Juan Marín Vega - Fernando Moral Algaba

Tecnologies multimèdia

Avaluació cont. 4b

1. Introducción

El objetivo de este trabajo es probar el algoritmo LZ77 con archivos de texto. En la primera parte de este trabajo se intenta comprimir una secuencia aleatoria de 0's y 1's. El resultado obtenido demostraba que no era posible comprimir estos datos generados aleatoriamente.

En esta segunda parte probaremos a comprimir dos archivos de texto. Analizaremos el factor de compresión en función del tamaño de las ventanas deslizante y de entrada y el tiempo de proceso empleado en la compresión para cada uno de ellos.

2. Resumen de la implementación

- Implementada la función codificadora especificando los parámetros
  - -f El archivo a comprimir o descomprimir.
  - -s El tamaño de la ventana deslizante (sliding window).
  - -w El tamaño de la ventana de entrada (input window).
- Implementada la función decodificadora, es preciso especificar el parámetro ***--mode d.***
- Implementada una funcionalidad para realizar tests, es preciso especificar el parámetro ***--test*** o ***-t*** *X*.

  Este modo de funcionamiento ejecutará tests con una el archivo de texto indicado tamaños de sliding window que variarán desde 8 hasta 4096 en potencias de 2, y tamaño de input window desde 8 hasta el tamaño del sliding window en dicha iteración.

  En el modo de test es igualmente necesario especificar los parámetros requeridos por el programa (Slide window, input window) aún cuando no son usados para los tests.

3. Pruebas y resultados

Pruebas con el archivo Hamlet

Se realiza una prueba de compresión midiendo las diferentes input-window y sliding-window desde 8 hasta 4096. Se comprueban los ratios de compresión y el tiempo requerido

