Compresión y Compactación de archivos

Unidad 5.

Almacenamiento y Estructura de Archivos

Compresión de datos

- Existen diversas razones para hacer más pequeños los archivos:
  - Uso de menor espacio para almacenamiento
  - Mayor velocidad de transmisión, decremento del acceso a tiempo al momento de realizar una comunicación
  - Un procesamiento secuencial más rápido

Técnicas de compresión

- Comprimir datos significa codificar la información en un archivo de tal manera que ocupe menos espacio
- Algunas de las técnicas de compresión más utilizadas son:

  - Uso de notaciones diferentes
  - Supresión de secuencias repetidas
  - Asignación de longitudes de código variables
  - Técnicas de compresión irreversibles

Uso de notaciones diferentes

- Los archivos de longitud fija son buenos candidatos para esta técnica
- Ejemplo:
  - Codificación de los estados del país
  - Si se utilizan 2 caracteres ASCII (16 bytes)
  - ¿Cuántos bits son realmente necesarios?

Solución

- Se tienen 32 estados
- Se pueden codificar solamente con 6 bits
- Con esto solo se requiere 1 byte
- Se ahorra un 50% de espacio

Desventajas

- Utilizando solo un código binario, la información no es legible para los usuarios
- Se deben incluir los costos para codificar y decodificarla información
- Se deben incluir los módulos de codificación y decodificación en todo el software con el que se trabaje

Supresión de secuencias repetidas

- Se debe elegir un valor no utilizado para indicar que sigue una secuencia de códigos

- Posteriormente se debe aplicar el algoritmo de *codificación por secuencia de código*

Codificación por secuencia de código

- Obtener una secuencia de valores
  - Copiar la secuencia de valores hasta que aparezca uno que se repita más de una vez
  - Cuando el mismo valor aparezca en sucesión, se sustituye por los siguientes 3 bytes:
    - El indicador especial de secuencia
    - El valor que se repite
    - El número de veces que el valor se repite

Ejemplo

- Suponga que se tiene la siguiente secuencia:

  22 23 24 24 24 24 24 24 25 26 26 26 26 26 26 25 24

- Utilizar el byte 0xff como indicador de secuencia

Solución

- Los tres primero códigos son copiados como aparecen
- La secuencia de 24 y 26 son codificados
- Los códigos restantes solo son copiados
- La secuencia resultante es:

  22 23 24 ff 07 25 ff 26 06 25 24

Conclusiones

- La codificación por secuencia es otro ejemplo de reducción por redundancia
- Este algoritmo de codificación es sencillo y su costo asociado rara vez afecta el desempeño de forma perceptible
- Es posible que esta técnica de codificación no reduzca el tamaño de la información

Asignando longitudes de código variables

- Técnica basada en el principio de que algunos valores ocurren con mayor frecuencia que otros
- Estos valores deben ocupar la menor cantidad de espacio
- Dado que los conjuntos de datos no tienen una distribución precisa se construyen tablas de codificación de manera dinámica (*Códigos de Huffman*)

Códigos de Huffman

- El algoritmo consiste en la creación de un ABB que tiene un símbolo por hoja, y construido de tal forma que siguiéndolo desde la raíz a cada una de sus hojas se obtiene el código *Huffman* de cada símbolo.
  - Se crean varias hojas, una por cada uno de los símbolos que se tengan, y etiquetada con su símbolo y su frecuencia de aparición.
  - Se toman los dos árboles (hojas) 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.
  - Se repite el paso anterior hasta que sólo quede un árbol.

