Processamento da Informação
– Teoria –

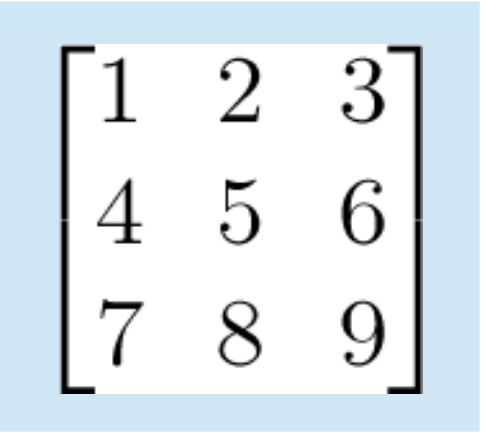
Matrizes

Semana 07

Prof. Jesús P. Mena-Chalco

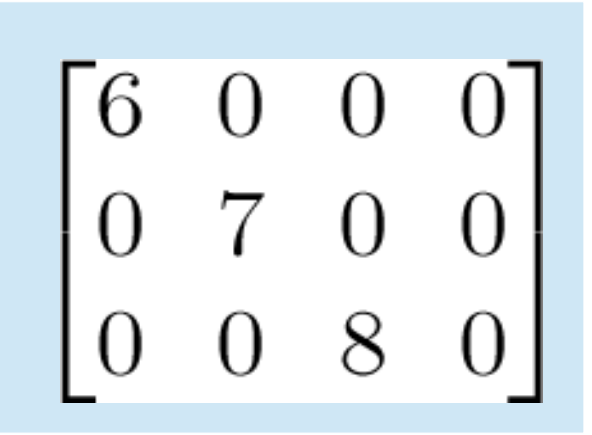
08/06/2013

Matrizes



```
>>> M = [ [1, 2, 3], [4, 5, 6], [7, 8, 9] ]

>>> M[1]                ← Seleção de uma linha
[4, 5, 6]

>>> M[1][1]             ← Seleção de um elemento
5
```

Matrizes




```
>>> P = [ [6, 0, 0, 0], [0, 7, 0, 0], [0, 0, 8, 0] ]

>>> len(P)
3

>>> len(P[0])
4
```

← Número de linhas

← Número de colunas

O operador *

```
>>> 'UF' + '5'
'UF5'

>>> 'UF' * 5
'UFUFUFUFUF'

>>> [1, 2, 3] * 5
```

O operador *

```
>>> 'UF' + '5'
'UF5'

>>> 'UF' * 5
'UFUFUFUFUF'

>>> [1, 2, 3] * 5
[1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3]
```

O operador *

```
>>> [0] * 5
[0, 0, 0, 0, 0]
```

O operador *

```
>>> [0] * 5
[0, 0, 0, 0, 0]

>>> M = [[0] * 5 ] *3
>>> print M
```

O operador *

```
>>> [0] * 5
[0, 0, 0, 0, 0]

>>> M = [[0] * 5 ] *3
>>> print M
[[0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0]]
```

O operador *

```
>>> [0] * 5
[0, 0, 0, 0, 0]

>>> M = [[0] * 5 ] *3
>>> print M
[[0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0]]

>>> M[1][4] = 42
>>> print M
```

O operador *

```
>>> [0] * 5
[0, 0, 0, 0, 0]

>>> M = [[0] * 5 ] *3
>>> print M
[[0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0]]

>>> M[1][4] = 42
>>> print M
[[0, 0, 0, 0, 42], [0, 0, 0, 0, 42], [0, 0, 0, 0, 42]]
```

Criação de matrizes

```
def criar_matriz_zeros(l,c):
    matriz = [0]*l
    for i in range(0,l):
        matriz[i] = [0]*c
    return matriz
```

Criação de matrizes

```
def criar_matriz_zeros(l,c):
    matriz = [0]*l
    for i in range(0,l):
        matriz[i] = [0]*c
    return matriz
```

```
>>> M = criar_matriz_zeros(3,5)
>>> M[1][4] = 42
>>> print M
[[0, 0, 0, 0, 0], [0, 0, 0, 0, 42], [0, 0, 0, 0, 0]]
```

