# tarea6

July 20, 2020

## 1 Vignere con braile

Para la ejecución de este notebook es necesario poseer un entorno Jupyter disponible. Este puede ser instalado manualmente o bien por medio de un entorno de anaconda.

Para esta demostración vamos a utilizar las mismas cadenas de entrada del enunciado

[1]:
```
key = "UTALCA"
message = "UNIVERSIDAD"
order = "123456"
```

Para mantener las restricciones impuestas por la conversión, todos los caracteres son convertidos a mayúsculas

[2]:
```
key = key.upper()
message = message.upper()
```

Ahora ingresamos nuestro alfabeto. Dado que para estos efectos el alfabeto ya viene dado, este se define por el subconjunto de caracteres presentes en el español sin considerar los espacios. Sin embargo, por extensión de la definición del alfabeto, es inútil para efectos de uso porque el braile restringido a una sola celda no considera en su estándar la letra ñ. Esto es explicado más adelante.

[3]:
```
alphabet = "ABCDEFGHIJKLMNÑOPQRSTUVWXYZ"
```

Con el alfabeto cargado, se procede a cifrar el mensaje por medio de una búsqueda sobre la tabla inducida por el alfabeto.

[4]:
```
skey_lkt = [i % len(key) for i in range(0,len(message),1)]
cyphered = [(alphabet[(alphabet.index(key[idx%len(key)])+alphabet.
 ↪index(chara))%len(alphabet)]) for idx, chara in enumerate(message, start=0)]
```

[5]:
```
cyphered
```

[5]:
```
['O', 'G', 'I', 'G', 'G', 'R', 'N', 'B', 'D', 'L', 'F']
```

La definición de la tarea no es compilante con el estándar braille ASCII y dado que esta es la única representación valida para englobar un caracter con un espacio, se fuerza su uso, se ignora el alfabeto español. Adicionalmente como el mapeo presentado en la tarea tampoco es compilante con el estándar, se usa un mapeo para representar el string de orden, asumiendo un orden de bits LE. Para solventar este problema, se reemplazan los símbolos ofensores.

[6]:
```
braille_lexicon = " A1B'K2L@CIF/MSP\"E3H9O6R^DJG>NTQ,*5<-U8V.%[$+X!&;:4\\0Z7(_?
 ↪W]#Y)="
braille_alphabet = "                                        "

braille_cmessage = [braille_alphabet[braille_lexicon.index(i)] if i in␣
 ↪braille_lexicon else " " for i in cyphered]
braille_skey = [braille_alphabet[braille_lexicon.index(i)] if i in braille_lexicon␣
 ↪else " " for i in key]
```

[7]:
```
braille_cmessage
```

[7]:
```
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
```

[8]:
```
braille_skey
```

[8]:
```
[' ', ' ', ' ', ' ', ' ', ' ']
```

A este punto, se limpia la cadena para eliminar caracteres no aceptados en la conversión

[9]:
```
c_cyphered = [i if i in braille_lexicon else " " for i in cyphered]
```

Se genera el diccionario utilizando operaciones bitwise

[10]:
```
bidx = [
    ["1" if (braille_lexicon.index(i) & 0b000001) else "0" for i in c_cyphered],
    ["1" if (braille_lexicon.index(i) & 0b001000) else "0" for i in c_cyphered],
    ["1" if (braille_lexicon.index(i) & 0b000010) else "0" for i in c_cyphered],
    ["1" if (braille_lexicon.index(i) & 0b010000) else "0" for i in c_cyphered],
    ["1" if (braille_lexicon.index(i) & 0b000100) else "0" for i in c_cyphered],
    ["1" if (braille_lexicon.index(i) & 0b100000) else "0" for i in c_cyphered]
]
```

Finalmente, presentamos la permutación de acuerdo a string de orden

[11]:
```
" ".join(["".join(bidx[int(i)-1]) for i in order])
```

[11]:
```
'11011111111 01111010101 01111101011 11011110100 10000110010 00000000000'
```

