Algoritmo y Árboles de Huffman

Por: Juan Daniel Torres - 2240082 & Sebastián Nossa

INTRODUCCIÓN

El algoritmo de Huffman es un método de compresión de datos sin pérdidas (es decir, un procedimiento algorítmico que reduce el tamaño de un conjunto de datos sin eliminar ni alterar ningún bit de información, de manera que al descomprimir se obtiene exactamente la secuencia original de bits) desarrollado por David A. Huffman en 1952. Su objetivo es generar códigos binarios de longitud variable, asignando códigos más cortos a los símbolos más frecuentes y códigos más largos a los menos frecuentes, esto permite reducir el tamaño total de un mensaje.

Este algoritmo garantiza un código de prefijo óptimo, es decir, ningún código es prefijo de otro, lo que permite una decodificación única y eficiente.

DESCRIPCIÓN

El proceso se basa en construir un árbol binario completo donde:
→ Cada hoja representa un símbolo del alfabeto.
→ La ruta desde la raíz hasta una hoja define el código del símbolo, teniendo en cuenta que se aplica:

- 0 al tomar una rama izquierda.
- 1 al tomar una rama derecha.

Pasos del algoritmo:

- Crear un nodo hoja para cada símbolo, etiquetado con su frecuencia $\left(\frac{Apariciones}{SimbolosTotales}\right)$.
- Insertar todos los nodos en una cola de prioridad (ordenada por frecuencia).
- Mientras haya más de un nodo en la cola:
  - Extraer los dos nodos con menor frecuencia.
  - Crear un nuevo nodo padre con frecuencia igual a la suma de ambos.
  - Asignar los nodos extraídos como hijos izquierdos (0) o como hijos derechos (1).
  - Insertar el nuevo nodo en la cola.
- El único nodo restante es la raíz del árbol de Huffman.

CODIFICACIÓN Y DECODIFICACIÓN

Para codificar un símbolo:
1. Iniciar desde la hoja del símbolo.
2. Subir por el árbol hasta la raíz, registrando los bits correspondientes (0 para la izquierda, 1 para la derecha).
3. Invertir el orden de los bits obtenidos.

Para decodificar un código:
1. Iniciar desde la raíz del árbol.
2. Leer el código bit a bit:
   - “0” → ir a la izquierda.
   - “1” → ir a la derecha.
3. Al llegar a una hoja, se obtiene el símbolo correspondiente.

NOTA: En la práctica, se recorren todos los símbolos una vez, se guarda el símbolo en una tabla, y luego el árbol puede descartarse.

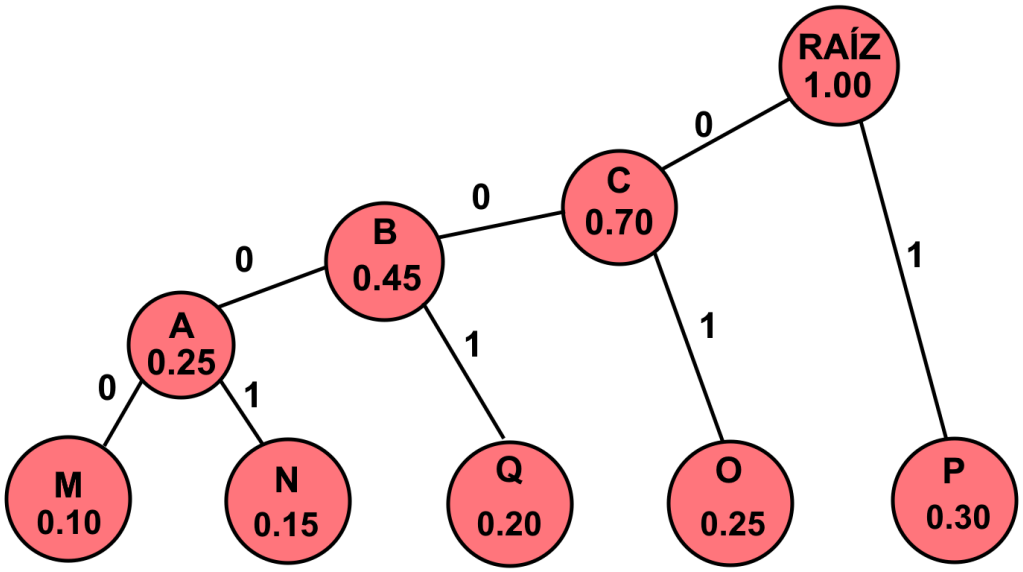
EJEMPLO

Consideremos los siguientes símbolos y sus frecuencias:

