Instituto Politécnico Nacional

Escuela Superior de Cómputo

Análisis de algoritmos

Practica 03: Codificación voraz de Huffman

M. en C. Edgardo Adrián Franco Martínez
http://www.eafranco.com
edfrancom@ipn.mx
@edfrancom edgardoadrianfrancom

Contenido

- Definición del problema
- Algoritmo de Huffman
  - Pasos del código de Huffman
- Actividades
- Observaciones
- Reporte de práctica
- Rubrica de evaluación del reporte
- Entrega vía Web
- Fechas de Entrega

Definición del problema

- Implementar el algoritmo de codificación de Huffman para codificar archivos de cualquier tipo bajo lenguaje C.
  - Implementar codificación voraz de Huffman
  - Implementar el algoritmo de decodificación

- Medir y comprobar las ventajas de tamaño de los archivos una vez realizadas diferentes codificaciones de archivos.

- Medir los tiempos de ejecución de las implementaciones (codificador y decodificador).

Algoritmo de Huffman

- El algoritmo de Huffman es un algoritmo para la construcción de códigos de Huffman, desarrollado por David A. Huffman en 1952 y descrito en A Method for the Construction of Minimum-Redundancy Codes.

- Este algoritmo toma un alfabeto de n símbolos, junto con sus frecuencias de aparición asociadas, y produce un código de Huffman para ese alfabeto y esas frecuencias.

- El algoritmo de Huffman es un algoritmo para la construcción de códigos de Huffman, desarrollado por David A. Huffman en 1952 y descrito en A Method for the Construction of Minimum-Redundancy Codes.

- Este algoritmo toma un alfabeto de n símbolos, junto con sus frecuencias de aparición asociadas, y produce un código de Huffman para ese alfabeto y esas frecuencias.

- Es un algoritmo usado para compresión de datos. El termino se requiere al uso de una tabla de códigos de longitud variable para codificar un determinado símbolo (como puede ser un carácter en un archivo), donde la tabla ha sido rellenada de una manera especifica basándose en la probabilidad estimada de aparición de cada posible valor de dicho símbolo.

- Huffman propuso un **algoritmo voraz** que obtiene una **codificación prefijo óptima**.

- Para ello construye un árbol binario de códigos de longitud variable de manera ascendente.

- El algoritmo funciona de la siguiente manera:
  - Parte de una secuencia inicial en la que los caracteres a codificar están colocados en orden creciente de frecuencia.
    - Esta secuencia inicial se va transformando, a base de fusiones, hasta llegar a una secuencia con un único elemento que es el árbol de codificación óptimo.

Pasos del código de Huffman

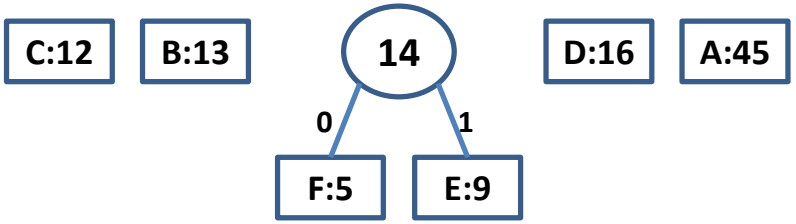
1. Se crean varios árboles, uno por cada uno de los símbolos del alfabeto, consistiendo cada uno de los árboles en un nodo sin hijos, y etiquetado cada uno con su símbolo asociado y su frecuencia de aparición.

2. Se toman los dos árboles de menor frecuencia, y se unen creando un nuevo árbol. La etiqueta de la raíz será la suma de las frecuencias de las raíces de los dos árboles que se unen, y cada uno de estos árboles será un hijo del nuevo árbol. También se etiquetan las dos ramas del nuevo árbol: con un 0 la de la izquierda, y con un 1 la de la derecha.

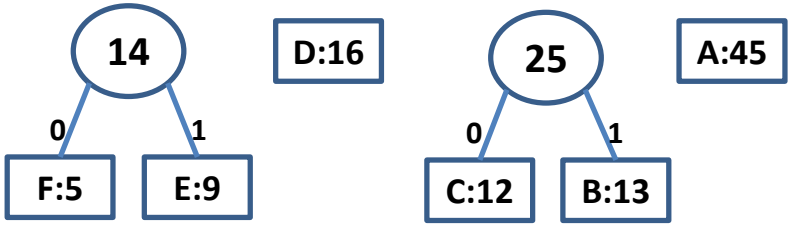
3. Se repite el paso 2 hasta que sólo quede un árbol.

