2	0	4	3
3	1	3	4
4	2	1	7
5	3	2	6

	0	1	2
0			
1			
2			
3			
4			
5			

TRANSPOSE OF A SPARSE MATRCES

	0	1	2
0	**4**	**5**	**5**
1	0	2	1
2	0	4	3
3	1	3	4
4	2	1	7
5	3	2	6

	0	1	2
0	5	4	5
1			
2			
3			
4			
5			

TRANSPOSE OF A SPARSE MATRCES

	0	1	2
0	4	5	5
1	0	**2**	1
2	0	**4**	3
3	1	**3**	4
4	2	**1**	7
5	3	**2**	6

	0	1	2
0	5	4	5
1			
2			
3			
4			
5			

Select a column with value 0

TRANSPOSE OF A SPARSE MATRCES

	0	1	2
0	4	5	5
1	0	**2**	1
2	0	**4**	3
3	1	**3**	4
4	2	**1**	7
5	3	**2**	6

	0	1	2
0	5	4	5
1			
2			
3			
4			
5			

Select a column with value 1

TRANSPOSE OF A SPARSE MATRCES

	0	1	2
0	4	5	5
1	0	**2**	1
2	0	**4**	3
3	1	**3**	4
4	**2**	**1**	**7**
5	3	**2**	6

	0	1	2
0	5	4	5
1			
2			
3			
4			
5			

Select a column with value 1

TRANSPOSE OF A SPARSE MATRCES

	0	1	2
0	4	5	5
1	0	**2**	1
2	0	**4**	3
3	1	**3**	4
4	**2**	**1**	**7**
5	3	**2**	6

	0	1	2
0	5	4	5
1	1	2	7
2			
3			
4			
5			

Select a column with value 1

TRANSPOSE OF A SPARSE MATRCES

	0	1	2
0	4	5	5
1	0	**2**	1
2	0	**4**	3
3	1	**3**	4
4	2	**1**	7
5	3	**2**	6

	0	1	2
0	5	4	5
1	1	2	7
2			
3			
4			
5			

Select a column with value 2

TRANSPOSE OF A SPARSE MATRCES

	0	1	2
0	4	5	5
1	**0**	**2**	**1**
2	0	**4**	3
3	1	**3**	4
4	2	**1**	7
5	3	**2**	6

	0	1	2
0	5	4	5
1	1	2	7
2			
3			
4			
5			

Select a column with value 2

TRANSPOSE OF A SPARSE MATRCES

	0	1	2
0	4	5	5
1	**0**	**2**	**1**
2	0	**4**	3
3	1	**3**	4
4	2	**1**	7
5	3	**2**	6

	0	1	2
0	5	4	5
1	1	2	7
2	2	0	1
3			
4			
5			

Select a column with value 2

TRANSPOSE OF A SPARSE MATRCES

	0	1	2
0	4	5	5
1	0	**2**	1
2	0	**4**	3
3	1	**3**	4
4	2	**1**	7
5	**3**	**2**	**6**

	0	1	2
0	5	4	5
1	1	2	7
2	2	0	1
3			
4			
5			

Select a column with value 2

TRANSPOSE OF A SPARSE MATRCES

	0	1	2
0	4	5	5
1	0	**2**	1
2	0	**4**	3
3	1	**3**	4
4	2	**1**	7
5	**3**	**2**	**6**

	0	1	2
0	5	4	5
1	1	2	7
2	2	0	1
3	2	3	6
4			
5			

Select a column with value 2

TRANSPOSE OF A SPARSE MATRCES

	0	1	2
0	4	5	5
1	0	**2**	1
2	0	**4**	3
3	1	**3**	4
4	2	**1**	7
5	3	**2**	6

	0	1	2
0	5	4	5
1	1	2	7
2	2	0	1
3	2	3	6
4			
5			

Select a column with value 3

TRANSPOSE OF A SPARSE MATRCES

	0	1	2
0	4	5	5
1	0	**2**	1
2	0	**4**	3
3	**1**	**3**	**4**
4	2	**1**	7
5	3	**2**	6

	0	1	2
0	5	4	5
1	1	2	7
2	2	0	1
3	2	3	6
4			
5			

Select a column with value 3

TRANSPOSE OF A SPARSE MATRCES

	0	1	2
0	4	5	5
1	0	**2**	1
2	0	**4**	3
3	**1**	**3**	**4**
4	2	**1**	7
5	3	**2**	6

	0	1	2
0	5	4	5
1	1	2	7
2	2	0	1
3	2	3	6
4	3	1	4
5			

Select a column with value 3

TRANSPOSE OF A SPARSE MATRCES

	0	1	2
0	4	5	5
1	0	**2**	1
2	0	**4**	3
3	1	**3**	4
4	2	**1**	7
5	3	**2**	6

	0	1	2
0	5	4	5
1	1	2	7
2	2	0	1
3	2	3	6
4	3	1	4
5			

Select a column with value 4

TRANSPOSE OF A SPARSE MATRCES

	0	1	2
0	4	5	5
1	0	**2**	1
2	**0**	**4**	**3**
3	1	**3**	4
4	2	**1**	7
5	3	**2**	6

	0	1	2
0	5	4	5
1	1	2	7
2	2	0	1
3	2	3	6
4	3	1	4
5			

Select a column with value 4

TRANSPOSE OF A SPARSE MATRCES

	0	1	2
0	4	5	5
1	0	**2**	1
2	**0**	**4**	**3**
3	1	**3**	4
4	2	**1**	7
5	3	**2**	6

	0	1	2
0	5	4	5
1	1	2	7
2	2	0	1
3	2	3	6
4	3	1	4
5	4	0	3

Select a column with value 4

TRANSPOSE OF A SPARSE MATRCES

	0	1	2
0	4	5	5
1	0	**2**	1
2	0	**4**	3
3	1	**3**	4
4	2	**1**	7
5	3	**2**	6

	0	1	2
0	5	4	5
1	1	2	7
2	2	0	1
3	2	3	6
4	3	1	4
5	4	0	3

Transpose

Algorithm SparseTranspose(A,B)

```
{       B[0][0]=A[0][1];   B[0][1]= A[0][0]; B[0][2]= A[0][2];
        k=1;   m=A[0][1];   n= A[0][2];
        for i=0 to m-1 do
        {       for j=1 to n do
                {       if  A[j][1] = i then
                        {       B[k][0] = A[j][1];
                                B[k][1] = A[j][0];
                                B[k][2] = A[j][2]
                                k=k+1
                        }
                }
        }
}
```

SPARSE MATRIX TRANSPOSE-PROGRAM

```
#include<stdio.h>
int S[10][10],T[10][10];
void Sparse(int A[10][10], int m, int n)
{  int i, j, k=1;
   S[0][0]=m;   S[0][1]=n;
   for (i=0; i<m; i++)
   {      for (j=0; j<n; j++)
          {          if (A[i][j] != 0)
                     {          S[k][0] = i;
                                S[k][1] = j;
                                S[k][2] = A[i][j];
                                k++;
                     }
          }
   }
   S[0][2] = k-1;
   printf("\nTuple Form:\n");
   for (i=0; i<=S[0][2]; i++)
   {      for (j=0; j<3; j++)
          {
                printf("%d\t",S[i][j]);
          }
          printf("\n");
   }
}
```

```
void Transpose()
{  int m,n,k,i,j;
   T[0][0]=S[0][1];     T[0][1]=S[0][0];          T[0][2]=S[0][2];
   k=1;
   m=S[0][1];           n= S[0][2];
   for(i=0;i<m;i++)
   {      for(j=1;j<=n;j++)
          {        if (S[j][1] == i)
                   {        T[k][0] = S[j][1];
                            T[k][1] = S[j][0];
                            T[k][2] = S[j][2];
                            k=k+1;
                   }
          }
   }
```

```
    printf("\nTranspose:\n");
    for (i=0; i<=T[0][2]; i++)
    {
        for (j=0; j<3; j++)
        {
            printf("%d\t",T[i][j]);
        }
        printf("\n");
    }
}
```

```
void main()
{  int i,j,A[10][10],m,n;
   printf("Enter the number of rows and columns of the matrix: ");
   scanf("%d%d",&m,&n);
   printf("Enter the matrix:");
   for(i=0;i<m;i++)
         for(j=0;j<n;j++)
               scanf("%d",&A[i][j]);

   Sparse(A,m,n);
   Transpose();
}
```

SPARSE MATRX ADDITION

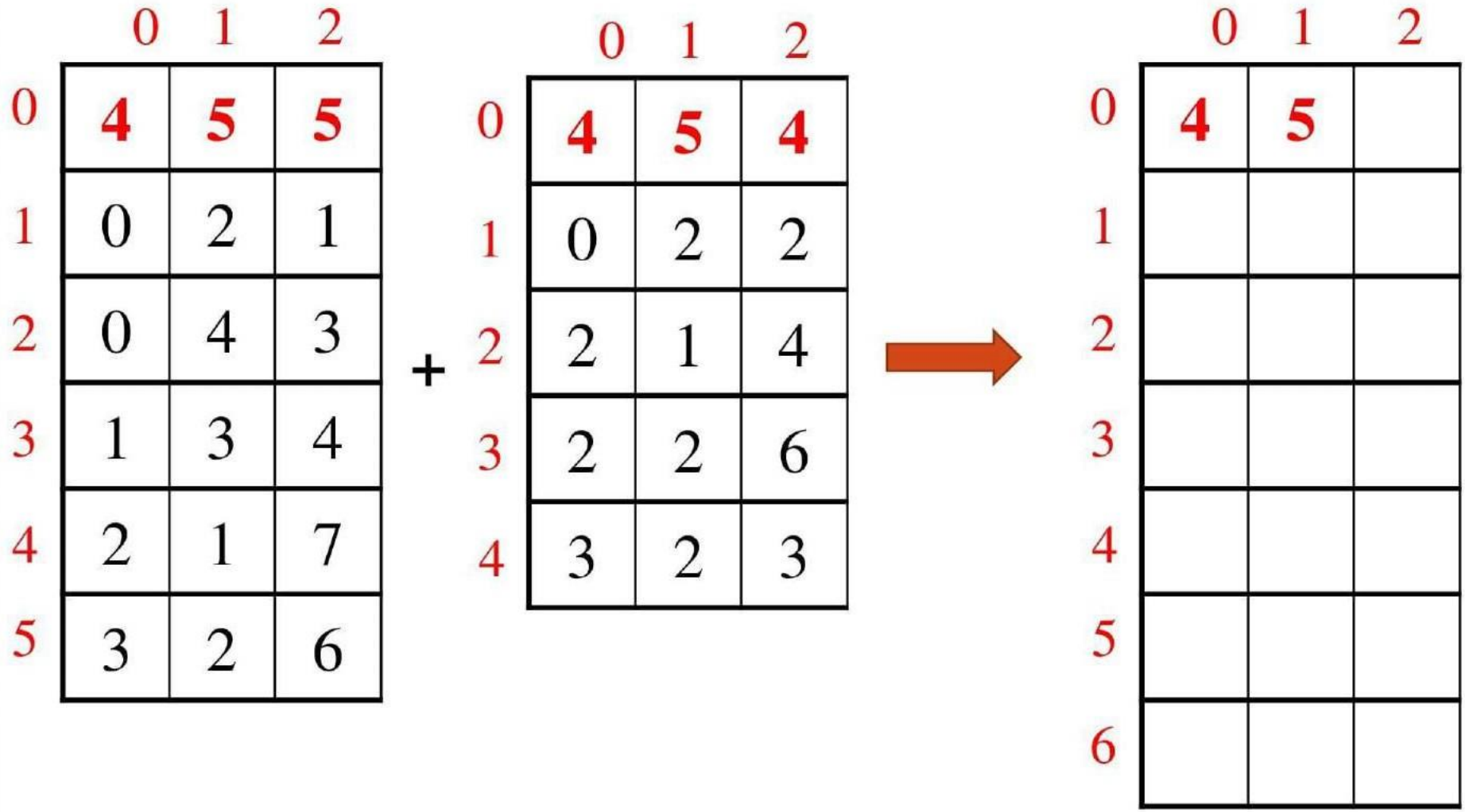
	0	1	2
0	**4**	**5**	**5**
1	0	2	1
2	0	4	3
3	1	3	4
4	2	1	7
5	3	2	6

