Teoría de la Información
Trabajo Integrador N°2

Grupo N°: 5

Integrantes:
Noelia Echeverría (noelia.echeverria@fi.mdp.edu.ar) Leg. 4087
María Camila Ezama (camilaezama2000@gmail.com) Leg. 15103
Juan Cruz Mateos (juanczmt@gmail.com). Leg. 15134

Github: https://github.com/JuanCruzMateos/TeoriaDeLaInformacion

Resumen

Este trabajo está separado en dos partes. En la primera parte se aplican los algoritmos de compresión de Hoffman, Shannon-Fano y Run-Length-Coding (RLC) a 3 archivos distintos (2 archivos de texto y una imagen) y se comparan los resultados obtenidos. En la segunda parte, dados 3 canales de comunicación se calcula la información mutua, la equivocación y las propiedades de los canales y se obtienen conclusiones.

Introducción

Codificación y Compresión :

Ya sea para ahorrar espacio en disco, reducir el tiempo de transmisión en una red o simplemente para reducir la redundancia de los datos presentes, la compresión constituye una herramienta útil y poderosa para reducir el tamaño de un archivo al buscar una manera más compacta de representar la información. De esta manera, la compresión permite aumentar la capacidad de almacenamiento de los dispositivos y reducir el tiempo de transmisión de la información por un canal. Los métodos de compresión pueden clasificarse en dos, métodos sin pérdida o reversibles, que son aquellos que mantienen la integridad de la información, es decir que al descomprimir los datos, los datos obtenidos son iguales a los originales, y métodos con pérdida o irreversibles, que son aquellos que no mantienen la integridad de la información dado que al descomprimir no se garantiza que los datos obtenidos sean iguales a los originales. En los métodos de reversibles se verifica que la longitud media del codigo es mayor o igual a la entropía ($H \leq L$), mientras que los métodos irreversibles no presentan a la entropía como una limitación ($L < H$). Dentro de los métodos de compresión sin pérdida puede identificarse el método de Shannon-Fano y el método de Huffman. En el caso del RLC, se trata de un método sin pérdida cuando se representan todos los símbolos posibles, pero si se utilizan reglas adicionales para reducir la cantidad de símbolos se convierte en un método irreversible.

A fin de evaluar la eficiencia de la compresión se pueden utilizar distintos parámetros, tales como la tasa de compresión, el tiempo de compresión/descompresión y la calidad (error, e.g. error cuadrático medio, entre los datos originales y los datos descomprimidos). La elección del parámetro dependerá del interés y los objetivos que persigan los usuarios al desarrollar los algoritmos de compresión.

Canales de comunicación:

Cuando comenzamos a pensar en el concepto de canal de información, es decir, preparar información para ser transmitida, inmediatamente debemos considerar la posibilidad de cometer errores durante este proceso. Es por ello que es necesario tener en cuenta las características del canal y de la fuente de información.

Un canal de información viene determinado por un alfabeto de entrada A = $\{a_i\}$ i=1, 2, …, r; un alfabeto de salida B= $\{b_j\}$, j=1, 2, …, s y un conjunto de probabilidades condicionales $P(b_j/a_i)$.

Desarrollo

Primera parte: Codificación y compresión

El práctico cuenta con tres archivos: *Argentina.txt*, *Hawai.txt* e *Imagen.raw*.
***Argentina.txt*:** contiene el texto completo del Himno Nacional de Argentina.
Hawai.txt: contiene el texto completo del Himno Nacional de Argentina (traducido mediante Google Traductor) al idioma Hawaiano.
Imagen.raw: contiene la imagen de Mafalda de la práctica y en el formato de la práctica.

En esta primera parte realizamos la compresión de los tres archivos utilizando los algoritmos de Huffman, Shanon-Fano y RLC.

Compresión utilizando algoritmo de Huffman

El algoritmo de Huffman constituye un procedimiento recursivo sencillo para la construcción de un código instantáneo óptimo. El algoritmo resuelve el problema de generar un código compacto en el caso del alfabeto binario, agrupando recursivamente los símbolos menos probables para formar un nuevo símbolo. De esta manera se crea un árbol de codificación que puede ser recorrido recursivamente acumulando un cero o un uno, en caso de pasar al subárbol izquierdo o derecho respectivamente, hasta alcanzar un nodo hoja y obtener así la codificación del símbolo huffman correspondiente al símbolo de la fuente. De esta manera se genera un diccionario de codificación que puede utilizarse para reconstruir el archivo original, reemplazando cada símbolo por el código huffman correspondiente. A continuación se describe el pseudocódigo del algoritmo de Huffman para la construcción del árbol y el diccionario de codificación. El algoritmo completo implementado en Java puede consultarse en el repositorio de github de la materia haciendo click aquí. La implementación del algoritmo realizada se corresponde a la de un método estadístico semi estático, en el que se requieren dos pasadas sobre el archivo, una para armar del diccionario de codificación y otra para generar el archivo codificado. De esta manera, la distribución de probabilidades obtenida se ajusta a los datos.

```
# definición de nodo :
    Nodo {
       String palabra,
       double prob,
       Nodo izq,
       Nodo der
    }
```

```
        # genero el arbol de codificación
       # probabilidad := diccionario(string, double)
        pq = ColaPrioridad<Nodo>
        para cada símbolo:
                pq.encolar(Nodo(símbolo, probabilidad[símbolo], null, null)
        mientras que tamaño(pq) > 1:
                izq = pq.desencolar()
                der = pq.desencolar()
                nodoNuevo = Nodo(null, izq.prob + der.prob, izq, der)
                pq.encolar(nodoNuevo)
        raíz = pq.desencolar()

        # función para generar los códigos de huffman una vez obtenido el árbol
```

```
codigosHuffman := diccionario(string, string)
función generarCodigosHuffman(Nodo nodo, String codigo):
        si nodo es hoja (izq=null y der=null):
                codigosHuffman[nodo.símbolo] = código
        sí no:
                generarCodigosHuffman(nodo.izq, código + “0”)
                generarCodigosHuffman(nodo.der, código + “1”)
```