ETAPA 1

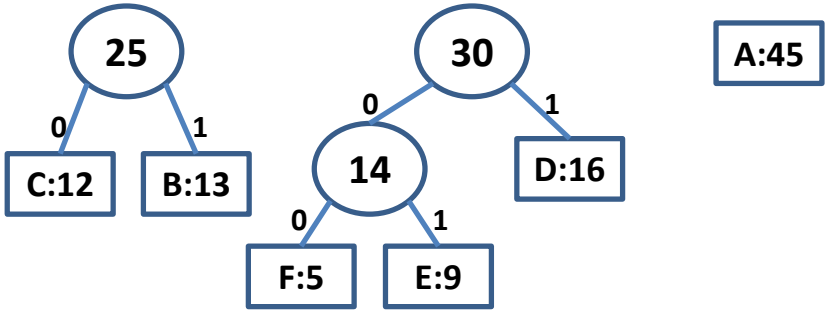
ETAPA 2

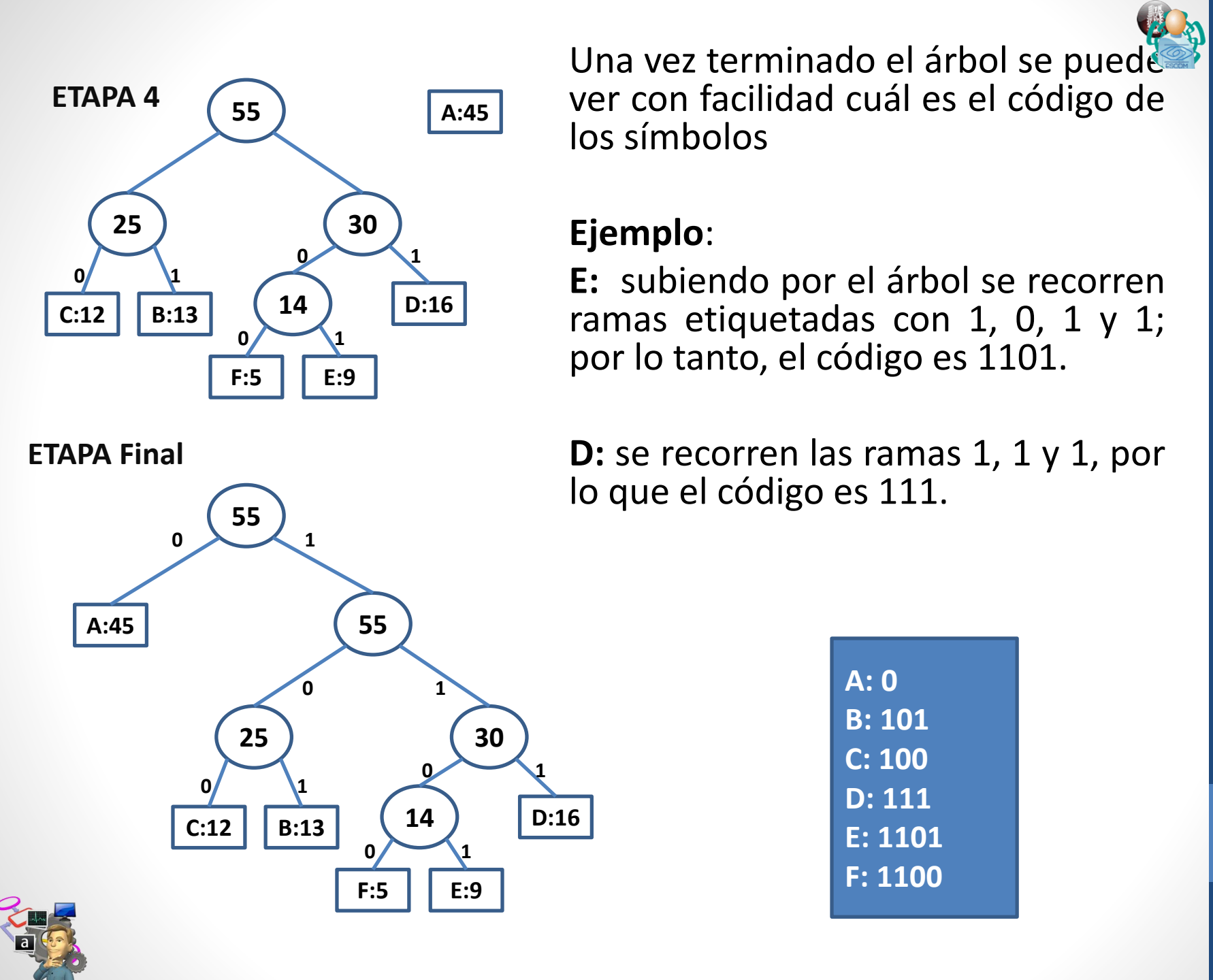


ETAPA 3



ETAPA 4





Una vez terminado el árbol se puede ver con facilidad cuál es el código de los símbolos

Ejemplo:

E: subiendo por el árbol se recorren ramas etiquetadas con 1, 0, 1 y 1; por lo tanto, el código es 1101.

D: se recorren las ramas 1, 1 y 1, por lo que el código es 111.

A: 0
B: 101
C: 100
D: 111
E: 1101
F: 1100

Decodificación del código de Huffman

1. A partir del árbol de codificación, comenzar a recorrer los caminos según los bits de la codificación. Al llegar a un nodo hoja se toma el valor de esta y coloca en el archivo original.

2. Se repite el paso 1 a partir del bit siguiente de la codificación comenzando un nuevo recorrido a partir de la raíz del árbol de la codificación.

3. La decodificación termina una vez se hallan recorrido todos los bits de la codificación.

Actividades

1. Entender el algoritmo de codificación voraz de Huffman e implementarlo en lenguaje C para **codificar cualquier archivo**.

2. Documentar y explicar el algoritmo y la implementación de este en C.

3. Construir el programa que sea capaz de reconstruir el archivo codificado a su codificación original.

4. Documentar la implementación y realizar estadísticas de compresión para distintos archivos.

5. Encontrar una aproximación de la función de compresión (grafica de % de comprensión alcanzados con distintos tamaños de archivos).

6. Para cada implementación realizada medir los tiempos de ejecución (tiempo de codificación y decodificación).

