程式作業 4

1. 演算法設計

   結合指標，以及使用者自訂 node 結構，並利用 priority queue 這個本身具有優先順序特性的資料結構，來方便儲存我們需要的二元樹，最終使用遞迴式，以及指標特性進行正確的回推並輸出

2. Pseudo code

```
void printNode (CombinNode *pt, string output){       //輸出函式
    if(pt == nullptr)
        return
    if(pt->left != nullptr)          //左節點輸出加 0
        output += "0"
        printNode (pt->left,output)        //遞迴
    if(!pt->left && !pt->right)          //列印
        printf("%c ",pt->key)
        for(int i=0;i<output.size();i++)
            cout<<output[i]
    output.pop_back()
    if(pt->right != nullptr){      //右節點輸出加 1
        output += "1"
        printNode (pt->right,output)            //遞迴
main()
    for(int i=0;i<Q;i++)          //讀取輸入
        cin>>word>>num
        CombinNode *pt = new CombinNode
        pt->key = word
        pt->freq = num
        priority_queue.push(pt)             //輸入放入 priority_queue，進行排序
    for(int i=1;i<Q;i++)          //頻率小合併為頻率大
        CombinNode *pt2,*pleft,*pright
        pleft = priority_queue.top();priority_queue.pop()
        pright = priority_queue.top();priority_queue.pop()
        pt2->freq = pleft->freq+pright->freq
        pt2->left = pleft
        pt2->right = pright
        priority_queue.push(pt2)
    string output = " "
    printNode (priority_queue.top(),output)
```

3. 時間複雜度分析

主要工作區使用單層迴圈時間複雜度為 O(n)，遞迴式最多也只跑 n 次，因此最終時間複雜度分析為 O(n)。