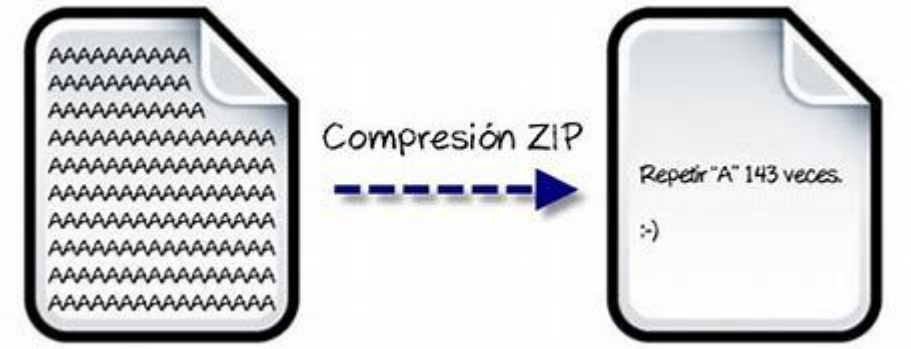
COMPRESIÓN
- Código de Huffman -

Maria Bogajo López

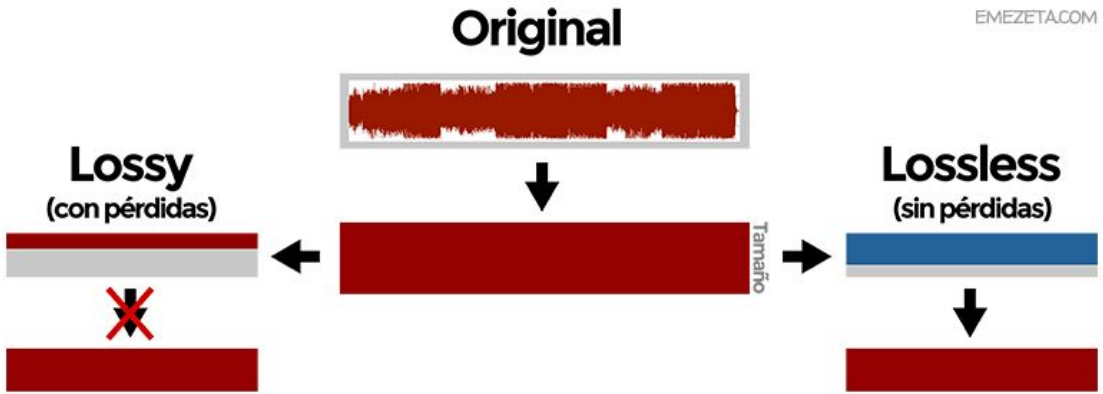
Compresión

Reducción de la cantidad de espacio necesario para representar un archivo.



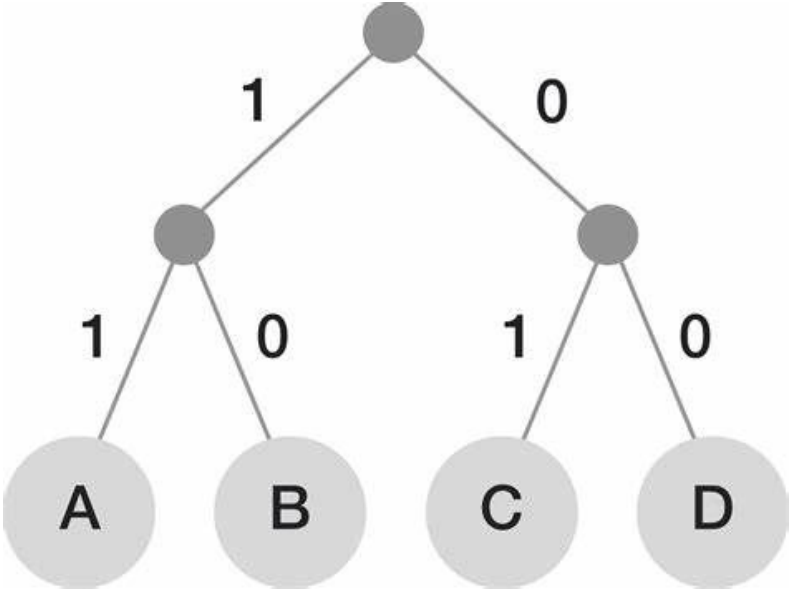
Tipos de compresión

- **Sin perdida (Lossless)**: Reducen el tamaño de los archivos sin perder ninguna información en ellos.
- **Con perdida (Lossy)**: Reducen el tamaño de los archivos al descartar información menos importante.



¿Qué es el código de Huffman?

El **Código de Huffman** se utiliza principalmente en **compresión de datos**, donde el objetivo es reducir el tamaño de los datos para almacenamiento o transmisión.



Resolución

Construcción del Árbol de Huffman

1. Ordenar los caracteres según su frecuencia.
2. Combinar los nodos con las dos frecuencias más bajas para crear un nuevo nodo (la suma de sus frecuencias).
3. Repetir hasta obtener un único árbol.

Asignación códigos binarios

A cada carácter según las reglas del árbol de Huffman se le asigna a la izquierda = 0, a la derecha = 1.

