Universidad Politécnica de Cartagena

Escuela Técnica Superior de Ingeniería de Telecomunicación

PRÁCTICAS DE TRANSMISIÓN DE DATOS

Práctica 2: Codificador de fuente Huffman

INTEGRANTES DEL GRUPO:

NOMBRE Y APELLIDOS	CORREO ELECTRÓNICO
DIEGO ISMAEL ANTOLINOS GARCÍA	diego.antolinos@edu.upct.es
ANDRÉS RUZ NIETO	andres.ruz@edu.upct.es

Apartado 3.1. Responda a las siguientes cuestiones (añada cualquier código auxiliar que haya utilizado):

-
  ¿Es él código Huffman único? En caso negativo, proporcione otro posible árbol de expansión.
  El código de Huffman no es único ya que depende de como se asignan los dígitos binarios durante la expansión del árbol.
- Usando las frecuencias absolutas del árbol anterior, ¿qué entropía posee el alfabeto?
  Posee una entropía H=2.1858
- Podemos calcular la longitud media que posee la codificación como:

$$\bar{L} = \sum_{i=1}^{N} p_i L_i$$

  Donde i, p_i y N tienen una interpretación idéntica a la de la fórmula de la entropía (práctica 3), y L_i es el número de dígitos binarios que la codificación asigna al símbolo i-ésimo, por ejemplo L_A=1, L_D=3. Comparé la entropía y la longitud media, ¿qué conclusiones saca?.
  Al realizar el sumatorio anterior se obtiene un valor de longitud media de 2.23
  Por tanto, H<=L media.

- Dado el mensaje “AAACEDCA”
  ¿Qué longitud en dígitos binarios tiene el mensaje si se codifica en ASCII 8-bit? Proporcione la codificación del mensaje (tiene una tabla ASCII en el aula virtual).
  Si cada símbolo está compuesto por 8 bits posee una longitud de 64 dígitos binarios.
  01000001 01000001 01000001 01000011 01000101 01000100 01000011 01000001

  - ¿Qué longitud en dígitos binarios tiene el mensaje si se codifica Huffman? Proporcione la codificación del mensaje de acuerdo al árbol de expansión anterior.
    Posee una longitud de 16 códigos binarios.
    00010111 11101010

  Dado el mensaje, “1011101111111010”, codificado Huffman de acuerdo al árbol de expansión anterior, se le pide que lo decodifique.
  C D E E C A

Apartado 3.2.

COMPLETE LOS SIGUIENTES CÓDIGOS

```
function [indA, indB, acumulado, nuevop]  = compacta ( p )
    nP = p;
    [vA,iA]=min(nP);
    nP(iA)=NaN;
    [vB,iB]=min(nP);
    acumulado=vA+vB;
    nP(iB)=acumulado;

    if or(isnan(vA),isnan(vB))
        iA=-1;
        iB=-1;
        acumulado=0;
        nP=[];
    end
end

function  [nuevocodigo]  = expande ( codigo, indA, indB )
    nuevocodigo=codigo;
    nuevocodigo{indA}=[codigo{indB},'1'];
    nuevocodigo{indB}=[codigo{indB},'0'];
end

function  [codigo]  = huffman ( p )
    nP = p;
    idx=zeros(1,2);
    i = 0;
    L = size(p,2);
    c = repmat({''},1,L);
    iA = 0;
    iB = 0;

    while(or(iA~=-1,iB~=-1))
        i = i+1;
        [iA,iB,acumulado,nP] = compacta(nP);
        if(iA~=-1)
            idx(i,1)= iA;
        end
        if(iB~=-1)
            idx(i,2) = iB;
        end
    end
    idx = flipud(idx);

    for j = 1:size(idx,1)
        c = expande(c,idx(j,1),idx(j,2));
    end
end
```

Apartado 3.2. Responda a las siguientes cuestiones (añada cualquier código auxiliar que haya utilizado):

