Taller Opcional 2022 – Merkle Tree

Condiciones de entrega

La fecha de sugerida de entrega del taller es durante el cuatrimestre. La entrega es digital: mandar por mail a lmeiners@dc.uba.ar los archivos MerkleTree.h, MerkleTree.hpp y main.cpp. Se recomienda subir todo el código para que compile, incluyendo la carpeta test, excepto por ejecutables/binarios.

Consigna

En este taller deben implementar un *Merkle Tree*. Un *Merkle Tree* consiste en un árbol binario cuyos nodos tienen como valor el hash (de una función de hash pre-seleccionada) de la concatenación de los hashes del nodo izquierdo y derecho, y sus hojas tienen el valor del hash del dato que representan.

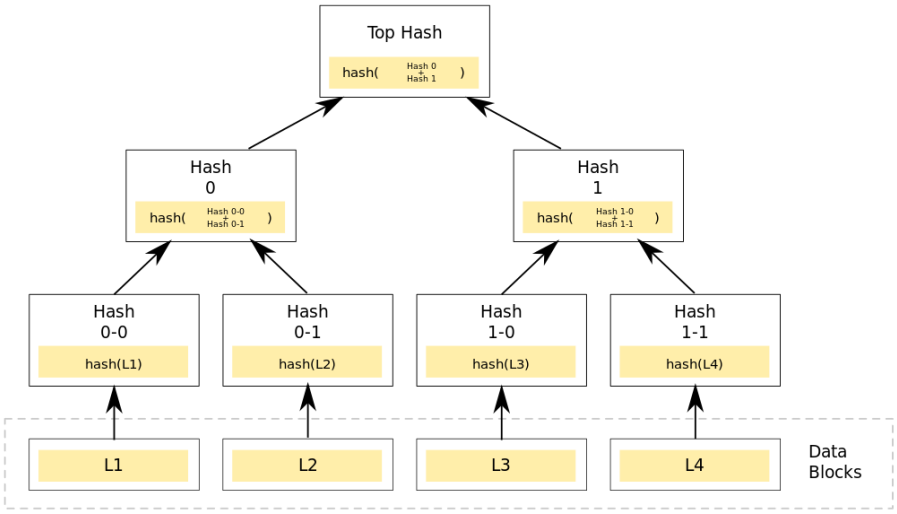

Figure 1: Merkle Tree de los Bloques de Datos L1-L4

Para resolver el taller cuentan con dos archivos: *MerkleTree.h* y *MerkleTree.hpp*. En el primero deberán completar la parte privada de la clase *MerkleTree* respetando la estructura de representación de *Merkle Tree* y en el segundo deberán completar la definición de las funciones que exporta la clase.

- MerkleTree (); Constructor por defecto de la clase MerkleTree (construye un árbol vacío).
- ~MerkleTree (); Destructor de la clase MerkleTree (debe liberar la memoria).
- **void** contruirMerkleTree (**std::vector<string>** & bloquesDatos); Arma el árbol en base al vector de datos

Además, de manera opcional, pueden completar la definición del método **void** mostrar(ostream& o) que sirve para mostrar el MerkleTree. Este método recibe como parámetro una variable de tipo ostream, que es el *output stream* sobre el que tienen que imprimir el mismo. Por ejemplo quisiéramos poder llamar a la función con el *output stream* que corresponde a la salida estándar de la siguiente manera: mi_mt.mostrar(std::cout).

La implementación que realicen **no debe perder memoria**. Recomendamos utilizar **valgrind** para testear si su implementación tiene *leaks* de memoria.