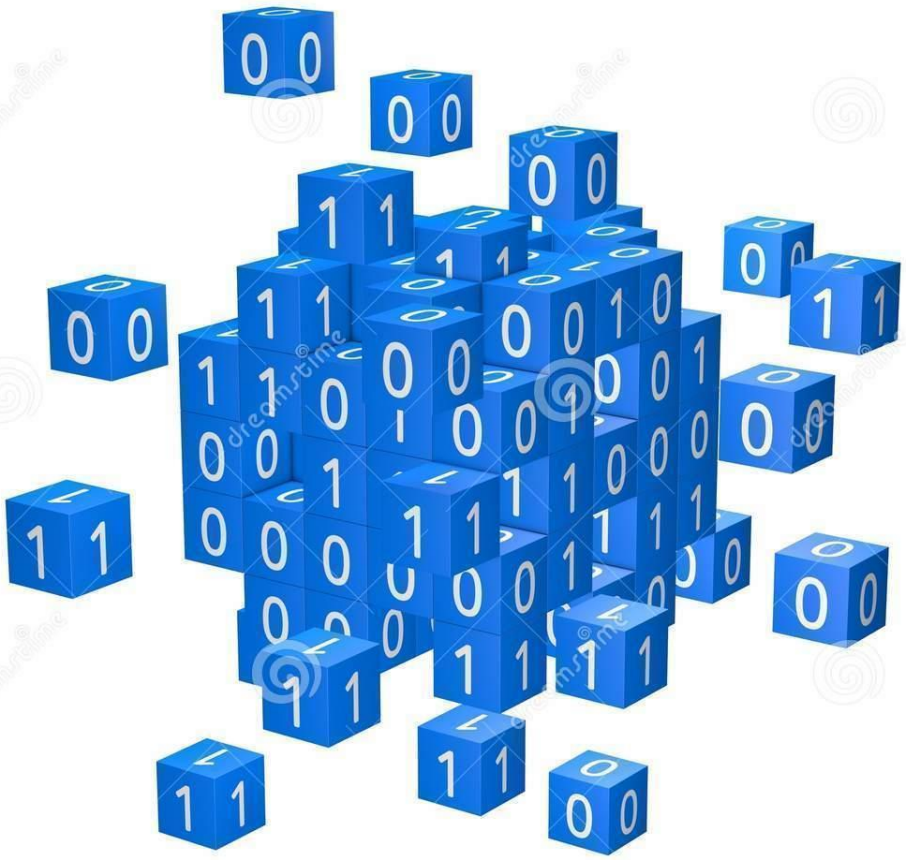
TRABAJO INTEGRADOR Nº1



Integrantes del grupo:

- Alan Gutiérrez, legajo: 13172, e-mail: alan.gutierrez@alumnos.fi.mdp.edu.ar
- Francisco Stimmler, legajo: 15409, e-mail: franciscostimmler@yahoo.com.ar
- María Josefina Oller, legajo: 11609, e-mail: josefinaoller19@gmail.com

Link del repositorio: https://github.com/JosefinaOller/TeoriaDeLaInformacion

Fecha de reentrega: 3 de noviembre de 2022

Índice

Resumen

En este informe se tratará sobre el análisis y resolución de diversos problemas que surgen al enfrentarse a la codificación y transmisión de la información. Se desarrollarán las técnicas y algoritmos utilizados para su resolución, y se harán conclusiones respecto a los resultados obtenidos. Se abordarán las temáticas de:

- Fuente de información de memoria, sea nula o no nula;
- Entropía;
- Inecuación de Kraft y de McMillan, longitud media, códigos compactos, rendimiento y redundancia;
- Codificación de símbolos de los códigos y sus propiedades.

Introducción

Antes de abordar los aspectos relevantes de las temáticas mencionadas anteriormente, se explicará la terminología necesaria para comprender el análisis que se realizará durante el informe.

Concepto de Información

Vamos a decir que un evento o un acontecimiento aporta información cuando éste se refiere a un hecho desconocido de antemano. Durante el desarrollo, se va a analizar que, desde el punto de vista probabilístico, cuanto menor sea la probabilidad de ocurrencia de un suceso, mayor será la información que éste brindará.

Concepto de fuente de Información

Llamaremos así a una fuente conformada de símbolos o sucesos, los cuales en su conjunto son denominados *"el alfabeto"* de la fuente, cada uno con una probabilidad de ocurrencia. Al transmitir información, en cada instante de tiempo, la fuente elige según su distribución de probabilidades de un símbolo para ser emitido y transmitir la información que éste posea.

Concepto de código

Un código es una aplicación entre un conjunto de símbolos llamado *alfabeto fuente* S = $\{s_1, s_2, ..., s_q\}$ y otro conjunto de símbolos llamado *alfabeto código* X = $\{x_1, x_2, ..., x_i\}$, de tal manera que, a cada símbolo del alfabeto fuente le corresponde una sucesión de símbolos diferente del alfabeto código.

Concepto de palabras código

Un código bloque es aquel que asigna cada uno de los símbolos del alfabeto fuente S a una secuencia fija de símbolos del alfabeto código X. Esas secuencias fijas (secuencias de x_i) reciben el nombre de palabras código. Nombraremos X_i a la palabra código que corresponde al símbolo s_i.

Concepto de longitud media

La longitud media de un código se define como la sumatoria de productos entre las probabilidades y longitudes de cada símbolo, es decir $L = \sum_i \quad P_i * l_i$. En nuestro caso,

es importante aclarar que no contemplaremos el cálculo de la longitud media ya que, por construcción se establecieron que las palabras código han de ser de 3, 5, 7 de longitud, es decir son palabras código de **longitud fija** (3,5,7).

Concepto de código compacto

Un código compacto es un código instantáneo cuya longitud media es mínima. Para que el código sea instantáneo ninguna palabra código debe ser prefijo de otra. El empleo de los códigos compactos conduce a una mayor eficacia en la transmisión de la información, en el sentido de ocupar el canal el menor tiempo posible para una cantidad de información dada.

Concepto de codificación de fuente

La codificación de fuente consiste en extraer toda la información redundante en la señal para reducir el número total de bits a transmitir. Esta extracción de información redundante es importante cuando queremos transmitir señales de audio o vídeo.

Desarrollo

Durante esta etapa, se analizará el comportamiento de la fuente proporcionada por la cátedra, obteniendo resultados teóricos y prácticos a analizar. Las tablas de todos los valores y los cálculos se mostrarán en la sección de apéndice.

Cálculo y análisis: Probabilidades condicionales

La probabilidad condicional se refiere a la probabilidad de que ocurra un evento A, sabiendo que también sucede otro evento B. Entonces para calcular las probabilidades condicionales, primero se genera un arreglo de datos para guardar todos los caracteres leídos de la fuente de información para luego generar una matriz de pasaje. Luego, se acumulan las apariciones de un símbolo con el otro en una matriz de apariciones. Una vez generada dicha matriz, se va a realizar la generación de una matriz de pasaje. Dicha generación se realiza a través del pseudocódigo:

Para la posición i hasta la cantidad de elementos de la matriz de apariciones {

Para la posición j hasta la cantidad de elementos de la matriz de apariciones {

Elemento [j][i] de la matriz de pasaje = elemento[j][i] de la matriz de apariciones/cantidad de veces que aparece el símbolo i en el archivo}}

De esa manera, se obtuvieron las probabilidades condicionales del alfabeto mostradas en la sección de apéndice **a)**.