Consideremos como variable **X** el lanzamiento de una moneda, de una codificación Huffman de la misma. ¿Cuál es la longitud media de la codificación?
Para este caso la longitud media de la codificación es 1, esto se debe a que solo necesitamos un digito binario por cada símbolo, y cada símbolo posee una probabilidad de 0.5
Para los demás casos utilizamos la función:

```
function [Longitud] = Lmedia(p)
q=huffman(p);
Longitud=0;

      for i=1:length(q)
      Longitud=Longitud+p(i)*numel(q{i});

      end
```

- Ídem para un dado de 6 caras.
  Longitud media igual a 2.67

- Ídem para un dado de 8 caras.
  Longitud media igual a 3

- Ídem para un dado de 14 caras.
  Longitud media igual a 3.857

- Ídem para un dado de 16 caras.
  Longitud media igual a 4

Consideremos ahora un dado de 6 caras cargado, tal que *p*=[.1 .2 .4 .2. .1 0], de una codificación Huffman y la longitud media de la misma.
p = [.1 .2 .4 .2 .1 0]
Codificacion de Huffman: '0110' '11' '00' '10' '010' '0111'
Longitud Media =2,3

- Consideremos ahora un dado de 8 caras cargado, tal que *p*=[.05 .2 .0.5 .2. .05 .2 .05 .2], de una codificación Huffman y la longitud media de la misma.
  p=[.05 .2 .05 .2 .05 .2 .05 .2]
  Codificacion de Huffman: '1011' '001' '1010' '000' '1001' '11' '1000' '01'
  Longitud Media =2,8

- A la vista de las respuestas anteriores, **¿bajo que condiciones la longitud media de la codificación coincidirá con la entropía?**
  Cuando nos encontramos ante un conjunto de símbolos con la misma probabilidad y cuando la longitud del conjunto es del orden de 2^L'

- Dé la longitud media de codificación Huffman para el castellano a partir del quijote.
  Longitud media igual a 4.4236

- Dé la longitud media de codificación Huffman para la fotografía foto.bmp
  Longitud media igual a 6.1763

- Dé la longitud media de codificación Huffman para la fotografía foto.jpg
  Longitud media igual a 7.9543

- Compare los tres resultados anteriores con la información contenida en cada fichero (que obtuvo en la práctica anterior).

  Quijote.txt
  H =4.3858
  L'=4.4236

  Foto.bmp
  H = 6.1517
  L'= 6.1763

  Foto.jpg
  H = 7.9292
  L'= 7.9543

  La longitud media siempre es menor que H+1 y mayor que H → H < L' < H+1

Apartado 3.3.

COMPLETE LOS SIGUIENTES CÓDIGOS

```
function  [nuevotexto, longitud]  = codifica( texto, codigo )
    nuevotexto = '';

    for i=1:length(texto)
        nuevotexto=[nuevotexto,codigo{texto(i)+1}];
    end

    longitud = length(nuevotexto);
end
```

Apartado 3.3. Responda a las siguientes cuestiones (añada cualquier código auxiliar que haya utilizado):

- Obtenga para la cadena `'Hola mundo!'` la codificación ASCII y su longitud.
  nuevotexto=0100100001101111011011000110000100100000011011010111010101101110011001000110111100100001
  longitud = 88

- Con el código del castellano obtenido a partir del quijote, ¿cuál es la codificación y la longitud de la misma cadena `'Hola mundo!'`? ¿Qué ahorro en % hay respecto a la codificación ASCII?
  nuevotexto=00100001001000100001011110000010010011111000011001000010000100000
  longitud = 66
  25% de ahorro

- Ídem si usa el código obtenido de la fotografía foto.jpg
  nuevotexto=11010000111111100011101111101110000101110000011100101000001101011100001111111110000010001
  longitud = 89

- Explique los resultados.
  Cuando la entropía de la fuente con la que codificamos disminuye, resulta que el ratio de compresión que obtenemos aumenta.

