Albert Folch Garcia
Xavi Moreno Liceras

Exercici AvCont-4d: LZ77 imatge

1.- Aplicad vuestro compresor LZ77 a los datos contenidos en la imagen BMP que teneis disponible en Campus Virtual. Se trata de una imagen de pequeñas dimensiones y 256 colores. Consejo:Extraed las componentes R,G,B de cada píxel, vectorizádlas (pasad de matriz 2D a vector 1D) y concatenádlas (primero vector de R, después vector de G y finalmente, vector B). Convertid esta secuencia de enteros de 8 bits a un string binario.

Ejecutar el lz77-bmp con el siguiente comando: python lz77-bmp.py
Veremos que nos saldrá:
Hint:
* compress: python lz77-bmp.py my/path/myfile.txt /my/path/namecompressfile

Por ejemplo podemos comprimir poniendo:
python lz77-bmp.py resource/cubo_LZ77.bmp cub

Si se quiere cambiar el *Mdes* i el *Ment*, se puede cambiar desde lz77-bmp.py, aunque por defecto ya está puesto un *Mdes* i *Ment* más óptimo.

Respecto la pràctica anterior, hemos creado una nueva función que lee una imagen, la convierte en RGB, luego concatena los tres canales en un flujo de bits. Hay que tener en cuenta que el bmp 'cubo_LZ77.bmp' no guarda la imagen en RGB. El razonamiento es el siguiente:

- Las características de la imagen són:
  - La imagen pesa: 5.878 bytes
  - Mide: 80 x 60 píxels
  - El rango de valor es de 0 a 255 (8 por valor en cada canal)

  Si estuviera guardado en RGB, la imagen pesaría:
  80*60*3 + cabecera ~ 14400 bytes

Ahora nos podemos preguntar: ¿Porqué pesa tan poco entonces? Bien, resulta que la imagen está guardada con una paleta de color, si nos fijamos un canal pesa: 4800 bytes (=80*60). Por tanto: tenemos 1078 bytes (=5.878 - 4800) donde se guarda la información de cabecera de la imagen más la paleta.

¿Porqué explicamos todo esto entonces? Nosotros en esta práctica cuando hemos hecho las pruebas, las hemos hecho partiendo que tenemos la imagen con los tres canales, es decir una imagen de 14400 bytes (115 200 bits)

2.- Ajustar los tamaños de las ventanas de compresión entre los valores habituales para lograr compresión. ¿Qué conclusiones sacáis?

Tamaño total en bits: 115 200

Los factores de compresión que obtenemos son los siguientes:

Mdes	Ment	Tiempo compresión (segons)	Mida	Factor
1024	32	1.62	57942	1.9882 : 1
1024	64	10.14	42199	2.7299 : 1
1024	128	37.30	41281	2.7906 : 1
1024	256	98.54	43494	2.6486 : 1
1024	512	225.44	45982	2.5053 : 1
2048	32	1.98	60551	1.9025 : 1
2048	64	14.19	41562	2.7718 : 1
2048	128	59.49	38619	2.9829 : 1
2048	256	166.07	40472	2.8464 : 1
2048	512	450.59	42659	2.7004 : 1
4096	32	2.59	64410	1.7885 : 1
4096	64	23.03	42757	2.6943 : 1
4096	128	108.82	38783	2.9704 : 1
4096	256	549.77	40383	2.8527 : 1
4096	512	1164.97	42352	2.7201 : 1
8192	32	3.23	69192	1.6649 : 1
8192	64	34.80	45612	2.5257 : 1
8192	128	164.18	40399	2.8515 : 1
8192	256	888.81	41673	2.7643 : 1
8192	512	1717.64	43452	2.6512 : 1

Conforme aumentamos Mdes y Ment, el tiempo de compresión aumenta.

La combinación de Mdes y Ment que cogeriamos sería:

- Mdes: 2048
- Ment: 128

Se puede ver que en la peor compresión de todas, nos da un resultado muchísimo mejor que las prácticas anteriores, comprimiendo con un factor de: *1.6649 : 1*. Esto es debido a la redundancia de datos que hay en el flujo de éste. Ya vimos que en una cadena aleatoria la compresión era nula, pero en cambio en estos datos, teniendo en cuenta que los valores de la imagen son muy uniformes y varían muy poco, la redundancia es mayor y comprime mejor.