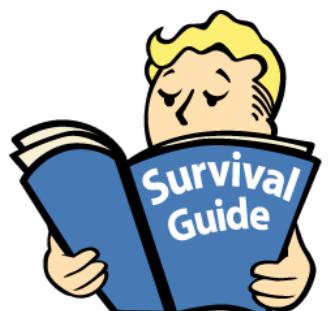


TÉCNICAS DE PROGRAMACIÓN

ORDENAMIENTO Y BÚSQUEDA

FUNCIONES ANÓNIMAS

```
add = lambda x, y: x + y  
print(add(3, 5)) # 8
```



MAP

MAP APLICA UNA FUNCIÓN A TODOS LOS ELEMENTOS DE UNA LISTA

```
items = [1, 2, 3, 4, 5, 6]
```

```
def multiply_elements_by_two(numbers):
    result = []
    for number in numbers:
        result.append(number * 2)
    return result
result = multiply_elements_by_two(items)
```

```
result = list(map(lambda x: x * 2, items))
```

FILTER

**FILTER CREA UNA LISTA DE ELEMENTOS PARA LOS QUE UNA
FUNCIÓN DEVUELVE TRUE**

```
items = [1, 2, 3, 4, 5, 6]
def get_low_than_three(numbers):
    result = []
    for number in numbers:
        if number < 3:
            result.append(number)
    return result
result = get_low_than_three(items)

result = list(filter(lambda x: x < 3, items))
```

REDUCE

REDUCE ES UNA FUNCIÓN APLICA UN CÁLCULO EN UNA LISTA Y DEVOLVIENDO UN ÚNICO RESULTADO.

```
items = [1, 2, 3, 4, 5, 6]

def get_average(numbers):
    result = 0
    for number in numbers:
        result += number
    return result / len(numbers)

average = get_average(items)

from functools import reduce
average = reduce((lambda x, y: x * y), items) / len(items)
```

REDUCE

REDUCE ES UNA FUNCIÓN APLICA UN CÁLCULO EN UNA LISTA Y DEVOLVIENDO UN ÚNICO RESULTADO.

```
items = [1, 2, 3, 4, 5, 6]

def get_average(numbers):
    result = 0
    for number in numbers:
        result += number
    return result / len(numbers)

average = get_average(items)

from functools import reduce
average = reduce((lambda x, y: x * y), items) / len(items)
```

USALOS CON CUIDADO

```
findShort = (s) => Math.min.apply(null, s.split(' ').map( (i) => i.length ))
```

```
function findShort (sentence) {
  words = sentence.split(' ')
  minWordSize = words[0].length

  words.forEach( word => {
    wordSize = word.length
    minWordSize = wordSize < minWordSize ? wordSize : minWordSize
  })

  return minWordSize
}
```



ALGORITMOS DE ORDENAMIENTO

SECTION SORT

1. DIVIDE LA INFORMACIÓN EN ORDENADO Y NO ORDENADO
2. ENCUENTRA EL ELEMENTO MÍNIMO DE LA SECCIÓN NO ORDENADA
3. AGRÉGALO A LA SECCIÓN ORDENADA



```
def section_sort(numbers):  
    result = numbers.copy()  
    for i in range(len(result)):  
        min_index = i  
        for j in range(i + 1, len(result)):  
            if result[min_index] > result[j]:  
                min_index = j  
        result[i], result[min_index] = result[min_index], result[i]  
    return result
```

$O(N^2)$

BUBBLE SORT

1. INICIA EL INDICE i EN 0
2. COMPARARA EL ELEMENTO $[i]$ E $[i + 1]$
3. SI EL ELEMENTO $[i + 1] > [i]$ INTERCAMBIAMOS DE LUGAR
4. REPITES CUANTAS VECES SEA NECESARIO

5 2 4 6 1 3

```
def bubble_sort(numbers):  
    result = numbers.copy()  
    size = len(numbers)  
  
    for i in range(size):  
        for j in range(0, size - i - 1):  
            if result[j] > result[j + 1]:  
                result[j], result[j + 1] = result[j + 1], result[j]  
    return result
```

$O(N^2)$

INSERTION SORT

1. INICIA EL INDICE **I** EN 0
2. COMPARA CON EL SIGUIENTE ELEMENTO
3. SI ES MAYO INCREMENTA EL INDICE
4. SI ES MENOR TOMA EL NUMERO MENOR Y ENCUENTRA SU LUGAR EN UN INDICE ANTERIOR

6 5 3 1 8 7 2 4

```
def insertion_sort(numbers):  
    result = numbers.copy()  
    for i in range(1, len(result)):  
        key = result[i]  
        j = i - 1  
        while j >= 0 and key < result[j] :  
            result[j+1] = result[j]  
            j -= 1  
        result[j + 1] = key  
    return result
```

$O(N^2)$

MERGE SORT

1. DIVIDE EL ARREGLO EN DOS
2. SE PUEDE SEGUIR DIVIDIENDO EN 2, DIVIDE
3. JUSTA Y ORDENA

6 5 3 1 8 7 2 4

MERGE SORT

```
def merge(left, right):  
    left_size = len(left)  
    right_size = len(right)  
    if not left_size or not right_size:  
        return left or right  
  
    result = []  
    i, j = 0, 0  
    while (len(result) < left_size + right_size):  
        if left[i] < right[j]:  
            result.append(left[i])  
            i += 1  
        else:  
            result.append(right[j])  
            j += 1  
        if i == left_size or j == right_size:  
            result.extend(left[i:] or right[j:])  
            break  
    return result
```

```
def merge_sort(numbers):  
    if len(numbers) < 2:  
        return numbers  
    middle = len(numbers) // 2  
    left = merge_sort(numbers[:middle])  
    right = merge_sort(numbers[middle:])  
    return merge(left, right)
```

O(N LOG N)

QUICK SORT

1. ELEGIR UN ELEMENTO ALEATORIO DEL MEDIO DEL ARREGLO
2. RESITUA LOS DEMÁS ELEMENTOS DE LA LISTA A CADA LADO DEL PIVOTE MENORES A LA IZQUIERDA, MAYORES A LA DERECHA
3. REPITE CON LAS DOS NUEVAS LISTA IZQUIERDA Y DERECHA