Matriz identidade

```
def criar_matriz_identidade(n):
    matriz = [0]*n
    for i in range(0,n):
        matriz[i] = [0]*n
    for i in range(0,n):
        matriz[i][i] = 1
    return matriz
```

Matriz identidade

```
def criar_matriz_identidade(n):
    matriz = [0]*n
    for i in range(0,n):
        matriz[i] = [0]*n
    for i in range(0,n):
        matriz[i][i] = 1
    return matriz
```

```
>>> I = criar_matriz_identidade(4)
>>> print I
[[1, 0, 0, 0], [0, 1, 0, 0], [0, 0, 1, 0], [0, 0, 0, 1]]
```

Criação de matrizes

```
def criar_matriz_uns(l,c):
    matriz = [1]*l
    for i in range(0,l):
        matriz[i] = [1]*c
    return matriz
```

Criação de matrizes

```
def criar_matriz_uns(l,c):
    matriz = [1]*l
    for i in range(0,l):
        matriz[i] = [1]*c
    return matriz
```

```
>>> M = criar_matriz_uns(3,5)
>>> print M
[[1, 1, 1, 1, 1], [1, 1, 1, 1, 1], [1, 1, 1, 1, 1]]
```

Visualizar matrizes

```
>>> P = criar_matriz_uns(5,7)

>>> print P
[[1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1,
1, 1], [1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1]]
```

Visualizar matrizes

```
def visualizar_matriz(matriz):
    for i in range(0,len(matriz)):
        print matriz[i]
```

Visualizar matrizes

```
def visualizar_matriz(matriz):
    for i in range(0,len(matriz)):
        print matriz[i]
```

```
>>> P = criar_matriz_uns(5,7)
>>> visualizar_matriz(P)
[1, 1, 1, 1, 1, 1, 1]
[1, 1, 1, 1, 1, 1, 1]
[1, 1, 1, 1, 1, 1, 1]
[1, 1, 1, 1, 1, 1, 1]
[1, 1, 1, 1, 1, 1, 1]
```

00. Matriz transposta

Crie uma função que permita calcular a Transposta de uma matriz dada como entrada.

Cabeçalho: **def transposta(A):**

$$A = \begin{pmatrix} 2 & 4 & 7 \\ 3 & 2 & 0 \\ 5 & 3 & 1 \\ 0 & 1 & 0 \end{pmatrix} \quad A^T = \begin{pmatrix} 2 & 3 & 5 & 0 \\ 4 & 2 & 3 & 1 \\ 7 & 0 & 1 & 0 \end{pmatrix}$$

00. Matriz transposta

$$A = \begin{pmatrix} 2 & 4 & 7 \\ 3 & 2 & 0 \\ 5 & 3 & 1 \\ 0 & 1 & 0 \end{pmatrix} \quad A^T = \begin{pmatrix} 2 & 3 & 5 & 0 \\ 4 & 2 & 3 & 1 \\ 7 & 0 & 1 & 0 \end{pmatrix}$$

```
def transposta(A):
    B = criar_matriz_zeros(len(A[0]), len(A))
    for i in range(0,len(A)):
        for j in range(0,len(A[0])):
            B[j][i] = A[i][j]
    return B
```

00. Matriz transposta

```
>>> P = criar_matriz_uns(5,7)
>>> visualizar_matriz(P)
[1, 1, 1, 1, 1, 1, 1]
[1, 1, 1, 1, 1, 1, 1]
[1, 1, 1, 1, 1, 1, 1]
[1, 1, 1, 1, 1, 1, 1]
[1, 1, 1, 1, 1, 1, 1]
```

00. Matriz transposta

```
>>> P = criar_matriz_uns(5,7)
>>> visualizar_matriz(P)
[1, 1, 1, 1, 1, 1, 1]
[1, 1, 1, 1, 1, 1, 1]
[1, 1, 1, 1, 1, 1, 1]
[1, 1, 1, 1, 1, 1, 1]
[1, 1, 1, 1, 1, 1, 1]
>>> Q = transposta(P)
>>> visualizar_matriz(Q)
[1, 1, 1, 1, 1]
[1, 1, 1, 1, 1]
[1, 1, 1, 1, 1]
[1, 1, 1, 1, 1]
[1, 1, 1, 1, 1]
[1, 1, 1, 1, 1]
[1, 1, 1, 1, 1]
```

01. Somatória de matrizes

Crie uma função que permita somar duas matrizes dadas como parâmetro.