+

	0	1	2
0	**4**	**5**	**4**
1	0	2	2
2	2	1	4
3	2	2	6
4	3	2	3

	0	1	2
0			
1			
2			
3			
4			
5			
6			

SPARSE MATRX ADDITION



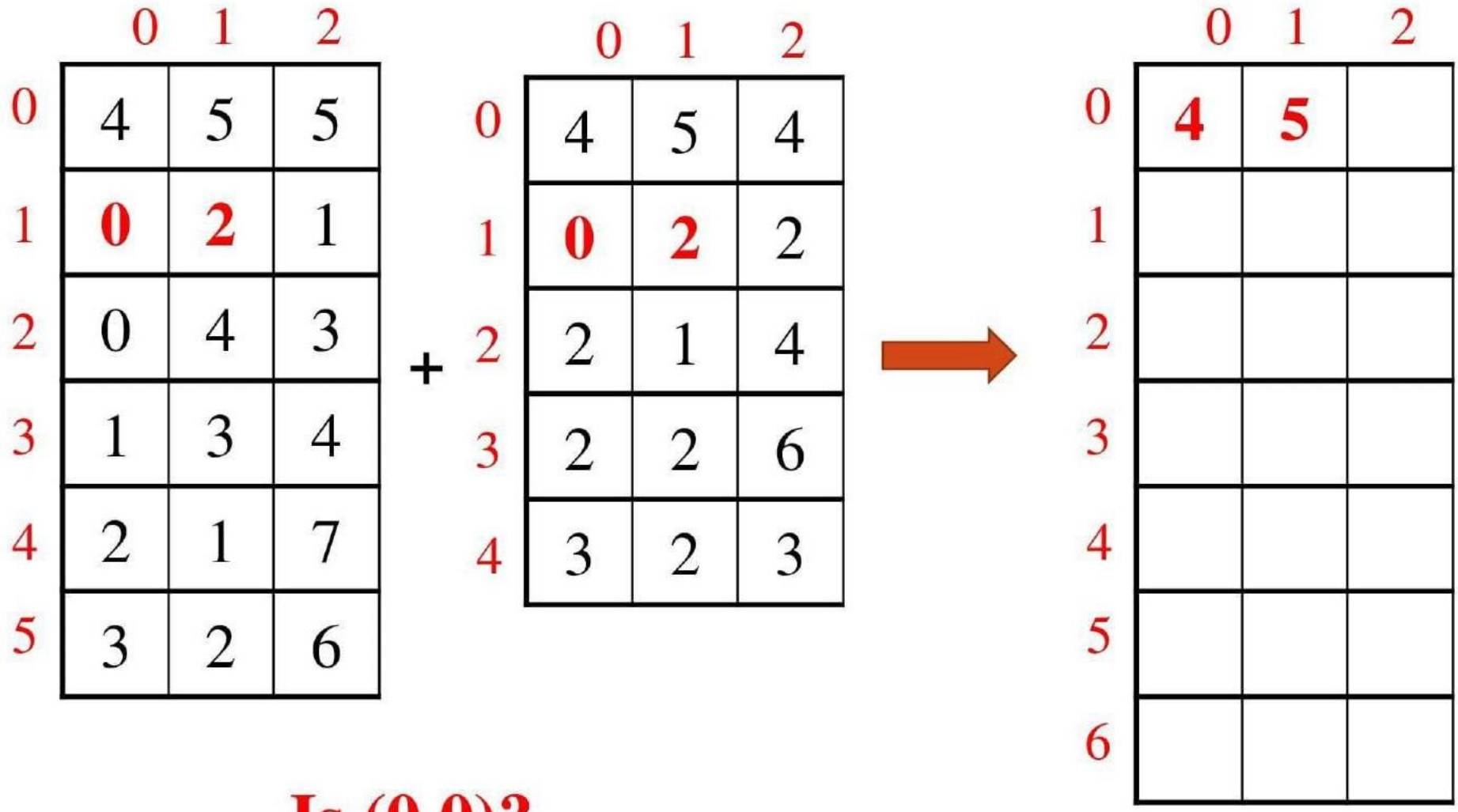
SPARSE MATRX ADDITION

The resultant matrix having dimension 4x5
All possible locations are:

(0,0)	(0,1)	(0,2)	(0,3)	(0,4)
(1,0)	(1,1)	(1,2)	(1,3)	(1,4)
(2,0)	(2,1)	(2,2)	(2,3)	(2,4)
(3,0)	(3,1)	(3,2)	(3,3)	(3,4)

Check all possible locations

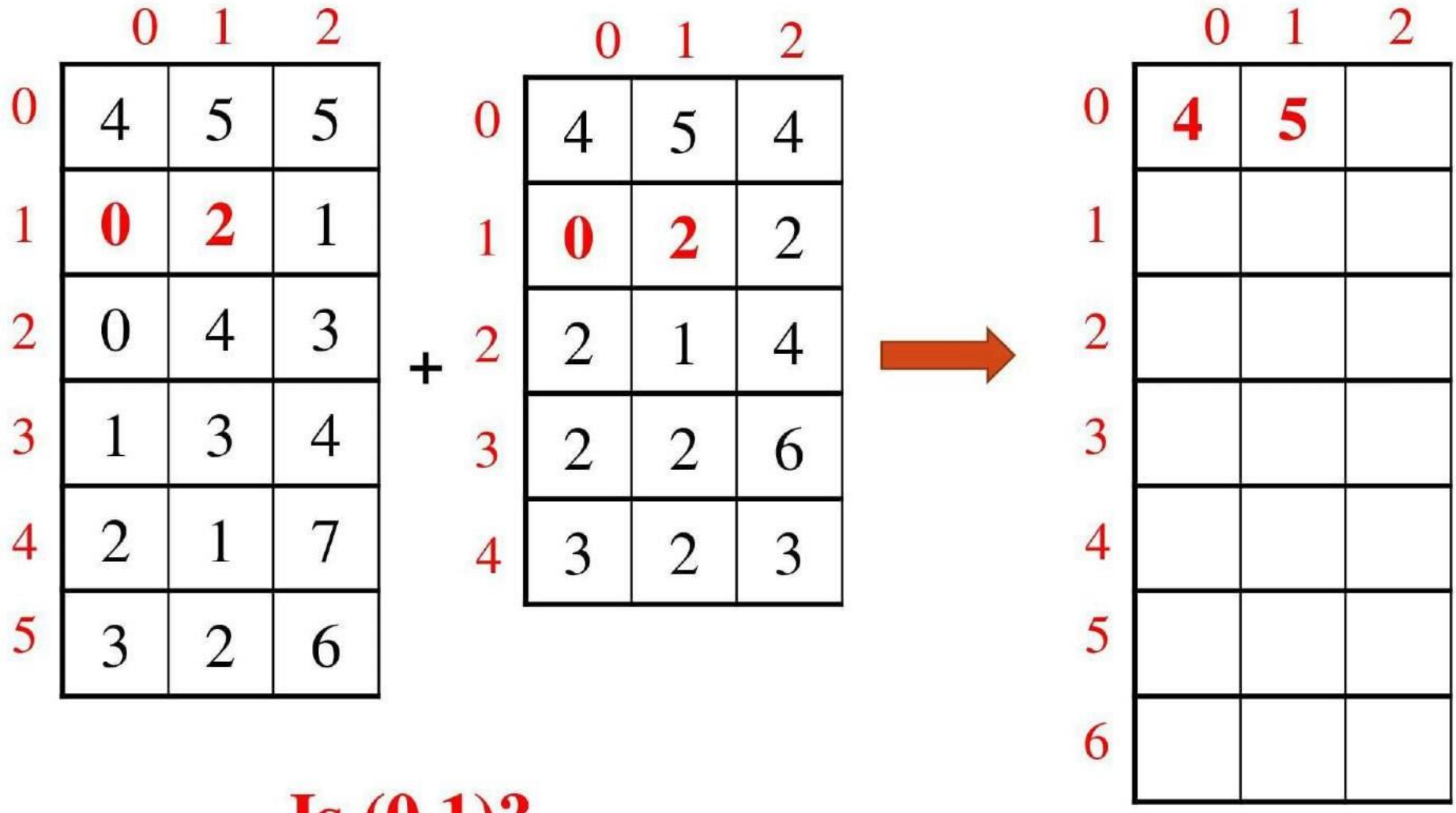
SPARSE MATRX ADDITION



Is (0,0)?

