



Instituto Politécnico Nacional  
Escuela Superior de Cómputo



Análisis de Algoritmos, Sem: 2021-1, 3CV1, Práctica 5, 8/12/20

## Práctica 5: Algoritmos greedy

Valle Martínez Luis Eduardo, Rivero Ronquillo Omar Imanol

*lvalle212@gmail.com, imanol.rivero7@gmail.com*

**Resumen:** En el actual documento se presenta el análisis de algoritmos que operan bajo la estrategia de búsqueda voraz o también llamados **Greedy**

**Palabras Clave:** Greedy, Kruskal, Huffman, Java

# Introducción

En la construcción de un algoritmo, existen variadas herramientas que nos permiten dotar de ciertas características específicas, que dependiendo del funcionamiento diseñado y tomando en consideración el ámbito en que se ejecutará, nos convendrán más o menos. Derivado de esto, nos es obligatorio conocer a fondo la naturaleza de la problemática a resolver, y nos es útil conocer para esto, las técnicas que mejor se ajusten a nuestras necesidades y las del algoritmo.

Los algoritmos voraces, o también llamados *Greedy*, son un tipo particular de algoritmos definidos por su manera de organizar los elementos y atacar al problema. En términos generales, estos algoritmos suelen tener un diseño menos complejo pues el horizonte del panorama que considera cuando se realizan decisiones, suele estar acotado localmente, y entre un conjunto de opciones escogerá aquella que ofrece un beneficio mayor inmediato.

Por esta misma característica, es de suma importancia comprender si el problema que se intenta resolver, requiere de alguna solución óptima pues el impacto de no resolver la más óptima es despreciable. Existiendo también problemas donde a partir de la elección de los elementos locales más óptimos nos lleva a una solución global aceptable. Este tipo de algoritmos cumplen su cometido sin la necesidad de realizar un esfuerzo mayor de diseño, que probablemente otra solución pudiera exigir.

A partir de su utilidad, se analiza un par de algoritmos conocidos por su eficiencia y capacidad de otorgar una solución óptima mediante la aplicación de este paradigma: El algoritmo para la compresión de texto de Huffman, y el algoritmo para la búsqueda de árboles recubridores mínimos.

# Conceptos Básicos

## Algoritmos Greedy

Los algoritmos voraces representan un paradigma algorítmico que contruye la solución paso a paso mediante la elección del siguiente elemento que ofrece el mayor beneficio inmediato. De forma que aquellos problemas donde la elección de locales óptimos lleva a una solución óptima global, solo la mejor opción para este tipo de algoritmos.

Existe cantidad de algoritmos que utiliza este paradigma, entre ellos uno de los más conocido es el problema de la *Mochila fraccionaria*. La estrategia óptima para escoger un elemento es aquel que tiene una relación valor/peso mayor. Esta estrategia lleva a la solución óptima global dada la característica de tomar fracciones de un elemento.

## Algoritmo de Kruskal

Algoritmo perteneciente a la teoría de grafos, es utilizado para encontrar un árbol recubridor mínimo en un grafo conexo y ponderado, de forma que busca un subconjunto de aristas que, formando un árbol, incluye todos los vértices y además asegura que el valor de todas las aristas que lo conforman suman el mínimo. En el caso donde el grafo no es conexo, se buscará un bosque expandido mínimo, lo que es igual a un árbol expandido mínimo para cada componente conexo.

El algoritmo trata a los grafos como bosques y cada nodo tiene un árbol individual. Un árbol conecta a otro, si y solo si, tiene el costo menor entre todas las opciones disponibles.

La razón inicial por la que los árboles de expansión mínima empezaron a estudiarse, se debió a la necesidad de realizar diseños de redes eléctricas de una manera que minimizara el costo del cableado.

El algoritmo de Kruskal es considerado dentro de la clasificación de algoritmos voraces, el ordenamiento y consumo de los elementos según su peso, rige que sean tomados aquellos desde el menor peso sin realizar comparaciones complejas de todas las posibles combinaciones.

Pasos del algoritmo:

1. Se eliminan todos los bucles y arcos paralelos del grafo, permaneciendo aquel de menor costo
2. Se acomodan todos los arcos en orden creciente de sus pesos
3. Se añaden los arcos que tienen el menor peso, considerando que todas las aristas seán contempladas en el árbol mínimo de expansión y que no se creen bucles.

## Codificación de Huffman

La compresión de archivos representa una gran ventaja en los momentos de enviar datos a través de la red o de simplemente almacenar archivos en alguna memoria, pues optimizas el espacio disponible, por lo tanto, un algoritmo que tenga altas tasas de compresión siempre será necesario.

También conocido como Algoritmo de Compresión de Código de Huffman, es un algoritmo utilizado para comprimir datos y forma la idea básica detrás de la compresión de archivos. Pertenecce a la rama de algoritmos voraces, por la forma en la que construye su codificación haciendo uso de una cola de prioridad ordenada de menor a mayor, de acuerdo a la frecuencia de repetición por caracter. Una de las grandes fortalezas de la codificación de Huffman es que previene cualquier

tipo de ambigüedad en el proceso de descompresión o decodificación, usando el concepto del código prefijo, esto es, un código asociado a un caracter no podrá estar presente como prefijo de cualquier otro código.

# Experimentación y Resultados

## Codificación de Huffman

### Análisis a priori

Consideremos el pseudocódigo descrito en la figura 1 para la codificación de Huffman.

```
Huffman(C: arreglo de caracteres sin repeticion)
  n = |C|
  Q = cola_prioridad(C)
  for i=1 to i<=n do
    x = Q.extraerMin()
    y = Q.extraerMin()
    z = nuevo_nodo
    z.izq = x
    z.der = y
    z.frec = x.frec + y.frec
    Q.insertar(z)
  return Q
```

Figura 1: Pseudocódigo del algoritmo de la codificación de Huffman

Podemos observar en la figura 1 que el algoritmo hace uso de una cola de prioridad, llamada Q para sacar los nodos con la menor frecuencia en cada iteración. Posteriormente se crea un nuevo nodo que tiene como hoja izquierda al nodo con la menor frecuencia de los dos y como hoja derecha al que tiene una mayor frecuencia. A continuación se calcula la complejidad del algoritmo por análisis por bloque.

$$\left. \begin{array}{l}
 \left. \begin{array}{l}
 n = C.size \\
 Q = \text{colaprioridad}(C)
 \end{array} \right\} \Theta(1) \\
 \left. \begin{array}{l}
 \text{for } i=1 \text{ to } i=n \text{ do} \\
 \left. \begin{array}{l}
 x = Q.\text{extraerMin}() \\
 y = Q.\text{extraerMin}() \\
 z = \text{nuevonodo} \\
 \left. \begin{array}{l}
 z.izq = x \\
 z.der = y
 \end{array} \right\} \Theta(\log(n)) \\
 z.frec = x.frec + y.frec
 \end{array} \right\} \Theta(\log(n)) \\
 Q.\text{insertar}(z) \\
 \text{return } Q
 \end{array} \right\} \Theta(n \log(n))
 \end{array} \right\} \Theta(n \log(n))$$

Finalmente, la complejidad del algoritmo de la codificación de Huffman esta dado por  $\Theta(n \log(n))$ . Es interesante observar que usar las operaciones de extraer e insertar son las que agregan la complejidad logarítmica  $\Theta(\log(n))$  debido a la forma en la que insertan y extraen elementos de su heap, siendo el heap también un árbol que describe la prioridad de sus elementos.

## Análisis a posteriori

Para la generación de nuestras gráficas, se realizó un conteo del número de operaciones realizadas en el algoritmo para cada arreglo de tamaño  $n$  de caracteres sin repetición. Se utilizaron archivos de texto que se trabajan para generar su codificación de Huffman, su árbol asociado y su decodificación final. Los archivos se construyeron considerando los caracteres que iban a contener quitando sus repeticiones. Por ejemplo:

Teniendo como entrada un archivo que contuviera la cadena "bbbbaaacfffttat", nuestro arreglo sin repeticiones sería [b,a,c,f,t] y la  $n$  asociada sería  $n=5$ .

En la figura 4 se presenta la gráfica generada por nuestra implementación.

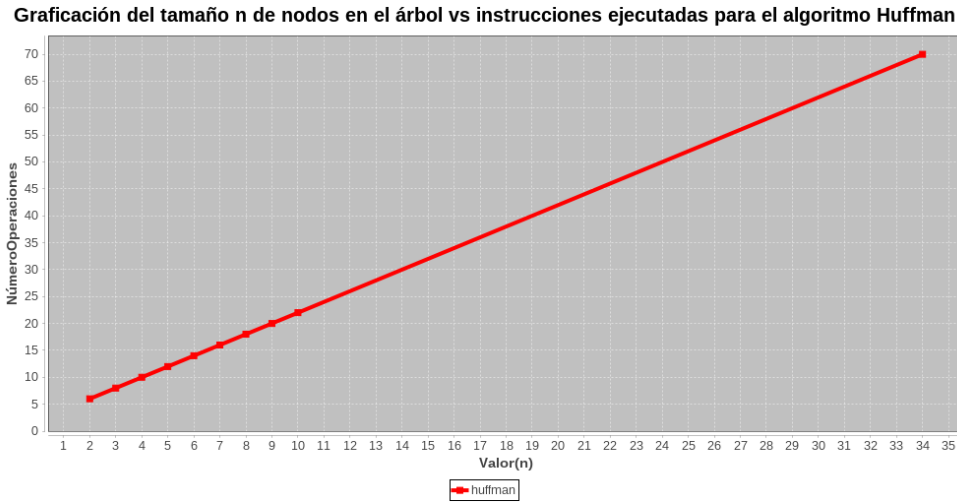


Figura 2: Representación gráfica de la complejidad temporal del algoritmo de la codificación de Huffman, mediante la evaluación del número de instrucciones realizadas.

