# Trabajo Práctico Integrador N°1

**Integrantes:**

Sobenko Bautista

*Legajo 15669*

*bautipriano@gmail.com*

Rojas Agustin

*Legajo 15757*

*agusaar@gmail.com*

Matarazzo Tomas

*tmatarazzo03@gmail.com*

**Materia:** Teoría de la Información

**Fecha de Entrega:** 20/10/2022

**Github:**
https://github.com/TomasMatarazzo/TeoriaDeLaInformacion-Facultad

# Contenidos

# Resumen

El trabajo consiste en leer un archivo que simula una fuente de información de 3 símbolos (A, B y C). A partir de esto se analizarán las propiedades de los códigos y premisas estudiadas para verificar su cumplimiento. Algunos de estos son la entropía, fuente ergódica, fuente de memoria nula, códigos instantáneos, codificación de huffman, entre otros.

# Introducción

Para la primera parte del informe, partiendo de la fuente de información otorgada, se calculan las probabilidades condicionales de cada símbolo y en consecuencia de los resultados obtenidos se determina si la fuente es de memoria nula o no.

En función del tipo de fuente obtenida, por un lado, se genera una extensión de la misma y se calcula su entropía, por otro lado, se analiza si es una fuente ergódica, dependiendo de la veracidad del postulado se calculará el vector estacionario y la correspondiente entropía.

La segunda parte consiste en formar distintas palabras código uniendo una determinada cantidad de caracteres repitiendo este proceso para casos de 3, 5 y 7 caracteres. Para cada una de ellas se calcula la cantidad de información y la entropía de la fuente, se reconocen los distintos tipos de código y se verifican las inecuaciones de Kraft y de McMillan. También se evalúan los cálculos sobre rendimiento y redundancia. Por último, se codifican las palabras reconstruyendo el archivo según el algoritmo seleccionado (Huffman) y se deducen conclusiones.

# Desarrollo

## *Primera Parte*

### *Inciso a)*

Probabilidad condicional es la probabilidad de que ocurra un evento A, habiendo ocurrido otro evento B previamente, o bien extrapolado a la teoría de la información es la probabilidad de que se emita un símbolo, habiéndose emitido otro anteriormente. Si la fuente es de memoria nula, la probabilidad de que se emita un símbolo es independiente de los símbolos que se hayan emitido anteriormente. En cambio, en las fuentes de memoria no nula, la probabilidad de que se emita un símbolo depende de los N símbolos emitidos anteriormente. N depende de su orden.

Para el cálculo de las probabilidades condicionales, se almacenan las probabilidades dentro de una matriz. Una matriz de transición de estados o matriz de posibilidades, donde cada elemento contiene la probabilidad de que salga el símbolo definido por la fila habiendo salido el símbolo definido por la columna.

$$P_{j/i} = \ probabilidad\ de\ j\ habiendo\ salido\ i$$

A fin de obtener una matriz de posibilidades, se realiza una lectura del archivo de símbolos, símbolo por símbolo. En esta lectura se obtienen las frecuencias de aparición de un símbolo "X" de la fuente dependiendo de si el símbolo previo resultaba A, B o C. Se va contabilizando esta frecuencia en la posición Mij de acuerdo a la regla de índices descrita anteriormente. Finalmente, una vez realizada la lectura del archivo se divide a cada elemento por las apariciones totales, obteniendo así la matriz de probabilidades.

Una vez calculada la matriz de transición de estados podemos determinar si la fuente es de memoria nula o no. La función que trabaja en ello, se encarga de recorrer la matriz de probabilidades por columna, verificando que los elementos de la columna sean distintos entre sí. Si los elementos de la columna son todos iguales, significa que la probabilidad de que salga ese símbolo es independiente del resto de símbolos. Es importante tener en cuenta que al estar trabajando con decimales pueden existir variaciones en los cálculos. Para que esto no ocurra, se considera que la fuente va a ser de memoria no nula siempre y cuando los valores de las columnas sean iguales con una variación de ± 0,05.

De esta manera, se obtuvo la siguiente matriz de probabilidades:

$$M_{ij} = \begin{pmatrix} 0{,}274 & 0{,}320 & 0{,}405 \\ 0{,}259 & 0{,}323 & 0{,}419 \\ 0{,}273 & 0{,}304 & 0{,}423 \end{pmatrix}$$

Al analizar las columnas se ve que los valores de las columnas son casi idénticos y la variación es menor a 0.05, permitiendo concluir que la fuente es de memoria nula. Coincidiendo con el resultado obtenido mediante el código.