No

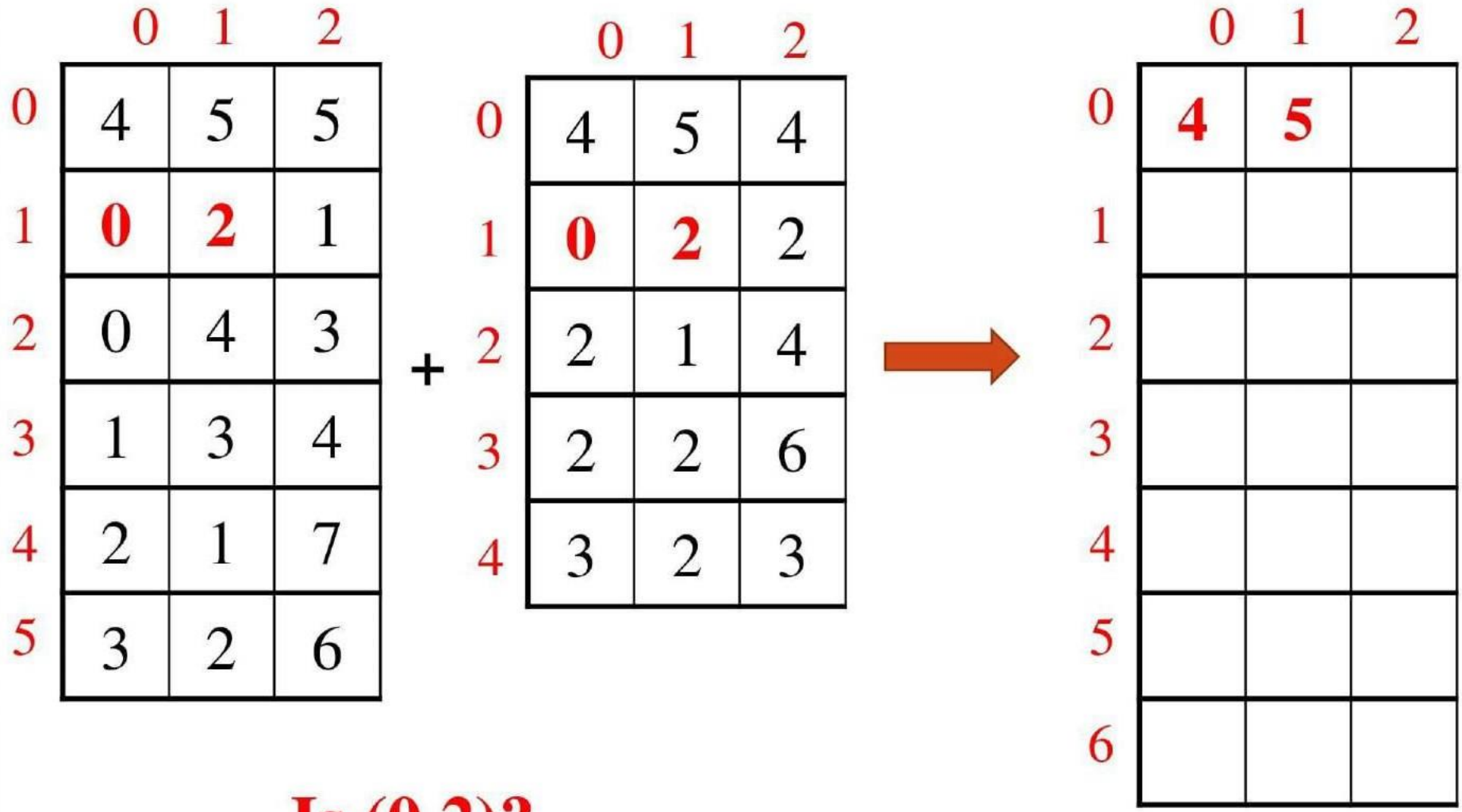
SPARSE MATRX ADDITION



Is (0,1)?

No

SPARSE MATRX ADDITION



Is (0,2)?

Both are (0,2). So add them

SPARSE MATRX ADDITION

	0	1	2
0	4	5	5
1	**0**	**2**	1
2	0	4	3
3	1	3	4
4	2	1	7
5	3	2	6

+

	0	1	2
0	4	5	4
1	**0**	**2**	2
2	2	1	4
3	2	2	6
4	3	2	3

	0	1	2
0	**4**	**5**	
1	**0**	**2**	**3**
2			
3			
4			
5			
6			

Is (0,2)?

Both are (0,2). So add them

SPARSE MATRX ADDITION

	0	1	2
0	4	5	5
1	0	2	1
2	**0**	**4**	3
3	1	3	4
4	2	1	7
5	3	2	6

+

	0	1	2
0	4	5	4
1	0	2	2
2	**2**	**1**	4
3	2	2	6
4	3	2	3

	0	1	2
0	**4**	**5**	
1	**0**	**2**	**3**
2			
3			
4			
5			
6			

SPARSE MATRX ADDITION

	0	1	2
0	4	5	5
1	0	2	1
2	**0**	**4**	3
3	1	3	4
4	2	1	7
5	3	2	6

+

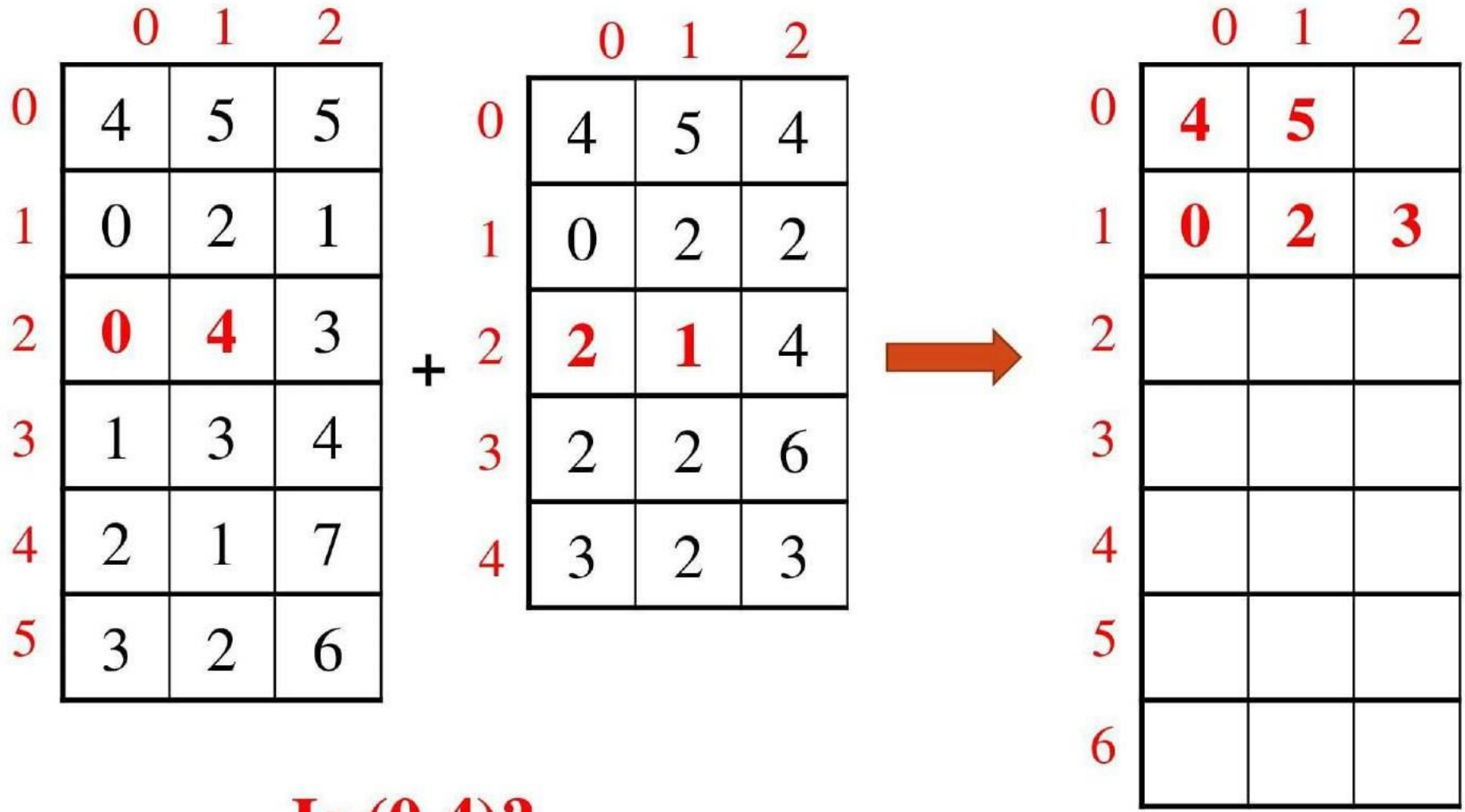
	0	1	2
0	4	5	4
1	0	2	2
2	**2**	**1**	4
3	2	2	6
4	3	2	3

	0	1	2
0	**4**	**5**	
1	**0**	**2**	**3**
2			
3			
4			
5			
6			

Is (0,3)?