QUICK SORT

```
def quick_sort(numbers):  
    size = len(numbers)  
    if size == 1:  
        return numbers  
  
    middle = size // 2  
    pivot = numbers[middle]  
    low_numbers = []  
    high_numbers = []  
  
    for index, number in enumerate(numbers):  
        if index == middle:  
            continue  
        if number > pivot:  
            high_numbers.append(number)  
        else:  
            low_numbers.append(number)  
    return quick_sort(low_numbers) + [pivot] + quick_sort(high_numbers)
```

$O(N \log N)$

ALGORITMOS DE BÚSQUEDA

BÚSQUEDA LINEAL

Linear Search



```
def search(arr, x):  
    for i in range(len(arr)):  
        if arr[i] == x:  
            return i  
    return -1
```

O(N)

BÚSQUEDA BINARIA

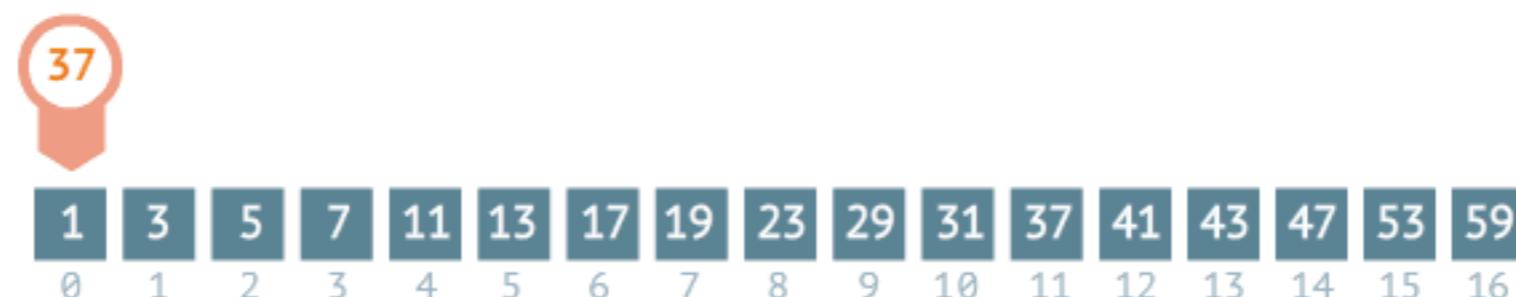
Binary search

steps: 0



Sequential search

steps: 0



ARCHIVOS

ARCHIVOS

SIMPLE TEXT FILE

.TXT

```
data_file = open('testfile.txt', 'w')
data_file.write('Hello World')
data_file.close()
```

MODES FILE

Modo	Descripción
r	Read Only
rb	Read Only Binary
r+	Read and Write, el puntero aparece al inicio del archivo
rb+	Read and Write Binary, el puntero aparece al inicio del archivo
w	Write Only, sobre escribe todo el archivo, crea el archivo si no existe
wb	Write Only Binary, sobre escribe todo el archivo, crea el archivo si no existe
w+	Read and Write, sobre escribe todo el archivo, crea el archivo si no existe
wb+	Read and Write Binary, sobre escribe todo el archivo, crea el archivo si no existe
a	Append, el puntero esta al final de archivo, si el archivo no existe lo crea.
ab	Append Binary, el puntero esta al final de archivo, si el archivo no existe lo crea.
a+	Append and Read Binary, el puntero esta al final de archivo, si el archivo no existe lo crea.
ab+	Append and Read Binary, el puntero esta al final de archivo, si el archivo no existe lo crea.

COMMA-SEPARATED VALUES FILE

.CSV

JAVA SCRIPT OBJECT NOTATION

.JSON

NOTA: EL MANEJO DE JSON EN PYTHON ES A TRAVÉS DE UN PARSER

JSON

```
{  
  "name": "Miguel",  
  "age": 25,  
  "is_alive": True,  
  "friends": ["Hugo", "Paco", "Luis"],  
  "kids": None  
}
```

```
{  
  "name": "Miguel",  
  "age": 25,  
  "is_alive": true,  
  "friends": ["Hugo", "Paco", "Luis"],  
  "kids": null  
}
```

RETO 10: GAME RANKING

SUPONIENDO QUE TRABAJAS EN UNA EMPRESA DE VIDEO JUEGOS QUE TIENE UN VIDEO JUEGO MUY EXITOSO CON UNA GRAN CANTIDAD DE USUARIOS.

SE TE DA LA TAREA DE MEJORAR EL ALGORITMO DE ORDENAMIENTO DEL RANKING DE LOS JUGADORES, YA QUE SE ESTÁN QUEJANDO QUE EL RANKING SE ACTUALIZA MUY LENTO. ACTUALMENTE SE ESTÁ UTILIZANDO UN ALGORITMO DE $O(N \log N)$.

- ▶ GENERAL UNA FUNCIÓN LLAMADA **SORT_BY_SCORE**, LA CUAL DEBE DE RECIBIR UN ARREGLO DE DICCIONARIOS CON LA INFORMATION DE CADA USUARIO Y RETORNA UN ARREGLO DE DICCIONARIOS ORDENADOS
- ▶ DEBE OPERAR EN $O(N)$
- ▶ ACTUALMENTE LA BASE DE DATOS DE LA EMPRESA TE REGRESA UN ARCHIVO **.CSV**, LLAMADO **USERS . CSV**
- ▶ TE PIDEN ALMACENAR EL RESULTADO ORDENADO COMO UN ARCHIVO EN FORMATO **.JSON**, CON EL NOMBRE **SORTED_USERS . JSON**



AMOR Y DINERO

AMOR Y DINERO

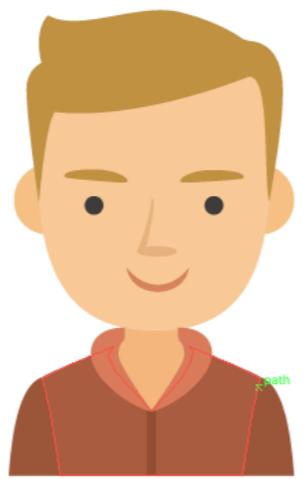
PROBLEMA DE LAS PAREJA ESTABLE



PROBLEMA DE LAS PAREJA ESTABLE



PAULA



ROMEO



JUELIETA



JORGE

- 1. MISMO NUMERO DE HOMBRES QUE DE MUJERES**
- 2. TODOS DEBEN TENER PAREJA**
- 3. NO CREAR PAREJAS INESTABLES**
- 4. SOLO MATCH ENTRE GÉNEROS**

