Institución

Escuela Superior Politécnica del Litoral

Título del Proyecto

Algoritmo de codificación de Huffman con la implementación de árboles

Integrantes

Melissa Ayllon

Paulina Loor

Gabriel Cañarte

Dayanara Bone

Allison Redrovan

Asignatura

Matemáticas Discretas

Docente

Domingo Quiroz

Fecha de la propuesta

20-01-2023

GUAYAQUIL-ECUADOR

PAO 2

INDICE

Objetivo General

Describir cómo se utiliza un árbol de Huffman para comprimir información mediante la asignación de códigos de longitud variable a símbolos individuales en una secuencia de datos. El uso de árboles en la codificación de Huffman permite una compresión eficiente de los datos al asignar códigos más cortos a los símbolos que aparecen con mayor frecuencia en la secuencia de datos original.

Objetivos Específicos

Describir cómo se construye un árbol de Huffman a partir de la frecuencia de los símbolos en una secuencia de datos.

Explicar cómo se utilizan los caminos en el árbol de Huffman para codificar y decodificar la información comprimida.

Proporcionar ejemplos concretos de cómo se aplica la codificación de Huffman en la práctica.

Términos claves

Método: un método es una porción de código que realiza una tarea hacia un objeto determinado.

Mapa: estructura que almacena parejas de datos, asociando una clave y un valor.

Cola de prioridad: una cola de prioridad es una estructura que organiza los datos en función de una prioridad establecida, de esta forma el primero en ser devuelto por la cola será el menor o el mayor objeto que cumpla dicha prioridad.

Descripción de los conceptos matemáticos

Los árboles forman una de las subclases de graficas que mas se utilizan. En particular, la ciencia de la computación hace uso de los arboles ampliamente. (Johnsonbaugh, 2005)

Según su definición formal, un árbol es una gráfica simple que satisface lo siguiente: si *v* y *w* son vértices en T, existe una trayectoria simple única de *v* a *w.*

En computación, los árboles de Huffman son una estructura de datos utilizada en la codificación de Huffman, un algoritmo de compresión de datos que se utiliza para codificar un texto o un archivo de audio o video. El árbol se construye a partir de un conjunto de símbolos y sus frecuencias de aparición en el texto o archivo original. Cada símbolo se representa mediante un nodo en el árbol, y las ramas entre los nodos representan las diferentes formas en que los símbolos pueden combinarse.

La idea es utilizar una técnica llamada codificación de longitud variable. Esta técnica se basa en el hecho de que algunos caracteres son más comunes que otros en un texto. El objetivo es diseñar un algoritmo que permita representar el mismo fragmento de texto utilizando menos bits. En la codificación de longitud variable, se asignan diferentes cantidades de bits a los caracteres según su frecuencia en el texto. Esto significa que algunos caracteres pueden requerir solo un bit, mientras que otros pueden requerir dos o tres bits. Sin embargo, un problema con esta técnica es su decodificación. (Techie Delight, s.f.)

Entrada: Sucesión de n frecuencias, $n \geq 2$
Salida: Árbol con raíz que define un código Huffman óptimo

```
huffman(f, n) {
    if(n == 2) {
        sean f_1 y f_2 las frecuencias
        sea T como en la figura 9.1.11
        return T
    }
    sean f_i y f_j las frecuencias más pequeñas
    se sustituyen f_i y f_j en la lista f por f_i + f_j
    T' = huffman(f, n − 1)
    se sustituye un vértice en T' etiquetado f_i + f_j por el
        árbol de la figura 9.1.12 para obtener el árbol T
    return T
}
```

Fig. 1. Algoritmo de codificación Huffman

Para evitar confusiones al momento de la decodificación, se asegurará de cumplir con la "regla del prefijo", lo que asegurará que los códigos sean decodificados de manera única. Esta regla establece que ningún código debe ser el prefijo de otro código.

Requerimientos y especificaciones

1. Número de personas que pueden interactuar:
   2 visitantes quienes tendrán interacción directa con el juego y 1 instructor quien será el encargado de aclarar dudas sobre cómo interactuar con el programa, además dirigirá la actividad.

2. Equipo necesario:
   Computadora con sistema operativo mínimo Windows 8 y lugar donde ubicarla.

3. Área requerida:
   Cualquier lugar donde los usuarios se sientan cómodos al utilizar la computadora, se incluyen áreas abiertas como parques.

4. Instalaciones requeridas:

Conexión eléctrica para el funcionamiento de la computadora en caso de ser de escritorio. Conexión a internet para poder visualizar el árbol construido.

5. Seguridad:
   Edad mínima para el juego: 7 años
   Condiciones de salud: Sin especificaciones

6. Supervisión:
   Un supervisor que vigile el correcto funcionamiento de la aplicación. Para los usuarios con las edades mínimas se recomienda que un adulto esté a su lado con el objetivo de que no cierren el programa y que sigan los pasos necesarios para obtener los resultados de aprendizaje deseados.

Funcionamiento

El programa está compuesto de una única pantalla en la cual se ingresará la palabra o la frase que se desea transformar.

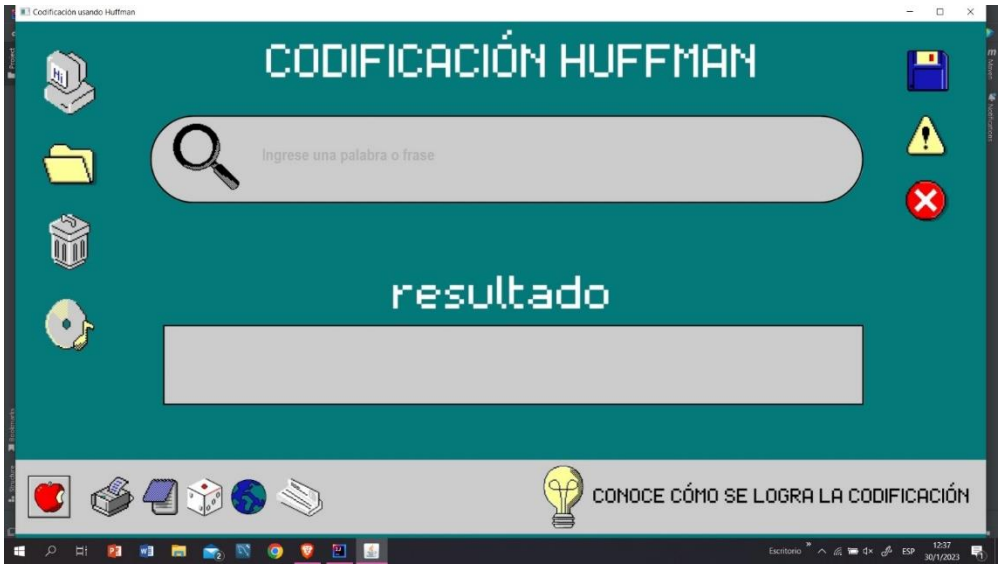

Fig. 2. Pantalla inicial del programa

Una vez que el usuario ha decidido que frase desea usar, se procede a dar click sobre la lupa. Aquí se ejecutará el algoritmo y se generará el código generado para la frase ingresada.

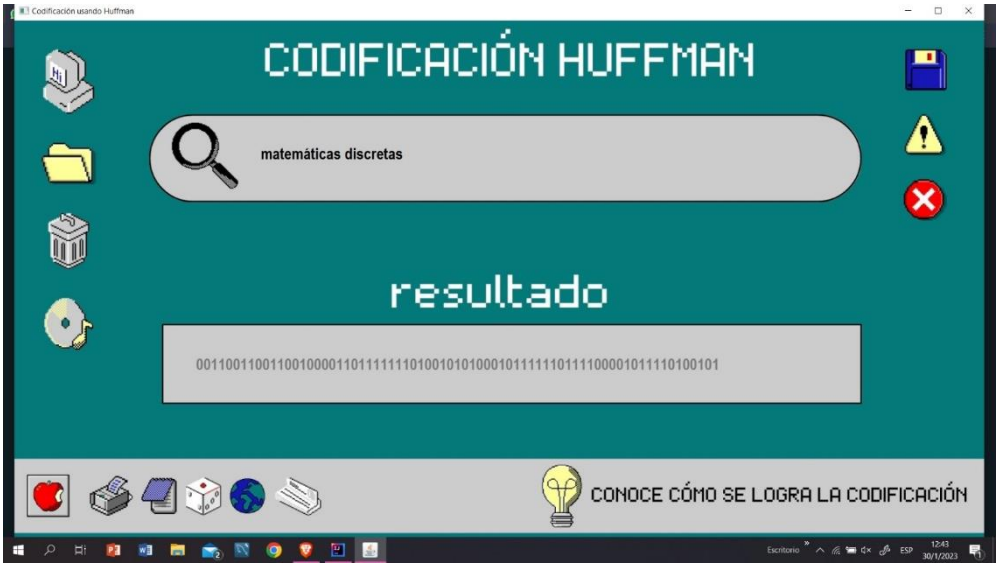

Fig. 3. Pantalla inicial con el resultado obtenido

Para poder ver el árbol resultado con los códigos de cada uno de los caracteres y sus frecuencias, se deberá presionar el botón de conoce como se logra. Con este recurso se buscará que los usuarios puedan ver realmente como se ha construido el árbol.

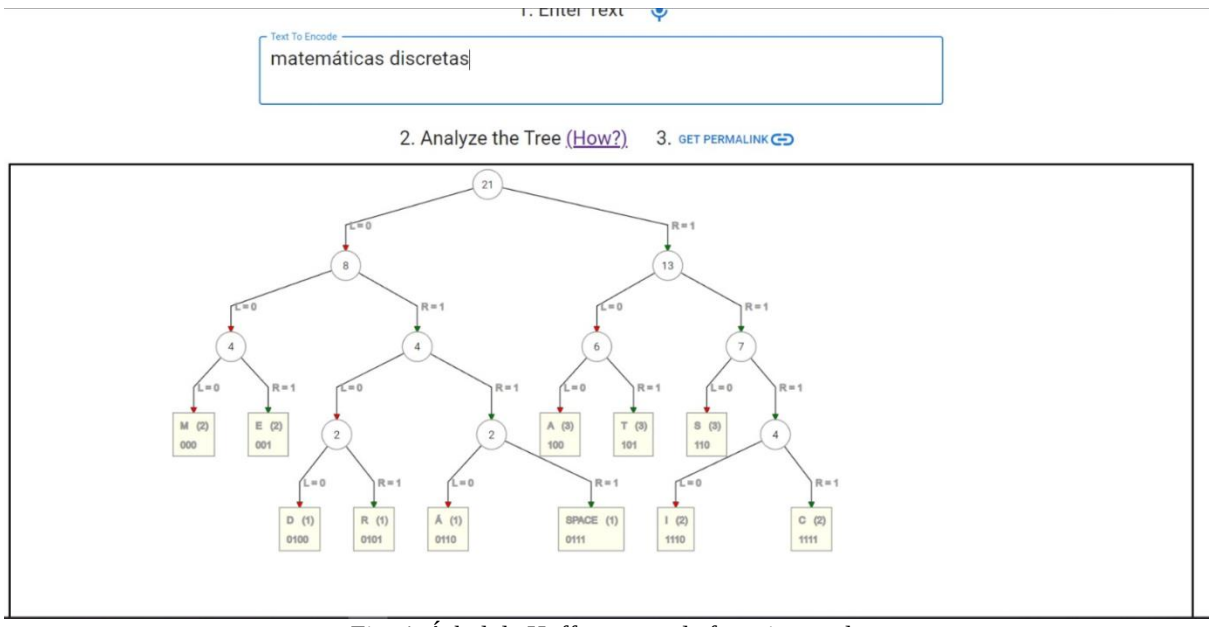

Fig. 4. Árbol de Huffman para la frase insertada

Algoritmo

Como se puede ver en la figura 1, el algoritmo recibe las frecuencias de todos los caracteres, en caso de que solo hayan dos de estos, se entrega un árbol con únicamente dos hijos. Por otro lado, si la sucesión posee mas de dos caracteres, se escogerán los dos con las frecuencias más pequeñas y mediante recursión se devolverá un nuevo nodo que será agregado al árbol final, este nuevo nodo reemplaza a los dos caracteres seleccionados, dentro de la sucesión inicial.

Para implementar el algoritmo de Huffman en el programa, se ha utilizado el lenguaje de programación Java. A continuación, los métodos usados para implementar el algoritmo.

```
public static Map<String, Integer> getFrecuencies(String cadena){
    Map<String, Integer> dicc = new LinkedHashMap<>();

    String c = null;
    for (int i = 0; i<cadena.length();i++){
        int contador =0;
        c = ""+cadena.charAt( index: i);
        for (int j = 0; j<cadena.length();j++){
            if (c.equals(""+cadena.charAt( index: j))){
                contador++;
            }
        }
        dicc.put( key: c,  value: contador);
    }
    return dicc;
}
```

Fig. 5. Método getFrecuencias

El método getFrecuencias de la figura 5, es la representación de la sucesión de frecuencias solicitada en el algoritmo de la figura 1. Este devuelve un mapa cuyas claves son los caracteres y los valores son las frecuencias.

```
public static BinaryTree<HuffmanInfo> buildHuffmanTree(Map<String, Integer> mapa){
    Queue<BinaryTree<HuffmanInfo>> pq=new PriorityQueue<>((f1,f2)->
    { return f1.getRoot().getContent().getFrecuencia() - f2.getRoot().getContent().getFrecuencia();});

    Set<String> keys=mapa.keySet();
    Iterator<String> it=keys.iterator();
    while(it.hasNext()){
        String letras=it.next();
        pq.offer(new BinaryTree<>(new BinaryNode<>(new HuffmanInfo( letra: letras, frecuencia: mapa.get( key: letras)))));
    }
    while(pq.size()!=1){
        BinaryTree<HuffmanInfo> bn1=pq.poll();
        BinaryTree<HuffmanInfo> bn2=pq.poll();
        BinaryTree<HuffmanInfo> bn3=new BinaryTree<>
                (new HuffmanInfo(bn1.getRoot().getContent().getLetra()+bn2.getRoot().getContent().getLetra(),
                        bn1.getRoot().getContent().getFrecuencia()+bn2.getRoot().getContent().getFrecuencia()));
        bn1.getRoot().getContent().setBit( bit: 0);
        bn2.getRoot().getContent().setBit( bit: 1);
        bn3.setLeft( tree: bn1);
        bn3.setRight( tree: bn2);

        pq.offer( e: bn3);
    }

    return pq.poll();
}
```

Fig. 6. Método buildHuffmanTree

El método buildHuffmanTree de la figura 6 es el encargado de construir el árbol, este ordena las frecuencias de menor a mayor haciendo uso de una cola de prioridad. Cada una de estas frecuencias es tratada como un árbol individual, al escoger a los dos de menor frecuencia se los une como hijos de un nuevo árbol y se las agrega a la cola.

Cuando la cola tiene únicamente un elemento, el proceso ya no se repite gracias al ciclo while definido en el algoritmo, de esta forma este único elemento es el árbol final con todas las frecuencias, mismo que será devuelto como resultado.

Bibliografía

Johnsonbaugh, R. (2005). *Matemáticas Discretas.* México: Pearson Educación.

Techie Delight. (s.f.). *Algoritme de compresión de codificación de Huffman*. Obtenido de Techie Delight: https://www.techiedelight.com/es/huffman-coding/