El proceso especificado en el enunciado considera la salida encriptada sobrela cadena original, no observando el braile (quizás esto fue premeditado en base al fenómeno antes descrito). La reconversión del mensaje es como sigue:

[12]:
```
skey_lkt = [i % len(key) for i in range(0,len(message),1)]
decyphered = [(alphabet[(alphabet.index(chara) - alphabet.
 ↪index(key[idx%len(key)]))%len(alphabet)]) for idx, chara in␣
 ↪enumerate(cyphered, start=0)]
```

[13]:
```
decyphered
```

[13]:
```
['U', 'N', 'I', 'V', 'E', 'R', 'S', 'I', 'D', 'A', 'D']
```

## 2 Generación de fuente e instrucciones de uso

A continuación se muestra el código fuente para ser ejecutado en cualquier consola python. La generación es automática por medio de Jupyter, sin embargo basta con copiar y pegar el script que está debajo.

El ingreso de parámetros es por entrada estándar separados de un delimitador de línea. El orden de ingreso es mensaje, llave, cadena de orden.

Como el proceso de instalación y ejecución se ve en los primeros cursos de la carrera y es prerequisito de este módulo, se asume conocido por el lector.

[14]:
```
%%writefile tarea6.py

import sys
message, key, order = [line.strip() for line in sys.stdin.readlines()]

key = key.upper()
message = message.upper()

alphabet = "ABCDEFGHIJKLMNÑOPQRSTUVWXYZ"

skey_lkt = [i % len(key) for i in range(0,len(message),1)]
cyphered = [(alphabet[(alphabet.index(key[idx%len(key)])+alphabet.
 ↪index(chara))%len(alphabet)]) for idx, chara in enumerate(message, start=0)]
print(cyphered)

braile_lexicon = " A1B'K2L@CIF/MSP\"E3H9O6R^DJG>NTQ,*5<-U8V.%[$+X!&;:4\\0Z7(_?
 ↪W]#Y)="
braile_alphabet = "                                      "

braile_cmessage = [braile_alphabet[braile_lexicon.index(i)] if i in␣
 ↪braile_lexicon else " " for i in cyphered]
braile_skey = [braile_alphabet[braile_lexicon.index(i)] if i in braile_lexicon␣
 ↪else " " for i in key]

print(braile_cmessage)
print(braile_skey)

c_cyphered = [i if i in braile_lexicon else " " for i in cyphered]

bidx = [
    ["1" if (braile_lexicon.index(i) & 0b000001) else "0" for i in c_cyphered],
    ["1" if (braile_lexicon.index(i) & 0b001000) else "0" for i in c_cyphered],
    ["1" if (braile_lexicon.index(i) & 0b000010) else "0" for i in c_cyphered],
```

```
    ["1" if (braile_lexicon.index(i) & 0b010000) else "0" for i in c_cyphered],
    ["1" if (braile_lexicon.index(i) & 0b000100) else "0" for i in c_cyphered],
    ["1" if (braile_lexicon.index(i) & 0b100000) else "0" for i in c_cyphered]
]


print(" ".join(["".join(bidx[int(i)-1]) for i in order]))

skey_lkt = [i % len(key) for i in range(0,len(message),1)]
decyphered = [(alphabet[(alphabet.index(chara) - alphabet.
 ↪index(key[idx%len(key)]))%len(alphabet)]) for idx, chara in␣
 ↪enumerate(cyphered, start=0)]
print(decyphered)
```

```
Overwriting tarea6.py
```

A modo de ejemplo, presentamos la misma ejecución descrita al comienzo

[15]:
```
%%writefile example.in
UNIVERSIDAD
UTALCA
123456
```

```
Overwriting example.in
```

[16]:
```
%%bash
python3 tarea6.py < example.in
```

```
['O', 'G', 'I', 'G', 'G', 'R', 'N', 'B', 'D', 'L', 'F']
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
[' ', ' ', ' ', ' ', ' ', ' ']
11011111111 01111010101 01111101011 11011110100 10000110010 00000000000
['U', 'N', 'I', 'V', 'E', 'R', 'S', 'I', 'D', 'A', 'D']
```

[ ]: