UTN-FRBA

GESTIÓN DE DATOS

COMPRESION DE DATOS – ALGORITMO DE HUFFMAN

DIRECTOR DE CATEDRA: ING. ENRIQUE REINOSA

COMPRESION

- Compresión
  - Existen dos tipos de algoritmos de Compresión
    - ***Con Pérdida***: son algoritmos que pierden información al comprimir y que por dicha pérdida no son reversibles. Este tipo de algoritmos se utiliza para los archivos multimedia.
    - ***Sin Pérdida***: son algoritmos que comprimen archivos sin perder información, por lo cual son reversibles, dado que permiten volver al estado original del archivo. Se utilizan para todos los archivos menos los multimedia.

COMPRESION

- Por que es posible comprimir un archivo?
  - El alfabeto ASCII se creó de longitud fija de forma tal que por más que un carácter se repita mucho màs que otro el espacio que ocupa es el mismo.
  - No todos los archivos contienen los 256 caracteres especificados en el archivo ASCII, por lo cual podría ocurrir que no se utilicen todos desperdiciando bits en su representación.

COMPRESION

- Algoritmo de Huffman

  - Huffman es un algoritmo de compresión de datos sin pérdida, que es muy eficiente con archivos de texto (txt, excel, word, pdf, etc).
  - Las computadoras codifican los caracteres mediante ASCII o Unicode, utilizando un tamaño fijo para cada uno de ellos, 1 byte en el primer caso y 4 bytes en el segundo.
  - El algoritmo identifica cada uno de los caracteres distintos en el conjunto de archivos a comprimir, y le asigna un código de longitud variable según la frecuencia.
  - Cuanto mas veces aparezca un carácter en los archivos, la longitud de su código va a ser mas pequeña.

COMPRESION

- ***Proceso de Compresión***

A los efectos de simplificar la explicación del funcionamiento del algoritmo, vamos a mostrar su comportamiento comprimiendo solo el contenido de un archivo txt que continene "EN NEUQUEN".

- Se comienza leyendo el archivo e identificando todos los caracteres distintos que lo componen.
- Esos caracteres identificados se almacenan conjuntamente con la cantidad de repeticiones del mismo en un vector.