Los resultados obtenidos luego de la compresión de cada archivo se resumen en la siguiente tabla y las codificaciones de Huffman para cada uno de los archivos puede consultarse en el apéndice al final de este informe o en el siguiente link:

Huffman	Argentina.txt	Hawai.txt	imagen.raw
Tamaño original (bytes)	2500	3033	196618
Tamaño comprimido (bytes)	1399	1498	52235
Tam.Orig / Tam. Compr	1.79	2.02	3.71
Tasa Compresión	2:1	2:1	4:1
Entropía (bits)	4.51	4.21	1.94
Longitud media	4.54	4.25	2.13
Rendimiento (%)	99.24	99.20	91.39
Redundancia (%)	0.76	0.80	8.61

Tabla I : Resultados de aplicar el algoritmo de Huffman a los archivos Argentina.txt, Hawai.txt y Imagen.raw

Compresión utilizando algoritmo de Shannon Fano

Al igual que el algoritmo de Huffman, el algoritmo de Shannon Fano constituye un método recursivo sencillo para construir un código pero que, a diferencia de Huffman que genera códigos instantáneos óptimos, representa un procedimiento subóptimo para la construcción del código ya que alcanza una cota de longitud media de $L \leq H(S) + 2$. El algoritmo inicia ordenando las probabilidades de los símbolos en forma decreciente y recursivamente parte el conjunto de manera que la diferencia de la sumatoria de las probabilidades de los dos subconjuntos sea mínima y asigna un bit diferente (0 o 1) a cada uno de los subconjuntos. A continuación se describe el pseudocódigo del algoritmo para la construcción del árbol y el diccionario de codificación. El algoritmo completo implementado en Java puede consultarse en el repositorio de github de la materia haciendo click aquí.

```
 # definición de nodo shannon fano :
NodoShannonFano {
 Lista<Nodo> nodos, //Nodo corresponde al tipo de datos definido para el algoritmo de huffman
   NodoShannonFano izq,
   NodoShannonFano der
}
```

```
    # genero el arbol de codificación
```

```
# probabilidad := diccionario(string, double)

función generarArbolDeCodificacion():
     nodos = Lista<Nodo>
     para cada símbolo:
             nodos.agregar(Nodo(palabra, probabilidad[símbolo], null, null)
    ordenar(nodos) // de forma creciente de acuerdo a la probabilidad
    raíz = NodoShannonFano(nodos, null, null)
    generarArbolCodificacionRecursivo(raíz)

función generarArbolCodificacionRecursivo(NodoShannonFano nodo):
    si nodo.cantidadNodosInternos() > 1:
        nodo.izq = NodoShannonFano(nodo.mitadIzquierda(), null, null)
        nodo.der = NodoShannonFano(nodo.mitadDerecha(), null, null)
        generarArbolCodificacionRecursivo(nodo.izq)
        generarArbolCodificacionRecursivo(nodo.der)

función generarCodigos():
    encode(raíz, “”)

# codigosShannon := diccionario(string, string)
función encode(NodoShannonFano nodo, string s):
    si nodo.esHoja():
        codigosShannon.agregar(nodo.simbolo(), s)
    sí no:
        encode(nodo.izq, s + “0”)
        encode(nodo.der, s + “1”)
```

Los resultados obtenidos luego de la compresión de cada archivo se resumen en la siguiente tabla y las codificaciones de Shannon-Fano para cada uno de los archivos puede consultarse en el apéndice al final de este informe o en el siguiente link:

Shannon Fano	Argentina.txt	Hawai.txt	imagen.raw
Tamaño original (bytes)	2500	3033	196618
Tamaño comprimido (bytes)	1446	1525	52553
Tam.Orig / Tam. Compr	1.73	1.99	3.74
Tasa Compresión	2:1	2:1	4:1
Entropía (bits)	4.51	4.21	1.94
Longitud media	4.70	4.32	2.14
Rendimiento (%)	96.03	97.42	90.84
Redundancia (%)	3.97	2.58	9.16

Tabla II : Resultados de aplicar el algoritmo de Shannon a los archivos Argentina.txt, Hawai.txt y Imagen.raw

Compresión utilizando algoritmo de RLC

El algoritmo RLC es un método sencillo de compresión en el que se codifican secuencias de símbolos iguales con el par (símbolo, cantidad de repeticiones del símbolo). En la implementación del algoritmo utilizamos 2 bytes para representar los símbolos (unicode-16) y 2 bytes para representar su frecuencia. Este algoritmo es sumamente útil en los casos en los que se

tienen muchos datos adyacentes iguales, como en el caso de los píxeles de las imágenes, por ejemplo. A continuación se describe el pseudocódigo del algoritmo. El algoritmo completo implementado en Java puede consultarse en el repositorio de github de la materia haciendo click aquí.

```
función comprimirTXT(string nombreArchivo):
    archivo = abrir(nombreArchivo)
    archComprimido = abrir(nombreArchivoComprimido)
    actual = leer(archivo)
    mientras que no sea fin de archivo:
         anterior = actual
         frecuencia = 1
         mientras ((actual = leer(archivo)) == anterior):
              frecuencia += 1
         escribir(archComprimido, anterior)
         escribir(archComprimido, frecuencia)
    cerrar(archivo)
    cerrar(archComprimido)


función comprimirRAW(string nombreArchivo):
    archivo = abrir(nombreArchivo)
    archComprimido = abrir(nombreArchivoComprimido)
    actual = leer_int(archivo) // leer_int :: ignoro blancos (\n, \t, \r, espacios, etc.)
    mientras que no sea fin de archivo:
         anterior = actual
         frecuencia = 1
         mientras ((actual = leer_int(archivo)) == anterior):
              frecuencia += 1
         escribir(archComprimido, anterior)
         escribir(archComprimido, frecuencia)
    cerrar(archivo)
    cerrar(archComprimido)
```

Los resultados obtenidos luego de la compresión de cada archivo se resumen en la siguiente tabla y las codificaciones RLC para cada uno de los archivos puede consultarse en el siguiente link:

RLC	Argentina.txt	Hawai.txt	imagen.raw
Tamaño original (bytes)	2500	3033	196618
Tamaño comprimido (bytes)	9768	11256	10796
Tam.Orig / Tam. Compr	0.26	0.27	18.21
Tasa Compresión	-	-	18:1
Entropía (bits)	4.51	4.55	1.07
Longitud media	32.0	32.0	32.0
Rendimiento (%)	14.09	14.23	3.35
Redundancia (%)	85.91	85.77	96.65

Tabla III : Resultados de aplicar el algoritmo RLC a los archivos Argentina.txt, Hawai.txt y Imagen.raw

Comparación entre los distintos algoritmos de compresión

Argentina.txt	Huffman	Shannon - Fano	RLC
Tam.Orig / Tam. Compr	1.79	1.73	0.26
Tasa Compresión	2:1	2:1	-
Rendimiento	99.24	93.03	14.09
Redundancia	0.76	3.97	85.91

Tabla IV : Resultados de los distintos métodos de compresión, aplicados al archivo Argentina.txt

Hawai.txt	Huffman	Shannon - Fano	RLC
Tam.Orig / Tam. Compr	2.02	1.99	0.27
Tasa Compresión	2:1	2:1	-
Rendimiento	99.20	97.42	14.23
Redundancia	0.80	2.58	85.77

Tabla V : Resultados de los distintos métodos de compresión, aplicados al archivo Hawai.txt

En las tablas III y IV podemos observar los resultados obtenidos para los archivos de texto *Argentina.txt* y *Hawai.txt*. La tasa de compresión para estos archivos de texto fue de 2:1, siendo el rendimiento para ambos textos superior al 97.5%.

Con respecto al algoritmo RLC, podemos ver que no es un método adecuado para comprimir archivos de texto (el tamaño del archivo “comprimido” es mayor que el del archivo original) debido a la poca repetición de símbolos consecutivos que hay en los mismos.

Analizando estos resultados podemos obtener dos conclusiones. En primer lugar que tanto el método de Huffman como el de Shannon-Fano resultan muy adecuados para comprimir archivos de texto, presentando el método de Huffman una leve mejora en el rendimiento con respecto al de Shannon por generar códigos instantáneos óptimos (obteniéndose una longitud media de código menor). En segundo lugar, que no es conveniente utilizar el método RLC para comprimir archivos en donde no haya secuencias de símbolos repetidos debido a que el tamaño del archivo “comprimido” puede ser de mayor tamaño que el original.

imagen.raw	Huffman	Shannon - Fano	RLC
Tam.Orig / Tam. Compr	3.71	3.74	18.21
Tasa Compresión	4:1	4:1	18:1
Rendimiento	91.39	90.84	3.35
Redundancia	8.61	9.16	96.65

Tabla VI : Resultados de los distintos métodos de compresión, aplicados al archivo imagen.raw

En la tabla VI podemos observar los resultados obtenidos al aplicar los métodos de compresión en el archivo *Imagen.Raw*. La mejor tasa de compresión la obtuvimos utilizando el

algoritmo RLC (18:1) debido a que este algoritmo es muy adecuado para comprimir archivos con símbolos consecutivos repetidos, como suele ocurrir en imágenes monocromáticas. Las tasas de compresión utilizando Huffman y Shannon-Fano también fueron buenas, pero muy inferiores (4.5 veces menos) que las obtenidas con RLC.

