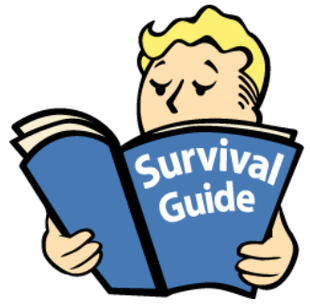
TÉCNICAS DE PROGRAMACIÓN

ORDENAMIENTO Y BÚSQUEDA

FUNCIONES ANÓNIMAS

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



anónimas

MAP

MAP APLICA UNA FUNCIÓN A TODOS LOS ELEMENTOS DE UNA LISTA

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

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

result = multiplay_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 (sentece) {
  words = sentece.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


```python
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

```python
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

```python
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

```python
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
```

```python
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 LOGN)

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

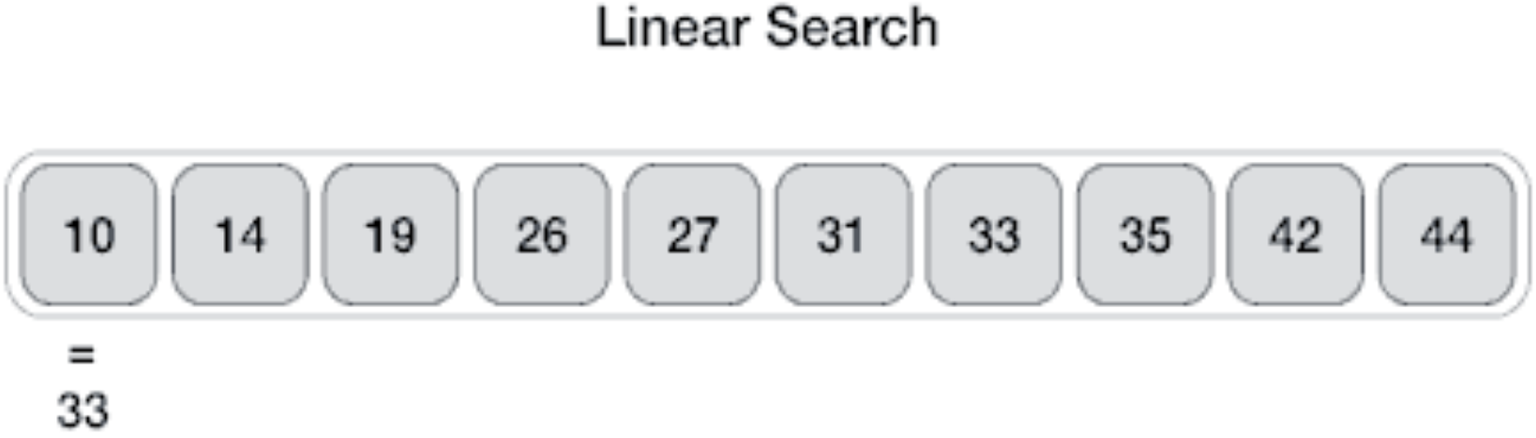
QUICK SORT

```python
