**Assignment 6**

**GitHub Link**

[https://github.com/sufiyanjunaidi13/Advance\_Algorithm\_Assignment-6 (github.com)](https://github.com/sufiyanjunaidi13/Advance_Algorithm_Assignment-6)

#include <iostream>

#include <queue>

#include <unordered\_map>

using namespace std;

struct Node {

char ch;

int freq;

Node\* left;

Node\* right;

Node(char ch, int freq) {

this->ch = ch;

this->freq = freq;

left = right = nullptr;

}

~Node() {

delete left;

delete right;

}

};

struct Compare {

bool operator()(Node\* a, Node\* b) {

return a->freq > b->freq;

}

};

void preorder(Node\* root, string code, unordered\_map<char, string>& codes) {

if (root == nullptr) {

return;

}

if (root->left == nullptr && root->right == nullptr) {

codes[root->ch] = code;

}

preorder(root->left, code + "0", codes);

preorder(root->right, code + "1", codes);

}

void buildHuffmanTree(string s, int\* freq, unordered\_map<char, string>& codes) {

priority\_queue<Node\*, vector<Node\*>, Compare> pq;

for (int i = 0; i < s.size(); i++) {

pq.push(new Node(s[i], freq[i]));

}

while (pq.size() > 1) {

Node\* left = pq.top();

pq.pop();

Node\* right = pq.top();

pq.pop();

Node\* parent = new Node('\0', left->freq + right->freq);

parent->left = left;

parent->right = right;

pq.push(parent);

}

Node\* root = pq.top();

pq.pop();

preorder(root, "", codes);

delete root;

}

int main() {

string s = "abcdef";

int freq[] = {5, 9, 12, 13, 16, 45};

unordered\_map<char, string> codes;

buildHuffmanTree(s, freq, codes);

for (int i = 0; i < s.size(); i++) {

cout << s[i] << " : " << codes[s[i]] << endl;

}

return 0;

}