PROBLEMA DE LAS PAREJA ESTABLE



MARCO

MARIA
SOFIA
ANA
JULIETA



JUAN

MARIA
JULIETA
SOFIA
ANA



HUGO

MARIA
SOFIA
ANA
JULIETA



PEDRO

MARIA
JULIETA
SOFIA
ANA



MARIA

MARCO
JUAN
HUGO
PEDRO



SOFIA

MARCO
JUAN
HUGO
PEDRO



ANA

JUAN
HUGO
PEDRO
MARCO



JULIETA

PEDRO
JUAN
HUGO
JUAN

DIA 1

DIA 2

DIA 3

DIA 4

DIA 5

MARIA

SOFIA

ANA

JULIETA

RETO 11: PAREJA FELIZ

**IMPLEMENTA EL ALGORITMO VISTO EN CLASE PARA HACER MATCH Y RESPONDE EL COMENTARIO
SEPARA LAS FUNCIONES DE MATCH DE LAS FUNCIONES DE MANEJO DE ARCHIVOSCOMPLETA EL
COMENTARIO FINAL**

SI LAS MUJERES PROPONEN

'ALLISON ESTRADA CARR' ->

'NIXON LEONARD ROSE' ->

'JEANNIE MORTON NOBLE' ->

SI LOS HOMBRES PROPONEN

'WILLIAM FRANCIS BENJAMIN' ->

'BUTLER MATTHEWS MUÑOZ' ->

'HOLLAND JENNINGS ENGLISH' ->



Pareja Feliz

ALGORITMOS DE CIFRADO Y LOS DINEROS

SSH



HTTPS

SCL



SHA-1 SECURE HASH ALGORITHM

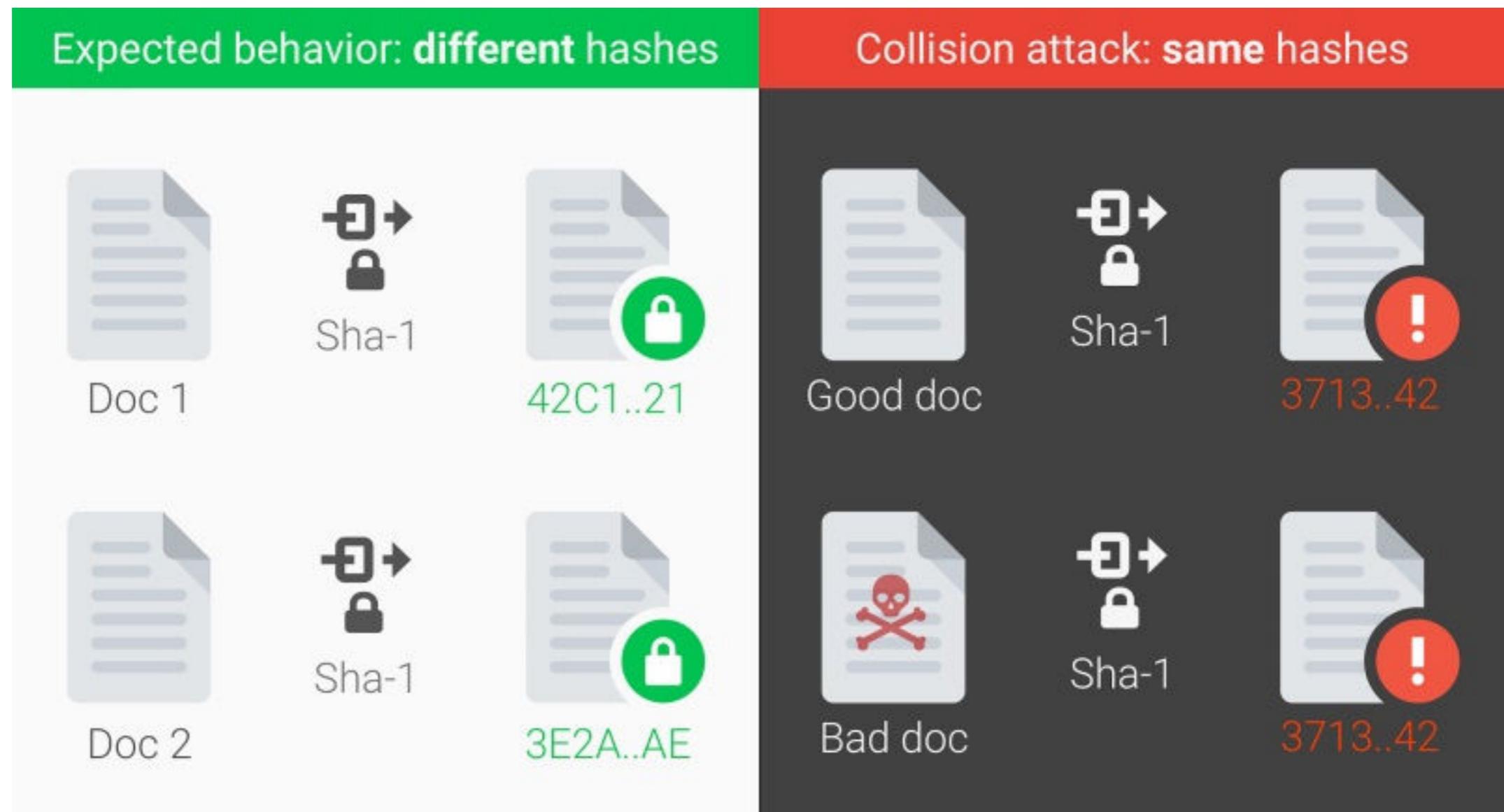
HOLA MAMA → AE23JV232

ESTE ES OTRO UN MENSAJE → WF53D62D

HOLA MAMA. → GSGJ29D3

LA HUELLA DE UN ARCHIVO

ALGORITMOS DE CIFRADO



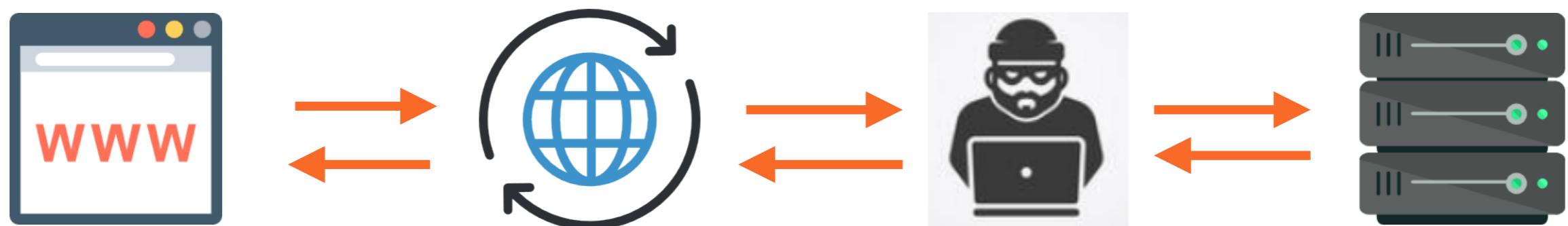
GUARDAR CONTRASEÑAS

VERIFICAR ARCHIVOS

ALGORITMOS DE CIFRADO



ALGORITMOS DE CIFRADO

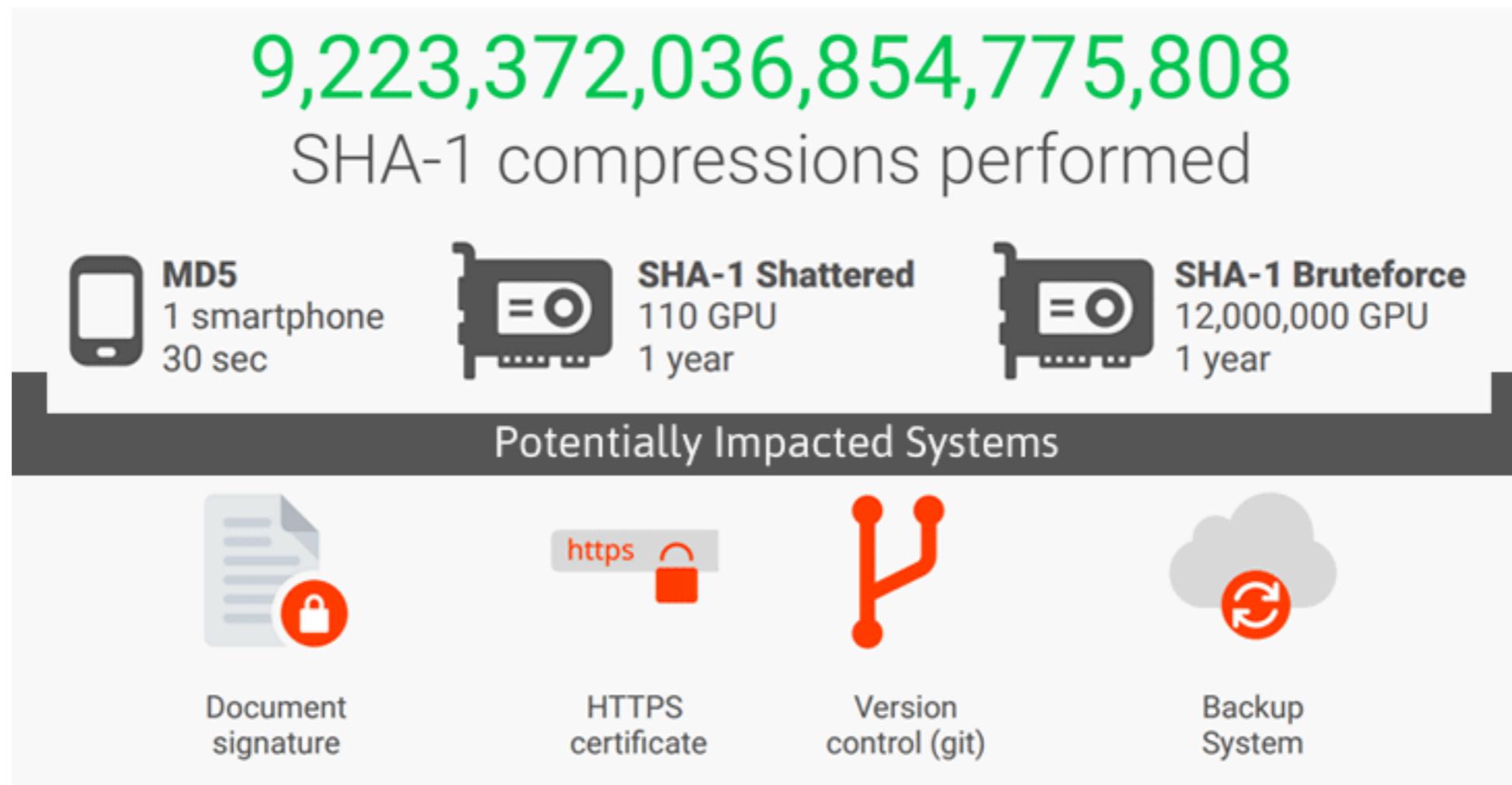


ALGORITMOS DE CIFRADO



**¿ENTONCES QUE ENCONTRO GOOGLE?
¿COMO LE HIZO?**

ALGORITMOS DE CIFRADO



HTTPS://ALF.NU/SHA1

```
$ hexdump -vC shattered-1.pdf
00000000 25 50 44 46 2d 31 2e 33 0a 25 e2 e3 cf d3 0a 0a
00000010 0a 31 20 30 20 6f 62 6a 0a 3c 3c 2f 57 69 64 74
00000020 68 20 32 20 30 48 65 67 68 74 20 33
00000030 20 30 20 52 2f 40 20 30 20 52 2f
00000040 53 75 62 74 79 48 65 67 68 74 20 33
00000050 6c 74 65 72 20 36 20 30 20 52 2f 43 6f 6c 6f 72
00000060 52 74 50 20 37 20 30 20 52 2f 4c 65 6e 67
00000070 74 6f 6d 70 6f 6e 65 6e 74 20 30 20 52 2f 42 6
00000080 43 6f 6d 70 6f 6e 65 6e 74 20 30 20 52 2f 4c 65 6e
00000090 72 65 61 6d 0a ff d8 ff fe 00 24 53 48 41 2d 31
000000a0 20 69 73 20 64 65 61 64 21 21 21 21 21 85 2f ec
000000b0 09 23 39 75 9c 39 b1 a1 c6 3c 4c 97 e1 ff fe 01
000000c0 73 46 dc 91 66 b6 7e 11 8f 02 9a b6 21 b2 56 0f
000000d0 f9 ca 67 cc a8 c7 f8 5b a8 4c 79 03 0c 2b 3d e2
000000e0 18 f8 6d b3 a9 09 01 d5 df 45 c1 4f 26 fe df b3
000000f0 dc 38 e9 6a Comment length = 0x173
00000100 3c 57 0f eb a8 2b e3 31
00000110 fe a4 80 37 b8 b5 d7 1f 0e 33 2e df 93 ac 35 00
00000120 eb 4d dc 0d ec c1 a8 64 79 0c 78 2c 76 21 56 60
00000130 dd 30 97 91 d0 6b d0 af 3f 98 cd a4 bc 46 29 b1
