El Teorema chino del residuo como un sistema de codificación y compresión

Grupo: Sigma

Julian David Pulido 25431778

Jonathan Alberto Ortiz 25431719

Jaime Andres Vargas 25431763

1. Problema

Muchos de los procesos utilizados para la transmisión de textos presentan un uso excesivo de recursos repercutiendo en el rendimiento de la trasmisión de datos, por ello se está en la constante búsqueda de métodos de codificación que puedan optimizar el proceso de transmisión como lo es la compresión de datos.

2. Objetivo

Realizar un algoritmo de compresión de textos que implemente un tipo de codificación propia y que consiga disminuir notablemente el tamaño del texto original, usando conceptos matemáticos de teoría de números, teoría de códigos y teoria de compresión.

3. Marco Teorico

3.1. Codificación Shannon-Fano (SF)

La codificación consiste en usar las probabilidades $P(x)$ de cada elemento x del conjunto de símbolos X y la longitud que contiene la asignación de bits. Así mismo se tiene en cuenta la magnitud de $P(x)$ como parte fundamental de la compresión. Teniendo esto en cuenta se divide en dos la lista de probabilidades de los conjuntos A y B, esto con el propósito de encontrar que la probabilidad entre ambas particiones sea la más cercana posible. Luego repetimos este proceso hasta que cada sub-división tenga cardinal 1, luego de esto asignamos bits de acuerdo a las particiones realizadas, así la partición superior siempre asignara el bit 0 al conjunto A y la partición inferior asignara el bit 1 al conjunto B .

3.2. Codificación Shannon-Fano-Elías (SFE)

Siendo X el alfabeto con símbolos $p_1, p_2, \ldots, p_n$ y dadas las probabilidades en base a la frecuencia de cada símbolo del alfabeto la codificacion del simbolo p_i consiste en aplicar la funcion:

$$F(x) = \sum_{x_i<x} P(x_i) + \frac{1}{2}P(x) \tag{1}$$

Sea $Ci = F(p_i)$ con $1 \leq i \leq n$ donde Ci representa la codificación del símbolo p_i, se debe convertir Ci en sistema binario y decidir su longitud de la cadena binaria por la formula:

$$L(x) = \left\lceil \log_2 \frac{1}{P(x)} \right\rceil + 1 \tag{2}$$

3.3. Codificación Aritmética (Cod. A.)

Es un tipo de codificación sin perdida, que involucra un modelo de probabilidades de aparición de cada carácter del alfabeto, representando cada símbolo como un numero entre 0 y 1.

Sea A un alfabeto con n símbolos $a_1, a_2 \ldots, a_n$ con $a_i \in A$ y $1 \leq i \leq n$ de tal forma que $P(a_i)$ representa la probabilidad de aparición del símbolo a_i en una cadena
El modelo de probabilidad esta definido por cada una de las probabilidades de aparición correspondientes a cada símbolo, de forma que

$$\sum_{i=1}^{n}(P_{a_i}) = 1 \tag{3}$$

y distribuidas entre el intervalo $[0, 1]$ de la siguiente forma

Símbolo	**Intervalo**
a_1	$[0, P(a_1)]$
a_2	$[0 + P(a_1), P(a_1)] + P(a_2)$
.	.
.	.
.	.
a_n	$[P(a_1) + P(a_2) + ... \ P(a_{n-1}), P(a_1) + P(a_2) + ... \ P(a_n)]$

dicho de otra forma

Simbolo	**Intervalo**
a_k	$[\sum_{i=1}^{k-1}(Pa_i), \sum_{i=1}^{k} P(a_i)]$

Sea C_k la Probabilidad Cumulativa definida como

$$C_k = \sum_{i=1}^{k} P(a_i) \tag{4}$$

Además sean definidos

$$L_k = C_1 + \sum_{i=2}^{k} C_i \prod_{j=1}^{i-1} P(a_j) + U \tag{5}$$

$$U_k = \prod_{j=1}^{k} P_j \tag{6}$$

La codificación de una cadena S con caracteres $a_{m_1} a_{m_2} a_{m_3} ... a_{m_k}$ debe seguir la siguiente secuencia [3]