### *Inciso b)*

Extensión de orden n de S, $S^n$, siendo S una fuente de memoria nula con $q^n$ símbolos $\{\sigma 1, \sigma 2, \ldots, \sigma_{qn}\}$. Donde el símbolo σi se corresponde con una secuencia determinada de n símbolos de la fuente S.

La probabilidad de $\sigma i \;\; P(\sigma_i)$, es la probabilidad de la secuencia correspondiente, si $\sigma_i$ representa la secuencia:$(S_{i1}, S_{i2}, \ldots, S_{in})$ con $S_{ij}$, entonces:

$$P(\sigma_i \;\;) = \; Pi1 * Pi2 * Pi3 \ldots Pin$$

Usando esta definición se puede hallar la entropía de la fuente como:

$$H\,(S\,n\,) = \sum Sn\,P(\sigma i\,)log\;1\,P(\sigma i\,)$$

Pudiéndose demostrar:

$$H\,(S\,n\,) = n\;*\;H(S)$$

Se trabajó con una fuente de 3 símbolos, la cantidad de símbolos para una extensión de la fuente a orden 20 va a ser de $3^{20} = 3{,}48 * 10^9\ símbolos$.

En función de encontrar cada símbolo se creó una función recursiva que reconstruye cada símbolo de atrás hacia adelante. Se concatenan los símbolos iniciales cubriendo todas las posibles combinaciones de la siguiente manera:

```
Procedimiento ProbabilidadesOrdenKRecursivo(simbolos, pal,
                                            cantSimbolos, orden, prob, entropia){

Si(casoBase(k==0))
    -calculo la probabilidad dependiendo de la probabilidad de cada simbolo en pal
    -calculo el valor que aportaria a la entopia
    -acumulo
sino{
   para cada simbolo de simbolos:
           -creo newpal = pal + simbolo[i]
           -vuelvo a llamar con newpal y k-1
}
```

Para hallar la entropía de la fuente a medida que se iba obteniendo cada símbolo se calculaba su probabilidad y también la entropía para cada símbolo, luego se acumuló el valor de la entropía de cada uno de ellos para obtener la entropía de la fuente dando como resultado:

19,6549 unidades de orden 3.

Pudiendo verificar la validez del resultado ya que si realizamos la entropía de la fuente siendo las probabilidades el promedio de cada columna de $M_{ij}$, aplicamos:

$$H(S)=\sum_{S} P(S_i)\log\frac{1}{P(S_i)}$$

obteniendo así,

H(s) = 0.9846 Unidades de orden 3

para verificar aplicamos el teorema mencionado previamente,

H(s ^ 20) = 20 * 0.9846 = 19,692 U. de orden 3

siendo $19{,}692 \simeq 19{,}6549$ valor obtenido con el código. La diferencia se debe al error de truncamiento.

## *Segunda parte*

### *Inciso a)*

Para el cálculo de la cantidad de información y entropía es necesario obtener la probabilidad de los códigos a partir de su frecuencia y frecuencia total.

La información es un valor numérico expresado en una base deseada, se calcula a partir de un suceso E que tiene probabilidad asociada al suceso P(E). Cuando E tiene lugar, decimos que hemos recibido $I(E) = log(1/P(E))$ unidades de información. La elección de la base del logaritmo determina la unidad en la que se mide la información.

Si se tiene una fuente S de memoria nula, se define la entropía H(S) como la cantidad media de información media de la fuente S:

$$\sum_{s} P(s_i) * \log\left(\frac{1}{P(Si)}\right)$$

Se puede concluir que a medida que los códigos tienen mayor cantidad de caracteres, menor va a ser su probabilidad de aparición. Generando que la cantidad de información otorgada sea mayor. De la misma manera ocurre para la entropía, si menor es la probabilidad mayor va a ser la incertidumbre, por lo tanto será mayor la entropía.

| Caso | Información [U. de orden 3] | Entropía |
|---|---|---|
| 1 | 82,32 | 2.95 |
| 2 | 851.2 | 3.32 |
| 3 | 3650 | 3.5 |

### *Inciso b)*

A partir de los códigos obtenidos se puede deducir lo siguiente de sus propiedades.

Código bloque: Se tomaron secuencias fijas de caracteres para los códigos, por lo tanto, cumplen con la condición de ser código bloque debido a que estas secuencias fijas no varían con el tiempo.

Código No Singular: es necesario que las palabras no se repitan para que sea no singular, al analizar el documento contabilizamos las palabras código y calculamos su frecuencia por lo tanto no existe la posibilidad de su repetición

Código unívocamente decodificable: Un código bloque no singular en el que todas las palabras código tienen la misma longitud es un código unívocamente decodificable, todos nuestros códigos son de bloque y no singulares esto implica que también van a ser unívocamente decodificables.

