2024 Data Structure - Homework 5

A1115513 劉沛辰

```cpp
#ifndef HUFFMANNODE_H
#define HUFFMANNODE_H

#include <string>
using namespace std;

class HuffmanNode {
public:
    char ch;
    int freq;
    HuffmanNode *left;
    HuffmanNode *right;
    static string huffman_code[]; // 靜態成員變數的聲明

    HuffmanNode(char ch, int freq) : ch(ch), freq(freq), left(NULL), right(NULL) {}
    HuffmanNode(char ch, int freq, HuffmanNode *left, HuffmanNode *right) : ch(ch), freq(freq), left(left), right(right) {}

    void traverse(HuffmanNode* ptr, string str) {
        if(left == nullptr && right == nullptr) {
            huffman_code[ch-'A'] = str;
            return;
        }
        if(left != nullptr) {
            left->traverse(ptr->left,str + '0');
        }
        if(right != nullptr) {
            right->traverse(ptr->right,str + '1');
        }
    }
};

#endif
```

我宣告了兩種不同的 constructor 來宣告我的 node，第一種兩個參數的是用來定義 node 的，第二個則是排序過後要建立樹的 constructor(放入父節點)，而父節點則是左右節點的總和，然後每產生一個父節點，我們必須重新再 sort 一遍，因為無法使用 vector，我想到用來刪除節點的方式是把節點移到最後面，並且少尋訪一次，達到如同刪除的效果。

而後面則是尋訪並進行編排 table 的過程，用遞迴的方式進行尋訪，只要往左節點走就多加一位 0，往右節點則多加一位 1，直到下個左右節點都為 null 時，把數值儲存在我設定的 static 變數中，可供 main 做使用。

```
bool compareFreq(const HuffmanNode* a, const HuffmanNode* b) {
    if (a == nullptr) return false;
    if (b == nullptr) return true;
    return a->freq < b->freq;
}
```

而此次的升冪排列則是自己寫的，使用 sort 函式配合 compare，來比較節點的 freq 並輸出，如上圖所示。

```
    int size=count;
    while(count > 1){
        sort(arr, arr + count, compareFreq);
        HuffmanNode *left = arr[0];
        HuffmanNode *right = arr[1];
        HuffmanNode *newroot = new HuffmanNode(' ', left->freq + right->freq, left, right);
        arr[0] = newroot;
        swap(arr[1], arr[count-1]);
        count--;
    }

    HuffmanNode *root = arr[0];
    root->traverse(root,"");

    for(int i = 0; i < size; i++) {
        cout<<char('A'+i)<<" "<<HuffmanNode::huffman_code[i]<<endl;
    }

    for(int j = 0; j < str.size(); j++) {
        cout<<HuffmanNode::huffman_code[str[j]-'A'];
    }


    return 0;
}
```

這張圖就是上面所述的建立樹的方式，每次要增加節點時要先進行 sort，取前兩位進行權重相加，然後再讓第二位和當下的最後一位互換以達到刪除的效果，因為是由末端進行減少，第一位會變成最高的 root 節點，所以我們直接拿 root 進行遞迴尋訪，然後之後輸出 table 和被 encode 的字串。