El bajo rendimiento obtenido al utilizar el algoritmo RLC es debido a que la longitud media del código es muy elevada (32 bits). Debido a que el RLC no genera un código compacto, y en consecuencia no responde al teorema de Shannon, utilizar el rendimiento como medida de calidad no tiene sentido para este algoritmo.

La conclusión que obtenemos al analizar estos resultados es que cuanto mayor sea el número de símbolos consecutivos con el mismo valor, mayor es la tasa de compresión lograda por el algoritmo RLC.

Segunda parte: Canales de comunicación.

En esta segunda parte, dados 3 canales de comunicación se pide calcular la Equivocación, la información mútua y las propiedades de cada canal.

Dado un canal, representado por la siguiente matriz de probabilidades condicionales y la probabilidad de enviar el símbolo a_i, $P(a_i)$:

P(a)	Canal x	b1	b2	...	bs
P(a1)	a1	P(b1/a1)	P(b2/a1)		P(bs/a1)
P(a2)	a2	P(b1/a2)	P(b2/a2)		P(bs/a2)
	...	...	...		
P(ar)	ar	P(b1/ar)	P(b2/ar)		P(bs/ar)

Probabilidades de entrada: $P(a_i)$

Probabilidades a posteriori: $P(a_{i/b_j}) = \frac{P(b_j/a_i) \,.\, P(a_i)}{\sum_{i=1}^{r} P(b_j/a_i).\, P(a_i)}$

Probabilidad simultánea: $P(a_i,\ b_j) = P(b_j/a_i).\, P(a_i) = P(a_i/b_j).\, P(b_j)$

Entropía a priori : $H(A) = \sum_{A} P(a)\ .\ log(\frac{1}{P(a)})$

Entropía a posteriori de A, recibido b_j: $H(A/b_j) = \sum_{A} P(a_i/\ b_j).\ log(\frac{1}{P(a_i/b_j)})$

Equivocación: $H(A/B) = \sum_{A,\,B} P(a_i,\ b_j).\ log(\frac{1}{P(a_i/b_j)})$

Información mútua: $I(A;\ B) = H(A) - H(A/B)$

Canal 1

Matriz de probabilidades del canal:

P(a)	Canal 1	b1	b2	b3
0.25	a1	0.3	0.3	0.4
0.2	a2	0.1	0.4	0.5
0.2	a3	0.3	0.3	0.4
0.3	a4	0.3	0.4	0.3
0.05	a5	0.3	0.1	0.6

Resultados obtenidos:

- Probabilidades de salida P(B):

bi	b1	b2	b3	
P(bi)	0.2600	0.3400	0.4000	1.0

- Probabilidades a Posteriori P(ai/bj):

P(ai/bj)	a1	a2	a3	a4	a5	
b1	0.2885	0.0769	0.2308	0.3462	0.0577	1.0
b2	0.2206	0.2353	0.1765	0.3529	0.0147	1.0
b3	0.2500	0.2500	0.2000	0.2250	0.0750	1.0

- Probabilidades Simultáneas: P(ai,bj)

P(ai,bj)	b1	b2	b3
a1	0.0750	0.0750	0.1000
a2	0.0200	0.0800	0.1000
a3	0.0600	0.0600	0.0800
a4	0.0900	0.1200	0.0900
a5	0.0150	0.0050	0.0300

- Entropía a priori: **H(A)** = 2.1660 bits
- Entropías a posteriori
    - **H(A/b1)** = 2.0574 bits **H(A/b2)** = 2.0336 bits **H(A/b3)** = 2.2289 bits
      Luego, puede verse que la entropía, es decir, la incertidumbre sobre la entrada enviada, es máxima cuando se recibe b3 y mínima cuando se recibe b2.
      Además, como H(A/b3) > H(A) se puede concluir que hay más incertidumbre al observar la salida.
- Equivocación de A respecto de B (ruido): **H(A/B)** = 2.1179 bits
- Información mutua: **I(A,B)** = 0.0481 bits
- Entropía a salida: **H(B)** = 1.5632 bits

- Pérdida: **H(B/A)** = 1.5152 bits
- Información mutua simetría: **I(B,A) = I(A, B)** = 0.0481 bits

Canal 2

Matriz de probabilidades del canal:

P(a)	Canal 2	b1	b2	b3	b4
0.25	a1	0.2	0.3	0.2	0.3
0.3	a2	0.3	0.3	0.2	0.2
0.2	a3	0.3	0.2	0.2	0.3
0.25	a4	0.3	0.3	0.3	0.1

- Probabilidades de salida P(B):

bi	b1	b2	b3	b4	
P(bi)	0.2750	0.2800	0.2250	0.2200	1.0

- Probabilidades a Posteriori P(ai/bj):

P(ai/bj)	a1	a2	a3	a4	
b1	0.1818	0.3273	0.2182	0.2727	1.0
b2	0.2679	0.3214	0.1429	0.2679	1.0
b3	0.2222	0.2667	0.1778	0.3333	1.0
b4	0.3409	0.2727	0.2727	0.1136	1.0

- Probabilidades Simultáneas: P(ai,bj)

P(ai,bj)	b1	b2	b3	b4
a1	0.0500	0.0750	0.0500	0.0750
a2	0.0900	0.0900	0.0600	0.0600
a3	0.0600	0.0400	0.0400	0.0600
a4	0.0750	0.0750	0.0750	0.0250

- Entropía a priori **H(A)** = 1.9855 bits
- Entropías a posteriori
    - **H(A/b1)** = 1.9650 bits **H(A/b2)** = 1.9455 bits **H(A/b3)** = 1.9620 bits **H(A/b4)** = 1.9082 bits

Luego, puede verse que la entropía, es decir, la incertidumbre sobre la entrada enviada, es máxima cuando se recibe b1 y mínima cuando se recibe b4.

- Equivocación de A respecto de B: **H(A/B)** = 1.9464 bits
- Información mutua: **I(A,B)** = 0.0391 bits
- Entropía a salida **H(B)** = 1.9912 bits
- Pérdida **H(B/A)** = 1.9521 bits
- Información mutua simetría: **I(B,A) = I(A, B)** = 0.0391 bits

Canal 3

Matriz de probabilidades del canal:

P(a)	Canal 3	b1	b2	b3	b4
0.25	a1	0.2	0.3	0.2	0.3
0.2	a2	0.3	0.3	0.3	0.1
0.1	a3	0.2	0.2	0.3	0.3
0.3	a4	0.3	0.3	0.2	0.2
0.1	a5	0.2	0.3	0.3	0.2
0.05	a6	0.2	0.3	0.3	0.2

- Probabilidades de salida P(B):

bi	b1	b2	b3	b4	
P(bi)	0.2500	0.2900	0.2450	0.2150	1.0

- Probabilidades a Posteriori P(ai/bj):

P(ai/bj)	a1	a2	a3	a4	a5	a6	
b1	0.2000	0.2400	0.0800	0.3600	0.0800	0.0400	1.0
b2	0.2586	0.2069	0.0690	0.3103	0.1034	0.0517	1.0
b3	0.2041	0.2449	0.1224	0.2449	0.1224	0.0612	1.0
b4	0.3488	0.0930	0.1395	0.2791	0.0930	0.0465	1.0

- Probabilidades Simultáneas: P(ai,bj)

P(ai,bj)	b1	b2	b3	b4
a1	0.0500	0.0750	0.0500	0.0750
a2	0.0600	0.0600	0.0600	0.0200
a3	0.0200	0.0200	0.0300	0.0300

a4	0.0900	0.0900	0.0600	0.0600
a5	0.0200	0.0300	0.0300	0.0200
a6	0.0100	0.0150	0.0150	0.0100

- Entropía a priori **H(A)** = 2.3660 bits
- Entropías a posteriori
    - **H(A/b1)** = 2.2579 bits, **H(A/b2)** = 2.3244 bits, **H(A/b3)** = 2.4508 bits, **H(A/b4)** = 2.2836 bits

Luego, puede verse que la entropía, es decir, la incertidumbre sobre la entrada enviada, es máxima cuando se recibe b3 y mínima cuando se recibe b1.

- Equivocación de A respecto de B: **H(A/B)** = 2.3300 bits
- Información mutua: **I(A,B)** = 0.0360 bits
- Entropía a salida **H(B)** = 1.9918 bits
- Pérdida **H(B/A)** = 1.9559 bits
- Información mutua simetría: **I(B,A) = I(A, B)** = 0.0360 bits

Observando los resultados obtenidos para los 3 canales con respecto a la información mutua podemos decir que si bien es cierto que $I(A, B) > 0$, denotando que no hay pérdida de información por observar la salida, su valor cercano a cero nos hace sospechar que el canal es demasiado ruidoso, haciendo que los símbolos de entrada y salida sean prácticamente independientes.

Observando la equivocación podemos decir que dado que H(A/B) es la cantidad de información sobre A que no deja pasar el canal (perdida causada por el canal) y que es del mismo orden que H(A), podemos concluir que o bien los canales son muy ruidosos o el uso que se está haciendo de ellos no es adecuado (hay pérdida casi total de la información), siendo los símbolos de entrada y salida casi independientes.

Conclusiones

Con respecto a la primera parte pudimos verificar que los algoritmos más adecuados para comprimir archivos de texto fueron los de Hoffman y Shannon-Fano y el más adecuado para comprimir imágenes (con mucha repetición de datos) fue el algoritmo RLC.

Con respecto a la segunda parte, y teniendo en cuenta los valores de Entropias a priori, Información mutua y equivocación podemos concluir que los 3 canales resultaron muy ruidosos, presentando mucha pérdida de información.