00000140 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000150 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000160 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000170 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000180 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000190 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001e0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001f0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000200 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000210 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000220 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000230 00 00 ff fe 00 fc 00 00 00 00 00 00 00 ff e0
00000240 00 10 4a 46 49 46 00 01 01 01 00 48 00 48 00 00
00000250 ff db 00 43 00 01 01 01 01 01 01 01 01 01 01 01
00000260 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
00000270 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
00000280 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
00000290 01 01 01 01 01 ff db 00 43 01 01 01 01 01 01 01 01
000002a0 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
000002b0 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
000002c0 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
000002d0 01 01 01 01 01 01 01 01 01 01 ff c2 00 11 08 02
000002e0 e4 04 00 Real JPEG data starts much later...
000002f0 1e 00 01 Huffman tables
00000300 00 00 00 07 08 05 06 09 03 04 0a 02 01 ff c4 00
00000310 1d 01 01 00 01 05 00 00 00 00 00 00 00 00 00 00
00000320 00 00 00 07 03 04 01 01 ff fe 00 06
00000330 ff fe 27 f4 ff da 00 0c 03 01 00 02 10 03 10 00
00000340 00 01 a1 fa ff 00 d8 c0 00 00 00 00 00 00 00 00 00

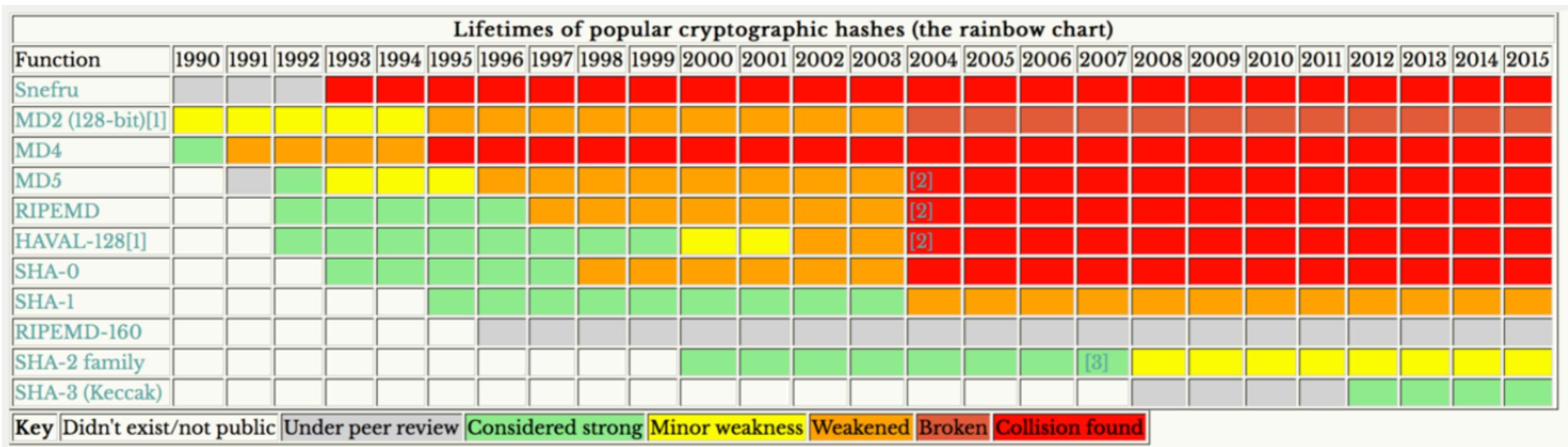
$ hexdump -vC shattered-2.pdf
00000000 25 50 44 46 2d 31 2e 33 0a 25 e2 e3 cf d3 0a 0a
00000010 0a 31 20 30 20 6f 62 6a 0a 3c 3c 2f 57 69 64 74
00000020 68 20 32 20 30 48 65 67 68 74 20 33
00000030 20 30 20 52 2f 40 20 30 20 52 2f
00000040 53 75 62 74 79 48 65 67 68 74 20 33
00000050 6c 74 65 72 20 36 20 30 20 52 2f 43 6f 6c 6f 72
00000060 52 74 50 20 37 20 30 20 52 2f 4c 65 6e 67
00000070 74 6f 6d 70 6f 6e 65 6e 74 20 30 20 52 2f 42 6
00000080 43 6f 6d 70 6f 6e 65 6e 74 20 30 20 52 2f 4c 65 6e
00000090 72 65 61 6d 0a ff d8 ff fe 00 24 53 48 41 2d 31
000000a0 20 69 73 20 64 65 61 64 21 21 21 21 21 85 2f ec
000000b0 09 23 39 75 9c 39 b1 a1 c6 3c 4c 97 e1 ff fe 01
000000c0 73 46 dc 91 66 b6 7e 11 8f 02 9a b6 21 b2 56 0f
000000d0 f9 ca 67 cc a8 c7 f8 5b a8 4c 79 03 0c 2b 3d e2
000000e0 18 f8 6d b3 a9 09 01 d5 df 45 c1 4f 26 fe df b3
000000f0 dc 38 e9 6a Comment length = 0x17F
00000100 3c 57 0f eb a8 2b e3 31
