Algorithm Project 2 - Pickle’s trouble

1. Description

Pickle is a lovely high school girl. She wants to show her love to a handsome and humor guy who has a cute nickname, Widthy. However, Pickle is too shy to write down the word directly. As a result, she comes up with a great idea -- using **dynamic Huffman coding** to encode her word! On the one hand, her word would not be seen by her classmate, on the other hand, she could use fewer character to send large amount of messages to Widthy (That’s right, she has a lot of words to tell him).

However, she is facing some problems. This encoding method is complex and hard to realize. Furthermore, Widthy doesn’t have the time to figure out what Pickle want to tell him because he spends lots of time watching the TV show Dora. Pickle calls you to help, since you’re the only one whom she can trust and she knows that you’re a master at programming. She will appreciate you very much if you solve her problem.

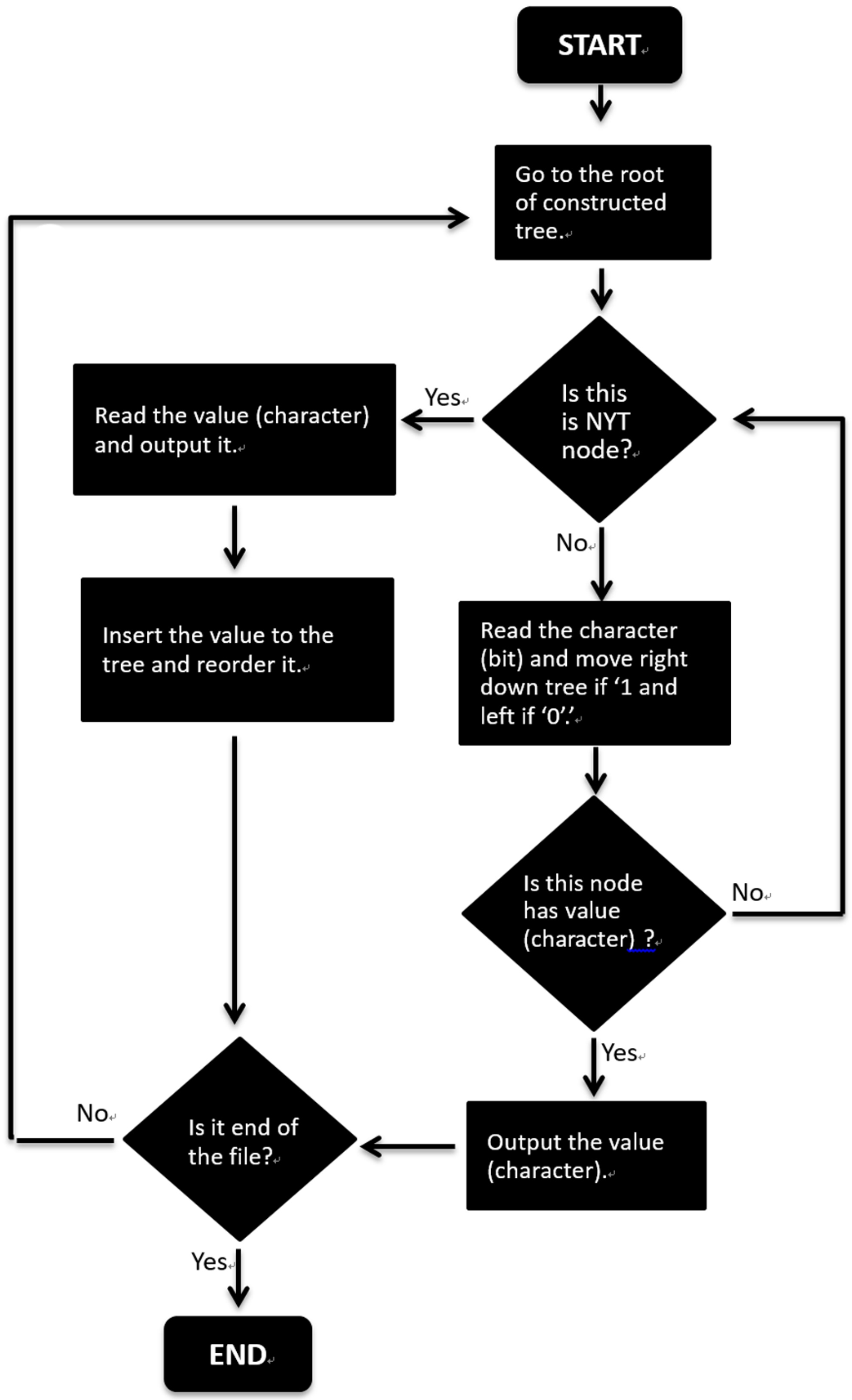
2. Method

In dynamic Huffman coding, you need to build a binary tree. Every node in the tree has its weight and order. When a new character is added to the tree, the node’s weight and its order will be adjusted in the same time. There’re some definitions to build the binary tree:

- Each node has a sibling
- Node's with higher weights have higher orders
- On each level, the node farthest to the right will have the highest order although there might be other nodes with equal weight
- Leaf nodes contain character values, except the Not Yet Transmitted(NYT) node which is the node whereat all new characters are added
- Internal nodes contain weights equal to the sum of their children's weights
- All nodes of the same weight will be in consecutive order.

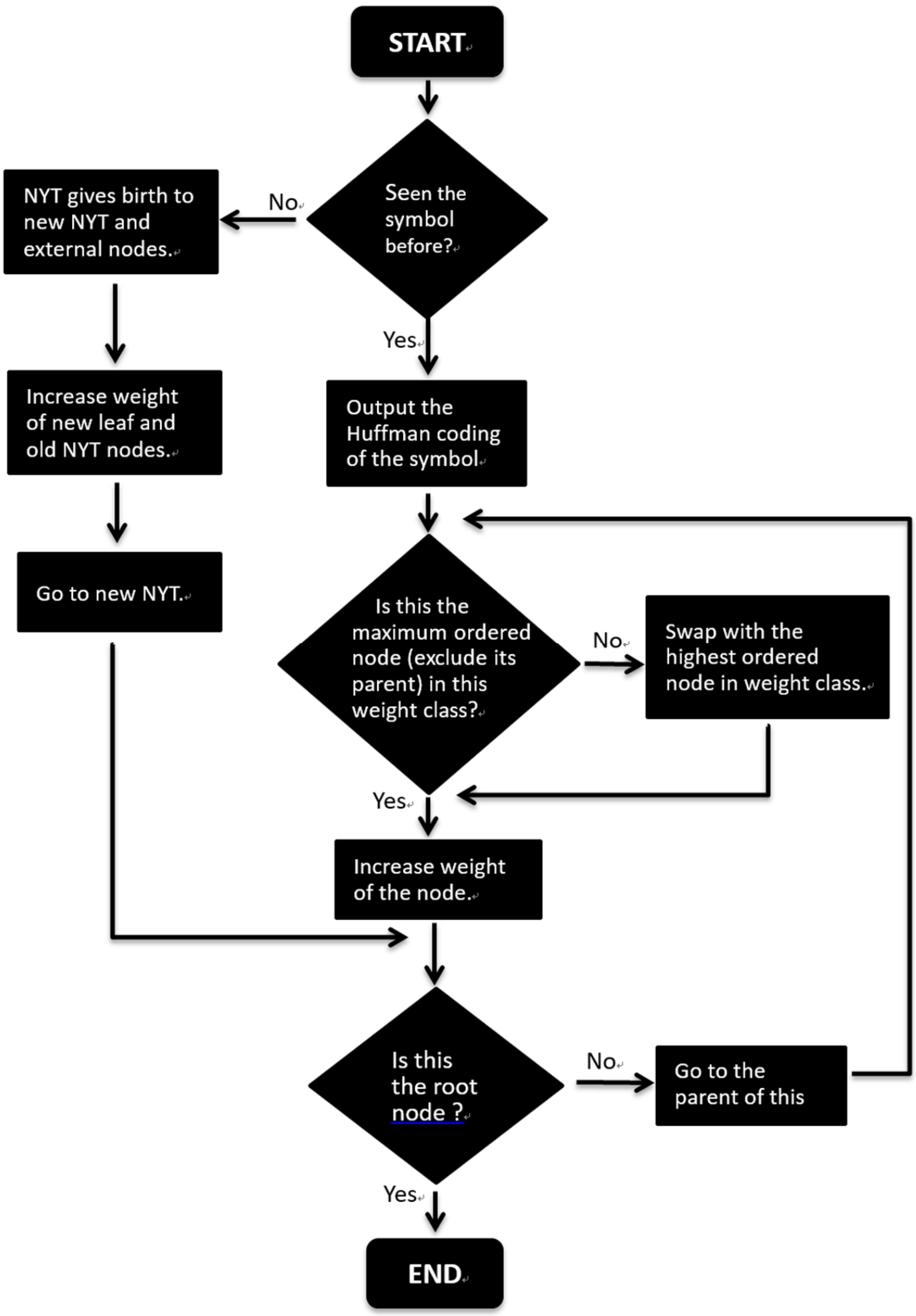
There’re the flow chart of decode and encode to build the tree and do the search.

