

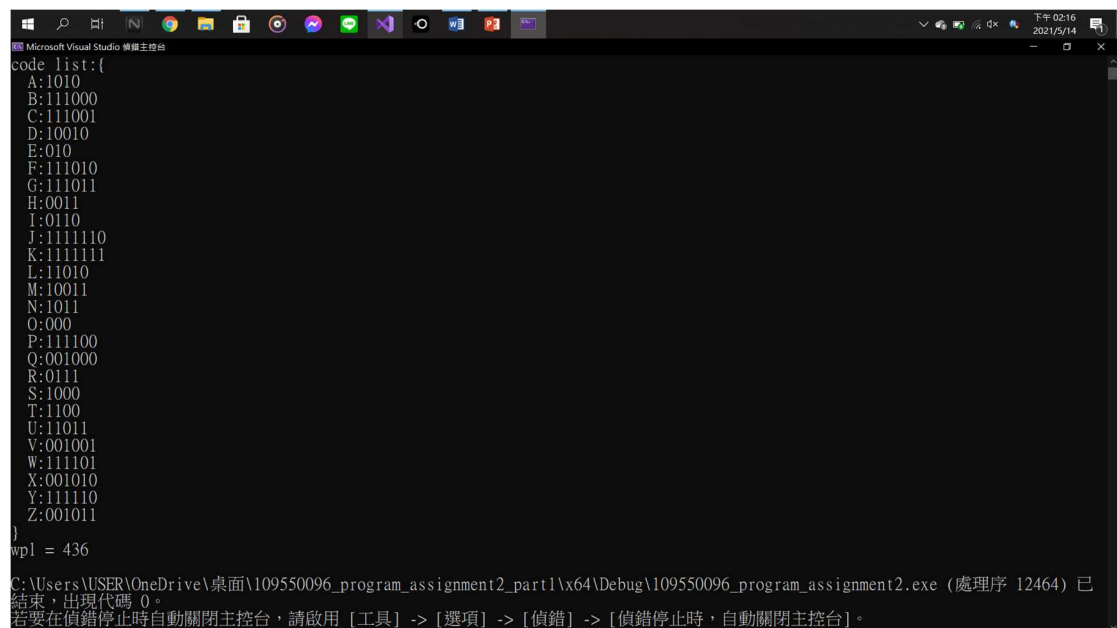
Student number: 109550096 Name: Du Feng

Program Assignment 2-Report

1. Huffman encoding is a kind of code which is used to compress file. In this compressing way, it does not lose any detail. At first, calculating how many times each character appears in the string, which also means its “frequency”. With these “frequency”, we can create a binary-search tree. From the root of the tree, if we go to left, encoding with ‘0’. If we go to right, encoding with ‘1’. Repeat the above action until arriving the leaf of the tree.

When all the leaves are visited, we can create a code list corresponding to the original string. Then we use this code list to transform the original string to a string which is only with ‘0’ and ‘1’. Then we can reach our target, “compressing the data”.

2. Part 1’s result:



```
code list:{
A:1010
B:111000
C:111001
D:10010
E:010
F:111010
G:111011
H:0011
I:0110
J:1111110
K:1111111
L:11010
M:10011
N:1011
O:000
P:111100
Q:001000
R:0111
S:1000
T:1100
U:11011
V:001001
W:111101
X:001010
Y:111110
Z:001011
}
wpl = 436
C:\Users\USER\OneDrive\桌面\109550096_program_assignment2_part1\x64\Debug\109550096_program_assignment2.exe (處理序 12464) 已結束，出現代碼 0。
若要在偵錯停止時自動關閉主控台，請啟用 [工具] -> [選項] -> [偵錯] -> [偵錯停止時，自動關閉主控台]。
```

3. Part 2’s result:

First test string: iaaaaaaamhhhhannnnddsssomeeeee

```
Microsoft Visual Studio 偵錯主控台
Enter characters:iaaaaaaamhhhhannnddssomeeeee
encoding result:00000101010101010111011001001000110110110111001100001001001000111011111111111111111
code list:{
  a:01
  d:1100
  e:111
  h:100
  i:0000
  m:1101
  n:101
  o:0001
  s:001
}
wpl = 88
decoding result:iaaaaaaamhhhhannnddssomeeeee

C:\Users\USER\OneDrive\桌面\109550096_program_assignment2_part2\Debug\109550096_program_assignment2_part2.exe (處理序 22936)
已結束，出現代碼 0。
若要在偵錯停止時自動關閉主控台，請啟用 [工具] -> [選項] -> [偵錯] -> [偵錯停止時，自動關閉主控台]。
按任意鍵關閉此視窗...
```

Second test string: howwwarreyooou

```
Microsoft Visual Studio 偵錯主控台
Enter characters:howwwarreyooou
encoding result:11001000000000110111111111011101010101101
code list:{
  a:0110
  e:0111
  h:1100
  o:10
  r:111
  u:1101
  w:00
  y:010
}
wpl = 42
decoding result:howwwarreyooou

C:\Users\USER\OneDrive\桌面\主豐堅_program assignment2\109550096_program_assignment2_part2\Debug\109550096_program_assignment2_part2.exe (處理序 18156) 已結束，出現代碼 0。
若要在偵錯停止時自動關閉主控台，請啟用 [工具] -> [選項] -> [偵錯] -> [偵錯停止時，自動關閉主控台]。
按任意鍵關閉此視窗...
```

4. Part 1:

1. Create 2 array to restore the data.
Ex. char={'A','B',.....,'Z'} and int={7,2,2,.....,2,1}
2. Add these data to a dequeue, then sort this dequeue by the number(frequency) of the data.
3. Pop two of the smallest data from the dequeue to create a node and give this node a value which equals to the sum of the two data's number(frequency), and push this node back to the dequeue. Continue this step until the size of the dequeue equals to 1, which also means there is a tree in the queue.

4. After creating a tree of the data, search this tree from the root. If we go to the left, we encode with '0'. If we go to the right, we encode with '1'. Repeat this step until arriving the leaf of the tree. Then assign this string(Ex. '10110' to the character of the leaf).
5. Repeating step 4 until all the leaves are visited.
6. Sort data after step 5. Then we can get the code list.
7. And the sum of each data's number(frequency)*length of code will be the WPL.

Part2:

1. Getting the input string character by character, and store them in the "map". If this character does not exist in the map, put this character to the map, and set the number to 1. If it has already existed, add its number by 1.
2. Then add this map to the dequeue 1 by 1.
3. Apply the steps 3 to 5. Then we can get the code list.
4. Create a new map to store the code list, and transform the input string character by character to the code with '0' and '1' by the new map.
5. The method to calculate WPL is equal to the step 7 in the part 1.
6. For decoding the string with '0' and '1':
 - a. Create a null string.
Ex. string temp = "";
 - b. Add the first character of the string to the temp string, and compare to the element's code string in the map which is created at step 4. If none of the element in the map can match with the temp string, add next character of the string to the temp string, and compare to the map again until get the match. After getting the match, print the character of the matched element.
 - c. Repeat the step 6-a and 6-b until all the characters in the string have been decode. Then we can get decoding string.