Cálculo y análisis: Verificación de memoria nula

Antes de comenzar con la verificación de la fuente de memoria nula, se va a explicar el concepto de la misma para su mejor comprensión. En este tipo de fuentes de información, la probabilidad de ocurrencia de cada suceso, o aparición de un símbolo del alfabeto, es independiente de los símbolos previamente obtenidos, es decir, los símbolos son estadísticamente independientes unos a otros. Entonces, para saber si la fuente es de memoria nula o no, se calculó la matriz de transición de estados *(matrizPasaje[][] en el pseudocódigo)*. De ahí, se obtienen el mayor y el menor

de cada columna y se obtiene la diferencia. Lo mismo se hace para las diferencias y si el resultado de la diferencia mayor y la menor es menor a 0.02 se puede afirmar que es de memoria nula. Caso contrario, se afirma que es de memoria no nula. A continuación, se muestra el siguiente pseudocódigo.

```
Algoritmo memoriaNula(){
    Constantes: N = 3;
    Variables: Real dMayor = 0.0; Real dMenor = 1.0; Real matrizPasaje [][];
    Para (int j = 0; j < N; j++) hacer
        Real mayor = 0.0; Real menor = 1.0; Real diferencia = 0.0;
        Para (int i = 0; i <N; i++)
            Si (matrizPasaje[j][i]>mayor) entonces
                mayor=matrizPasaje[j][i];
            Fin Si
            Si (matrizPasaje[j][i]<menor) entonces
                menor=matrizPasaje[j][i];
            Fin Si
        Fin Para
        diferencia = mayor-menor;
        Si (diferencia>dMayor) entonces
            dMayor=diferencia;
        Fin Si
        Si (diferencia<dMenor) entonces
            dMenor=diferencia;
        Fin Si
    Fin Para
    Si (dMayor-dMenor<=0.02)
        Escribir: "Es una fuente de memoria nula";
    Sino
        Escribir: "Es una fuente de memoria no nula";
    Fin Si
Fin Algoritmo
```

De esa forma, se realizó la verificación de memoria nula y en nuestro caso, se verificó que es una fuente de **memoria nula**. Para comprobar si esta verificación es correcta, la suma de cada columna de la matriz tiene que ser menor o igual que 1. Los resultados de las sumas se muestran en la sección de apéndice **b)**.

Al analizar dichos resultados, podemos determinar que la verificación de memoria nula es correcta, ya que cada suma de las columnas es menor o igual a 1, condición necesaria para determinar si una fuente de memoria es nula. Como se comprobó que es una fuente de memoria nula, se va a realizar el cálculo de la entropía de la fuente inicial y la de orden 20. Para eso, vamos a mencionar los siguientes aspectos necesarios para calcular la entropía.

Cálculo y análisis: Cantidad de información

Para calcular la entropía, es necesario calcular la cantidad de información de cada símbolo. Cada símbolo del alfabeto de una fuente tiene cierta posibilidad de transmitir información. La cantidad de información de un símbolo depende exclusivamente de su probabilidad de aparición. A continuación, se muestra el pseudocódigo en donde se calculó la cantidad de información que transmite cada símbolo.

Para cada símbolo i { //Suponiendo que todas las P (ocurrencia del símbolo i)! = 0

Cantidad de información que transmite el símbolo i = -log3$(P(ocurrencia))$ *}*

Mediante dicho pseudocódigo, obtuvimos la cantidad de información que transmite cada símbolo, y para comprobar que estos valores obtenidos son correctos, vamos a calcular nuevamente la cantidad de información de manera teórica utilizando las probabilidades mostradas en la sección de apéndice **c)**. Por el análisis de los valores obtenidos, podemos determinar que la información transmitida por los símbolos es correcta y, como aspecto importante a mencionar, cuanto menor sea la probabilidad de ocurrencia de un símbolo, mayor será la información que éste transmite.

Este aspecto se puede evidenciar al hacer la comparación de los valores obtenidos para los símbolos "A", "B" y "C" respectivamente. Sin embargo, es importante aclarar que, en el caso que un símbolo tenga posibilidad de ocurrencia cero, la cantidad de información que transmite también será cero, ya que, se sabe anteriormente que nunca va a ser transmitido.