Apéndice

Glosario

' '	Espacio en blanco
\n	Nueva línea
\r	Retorno de carro

Codificaciones de Huffman para Argentina.txt

Símbolo	Frecuencia	Probabilidad	Cant. información	Código Huffman
' '	369	0.150000000000000	2.736965594166206	110
a	215	0.087398373983740	3.516249750637142	000
e	204	0.082926829268293	3.592017258255106	1111
o	203	0.082520325203252	3.599106683041426	1110
r	175	0.071138211382114	3.813231488394274	1010
s	158	0.064227642276423	3.960661852049500	1000
n	136	0.055284552845528	4.176979758976263	0101
l	121	0.049186991869919	4.345579362952008	0011
i	101	0.041056910569106	4.606231117474808	10111
u	78	0.031707317073171	4.979040381364354	10010
d	77	0.031300813008130	4.997656059531701	01111
t	77	0.031300813008130	4.997656059531701	01110
\n	64	0.026016260162602	5.264442600226603	01001
c	44	0.017886178861789	5.805010981589304	101100
b	36	0.014634146341463	6.094517598784290	011011
m	35	0.014227642276423	6.135159583281637	011010
g	34	0.013821138211382	6.176979758976263	011001
v	31	0.012601626016260	6.310246289839727	001011
p	25	0.010162601626016	6.620586410451877	001001
,	18	0.007317073170732	7.094517598784290	1001101
.	16	0.006504065040650	7.264442600226603	0100011
C	16	0.006504065040650	7.264442600226603	0100010
S	15	0.006097560975610	7.357552004618085	0100000
f	14	0.005691056910569	7.457087678168998	0010100
A	13	0.005284552845528	7.564002882085511	0010001
h	12	0.004878048780488	7.679480099505446	10110111
z	12	0.004878048780488	7.679480099505446	10110101
L	11	0.004471544715447	7.805010981589305	10011111
y	11	0.004471544715447	7.805010981589305	10011110
í	11	0.004471544715447	7.805010981589305	10011101
P	9	0.003658536585366	8.094517598784290	10011000
Y	9	0.003658536585366	8.094517598784290	10011001
ó	9	0.003658536585366	8.094517598784290	01100011
:	8	0.003252032520325	8.264442600226602	01100000
q	7	0.002845528455285	8.457087678168998	00101011
!	6	0.002439024390244	8.679480099505447	00100001
D	6	0.002439024390244	8.679480099505447	00100000
E	6	0.002439024390244	8.679480099505447	101101101
j	6	0.002439024390244	8.679480099505447	101101000
¡	6	0.002439024390244	8.679480099505447	100111001
O	5	0.002032520325203	8.942514505339240	100111000
"	4	0.001626016260163	9.264442600226602	001010101
;	4	0.001626016260163	9.264442600226602	010000111
?	4	0.001626016260163	9.264442600226602	010000110
N	4	0.001626016260163	9.264442600226602	011000100
T	4	0.001626016260163	9.264442600226602	011000101
¿	4	0.001626016260163	9.264442600226602	011000010
é	4	0.001626016260163	9.264442600226602	010000101

M	3	0.001219512195122	9.679480099505446	1011011001
U	3	0.001219512195122	9.679480099505446	1011011000
á	3	0.001219512195122	9.679480099505446	001010100
ñ	3	0.001219512195122	9.679480099505446	1011010010
B	2	0.000813008130081	10.264442600226602	0100001000
I	2	0.000813008130081	10.264442600226602	10110100111
V	2	0.000813008130081	10.264442600226602	0100001001
F	1	0.000406504065041	11.264442600226602	01100001100
G	1	0.000406504065041	11.264442600226602	01100001101
J	1	0.000406504065041	11.264442600226602	01100001110
Q	1	0.000406504065041	11.264442600226602	01100001111
R	1	0.000406504065041	11.264442600226602	10110100110
60	2460	1.0		

Codificaciones de Huffman para Hawai.txt

Símbolo	Frecuencia	Probabilidad	Cant. información	Código Huffman
' '	541	0.191843971631206	2.381994663459341	00
a	367	0.130141843971631	2.941843194442944	100
o	226	0.080141843971631	3.641300484870969	1101
k	205	0.072695035460993	3.781999347780710	1011
i	187	0.066312056737589	3.914584987398519	1010
e	166	0.058865248226950	4.086440015939231	0110
n	128	0.045390070921986	4.461479447286155	0100
u	119	0.042198581560284	4.566661683978213	11110
h	118	0.041843971631206	4.578836397924315	11101
ʻ	112	0.039716312056738	4.654124525228552	11100
l	106	0.037588652482270	4.733558992722958	11000
ā	72	0.025531914893617	5.291554445843844	01011
m	70	0.024822695035461	5.332196430341190	01010
\n	62	0.021985815602837	5.507283136899281	111110
p	29	0.010283687943262	6.603498452158584	1100111
r	27	0.009574468085106	6.706591945122688	1100101
A	21	0.007446808510638	7.069162024507396	0111101
,	20	0.007092198581560	7.139551352398794	0111011
s	19	0.006737588652482	7.213551933842570	0111001
.	16	0.005673758865248	7.461479447286156	11111110
w	16	0.005673758865248	7.461479447286156	11111101
H	13	0.004609929078014	7.761039729145064	01111111
O	13	0.004609929078014	7.761039729145064	11001001
t	13	0.004609929078014	7.761039729145064	11001000
ū	13	0.004609929078014	7.761039729145064	01111110
K	11	0.003900709219858	8.002047828648859	01111001
C	10	0.003546099290780	8.139551352398794	01111000
:	8	0.002836879432624	8.461479447286157	01110000
I	7	0.002482269503546	8.654124525228552	110011001
M	7	0.002482269503546	8.654124525228552	110011010
c	7	0.002482269503546	8.654124525228552	111111000
ē	7	0.002482269503546	8.654124525228552	110011000
!	6	0.002127659574468	8.876516946565000	011111010
N	6	0.002127659574468	8.876516946565000	011111011
U	6	0.002127659574468	8.876516946565000	011111000
E	5	0.001773049645390	9.139551352398794	011101001
L	5	0.001773049645390	9.139551352398794	011101010
P	5	0.001773049645390	9.139551352398794	011101011