No

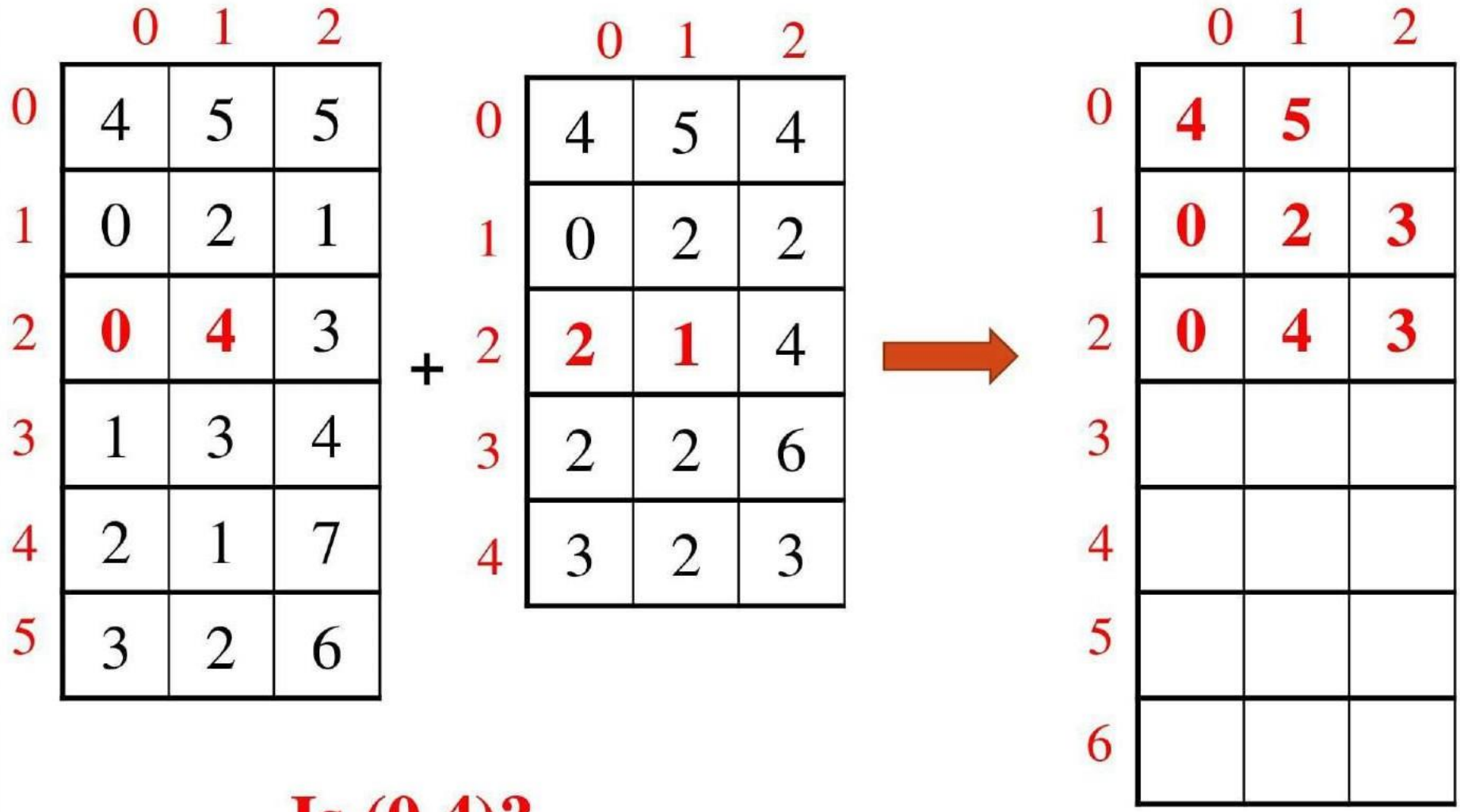
SPARSE MATRX ADDITION



Is (0,4)?

First one is (0,4). Move it.

SPARSE MATRX ADDITION



Is (0,4)?

First one is (0,4). Move it.

SPARSE MATRX ADDITION

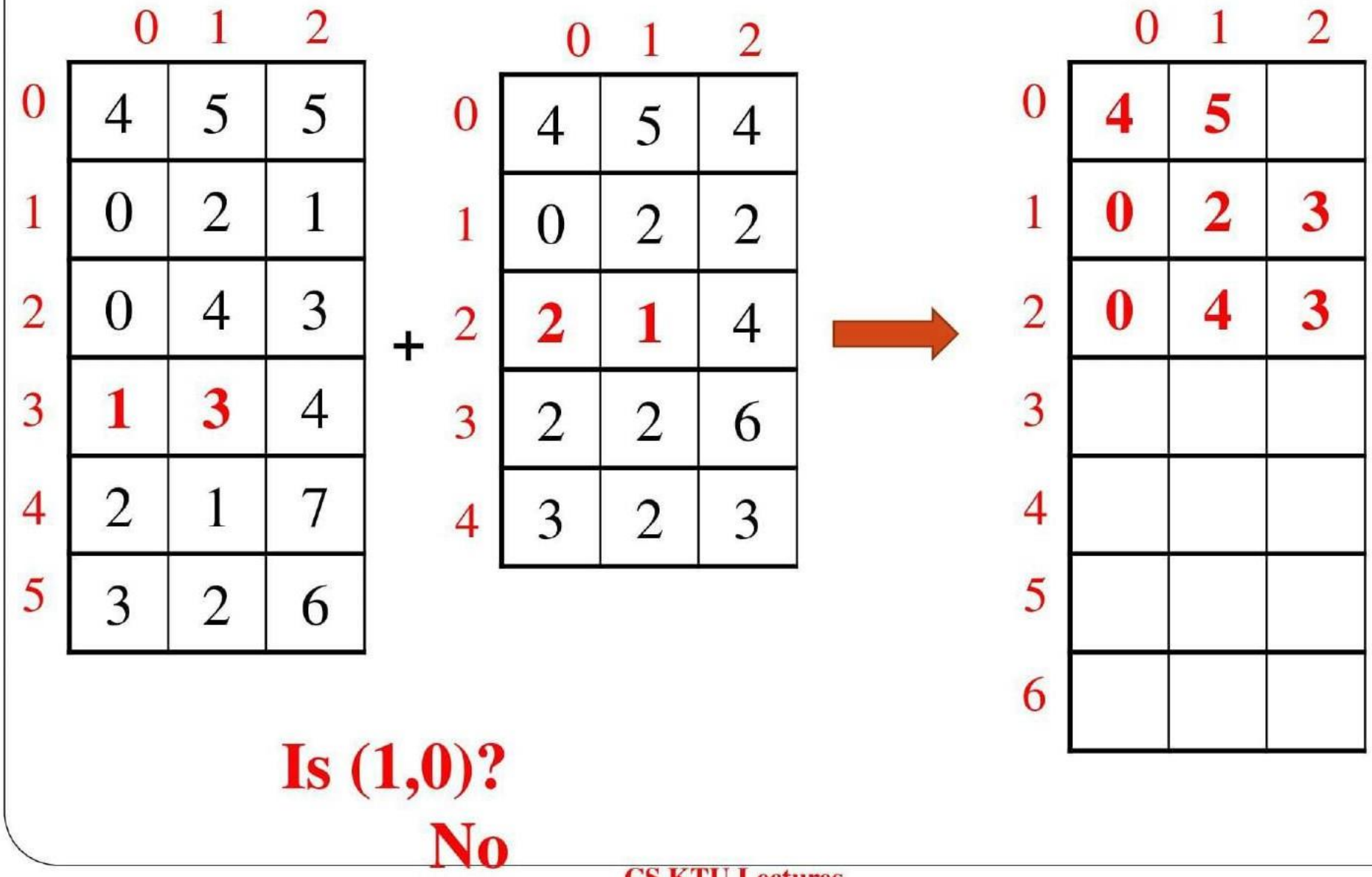
	0	1	2
0	4	5	5
1	0	2	1
2	0	4	3
3	**1**	**3**	4
4	2	1	7
5	3	2	6

+

	0	1	2
0	4	5	4
1	0	2	2
2	**2**	**1**	4
3	2	2	6
4	3	2	3

	0	1	2
0	**4**	**5**	
1	**0**	**2**	**3**
2	**0**	**4**	**3**
3			
4			
5			
6			

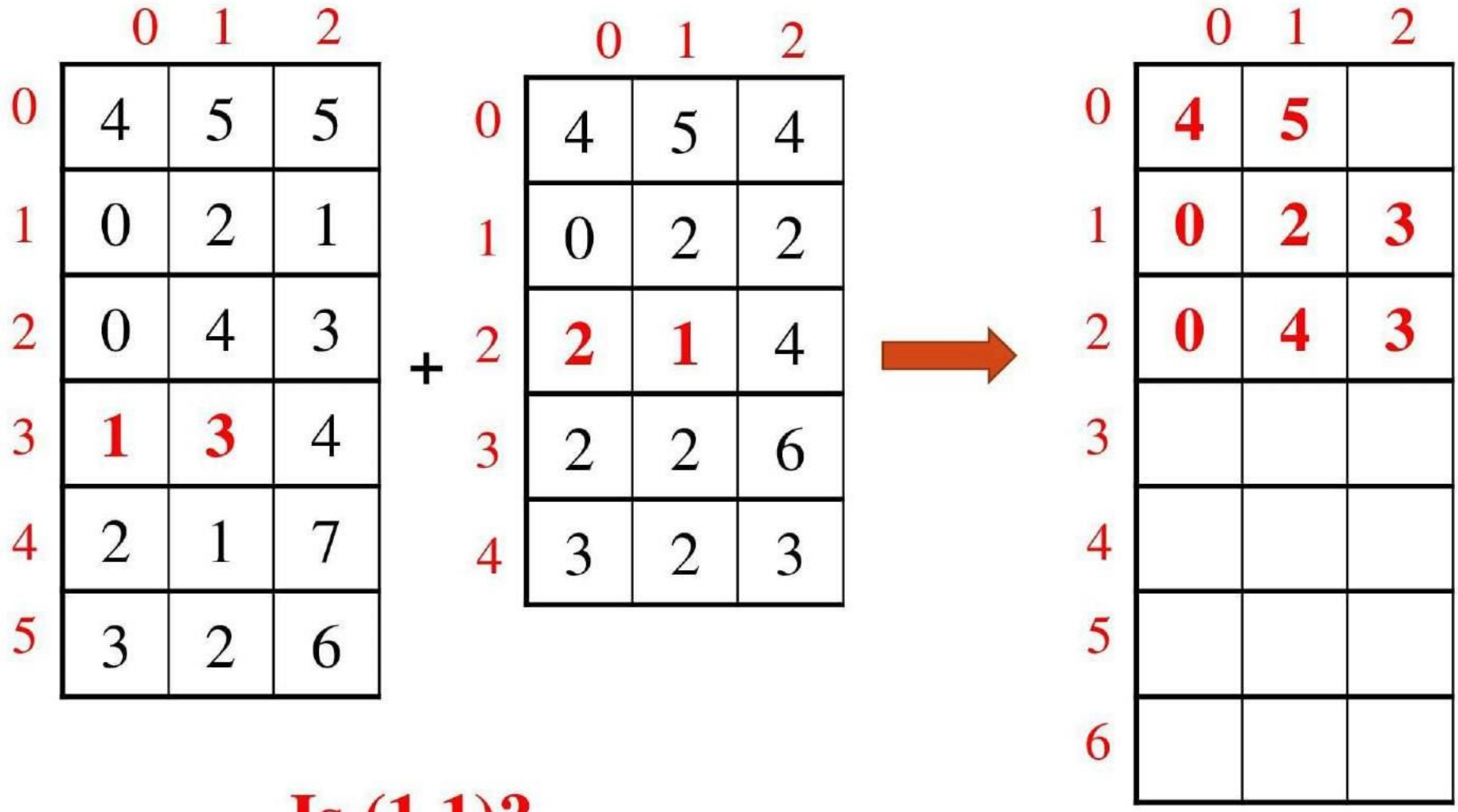
SPARSE MATRX ADDITION



Is (1,0)?

