編譯環境：CODEBLOCKS 16.01

Release 16.01 rev 10702 (2016-01-25 19:50:14) gcc 4.9.2 Windows/unicode - 32 bit

參考資料：

<http://www.sanfoundry.com/cpp-program-implement-binary-tree-2/>

<http://blog.csdn.net/here1009/article/details/8071251>

心得：

Binary search tree 由Insert(包含search)、Inorder、Postorder，3個function去組成，讀檔的部分很麻煩，因為和以前不同，這次還有：和、，所以不能用getline直接讀取整行，要分開，遇到空格、符號時要分別讀取做判斷

至於霍夫曼編碼課本寫太少，雖然知道往左子樹:0，往右子樹:1的概念，我還是不知道如何下手，便查詢超多資料，大部分都是不能用的，不是struct node中的 leftchild->data 在進行運算時超級複雜，幾乎接近300行的function完全不能解讀，最後與同學討論找到這篇

1. #include<iostream>
2. #include<string>
3. #include<queue>
4. **using** **namespace** std;
6. **class** node{
7. **public**:
8. node(string con, **float** wht, node\* left, node\* right, string co ){
9. content=con;
10. weight=wht;
11. leftchild=left;
12. rightchild=right;
13. code=co;
14. }
15. string content;
16. **float** weight;
17. node\* leftchild;
18. node\* rightchild;
19. string code;
20. };
22. **void** insertion\_sort(node\*\* array, **int** low, **int** high){
23. **for**(**int** i=low+1;i<high;i++){
24. node\* tem=array[i];
25. **int** j=i-1;
26. **while**(array[j]->weight>tem->weight&&j>=low){
27. array[j+1]=array[j];
28. j--;
29. }
30. array[j+1]=tem;
31. }
32. }
33. **void** create\_huffman\_tree(string\* s, **float**\* w,**int** n,node\*\* array){
34. **for**(**int** i=0;i<n;i++){
35. array[i]=**new** node(s[i],w[i],NULL,NULL,"");
36. }
37. insertion\_sort(array,0,n);
38. //~ for(int i=0;i<n;i++){
39. //~ cout<<array[i]->content<<"\*";
40. //~ }
41. **int** p=0;
42. **while**(p!=n-1){
43. node\* min\_1=array[p];
44. node\* min\_2=array[p+1];
45. node\* new\_node=**new** node("",min\_1->weight+min\_2->weight,min\_1,min\_2,"");
46. //cout<<new\_node->weight<<endl;
47. array[p+1]=new\_node;
48. p=p+1;
49. insertion\_sort(array,p,n);
50. }
52. }
54. **void** create\_huffman\_code(node\* p){
55. queue<node\*> nq;
56. nq.push(p);
57. **while**(nq.size()!=0){
58. node\* cur=nq.front();
59. nq.pop();
60. node\* l=cur->leftchild;
61. **if**(l!=NULL){l->code=cur->code+"0"; nq.push(l);}
62. node\* r=cur->rightchild;
63. **if**(r!=NULL){r->code=cur->code+"1"; nq.push(r);}
64. **if**(l==NULL&&r==NULL){
65. cout<<cur->content<<": "<<cur->code<<endl;
66. }
67. }
68. }
69. **int** main(**int** argc, **char**\*\* argv){
70. node\* array[8];
71. string s[8]={"a","b","c","d","e","f","g","h"};
72. **float** w[8]={1,1,2,3,5,8,13,21};
73. create\_huffman\_tree(s,w,8,array);
74. create\_huffman\_code(array[7]);
75. }

程式碼基本上將main中改為讀檔

但這還是失敗，output無論如何都是空白

於是我便用同學的方法

將Input中的計算行數和行中的位置分開去讀

1個STR1， 1個STR2

避免數值在運算過程跑掉

fs.open("Input2.txt",fstream::in); //open the file

fp.open("Output2.txt",fstream::out|fstream::trunc); //write the file

int line = 0;

while(getline(fs,str1)){

line++; //算幾行

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*分開的目的是為了防止2邊的檔案互相混雜\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*先算Input有幾行 再計算Input中的某行 他其中的字串\*\*\*\*\*/

fs.close();

fs.open("Input2.txt",fstream::in);