Árboles de Merkle

Fundamentos de la programación

Victor Hugo Martínez Huicochea

Escuela Superior de Física y Matemáticas
Instituto Politécnico Nacional

December 14, 2021

Contenidos

¿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**. El Hash del nodo padre será obtenido a partir del Hash de sus hijos.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.

Partes del 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.

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.

Conceptualización del programa

- El código que vamos a mostrar es bastante simple, el programa te pedirá 5 frases, las cuales codificará, para luego unirlas en un nodo raíz y mostrar dicha codificación en pantalla.

Comando en Java

- 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);
        }
```

Comando en Java

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

Comando en Java

```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 }
}
```

Comando en Java

- 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.

```
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;
```

Comando en Java

```
    //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);
}
```

Comando en Java

- 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.

Comando en Java

```
        //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);
    }
```

Comando en Java

- 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.

Comando en 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();
            }
```

Comando en Java

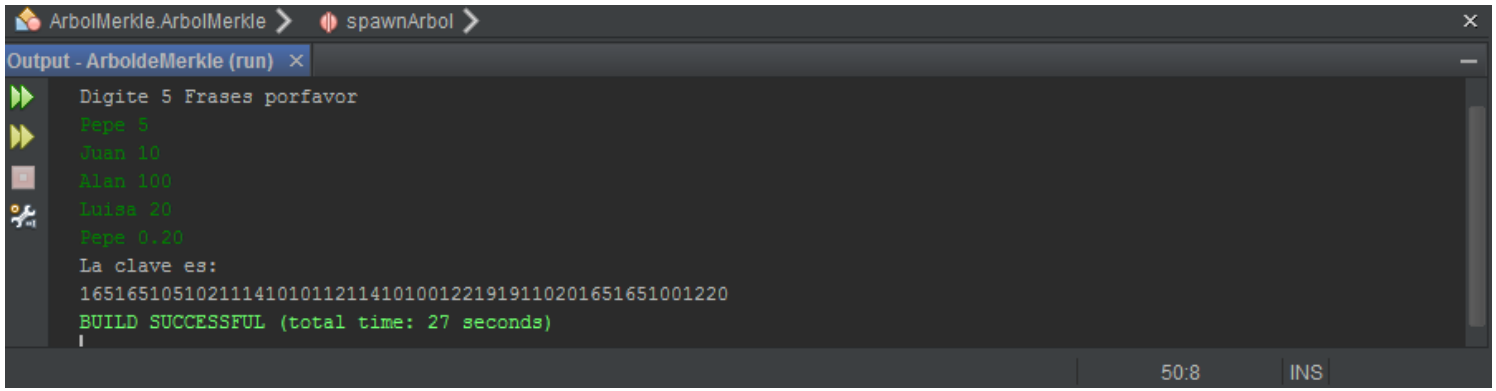
```
            Rama.add(new Nodo(hojaizq, hojader, Palo));
            i += 2;

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

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

Comando en Java

- Finalmente retornamos el nodo raíz e imprimimos su Hash.



Ralph Merkle

- Ralph Merkle fue el creador de los árboles de Hash o árboles de Merkle, pero su creación estuvo lleno de tropiezos. Muchos pensarón que dicho proyecto nunca iba a funcionar y no se atrevían a publicar sus resultados. Pero él nunca se rindió y ahora su investigación es uno de los elementos más importantes dentro de la criptografía.