No

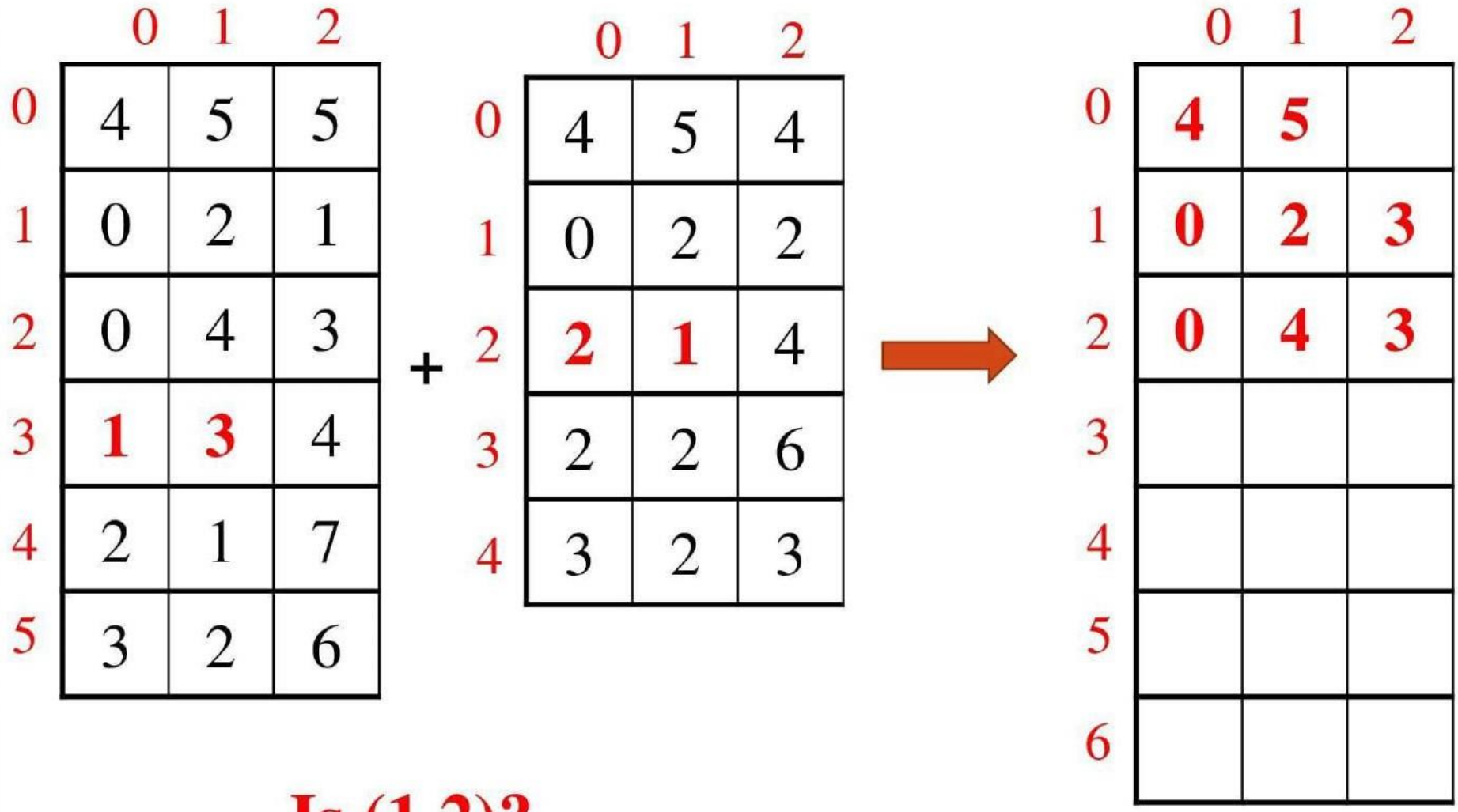
SPARSE MATRX ADDITION



Is (1,1)?

No

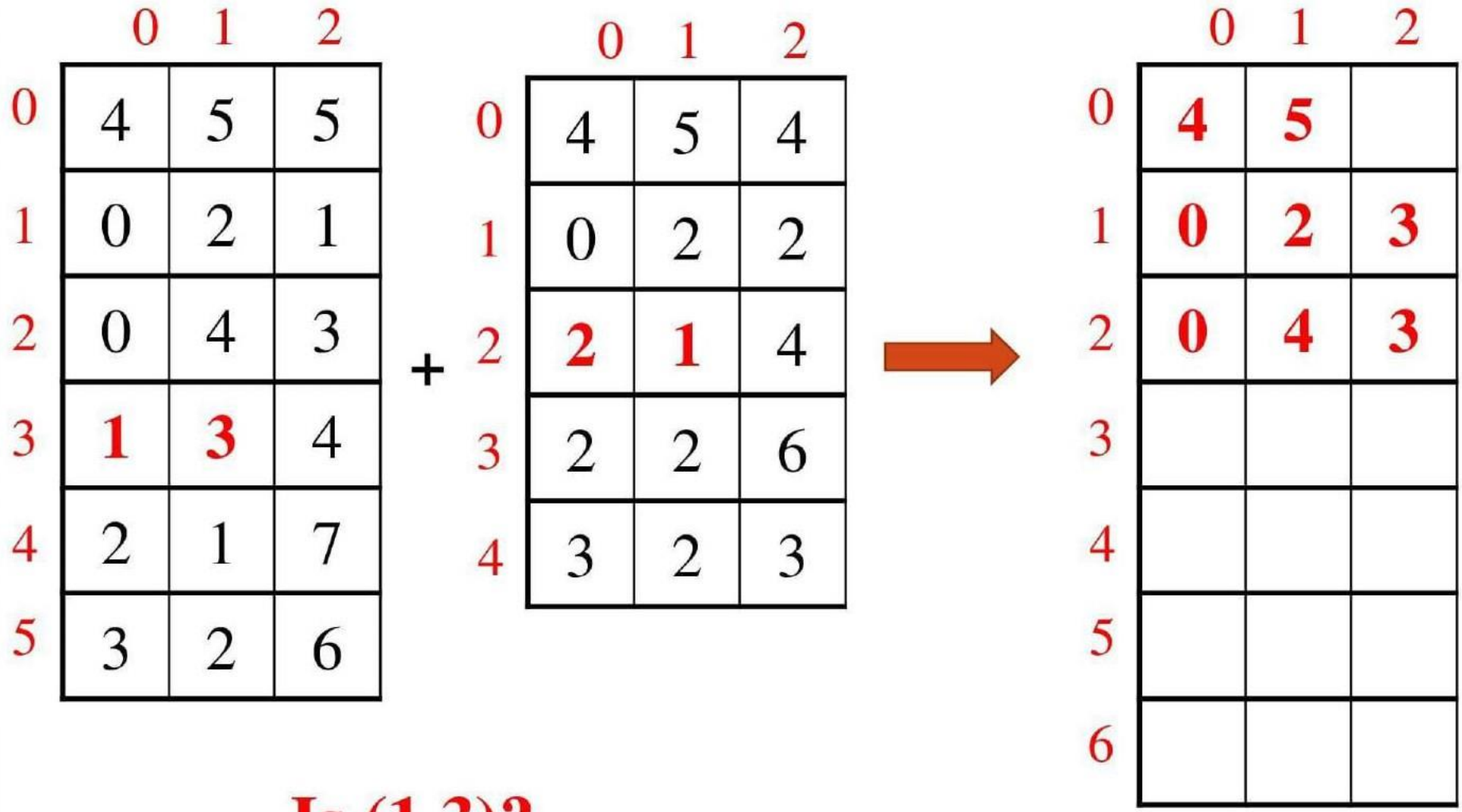
SPARSE MATRX ADDITION



Is (1,2)?

No

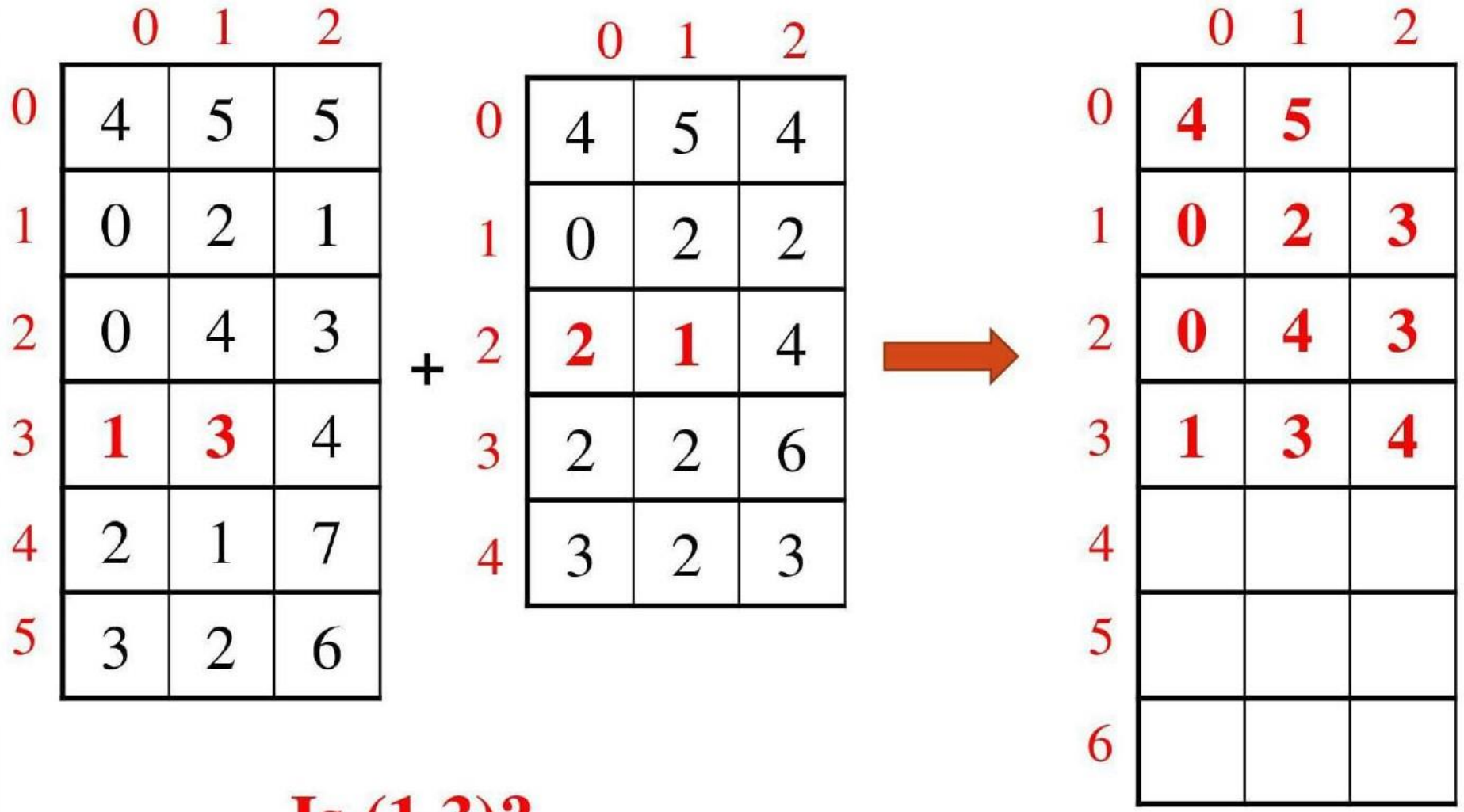
SPARSE MATRX ADDITION



Is (1,3)?

