Discussion 5

PA3 Huffman encoding

Ning Liu
The whole picture

- Use Huffman coding and decoding technique to compress a large file to a smaller file and recover the large file with the acquired smaller file.
Huffman coding

Suppose in the given file, the content is like this: 
AAAAAAABBBBBCCCCCCGH

Then how many times does each character occur in the file?
Building the coding tree: the initial forest
Building the coding tree: forest after step 1

6
A

4
B

4
C

1
G

2
H
Building the coding tree: forest after step 2
Building the coding tree: forest after step 3

- **10**
  - **6** (A)
  - **4** (B)

- **7**
  - **4** (C)

- **3**
  - **1** (G)
  - **2** (H)
Building the coding tree: done!

```
  17
   /\          /
  10  7          3
 /   \        /   /
6 A   4 B     4 C  1 G
  \   /       \   /
   1 \     1 0   1 \  0
     0     1     0
```

6 (A), 4 (B), 4 (C), 1 (G), 2 (H)
The compressed version of the original file

- 111111111111010101001010101001000
- Only 37 bits are needed
- But originally, we need 17 bytes, which is 136 bits!
One detail

• You are standing at the root and want to code ‘A’, then how can you figure out which path leads to ‘A’ so that you can code it?

• Start from ‘A’ and go to the root, then the code is in the reverse order.
Huffman decoding
The compressed file

- 111111111111101010100101010101001001000
The Huffman tree

Building the coding tree: done!

```
          17
           |
         1 0
        /    \
       1 1
      /     \
     10     7
    /  \
   /    \
   6 A  4 B
      /    \
     /      \
     1 0 1 0
     /       \
    /         \
   4 C        3
   /          /   \
  / A        1 G   2 H
```
PA3-compress a file
How to compress any kinds of files

- You will be asked to compress jpg, pdf, dat files and so on.
- Use the flag `ios::binary` when you open a file, to specify that the file stream is a binary stream.
- `ifstream in;
  in.open(filename, ios::binary);
  ofstream out;
  out.open(filename, ios::binary);`
Always read a byte per time

- Always read a byte from the file.
- Since one byte has 8 bits, then we have 256 different kinds of bytes.
- Count the frequencies of each byte to construct the Huffman coding tree.
- Then code the file to its compressed version.
Detail one

• How to write the compressed version, namely, a string of bits into files?

• Need to implement a “writeBit” function for bit-by-bit writing.
Detail two

How to save the established Huffman coding tree in the compressed file?
The file header

• Probably the easiest way to do it is to save the frequency counts of the bytes in the original uncompressed file as a sequence of 256 ints.
• Or you can try some smarter ways!
• Recover the Huffman Tree by the header when we uncompres the file
PA3-uncompress a file
Recover the Huffman tree

- When trying to recover the original file from the compressed file, your first task is to rebuild the Huffman tree with the information in the compressed file.

- Yeah, you are given only the compressed file and no more.
Recover the original content

• Have to implement a “bitRead” function to read the compressed file bit-by-bit.

• Then with the help of the recovered Huffman tree, you can recover the original file.
How to start with your PA3 Compress

• Step 1
Read the original file
• Step 2
Count how many times each character appears in the file
• Step 3
Build the huffman tree.
• Step 4
Encode the characters from the given file using huffman tree and write it to the output file
• Step 5
Now you get the compressed file! Compare it with the reference compression we give you.
Uncompress

- **Step 1**
  Read the compressed file

- **Step 2**
  Reconstruct the huffman tree from the header of the compressed file

- **Step 3**
  Decode the compressed file using the huffman tree and write it to the uncompressed file

- **Step 4**
  Now you get the uncompressed file! Compare it with the original file. These two files need to be exactly the same!
  The way to check is to run `diff` command.
What to submit?

- For Huffman tree: HCNode.hpp, HCNode.cpp, HCTree.hpp, HCTree.cpp,
- For bitwise operation: BitInputStream.hpp, BitInputStream.cpp, BitOutputStream.hpp, BitOutputStream.cpp
- For compress your file: compress.cpp.
- For uncompressed your file: uncompress.cpp,
In all

- compress.cpp, uncompress.cpp, HCNode.hpp, HCNode.cpp, HCTree.hpp, HCTree.cpp, BitInputStream.hpp, BitInputStream.cpp, BitOutputStream.hpp, BitOutputStream.cpp