E	N	' '	U	Q
3	3	1	2	1

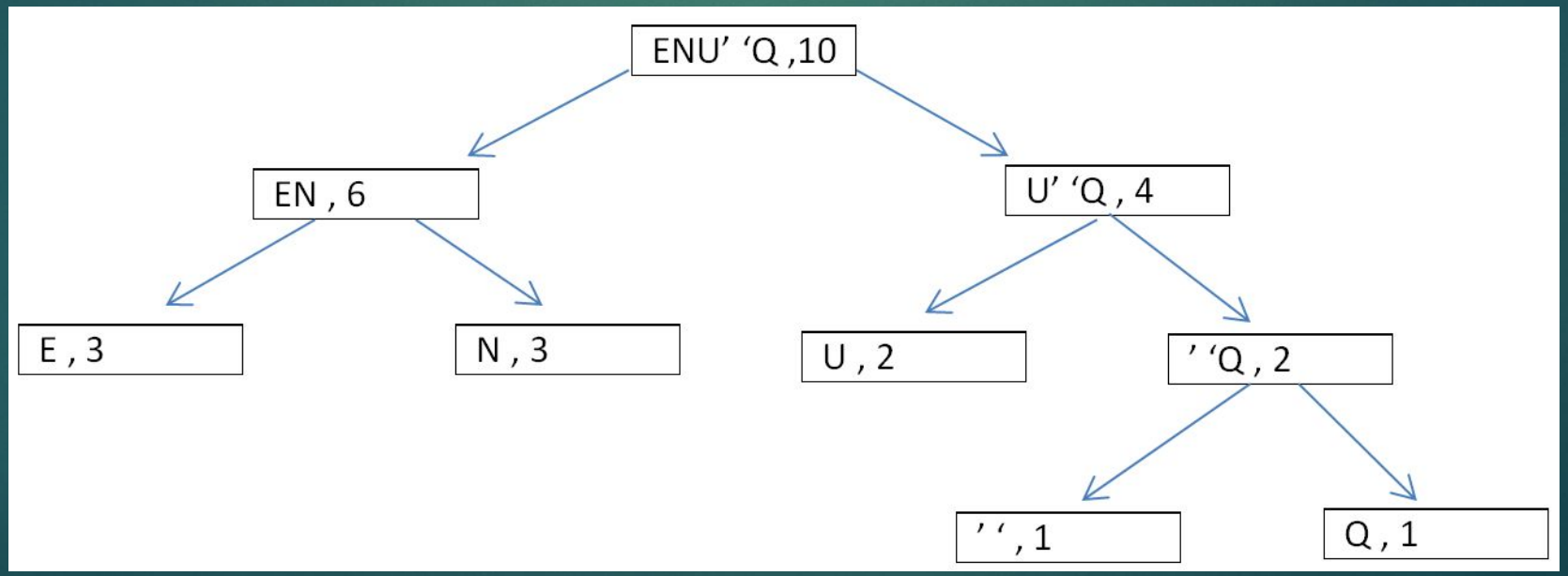
COMPRESION

- Luego se ordena el vector por cantidad de repeticiones en forma descendente, de forma tal que quede en la primer posición el elemento que más se repite.

E	N	U	‘ ‘	Q
3	3	2	1	1

COMPRESION

- Por último crea un árbol binario dividiendo el vector de a dos en función de la cantidad de repeticiones de cada carácter, comenzando con todo el vector como raíz del árbol y llegando hasta que las hojas esten compuestas por un solo caracter.



COMPRESION

- Luego por convención se define que la izquierda se representa con 0 y la derecha con 1.

- Se vuelve a leer el archivo y para cada carácter se va a buscando en el árbol generando el código de bits que representará ese carácter en función de que la búsqueda vaya por izquierda o por derecha.

- Por Ejemplo la codificación generada para la primera E del archivo es 00 daado que para acceder a la hoja que contiene el carácter E nos trasladamos dos veces a la izquierda.

COMPRESION

- Al leer el archivo se genera la siguiente codificación binaria:

EN NEUQUEN

'00:01:110:01:00:10:111:10:00:01'

le agrega 2 bits más, para completar 8bits = 1 Byte

- Luego se toma esa cadena de 1 y 0 y se convierte en caracteres para generar el archivo comprimido completando los caracteres faltantes con 0 o 1 hasta llegar al tamaño de byte requerido.

[00011100][10010111][10001000]

8 bits + 8 bits + 8 bits = 24 bits (3*1 Bytes = 3 bytes)

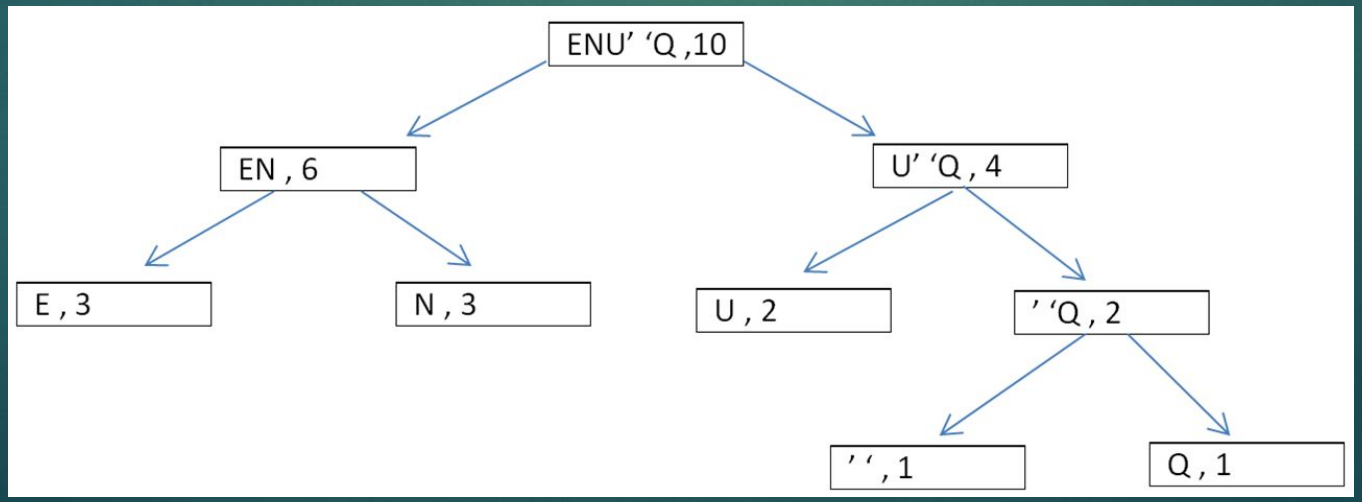
- Ese archivo es el que se genera como archivo comprimido.