Decode:



START
Go to the root of constructed tree.
Is this is NYT node?
Yes
Read the value (character) and output it.
Insert the value to the tree and reorder it.
No
Read the character (bit) and move right down tree if '1 and left if '0'.
Is this node has value (character) ?
No
Yes
Output the value (character).
Is it end of the file?
No
Yes
END

Encode:


START
Seen the symbol before?
No
NYT gives birth to new NYT and external nodes.
Increase weight of new leaf and old NYT nodes.
Go to new NYT.
Yes
Output the Huffman coding of the symbol
Is this the maximum ordered node (exclude its parent) in this weight class?
No
Swap with the highest ordered node in weight class.
Yes
Increase weight of the node.
Is this the root node ?
No
Go to the parent of this
Yes
END

3. Input/ Output

In this project you should implement the dynamic Huffman encoding and decoding method. You should read the file and output the decoding/encoding result of that file’s content. There are two parts of this assignment: For the first part, you have to make your program execute text-only input/output files correctly, and for the second part (the bonus part), your program should read any type of files and encode/decode them bit by bit.

The test data will use standard input in this format: **[D/E] [input filename] [output filename].** If the first character is ‘D’, it means that you should decode the file “input filename” and output your result to the output file “output filename”. If the first character is ‘E’, then you should encode the input file and output to the output file, respectively.

For the first part, you will only need to read the text file and simply output another text file. The file consists of uppercase and lowercase letter only. If the encode result have the left bit, simply output the character ‘0’; if the encode result have the right bit, simply output ‘1’. If there are a new character insert to the tree, then just show the path to the NYT node, and then output the character value you read.

In the second part, you should read any type of file and output the result bit by bit. In the encoding case, you should read 8-bits as a unit, and make these bits stored in the Huffman tree. However, there is a problem, since the file’s minimum unit is byte, not the bit, the encode result may not end to the last bit of the byte. How to make the program know when the file has ended? Here is the solution: you could consider the EOF as a new character to be added to the tree. Every character’s range is 0(0000 0000) ~ 255(1111 1111) originally, you could add an addition bit before the original 8-bits character. That is, every new character should encode to 9-bits. If the first bit is 0, then the next 8 bits is the character; if the first bit is 1, then it means that the file reaches its end.

We will test your program on the workstation, make sure it can run successfully before you hand in the project.

4. Sample testing data:

Part1 (text only):

Input command(standard input):
E test1.txt encode.txt
test1.txt
abcddbb
Output (encode.txt):
a0b00c100d00100110

Input command(standard input):
D test2.txt decode.txt
test2.txt
w0i00d100t000h1100y001000m010000s1010100111100o10000u
Output (decode.txt):
widthyimissyou

Part2 (Any file type):

Input command(standard input):
E t1.bmp t1.dat
Output (t1.dat):
Reference to the file in the folder.

Input command(standard input):
D t1.dat t2.bmp
Output (t2.bmp):
Should be same as t1.bmp.

5. Requirements

Program

I. You need to turn in the codes. “Part1.cpp” and “Part2.cpp”

II. Your program must be readable (Ex. Comments, variable names, function names)

III. STL structure is available (Stack, queue, etc.).

Report

I. Name the file"Project2_StudentID.pdf", Ex: Project2_0316000.pdf)

II. Describe your implementation. (Ex: algorithm, program executing process)

III. What is the time complexity/ space complexity of the decoding/ encoding method?

IV. In what case can the dynamic Huffman coding has a better compression ratio? That is, in what situation, this method can make the input file much smaller?

6. Grading policy

I. Part1 (text only data) (80%)

Your code can run without crashing and can give correct results to the sample files in your report.

Decode 40% + Encode 40%

II. Part2 - Bonus (Any file type) (30%)

III. Report (20%)

If there are any undesirable mistakes to the requirement. Each mistake minus 10 points.

7. Submit (e3 will be closed on time)

Compress all your files (including your code(.h/.cpp) and report(.pdf)) and name your compressed file as "**Project2_studentID.zip**". Upload your compressed file to e3.

Deadline: 2016.6.20, 23:59

No late upload.