COMPLETE LOS SIGUIENTES CÓDIGOS

```
function  [longitud, texto, nuevoflujo] = decodifica( flujo, codigo )
    nuevoflujo='';
    flujoVentana='';
    textoD='';
    flujoUsado='';

    for i=1:length(flujo)
        flujoVentana=[flujoVentana,flujo(i)];

        for j=1:length(codigo)
            if strcmp(flujoVentana,codigo{j})
                textoD=[textoD,j-1];
                flujoUsado=[flujoUsado,flujoVentana];

                flujoVentana='';
            end
        end
    end

    texto=textoD;
    longitud = length(texto);
    nuevoflujo=regexprep(flujo,flujoUsado,'');
end
```

Apartado 3.3. Responda a las siguientes cuestiones (añada cualquier código auxiliar que haya utilizado):

- A partir de las codificaciones obtenidas en el apartado anterior, realice llamadas sucesivas a:

  ```
  [l, t, f] = decodifica(codifica('Hola mundo', codigo), codigo);
  ```

  Compruebe que en todos los casos la función es capaz de devolver el mensaje original.

```
>> [longitud,texto,nuevoflujo] = decodifica(codifica('Hola mundo!', huffman(calculofrecuencias('quijote.txt'))),huffman(calculofrecuencias('quijote.txt')));

texto =

    'Hola mundo!'


longitud =

    11
```

```
>> [longitud,texto,nuevoflujo] = decodifica(codifica('Hola mundo!', huffman(calculofrecuencias('foto.jpg'))),huffman(calculofrecuencias('foto.jpg')));

texto =

    'Hola mundo!'


longitud =

    11
```

```
>> [longitud,texto,nuevoflujo] = decodifica(codifica('Hola mundo!', huffman(calculofrecuencias('foto.bmp'))),huffman(calculofrecuencias('foto.bmp')));

texto =

    'Hola mundo!'


longitud =

    11
```

Apartado 3.4. Responda a las siguientes cuestiones (añada cualquier código auxiliar que haya utilizado):

- Compruebe con algún fichero que el contenido de un fichero comprimido y vuelto a descomprimir coincide con el original (puede hacerlo con la orden `diff` en Linux).
- Rellene la siguiente tabla con la diferencia de tamaños en tanto por cierto respecto al original, obtenidos tras la compresión para cada par fuente/codigo:

\ Fuente Codigo \	Quijote	Hamlet	Foto.jpg
Quijote	0.446	-0.0165	-8.9123
Hamlet	0.3333	0.4141	-9.2419
Foto.jpg	-0.0227	-0.0245	0.0081
Platero y yo	0.0356	-0.1840	-5.9435

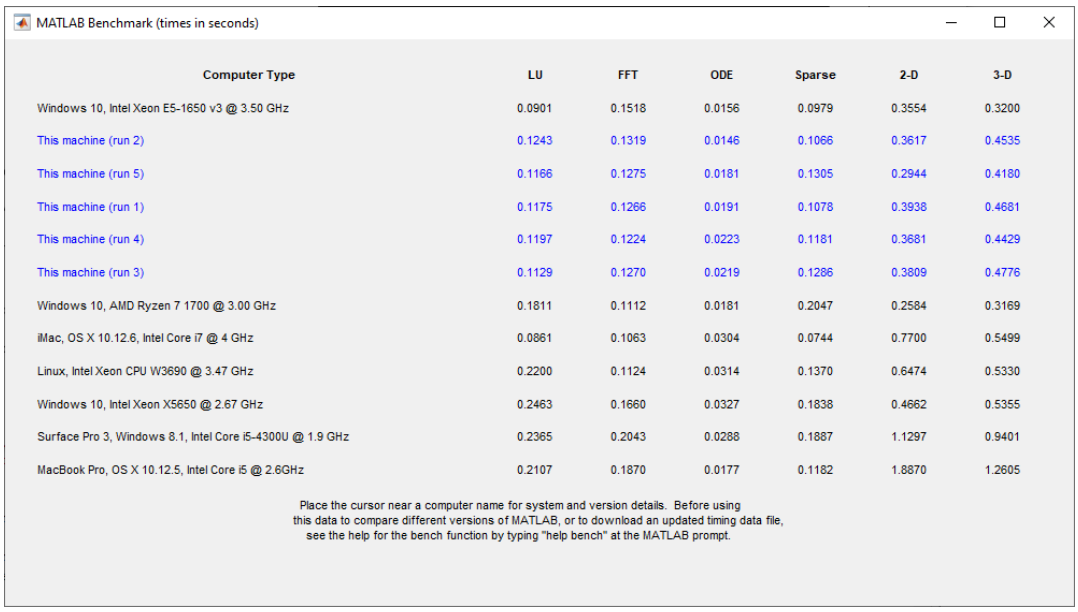
Explique los resultados de dicha tabla, señalando lo más significativo.