First One is (1,3). Move it.

SPARSE MATRX ADDITION



Is (1,3)?

First One is (1,3). Move it.

SPARSE MATRX ADDITION

	0	1	2
0	4	5	5
1	0	2	1
2	0	4	3
3	1	3	4
4	**2**	**1**	7
5	3	2	6

+

	0	1	2
0	4	5	4
1	0	2	2
2	**2**	**1**	4
3	2	2	6
4	3	2	3

	0	1	2
0	**4**	**5**	
1	**0**	**2**	**3**
2	**0**	**4**	**3**
3	**1**	**3**	**4**
4			
5			
6			

SPARSE MATRX ADDITION

	0	1	2
0	4	5	5
1	0	2	1
2	0	4	3
3	1	3	4
4	**2**	**1**	7
5	3	2	6

+

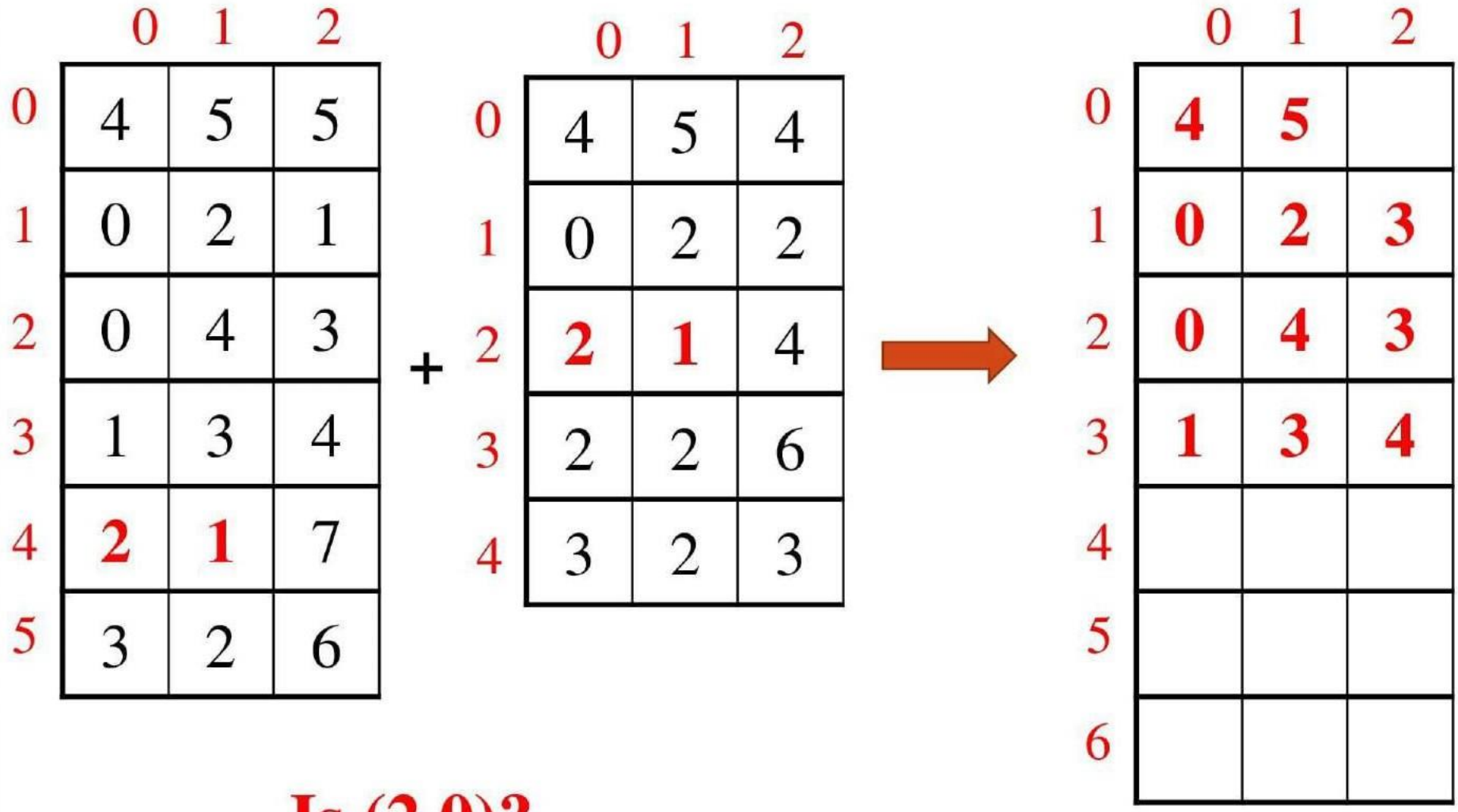
	0	1	2
0	4	5	4
1	0	2	2
2	**2**	**1**	4
3	2	2	6
4	3	2	3

	0	1	2
0	**4**	**5**	
1	**0**	**2**	**3**
2	**0**	**4**	**3**
3	**1**	**3**	**4**
4			
5			
6			

Is (1,4)?

No

SPARSE MATRX ADDITION



Is (2,0)?

No

SPARSE MATRX ADDITION

	0	1	2
0	4	5	5
1	0	2	1
2	0	4	3
3	1	3	4
4	**2**	**1**	7
5	3	2	6

+

	0	1	2
0	4	5	4
1	0	2	2
2	**2**	**1**	4
3	2	2	6
4	3	2	3

	0	1	2
0	**4**	**5**	
1	**0**	**2**	**3**
2	**0**	**4**	**3**
3	**1**	**3**	**4**
4			
5			
6			

Is (2,1)?

Both are (2,1). Add them.

SPARSE MATRX ADDITION

	0	1	2
0	4	5	5
1	0	2	1
2	0	4	3
3	1	3	4
4	**2**	**1**	7
5	3	2	6

+

	0	1	2
0	4	5	4
1	0	2	2
2	**2**	**1**	4
3	2	2	6
4	3	2	3

	0	1	2
0	**4**	**5**	
1	**0**	**2**	**3**
2	**0**	**4**	**3**
3	**1**	**3**	**4**
4	**2**	**1**	**11**
5			
6			

Is (2,1)?

Both are (2,1). Add them.

SPARSE MATRX ADDITION

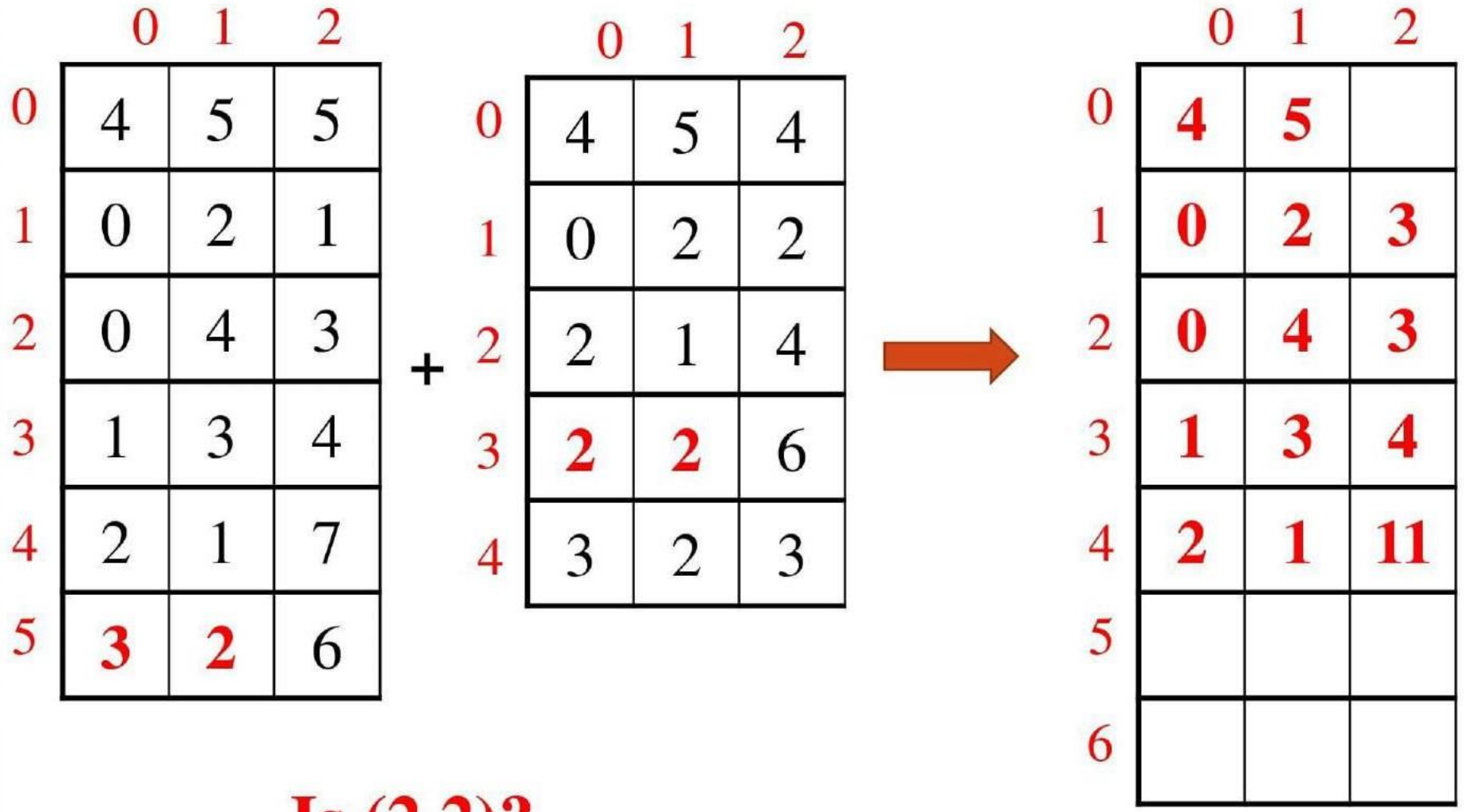
	0	1	2
0	4	5	5
1	0	2	1
2	0	4	3
3	1	3	4
4	2	1	7
5	**3**	**2**	6

+

	0	1	2
0	4	5	4
1	0	2	2
2	2	1	4
3	**2**	**2**	6
4	3	2	3

	0	1	2
0	**4**	**5**	
1	**0**	**2**	**3**
2	**0**	**4**	**3**
3	**1**	**3**	**4**
4	**2**	**1**	**11**
5			
6			

SPARSE MATRX ADDITION



Is (2,2)?

