# Árboles de Merkle

Victor Hugo Martínez Huicochea

14 de diciembre de 2021

# Índice



**Resumen**

Los árboles de Merkle surgieron como una estructura de datos no lineal sin esperanzas, pero con el pasar del tiempo, se convirtió en un elemento importante dentro de la criptografía moderna.

## 1. ¿Qué es un árbol de Merkle

El árbol de Merkle o árbol Hash (en inglés Merkle Tree o Merkle Hash Tree) es una estructura de datos no lineal que es derivado de una estructura de árbol.

Creado y patentado por Ralph Merkle en 1979 con el fin de agilizar el proceso de verificación de grandes cantidades de datos, el árbol de Merkle ha trascendido hasta convertirse en la estructura de datos más segura para resumir y verificar la integridad de una base de datos.

Dicha estructura de datos esta dividida en varias capas que tienen como finalidad relacionar cada nodo con la **raíz**. Para lograr esto, cada nodo tiene un identificador único llamado **Hash**. Los nodos iniciales se llamarán nodos **hijos** o nodos **hojas**, que se asociará luego con un nodo de nivel superior llamado nodo **padre** o nodo **rama**. El Hash del nodo padre será obtenido a partir del Hash de sus hijos. Luego los nodos padres pasarán a ser nodos hijos y se repite el mismo proceso, Así esta

estructura se repetrirá hasta llegar al nodo **raíz** o **raíz Merkle** (**Merkle Root** en inglés), cuyo Hash está asociado a todos los nodos del árbol, logrando así que con solo verificar el Hash del nodo raíz, podamos comprobar si hubo alguna alteración de los datos.

# 2. Partes de un árbol de Merkle

Un árbol de merkle tiene distintas partes, las cuales mencionaremos a continuación:

1. **Hash**: Es el identificador único de cada nodo, sirve para comprobar que la estructura este inalterada.
2. **Nodo Hijo o Nodo Hoja**: Son los nodos de nivel bajo, son aquellos datos que se van a unir.
3. **Nodo Padre o Nodo Rama**: Son los nodos en donde se uniran los datos de los nodos hoja.
4. **Nodo Raíz o Raíz Merkle**: Es el nodo de mayor nivel del árbol y a donde terminarán de unificarse todos los nodos.

## 2.1. Hash

Aunque hagamos referencia a Hash como un identificador, lo cierto es que Hash también se le denomina a la función que asigna datos de longitud arbitraria a datos de longitud fija. Así, la **función Hash** asigna un **valor Hash** a todos los datos.

Con lo cual, cuando se requiera verificar la integridad de los datos, podemos usar la función Hash para asignar un valor Hash a los datos y así obtener un valor Hash de longitud fija. Lo mismo haremos en otro entorno, y luego compararemos los Hash, si la comparación da que son iguales, entonces dichos datos no se han dañado, en cambio, si se ha sufrido un ligero cambio en los datos de entrada, el resultado de la operación Hash será completamente irreconocible.

## 2.2. Lista Hash

La **lista Hash** (**Hash list** en inglés) es una estructura de datos formada por el conjuto de valores hash de una lista de cadenas de bloques de datos. Es una extensión del concepto de hash.

Se suelen usar para la busqueda rápida y para la protacción de la integridad de datos para que permanezcan inalterados.

Comunmente se le añade un valor hash adicional a la lista, el cual, es el valor de la lista de hash completa. A este valor se le llama **hash raíz**. El cual facilita comprobar la integridad de la lista. Así, podemos decir que , un árbol de Merkle es una generalización de una Lista Hash y una Lista Hash es un tipo especial de árbol de Merkle que consta de dos niveles.

# 3. Características de un árbol de Merkle

Sus principales características son:

- **Es Eficiente**: Siendo esta la base del porque se creó, permite verificar grandes cantidades de datos de manerá rápida y eficaz.
- **Una sincronización rápida de datos**: Esta estructura permite que la actualización de los datos que la componen se haga de una manera distribuida entre todos los pares que contengan la información y no solo a través de un único nodo. Lo cual facilita cualquier sincronización.
- **Es segura**: La más mínima modificación generará un identificador distinto, con lo cual se vuelve más complicado de alterar.

# 4. Implementación de un Árbol de Merkle en un código java

Se va a mostrar a continuación un código donde se implementan los árboles de Merkle.

Dicho código es simple, el programa te pedirá 5 frases, las cuales codificará, para luego unirlas en un nodo raíz y mostrar dicha codificación en pantalla.
Primero crearemos una Lista donde guardaremos todas las frases e invocamos la función Scanner para la lectura de estas.

```
public class ArbolMerkle {


    public static void main(String[] args) {
        // Se crean los elementos del arbol
        ArrayList<String> Bloque = new ArrayList<>();
        Scanner P1= new Scanner(System.in);
        String F1;

        int i=0;
        //agregaremos 5 elementos de prueba, pero podemos aumentarlo
        System.out.println("Digite 5 Frases porfavor");
        for(i=0;i<5;i++)
        {
            F1=P1.nextLine();
            Bloque.add(F1);
        }
```

Luego implementaremos la creación de las funciones y métodos que contendra al Nodo con sus respectivos getters y setters, incluyendo uno para el Hash.

```java
//Estos nodos serviran para la creacion del arbol
public class Nodo {
    private Nodo izq;
    private Nodo der;
    private String hash;

    public Nodo(Nodo izq, Nodo der, String hash) {
        this.izq = izq;
        this.der = der;
        this.hash = hash;
    }


    //Implementamos los getters y setters
    public Nodo getIzq() {...3 lines }

    public void setIzq(Nodo izq) {...3 lines }

    public Nodo getDer() {...3 lines }

    public void setDer(Nodo der) {...3 lines }

    public String getHash() {...3 lines }

    public void setHash(String hash) {...3 lines }
}
```

Ahora crearemos el programa para codificar las frases ingresadas, en este caso, convierte el String en un Array y va convirtiendo cada letra a un número del 1 al 27, para luego regresarlo a un String, que será retornado.

```java
public class Codificador {
    public static String Codigin (String F1) {
        int i=0;


        char Fra1[]=F1.toCharArray();
        int Conv[];
        Conv=new int[F1.length()];
        while(i<F1.length())
            {

                switch(Fra1[i])
                {
                    case'0':
                        Conv[i]=0;
                        break;
                    case 'A':
                    case'a':
                    case'1':
                        Conv[i]=1;
```

```
    //convertimos de nuevo a String
    StringBuffer sb = new StringBuffer();
            for(i=0;i<F1.length();i++){
               sb.append(Conv[i]);
            }

    String Cod;
    Cod=sb.toString();



    //retornamos
    return new String(Cod);
}
```

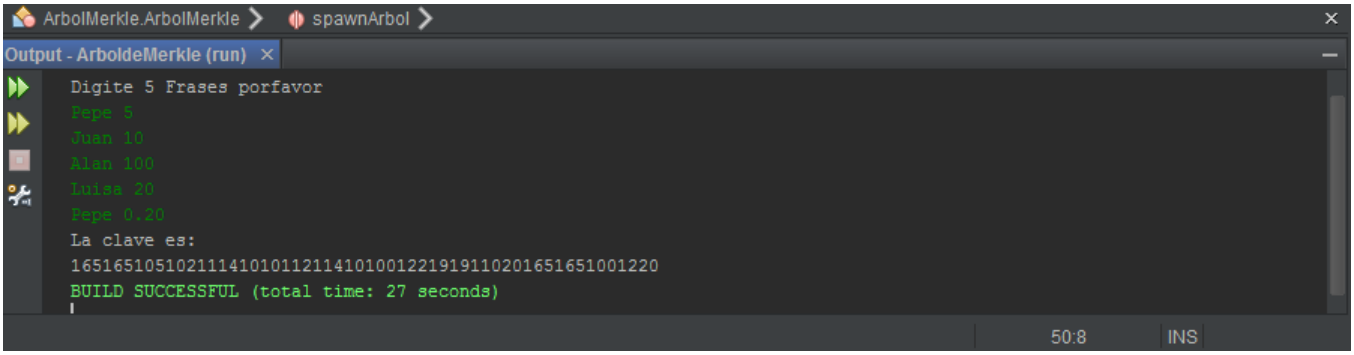
Obtendremos el nodo raíz a partir de insertar la lista de frases en el método **spawnArbol**, el cual creará una lista de tipo Nodo, en donde cada elemento de la lista será un nodo con la frase ya codificada, enviaremos dicha lista al método **Arbololvo** el cual creará al árbol y retornará la raíz, la cual a su vez retornará al método principal para ser mostrado.

```
    //generamos el arbol, le mandamos la lista y obtenemos la raiz
    Nodo raiz = spawnArbol(Bloque);



    System.out.println("La clave es:");
    System.out.println(raiz.getHash());
}

public static Nodo spawnArbol(ArrayList<String> Bloque) {
    //generamos una nueva lista
    ArrayList<Nodo> Codificar = new ArrayList<>();

    //Con el for each, mandamos a codificar los Strings y los agregamos a la lista
    for (String Codin : Bloque) {
        Codificar.add(new Nodo(null, null, Codificador.Codigin(Codin)));
    }

    //retornaremos el nodo raiz que se obtiene del metodo al que se le pasa la lista "Codificar"
    return Arbololvo(Codificar);
}
```

El método Arbololvo creará al árbol, Para esto crearemos una **lista "padre"** o lista rama"donde contendremos al nodo resultado de la unión de otros nodos "hijo.º "hoja". Así mismo, usamos un bucle **while** donde se unificaran los elementos de las hojas a la lista rama, las cuales pasará a ser ahora la lista hojas y se repetirá el proceso hasta que se obtenga el nodo raíz.

```java
private static Nodo Arbololvo(ArrayList<Nodo> hojas) {
    //Creamos la lista que contendra a las ramas de los nodos hoja
    String Palo;
    ArrayList<Nodo> Rama = new ArrayList<>();

    /*unificamos a las "hojas" en una "rama",
    que si no es la RAIZ, entonces las ramas de vuelven hojas y se repite el ciclo*/
    while (hojas.size() != 1) {
        int i = 0, t = hojas.size();
        while (i < t) {
            Nodo hojaizq = hojas.get(i);
            Nodo hojader = null;

            //Condiciona para que cuando no exista el elemento, no se agregue
            if ((i + 1) < t) {
                hojader = hojas.get(i + 1);
            } else {
                hojader = new Nodo(null, null, null);
            }

            if(hojaizq.getHash()!=null && hojader.getHash()!=null)
            {
            Palo =hojaizq.getHash() + hojader.getHash();
            }
            else
            {
            Palo =hojaizq.getHash();
            }
             Rama.add(new Nodo(hojaizq, hojader, Palo));
             i += 2;

        }
        //reiniciamos a la lista Rama y la Rama que teniamos vuelve a ser hoja
        hojas = Rama;
        Rama = new ArrayList<>();
        }

    //Retornamos
    return hojas.get(0);
}
```

Finalmente retornamos el nodo raíz e imprimimos su Hash.




## 5. Ralph Merkle

Ralph Charles Merkle nació en Berkeley, California en 1952. Hijo de Theodore Charles Merkle. Desde temprana edad se le introdujo en un ambiente científico.

El camino educativo de Merkle comenzó en 1963 al empezar sus estudios de secundaria en la Livermore High School. Para luego ingresar en 1970, a la Universidad de California. En ella, se graduó como especialista en Ciencias de la Computación en el año de 1974. Más tarde en 1977, presentaría en Berkeley su Maestría en Ciencias de la Computación y en 1979, su Doctorado.

Merkle inició sus trabajos en Criptografía antes de terminar su estancia en Berkeley. Fue durante este periodo, que, bajo la tutoría de Lance Hoffman, cuando se presentó su primer trabajo para la creación de un sistema de cifrado público. El cual fue rechazado y catalogado como inservible. Sin embargo, Merkle siguió trabajando en el mismo para seguir mejorándolo sin saber que su investigación iba a ser completamente revolucionaria para la época.

No fue hasta que Merkle contactó con Bob Fabry, quien reconoció el enorme valor de este trabajo y comentó a Merkle lo siguiente: **"Publícalo, haz fama y fortuna"**. Sin embargo, decirlo era más fácil que hacerlo, puesto que ningún espacio de publicación académico quería publicarlo. No fue hasta 4 años después que vió finalmente la luz.

Actualmente Merkle está casado con Carol Shaw, una diseñadora de videojuegos mejor conocida por su juego: River Raid.

# 6. Bibliografía


- Bit2Me. (2020, noviembre 9). ¿Qué es un Árbol Merkle? Bit2Me Academy. Recuperado de: https://academy.bit2me.com/que-es-un-arbol-merkle/
- Bit2Me. (2020, noviembre 10). ¿Quién es Ralph Merkle? Bit2Me Academy. Recuperado de: https://academy.bit2me.com/quien-es-ralph-merkle/
- ESAN Graduate School of Business. (2019, 4 diciembre). Fundamentos de Blockchain: ¿qué es un Merkle tree? Tecnología | Apuntes empresariales| ESAN. Recuperado de: https://www.esan .edu.pe/apuntes-empresariales/2019/12/fundamentos-de-blockchain-que-es-un-merkle-tree/
- Programador Clic. (2021, 14 marzo). Análisis del algoritmo Merkle Tree (árbol merkle). Programmerclick.com. Recuperado de: https://programmerclick.com/article/52821764560/
- Wikipedia. (2019, 25 octubre). Lista hash. Wikipedia, la enciclopedia libre.Recuperado de: https://es.wikipedia.org/wiki/Lista_hash