Código instantáneo: Para que un código sea instantáneo la condición necesaria y suficiente es que ninguna palabra del código coincida con el prefijo de otra. Los códigos van a ser instantáneos ya que las palabras no se repiten y están son siempre de la misma longitud por lo que una palabra no va a poder ser prefijo de otra.

*Inciso c)*

La inecuación de Kraft es una medida cuantitativa que informa si el código que utilizamos puede llegar a ser instantáneo, sin embargo, no te asegura que el código lo sea. Como estamos trabajando con códigos instantáneos, va a ser necesario que cumplan la inecuación de Kraft, formula de la siguiente manera:

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

La inecuación de McMillan sostiene que, si existe un código unívoco, este va a cumplir la condición anterior.

La longitud media del código va a estar dada por la sumatoria de los símbolos, dada su probabilidad multiplicada por su respectiva longitud, en nuestro código como la cantidad de caracteres es fija. La longitud media del código coincide con la cantidad de caracteres por palabra.

Un código será compacto (respecto a S) si su longitud media es igual o menor que la longitud media de todos los códigos unívocos que pueden aplicarse a la misma fuente y el mismo alfabeto.

| Casos | Inecuación de Kraft/McMillan | Longitud Media | H(s) | Compacto |
|---|---|---|---|---|
| 1 | 1.0 | 3 | 2.94 | Si |
| 2 | 1.0 | 5 | 4.86 | Si |
| 3 | 0.46 | 7 | 6.21 | Si |

Analizando cada columna de la tabla se puede apreciar como en todos los casos se cumple la inecuación de Kraft, esto se debe a lo explicado en el *inciso a,* las palabras al mantener una cantidad fija de caracteres y no repetirse no da lugar a que sean prefijos de otras.

Código compacto es el código con menor longitud promedio para una fuente dada, los códigos instantáneos y fuente de memoria nula serán compactos si $Hr(S) \leq L$. En nuestros casos los códigos tienen la longitud media preestablecida y se cumple para cada uno de ellos que la entropía es menor a la longitud media por lo tanto son códigos compactos.

### *Inciso d)*

El rendimiento de un código, es decir que tan eficiente es, se calcula con la siguiente ecuación $Hr(S)/L$. Siendo Hr(S) la entropía de la fuente y L la longitud media del mismo.

La redundancia se encuentra definida como $1 - eficiencia$, a mayor redundancia implica que la información va a ser menor.

| Caso | Rendimiento | Redundancia |
|---|---|---|
| 1 | 0.98 | 0.02 |
| 2 | 0.97 | 0.03 |
| 3 | 0.87 | 0.13 |

Se puede apreciar cómo a medida que aumentamos la cantidad de caracteres, la longitud promedio es mayor en relación a la entropía. Para el primer caso la redundancia es casi nula, sin embargo, para el resto de los casos el desperdicio de información es cada vez mayor.

### *Inciso e)*

Para la codificación de los símbolos de los códigos y reconstrucción del archivo se decidió utilizar el **algoritmo de Huffman.**

El algoritmo consiste en la creación de un árbol binario que tiene cada uno de los símbolos alojados en los nodos hoja. La codificación se construye de forma tal que siguiendo el árbol desde la raíz a cada una de sus hojas se obtiene el código Huffman asociado.

El árbol fue construido con Nodos estructurados de la siguiente manera:

**Nodo:**

Frecuencia : Entero
Hijo a Izquierda: Nodo
Hijo a Derecha: Nodo

**Nodo hoja (Extiende de Nodo):**

Frecuencia: Entero
Palabra: String
Hijo a Izquierda: Nodo
Hijo a Derecha: Nodo

**Breve explicación del funcionamiento del algoritmo.**

Se crearán tantos nodos hijos como símbolos tengamos, etiquetando cada uno con su correspondiente símbolo asociado y su frecuencia de aparición.

Tomando los dos nodos de menor frecuencia, estos se unirán para crear un nuevo nodo. Este nodo será etiquetado con la suma de las frecuencias y tendrá como hijos a los nodos de menor frecuencia mencionados anteriormente. Además, se etiqueta cada rama del árbol con un "0" a la rama izquierda y un "1" a la rama derecha.

Este algoritmo se repite hasta que quede un único nodo que será el nodo raíz del árbol binario generado.