Carácter	**Codificación**
a_{m_1}	$[C_{m_1-1}, C_{m_1}]$
$a_{m_1}a_{m_2}$	$[C_{m_1-1} + (C_{m_2-1})(P(a_{m_1})), C_{m_1-1} + C_{m_2-1}(P(a_{m_1}) + (P(a_{m_1})P(a_{m_2})]$
.	.
.	.
$a_{m_1}a_{m_2}...a_{m_k}$	$[L_k, L_k + U_k]$

La codificación de la cadena S estará dada por un numero c tal que $c \in [L_k, L_k + U_k]$ Es importante mencionar que la verdadera compresión ocurre al escoger un numero c lo mas corto posible de otra forma la compresión será mínima

3.4. American Standard Code for Information Interchange (ASCII)

Es un tipo de codificación estándar para el intercambio de información basado en el alfabeto occidental . La codificacion ASCII es altamente utilizada por su sencilla implementacion pero como desventaja representa cada caracter por un extenso codigo de 8 bits

3.5. Teorema Chino del Residuo (TCR)

Sea $N = n_1 \cdot n_2 \cdot .... \cdot n_k$ donde n_i y n_r con $1 \leq i \leq k, 1 \leq r \leq k$ y $i \neq r$ Entonces la función f

$$f : \mathbb{Z}/N \longrightarrow \mathbb{Z}/n_1 \times \mathbb{Z}/n_2 \times ... \times \mathbb{Z}/n_k \tag{7}$$

tal que

$$f([X]_N) = ([X]_{n_1}, \cdots, [X]_{n_k}) \tag{8}$$

donde $[X]_N$ representa $XmodN$
El teorema chino del residuo dice que f es un isomorfismo.

4. Desarrollo

Con el fin de lograr el objetivo del trabajo se plantea un algoritmo de codificación alterno a las codificaciones conocidas, este algoritmo utiliza como base uno de los teoremas mas importantes de la teoría de números como lo es el teorema chino del residuo.

4.1. Codificación por Teorema Chino del Residuo (Cod. TCR)

El método de codificación a proponer aprovecha el teorema chino del residuo 3.5 como un sistema de codificación con el fin de comprimir cadenas de caracteres.

Definición del Algoritmo de codificación

Sea$A$ un alfabeto de n símbolos tal que $A = a_1, a_2 \cdots a_n$ y $n = m1 \cdot m2 \cdots mk$ con m_i primos relativos. Sea $N = 1, 2, \cdots, n-1$ y $F(x)$ una función biyectiva de A en N. Si $F(a_i) = c_i$ con $1 \leq i \leq n$ entonces la codificación del símbolo a_i está dada por la función:

$$C(x) = ([c_i]_{m_1}, [c_i]_{m_2}, \cdots, [c_i]_{m_k}]) \tag{9}$$

Esta función da como resultado una k-tupla y por el teorema chino del residuo sabemos que la función $C(x)$ es una bisección, por lo tanto es un sistema unívocamente codificable o lo que es lo mismo, cada símbolo del alfabeto A tiene una única codificación y cada codificación tiene una sola decodificación.
Ahora para que la codificación del símbolo a_i tenga sentido es necesario pasar cada una de las componentes de la $k-tupla$ a sistema binario . Así la codificación final del símbolo a_i. Para codificar una cadena el sistema simplemente remplazara cada carácter por su respectiva codificación.

Ejemplo de implementación

Tomando como alfabeto $A = a_1, a_2 \cdot \cdot, a_n$ con n símbolos, donde $n = 102 = 2 \cdot 3 \cdot 17$ (producto de primos relativos), el símbolo a_i con $F(a_i) = 86$ tendrá como codificación $C(a_i) = ([86]_2, [86]_3, [86]_{17}) = (0, 2, 1) = (0_{base2}, 10_{base2}, 01_{base2})$, esto presenta una mejora con respecto a la codificación binaria directa ya que $86_{base2} = 1010110$ lo que disminuye tres bits.