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 LOGN)

ALGORITMOS DE BÚSQUEDA

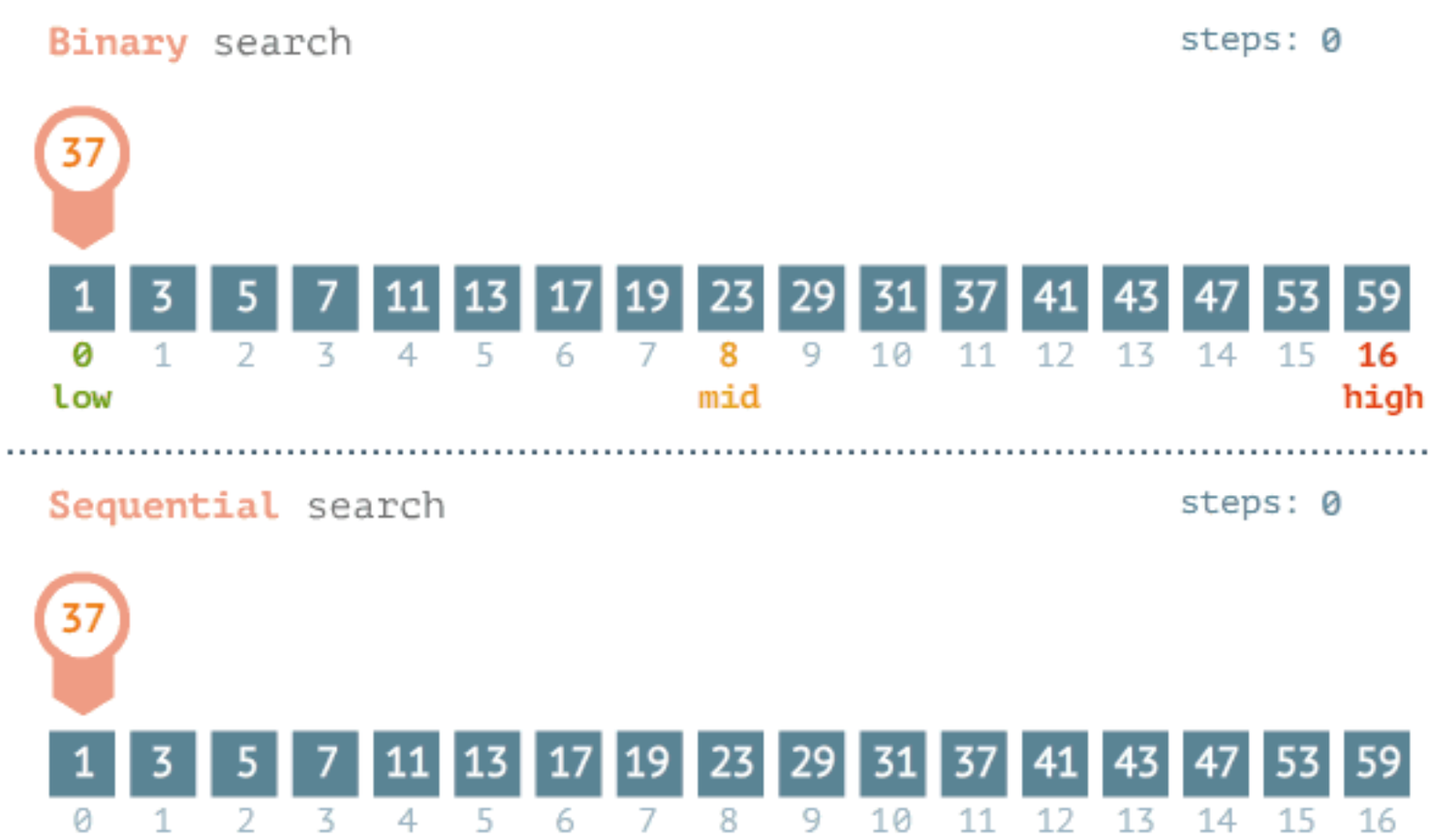
BÚSQUEDA LINEAL




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

O(N)

BÚSQUEDA BINARIA



www.penjee.com

ARCHIVOS

https://docs.python.org/3/tutorial/inputoutput.html

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

https://docs.python.org/2/library/json.html

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 ESTA UTILIZANDO UN ALGORITMO DE O(N LOGN).

- 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`

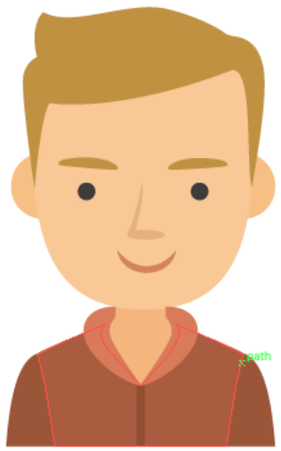
game ranking

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

MARIA
SOFIA
ANA
JULIETA

MARCO

MARIA
JULIETA
SOFIA
ANA

JUAN

MARIA
SOFIA
ANA
JULIETA

HUGO

MARIA
JULIETA
SOFIA
ANA

PEDRO

MARCO
JUAN
HUGO
PEDRO

MARIA

MARCO
JUAN
HUGO
PEDRO

SOFIA

JUAN
HUGO
PEDRO
MARCO

ANA

PEDRO
JUAN
HUGO
JUAN

JULIETA

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 MUNOZ' ->
# 'HOLLAND JENNINGS ENGLISH' ->
```