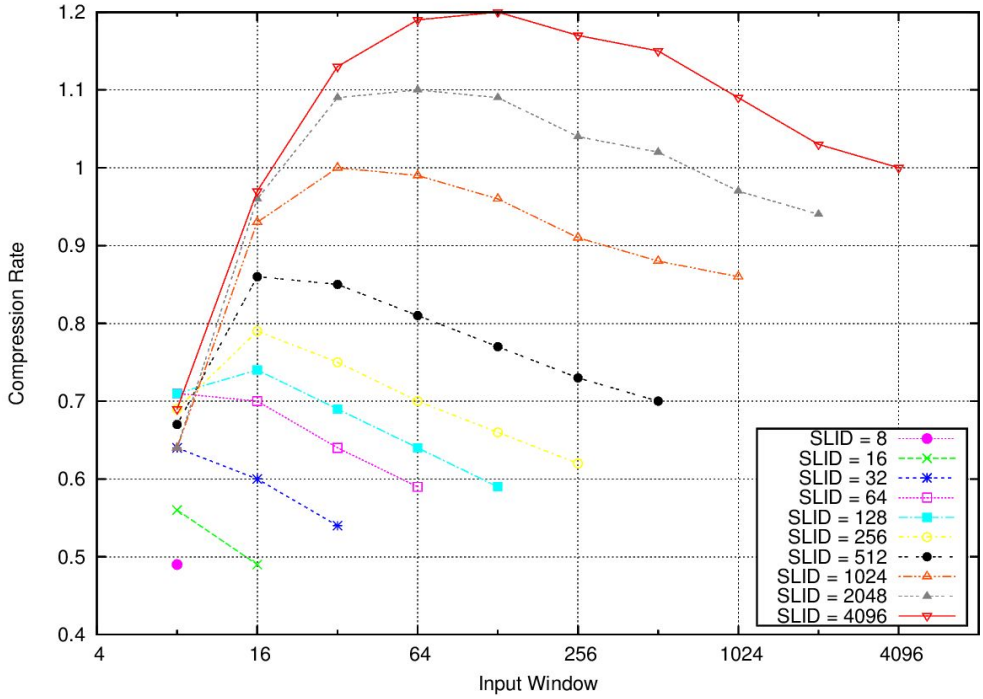

Figura 1. Relación de compresión para el archivo hamlet

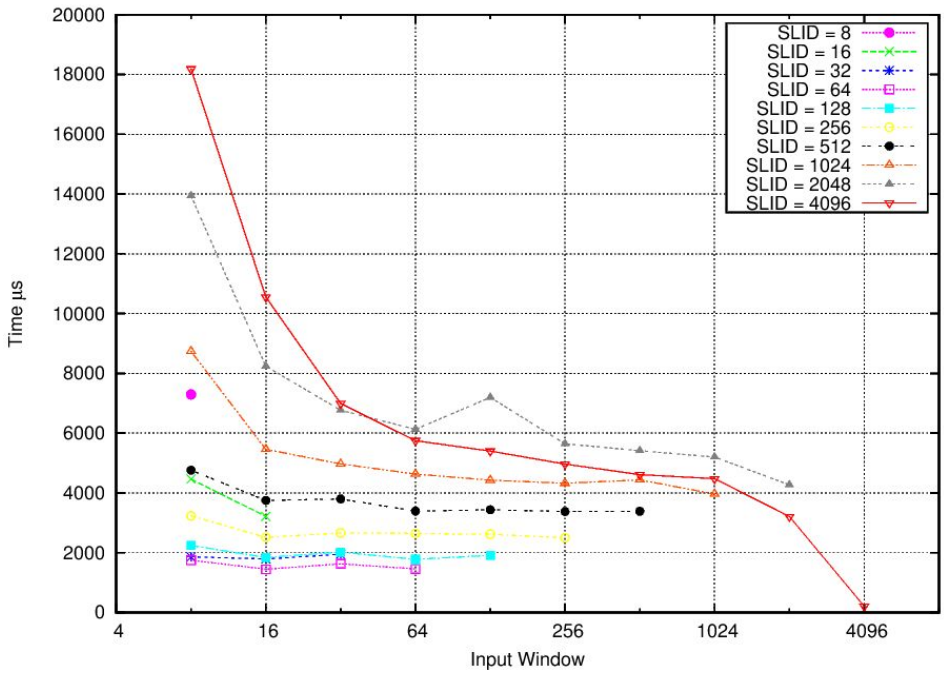

Figura 2. Tiempo requerido para el archivo hamlet

Pruebas con el archivo Quijote

Con el archivo quijote realizamos las mismas pruebas.