Cálculo y análisis: Entropía

Antes de realizar el cálculo, vamos a explicar el concepto breve de la entropía. La entropía es una medida de la cantidad de información que contienen los símbolos de una fuente. Tal medida se basa en la interpretación de la generación de un símbolo como el resultado de un experimento aleatorio, pudiéndose representar la cantidad de información del mismo como una función (logarítmica) de la inversa de la probabilidad de que tal resultado ocurra. Así, la información media de los símbolos dependerá de la distribución de probabilidades de los mismos, y será por tanto característica de la fuente que los genera. Por lo tanto, la forma de calcular la entropía de una fuente es la siguiente:

entropía = 0

Para cada símbolo i { //Suponiendo que todas las P (ocurrencia del símbolo i)! = 0

*entropía = entropía + cantidad de información que transmite el símbolo i * probabilidad del símbolo i }*

Después de realizar dicho cálculo, se obtuvo la entropía de la fuente inicial:

H(S)=0,9995 [binits/símbolo]

Según la sección de apéndice d), el valor teórico de la entropía de la fuente inicial es:

H(S)=0,9995 [binits/símbolo]

Para la extensión de orden 20, se pensó en 20 ciclos *for* anidados y se llegó al valor de **19,989294 [binits/símbolo]**. El valor teórico es **19,990620 [binits/símbolo]** y se llega al multiplicar 20×la entropía inicial del alfabeto

$H(S^n) = nH(S)$, con n=20.

Si bien al hacer la extensión, se obtuvo un resultado algo menor al valor teórico, igualmente está muy cerca de dicho valor. Por lo tanto, podemos determinar que los valores de la entropía son correctos.

A partir de ahora, tomando el archivo precedente, vamos a considerar que las cadenas representan palabras código, de 3, 5 y 7 caracteres.

Cálculo y análisis: Generación de palabras código de N caracteres

Teniendo en cuenta el concepto de palabras código mencionado anteriormente, se explicará la forma para generar palabras código de N caracteres. Se agarra el archivo y por cada cantidad N de caracteres se agrega a la lista de palabras.

```
procesamiento(cantCaracteres ){
        i = 0
        mientras(i sea menor que cantCaracteresArchivo){
                j=i
                mientras( j sea menor que  (i+cantCaracteres){
                        si( j es menor que  cantCaracteresArchivo )
                                palabra=palabra+datos[j]
                        j=j +1}
                si( largo de palabra es igual que cantCaracteres)
                        se añade palabra al arreglo de palabras
        i=i+cantCaracteres }
```

De esta manera, obtenemos un listado con todas las palabras código de N caracteres dentro del archivo, el cual se muestra en la consola, cabe destacar que si la cantidad de caracteres del archivo no es divisible por N los caracteres de sobra son ignorados.

Cálculo y análisis: Frecuencia de palabras código de N caracteres

Para calcular las frecuencias de palabras código de N caracteres, se genera un arreglo de datos para guardar todos los caracteres leídos del archivo proporcionado por la cátedra y se agrupan esos caracteres de a N cantidad en palabras y se calcula las

frecuencias de la siguiente manera: Se genera un *HashMap (arreglo con claves)* con los caracteres y su frecuencia (inicialmente vacío). Si los caracteres agrupados no están en el mapa se agrega con frecuencia 1. En caso contrario, se suma 1 a la frecuencia. Esto se repite hasta que no haya más caracteres agrupados de a N. De esa manera, se obtuvieron las frecuencias de las palabras de N caracteres, las cuales se muestran en la consola junto con sus palabras código.

Cálculo y análisis: Cantidad de información en palabras de N caracteres

En este caso, el pseudocódigo para calcular la cantidad de información que transmite una palabra de N caracteres es igual al de cálculo y análisis de cantidad de información explicado anteriormente, la única diferencia es que, en vez de aplicar la fórmula $-log3(P(ocurrencia))$, se debe utilizar la siguiente fórmula $-logN(P(ocurrencia))$. Los resultados de la cantidad de información que transmite cada palabra de N caracteres se muestran en la consola. Como se explicó anteriormente, nuevamente podemos decir que los resultados son correctos.

Cálculo y análisis: Entropía en palabras de N caracteres

Al igual que el caso anterior de la cantidad de información, el pseudocódigo para calcular la entropía es igual al de cálculo y análisis de la entropía explicado anteriormente. Mediante dicho pseudocódigo, se obtuvieron los resultados de la entropía total de palabras de N caracteres, los cuales son mostrados en la sección de apéndice **e)**. Como ya se analizó la comprobación de datos correctos anteriormente, nuevamente podemos decir que los resultados son correctos.

Cálculo y análisis: Inecuación de Kraft y de McMillan

Ambas inecuaciones utilizan la misma fórmula, pero en el caso de McMillan, si existe un código unívoco, entonces cumple con la inecuación. Esta condición es necesaria pero no suficiente. Kraft demostró la condición suficiente: si cumple con la inecuación, entonces el código es instantáneo porque los códigos instantáneos son unívocamente decodificables (es decir, que son los que hacen corresponder a toda secuencia de palabras de código, una única secuencia de símbolos).

$$\sum_{i=1}^{q} r^{-l_i} \leq 1$$

donde r es el número de símbolos diferentes que constituyen el alfabeto código y q la cantidad de palabras del alfabeto fuente.

El siguiente pseudocódigo de la inecuación de Kraft se muestra a continuación:

```
desigualdad_de_kraft = 0

Para cada palabra i  {

        desigualdad_de_kraft = desigualdad_de_kraft +
        (cantidad_de_simbolos_diferentes elevado a -(largo de la palabra i)

}

SI desigualdad_de_kraft <= 1 Entonces:
```

"Cumple la desigualdad de kraft, por lo tanto es instantáneo."

Sí no

"No cumple la desigualdad de kraft, no es instantáneo."

Para el caso de longitudes de 3 y 5 caracteres, como el valor de q se acerca a la cantidad esperada de combinaciones, la sumatoria de la inecuación de Kraft converge a 1. En cambio, para el caso de longitud de 7 caracteres, el valor de q no se acerca a la cantidad esperada de combinaciones. Por lo tanto, se esperaría que no converja a 1. Los valores obtenidos de la inecuación de Kraft se muestran en la sección de apéndice **f)**.

Cálculo y análisis: Código compacto

Teniendo en cuenta el concepto de código compacto explicado anteriormente, se va a explicar la manera para verificar si las palabras código de N caracteres son códigos compactos. Se calcula la cantidad ideal de combinaciones de palabras código de N caracteres en una fuente ternaria $\rightarrow 3^N$, si la cantidad de combinaciones generadas del archivo es igual a la cantidad ideal, se verifica que sean equiprobables. Si se cumplen dichas condiciones son códigos compactos, caso contrario no son códigos compactos.

Se esperaría que ningún código sea compacto debido a las combinaciones al azar del archivo y su distribución de probabilidades.

Cálculo y análisis: Redundancia y Rendimiento

Supongamos que L es la longitud media de un código r-ario, unívoco, de la fuente S. L no puede ser inferior a H_r (S). Según esto, se define η, rendimiento del código como $\eta = \frac{H_r(S)}{L}$. Y para la redundancia de un código se puede definir como 1 - η.

En los casos de 3 y 5 caracteres, como utilizan todas las combinaciones posibles del archivo, se esperaría que tengan un alto rendimiento y una baja redundancia. Y en el caso de 7 caracteres, como no utiliza todas las combinaciones posibles del archivo, se esperaría que no tenga un rendimiento tan alto como los casos anteriores y una redundancia mayor. Los valores obtenidos de rendimiento y redundancia se muestran en la sección de apéndice **g)**.

Cálculo y análisis: Codificación de Huffman

Para la codificación se decidió usar el método Huffman porque se necesitaba obtener códigos binarios para después comprimir un archivo. Para obtener los códigos de Huffman se construyó un árbol binario de nodos, a partir de una lista de nodos, cuyo tamaño depende del número de símbolos. Los nodos contienen dos campos, las claves, ya sean de 3, 5 o 7 caracteres, y la probabilidad. El algoritmo consiste en la creación de un árbol binario que tiene cada uno de los símbolos por hoja, y construido de tal forma que siguiéndolo desde la raíz a cada una de sus hojas se obtiene el código Huffman asociado a él.