Second one is (2,2). Move it.

SPARSE MATRX ADDITION

	0	1	2
0	4	5	5
1	0	2	1
2	0	4	3
3	1	3	4
4	2	1	7
5	**3**	**2**	6

+

	0	1	2
0	4	5	4
1	0	2	2
2	2	1	4
3	**2**	**2**	6
4	3	2	3

	0	1	2
0	**4**	**5**	
1	**0**	**2**	**3**
2	**0**	**4**	**3**
3	**1**	**3**	**4**
4	**2**	**1**	**11**
5	**2**	**2**	**6**
6			

Is (2,2)?

Second one is (2,2). Move it.

SPARSE MATRX ADDITION

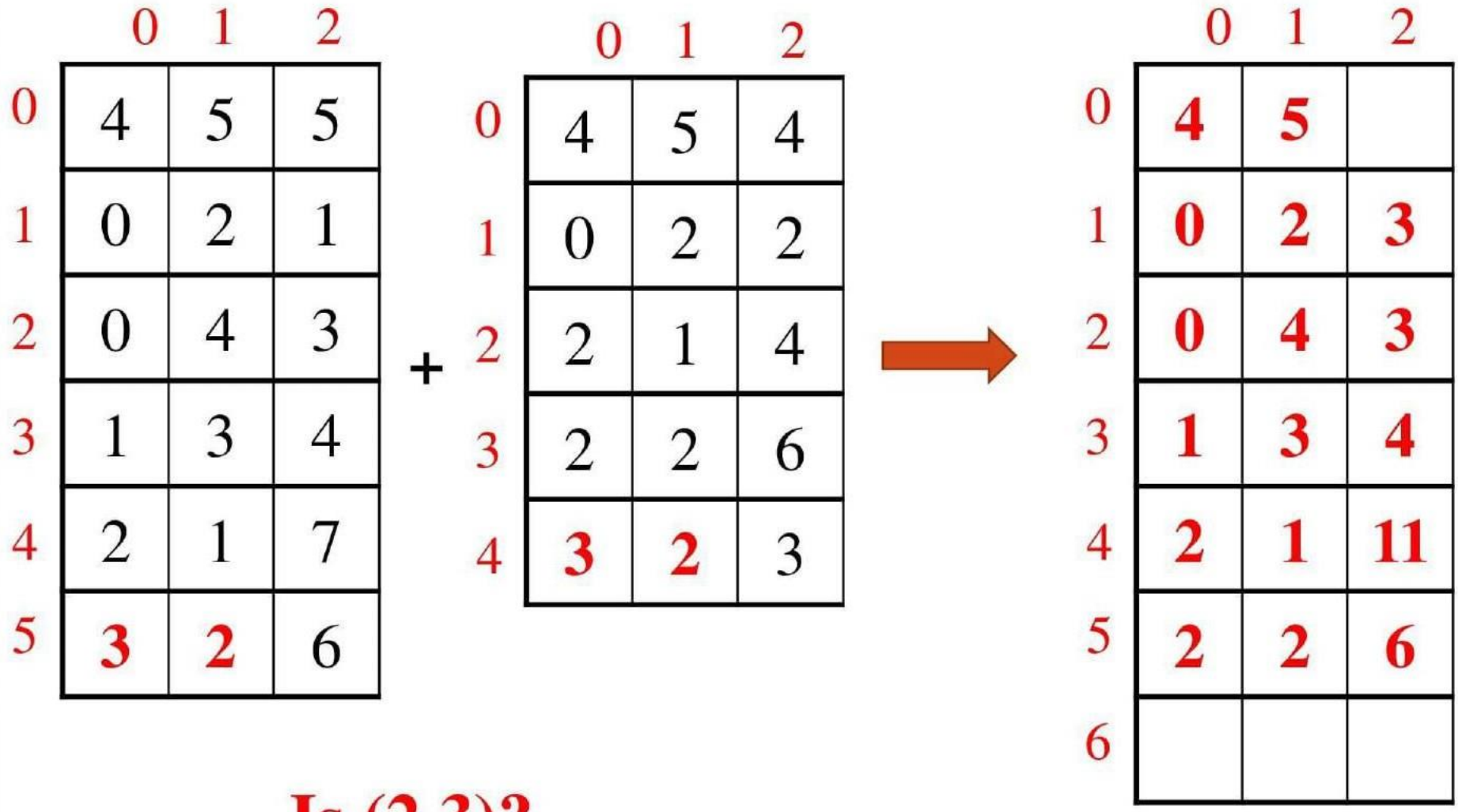
	0	1	2
0	4	5	5
1	0	2	1
2	0	4	3
3	1	3	4
4	2	1	7
5	**3**	**2**	6

+

	0	1	2
0	4	5	4
1	0	2	2
2	2	1	4
3	2	2	6
4	**3**	**2**	3

	0	1	2
0	**4**	**5**	
1	**0**	**2**	**3**
2	**0**	**4**	**3**
3	**1**	**3**	**4**
4	**2**	**1**	**11**
5	**2**	**2**	**6**
6			

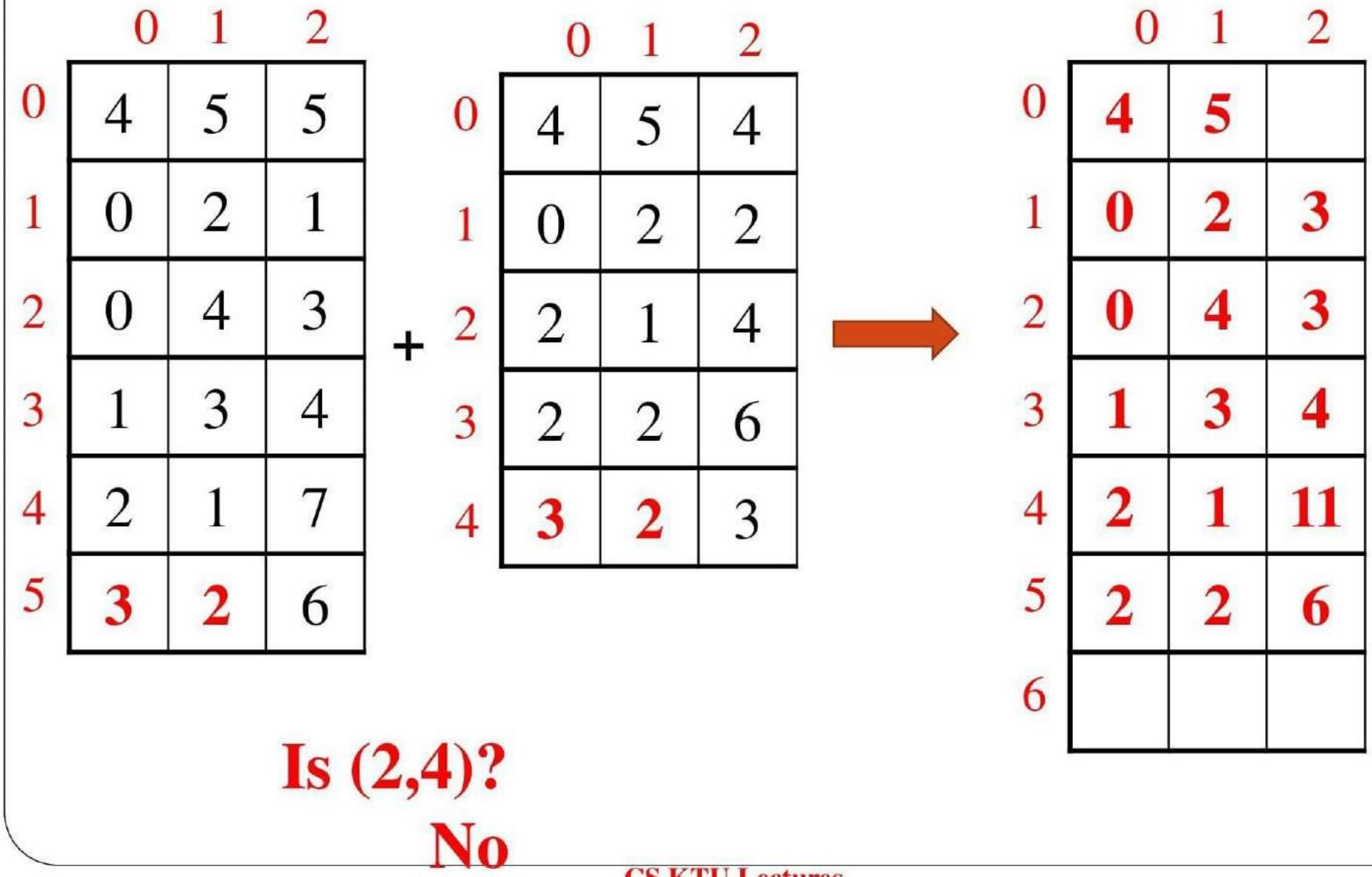
SPARSE MATRX ADDITION



Is (2,3)?

No

SPARSE MATRX ADDITION



Is (2,4)?

No

SPARSE MATRX ADDITION

	0	1	2
0	4	5	5
1	0	2	1
2	0	4	3
3	1	3	4
4	2	1	7
5	**3**	**2**	6

+

	0	1	2
0	4	5	4
1	0	2	2
2	2	1	4
3	2	2	6
4	**3**	**2**	3

	0	1	2
0	**4**	**5**	
1	**0**	**2**	**3**
2	**0**	**4**	**3**
3	**1**	**3**	**4**
4	**2**	**1**	**11**
5	**2**	**2**	**6**
6			

Is (3,0)?

No

SPARSE MATRX ADDITION

	0	1	2
0	4	5	5
1	0	2	1
2	0	4	3
3	1	3	4
4	2	1	7
5	**3**	**2**	6

+

	0	1	2
0	4	5	4
1	0	2	2
2	2	1	4
3	2	2	6
4	**3**	**2**	3

	0	1	2
0	**4**	**5**	
1	**0**	**2**	**3**
2	**0**	**4**	**3**
3	**1**	**3**	**4**
4	**2**	**1**	**11**
5	**2**	**2**	**6**
6			

Is (3,1)?