00000110 fe a4 80 37 b8 b5 d7 1f 0e 33 2e df 93 ac 35 00
00000120 eb 4d dc 0d ec c1 a8 64 79 0c 78 2c 76 21 56 64
00000130 dd 30 97 91 d0 6b d0 af 3f 98 cd a4 bc 46 29 a1
00000140 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000150 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000160 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000170 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000180 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000190 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001e0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001f0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000200 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000210 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000220 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000230 00 00 ff fe 00 fc 00 00 00 00 00 00 00 ff e0
00000240 00 10 4a 46 49 46 00 01 01 01 00 48 00 48 00 00
00000250 ff db 00 43 00 01 01 01 01 01 01 01 01 01 01 01
00000260 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
00000270 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
00000280 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
00000290 01 01 01 01 01 ff db 00 43 01 01 01 01 01 01 01 01
000002a0 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
000002b0 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
000002c0 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
000002d0 01 01 01 01 01 01 01 01 01 01 ff c2 00 11 08 02
000002e0 e4 04 00 JPEG Comment
000002f0 1e 00 01 Huffman tables
00000300 00 00 00 07 08 05 06 09 03 04 0a 02 01 ff c4 00
00000310 1d 01 01 00 01 05 00 00 00 00 00 00 00 00 00 00
00000320 00 00 00 07 03 04 01 01 ff fe 00 06
00000330 ff fe 27 f4 ff da 00 0c 03 01 00 02 10 03 10 00
00000340 00 01 a1 fa ff 00 d8 c0 00 00 00 00 00 00 00 00 00

$ hexdump -vC shattered-3.pdf
00000000 %PDF-1.3.%...%PDF-1.3.%...%PDF-1.3.%...%PDF-1.3.%...
00000010 .1 0 obj.<</Widt .1 0 obj.<</Widt .1 0 obj.<</Widt .1 0 obj.<</Widt .1 0 obj.<</Widt
00000020 h 2 0 R/Height 3 h 2 0 R/Height 3
00000030 0 R/Type 4 0 R/ 0 R/Type 4 0 R/
00000040 Subtype 5 0 R/Fi Subtype 5 0 R/Fi Subtype 5 0 R/Fi Subtype 5 0 R/Fi Subtype 5 0 R/Fi
00000050 lter 6 0 R/Color lter 6 0 R/Color lter 6 0 R/Color lter 6 0 R/Color lter 6 0 R/Color
00000060 Space 7 0 R/Leng Space 7 0 R/Leng Space 7 0 R/Leng Space 7 0 R/Leng Space 7 0 R/Leng
00000070 th 8 0 R/BitsPer th 8 0 R/BitsPer th 8 0 R/BitsPer th 8 0 R/BitsPer th 8 0 R/BitsPer
00000080 Component 8>.st ream. Component 8>.st ream. Component 8>.st ream. Component 8>.st ream. Component 8>.st ream.