Obtención del código de un símbolo

1. Comenzar con un código vacío
2. Iniciar el recorrido del árbol en la hoja asociada al símbolo
3. Comenzar un recorrido del árbol hacia arriba
4. Cada vez que se suba un nivel, añadir al código la etiqueta de la rama que se ha recorrido
5. Tras llegar a la raíz, invertir el código
6. El resultado es el código *Huffman* deseado

Obtención del símbolo a partir del código

1. Comenzar el recorrido del árbol en la raíz de éste
2. Extraer el primer símbolo del código a descodificar
3. Descender por la rama etiquetada con ese símbolo
4. Volver al paso 2 hasta que se llegue a una hoja, que será el símbolo asociado al código

Ejemplo

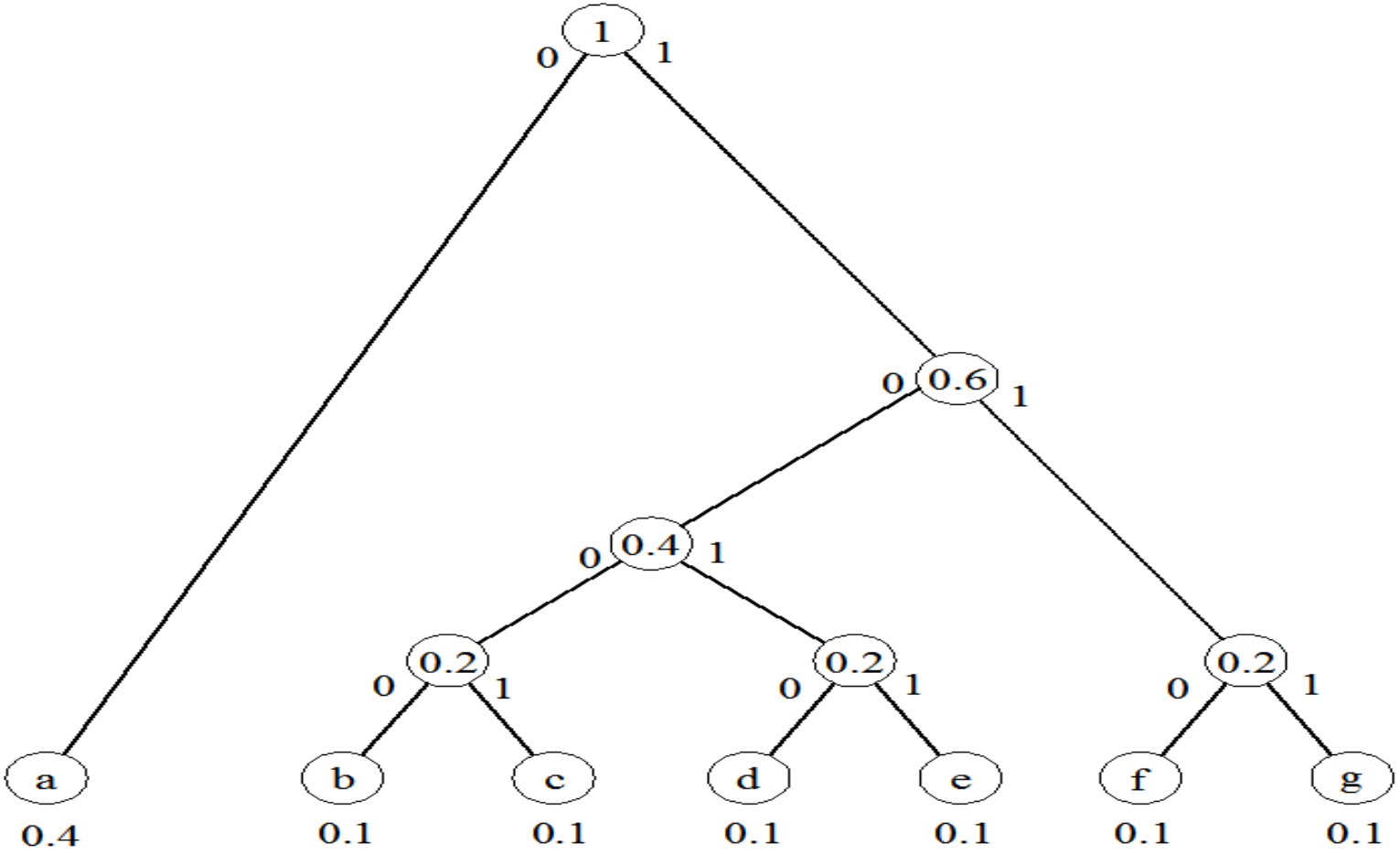
- Un conjunto de datos contiene solo 7 caracteres:

a, b, c, d, e, f, g

- Cada carácter tiene la siguiente probabilidad de aparecer y el siguiente código de *Huffman*:

Símbolo	a	b	c	d	e	f	g
Probabilidad	0,4	0,1	0,1	0,1	0,1	0,1	0,1

Árbol de generación



Códigos de los símbolos

Símbolo	a	b	c	d	e	f	g
Código *Huffman*	0	1000	1001	1010	1011	110	111

- Mensaje “abde” codificado: 01000101011011

Técnicas de compresión irreversible

- Es otro tipo de compresión en donde la información no puede recuperarse
- Ejemplos de esta técnica es la compresión de una imagen (400 x 400 pixeles) a una de menor resolución (100 x 100 pixeles)
- Esta técnica es poco común en los archivos de datos

Reutilización del espacio en disco

Introducción

- Conforme un archivo se va modificando, su estructura se va degradando.

- Causas de la modificación:
  - Adición de registros
  - Actualización de registros
  - Eliminación de registros

Eliminación de registros

- El objetivo es reutilizar el espacio que deja un registro al ser eliminado

- Se tienen dos maneras de borrar un registro
  - Eliminar de manera inmediata el registro para utilizar su espacio
  - Mantener el registro un tiempo solamente indicando que no está disponible

Marcar el registro como eliminado

- En esta técnica se coloca un marcador especial al registro para indicar que no está disponible en el archivo
- El programa debe saber reconocer cuando un registro está marcado como eliminado
- Después de cierto tiempo se deben eliminar del archivo
- Esta técnica permite recuperar de manera sencilla un registro eliminado

Re utilización dinámica de espacio

- Ésta técnica se utiliza para utilizar de manera inmediata el espacio liberado por un registro eliminado

- Para su uso se debe garantizar que:
  - El registro debe ser marcado de manera especial
  - Se pueda encontrar el espacio dejado por ese registro

Requisitos

- Para realizar esto, es necesario tener:
  - Una manera de saber si existen *slots* (espacios) libres en el archivo
  - Una manera de saltar directamente a uno de esos espacios

Uso de listas ligadas

- Lista de disponibilidad. Es una lista ligada en la que se encuentran los registros eliminados
- Cuando se inserta un nuevo registro en el archivo, se puede insertar en cualquier espacio disponible. Esto permite:
  - Que los registros libres se inserten en cualquier posición de la lista
  - Que no se tenga que mantener la lista ordenada

Uso de pilas

- La manera más sencilla es el uso de una pila (LIFO), en donde la inserción y la extracción se realizan en el primer elemento
- Lo que se inserta en la pila es el RRN del registro vacío así como un apuntador a otro registro disponible
- El último registro disponible debe apuntar hacia un valor de RRN nulo (-1)

Ligado y apilado de registros eliminados

- El funcionamiento de esta pila es el siguiente:
- Se consideran dos posibles casos
  - El apuntador al inicio de la pila indica el fin de lista (-1)
  - El apuntador indica un RRN válido

Caso 1: RRN no válido

- Si el apuntador al inicio de la pila contiene el fin de lista (-1), significa que no hay espacio dejado por algún registro y el nuevo registro se inserta al final del archivo

Caso 2: RRN válido

- Si el apuntados contiene un RRN válido, simplemente se utiliza el RRN para ir a la posición donde se insertará el nuevo registro
- Posteriormente ese registro se debe eliminar de la pila