Cabeçalho: def somar_matrizes(A,B):

$$A+B=\begin{bmatrix}1 & 3\\ 1 & 0\\ 1 & 2\end{bmatrix}+\begin{bmatrix}0 & 0\\ 7 & 5\\ 2 & 1\end{bmatrix}=\begin{bmatrix}1+0 & 3+0\\ 1+7 & 0+5\\ 1+2 & 2+1\end{bmatrix}=\begin{bmatrix}1 & 3\\ 8 & 5\\ 3 & 3\end{bmatrix}.$$

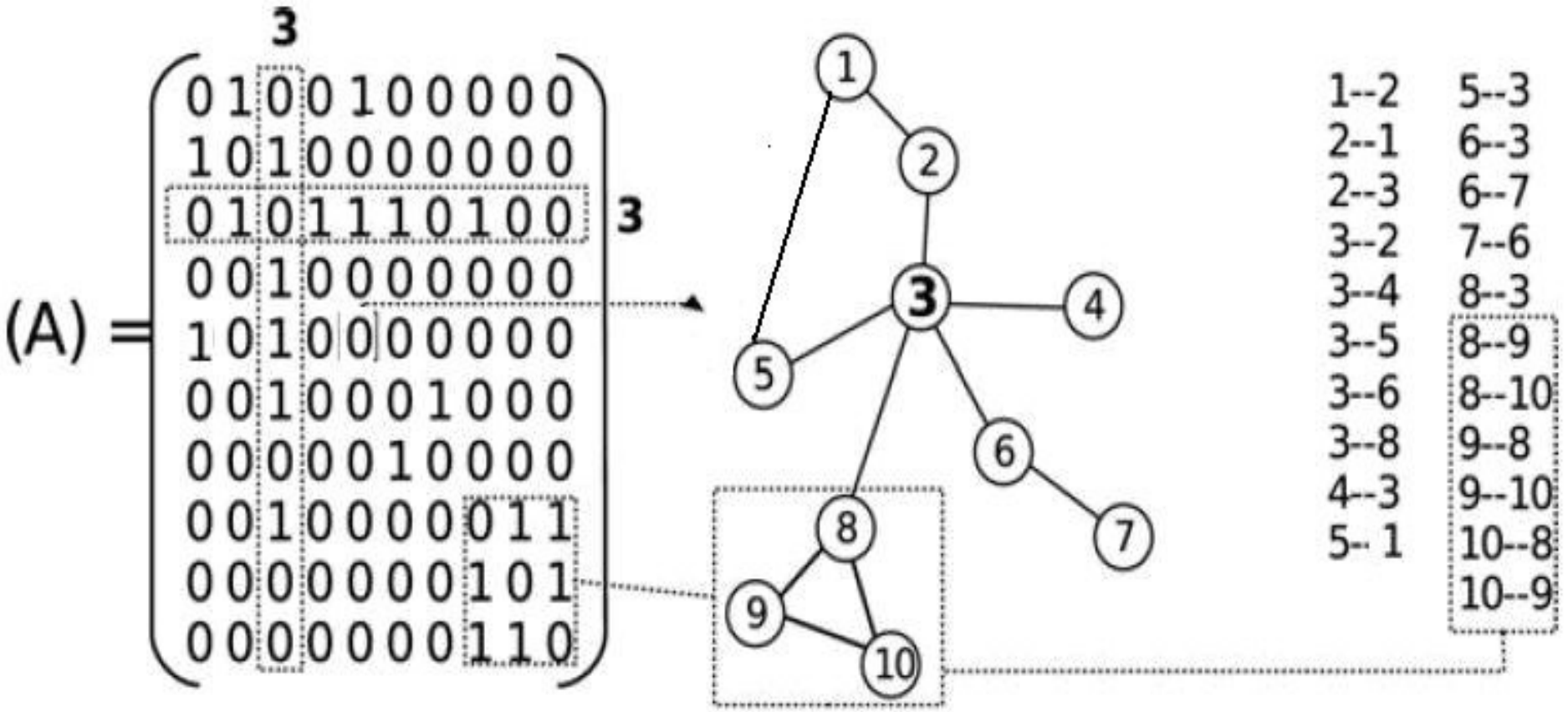
01. Somatória de matrizes

```
def somar_matrizes(A,B):
    if len(A)!=len(B) or len(A[0])!=len(B[0]):
        print 'Matrizes com dimensoes diferentes'
    else:
        C = criar_matriz_zeros(len(A),len(A[0]))
        for i in range(0,len(A)):
            for j in range(0,len(A[0])):
                C[i][j] = A[i][j]+B[i][j]
        return C
```