Pareja Feliz

ALGORITMOS DE CIFRADO Y LOS DINEROS

SSH

HTTPS

SCL


Google's
Project Zero

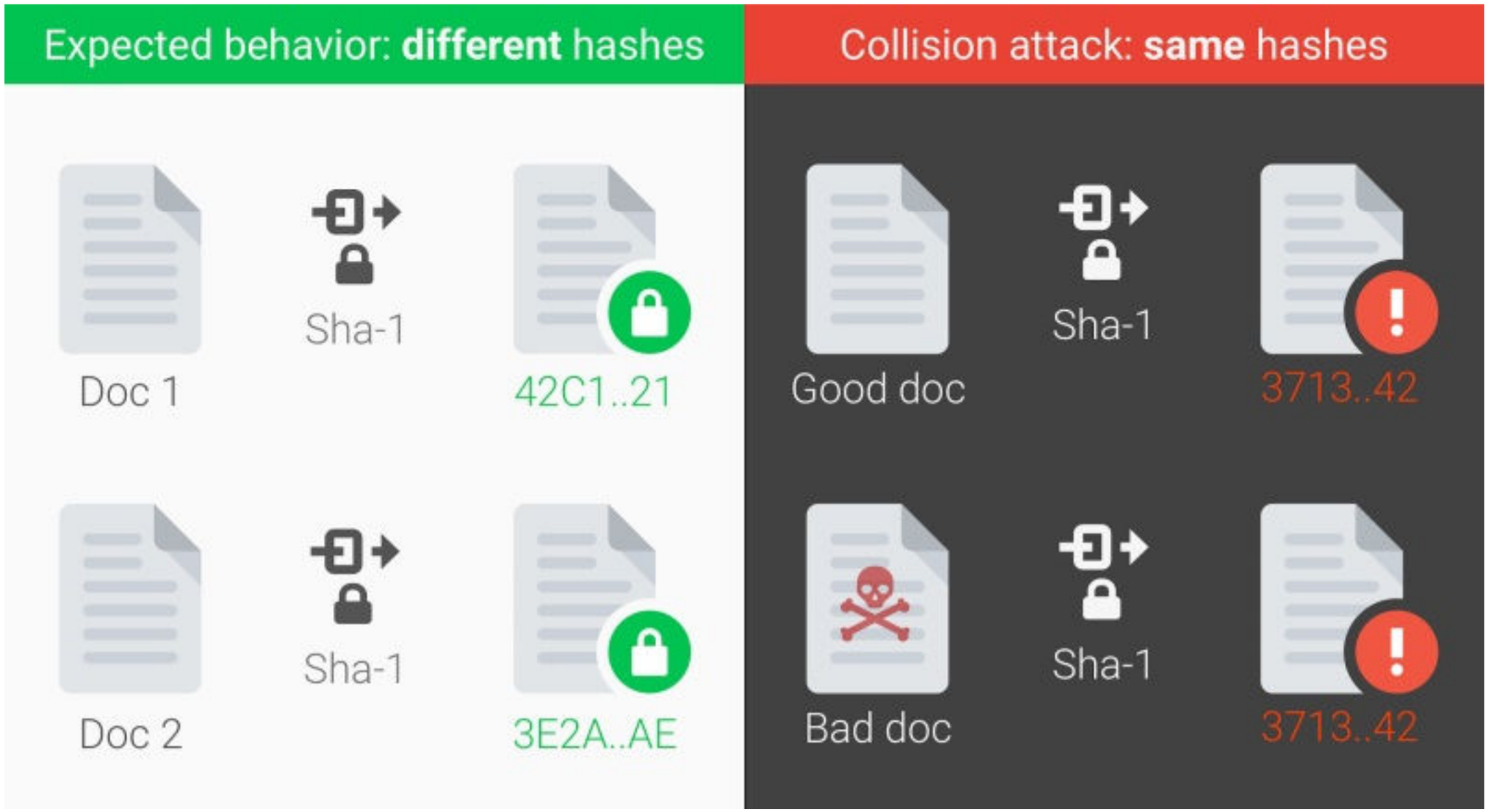
SHA-1
SECURE HASH ALGORITHM

HOLA MAMA → AE23JV232

ESTE ES OTRO UN MENSAJE → WF53D62D

HOLA MAMA. → GSGJ29D3

LA HUELLA DE UN ARCHIVO

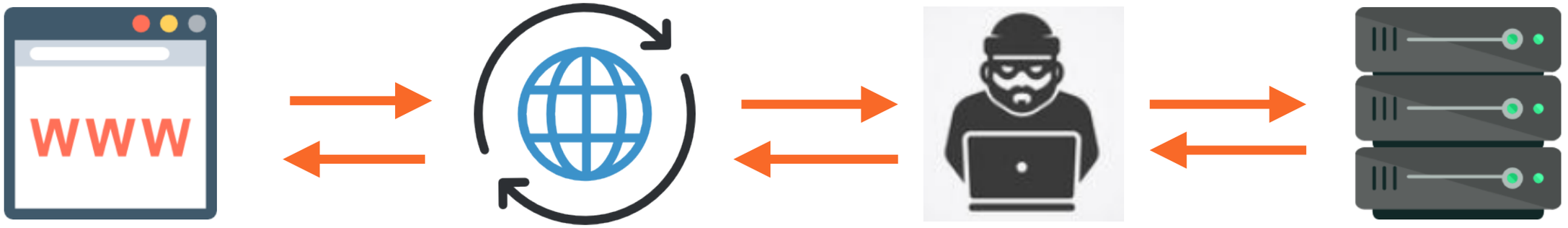

Expected behavior: **different** hashes
Collision attack: **same** hashes
Doc 1
Sha-1
42C1..21
Doc 2
Sha-1
3E2A..AE
Good doc
Sha-1
3713..42
Bad doc
Sha-1
3713..42