00000090 is dead!!!!/. is dead!!!!/. is dead!!!!/. is dead!!!!/. is dead!!!!/.
000000a0 .#9u.9...<L .#9u.9...<L .#9u.9...<L .#9u.9...<L .#9u.9...<L
000000b0 sF..f.~...!V. sF..f.~...!V. sF..f.~...!V. sF..f.~...!V. sF..f.~...!V.
000000c0 ..g...[.Ly.+=. ..g...[.Ly.+=. ..g...[.Ly.+=. ..g...[.Ly.+=. ..g...[.Ly.+=.
000000d0 ..m...E.0&.. ..m...E.0&.. ..m...E.0&.. ..m...E.0&.. ..m...E.0&
000000e0 .8.j./.r..E.F. .8.j./.r..E.F. .8.j./.r..E.F. .8.j./.r..E.F. .8.j./.r..E.F.
000000f0 <W...U...+.1 <W...U...+.1 <W...U...+.1 <W...U...+.1 <W...U...+.1
00000100 7...3...5. M...dy.x,v!V. M...dy.x,v!V. M...dy.x,v!V. M...dy.x,v!V. M...dy.x,v!V.
00000110 0...k...?...F. 0...k...?...F. 0...k...?...F. 0...k...?...F. 0...k...?...F.
00000120 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000130 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000140 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000150 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000160 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000170 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000180 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000190 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001e0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001f0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000200 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000210 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000220 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000230 00 00 ff fe 00 fc 00 00 00 00 00 00 00 ff e0
00000240 00 10 4a 46 49 46 00 01 01 01 00 48 00 48 00 00
00000250 ff db 00 43 00 01 01 01 01 01 01 01 01 01 01 01
00000260 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
00000270 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
00000280 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
00000290 01 01 01 01 01 ff db 00 43 01 01 01 01 01 01 01 01
000002a0 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
000002b0 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
000002c0 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
000002d0 01 01 01 01 01 01 01 01 01 01 ff c2 00 11 08 02
000002e0 e4 04 00 Quantization table
000002f0 1e 00 01 Quantization table
00000300 00 00 00 07 03 04 01 11 00 02 11 01 03 11 01 ff c4 00
00000310 1d 01 01 00 01 05 00 02 03 00 03 01 01 00 00 00 00 00
00000320 00 00 00 07 03 04 01 01 ff fe 00 06
00000330 ff fe 27 f4 ff da 00 0c 03 01 00 02 10 03 10 00
00000340 00 01 a1 fa ff 00 d8 c0 00 00 00 00 00 00 00 00 00
```