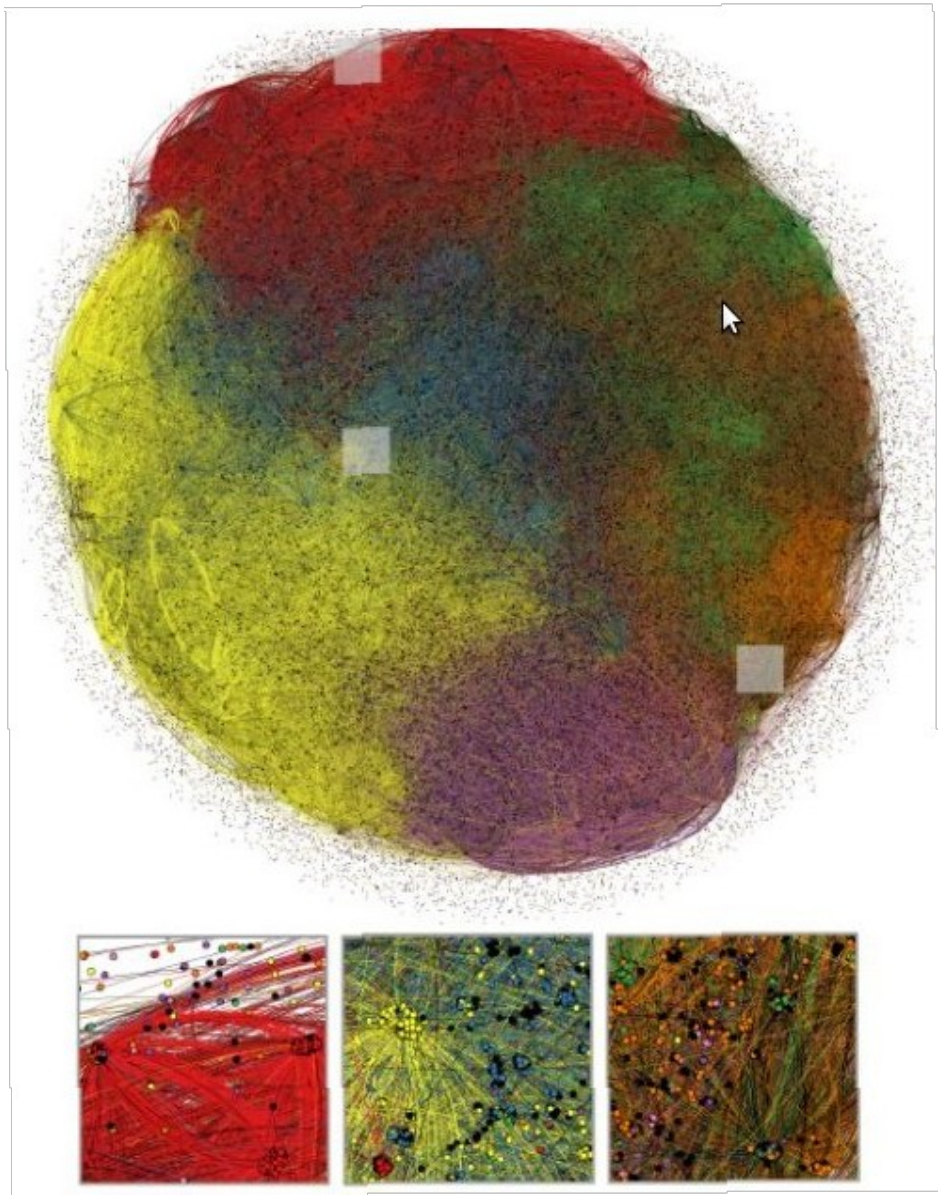
01. Somatória de matrizes

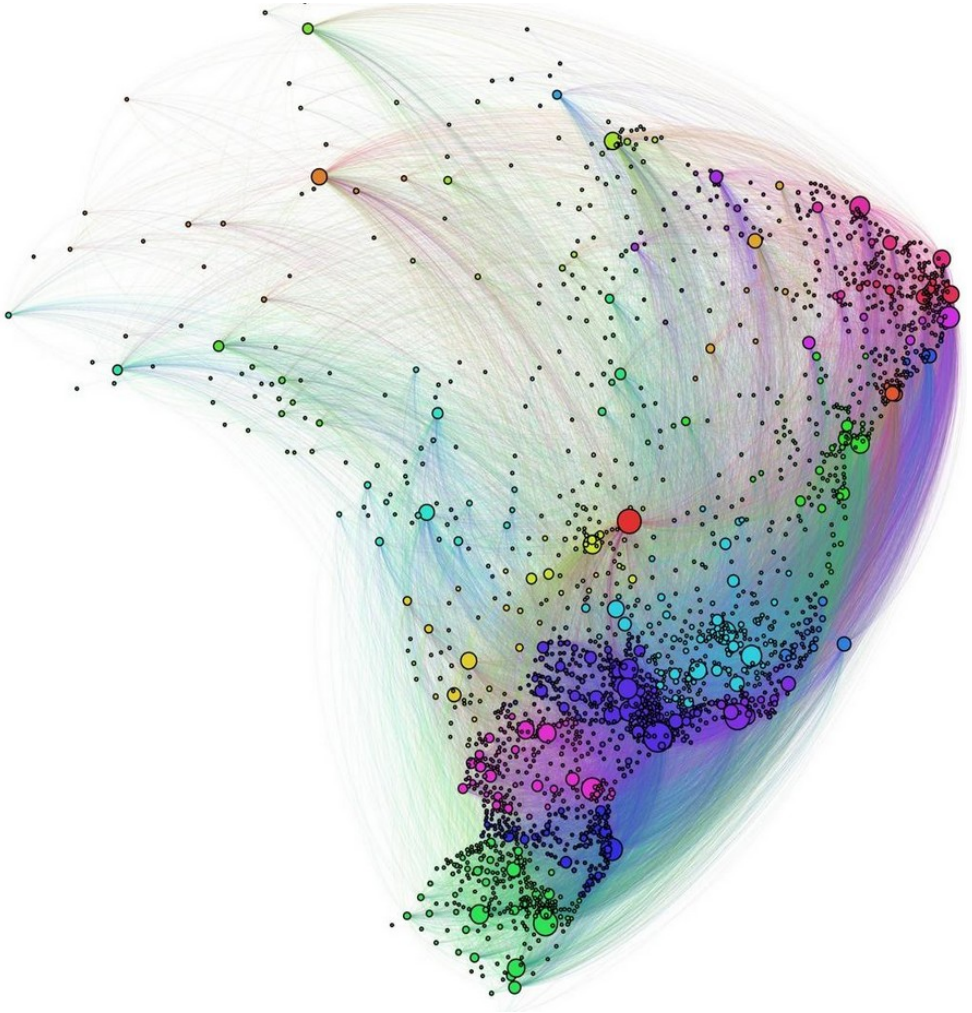
```
>>> A = criar_matriz_uns(2,4)
>>> B = criar_matriz_uns(2,4)

>>> C = somar_matrizes(A,B)

>>> print C
[[2, 2, 2, 2], [2, 2, 2, 2]]
```

Algumas aplicações com matrizes



Algumas aplicações com matrizes





02. Menor elemento

Crie uma função que permita determinar o menor elemento de uma matriz dada como parâmetro.

