Implementation of Merkel Tree

April 21, 2023

Code (1/5)

```
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>
#include <iomanip>
#include<bits/stdc++.h>
using namespace std;
class MerkleTree {
    public:
     string hash;
    MerkleTree* parent;


    MerkleTree(const vector<string>& data_list) {
        data_list_ = data_list;
        levels_ = BuildTree();
    }
```

Code (2/5)

```
    vector<vector<string>> GetLevels() const {
        return levels_;
    }
private:
    vector<string> data_list_;
    vector<vector<string>> levels_;

    vector<vector<string>> BuildTree() {
        vector<vector<string>> levels;
        for(int i =0;i<data_list_.size();i++){
            data_list_[i]= HashNodes(data_list_[i]);
        }
        vector<string> level =data_list_;

        while (level.size() > 1) {
            levels.push_back(level);
            level = GetNextLevel(level);
        }
```

Code (3/5)

```
    levels.push_back(level);
    return levels;
}

vector<string> GetNextLevel(const vector<string>& level) {
    vector<string> next_level;
    int i = 0;

    while (i < level.size()) {
        if (i + 1 < level.size()) {
            next_level.push_back(HashNodes(level[i], level[i+1]));
            i += 2;
        } else {
            next_level.push_back(HashNodes(level[i], level[i]));
            i += 1;
        }
    }
```

Code (4/5)

```
    return next_level;
}

string HashNodes(const string& left, const string& right) {
   string ss  ;
   ss.push_back( left[left.size()-1] );
   ss.push_back(right[right.size()-1]) ;

    return ss;
}
string HashNodes(const string& left) {
   string ss  ;
   ss.push_back( left[left.size()-2] );
   ss.push_back(left[left.size()-1]) ;

    return ss;
}
```

Code (5/5)

```
};

int main()
{
    vector<string> data_list = {"hello", "world", "how", "are", "you","nara"};
    MerkleTree merkle_tree(data_list);

    vector<vector<string>> levels = merkle_tree.GetLevels();
    for (int i = 0; i < levels.size(); i++) {
        cout << "Level " << levels.size()-i-1 << ": ";
        for (int j = 0; j < levels[i].size(); j++) {
            cout << levels[i][j] << " ";
        }
        cout << endl;
    }

    return 0;
}
```

Output

```
Level 3: lo ld ow re ou ra
Level 2: od we ua
Level 1: de aa
Level 0: ea
```