Source Coding - Decoding

Information Theory Lab 5

Objective

Understand linear block codes encoding and decoding by implementing a basic decoding application.

Theoretical notions

See lecture notes.

Exercises

1. Write a C program to perform decoding of the files encoded from the previous laboratory (given separately in a `.zip` file). The program shall be called as follows:

   ```
   Decode.exe code.dat input.txt output.txt
   ```

- The arguments are:
  - `code.dat`: a file containing the code to be used (known as the "codebook" file)
  - `input.txt`: the encoded file
  - `output.txt`: the output file (decoded)

The codebook file contains a vector of 256 elements of the following structure type:

```
typedef struct
{
    int len;                /* length of code, in bits */
    unsigned long code;     /* the first "len" bits are the codeword */
} CODE32BIT;
```

- The program will follow the following steps:
  - Read the full vector from the codebook file;

  - Read the full input encoded file into an array `data` of type `unsigned char`, of max size 1MB (i.e. 1.000.000 bytes);
  - Store the number of bytes actually read (return value of `fread`), so that you know how much of the array is actually used;
  - Decode the characters from the `data` array, as follows:
    * While we haven't processed all bytes actually read, do the following:
      · For all characters, try and see which codeword matches the next bits in `data`;
      · When a codeword matches, write that character in the output file;
      · Advance in the `data` array with the size of the matched codeword;
- **Note**: every time there is a **single codeword** that fully matches the next bits in the `data` array.

2. Decode the file `textro.enc` with the provided codebook `codero.dat`. Open the output and check that the data is recovered correctly. Check the size of the input and output files and compute the compression ratio.

3. Repeat 2. for `texten.txt` with codebook `codeen.dat`.

4. Decode a file with the codebook from the other language. How does the output look like?

5. Open an encoded file and randomly make an error (for example, delete one character). Then attempt to decode it. How does the error affect the decoded output?

Implementation hints

- The following C functions may be used for file-based operations. Look up their documentation on the Internet (e.g. *cplusplus.com*, or Google search).
  - `fopen(...)`, to open a file for reading;
  - `fread(...)`, to read byte data from the file;
  - `fclose()`, to close the file when finished.
- The following macros implement the basic bit operations:
  - reading a single bit i from a number x;
  - set bit i from a number x to 1;
  - clear bit i from a number x (i.e. set to value 0);
  - change the value of bit i from a number x (i.e. if 0 make 1, if 1 make 0).

```
#define READ_BIT(x,i)       (((x) & (1U << (i))) != 0)
#define SET_BIT(x,i)        ((x) = (x) | (1U << (i)))
#define CLEAR_BIT(x,i)      ((x) = (x) & ~(1U << (i)))
#define TOGGLE_BIT(x,i)     ((x) = (x) ^ (1U << (i)))
#define WRITE_BIT(x,i,val)  ((val) ? SET_BIT((x),(i)) : CLEAR_BIT((x),(i)))

#define VECREAD_BIT(v,i)       (READ_BIT((v[(i)/8]),(i)%8))
#define VECWRITE_BIT(v,i,val)  (WRITE_BIT((v[(i)/8]),
```

- When writing the i-th bit in a large vector of bytes, `i/8` is the index of the byte and `i%8` is the bit inside that byte.
  - Example: the 20th bit in vector `out` = bit number 4 from byte `out[2]`.

Final questions

1. TBD
2. TBD