7. Encontrar una aproximación de la función temporal para distintos tamaños de problema.

8. Finalmente responda a las siguientes preguntas:

i. ¿Los niveles de codificación de archivos proporcionan una ventaja respecto al tamaño del archivo original en el promedio de los casos?

ii. ¿Los tiempos de codificación o decodificación del archivo son muy grandes?

iii. ¿Ocurrieron perdidas de la información al codificar los archivos?

iv. ¿El comportamiento experimental del algoritmos era el esperado? ¿Por que?

v. ¿Qué características deberá tener una imagen BMP para codificarse en menor espacio?

vi. ¿Qué características deberá de tener un archivo de texto para tener una codificación en menor espacio?

vii. De 3 aplicaciones posibles en problemas de la vida real a la codificación de Huffman.

viii. ¿Existió un entorno controlado para realizar las pruebas experimentales? ¿Cuál fue?

ix. ¿Qué recomendaciones darían a nuevos equipos para realizar esta practica?

Observaciones

- Es necesario probar con una gran cantidad de archivos para poder concluir mejor aspectos como % de compresión esperados, factores que modifican la codificación, ventajas y desventajas.

- Indique cual fue su plataforma experimental (Características del hardware, compilador, sistema operativo, entorno controlado, etc.)

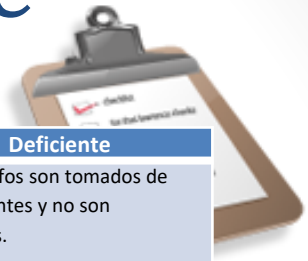
- Se sugiere crear scripts que faciliten la experimentación.

- En el laboratorio mostrar el funcionamiento de las tres implementaciones.
  - Autodocumentación del código
  - Documentación de funciones y algoritmos
  - Archivos de prueba suficientes y variados clasificados por tipo y características.