Símbolo	Frecuencia
M	0.10
N	0.15
O	0.25
P	0.30
Q	0.20

Construcción del árbol:
1. Unir M (0.10) y N (0.15) → nodo A (0.25)
2. Unir Q (0.25) y A (0.20) → nodo B (0.45)
3. Unir O (0.25) y B (0.45) → nodo C (0.70)
4. Unir C (0.70) y P (0.30) → raíz (1.00)

Imagen del árbol:



Códigos resultantes:
→ M = 0000
→ N = 0001
→ Q = 001
→ O = 01
→ P = 1

COMPLEJIDAD Y EFICIENCIA

- **Tiempo de ejecución:** El uso de una cola de prioridad hace que el algoritmo tenga una complejidad de O(n log n), donde n es el número de símbolos.
- **Optimalidad:** Huffman demostró que el algoritmo produce un código de longitud media mínima entre todos los códigos de prefijo posibles.
- **Espacio:** Aunque el árbol de Huffman puede descartarse después de generar los códigos, es necesario almacenar la tabla de códigos resultante para la codificación y decodificación. Esta tabla puede representarse de forma eficiente utilizando estructuras tipo diccionario (mapas clave-valor).

LIMITACIONES

- Requiere conocer de antemano las frecuencias de aparición de cada símbolo, si no son exactas, el código podria no ser el más eficiente.
- Si todos los símbolos tienen frecuencia igual, se obtiene un código binario uniforme sin compresión.
- En alfabetos pequeños (como dos símbolos), no se obtiene compresión ya que no hay redundancia que se pueda aprovechar.
- El algoritmo no es óptimo si los símbolos tienen dependencias contextuales puesto que se asume que los símbolos son independientes.

Soluciones parciales: agrupar símbolos en bloques para capturar patrones o utilizar algoritmos adaptativos que ajusten las frecuencias durante la lectura de datos en tiempo real.

VARIANTES DEL ALGORITMO

Huffman N-ario

- Se pueden construir árboles ternarios, cuaternarios, o en general, de n hijos.
- Requiere que el número de símbolos (s) menos uno (s-1) sea múltiplo de (n − 1) donde n es el orden del arbol.
- Si no se cumple la condición anterior, se pueden agregar símbolos ficticios con frecuencia 0 para completar la estructura del árbol.

APLICACIONES

Los árboles de Huffman se utilizan ampliamente en tecnologías de compresión de datos sin pérdida y como parte de procesos en compresión con pérdida:
→ **ZIP**, **GZIP**: dentro del algoritmo **DEFLATE**.

→ **PNG:** compresión de imágenes sin pérdida (usa **DEFLATE**).
→ **JPEG:** En la codificación final del flujo comprimido (Huffman es sin pérdida, pero el JPEG global **sí tiene pérdida**).
→ **MP3 / MPEG:** compresión de audio y vídeo (Huffman se aplica en etapas específicas dentro de un esquema con pérdida.).
→ **PDF:** Para compresión de texto y datos.

CONCLUSIÓN

El algoritmo de Huffman es un método óptimo y eficiente para la compresión sin pérdida de datos, a su vez, es usado ampliamente en etapas específicas dentro de esquemas de compresión con pérdida (como JPEG o MP3) y, debido a esto, demuestra que es un algoritmo versátil e ingenioso que facilita la compresión de diferentes datos como lo pueden ser archivos binarios, multimedia y texto.

BIBLIOGRAFÍA

de Vigo, U. (s/f). *Algoritmo de Huffman*. Uvigo.es. Recuperado el 20 de abril de 2025, de https://joselu.webs.uvigo.es/material/Algoritmo%20de%20Huffman.pdf

Purdue. (s/f). *ECE264: Codificación Huffman*. Purdue.edu. Recuperado el 20 de abril de 2025, de https://engineering.purdue.edu/ece264/17au/hw/HW13?alt=huffman