GUARDAR CONTRASEÑAS

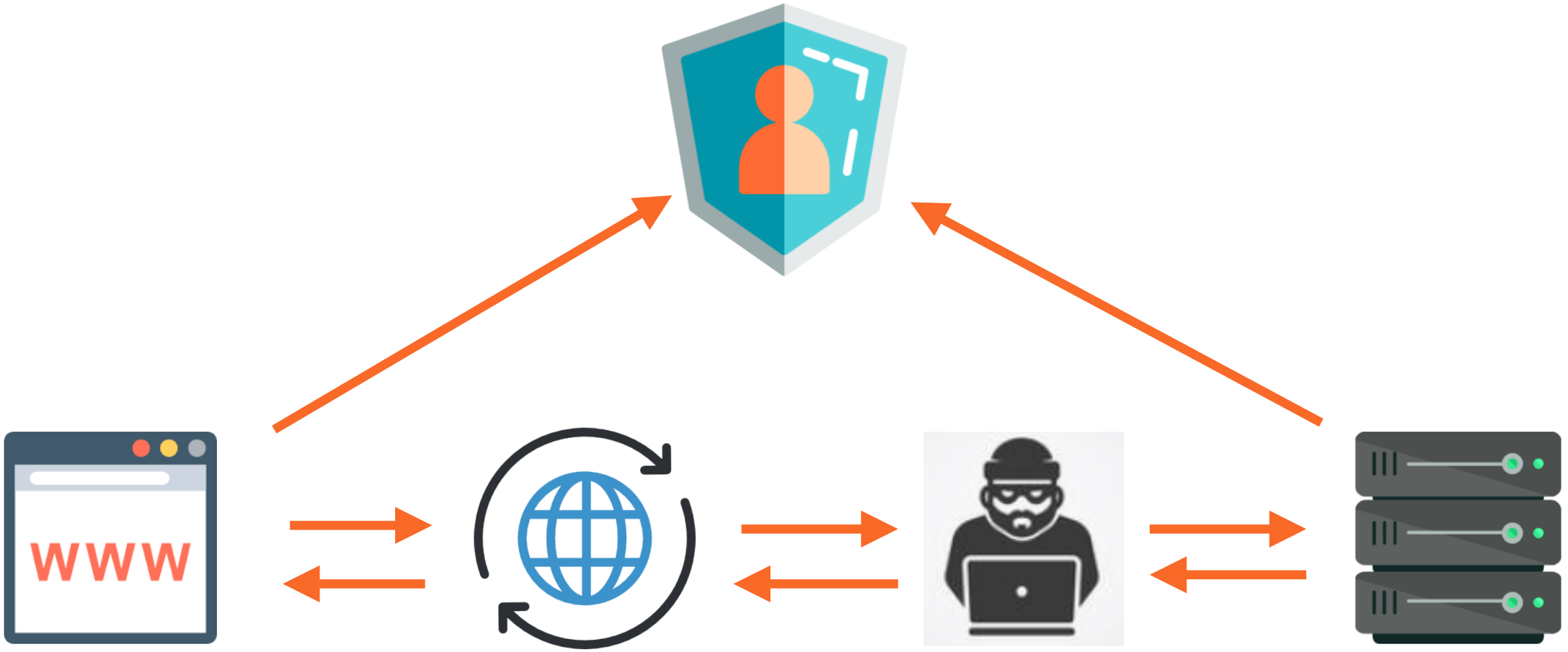
VERIFICAR ARCHIVOS

ALGORITMOS DE CIFRADO

ALGORITMOS DE CIFRADO

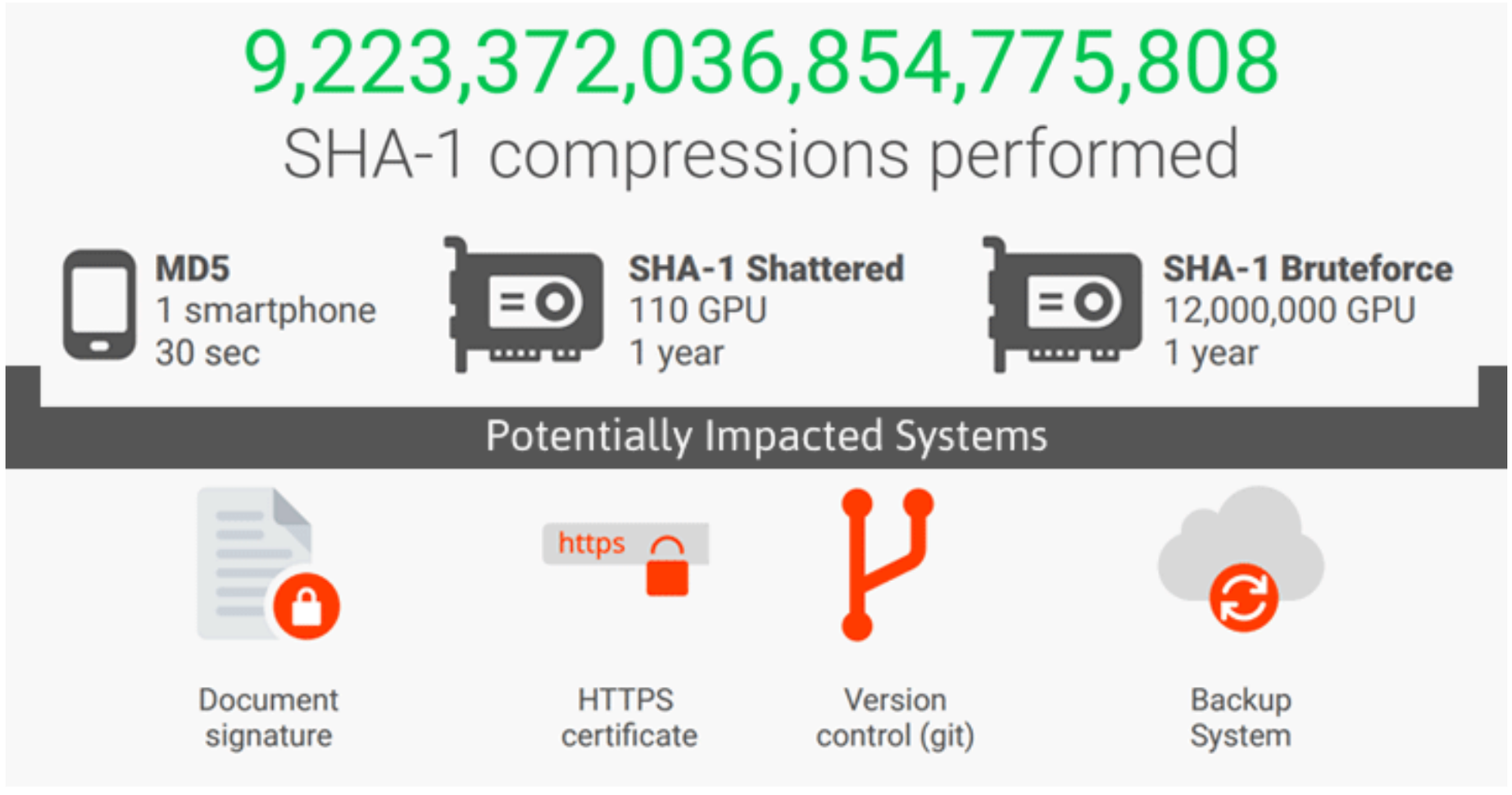


ALGORITMOS DE CIFRADO

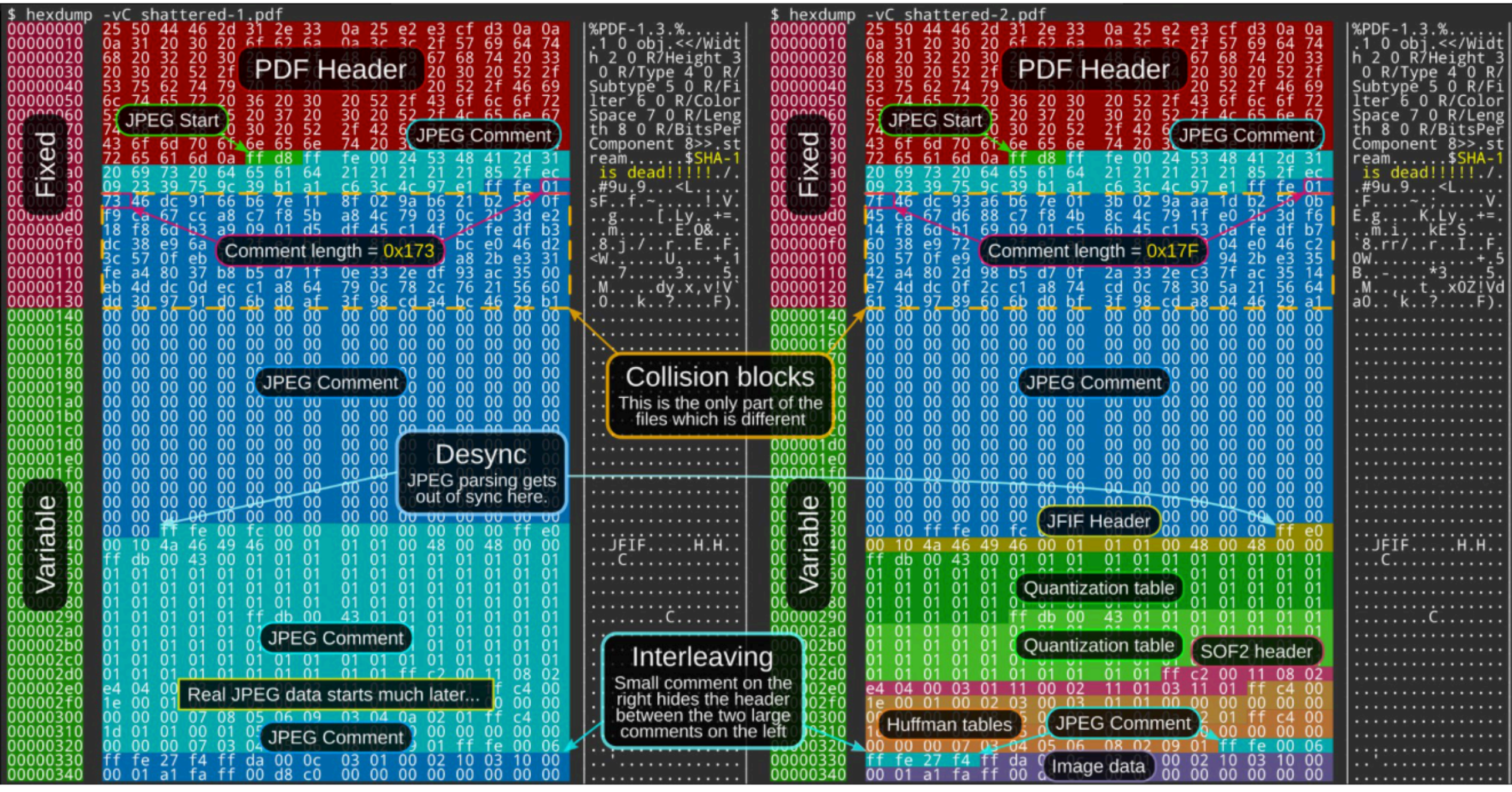


¿ENTONCES QUE ENCONTRO GOOGLE?

¿COMO LE HIZO?


9,223,372,036,854,775,808
SHA-1 compressions performed
MD5
1 smartphone
30 sec
SHA-1 Shattered
110 GPU
1 year
SHA-1 Bruteforce
12,000,000 GPU
1 year
Potentially Impacted Systems
https
Document signature
HTTPS certificate
Version control (git)
Backup System

