LZW Compressor and Decompressor Algorithm

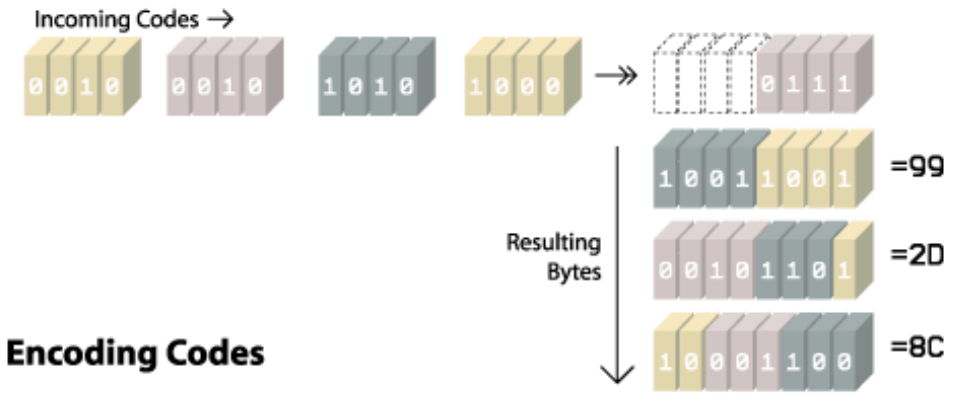
Javier Acosta
Daniel Méndez
Willy Villalobos

Universidad de Costa Rica

9 de julio de 2014

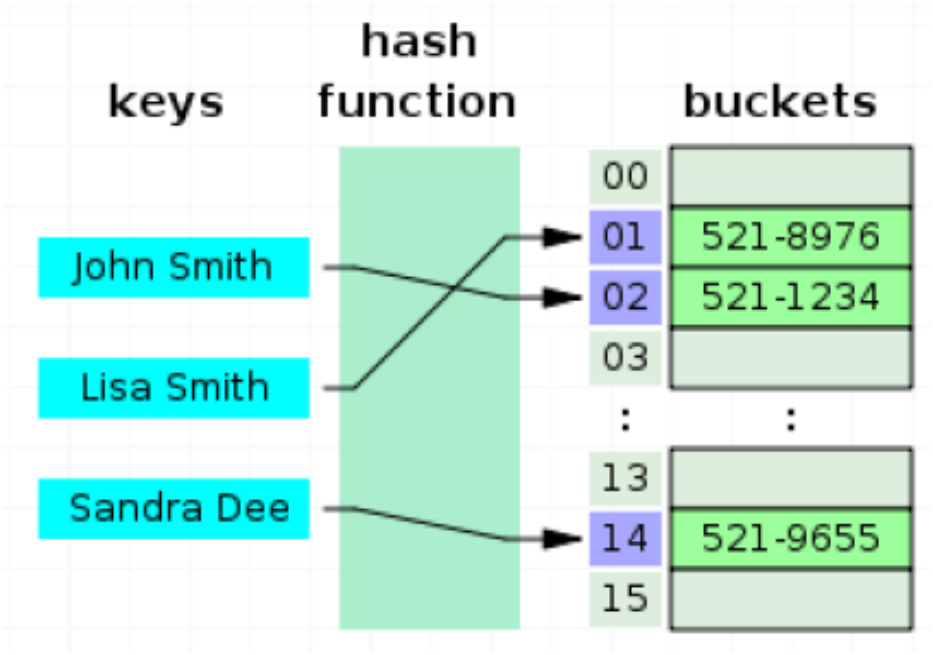
¿Qué es LZW?

Es un algoritmo de compresión sin pérdida desarrollado en 1984 por Terry Welch. La actual implementación hace uso de la función hash para comprimir y descomprimir los datos.