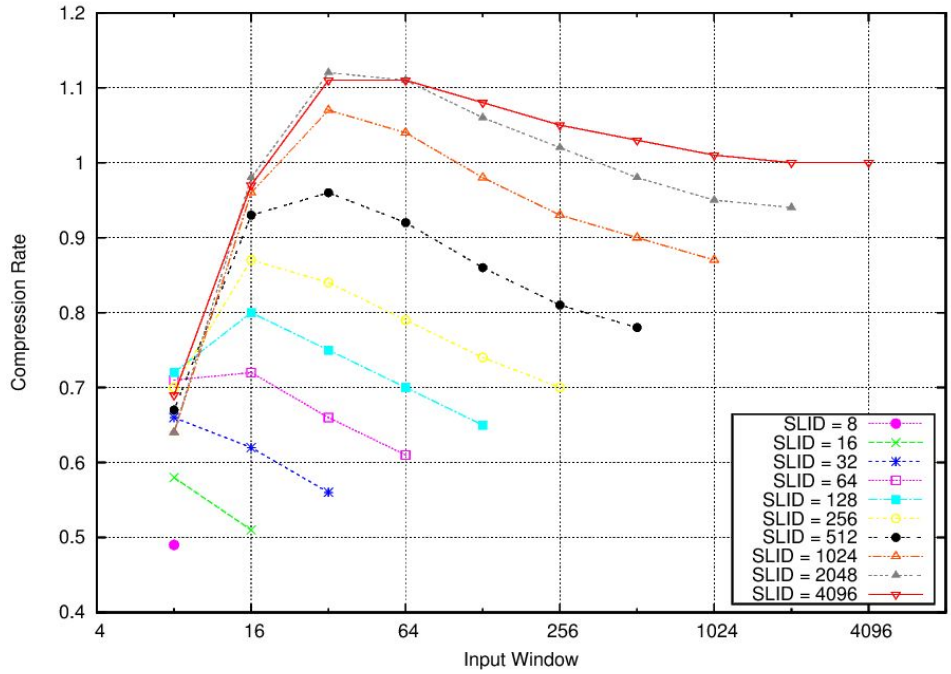

Figura 3. Relación de compresión para el archivo hamlet

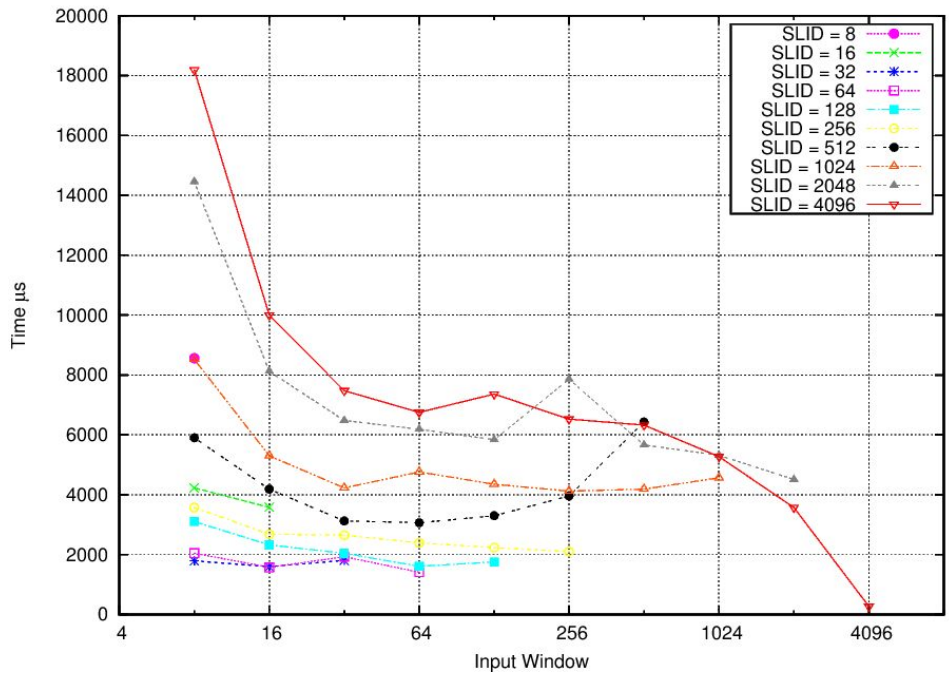

Figura 4. Tiempo requerido para el archivo quijote

4. Conclusiones

A diferencia del ejercicio anterior en el que intentamos comprimir una cadena binaria aleatoria, comprobamos que el algoritmo LZ77 consigue comprimir archivos de texto.

Se observa un mayor ratio de compresión para el archivo *hamlet*. Esto puede deberse a que contiene muchas palabras repetidas (al inicio de cada línea aparece el nombre del personaje que habla) con lo cual el algoritmo de compresión acaba siendo más eficiente. También observamos que hay más saltos de línea, esto podría influir en el ratio de compresión por idéntico motivo.

El archivo *quijote* contiene caracteres con acentos, que ocupan 2 bytes pero es un archivo en UTF-8 que utiliza codificación variable, con lo cual, no parece influir en el ratio de compresión. Más aún si consideramos que el cálculo del ratio de compresión no se hizo contando el número de letras que componen el texto sino que se obtuvo comparando la longitud de la cadena binaria correspondiente al texto comprimido y sin comprimir.

Así pues, para el archivo *hamlet* los resultados indican que podríamos utilizar un sliding window de 4096 y un input window de 128 ya que nos da un buen ratio de compresión y no nos penaliza en el tiempo.

En el caso de *quijote*, la mejor opción parece ser un sliding window de 2048 y un input window de 32, aunque la diferencia en el ratio de compresión al utilizar un sliding window de 4096 es tan pequeña que podría no ser significativa. Respecto a la velocidad de compresión si se aprecia una mayor velocidad si se usa el valor de 2048 en lugar de 4096.