Annotations:

- Fixed:** Labels the first 173 bytes of the PDF header.
- Variable:** Labels the variable part of the PDF header, starting at byte 174.
- Collision blocks:** Labels the collision blocks between the two PDF headers.
- Interleaving:** Labels the interleaving of the two large comments.
- Real JPEG data starts much later...** Labels the real JPEG data starting at byte 291.
- JPEG Comment:** Labels the first JPEG

ALGORITMOS DE CIFRADO

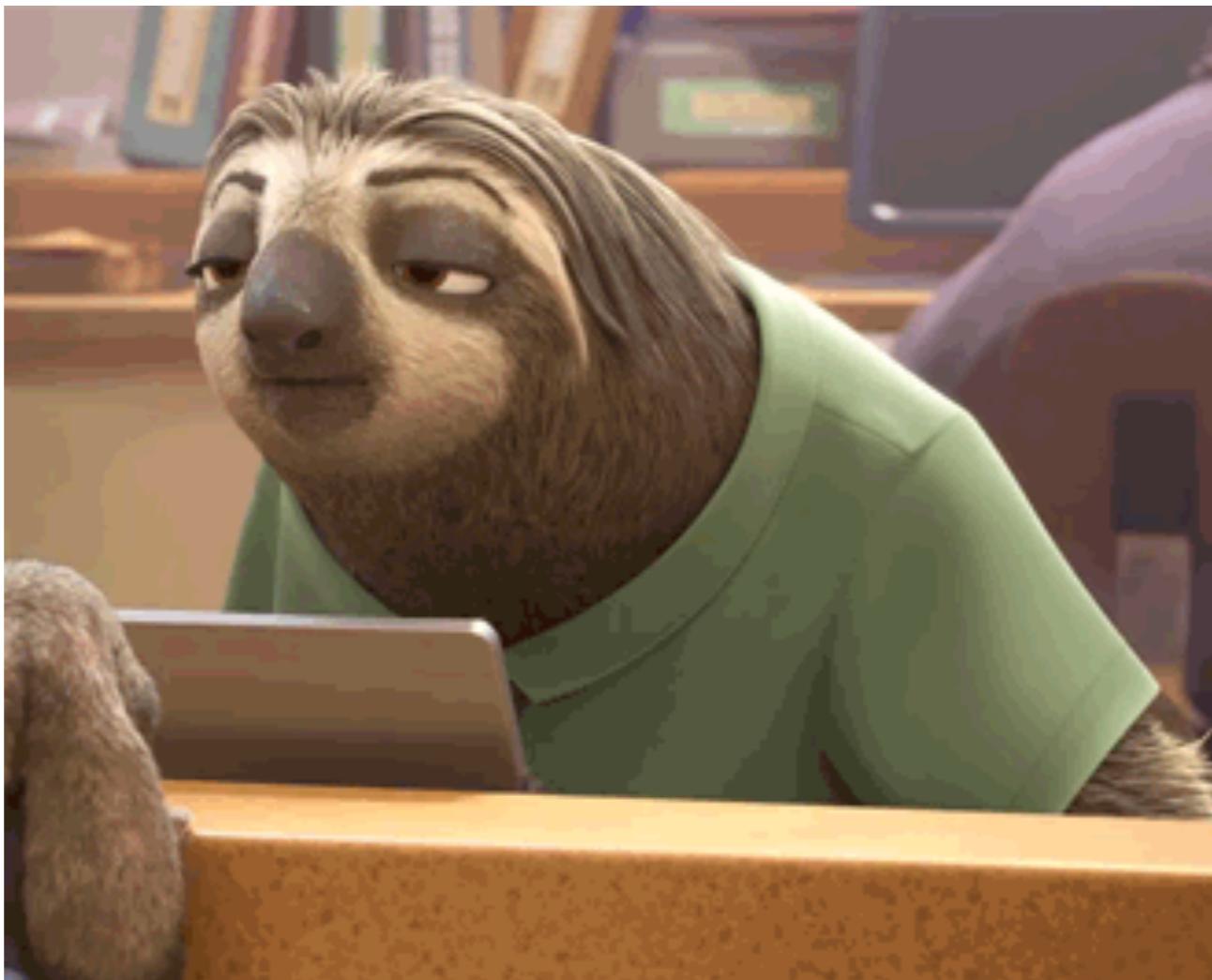


¿POR QUE ES PREOCUPANTE?