También existen casos en donde no es muy ventajoso utilizar dicha codificación como con $F(X) = 4$ ya que $4_{base2} = 100$, pero con la codificación TCR será $(0, 1, 100)$ lo que aumentó dos bits.

Como se puede ver, de la función $F(x)$ depende notablemente la compresión ya que si el símbolo a_i es un carácter muy utilizado del alfabeto, entonces la función F(x) debería relacionar el símbolo a_i con un numero que tenga la menor codificación posible.

Comparación de algoritmos

Para medir el nivel de utilidad del algoritmo propuesto es imprescindible compararlo con otros algoritmos de codificación para ello decidimos tomar dos criterios de comparación fundamentales, la eficiencia o complejidad y el nivel de compresión.

4.2. Análisis de Complejidad

La complejidad de cada algoritmo esta calculada a partir del número de operaciones que llevan a cabo a medida que su entrada crece, es decir cuando la cantidad de datos a codificar aumenta.

SF es $O(x + x \log x)$

SFE es $O(\frac{x^2-3x+6}{2})$

Cod. A. es $O(x^2 - x)$

Cod. TCR es $O(6x)$

A partir de figura 1 se puede observar el siguiente comportamiento:

- Cod. A. es la menos eficiente con comportamiento polinomial de orden 2
- SFE es mas eficiente que Cod. A. con complejidad polinomial de orden 2
- Algoritmo TCR es mas eficiente que los dos anteriores con comportamiento lineal

- SF tiene la mejor eficiencia con comportamiento cuasi-lineal

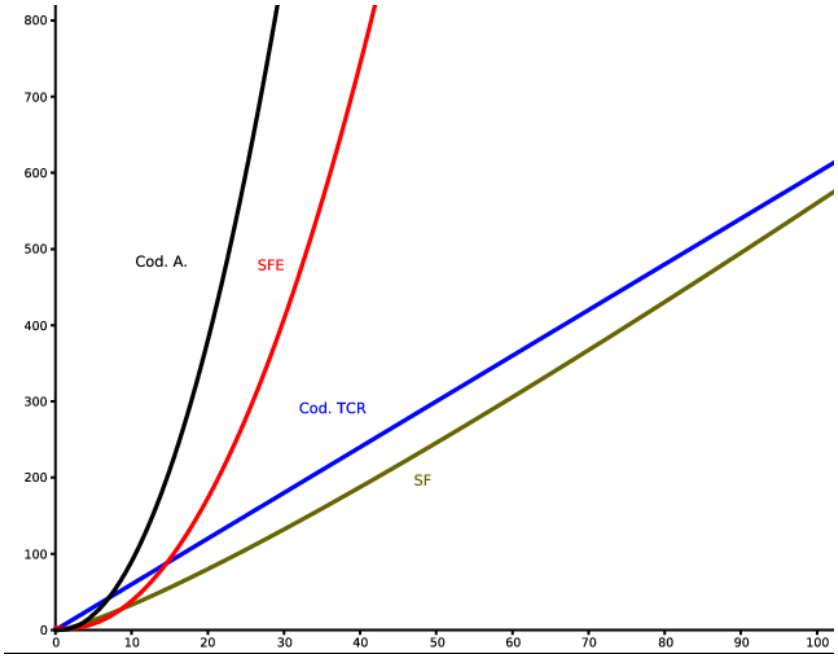

Figura 1: Grafico1

La grafica anterior representa la cantidad de tiempo computacional de acuerdo al tamaño de datos

4.3. Análisis de Compresión

Para comparar las codificaciones por nivel de compresión se realizo una serie de pruebas que median el porcentaje de compresión de cada una de las codificaciones con respecto a ASCII para un conjunto de textos de diferentes longitudes, obteniendo como resultado figura 2.
De acuerdo a los datos recaudados podemos concluir:

- El promedio de compresion de SF es de 4.44 % con respecto a ASCII, su mejor compresión se presenta con longitudes pequeñas.
- El promedio de compresión de SFE es de 4.85 % con respecto a ASCII , del diagrama 1 podemos decir que SFE comprime notablemente con longitudes pequeñas.