Cabeçalho: def menor_elemento(A):

[[-1,-2,-3,-4,-5,-6], [1,2,3,4,5,6], [7,8,9,10,11,12]] → -6

02. Menor elemento

```
def menor_elemento(A):
    menor = A[0][0]
    for i in range(0,len(A)):
        for j in range(0,len(A[0])):
            if menor>A[i][j]:
                menor = A[i][j]
    return menor
```

03. Segundo menor elemento

Crie uma função que permita determinar apenas o segundo menor elemento de uma matriz dada como parâmetro.

Cabeçalho: def segundo_menor_elemento(A)

[[-1,-2,-3,-4,-5,-6], [1,2,3,4,5,6], [7,8,9,10,11,12]] → -5

03. Segundo menor elemento

```
def segundo_menor_elemento(A):
    menor1 = A[0][0]
    menor2 = A[0][0]
    for i in range(0,len(A)):
        for j in range(0,len(A[0])):
            if menor1>A[i][j]:
                menor2 = menor1
                menor1 = A[i][j]
    return menor2
```

04. funcaoM2L

Indique o que realiza a seguinte função:

```
def funcaoM2L(M):
    L = [0]*len(M)*len(M[0])
    for i in range(0,len(M)):
        for j in range(0,len(M[0])):
            L[i*len(M[0])+j] = M[i][j]
    return L
```

Considere:

M = [[-1,-2,-3,-4], [1,2,3,4], [7,8,9,10]]

04. funcaoM2L

Função que converte uma matriz em uma lista

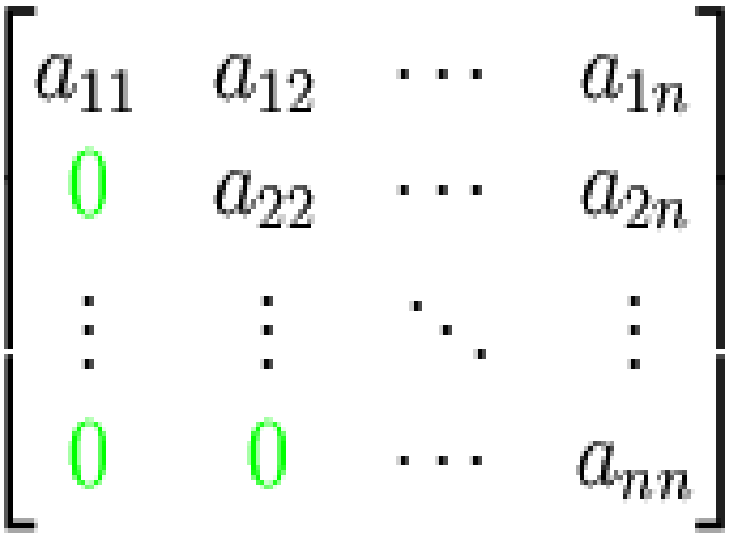
Se
M = [[-1,-2,-3,-4], [1,2,3,4], [7,8,9,10]]

Então:
L = [-1, -2, -3, -4, 1, 2, 3, 4, 7, 8, 9, 10]

05. Matriz triangular superior

Crie uma função que permita verificar se a matriz, dada como parâmetro, é triangular superior.

Cabeçalho: def matriz_triangular_superior(A):

