INSTITUTO POLITECNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

ANÁLISIS Y DISEÑO DE ALGORITMOS

PRÁCTICA 06

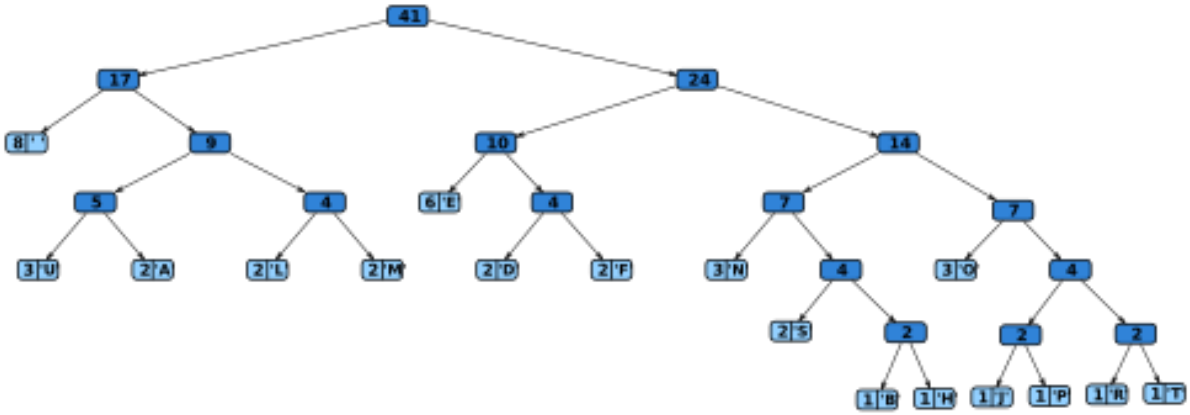
Rodrigo García Mayorga

2012630554

Introducción

El **algoritmo de Huffman** es un método de **compresión de datos sin pérdida** que asigna códigos binarios más cortos a los símbolos más frecuentes y códigos más largos a los menos frecuentes, optimizando así el espacio de almacenamiento.

Pertenece a la categoría de **algoritmos codificadores de longitud variable** y se basa en la técnica de **greedy (voraz)**. Esto significa que toma decisiones óptimas en cada paso local con la esperanza de encontrar una solución global óptima. Para construir el árbol de Huffman, el algoritmo selecciona de forma iterativa los dos nodos de menor frecuencia y los combina, repitiendo el proceso hasta formar un solo árbol que define los códigos óptimos para cada símbolo.



La técnica utilizada es el propio algoritmo de Huffman. Consiste en la creación de un árbol binario en el que se etiquetan los nodos hoja con los caracteres, junto a sus frecuencias, y de forma consecutiva se van uniendo cada pareja de nodos que menos frecuencia sumen, pasando a crear un nuevo nodo intermedio etiquetado con dicha suma. Se procede a realizar esta acción hasta que no quedan nodos hoja por unir a ningún nodo superior, y se ha formado el árbol binario.

Posteriormente se etiquetan las aristas que unen cada uno de los nodos con ceros y unos (hijo derecho e izquierdo, respectivamente, por ejemplo). El código resultante para cada carácter es la lectura, siguiendo la rama, desde la raíz hacia cada carácter (o viceversa) de cada una de las etiquetas de las aristas.

Desarrollo

Implementación del Algoritmo de Huffman

Descripción General

El programa implementa el algoritmo de compresión de Huffman para comprimir archivos de texto. Este algoritmo utiliza una codificación de longitud variable basada en la frecuencia de aparición de los caracteres.

Estructuras de Datos Principales

1. Nodo del Árbol Huffman

```
struct MinHeapNode {
    char data;           // Carácter
    unsigned freq;       // Frecuencia
    struct MinHeapNode *left, *right;  // Hijos izquierdo y derecho
};
```

2. Montículo Mínimo (MinHeap)

```
struct MinHeap {
    unsigned size;       // Tamaño actual
    unsigned capacity;   // Capacidad máxima
    struct MinHeapNode** array;  // Array de punteros a nodos
};
```

Proceso de Compresión

1. Análisis de Frecuencias

- Lee el archivo de entrada carácter por carácter
- Mantiene un contador de frecuencias para cada carácter
- Utiliza un arreglo dinámico para almacenar solo los caracteres que aparecen

2. Construcción del Árbol Huffman

- Crea un nodo hoja para cada carácter
- Utiliza un montículo mínimo para construir el árbol
- Une los dos nodos con menor frecuencia iterativamente

3. Generación de Códigos

- Recorre el árbol desde la raíz hasta las hojas
- Asigna '0' para ramas izquierdas y '1' para ramas derechas

- Almacena los códigos en una estructura `HuffmanCode`

Resultados

Códigos Generados

Como se puede ver en el archivo codificado.txt, los códigos varían en longitud:

- Caracteres frecuentes: códigos cortos (ej: 'e': 111)

- Caracteres raros: códigos largos (ej: 'h': 0010100)

Estadísticas de Compresión

- **Archivo Original**: 1768 bits (221 caracteres × 8 bits)

- **Archivo Comprimido**: 874 bits

- **Tasa de Compresión**: 50.57%

Proceso de Decodificación

1. Carga el texto codificado

2. Recorre el árbol Huffman bit a bit:

  - '0': movimiento a la izquierda

  - '1': movimiento a la derecha

3. Al llegar a una hoja, obtiene el carácter original

Archivos Generados

1. **codificado.txt**

  - Diccionario de códigos Huffman

  - Texto codificado en formato binario

2. **decodificado.txt**

  - Texto original recuperado tras la decodificación

Eficiencia

- **Tiempo**: O(n log n) para la construcción del árbol

- **Espacio**: O(n) para almacenar el árbol y los códigos

- **Compresión**: ~50% del tamaño original en este caso

Bibliografía

- **David A. Huffman**, *"A Method for the Construction of Minimum-Redundancy Codes"*, Proceedings of the IRE, 1952.
    - Artículo original donde Huffman propuso su algoritmo.
- **Khalid Sayood**, *"Introduction to Data Compression"*, 5th Edition, Morgan Kaufmann, 2017.
    - Un texto muy completo sobre técnicas de compresión, incluyendo Huffman, LZW y más.

Enlace del código (github):

https://github.com/LowisN/ADA_Practica-7