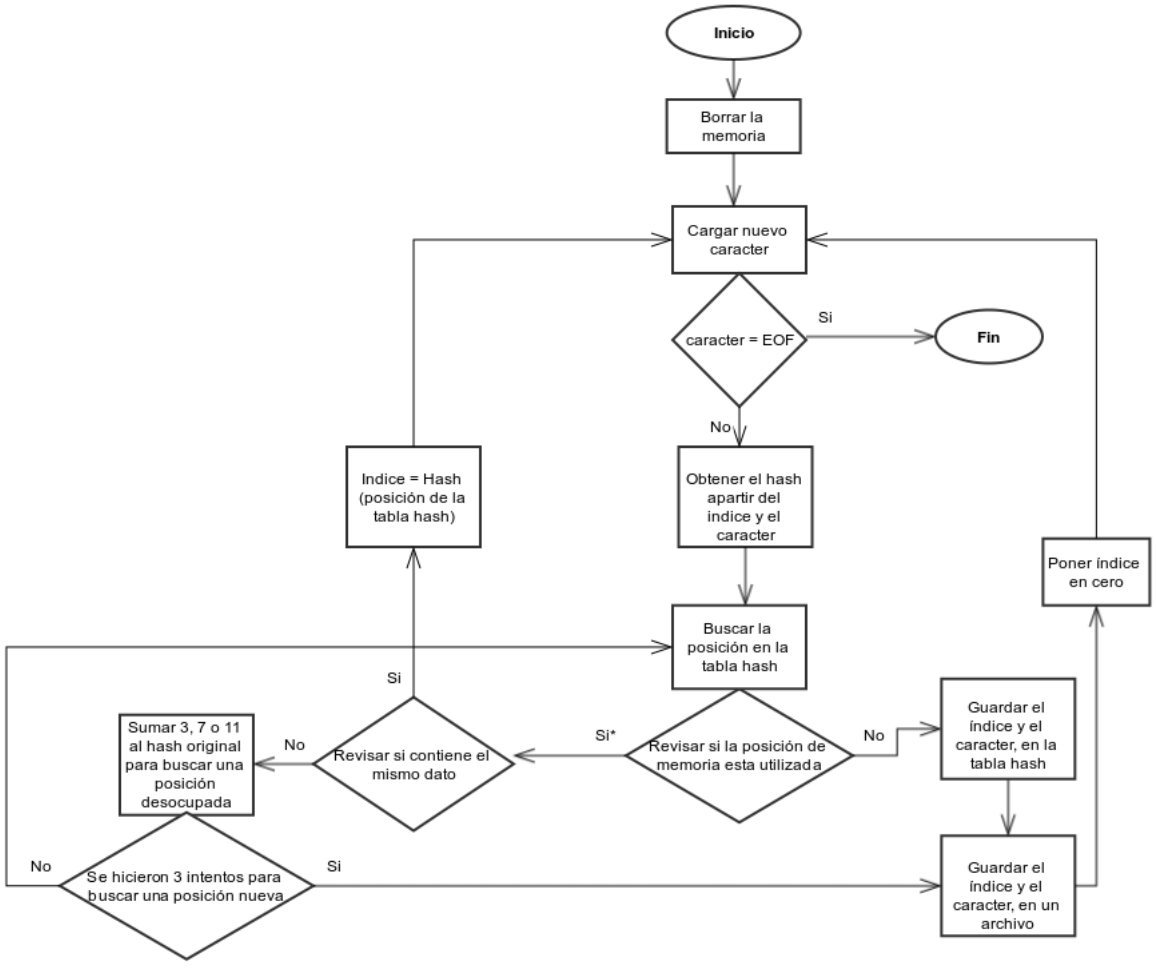


¿Qué es LZW?

La función hash, básicamente, se emplea como método para mapear datos de tamaño o longitud arbitrarios en bloques de un tamaño fijo, de manera que la más mínima diferencia en los datos de entrada de la función genera drásticas diferencias en la salida. Aplicación de esto sería ssh



Compresor LZW



Análisis de complejidad

De acuerdo con el análisis de complejidad, para el peor escenario tenemos

$$\Theta\left(n^2\right); \quad O\left(n^2\right); \quad o\left(n^3\right); \quad \Omega\left(n\right); \quad \omega\left(n\right) \tag{1}$$

Para el mejor escenario tenemos:

$$\Theta\left(n^0\right); \quad O\left(n^0\right); \quad o\left(n\right); \quad \Omega\left(n^{-1}\right); \quad \omega\left(n^{-1}\right) \tag{2}$$

Análisis de correctitud

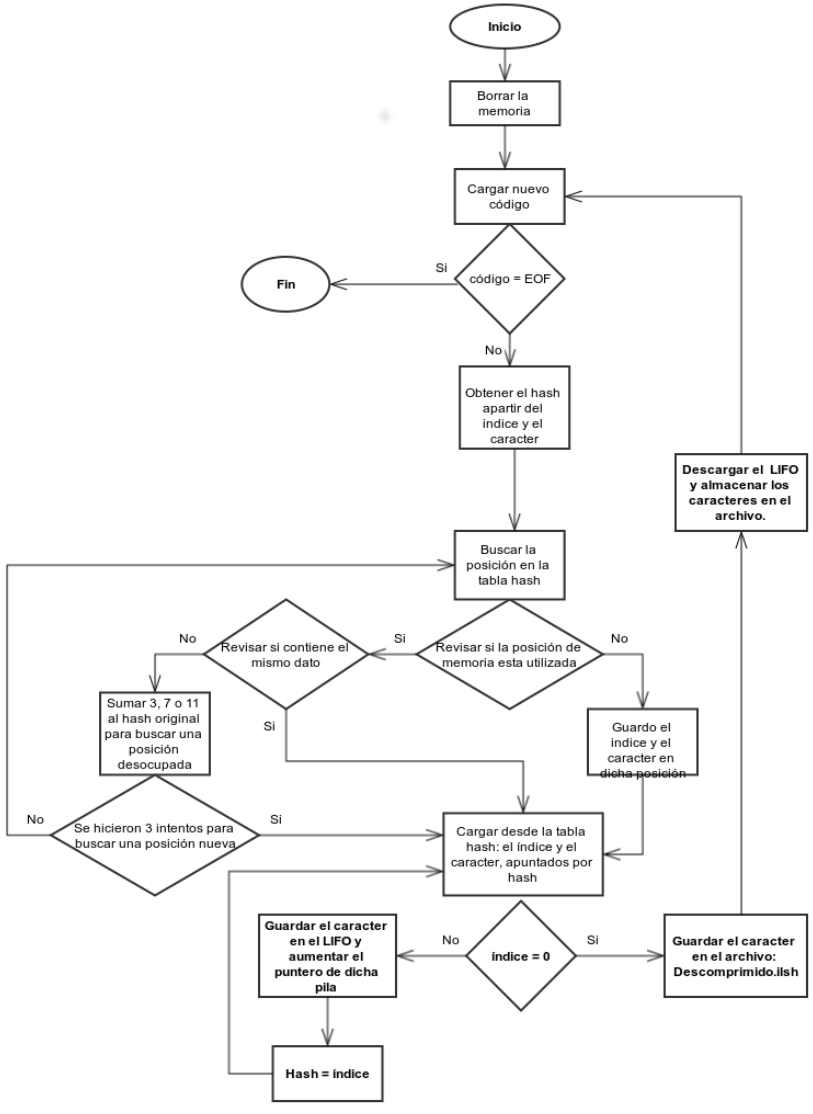
Se emplea el invariante de ciclo "*El índice más el caracter son una nueva entrada, o ya está guardado*". Se obtienen las siguientes conclusiones para inicialización, mantenimiento y finalización:

- **Inicialización:** Cuando se obtiene el primer valor, como la tabla está inicializada en 0, al aplicar el hash, se va a obtener un nuevo valor, por lo que sería una nueva entrada.

Análisis de correctitud

- **Mantenimiento:** Si durante el proceso de llenado de la tabla, se da el peor caso que es que la tabla está llena, implicaría que ya existe el hash que estamos analizando, por tanto siempre sería un valor previamente guardado, cumpliéndose el invariante.
- **Terminación:** Dado el último valor, puede darse el peor caso, que como se mencionó antes implicaría que el valor ya está previamente guardado, si fuese un nuevo caracter, simplemente se guardaría en la tabla para ser procesado, para finalmente el final del archivo hiciera el break y terminar la ejecución del programa.

Descompresor LZW



Análisis de complejidad

De acuerdo con el análisis de complejidad, para el peor escenario tenemos

$$\Theta\left(n^2\right); \quad O\left(n^2\right); \quad o\left(n^3\right); \quad \Omega\left(n\right); \quad \omega\left(n\right) \tag{3}$$

Para el mejor escenario tenemos:

$$\Theta\left(n^0\right); \quad O\left(n^0\right); \quad o\left(n\right); \quad \Omega\left(n^{-1}\right); \quad \omega\left(n^{-1}\right) \tag{4}$$

Análisis de correctitud

Se emplean dos invariantes de ciclo. El primero es "*El código existe dentro de la tabla hash o es uno nuevo*". Se obtienen las siguientes conclusiones para inicialización, mantenimiento y finalización:

- **Inicialización:** para el primer dato, dado que la tabla está en cero, hay que guardarlo, por tanto es un valor nuevo.

Análisis de correctitud

- **Mantenimiento:** Durante el proceso de llenado de la tabla, si se da una repetición de hash, significa que el código ya está guardado, por tanto ya existe dentro de la tabla hash.
- **Terminación:** Si no hubiesen colisiones, la tabla hash se llena con puros valores nuevos, caso contrario, el último valor choca y haría que ya existiese el valor dentro de la tabla.

Análisis de correctitud

La segunda invariante es "*El valor del índice de la tabla hash corresponde a un valor distinto de 0 para la siguiente posición en la tabla y se guarda en el LIFO, o es 0 y se limpia el LIFO*". Se obtienen las siguientes conclusiones para inicialización, mantenimiento y finalización:

- **Inicialización:** cuando llega el primer código de índice, el valor se guarda en la tabla de hash. Se carga y se analiza, como es el primero existen dos opciones, o es parte de un código nuevo, lo que implicaría que su índice no es cero, por lo que se debe guardar en el LIFO, o que el valor sea un código único, lo que implicaría que su índice es 0 y se pasa directamente al archivo de salida. Sin embargo, dado que la idea original es comprimir un archivo, lo más probable es que sea un código con un índice distinto de 0.

Análisis de correctitud

- **Mantenimiento:** Cuando ya se vaya llenando la tabla, habrá colisiones que harán que se vaya obteniendo nuevos índices. Si hay un choque en la tabla de hash, esto indicará que el valor ya está en la tabla, por lo que su índice es distinto de cero y pasará a ser guardado en el LIFO. Si fuese el último código de la trama, su índice sería 0 y pasaría a formar parte del documento que se está creando, a su vez que el LIFO va a vaciarse en el orden "último en entrar, primero en salir". Por tanto se cumple el invariante.
- **Terminación:** Para el último valor del archivo, este siempre será único, pues fue el primer valor que se guardó en la tabla para la compresión, por lo que su índice siempre será 0, por lo que se cumple el invariante también.

Tiempo para una demostración

Gracias por su atención