HTTPS://ALF.NU/SHA1



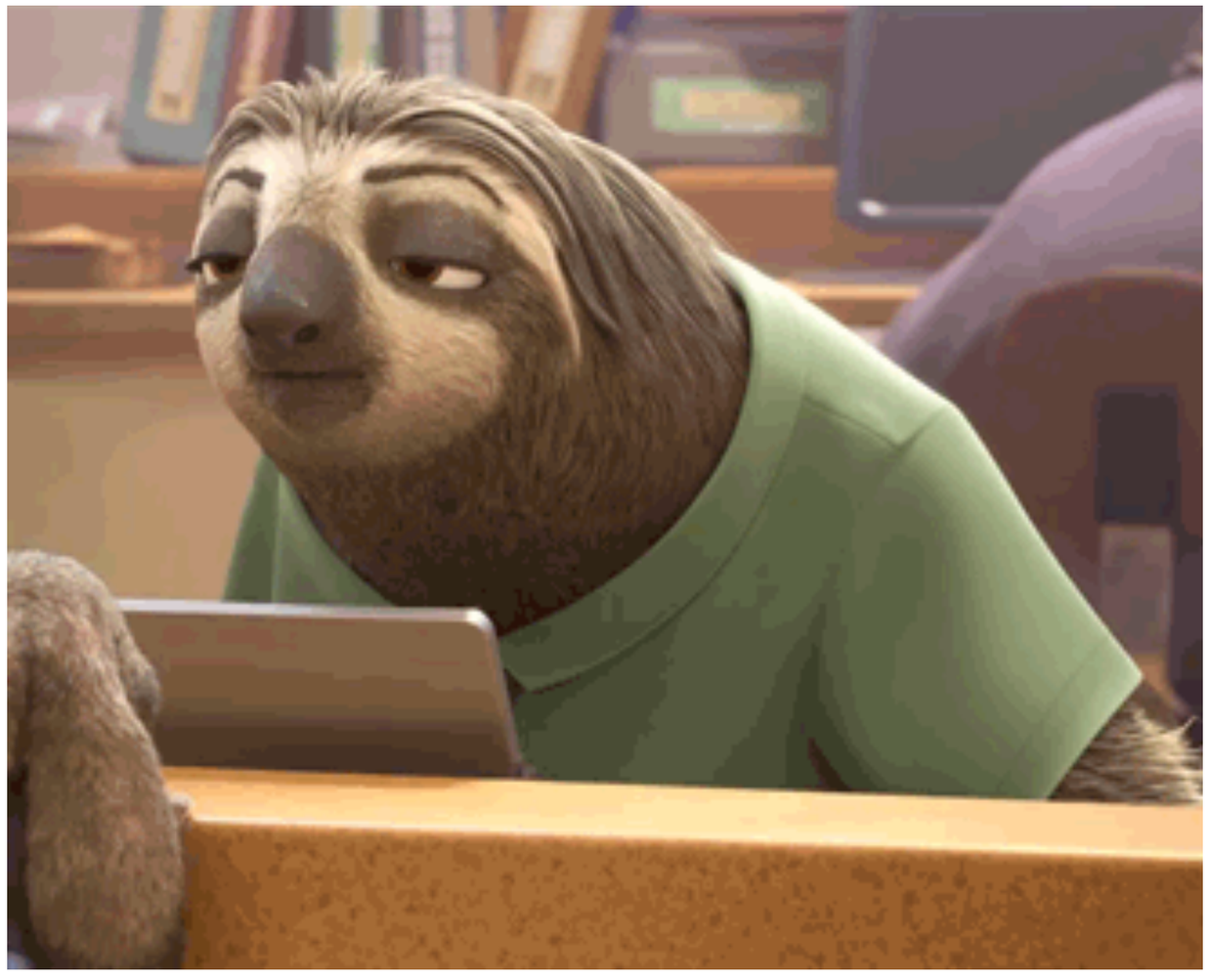
ALGORITMOS DE CIFRADO

Lifetimes of popular cryptographic hashes (the rainbow chart)																									
Function	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
Snefru																										
MD2 (128-bit)[1]																										
MD4																										
MD5															[2]											
RIPEMD															[2]											
HAVAL-128[1]															[2]											
SHA-0																										
SHA-1																										
RIPEMD-160																										
SHA-2 family																		[3]								
SHA-3 (Keccak)																										

Key	Didn't exist/not public	Under peer review	Considered strong	Minor weakness	Weakened	Broken	Collision found

¿POR QUE ES PREOCUPANTE?