Teniendo como puntos al siguiente arreglo, mostrado en la figura 3:

|              |               |
|--------------|---------------|
| P1( 2,6 )    | P5( 6,14 )    |
| P2( 3,8 )    | P6( 7,16 )    |
| P3( 4,10 )   | P7( 8,18 )    |
| P4( 5,12 )   | P8( 9,20 )    |
| P10( 10,22 ) | P11( 34, 70 ) |

Figura 3: Pares obtenidos de la evaluación del algoritmo de la codificación de Huffman

Como se puede notar a simple vista, el algoritmo parece tener una complejidad lineal dada por  $\Theta(n)$ . Sin embargo, la causa de este desvío con nuestro análisis a priori fue que no se hizo una implementación propia de una cola de prioridad con sus operaciones, si no, que se utilizó la clase PriorityQueue definida por el lenguaje Java. Por lo tanto no fue posible considerar las instrucciones ejecutadas durante cada operación de la cola de prioridad, causando que la complejidad aparente fuera lineal cuando debería ser  $\Theta(n \log(n))$ . A continuación en la figura , se representan los puntos en un graficador con la ecuación que los delimita  $y = 2x + 2$  y la ecuación por la que deberían de estar acotados  $y = x \log(x)$ .

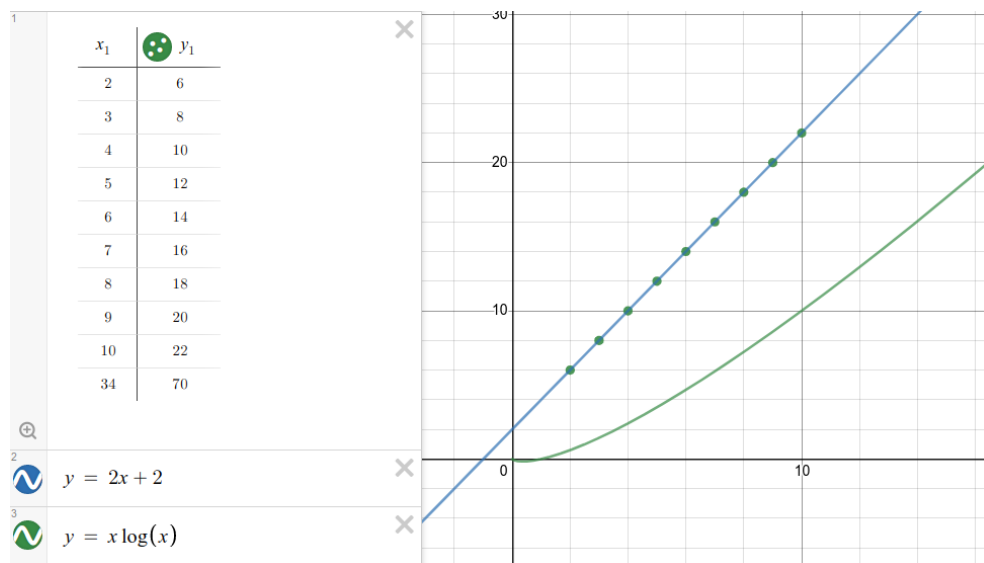


Figura 4: Representación gráfica de la complejidad temporal del algoritmo de la codificación de Huffman, comparado con la ecuación que los delimita en color azul  $y = 2x + 2$  y la ecuación por la que deberían de estar acotados  $y = x \log(x)$

## Proceso de compresión y descompresión

Para comprimir nuestro archivo de entrada sustituimos cada caracter por su representación de la configuración de Huffman que se guarda en un diccionario, y se forma una nueva cadena con una representación binaria de la salida comprimida.

Para descomprimir una cadena que esta en su representación de Huffman, necesitamos el árbol que contiene su representación. Se recorre el árbol y si se alcanza una de las hojas, se extrae el caracter que representa y se guarda en un string que tendrá como salida a la cadena decodificada.

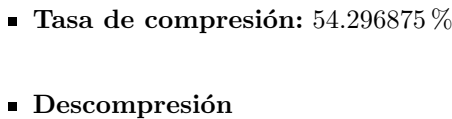
La tasa de compresión se calcula considerando que la cadena de entrada original esta codificada en utf-8, por lo que cada caracter representa un byte, es decir, 8 bits. En nuestra salida codificada, cada caracter representa un bit. Finalmente, mediante una regla de 3, siendo el 100 % el numero de bits de la cadena original, calculamos la tasa de compresión.

### Ejemplo 1

#### ■ Texto de origen

Era de noche. En la vasta sala silenciosa, tenuemente alumbrada por unas luces ocultas en los muros transparentes, los cuatro terrestres, sentados alrededor de una mesa de madera conversaban en voz baja, con los rostros juntos y pálidos. Hombres y mujeres yacían desordenadamente por el suelo. En los rincones oscuros había leves estremecimientos: hombres o mujeres solitarios que movían las manos. Cada media hora uno de los terrestres intentaba abrir la puerta de plata. -No hay nada que hacer. Estamos encerrados. -¿Creen realmente que somos locos, capitán? -No hay duda. Por eso no se entusiasmaron al vernos. Se limitaron a tolerar lo que entre ellos debe de ser un estado frecuente de psicosis. -Señaló las formas oscuras que yacían alrededor.-Paranoicos todos. ¡Qué bienvenida! -Una llamita se alzó y murió en los ojos del capitán.-Por un momento creí que nos recibían como merecíamos. Gritos, cantos y discursos. Todo estuvo muy bien, ¿no es cierto? Mientras duró.

#### ■ Compresión





- Codificación asignada

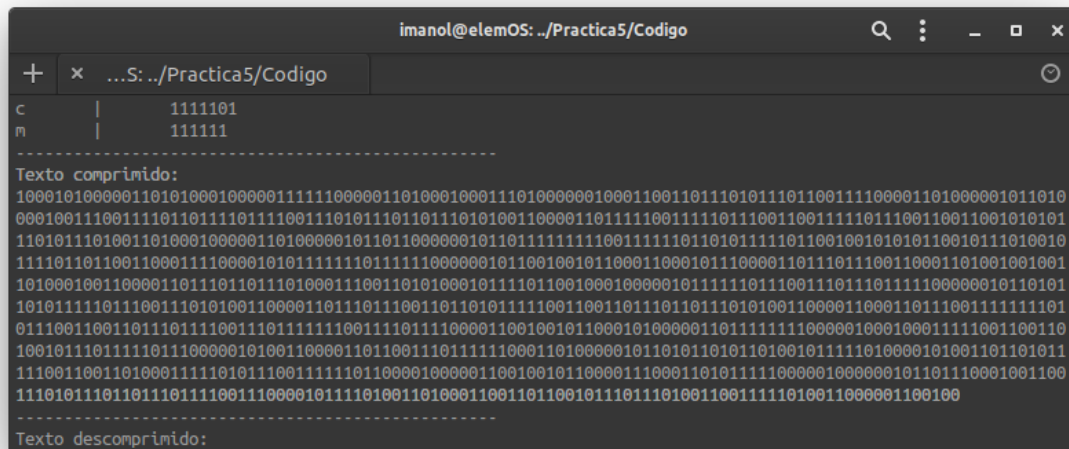
```
-----
Arbol de Huffman:
Char | Huffman
a | 000
p | 001000
y | 001001
j | 0010100
: | 001010100
N | 001010101
á | 00101011
b | 001011
q | 0011000
, | 0011001
i | 0011010
h | 0011011
c | 00111
e | 010
n | 0110
m | 01110
l | 01111
r | 1000
- | 1001000
E | 10010010
ó | 10010011
. | 100101
d | 10011
 | 101
ñ | 1100000000
i | 1100000001
z | 110000001
H | 1100000100
M | 1100000101
! | 1100000110
U | 1100000111
G | 1100001000
T | 1100001001
Q | 1100001010
é | 1100001011
f | 110000110
S | 110000111
v | 1100010
? | 110001100
¿ | 110001101
C | 110001110
P | 110001111
u | 11001
l | 11010
t | 11011
s | 1110
o | 1111
-----
Texto comprimido:
10010010100000010110011010101011011
```

## Ejemplo 2

### ■ Texto de origen

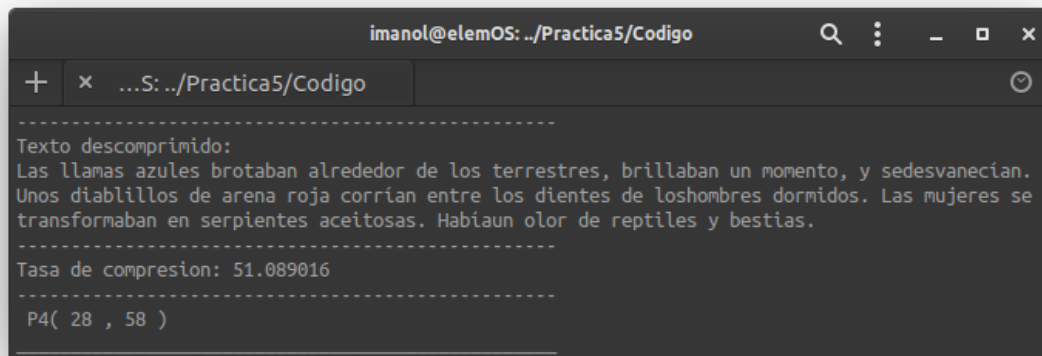
Las llamas azules brotaban alrededor de los terrestres, brillaban un momento, y se desvanecían. Unos diablillos de arena roja corrían entre los dientes de los hombres dormidos. Las mujeres se transformaban en serpientes aceitosas. Había un olor de reptiles y bestias.

### ■ Compresión



```
imanol@elemOS: ~/Practica5/Codigo
+ x ...S: ~/Practica5/Codigo
c | 1111101
m | 111111
-----
Texto comprimido:
1000101000001101010001000001111110000011010001000111010000001000110011011101011101100111100001101000001011010
00010011100111101101111001110101110110111010100110000011011111001111101110011001111011100110011001010101
11010111010011010001000001101000001011011000001011011111110011111011010111101100100101010110010111010010
1111011011001100011110000101011111101111100000101100100101100011000101110000110111011100110001101001001001
1010001001100001101110110111010001110011010100010111101100100010000010111101110011101111100000010110101
101011111011100111010100110000110111011100110110101111100110011011101101110101001100001100011011100111111101
01110011001101110111100111011111100111101111000011001001011000101000001101111111000001000100011111001100110
1001011101111011100000101001100001101100111011111100011010000010110101101010100101111010000101001101101011
111001100110100011111010111001111110110000100000110010010110000110001101011111000001000000101101110001001100
11101011101101110111100111000010111101001101000110011011001011101110100110011111010011000001100100
```

- Tasa de compresión: 51.089016 %
- Descompresión

A terminal window with a dark background and light text. The title bar reads 'imanol@elemOS: ../Practica5/Codigo'. The window contains the following text:

```
-----  
Texto descomprimido:  
Las llamas azules brotaban alrededor de los terrestres, brillaban un momento, y se desvanecían.  
Unos diablillos de arena roja corrian entre los dientes de los hombres dormidos. Las mujeres se  
transformaban en serpientes aceitosas. Había un olor de reptiles y bestias.  
-----  
Tasa de compresion: 51.089016  
-----  
P4( 28 , 58 )
```

- Codificación asignada

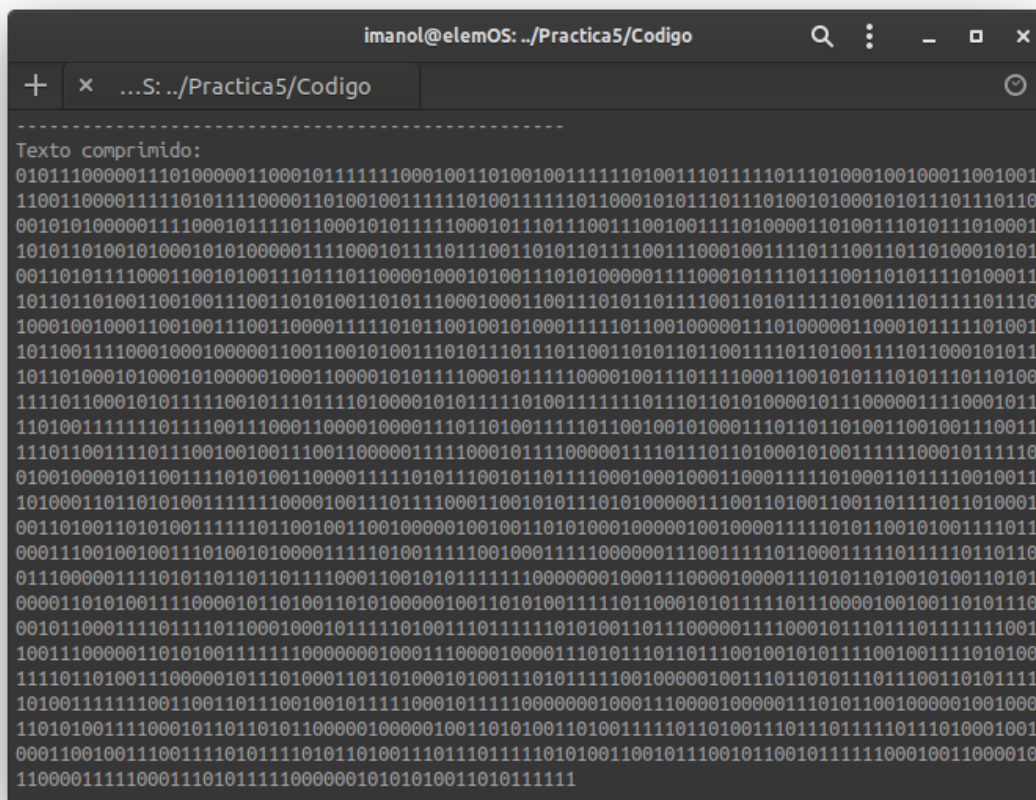
```
-----
Arbol de Huffman:
Char | Huffman
a | 000
s | 001
l | 0100
n | 0101
e | 011
u | 100000
p | 1000010
f | 10000110
H | 10000111
j | 1000100
L | 1000101
U | 10001100
h | 10001101
z | 10001110
v | 10001111
. | 100100
, | 1001010
y | 1001011
i | 10011
o | 101
b | 1100
d | 11010
r | 11011
t | 1110
i | 111100
c | 1111101
m | 111111
-----
Texto comprimido:
10001010000011010100010000011111100
000110100010001101000001000100011
```

### Ejemplo 3

#### ■ Texto de origen

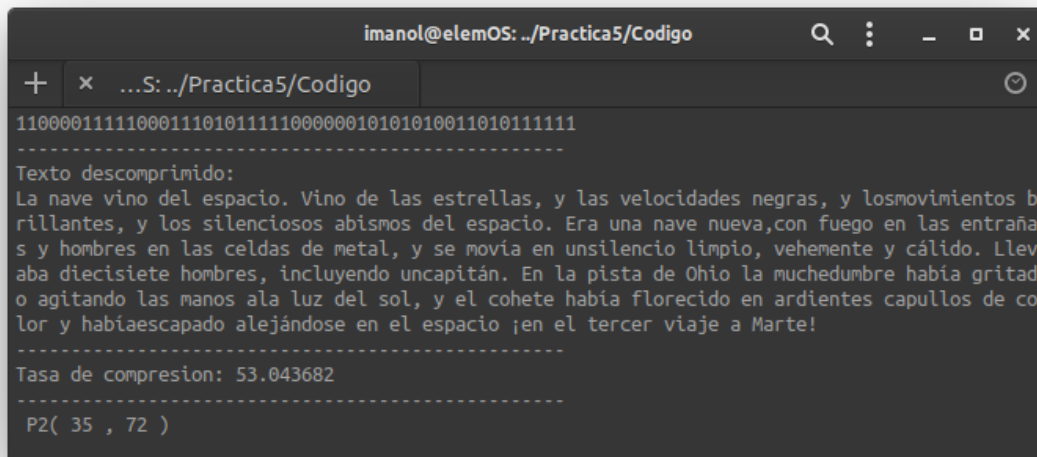
Qué demencia más hermosa. Metal, caucho, gravitadores, comida, ropa, combustible, armas, escaleras, tuercas, cucharas. He comprobado que en su nave hay diez mil artículos distintos. Nunca había visto tal complejidad. Hay hasta sombras debajo de las literas y debajo de todo. ¡Qué poder de concentración! Y todo, no importan cuándo o cómo se pruebe, tiene olor, solidez, gusto, sonido. Permítame que lo abrace.-El psiquiatra abrazó al capitán.- Consignaré todo esto en lo que será mi mejor monografía. El mes que viene hablaré en la Academia Marciana. Mírese. Ha cambiado usted hasta el color de sus ojos, del amarillo al azul, y la tez de morena a sonrosada. ¡Y su ropa, y sus manos de cinco dedos en vez de seis! ¡Metamorfosis biológica a través del desequilibrio psicológico! Y sus tres amigos...

#### ■ Compresión



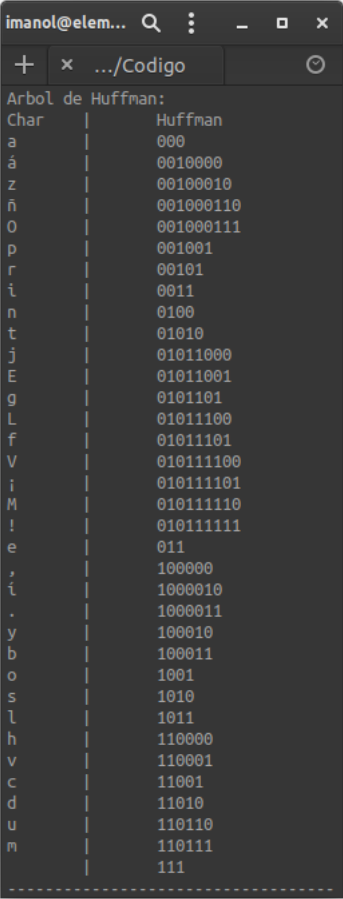
```
imanol@elemOS: ~/Practica5/Codigo
+ x ...S: ~/Practica5/Codigo
-----
Texto comprimido:
010111000001110100000100010111111000100110100100111111010011101111011101000100100011001001
11001100001111101011110000110100100111111010011111011000101011101110100101000101011101110110
00101010000011110001011110110001010111100010111011100111001001111010000110100111010111010001
101011010010100010101000001111000101111011100110101111001110001001111011100110110100010101
001101011110001100101001110111011000010001010011101010000011110001011110111001101011110100011
10110110100110010011100110101001101011100010001100111010111100110101111010011110111101110
100010010001100100111001100001111101011001001010001111101100100000111010000011000101111101001
101100111100010001000001100110010100111010111011101100110101101100111101101001111011000101011
101101000101000101000001000110000101011110001011111000010011101111000110010101110101110110100
1111011000101011111001011101111010000101011110100111111011101101010000101110000011110001011
110100111111101111001110001100001000011101101001111101100100101000111011011010011001001110011
11101100111101110010010011100110000011111000101111000001111011101101000101001111110001011110
0100100001011001111010100110000111110101110010110111100010001000111110100011011110010011
10100011011010100111111100001001110111100011001010111010100000111001101001100110111011010001
001101001101010011111101100100110010000010010011010100010000010010000111110101100101001111011
00011100100100111010010100001111101001111100100011111000001110011111011000111110111110110110
01110000011110101101101111000110010101111110000000100011100001000011101011010010100110101
00001101010011110000101101001101010000010011010100111110110001010111101110000100100110101110
0010110001111011110110001000101111101001101111110101001101110000011110001011101110111111001
10011100000110101001111110000000100011100001000011101011101110010010101111001001111010100
11110110100111100000101110100011011010001010011101011110010000010011101101011110011010111
101001111111001100110111001001011111000101111100000001000111000010000011101011001000001001000
110101001111000101101101110000010000010011010100110100111110110100111011110111110111010001001
00011001001110011110101111010110100111011111011110100110010111001011111100010011000010
110000111110001110101111100000010101010011010111111000100110000101111100010011000010
```

- Tasa de compresión: 56.04378 %
- Descompresión



```
imanol@elemOS: ../Practica5/Codigo
+ x ...S: ../Practica5/Codigo
110000111110001110101111100000010101010011010111111
-----
Texto descomprimido:
La nave vino del espacio. Vino de las estrellas, y las velocidades negras, y los movimientos brillantes, y los silenciosos abismos del espacio. Era una nave nueva, con fuego en las entrañas y hombres en las celdas de metal, y se movía en un silencio limpio, vehemente y cálido. Llevaba diecisiete hombres, incluyendo un capitán. En la pista de Ohio la muchedumbre había gritado agitando las manos a la luz del sol, y el cohete había florecido en ardientes capullos de color y había escapado alejándose en el espacio ¡en el tercer viaje a Marte!
-----
Tasa de compresion: 53.043682
-----
P2( 35 , 72 )
```

- Codificación asignada



A terminal window titled 'imanol@elem...' with a tab labeled '.../Codigo'. The window displays the output of a Huffman tree construction process. The text 'Arbol de Huffman:' is followed by a table mapping characters to their binary Huffman codes. The window has standard Linux terminal window controls (minimize, maximize, close) and a search icon.

| Char | Huffman   |
|------|-----------|
| a    | 000       |
| á    | 0010000   |
| z    | 00100010  |
| ñ    | 001000110 |
| o    | 001000111 |
| p    | 001001    |
| r    | 00101     |
| i    | 0011      |
| n    | 0100      |
| t    | 01010     |
| j    | 01011000  |
| E    | 01011001  |
| g    | 0101101   |
| L    | 01011100  |
| f    | 01011101  |
| V    | 010111100 |
| i    | 010111101 |
| M    | 010111110 |
| !    | 010111111 |
| e    | 011       |
| ,    | 100000    |
| í    | 1000010   |
| .    | 1000011   |
| y    | 100010    |
| b    | 100011    |
| o    | 1001      |
| s    | 1010      |
| l    | 1011      |
| h    | 110000    |
| v    | 110001    |
| c    | 11001     |
| d    | 11010     |
| u    | 110110    |
| m    | 110111    |
|      | 111       |

### Ejemplo 4

- Texto de origen

Quería ir a Marte en el cohete. Bajó a la pista en las primeras horas de la mañana y a través de los alambres les dijo a gritos a los hombres uniformados que quería ir a Marte. Les dijo que pagaba impuestos, que se llamaba Pritchard y que tenía el derecho de ir a Marte. ¿No había nacido allí mismo en Ohio? ¿No era un buen ciudadano? Entonces, ¿por qué no podía ir a Marte? Los amenazó con los puños y les dijo que quería irse de la Tierra; todas las gentes con sentido común querían irse de la Tierra. Antes que pasaran dos años iba a estallar una gran guerra atómica, y él no quería estar en la Tierra en ese entonces. Él y otros miles como él, todos los que tuvieran un poco de sentido común, se irían a Marte. Ya lo iban a 32 ver. Escaparían de las guerras, la censura, el estatismo, el servicio militar, el control gubernamental de esto o aquello, del arte y de la ciencia. ¡Que se quedaran otros! Les ofrecía la mano derecha, el corazón, la cabeza, por la oportunidad de ir a Marte. ¿Qué había que hacer, qué había que firmar, a quién había que conocer para embarcar en un cohete?

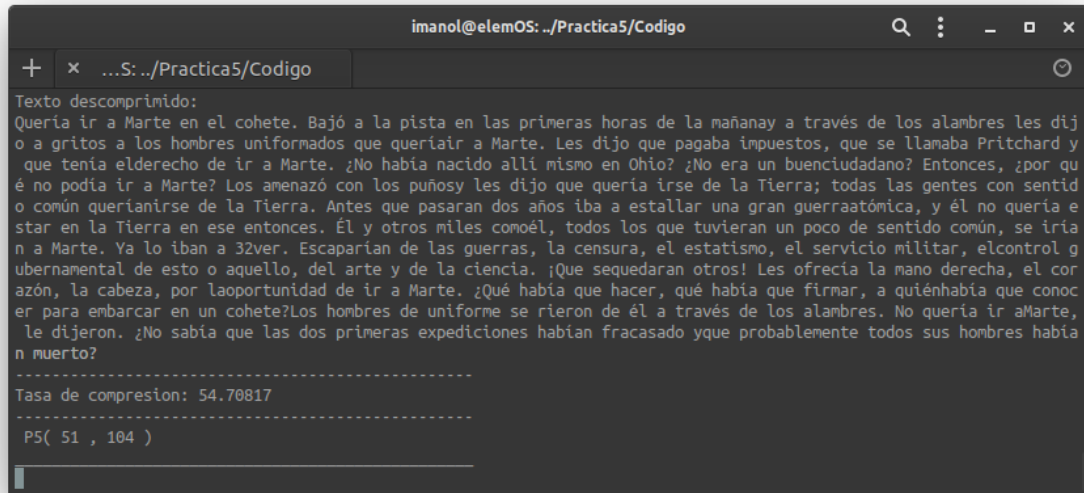
Los hombres de uniforme se rieron de él a través de los alambres. No quería ir a Marte, le dijeron. ¿No sabía que las dos primeras expediciones habían fracasado y que probablemente todos sus hombres habían muerto?

- **Compresión**

[illegible]



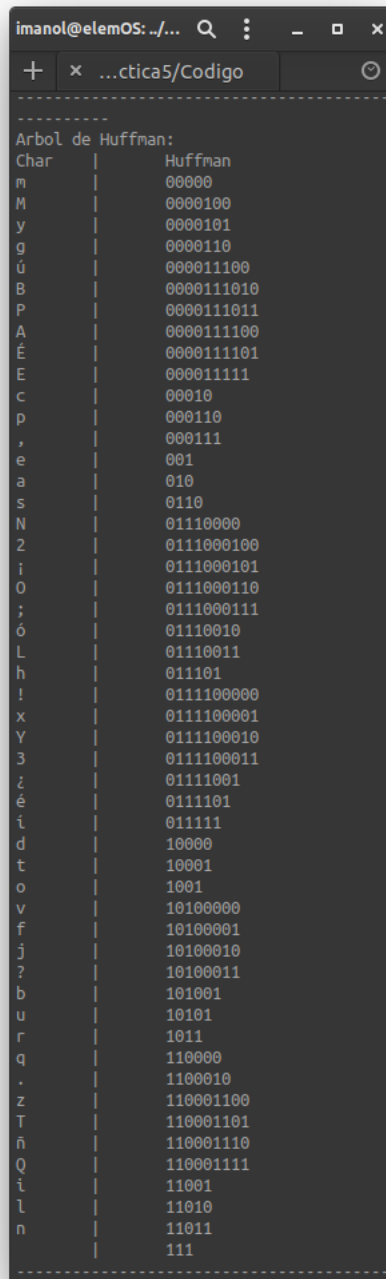
- Tasa de compresión: 54.70817 %
- Descompresión



The screenshot shows a terminal window titled "imanol@elemOS: ../Practica5/Codigo". The terminal output displays a paragraph of text, followed by the compression rate "Tasa de compresion: 54.70817" and a prompt "P5( 51 , 104 )".

```
imanol@elemOS: ../Practica5/Codigo
+ x ...S: ../Practica5/Codigo
Texto descomprimido:
Quería ir a Marte en el cohete. Bajó a la pista en las primeras horas de la mañana y a través de los alambres les dijo o a gritos a los hombres uniformados que querían ir a Marte. Les dijo que pagaba impuestos, que se llamaba Pritchard y que tenía el derecho de ir a Marte. ¿No había nacido allí mismo en Ohio? ¿No era un buenc Ciudadano? Entonces, ¿por qué no podía ir a Marte? Los amenazó con los puños y les dijo que quería irse de la Tierra; todas las gentes con sentido común querían irse de la Tierra. Antes que pasaran dos años iba a estallar una gran guerra atómica, y él no quería estar en la Tierra en ese entonces. Él y otros miles como él, todos los que tuvieran un poco de sentido común, se irían a Marte. Ya lo iban a saber. Escapaban de las guerras, la censura, el estatismo, el servicio militar, el control gubernamental de esto o aquello, del arte y de la ciencia. ¿Que se quedaran otros! Les ofrecía la mano derecha, el corazón, la cabeza, por la oportunidad de ir a Marte. ¿Qué había que hacer, qué había que firmar, a quién había que conocer para embarcar en un cohete? Los hombres de uniforme se rieron de él a través de los alambres. No quería ir a Marte, le dijeron. ¿No sabía que las dos primeras expediciones habían fracasado y que probablemente todos sus hombres habían muerto?
-----
Tasa de compresion: 54.70817
-----
P5( 51 , 104 )
```

- Codificación asignada



Arbol de Huffman:

| Char | Huffman    |
|------|------------|
| m    | 00000      |
| M    | 0000100    |
| y    | 0000101    |
| g    | 0000110    |
| ú    | 000011100  |
| B    | 0000111010 |
| P    | 0000111011 |
| A    | 0000111100 |
| É    | 0000111101 |
| E    | 000011111  |
| c    | 00010      |
| p    | 000110     |
| ,    | 000111     |
| e    | 001        |
| a    | 010        |
| s    | 0110       |
| N    | 01110000   |
| 2    | 0111000100 |
| i    | 0111000101 |
| O    | 0111000110 |
| ;    | 0111000111 |
| ó    | 01110010   |
| L    | 01110011   |
| h    | 011101     |
| !    | 0111100000 |
| x    | 0111100001 |
| Y    | 0111100010 |
| 3    | 0111100011 |
| ¿    | 01111001   |
| é    | 0111101    |
| í    | 011111     |
| d    | 10000      |
| t    | 10001      |
| o    | 1001       |
| v    | 10100000   |
| f    | 10100001   |
| j    | 10100010   |
| ?    | 10100011   |
| b    | 101001     |
| u    | 10101      |
| r    | 1011       |
| q    | 110000     |
| .    | 1100010    |
| z    | 110001100  |
| T    | 110001101  |
| ñ    | 110001110  |
| Q    | 110001111  |
| ì    | 11001      |
| l    | 11010      |
| n    | 11011      |
|      | 111        |

## Ejemplo 5

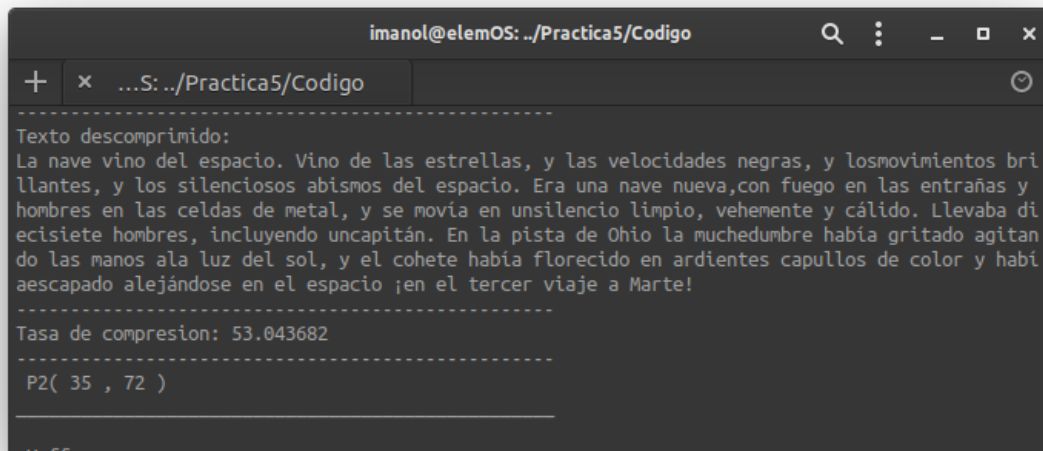
### ■ Texto de origen

La nave vino del espacio. Vino de las estrellas, y las velocidades negras, y los movimientos brillantes, y los silenciosos abismos del espacio. Era una nave nueva, con fuego en las entrañas y hombres en las celdas de metal, y se movía en un silencio limpio, vehemente y cálido. Llevaba diecisiete hombres, incluyendo un capitán. En la pista de Ohio la muchedumbre había gritado agitando las manos a la luz del sol, y el cohete había florecido en ardientes capullos de color y había escapado alejándose en el espacio jen el tercer viaje a Marte!

### ■ Compresión

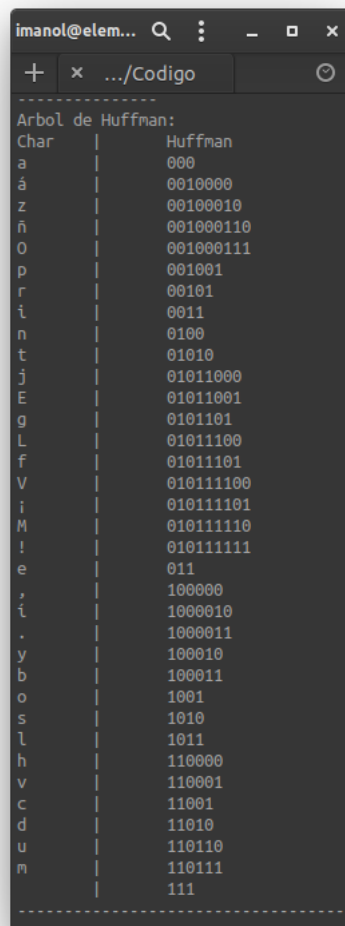
```
imanol@elemOS: ~/Practica5/Codigo
+ x ...S: ../Practica5/Codigo
-----
Texto comprimido:
010111000001101000001000101111110001001101001001111110100111011110111010001001000110010011100110000011110101
111000011010010011111101001111101100010101110111010010100010101110111011000101010000011110001011110110001010111
11000101110111001110010011110100001010011101011101000110101101001010001010100000111100010111101110011010101111
00111000100111101110011011000101010111000110010100111011101100001000101001110101000001111000101111011110
0110101111010001110110101001100100111001101010011010111000100011001110101101111001101011110100111011110111010
00100100011001001110011000011110101100100101000111101100100000111010000011000101111101001101100111100010001000
00110011001001110101110111011001101011101101001111011010011110110001010111011010001010001010000010001100001010111
100010111110000100111101111000110010101110110110110011110110001010111100101111011010000101011111010011111101
11011010100001011100000111100010111110111100111000110000100001110110100111110110010010000111011011010
011001001110011111011001111011100100100111001100000111110001011110000011110111011010001010011111100010111100100
1000010110011110101001100001111101011100101101110001000100011111010001101111001001110100011011010100111111
1000010011101111000110010101110101000001100110100110011011110110100010011010011010011111101100100110010000010
01001101010001000001001000011111010110010100111101100011100100100111010011111001001111100000011
1001111101100011111011110110110011100000111101011011011110001100101011111100000010001110000100001110101101
0010100110101000011010100111100001011010011010100000100110101001111101100010101111011100001001001110001011
00011110111101100010001011110100111011111010100110111000001111000101110111111001110000011010100111111
100000010001110000100001110101110111001001010111001001111010100111000001011101000110110100010100
11101011111001000001001110110101110111001101011110011001100100101111100010111100000001000111000
010000011101011001000001001000110101001111000101101101110000010000010011010100111011101110111101
11010001001000110010011100111101011101011010011101111101010011001011100101111110001001100001011000011
1110001110101111100000010101001101011111
-----
Texto descomprimido:
```

- Tasa de compresión: 53.043682 %
- Descompresión



```
imanol@elemOS: ../Practica5/Codigo
+ x ...S: ../Practica5/Codigo
-----
Texto descomprimido:
La nave vino del espacio. Vino de las estrellas, y las velocidades negras, y losmovimientos bri
llantes, y los silenciosos abismos del espacio. Era una nave nueva,con fuego en las entrañas y
hombres en las celdas de metal, y se movia en unsilencio limpio, vehemente y cálido. Llevaba di
ecisiete hombres, incluyendo uncapitán. En la pista de Ohio la muchedumbre habia gritado agitan
do las manos ala luz del sol, y el cohete había florecido en ardientes capullos de color y habí
aescapado alejándose en el espacio ¡en el tercer viaje a Marte!
-----
Tasa de compresion: 53.043682
-----
P2( 35 , 72 )
-----
```

- Codificación asignada



A terminal window titled 'imanol@elem...' with a tab labeled '.../Codigo'. The window displays the output of a Huffman coding process. It starts with the text 'Arbol de Huffman:' followed by a table mapping characters to their binary Huffman codes. The window has standard Linux terminal window controls (minimize, maximize, close) and a search icon.

| Char | Huffman   |
|------|-----------|
| a    | 000       |
| ä    | 0010000   |
| z    | 00100010  |
| ñ    | 001000110 |
| o    | 001000111 |
| p    | 001001    |
| r    | 00101     |
| ì    | 0011      |
| n    | 0100      |
| t    | 01010     |
| j    | 01011000  |
| E    | 01011001  |
| g    | 0101101   |
| L    | 01011100  |
| f    | 01011101  |
| V    | 010111100 |
| i    | 010111101 |
| M    | 010111110 |
| !    | 010111111 |
| e    | 011       |
| ,    | 100000    |
| ì    | 1000010   |
| .    | 1000011   |
| y    | 100010    |
| b    | 100011    |
| o    | 1001      |
| s    | 1010      |
| l    | 1011      |
| h    | 110000    |
| v    | 110001    |
| c    | 11001     |
| d    | 11010     |
| u    | 110110    |
| m    | 110111    |
|      | 111       |

# Algoritmo de Kruskal

## Aplicaciones del Algoritmo de Kruskal

Existen diversas aplicaciones del algoritmo en un escenario real mediante la industria.

Los problemas más socorridos por este algoritmo son aquellos que tienen que ver con distancias y puntos fijos en un área geográfica. De esta forma el problema de la distribución de los cables a las tomas o estaciones fijas repartidas en la ciudad, a cargo empresas del sector de las telecomunicaciones y servicios eléctricos, se auxilian de este para identificar el cableado más corto y conecte a un número de nodos fijos, significa menores costos en muchos de los aspectos infraestructurales.

Así mismo, puede ser considerada una problemática similar la ruta que conecta estaciones perteneciente a algún servicio de transporte, tal como un Metro, Tren o similares. Este algoritmo les auxilia permientiendo no solo disminuir costos para la construcción, y mantenimiento, pero asegura la ruta que más rápidamente permite el transporte de los usuarios, significando un ahorro también en el tiempo.

## Análisis a posteriori

Para obtener una gráfica descriptiva y que permitiera realizar un análisis mediante una aproximación de forma correcta, se decidió que se evaluarían, al menos, grafos con un número total de 10 nodos o aristas.

Para la descripción de los grafos, se utilizaron archivos donde cada fila corresponde a la descripción de un nodo. La sintaxis para la descripción del nodo inicia con el nombre del nodo, se colocan entre llaves los pares que corresponden a las trasiciones con otros nodos y el costo de la arista separando cada par por comas, de forma que se coloca primero el nombre del nodo destino y seguido de un signo de \$ el costo del vértice. Un ejemplo de la descripción de un grafo sería la siguiente 5:

```
Nodo1{ Nodo2$5 , Nodo3$12}  
Nodo2{ Nodo1$1 }  
Nodo3{ Nodo1$6 , Nodo2$8}
```

Figura 5: Ejemplo de grafo descrito en el archivo ingresado al programa

Los resultados obtenidos por el programa se muestran iniciando con **F**, le procede un signo de \$ y un número entero, que indica el coste total de árbol encontrado, y finalmente entre llaves se colocan los pares de nodos que conforman a este árbol mínimo 6:

```
F$7 = {(Nodo2,Nodo1) (Nodo3,Nodo1)}
```

Figura 6: Ejemplo de los elementos que conforman los resultados obtenidos del programa

A continuación se muestran los grafos evaluados:

### Grafo de 3 nodos

Grafo muy sencillo de únicamente 3 nodos. Fig7 y su estructura en 8.

El archivo que describe la estructura del grafo es:

Y el árbol mínimo recorrido encontrado fue: **F\$18={ (A,B) (B,C) }**

### Grafo de 4 nodos

Grafo muy sencillo de únicamente 4 nodos. Fig9 y su estructura en 10.

El archivo que describe la estructura del grafo es:

Y el árbol mínimo recorrido encontrado fue: **F\$7={ (3,4) (2,4) (1,4) }**

### Grafo de 5 nodos

Grafo de únicamente 5 nodos. Fig11 y su estructura en 12.

El archivo que describe la estructura del grafo es:

Y el árbol mínimo recorrido encontrado fue: **F\$11={ (A,B) (D,E) (B,C) (A,E) }**

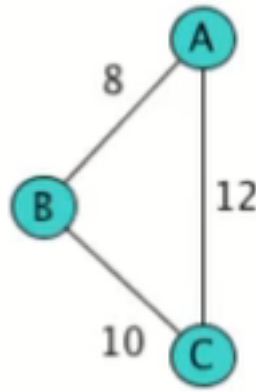


Figura 7: Gráfico de 3 nodos y 3 arcos

$A\{B\$8,C\$12\}$   
 $B\{A\$8,C\$10\}$   
 $C\{A\$12,B\$10\}$

Figura 8: Estructura del grafo 7 en el archivo

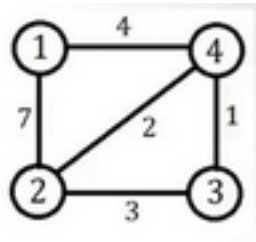


Figura 9: Gráfico de 4 nodos y 5 arcos

$1\{2\$7,4\$4\}$   
 $2\{1\$7,3\$3,4\$2\}$   
 $3\{2\$3,4\$1\}$   
 $4\{1\$4,2\$2,3\$1\}$

Figura 10: Estructura del grafo 9 en el archivo

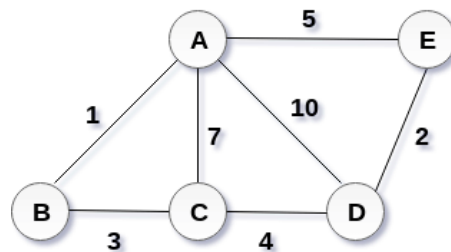


Figura 11: Gráfico de 5 nodos y 7 arcos

## Grafo de 6 nodos

Grafo de 6 nodos. Fig13 y su estructura en 14.

$A\{B\$1, C\$7, D\$10, E\$5\}$   
 $B\{A\$1, C\$3\}$   
 $C\{A\$7, B\$3, D\$4\}$   
 $D\{A\$10, C\$4, E\$2\}$   
 $E\{A\$5, D\$2\}$

Figura 12: Estructura del grafo 11 en el archivo

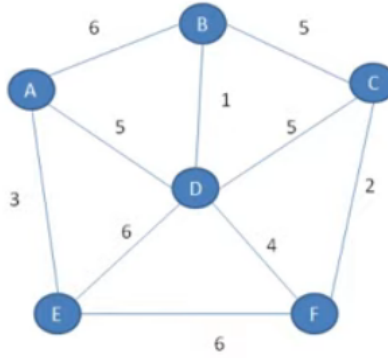


Figura 13: Grafo de 6 nodos y 10 arcos

El archivo que describe la estructura del grafo es:

$A\{B\$6, D\$5, E\$3\}$   
 $B\{A\$6, C\$5, D\$1\}$   
 $C\{B\$5, D\$5, F\$2\}$   
 $D\{A\$5, B\$1, C\$5\}$   
 $E\{A\$3, D\$6, F\$6\}$   
 $F\{C\$2, D\$4, E\$6\}$

Figura 14: Estructura del grafo 13 en el archivo

Y el árbol mínimo recorrido encontrado fue:  $F\$15 = \{ (B,D) (C,F) (A,E) (F,D) (A,D) \}$

### Grafo de 7 nodos

Grafo de 7 nodos. Fig15 y su estructura en 16.

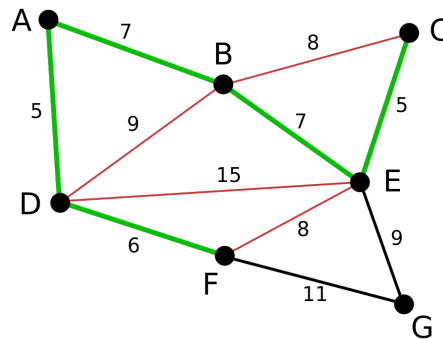


Figura 15: Grafo de 7 nodos y 11 arcos

El archivo que describe la estructura del grafo es:

Y el árbol mínimo recorrido encontrado fue:  $F\$41 = \{ (C,E) (A,D) (D,F) (B,E) (E,G) (B,D) \}$



```

A{B$7,D$5}
B{A$7,C$8,D$9,E$7}
C{B$8,E$5}
D{A$5,B$9,E$15,F$6}
E{B$7,C$5,D$15,F$8,G$9}
F{D$6,E$8,G$11}
G{E$9,F$11}

```

Figura 16: Estructura del grafo 15 en el archivo

### Grafo de 8 nodos

Grafo de 8 nodos. Fig17 y su estructura en 18.

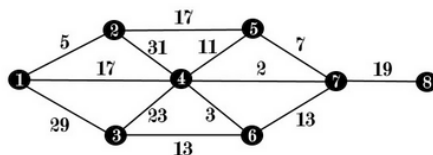


Figura 17: Gráfo de 8 nodos y 13 arcos

El archivo que describe la estructura del grafo es:

```

1{2$5,3$29,4$17}
2{1$5,4$31,5$17}
3{1$29,4$23,6$13}
4{1$17,2$31,3$23,5$11,6$3,7$2}
5{2$17,4$11,7$7}
6{3$13,4$3,7$13}
7{4$2,5$7,6$13,8$19}
8{7$19}

```

Figura 18: Estructura del grafo 17 en el archivo

Y el árbol mínimo recorrido encontrado fue: **F\$64**= { (4,7) (4,6) (1,2) (5,7) (3,6) (2,5) (1,4) }

## Grafo de 9 nodos

Grafo de 9 nodos. Fig19 y su estructura en 20.

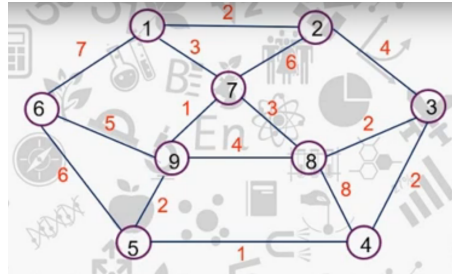


Figura 19: Gráfico de 9 nodos y 15 arcos

El archivo que describe la estructura del grafo es:

```

1{2$2,7$3,6$7}
2{1$2,7$6,3$4}
3{2$4,8$2,4$2}
4{3$2,8$8,5$1}
5{4$1,9$2,6$6}
6{5$6,9$5,1$7}
7{1$3,2$6,8$3,9$1}
8{3$2,4$8,9$4,7$3}
9{7$1,8$4,5$2,6$5}

```

Figura 20: Estructura del grafo 19 en el archivo

Y el árbol mínimo recorrido encontrado fue:  $\mathbf{F\$18} = \{ (7,9) (4,5) (5,9) (3,4) (3,8) (1,2) (1,7) (6,9) \}$

## Grafo de 10 nodos

Grafo de 10 nodos. Fig21 y su estructura en 22.

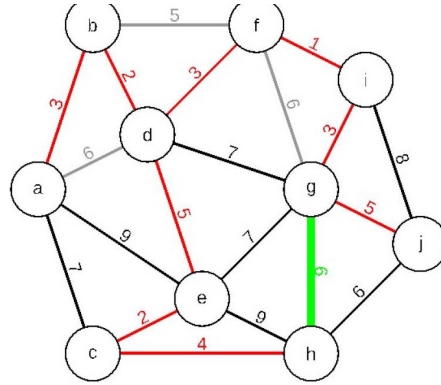


Figura 21: Gráfico de 10 nodos y 15 arcos

El archivo que describe la estructura del grafo es:

```
a{b$3,d$6,e$9,c$7}
b{a$3,d$2,f$5}
c{a$7,e$2,h$4}
d{a$6,b$2,e$5,f$3,g$7}
e{a$9,c$2,d$5,g$7,h$9}
f{b$5,d$3,g$6,i$1}
g{d$7,e$7,f$6,h$6,i$3,j$5}
h{c$4,e$9,g$6,j$6}
i{f$1,g$3,j$8}
j{g$5,h$6,i$8}
```

Figura 22: Estructura del grafo 21 en el archivo

Y el árbol mínimo recorrido encontrado fue: **F\$28**= { (f,i) (c,e) (b,d) (g,i) (d,f) (a,b) (c,h) (g,j) (b,f) }

## Propuesta de complejidad

A partir de estos 8 archivos ingresados y evaluados por el programa, la gráfica en la figura 23 muestra el resultado de la comparación del número de nodos o aristas del grafo ingresado, contra el número de operaciones realizadas para cada archivo: Y los puntos que genera esta gráfica son 24:

**Graficación del número de nodos en el grafo vs instrucciones ejecutadas para el algoritmo Kruskal**

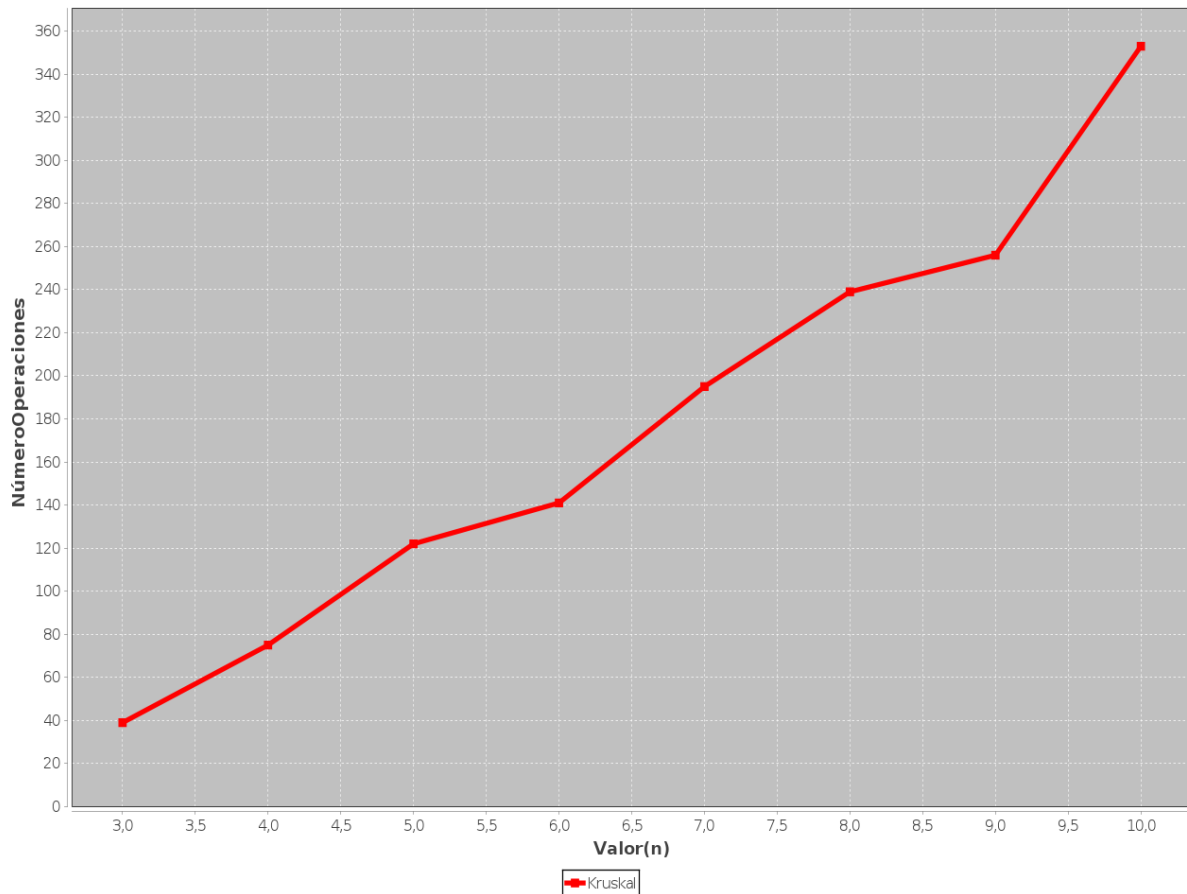


Figura 23: Gráfica generada a partir de la evaluación de los archivos con los grafos descritos

En el eje de las abscisas se encuentra el número de nodos del grafo y en el de las ordenadas el número de operaciones realizadas

|              |               |
|--------------|---------------|
| P1( 3 ,39 )  | P5( 7 ,195 )  |
| P2( 4 ,75 )  | P6( 8 ,239 )  |
| P3( 5 ,122 ) | P7( 9 ,256 )  |
| P4( 6 ,141 ) | P8( 10 ,353 ) |

Figura 24: Puntos obtenidos de la evaluación de los 8 grafos especificados anteriormente

Y es con estos datos obtenidos que se propone la complejidad para este algoritmo. En la figura ?? se muestran los puntos obtenidos y 2 ecuaciones que acotan a estos mismos. Es evidente desde la figura 23 que su cota superior no podría ser lineal pues el crecimiento que experimenta en el eje de las ordenadas es muy veloz en comparación con una complejidad lineal.

Por esta razón se propone hacer una comparación de las 2 cotas siguientes superiores, la cota  $n \log n$  y  $n^2$ .

Mediante la asignación arbitraria de valores constantes que multiplican a los términos, observamos claramente que una

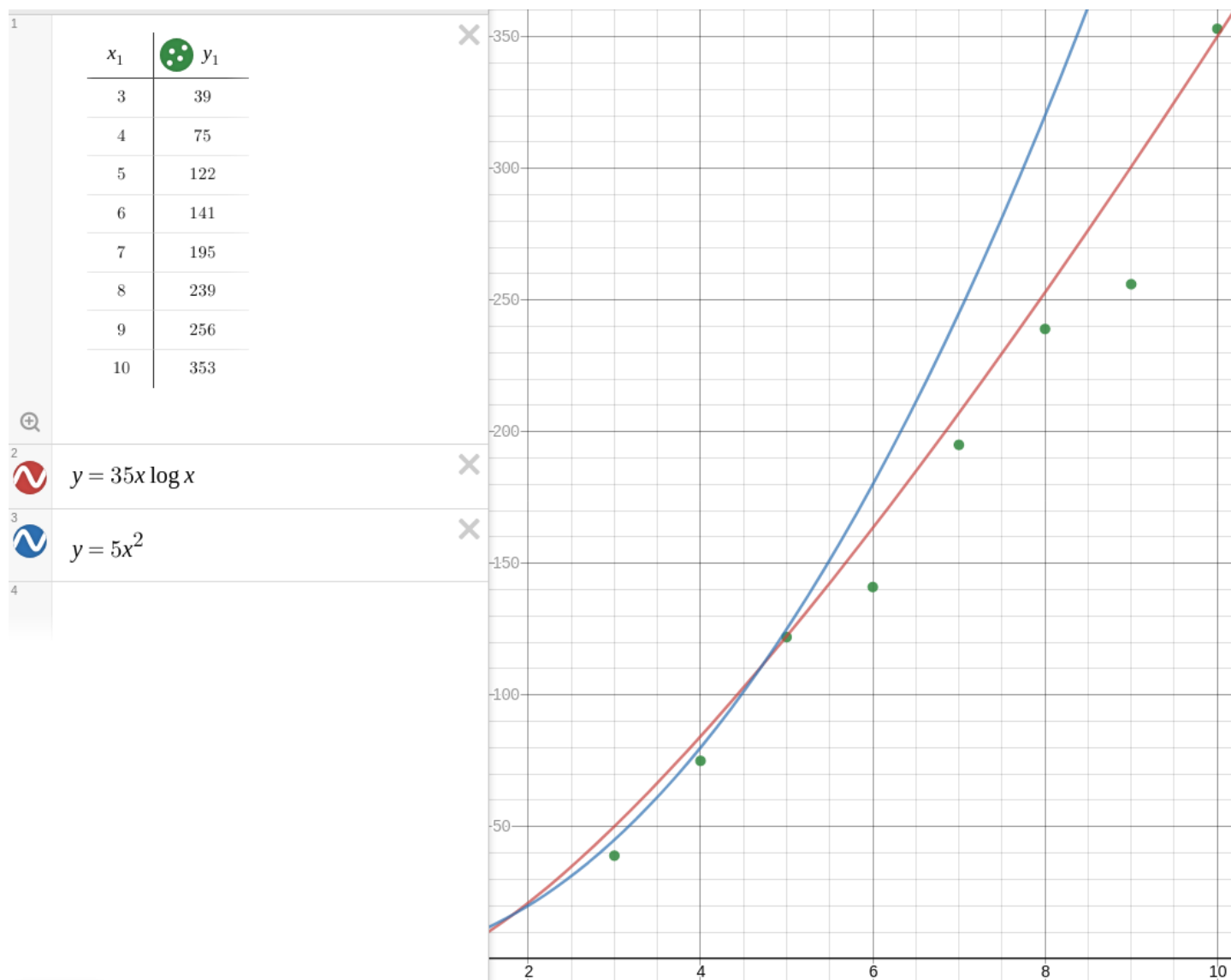


Figura 25: Gráfica de la proposición para la complejidad para el algoritmo de Kruskal  
 Los puntos verdes corresponden a los puntos obtenidos de la evaluación de los grafos  
 La curva de color rojo corresponde a la cota de orden  $n \log n$   
 La curva de color azul corresponde a la cota de orden  $n^2$

cota del orden  $n \log n$ , se ajusta de forma más justa y describiendo un comportamiento parecido al de los puntos.

Mientras que la cota cuadrática en un principio parece acotar de forma más justa a los primeros 3 puntos, su crecimiento será muy superior con los datos obtenidos posteriores.

De tal manera que se afirma que para el algoritmo de Kruskal que permite encontrar un árbol mínimo recubridor en un grafo conexo no dirigido y ponderado será:

$$\text{kruskal}(n) \in O(n \log n)$$

## Investigación complejidad

Los elementos esenciales para el funcionamiento del algoritmo, y por consiguiente los que nos darán el tiempo de ejecución, serán en general el grafo, pero siendo más específicos, los elementos que componen al árbol. Las aristas(**a**) y los vértices(**v**), que ya se especificó antes se encuentran ponderados mediante un costo, son los 2 elementos principales que

intervienen en la complejidad del algoritmo.

Se considera que para el algoritmo de Kruskal la complejidad será a lo más  $O(a \log a)$ , siendo equivalente la cota  $O(a \log n)$ , cuando los datos se encuentran almacenados mediante estructuras simples.

La razón por la que estas 2 cotas son equivalentes se debe a:

- Las aristas serán a lo más  $v^2$ , y el  $\log v^2 = 2 \log v$  que puede calcularse mediante diversos métodos, pero otorga una complejidad de  $O(\log v)$
- Se ignoran los vértices aislados, esto debido que forman su propia componente del árbol de expansión mínimo, los vertices son menores o iguales a el doble de las aristas  $v \leq 2a$ , resultando entonces que el  $\log n$  es  $O(\log a)$

Para poder conseguir esta complejidad es primordial ordenar las aristas en función del peso, mediante un algoritmo con complejidad  $O(a \log a)$ , como lo es *Comparison sort*. Esta ordenación permite que la eliminación de una arista con peso mínimo pueda ejecutarse en un tiempo constante.

Ahora, dado que es necesario el control de los vértices para identificar cuales de estos están en que componentes, se utiliza una estructura de datos sobre conjuntos disjuntos. La complejidad del ordenamiento de este será  $O(a)$  operaciones, esto se debe a 2 operaciones de búsqueda por arista, y una posible unión de conjuntos.

Finalmente se considera que inclusive con una estructura de conjuntos disjuntos simples con uniones por rangos, es capaz de ejecutar las operaciones mencionadas en un tiempo  $O(a \log v)$ , se considera que el orden de este algoritmo es:

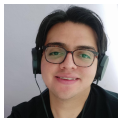
$$O(a \log a) = O(a \log v)$$

Este algoritmo permite además encontrar soluciones para estructuras de conjuntos disjuntos complejos, tales como los bosques, es posible acotar los tiempos de ejecución en  $O(a \alpha(v))$ , de donde  $\alpha$  es la inversa de la *Función de Ackermann*.

La función de *Ackermann*, es un función matemática recursiva que fue encontrada en el año de 1926 por el matemático alemán Wilhelm Ackermann. Que tienen como particularidad, un crecimiento extremadamente rápido, lo que consecuentemente las volvería un recurso interesante para la ciencia computacional teórica y la teoría de computabilidad.

Es precisamente por esta característica, que el valor de su inversa tendrá el comportamiento contrario, un crecimiento sumamente lento, dotando de una complejidad aún muy controlada inclusive cuando se trabaja con estructuras complejas para el algoritmo de Kruskal.

# Conclusiones



**Rivero Ronquillo Omar Imanol**

Nunca había pensado en las formas en las que se podría optimizar la solución a un problema por medio de un algoritmo programable. Me pareció un enfoque increíble y ahora me parece que tengo claro porque se usan estos algoritmos y en qué problemas se les podría dar un uso. A pesar de que en comparación con las prácticas anteriores se tuvieron que incluir más cosas además de los algoritmos, para poder probarlos y estudiarlos, me parecieron un gran complemento en mi formación académica.

En conclusión, me gustaría conocer más algoritmos de este tipo, me interesaron especialmente los algoritmos de compresión, aunque puedo estar bastante seguro de que las compresiones modernas deberán de tener una complejidad mayor de aprenderse y comprender su funcionamiento, que seguramente justificaran todo esto con las tasas de compresiones que puedan lograr.



**Valle Martínez Luis Eduardo**

Para esta práctica se abordaron algoritmos interesantes que tiene aplicaciones prácticas reales, de las que yo no tenía presente cual era su funcionamiento, ni los elementos que le permitían realizar tal implementación. Es especialmente rescatable el algoritmo de Kruskal pues resuelve, aunque problemas similares, puede aplicarse a una variedad de industrias.

Me pareció sumamente interesante el tener un primer acercamiento a una función tan recurrentemente utilizada por todos nosotros pero que no solemos tener perspectiva de como se implementa, este es el algoritmo para la compresión de archivos desarrollado por Huffman.

Finalmente durante la elaboración y codificación de las funciones, afronté algunos problemas en el momento de seleccionar los elementos que forman la solución para el algoritmo de Kruskal, esto debido a que consideraba los mismos arcos pero con los nodos origen y destino volteados generando una entrada doble, sin embargo esto fue corregido en la función que permite ordenar en un arreglo los elementos por sus pesos, donde modifiqué para que solo contara uno de estos pares.