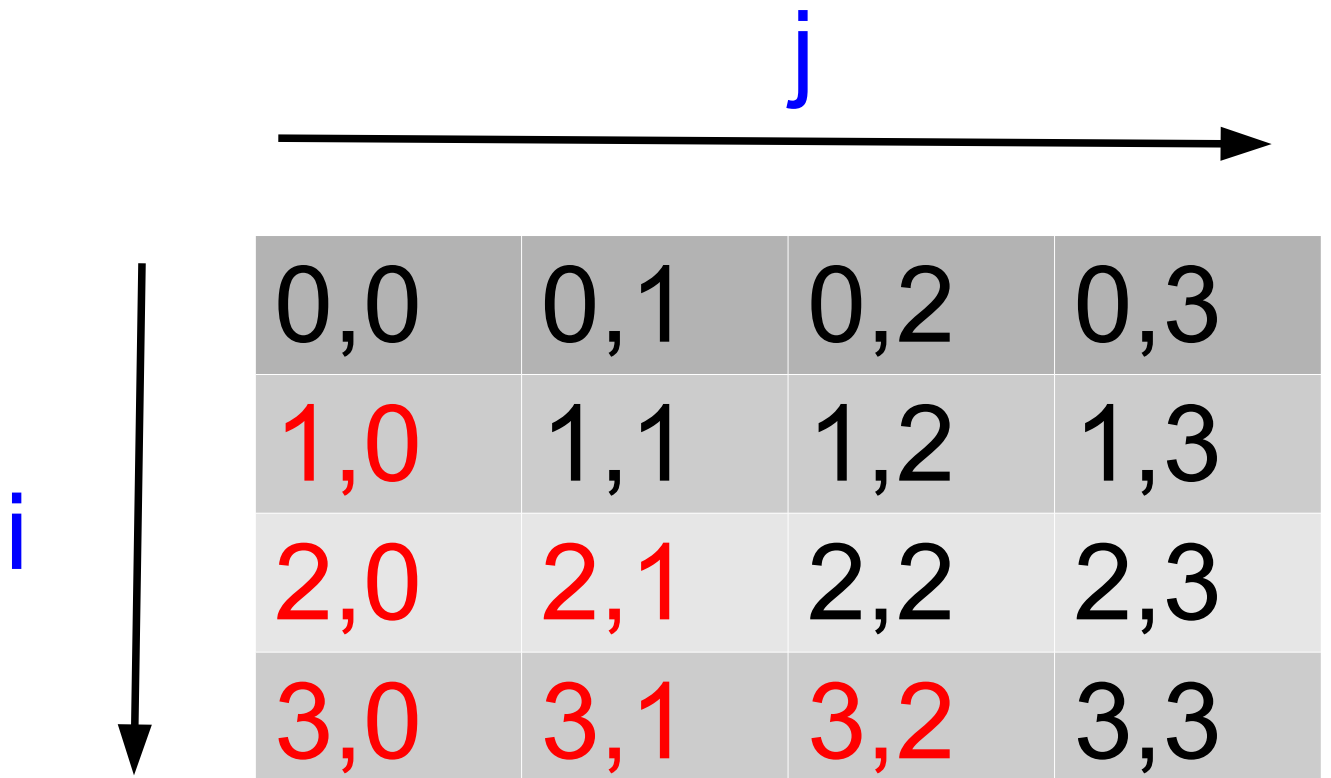


Caso particular de matrizes quadradas.

Abaixo da diagonal principal existem apenas elementos nulos.

Os restantes elementos estão posicionados acima dessa mesma diagonal, com a condição de não serem todos nulos.

05. Matriz triangular superior



Índices em uma matriz 4x4

05. Matriz triangular superior

```
def matriz_triangular_superior(A):
    if len(A[0])!=len(A):
        return False
    contador_zeros_inf = 0
    contador_zeros_sup = 0
    for i in range(0,len(A)):
        for j in range(0,len(A)):
            if i>j and A[i][j]==0:
                contador_zeros_inf +=1
            if i<j and A[i][j]==0:
                contador_zeros_sup +=1
    x = len(A)*(len(A)-1)/2
    if contador_zeros_inf==x and contador_zeros_sup!=x:
        return True
    else:
        return False
```

06. Multiplicação de matrizes (casa)

Crie uma função que permita **multiplicar** duas matrizes dadas como parâmetro.

Cabeçalho: def multiplicar_matrizes(A,B):

$$C = A \cdot B$$

$$A = \begin{pmatrix} a & b & c \\ d & e & f \end{pmatrix} \qquad B = \begin{pmatrix} g & h & i & j \\ k & l & m & n \\ o & p & q & r \end{pmatrix}$$

$$C = \begin{pmatrix} a*g+b*k+c*o & a*h+b*l+c*p & a*i+b*m+c*q & a*j+b*n+c*r \\ d*g+e*k+f*o & d*h+e*l+f*p & d*i+e*m+f*q & d*j+e*n+f*r \end{pmatrix}$$