ALGORITMOS DE CIFRADO



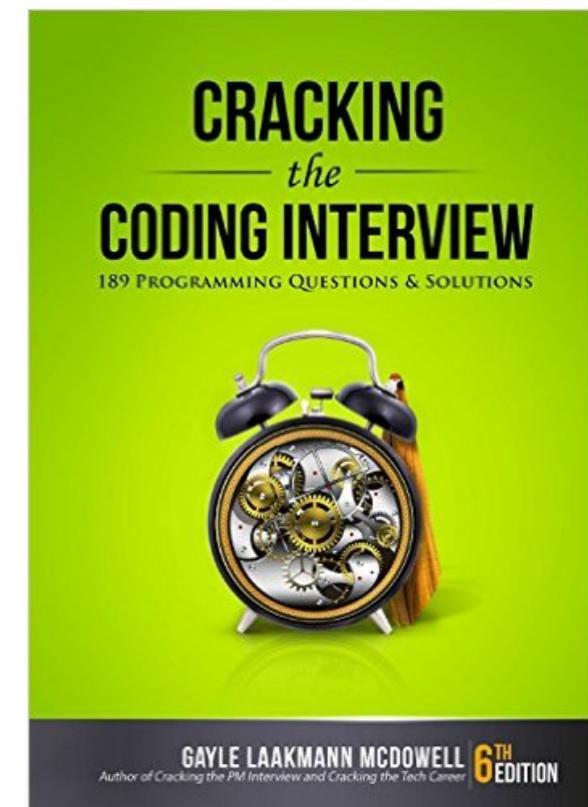
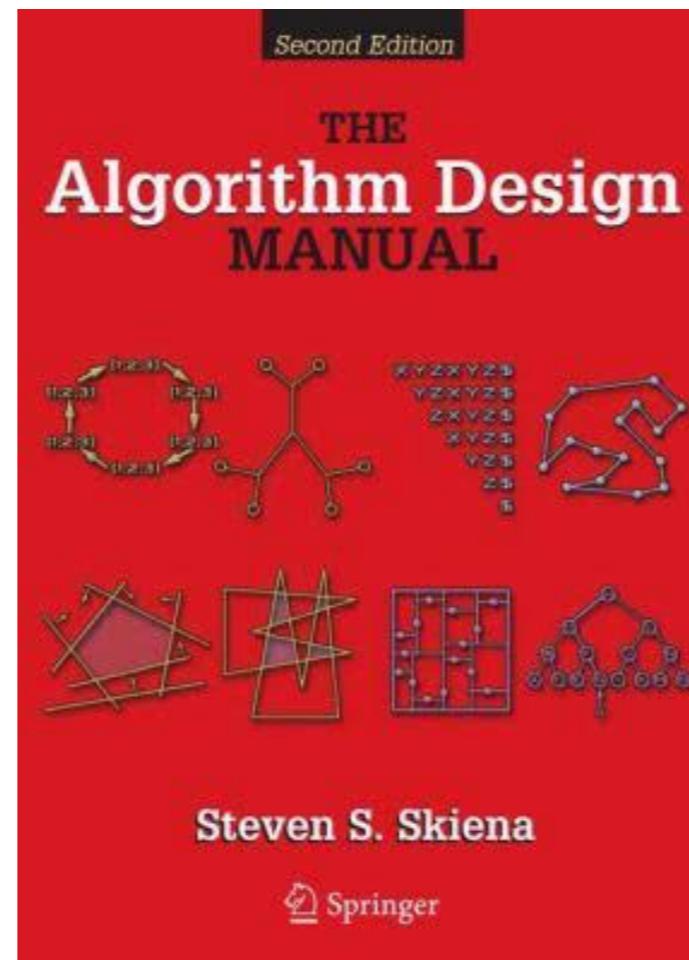
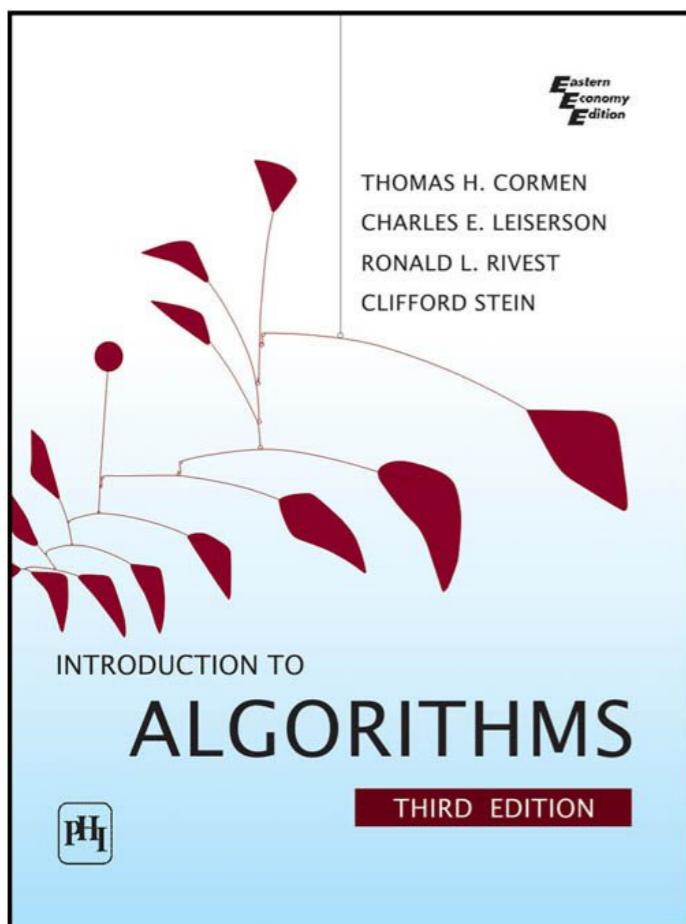
ALGORITMOS DE CIFRADO



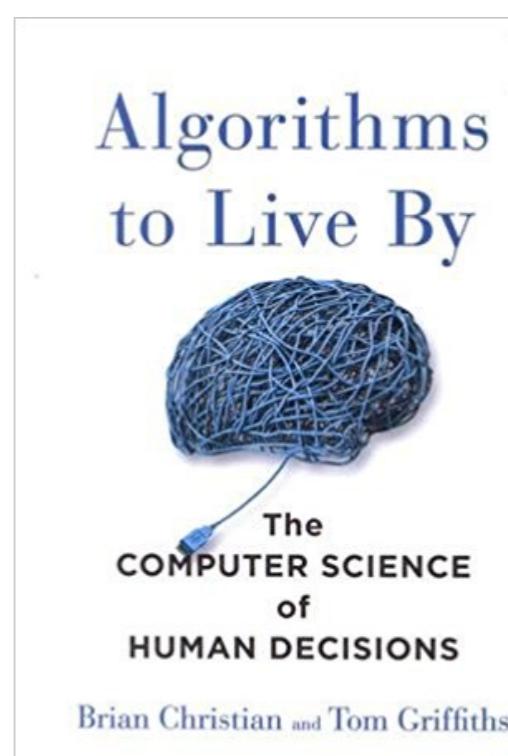
RSA-2048
US\$200,000

```
251959084756578934940271832400483985714292821262040320277713783604366202070  
7595556264018525880784406918290641249515082189298559149176184502808489120072  
8449926873928072877767359714183472702618963750149718246911650776133798590957  
0009733045974880842840179742910064245869181719511874612151517265463228221686  
9987549182422433637259085141865462043576798423387184774447920739934236584823  
8242811981638150106748104516603773060562016196762561338441436038339044149526  
3443219011465754445417842402092461651572335077870774981712577246796292638635  
6373289912154831438167899885040445364023527381951378636564391212010397122822  
120720357
```

CONCLUSIÓN



MIT 6.006 INTRODUCTION TO ALGORITHMS, FALL 2011
MIT 6.854 (ADVANCED ALGORITHMS), SPRING 2016



- 1. ALGORITMOS DE ORDENAMIENTO**
- 2. ALGORITMOS DE BÚSQUEDA**
- 3. MANEJO Y TIPOS DE ARCHIVOS**
- 4. OTROS ALGORITMOS**