No

SPARSE MATRX ADDITION

	0	1	2
0	4	5	5
1	0	2	1
2	0	4	3
3	1	3	4
4	2	1	7
5	**3**	**2**	6

+

	0	1	2
0	4	5	4
1	0	2	2
2	2	1	4
3	2	2	6
4	**3**	**2**	3

	0	1	2
0	**4**	**5**	
1	**0**	**2**	**3**
2	**0**	**4**	**3**
3	**1**	**3**	**4**
4	**2**	**1**	**11**
5	**2**	**2**	**6**
6			

Is (3,2)?

Both are (0,2).Add them.

SPARSE MATRX ADDITION

	0	1	2
0	4	5	5
1	0	2	1
2	0	4	3
3	1	3	4
4	2	1	7
5	**3**	**2**	6

+

	0	1	2
0	4	5	4
1	0	2	2
2	2	1	4
3	2	2	6
4	**3**	**2**	3

	0	1	2
0	**4**	**5**	
1	**0**	**2**	**3**
2	**0**	**4**	**3**
3	**1**	**3**	**4**
4	**2**	**1**	**11**
5	**2**	**2**	**6**
6	**3**	**2**	**9**

Is (3,2)?

Both are (0,2).Add them.

SPARSE MATRX ADDITION

	0	1	2
0	4	5	5
1	0	2	1
2	0	4	3
3	1	3	4
4	2	1	7
5	3	2	6

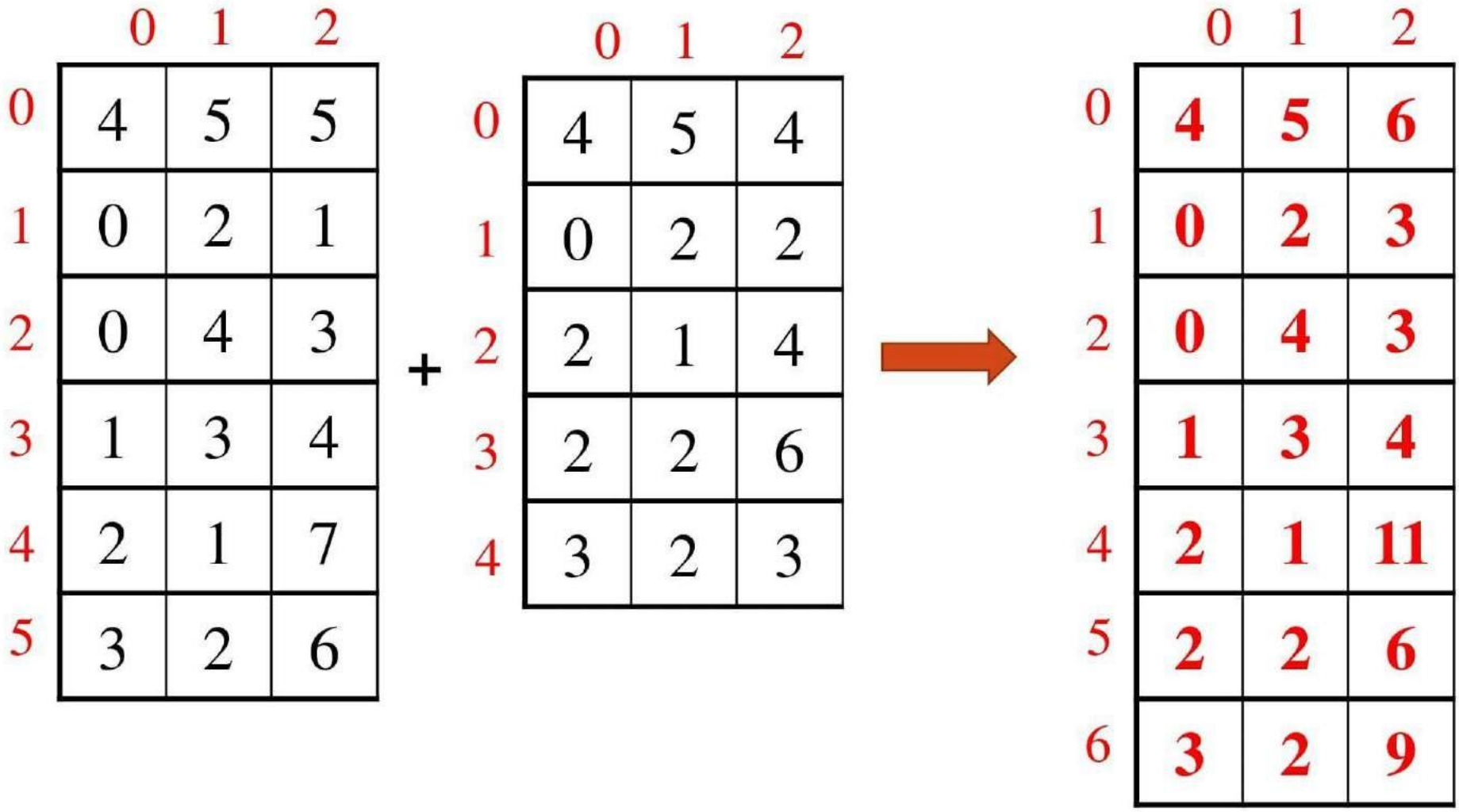
+

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

	0	1	2
0	**4**	**5**	
1	**0**	**2**	**3**
2	**0**	**4**	**3**
3	**1**	**3**	**4**
4	**2**	**1**	**11**
5	**2**	**2**	**6**
6	**3**	**2**	**9**

Total number of nonzero elements=6

SPARSE MATRX ADDITION



Total number of nonzero elements=6

SPARSE MATRIX ADDITION ALGORITHM

Algorithm SparseAddition(A,B,C)

```
{
        r1=A[0][0];    c1=A[0][1];
        r2=B[0][0];    c2=B[0][1];
        if r1!=r2 or c1!=c2 then
                Print "Incompatible Matrix size"
        else
        {       C[0][0]=A[0][0]
                C[0][1]=A[0][1]
                m=1;    n=1;    k=1;
                for i=0 to r1-1 do
                {       for j=0 to c1-1 do
                        {
```

```
if A[m][0]=i and A[m][1]=j and
                B[n][0]=i and B[n][1]=j then
{       C[k][0]=A[m][0]
        C[k][1]=A[m][1]
        C[k][2]=A[m][2]+B[n][2]
        m=m+1;      n=n+1;      k=k+1;
}
else if A[m][0]=i and A[m][1]=j then
{       C[k][0]=A[m][0]
        C[k][1]=A[m][1]
        C[k][2]=A[m][2]
         m=m+1;     k=k+1;
}
```

```
                else if B[n][0]=i and B[n][1]=j then
                {       C[k][0]=B[n][0]
                        C[k][1]=B[n][1]
                        C[k][2]=B[n][2]
                         n=n+1;          k=k+1;
                }
            }//end of for j
        } //end of for i
        C[0][2]=k-1

        Print "SUM:"
        for i=0 to C[0][2] do
                for j=0 to 3 do
                        Print C[i][j]
    }
}
```

SPARSE MATRIX ADDITION PROGRAM

```
#include<stdio.h>
void PrintSparse(int X[20][3])
{
   int i,j;
   printf("\nTuple representation of Sparse Matrix is:\n");
   for (i=0; i<=X[0][2]; i++)
   {
         for (j=0; j<3; j++)
         {
               printf("%d\t",X[i][j]);
         }
         printf("\n");
   }
}
```

```
void Sparse(int A[10][10], int m, int n, int S[20][3])
{  int i, j, k=1;
   S[0][0]=m;  S[0][1]=n;
   for (i=0; i<m; i++)
   {      for (j=0; j<n; j++)
          {          if (A[i][j] != 0)
                     {          S[k][0] = i;     S[k][1] = j;     S[k][2] = A[i][j];
                                k++;
                     }
          }
   }
   S[0][2] = k-1;
   PrintSparse(S);
}
```

```
void AddSparse(int S[20][3],int T[20][3])
{  int r1,c1,r2,c2,m,n,i,j,k,A[20][3];
   r1=S[0][0];          c1=S[0][1];
   r2=T[0][0];          c2=T[0][1];

   if(r1!=r2 || c1!=c2)
         printf("Incompatible Matrix size\n");
   else
   {     A[0][0]=S[0][0];      A[0][1]=S[0][1];
         m=1;    n=1;   k=1;
         for(i=0;i<r1;i++)
         {        for(j=0;j<c1;j++)
                  {
```

```
if(S[m][0]==i && S[m][1]==j && T[n][0]==i
                                && T[n][1]==j)
{
        A[k][0]=S[m][0];
        A[k][1]=S[m][1];
        A[k][2]=S[m][2]+T[n][2];
        m++;
        n++;
        k++;
}
```

```
                else if(S[m][0]==i && S[m][1]==j)
                {       A[k][0]=S[m][0];
                        A[k][1]=S[m][1];
                        A[k][2]=S[m][2];
                        m++;            k++;
                }
                else if(T[n][0]==i && T[n][1]==j)
                {       A[k][0]=T[n][0];
                        A[k][1]=T[n][1];
                        A[k][2]=T[n][2];
                        n++;            k++;
                }
        }
}
A[0][2]=k-1;
```

```
        printf("SUM:\n");
        for(i=0;i<=A[0][2];i++)
        {
                for(j=0;j<3;j++)
                {
                        printf("%d\t",A[i][j]);
                }
                printf("\n");
        }
    }
}
```

```
void main()
{  int i,j,A[10][10],B[10][10],m1,n1,m2,n2,S[20][3],T[20][3];
   printf("Enter the number of rows and columns of the first matrix: ");
   scanf("%d%d",&m1,&n1);
   printf("Enter the matrix:");
   for(i=0;i<m1;i++)
         for(j=0;j<n1;j++)
               scanf("%d",&A[i][j]);
   printf("Matrix1:\n");
   for (i=0; i<m1; i++)
   {     for (j=0; j<n1; j++)
         {        printf("%d\t",A[i][j]);
         }
         printf("\n");
   }
   Sparse(A,m1,n1,S);
```

```
printf("Enter the number of rows and columns of the Second matrix: ");
scanf("%d%d",&m2,&n2);
printf("Enter the matrix:");
for(i=0;i<m2;i++)
        for(j=0;j<n2;j++)
                scanf("%d",&B[i][j]);
printf("Matrix2:\n");
for (i=0; i<m2; i++)
{       for (j=0; j<n2; j++)
        {       printf("%d\t",A[i][j]);
        }
        printf("\n");
}
Sparse(B,m2,n2,T);
AddSparse(S,T);
}
```