Tree: Huffman Decoding

```
//
//  main.cpp
//  Huffman
//
//  Created by Vatsal Chanana

#include<bits/stdc++.h>
using namespace std;

typedef struct node {

    int freq;
    char data;
    node * left;
    node * right;

} node;

struct deref:public binary_function<node*, node*, bool> {
    bool operator()(const node * a, const node * b)const {
        return a->freq > b->freq;
    }
};

typedef priority_queue<node *,vector<node*>, deref> spq;

node * huffman_hidden(string s) {

    spq pq;
    vector<int>count(256, 0);

    for(int i = 0; i < s.length(); i++) {
        count[s[i]]++;
    }

    for(int i = 0; i < 256; i++) {

        node * n_node = new node;
        n_node->left = NULL;
        n_node->right = NULL;
        n_node->data = (char)i;
        n_node->freq = count[i];
```

```
        if( count[i] != 0 )
            pq.push(n_node);

    }

    while( pq.size() != 1 ) {

        node * left = pq.top();
        pq.pop();
        node * right = pq.top();
        pq.pop();
        node * comb = new node;
        comb->freq = left->freq + right->freq;
        comb->data = '\0';
        comb->left = left;
        comb->right = right;
        pq.push(comb);

    }

    return pq.top();

}

void print_codes_hidden(node * root, string code, map<char,
string>&mp) {

    if(root == NULL)
        return;
    if(root->data != '\0') {
        mp[root->data] = code;
    }

    print_codes_hidden( root->left, code+'0', mp );
    print_codes_hidden( root->right, code+'1', mp );

}

/*
The structure of the node is

typedef struct node
{
    int freq;
    char data;
    node * left;
```

```
    node * right;

}node;

*/


void decode_huff(node * root, string s) {
    node* curr = root;
    for(char c : s){
        if(c=='0') curr = curr->left;
        else curr = curr->right;
        if(!curr->left && !curr->right){
            cout<<curr->data;
            curr = root;
        }
    }
}

int main() {

    string s;
    std::cin >> s;

    node * tree = huffman_hidden(s);
    string code = "";

    map<char, string> mp;
    print_codes_hidden(tree, code, mp);

    string coded;

    for(int i = 0; i < s.length(); i++) {
        coded += mp[s[i]];
    }

    decode_huff(tree, coded);

    return 0;
}ffman
//
//  Created by Vatsal Chanana

#include<bits/stdc++.h>
using namespace std;
```

```
typedef struct node {

    int freq;
    char data;
    node * left;
    node * right;

} node;

struct deref:public binary_function<node*, node*, bool> {
    bool operator()(const node * a, const node * b)const {
        return a->freq > b->freq;
    }
};

typedef priority_queue<node *,vector<node*>, deref> spq;

node * huffman_hidden(string s) {

    spq pq;
    vector<int>count(256, 0);

    for(int i = 0; i < s.length(); i++) {
        count[s[i]]++;
    }

    for(int i = 0; i < 256; i++) {

        node * n_node = new node;
        n_node->left = NULL;
        n_node->right = NULL;
        n_node->data = (char)i;
        n_node->freq = count[i];

        if( count[i] != 0 )
            pq.push(n node);

    }

    while( pq.size() != 1 ) {

        node * left = pq.top();
        pq.pop();
        node * right = pq.top();
        pq.pop();
        node * comb = new node;
```

```
        comb->freq = left->freq + right->freq;
        comb->data = '\0';
        comb->left = left;
        comb->right = right;
        pq.push(comb);

    }

    return pq.top();

}

void print_codes_hidden(node * root, string code, map<char,
string>&mp) {

    if(root == NULL)
        return;
    if(root->data != '\0') {
        mp[root->data] = code;
    }

    print_codes_hidden( root->left, code+'0', mp );
    print_codes_hidden( root->right, code+'1', mp );

}

/*
The structure of the node is

typedef struct node
{
    int freq;
    char data;
    node * left;
    node * right;

}node;

*/


void decode_huff(node * root, string s) {
    node* curr = root;
    for(char c : s){
        if(c=='0') curr = curr->left;
        else curr = curr->right;
```

```
        if(!curr->left && !curr->right){
            cout<<curr->data;
            curr = root;
        }
    }
}

int main() {

    string s;
    std::cin >> s;

    node * tree = huffman_hidden(s);
    string code = "";

    map<char, string> mp;
    print_codes_hidden(tree, code, mp);

    string coded;

    for(int i = 0; i < s.length(); i++) {
        coded += mp[s[i]];
    }

    decode_huff(tree, coded);

    return 0;
}
```