S	5	0.001773049645390	9.139551352398794	011101000
g	5	0.001773049645390	9.139551352398794	011100011
"	4	0.001418439716312	9.461479447286157	1100110111
;	4	0.001418439716312	9.461479447286157	1111111100
?	4	0.001418439716312	9.461479447286157	1111111110
d	4	0.001418439716312	9.461479447286157	1111111111
b	3	0.001063829787234	9.876516946565001	1100110110
ī	3	0.001063829787234	9.876516946565001	0111110011
ō	3	0.001063829787234	9.876516946565001	0111110010
B	2	0.000709219858156	10.461479447286157	11111100110
F	2	0.000709219858156	10.461479447286157	0111000101
y	2	0.000709219858156	10.461479447286157	11111100111
z	2	0.000709219858156	10.461479447286157	0111000100
J	1	0.000354609929078	11.461479447286155	111111110110
Q	1	0.000354609929078	11.461479447286155	111111001000
T	1	0.000354609929078	11.461479447286155	111111001001
V	1	0.000354609929078	11.461479447286155	111111110101
v	1	0.000354609929078	11.461479447286155	111111110111
á	1	0.000354609929078	11.461479447286155	111111001010
é	1	0.000354609929078	11.461479447286155	111111001011
í	1	0.000354609929078	11.461479447286155	111111110100
59	2820	1.0		

Codificaciones de Huffman para imagen.raw

Símbolo	Frecuencia	Probabilidad	Cant. información	Código Huffman
\n	65538	0.333326551994222	1.584991851232629	10
\r	65538	0.333326551994222	1.584991851232629	11
6	52854	0.268815673030953	1.895310839416330	01
0	5333	0.027123661109359	5.204304265436463	000
7	4570	0.023243039803070	5.427057428174025	0011
4	1884	0.009582032163891	6.705452628506673	00101
1	276	0.001403737195984	9.476511421341288	0010011
2	255	0.001296931104985	9.590682441260599	0010010
3	224	0.001139264970654	9.777680956061852	0010001
5	146	0.000742556632658	10.395211319239440	0010000
10	196618	1.0		

Codificación de Shannon-Fano para Argentina.txt

Símbolo	Frecuencia	Probabilidad	Cant. información	Código Shannon
' '	369	0.150000000000000	2.736965594166206	1111
a	215	0.087398373983740	3.516249750637142	1110
e	204	0.082926829268293	3.592017258255106	110
o	203	0.082520325203252	3.599106683041426	1011
r	175	0.071138211382114	3.813231488394274	1010
s	158	0.064227642276423	3.960661852049500	100
n	136	0.055284552845528	4.176979758976263	01111
l	121	0.049186991869919	4.345579362952008	01110
i	101	0.041056910569106	4.606231117474808	0110
u	78	0.031707317073171	4.979040381364354	01011
d	77	0.031300813008130	4.997656059531701	0100
t	77	0.031300813008130	4.997656059531701	01010
\n	64	0.026016260162602	5.264442600226603	001111

c	44	0.017886178861789	5.805010981589304	001110
b	36	0.014634146341463	6.094517598784290	00110
m	35	0.014227642276423	6.135159583281637	0010111
g	34	0.013821138211382	6.176979758976263	0010110
v	31	0.012601626016260	6.310246289839727	001010
p	25	0.010162601626016	6.620586410451877	001001
,	18	0.007317073170732	7.094517598784290	001000
.	16	0.006504065040650	7.264442600226603	00011110
C	16	0.006504065040650	7.264442600226603	00011111
S	15	0.006097560975610	7.357552004618085	0001110
f	14	0.005691056910569	7.457087678168998	0001101
A	13	0.005284552845528	7.564002882085511	0001100
h	12	0.004878048780488	7.679480099505446	00010110
z	12	0.004878048780488	7.679480099505446	00010111
L	11	0.004471544715447	7.805010981589305	0001000
y	11	0.004471544715447	7.805010981589305	0001001
í	11	0.004471544715447	7.805010981589305	0001010
P	9	0.003658536585366	8.094517598784290	00001101
Y	9	0.003658536585366	8.094517598784290	00001110
ó	9	0.003658536585366	8.094517598784290	00001111
:	8	0.003252032520325	8.264442600226602	00001100
q	7	0.002845528455285	8.457087678168998	000010111
!	6	0.002439024390244	8.679480099505447	000001111
D	6	0.002439024390244	8.679480099505447	00001000
E	6	0.002439024390244	8.679480099505447	00001001
j	6	0.002439024390244	8.679480099505447	00001010
¡	6	0.002439024390244	8.679480099505447	000010110
O	5	0.002032520325203	8.942514505339240	000001110
"	4	0.001626016260163	9.264442600226602	0000001110
;	4	0.001626016260163	9.264442600226602	0000001111
?	4	0.001626016260163	9.264442600226602	00000100
N	4	0.001626016260163	9.264442600226602	000001010
T	4	0.001626016260163	9.264442600226602	000001011
¿	4	0.001626016260163	9.264442600226602	000001100
é	4	0.001626016260163	9.264442600226602	000001101
M	3	0.001219512195122	9.679480099505446	0000000111
U	3	0.001219512195122	9.679480099505446	000000100
á	3	0.001219512195122	9.679480099505446	000000101
ñ	3	0.001219512195122	9.679480099505446	000000110
B	2	0.000813008130081	10.264442600226602	0000000011
I	2	0.000813008130081	10.264442600226602	000000010
V	2	0.000813008130081	10.264442600226602	0000000110
F	1	0.000406504065041	11.264442600226602	0000000000
G	1	0.000406504065041	11.264442600226602	00000000010
J	1	0.000406504065041	11.264442600226602	00000000011
Q	1	0.000406504065041	11.264442600226602	00000000100
R	1	0.000406504065041	11.264442600226602	00000000101
60	2460	1.0		