- El promedio de compresión de Codificación TCR es de 14.97 % con respecto a ASCII, la variación del porcentaje de compresión es menor al 3 %.
- El promedio de compresión de Cod. A. es de 17.6 % con respecto a ASCII, la variación del porcentaje de compresión es mas estable que el de Codificación TCR, aproximadamente 1 % de variación.

Nota: las pruebas fueron realizadas con los modelos mas básicos de cada algoritmo. Los datos arriba mencionados se pueden verificar en el Anexo1

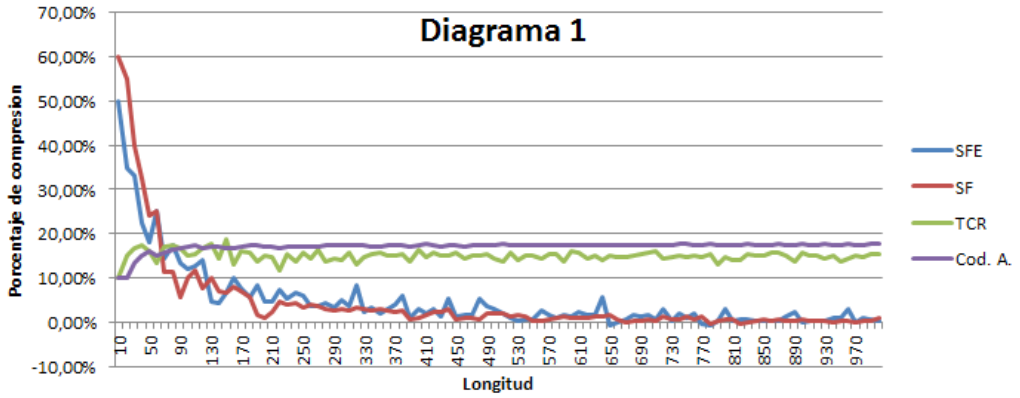

Figura 2: Diagrama1

5. Conclusiones

- De los análisis realizados ver que la Codificación TCR es una buena alternativa en cuanto a complejidad ya que el numero de operaciones que debe realizar el algoritmo es bajo con respecto a SFE y Cod. A.
- Teniendo en cuenta el porcentaje promedio de tasa de compresión de cada una de las codificaciones, la Codificación TCR demuestra tener un promedio sobresaliente y bastante estable.
- El TCR reúne las mejores características de cada codificación , como menor tiempo computacional y compresión.

6. Bibliografia

Referencias

[1] Niels Lauritzen - Concrete Abstract Algebra. From Numbers to Gröbner Bases. Cambridge University Press. 2003

[2] Standford University(June,1987).Aitthmetic coding for data compresssion. Tomado de http://www.stanford.edu/class/ee398a/handouts /papers/WittenACM87ArithmCoding.pdf.

[3] Arithmetic Coding(S.F). Tomado de http://researcher.nsc.gov.tw/public/cysu/Data/1339251471.pdf

[4] Reanatomy(S.F).Coding. Tomado de http://ezcodesample.com/reanatomy.html

[5] Coding(S.F).Tomado de http://www.di.univr.it/documenti/OccorrenzaIns/matdid/matdid014355.pdf.

[6] Data compression(S.f).Static Defined-Word-Schemes. Tomado de http://www.ics.uci.edu/ dan/pubs/DC-Sec3.html

[7] Courses(S.F).Shannon Fanno Elias.Tomado de http://www.icg.isy.liu.se/courses/tsbk08/lect5b.pdf

[8] Wikipedia(S.F).Shannon Fanno Elias coding.Tomado de http://en.wikipedia.org/ShannonEliasCoding.

[9] Coding (S.F) Tomado de http://www.ics.uci.edu/ dan/pubs/DC-Sec3.html

[10] Coding (S.F) Tomado de http://www-public.it-sudparis.eu/ uro/cours-pdf/poly.pdf

[11] Ascii (S.F) Tomado de http://es.wikipedia.org/wiki/ASCII

7. Anexos

- Anexo1, se anexa una tabla de calculo con los porcentajes de compresion de cada algoritmo.
- Anexo2, poster sobre el trabajo realizado.
- Codigos en java y mathlab de las pruebas realizadas.