ALGORITMOS DE CIFRADO

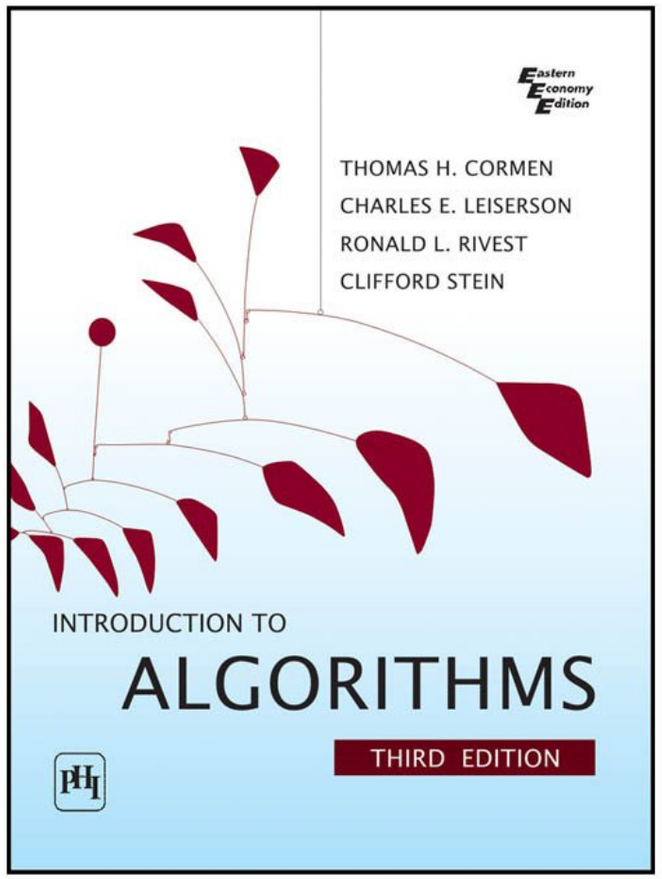


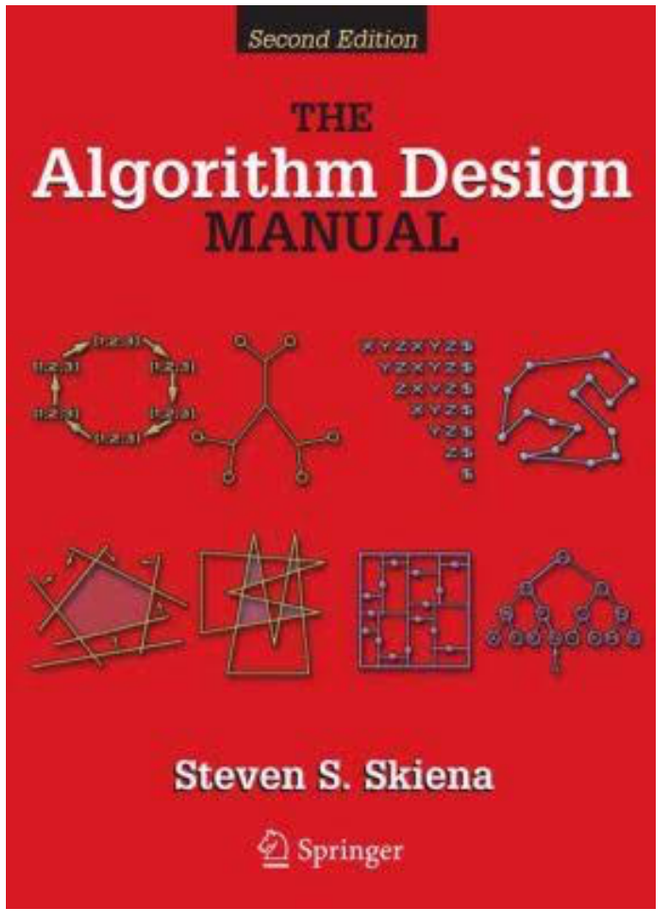
RSA-2048

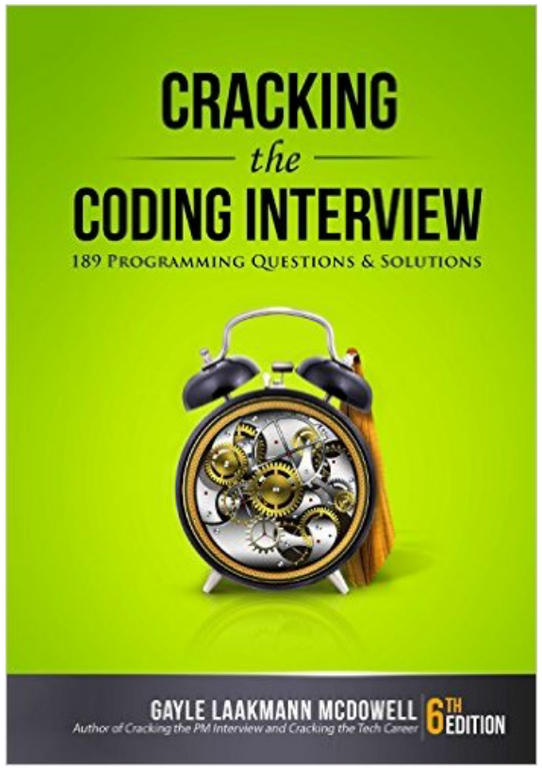
US$200,000

25195908475657893494027183240048398571429282126204032027777137836043662020707595556264018525880784406918290641249515082189298559149176184502808489120072844992687392807287776735971418347270261896375014971824691165077613379859095700097330459748808428401797429100642458691817195118746121515172654632282216869987549182422433637259085