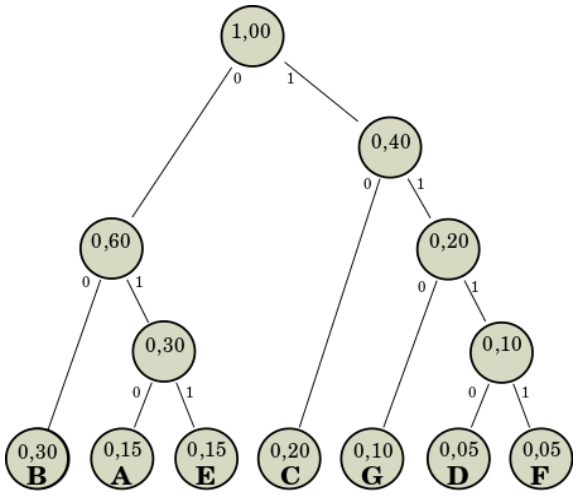
Ejemplo del árbol binario resultante luego de aplicado el algoritmo:

*Se puede observar que:*

> En los nodos hoja se encuentra el Símbolo y la frecuencia asociada.

> En los nodos restantes se tiene una etiqueta con la frecuencia resultante de la suma de los dos nodos hijos

> Las ramas se encuentran etiquetadas con "0" (izquierda) y "1" (derecha) permitiendo la codificación.



Para poder observar con mayor detalle el funcionamiento del algoritmo de Huffman se presenta el siguiente pseudocódigo:

```
Proceso algoritmoHuffman()
  //frecPalabra: mapeo de [palabra, frecuencia]
  pQ: ColaPrioridad
  para cada pal en frecPalabra:
    CrearNodoHoja(nodoHoja)
    nodoHoja.pal = pal
    nodoHoja.frec = pal.frecuencia
    Inserta(pQ , nodoHoja)

  mientras (tamaño pQ > 1):
    //El nodo resultante sera una fusion de las dos nodos con menor frecuencia
    CrearNodo(nodo)
    nodo.hijoIzq = Elimina(pQ)
    nodo.hijoDer = Elimina(pQ)
    nodo.frec = nodo.HijoIzq.frec + nodo.HijoDer.frec
    Inserta(pQ, nodo)

  //Etiqueto el arbol concantenando "1s" y "0s" de forma recursiva
  generaCodigoHuffman(raiz = Elimina(pQ) , "")
```

Para la realización de este inciso hemos reconstruido el archivo original con un archivo binario en donde se encontrará la codificación y la tabla generada por Huffman.
Se colocan al comienzo del binario las palabras código pertenecientes a cada uno de los símbolos. El orden de estas palabras código se corresponderá a la tabla incluida dentro del archivo que se observa sobre la carpeta de resultados. Este archivo que se utilizará en el proceso de decodificación contendrá únicamente la palabra y la longitud de la palabra código asociada, esto permitirá la decodificación de la tabla código que se encuentra dentro del archivo.

Al leer el archivo binario se podrá ir decodificando cada una de las palabras código ya que tienen la propiedad de ser instantáneas. Un código resulta Instantáneo si y sólo si ninguna de sus palabras código es prefijo de otra, esto permitirá encontrar y extraer cada palabra código del archivo binario analizando su prefijo, ya que solo una coincidirá y posteriormente se asociará a su palabra correspondiente.
(Ej: AAB -> 1101)

**Conclusiones Huffman:**

Luego de aplicado el algoritmo de Huffman se encontró un código compacto para la codificación de los símbolos.

Esto se puede afirmar ya que, para los símbolos de mayor frecuencia de aparición en

el mensaje, el algoritmo otorga palabras código de menor longitud y viceversa para aquellos símbolos de menor frecuencia. Esto permite poseer un código con mayor eficiencia, ya que se ahorrará tiempo de transmisión y almacenamiento. Además, el código será compacto.

Para llegar a una conclusión más determinista se calculó la longitud media para cada una de las codificaciones Huffman obteniendo así los siguientes resultados:

| Tamaño de palabra | Longitud media |
|---|---|
| 3 | 4.69 |
| 5 | 7.73 |
| 7 | 9.91 |

# Conclusión

El exceso de información de hoy en día es notorio, los sistemas informáticos tienen una potencia computacional que antes no se podía dimensionar. Esto permitió dejar de lado ciertas cuestiones técnicas a la hora de desarrollar algoritmos o software. Sin embargo, el ahorro de memoria es esencial, demostrando que es de suma importancia no dejar de lado el estudio e implementación de las distintas técnicas abarcadas en este informe.

Durante la elaboración del presente trabajo se pudo comprender, gracias a su aplicación, la importancia de los contenidos vistos en la parte teórica de la materia. El enfoque práctico permitió una mejor comprensión y consolidación de los conceptos trabajados. Atravesando conceptos como la información, la determinación del tipo de fuente, el cálculo de información y entropía, hasta el hallazgo de códigos compactos por medio del algoritmo de Huffman y la evaluación de propiedades de los códigos. Por lo tanto, se concluye que el trabajo ha sido exitoso.