Coste total

$$\text{Costo total} = \sum(\text{longitud del código del carácter} \times \text{frecuencia del carácter})$$

Caso práctico

Supongamos que se tiene los siguientes caracteres y sus frecuencias asociadas en un texto:

CARÁCTER	FRECUENCIA
A	5
B	9
C	12
D	13
E	16
F	45

¿Y si lo hacemos con código?

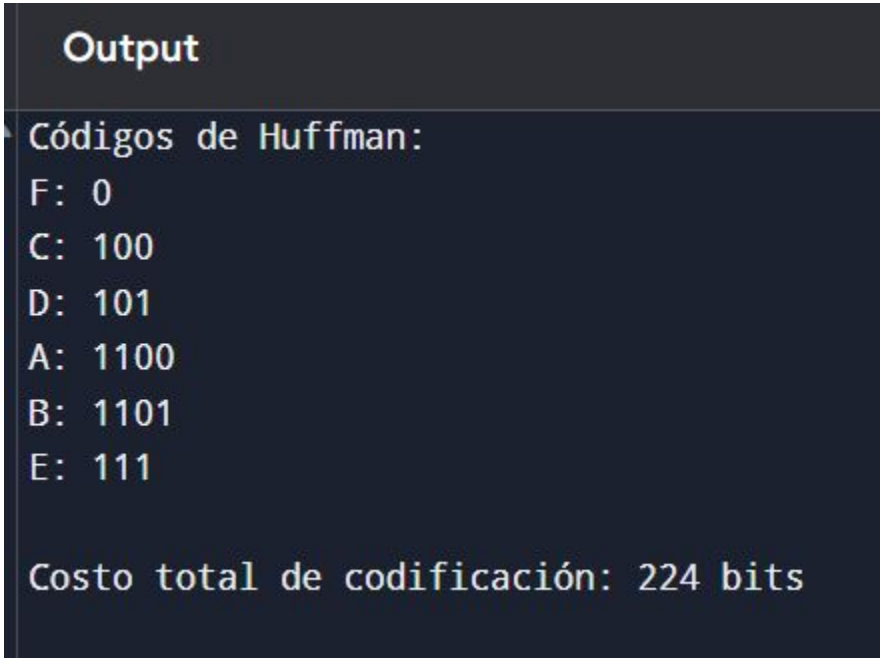
```python
import heapq

# Clase para representar los nodos del Árbol de Huffman
class HuffmanNode:
    def __init__(self, char, freq):
        self.char = char
        self.freq = freq
        self.left = None
        self.right = None

    def __lt__(self, other):
        return self.freq < other.freq

# Función para construir el Árbol de Huffman
def build_huffman_tree(frequencies):
    heap = [HuffmanNode(char, freq) for char, freq in frequencies]
    heapq.heapify(heap)

    while len(heap) > 1:
        left = heapq.heappop(heap)
        right = heapq.heappop(heap)

        merged = HuffmanNode(None, left.freq + right.freq)
        merged.left = left
        merged.right = right
        heapq.heappush(heap, merged)

    return heap[0]

# Función para generar los códigos de Huffman
def generate_huffman_codes(node, code="", codes={}):
    if node is not None:
        if node.char is not None:
            codes[node.char] = code
        generate_huffman_codes(node.left, code + "0", codes)
        generate_huffman_codes(node.right, code + "1", codes)
    return codes

# Función para calcular el costo total
def calculate_total_cost(codes, frequencies):
    cost = 0
    for char, freq in frequencies:
        cost += len(codes[char]) * freq
    return cost

# Frecuencias iniciales
frequencies = [
    ('A', 5),
    ('B', 9),
```

```python
# Frecuencias iniciales
frequencies = [
    ('A', 5),
    ('B', 9),
    ('C', 12),
    ('D', 13),
    ('E', 16),
    ('F', 45)
]

# Construir el Árbol de Huffman
huffman_tree = build_huffman_tree(frequencies)

# Generar los códigos de Huffman
huffman_codes = generate_huffman_codes(huffman_tree)

# Calcular el costo total
total_cost = calculate_total_cost(huffman_codes, frequencies)

# Mostrar los resultados
print("Códigos de Huffman:")
for char, code in huffman_codes.items():
    print(f"{char}: {code}")

print(f"\nCosto total de codificación: {total_cost} bits")

```

Cómo funciona el código

1. **Entrada de frecuencias**: La lista frequencies contiene los caracteres y sus frecuencias asociadas.
2. **Construcción del árbol**:
   - Se usa una cola de prioridad para combinar nodos basados en frecuencias.
   - Los nodos más pequeños se combinan para formar nodos internos del árbol.
3. **Generación de códigos**:
   - Recorre el árbol asignando 0 para ir a la izquierda y 1 para ir a la derecha.
   - Los códigos resultantes se almacenan en un diccionario.
4. **Cálculo del costo total**:
   - Multiplica la longitud del código de cada carácter por su frecuencia y suma estos valores.
5. **Salida**:
   - Muestra los códigos de Huffman para cada carácter y el costo total.

Resultado


Output
Códigos de Huffman:
F: 0
C: 100
D: 101
A: 1100
B: 1101
E: 111
Costo total de codificación: 224 bits

Usos

Diferentes métodos de compresión:

- Texto plano
- Deflación y códec multimedia como JPEG y MP3 (Cuantificación digital basada en codificación de Huffman)
- ...