1. Se crea una lista de árboles, uno por cada uno de los símbolos del alfabeto, consistiendo cada uno de los árboles en un nodo sin hijos, y etiquetado cada uno con su símbolo asociado y su frecuencia de aparición.

2. Se toman los dos árboles de menor frecuencia, y se unen creando un nuevo árbol. La etiqueta de la raíz será la suma de las frecuencias de las raíces de los dos árboles que se unen, y cada uno de estos árboles será un hijo del nuevo árbol. Esos dos árboles se eliminan de la lista. También se etiquetan las dos ramas del nuevo árbol: con un 0 la de la izquierda, y con un 1 la de la derecha.
3. Se repite el paso 2 hasta que solo quede un árbol.

El siguiente pseudocódigo donde se realiza el algoritmo Huffman:

```
Algoritmo Huffman
        Lista lista;//lista de árboles Lista list;//lista de probabilidades
        Entero cantCaracteresCodigo;//depende si es de 3, 5 o 7 caracteres
Llamar NuevaLista(lista)
        //Genero una lista con las probabilidades y la ordenó
        Llamar NuevaLista(list)
        Llamar OrdenarLista(list)
        Para cada elemento de list hacer
                Nodo nodo = Llamar NuevoNodo(list.clave, list.valor, nulo, nulo);
                Llamar AgregarNodo(Lista, nodo);
        Fin Para
        Mientras lista.tamano!=1 hacer
                Entero i = 0; Entero j = 1; Real min1=1; Real min2=1;
                Para k = 0 hasta lista.tamano hacer
                        Si lista.k.Probabilidad<min1 entonces
                                min2 = min1;
                                j = i;
                                min1 = lista.k.Probabilidad;
                                i = k;
                        Sino
                                Si lista.k.Probabilidad<min2 entonces
                                        min2 = lista.k.Probabilidad;
                                        j = k;
                                Fin Si
                        Fin Si
                Fin Para
```

nodo = Llamar NuevoNodo(lista.i.Clave+lista.j.Clave, lista.i.Probabilidad + lista.j.Probabilidad, lista.i, lista.j)

Llamar AgregarNodo(Lista, nodo);

Si i < j entonces

Llamar EliminarNodo(Lista, j);

Llamar EliminarNodo(Lista, i);

Sino

Llamar EliminarNodo(Lista, i);

Llamar EliminarNodo(Lista, j);

Fin Si

Fin Mientras

Llamar recorrido(lista.0);

Llamar codificación(cantCaracteresCodigo));

Fin Algoritmo

Los códigos codificados por el algoritmo de Huffman para cada caso de N caracteres se muestran en la consola.

Conclusiones

En este trabajo práctico, se realizaron cálculos de probabilidades condicionales, información y entropía de la fuente de información nula. También se generaron palabras código de 3, 5 y 7 caracteres, en las cuales se hicieron los cálculos de frecuencias, entropía, inecuaciones de Kraft y McMillan, análisis de código compacto, rendimiento, redundancia, y la codificación de Huffman. Para ello, se utilizaron los conceptos teóricos y prácticos vistos en la asignatura. Entre las conclusiones principales durante el desarrollo del informe se pueden mencionar:

- La cantidad de información que transmiten los símbolos de una fuente está ligada directamente a su probabilidad de aparición. Es decir, cuanto menor sea la probabilidad de ocurrencia de un símbolo, mayor será la información que éste transmite.

- La fuente de memoria es nula ya que, la suma de cada columna es menor o igual que 1, condición necesaria para determinar si es nula o no.

- En cuanto a la entropía de palabras código de N caracteres, concluimos que cuanto mayor sea la longitud, mayor será la entropía, esto se debe a cuanto sea mayor longitud, mayor será la cantidad de información.

- Con respecto al código instantáneo, en todos los casos, las palabras son todas de un mismo largo N, ninguna palabra del código coincide con el prefijo de otra, eso es una condición necesaria y suficiente para que un código sea instantáneo, por lo tanto, en todos los casos obtenemos un código instantáneo. Además, era

esperable que en los casos de 3 y 5 caracteres las sumatorias converjan a 1 ya que, utilizan todas las combinaciones del archivo, y en el caso de 7 caracteres se aleja de la unidad ya que, no utiliza todas las combinaciones posibles del archivo debido a que el archivo contiene 10.000 caracteres.

- Al ser códigos de longitud fija, solo serían compactos si todas sus palabras código fueran equiprobables. En el caso de la longitud fija de 7, no todas las combinaciones están en el archivo, por lo tanto, el código no puede ser compacto con los datos provistos, y en los casos de las longitudes fijas de 3 y de 5, cada combinación tiene valores distintos por ende no son equiprobables y los códigos no son compactos. Podemos concluir que cuando se nos da un código basado en combinaciones al azar es muy improbable obtener un código compacto.
- Con respecto al rendimiento y redundancia, podemos concluir que en los casos de 3 y 5 caracteres los rendimientos son cercanos al 1, es decir que son máximos, y sus redundancias son cercanos al 0. Quiere decir que en ambos casos aprovechan toda la información posible. No obstante, en el caso de 7 caracteres, el rendimiento se aleja de la unidad y su redundancia es mayor a las de 3 y 5 caracteres, quiere decir que debería mejorar el rendimiento para sacarle el mayor provecho a la información. Y además, concluimos que cuanto menor sea la longitud media, mayor será el rendimiento.
- Gracias a la codificación, es posible comprimir un archivo de gran tamaño en un archivo de menor tamaño, ya que al aplicar el algoritmo de Huffman, los símbolos más frecuentes tienen una longitud de código más corta que uno menos frecuente.

Apéndice

Primer Parte

a) Probabilidades condicionales

	A	B	C
A	0,324984346	0,326530612	0,307426598
B	0,326549781	0,332232893	0,340241796
C	0,348152786	0,341236495	0,352331606

b) Verificación de una fuente de memoria nula

Resultados

Suma de la columna 1: **0,99968**

Suma de la columna 2: **0,99999**

Suma de la columna 3: **1**

c) Valores de cantidad de información de cada símbolo

Resultados teóricos

I teórica (A) = **1,037** binits

I teórica (B) = **1,009** binits

I teórica (C) = **0,956** binits

Resultados prácticos

I experimental (A) = **1,039** binits

I experimental (B) = **1,000** binits

I experimental (C) = **0,962** binits

d)

Valores teóricos

Símbolo	Frecuencia	Probabilidad	Información	Entropía
A	3194	0,32	1,037	0,331218
B	3332	0,33	1,009	0,336199
C	3474	0,35	0,956	0,332114

La entropía inicial es de **0,999531** [binits/símbolo]
La entropía de orden 20 es de **19,99062** [binits/símbolo]

Valores experimentales

Símbolo	Frecuencia	Probabilidad	Información	Entropía
A	3194	0,3194	1,038866081	0,331814
B	3332	0,3332	1,000364169	0,333321
C	3474	0,3474	0,962376296	0,33433

La entropía inicial es de **0,9994647** [binits/símbolo]
La entropía de orden 20 es de **19,989294** [binits/símbolo]

Segunda Parte

e) Entropía

N caracteres	Entropía [binits/símbolo]
3	2,9952140494741206
5	4,947641829733079
7	6,250319662922674

f) Desigualdad de Kraft

N caracteres	Desigualdad de Kraft
3	1
5	1
7	0,481

g) Rendimiento y Redundancia

N caracteres	Rendimiento	Redundancia
3	0,9984	0,0016
5	0,9895	0,0105
7	0,8929	0,1071