CURSO MATRICES DISTRIBUIDAS

Sesión 2 - Fecha: 20 de junio 2021

Cifrado por transposición

In [4]:
```
mensaje = 'Matrices distribuidas'
transpuesto = ''
i = len(mensaje) - 1

while i >= 0:
    transpuesto = transpuesto + mensaje[i]
    i = i - 1
print ("Mensaje cifrado: \n", transpuesto)
```

```
Mensaje cifrado:
 sadiubirtsid secirtaM
```

Cifrado Cesar

In [8]:
```
def cifradoCesar(mensaje,recorrido):
    resultado = ""
    for i in range(len(mensaje)):
        caracter = mensaje[i]
        if (caracter.isupper()):
            resultado += chr((ord(caracter)+ recorrido - 65) % 26 + 65
        else:
            resultado += chr((ord(caracter)+ recorrido - 97) % 26 + 97
    return resultado


m = "DataSciencie"
r = 1
print("Mensaje cifrado: \n", cifradoCesar(m,r))
```

```
Mensaje cifrado:
 EbubTdjfodjf
```

Cifrado Vignere

In [10]:
```
LETTERS = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
def main():
   myMessage = "MATEMATICAS"
   myKey = 'PIZZA'
   myMode = 'encrypt'

   if myMode == 'encrypt':
       translated = encryptMessage(myKey, myMessage)
   elif myMode == 'decrypt':
       translated = decryptMessage(myKey, myMessage)

   print('%sed message:' % (myMode.title()))
   print(translated)
   print()

def encryptMessage(key, message):
    return translateMessage(key, message, 'encrypt')

def decryptMessage(key, message):
    return translateMessage(key, message, 'decrypt')

def translateMessage(key, message, mode):
    translated = [] # stores the encrypted/decrypted message string
    keyIndex = 0
    key = key.upper()

    for symbol in message:
        num = LETTERS.find(symbol.upper())
        if num != -1:
            if mode == 'encrypt':
                num += LETTERS.find(key[keyIndex])

            #elif mode == 'decrypt':
                #num -= LETTERS.find(key[keyIndex])
                num %= len(LETTERS)
            if symbol.isupper():
                translated.append(LETTERS[num])
            elif symbol.islower():
                translated.append(LETTERS[num].lower())
            keyIndex += 1

            if keyIndex == len(key):
                keyIndex = 0
        else:
             translated.append(symbol)
    return ''.join(translated)
```

```
if __name__ == '__main__':
    main()
```

```
Encrypted message:
BISDMPBHBAH
```

MATRICES DISPERSAS

Almacenamiento por coordenadas

In [14]:
```
import numpy as np

def coordenadas(matriz):
    # V = valores, I = renglones, J = Columnas
    V = []
    I = []
    J = []

    for i in range(len(matriz)):
        for j in range(len(matriz)):
            if (matriz[i,j]!=0):
                V.append(matriz[i,j])
                I.append(i)
                J.append(j)
    return(V,I,J)

def multiplicaMV(V,I,J,b):
    r = np.zeros(len(b))
    aux = I[0]
    suma = 0
    for i in range(len(I)):
        if (I[i] == aux):
            suma += V[i]*b[J[i]]
        else:
            r[I[i-1]]=suma
            suma =0
            suma += V[i]*b[J[i]]
        aux = I[i]
    r[I[-1]]=suma
    return r


m = np.matrix('3 0 4; 0 2 0; 0 0 1')
v,i,j = coordenadas(m)
b = np.array([5,2,3])
print(m)
print(b)
print(v,i,j)
print(multiplicaMV(v,i,j,b))
```

```
[[3 0 4]
 [0 2 0]
 [0 0 1]]
[5 2 3]
[3, 4, 2, 1] [0, 0, 1, 2] [0, 2, 1, 2]
[27.  4.  3.]
```

In [ ]: