
Insertion in Quad Merkle Tree

```
bool verifyQuad(string t, int pos, vector<string> &trans, unordered_map<int, vector<string>> mp, string &root){
string hash = sha256(t);
int size = trans.size();
int levels = ((ceil(log2(size) / 2)) + 1);
int i = 2;
while (i <= levels){
int prevSz = mp[i - 1].size();
int prevPos = pos;
if (prevPos % 4 == 1){
string h1 = hash;
string h2 = "", h3 = "", h4 = "";
if (prevPos + 1 <= mp[i - 1].size())
h2 = mp[i - 1][prevPos];
if (prevPos + 2 <= mp[i - 1].size())
h3 = mp[i - 1][prevPos + 1];
if (prevPos + 3 <= mp[i - 1].size())
h4 = mp[i - 1][prevPos + 2];
reverse(h1.begin(), h1.end());
reverse(h2.begin(), h2.end());
reverse(h3.begin(), h3.end());
reverse(h4.begin(), h4.end());
string str = "";
str += h4;
str += h3;
str += h2;
str += h1;
hash = sha256(str);
pos = (prevPos / 4) + (prevPos % 4 != 0 ? 1 : 0);
}
else if (prevPos % 4 == 2){
string h1 = mp[i - 1][prevPos - 2];
string h2 = hash;
string h3 = "";
string h4 = "";
if (prevPos + 1 <= mp[i - 1].size())
h3 = mp[i - 1][prevPos];
if (prevPos + 2 <= mp[i - 1].size())
h4 = mp[i - 1][prevPos + 1];
reverse(h1.begin(), h1.end());
reverse(h2.begin(), h2.end());
reverse(h3.begin(), h3.end());
reverse(h4.begin(), h4.end());
string str = "";
str += h4;
str += h3;
str += h2;
str += h1;
hash = sha256(str);
pos = (prevPos / 4) + (prevPos % 4 != 0 ? 1 : 0);
}
else if (prevPos % 4 == 3){
string h1 = mp[i - 1][prevPos - 3];
string h2 = mp[i - 1][prevPos - 2];
string h3 = hash;
string h4 = "";
if (prevPos + 1 <= mp[i - 1].size())
h4 = mp[i - 1][prevPos];
reverse(h1.begin(), h1.end());
reverse(h2.begin(), h2.end());
reverse(h3.begin(), h3.end());
reverse(h4.begin(), h4.end());
string str = "";
str += h4;
str += h3;
str += h2;
str += h1;
hash = sha256(str);
pos = (prevPos / 4) + (prevPos % 4 != 0 ? 1 : 0);
}
else if (prevPos % 4 == 0){
string h1 = mp[i - 1][prevPos - 4];
string h2 = mp[i - 1][prevPos - 3];
string h3 = mp[i - 1][prevPos - 2];
string h4 = hash;
reverse(h1.begin(), h1.end());
reverse(h2.begin(), h2.end());
reverse(h3.begin(), h3.end());
reverse(h4.begin(), h4.end());
string str = "";
str += h4;
str += h3;
str += h2;
str += h1;
hash = sha256(str);
pos = (prevPos / 4) + (prevPos % 4 != 0 ? 1 : 0);
}
i++;
}
if (hash == root)
return true;
return false;
}
----------------------------------------------------------------------------------------------------------------------------------------------------
```