Codificación de Shannon-Fano para Hawai.txt

Símbolo	Frecuencia	Probabilidad	Cant. información	Código Shannon
' '	541	0.191843971631206	2.381994663459341	111
a	367	0.130141843971631	2.941843194442944	110
o	226	0.080141843971631	3.641300484870969	1011
k	205	0.072695035460993	3.781999347780710	1010

i	187	0.066312056737589	3.914584987398519	100
e	166	0.058865248226950	4.086440015939231	01111
n	128	0.045390070921986	4.461479447286155	01110
u	119	0.042198581560284	4.566661683978213	0110
h	118	0.041843971631206	4.578836397924315	01011
‘	112	0.039716312056738	4.654124525228552	01010
l	106	0.037588652482270	4.733558992722958	0100
ā	72	0.025531914893617	5.291554445843844	00111
m	70	0.024822695035461	5.332196430341190	00110
\n	62	0.021985815602837	5.507283136899281	001011
p	29	0.010283687943262	6.603498452158584	001010
r	27	0.009574468085106	6.706591945122688	001001
A	21	0.007446808510638	7.069162024507396	001000
,	20	0.007092198581560	7.139551352398794	0001111
s	19	0.006737588652482	7.213551933842570	0001110
.	16	0.005673758865248	7.461479447286156	0001100
w	16	0.005673758865248	7.461479447286156	0001101
H	13	0.004609929078014	7.761039729145064	0001001
O	13	0.004609929078014	7.761039729145064	0001010
t	13	0.004609929078014	7.761039729145064	00010110
ū	13	0.004609929078014	7.761039729145064	00010111
K	11	0.003900709219858	8.002047828648859	0001000
C	10	0.003546099290780	8.139551352398794	000011111
:	8	0.002836879432624	8.461479447286157	000011110
I	7	0.002482269503546	8.654124525228552	000010111
M	7	0.002482269503546	8.654124525228552	00001100
c	7	0.002482269503546	8.654124525228552	00001101
ē	7	0.002482269503546	8.654124525228552	00001110
!	6	0.002127659574468	8.876516946565000	00001001
N	6	0.002127659574468	8.876516946565000	00001010
U	6	0.002127659574468	8.876516946565000	000010110
E	5	0.001773049645390	9.139551352398794	000001100
L	5	0.001773049645390	9.139551352398794	000001101
P	5	0.001773049645390	9.139551352398794	000001110
S	5	0.001773049645390	9.139551352398794	000001111
g	5	0.001773049645390	9.139551352398794	00001000
"	4	0.001418439716312	9.461479447286157	0000001111
;	4	0.001418439716312	9.461479447286157	00000100
?	4	0.001418439716312	9.461479447286157	000001010
d	4	0.001418439716312	9.461479447286157	000001011
b	3	0.001063829787234	9.876516946565001	000000101
ī	3	0.001063829787234	9.876516946565001	000000110
ō	3	0.001063829787234	9.876516946565001	0000001110
B	2	0.000709219858156	10.461479447286157	0000000101
F	2	0.000709219858156	10.461479447286157	0000000110
y	2	0.000709219858156	10.461479447286157	0000000111
z	2	0.000709219858156	10.461479447286157	000000100
J	1	0.000354609929078	11.461479447286155	0000000000
Q	1	0.000354609929078	11.461479447286155	00000000010
T	1	0.000354609929078	11.461479447286155	00000000011
V	1	0.000354609929078	11.461479447286155	00000000100
v	1	0.000354609929078	11.461479447286155	00000000101
á	1	0.000354609929078	11.461479447286155	00000000110
é	1	0.000354609929078	11.461479447286155	00000000111
í	1	0.000354609929078	11.461479447286155	0000000100
59	2820	1.0		

Símbolo	Frecuencia	Probabilidad	Cant. información	Código Shannon
\n	65538	0.333326551994222	1.584991851232629	10
\r	65538	0.333326551994222	1.584991851232629	11
6	52854	0.268815673030953	1.895310839416330	01
0	5333	0.027123661109359	5.204304265436463	0011
7	4570	0.023243039803070	5.427057428174025	0010
4	1884	0.009582032163891	6.705452628506673	0001
1	276	0.001403737195984	9.476511421341288	000011
2	255	0.001296931104985	9.590682441260599	000010
3	224	0.001139264970654	9.777680956061852	000001
5	146	0.000742556632658	10.395211319239440	000000
--10	196618	1.0		