**Assignment 6**

**GitHub Link:**

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

#include <iostream>

#include <queue>

#include <map>

#include <string>

using namespace std;

// A Huffman tree node

struct Node {

char data;

int freq;

Node\* left;

Node\* right;

Node(char data, int freq) : data(data), freq(freq), left(nullptr), right(nullptr) {}

};

// Comparison object to be used to order the heap

struct compare {

bool operator()(Node\* l, Node\* r) {

return l->freq > r->freq;

}

};

// Traverses the Huffman tree and stores the codes in a map

void encode(Node\* root, string code, map<char, string>& huffmanCode) {

if (!root) {

return;

}

if (!root->left && !root->right) {

huffmanCode[root->data] = code;

}

encode(root->left, code + "0", huffmanCode);

encode(root->right, code + "1", huffmanCode);

}

// Builds the Huffman tree and returns the root

Node\* buildHuffmanTree(const string& s, const int\* freq, int n) {

priority\_queue<Node\*, vector<Node\*>, compare> minHeap;

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

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

}

while (minHeap.size() > 1) {

Node\* left = minHeap.top(); minHeap.pop();

Node\* right = minHeap.top(); minHeap.pop();

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

top->left = left;

top->right = right;

minHeap.push(top);

}

return minHeap.top();

}

// Prints the codes

void printHuffmanCodes(Node\* root) {

map<char, string> huffmanCode;

encode(root, "", huffmanCode);

cout << "Huffman Codes:\n";

for (const auto& p : huffmanCode) {

cout << " " << p.first << " : " << p.second << endl;

}

}

// Driver program to test the above functions

int main() {

string s = "abcdef";

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

int n = sizeof(freq) / sizeof(freq[0]);

Node\* root = buildHuffmanTree(s, freq, n);

printHuffmanCodes(root);

return 0;

}