141865462043576798423387184774447920739934236584823824281198163815010674810451660377306056201619676256133844143603833904414952634432190114657544454178424020924616515723350778707749817125772467962926386356373289912154831438167899885040445364023527381951378636564391212010397122822120720357

CONCLUSIÓN

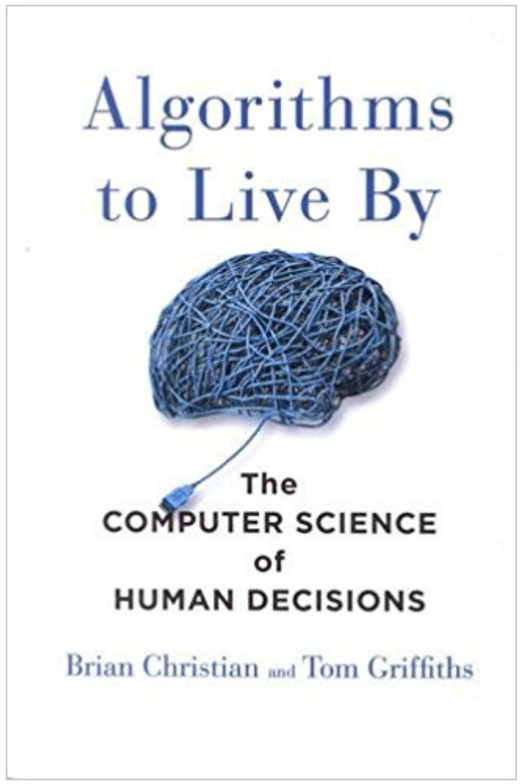






MIT 6.006 INTRODUCTION TO ALGORITHMS, FALL 2011

MIT 6.854 (ADVANCED ALGORITHMS), SPRING 2016



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