# Assignment 6: Huffman Encoding YEAH

Ashwin Siripurapu

May 30, 2013

- Q. What's `ext_char`?

  A. Recall that nodes in the Huffman tree can have three `character` values: they can either contain a true character, the special value `NOT_A_CHAR` (containd by all interior nodes in the tree), or the special value `PSEUDO_EOF` (contained by exactly one of the leaf nodes and used to signal the end of the file).

  We can't contain these three types of values in the regular old `char` type, because `char` itself can only contain the regular characters, and not the two special values. Therefore, we created the extended character type `ext_char` which is big enough to contain all these values.

- Again, you have broad freedom to implement the interface that we specify for you. Think hard about any design choices that you have to make.

  One specific design choice that will be of paramount importance is this: when you've finished building the encoding tree to compress the input file, you'll be ready to compress the input file character by character. But, how will you find the encoding that corresponds to a given character of input? One way to do this is to run a depth– or breadth–first search through the entire tree to find the character you want, and keep track of the sequence of left/right moves that led to it. Another way is to make one exhaustive search through the entire tree and build up a map that goes from each character to its compressed bit pattern. What are the tradeoffs involved? Which way is better?

- The first thing you do when compressing the file is read through it to figure out the frequency counts for each character. After doing this,

the input file pointer will be at the very end of the file (duh). But then, after you build up the tree, you need to go back to the beginning of the file to start writing out the compressed version of each character in the input file.

To go back to the beginning of an `ibstream` object (for our purposes, a file), you can use the `rewind` method, like so:

```
ifbstream infile(filename.c_str()); // ifbstream is a subclass of ibstream

// read through the file
process(infile);
// now infile is at the end of the file

infile.rewind();
/* now infile is at the beginning of the file again
  ready for a second pass! */
```

Note: an earlier version of these notes told you that the `istream` class supports member function `rewind`. This is *incorrect!* Only the `ibstream` class has the `rewind` method. You will have to take this into account when rewinding your stream. Also note that `ibstream` is a subclass of `istream`, so any method that takes an `istream` can accept an `ibstream` object. Keep this in mind and rewinding should be a snap.