COMPRESION

- ***Proceso de Descompresión***

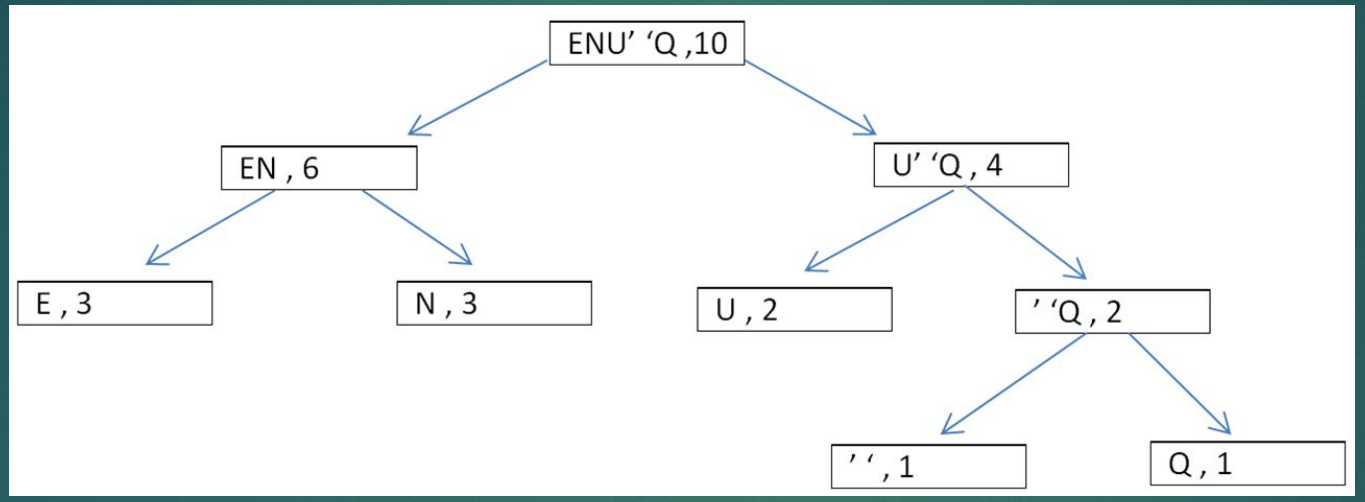
Para descomprimir el archivo generado se parte del archivo comprimido y se realiza la lectura del mismo para que en función del árbol se obtengan los caracteres originales

[00011100][10010111][10001000]



COMPRESION

[00011100][10010111][10001000]



Generando 'EN NEUQUEN'

COMPRESION

- ***Condiciones** a tener en cuenta*:
  - Es necesario guardar el vector en archivo comprimido, poe ese motivo los archivos muy pequeños no se comprimen.
  - Determinar donde finaliza el vector y donde comienzas los caracteres que representan los 1 y 0.
  - Identificar caracteres EOF que pueden generarse al azar.

COMPRESION MULTIMEDIA

- ***Dado que la codificación se da a partir de los bites y no de los bytes definidos mediante el alfabeto ASCII***

  - Es necesario guardar el vector en archivo comprimido, poe ese motivo los archivos muy pequeños no se comprimen.

  - Determinar donde finaliza el vector y donde comienzas los caracteres que representan los 1 y 0.

  - Identificar caracteres EOF que pueden generarse al azar.

COMPRESION MULTIMEDIA

- ***La forma de comprimir archivos multimedia es modificar su codificación:***

  - Recodificando la codificación de ***resolución*** de los archivos bajando la calidad de los mismos para utilizar menos bits para su representación.

  - Recodificando la codificación de ***definición*** de los archivos bajando la calidad de los mismos para utilizar menos bits para su representación.