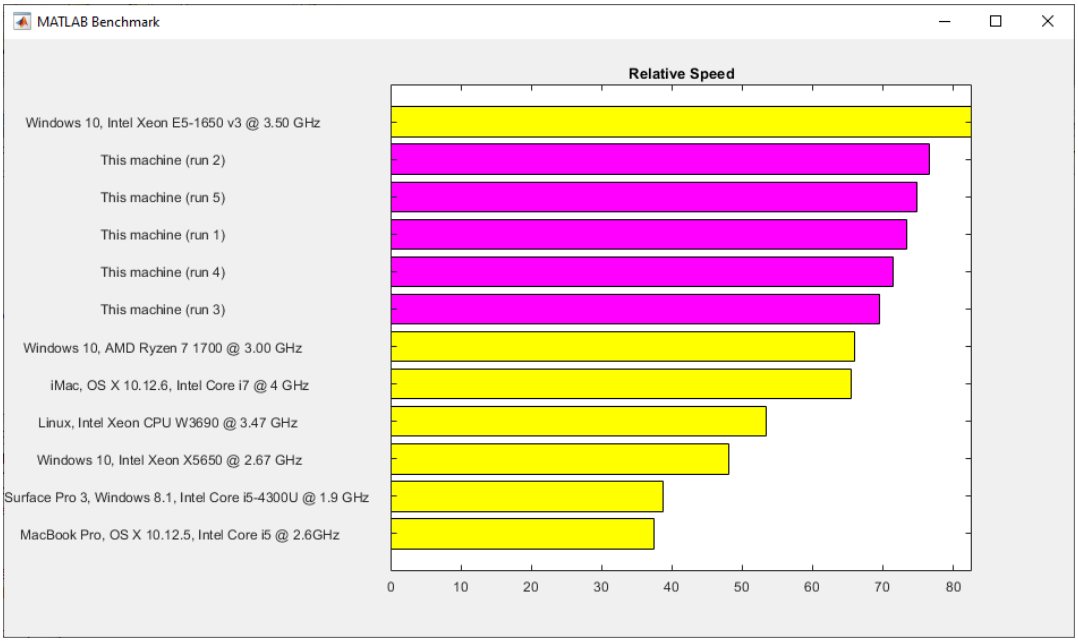
Vemos que existe una mejor compresión cuando usamos el código de un archivo sobre ese mismo archivo.

- Rellene los tiempos de ejecución correspondientes a cada caso (**ver video explicativo**)

\ Fuente Codigo \	Quijote	Hamlet	Foto.jpg
Quijote	203 s	672s	>30min
Hamlet	271s	104s	--
Foto.jpg	372.39s	202.77s	147.54s
Platero y yo	1049.51s	815.67s	--

- **Proporcione una captura de pantalla de la ejecución del comando** `tic; bench(5); toc`






```
>> tic; bench(5); toc;
Elapsed time is 3.547001 seconds.
```