Reporte de practica

- Portada
- Introducción (Teoría sobre Programación voraz y Huffman)
- Planteamiento del problema
- Algoritmos (Codificación y decodificación) Descripción de la abstracción del problema y el algoritmo que da solución, apoyándose de pseudocódigo, diagramas y figuras en un lenguaje claro.
- Implementación de los algoritmos (Según los algoritmos utilizados como se implementaron en el código)
- Actividades y Pruebas (Actividades, verificación de la solución, pruebas y resultados de la práctica según lo solicitado)
- Errores detectados (Si existe algún error detectado, el cuál no fue posible resolver o se desconoce el motivo y solo ocurre con ciertas condiciones es necesario describirlo)
- Posibles mejoras (Describir posibles disminuciones de código en la implementación o otras posibles soluciones)
- Conclusiones (Por cada integrante del equipo)
- Anexo (Códigos fuente *con colores e instrucciones de compilación)
- Bibliografía (En formato IEEE)

Rubrica de evaluación del reporte



Indicador	Excelente	Muy bien	Bien	Deficiente
Construcción de párrafos	Todos los párrafos incluyen una introducción, explicaciones o detalles y una conclusión	Los párrafos incluyen información relacionada pero no fueron generalmente bien organizados	La estructura del párrafo no estaba clara y las oraciones no estaban generalmente relacionadas	Los párrafos son tomados de otras fuentes y no son originales.
Redacción	No hay errores de gramática, ortografía y puntuación y la redacción es coherentemente	No hay errores de gramática, ortografía y puntuación, pero la redacción presenta incoherencias	Pocos errores de gramática, ortografía y puntuación	Muchos errores de gramática, ortografía y puntuación
Cantidad de información Portada, Introducción, Planteamiento del problema, algoritmos e implementación, actividades y pruebas, errores detectados, posibles mejoras, conclusiones y anexos	Todos los temas son tratados de manera clara y precisa, según lo solicitado.	La mayoría de los temas son tratados de manera clara y precisa	Dos temas no están tratados o están imprecisos y no cumplen lo solicitado.	Tres o más temas no están tratados o están imprecisos y no cumplen lo solicitado.
Calidad de la información	La información está claramente relacionada con el tema principal y proporciona varias ideas secundarias y/o ejemplos	La información da respuestas a las preguntas principales, y solo da algunos detalles y/o ejemplos	La información da respuestas a las preguntas principales, pero no da detalles y/o ejemplos	La información tiene poco o nada que ver con las preguntas planteadas.
Algoritmos	Los algoritmos dan solución apoyándose de pseudocódigo, diagramas y/o figuras en un lenguaje claro.	La mayoría de los algoritmos dan solución apoyándose de pseudocódigo, pero diagramas y/o figuras.	Los algoritmos son mencionados textualmente pero no se describen	Los algoritmos no son expresados en el reporte.
Organización	La información está muy bien organizada con párrafos bien redactados y con subtítulos con estilos adecuados	La información está organizada, pero no se distingue en estilos adecuados	La información está organizada, pero los párrafos no están bien redactados	La información proporcionada no parece estar organizada o es copiada de referencias externas de manera literal

Entrega vía Web

Grupo	Contraseña
3CM1	analisis3cm1
3CM3	analisis3cm3

- **En un solo archivo comprimido** (ZIP, RAR, TAR, JAR o GZIP)
  - Reporte (DOC, DOCX o PDF)
  - Códigos fuente (.C, .H, etc.)
    - **Código documentado:** Titulo, descripción, fecha, versión, autor.
    - *(Funciones y Algoritmos: ¿Qué hace?, ¿Cómo lo hace?, ¿Qué recibe?, ¿Qué devuelve?, ¿Causa de errores?).*
    - OBSERVACIONES
    - *NO enviar ejecutables o archivos innecesarios, las instrucciones de compilación van en el anexo del reporte. (Yo compilare los fuente)

Fechas de entrega

- **Demostración** *Laboratorio de Programación 2 (2107)*
  - 3CM3 “Lunes 11 o lunes 18 de junio de 2018”.
  - 3CM1 “Miércoles 13 o miércoles 20 de junio de 2018”.

Entrega de reporte y código

- En un solo archivo comprimido.

Fecha y hora limite de entrega vía Web

- Martes 19 de Junio de 2018 a las 23:59:59 hrs.