Arithmetic Coding (Implementation #2)

Encoder

A symbol is encoded by using a specific array of integers (or a model) and by calling the following procedure. The values of low, high, and opposite_bits are initialized to 0, Top, and 0, respectively. The model is specified through cum_freq[], where cum_freq[0] acts as a scale factor. The symbol is indexed as 1...N.

```c
#define c 8
#define Top (2^c-1)
#define Qtr (Top/4+1)
#define Half (2*Qtr)
#define 3Qtr (3*Qtr)

static long low, high, opposite_bits, range;

void encode_a_symbol(int index, int cum_freq[]) {
    range = high - low + 1;
    high = low + (range * cum_freq[index-1]) / cum_freq[0] - 1;
    low  = low + (range * cum_freq[index])  / cum_freq[0];
    for ( ; ; ) {
        if (high < Half) {
            send out a bit “0”;
            while (opposite_bits > 0) {
                send out a bit “1”;
                opposite_bits--;
            }
        }
        else if (low >= Half) {
            send out a bit “1”;
            while (opposite_bits > 0) {
                send out a bit “0”;
                opposite_bits--;
            }
            low -= Half;
            high -= Half;
        }
        else if (low >= Qtr && high < 3Qtr) {
            opposite_bits += 1;
            low -= Qtr;
            high -= Qtr;
        }
        else break;

        low  = 2 * low;
        high = 2 * high+1;
    }
}
```

At the of the coding process, the encoder is flushed by calling the following procedure:

Flush at the Encoder
void encoder_flush()
{
    opposite_bits++;
    if (low < Qtr) {
        send out a bit "0";
        while (opposite_bits > 0) {
            send out a bit "1";
            opposite_bits--;
        }
    } else {
        send out a bit "1";
        while (opposite_bits > 0) {
            send out a bit "0";
            opposite_bits--;
        }
    }
    low = 0;
    high = Top;
}

Decoder

A symbol is decoded by using the model and by calling the following procedure.

static long low, high, value, bit, range, index, cum;

int decode_a_symbol(int cum_freq[ ])
{
    range = high - low + 1;
    cum = ( (value - low + 1) * cum_freq[0] - 1) / range;
    find index such that cum_freq[index] <= cum < cum_freq[index-1];
    high = low + (range * cum_freq[index-1]) / cum_freq[0] - 1;
    low = low + (range * cum_freq[index]) / cum_freq[0];
    for ( ; ; ) {
        if (high < Half);
        else if (low >= Half) {
            value -= Half;
            low -= Half;
            high -= Half;
        } else if (low >= Qtr && high < 3Qtr) {
            value -= Qtr;
            low -= Qtr;
            high -= Qtr;
        } else break;

        low = 2 * low;
        high = 2 * high + 1;
        get one bit;
        value = 2 * value + bit;
    }
    return (index);
}
Again the model is specified through \texttt{cum\_freq[].} The decoded symbol is returned through its index in the model. The decoder is initialized to start decoding an arithmetic coded bitstream by calling the following procedure:

\textbf{Initialization at the Decoder}

\begin{verbatim}
void decoder_reset( )
{
    value = 0;
    low = 0;
    high = Top;
    for (int i = 1; i <= c; i++) {
        get one bit;
        value = 2 * value + bit;
    }
}
\end{verbatim}

\texttt{NOTE: If there is no more bit to get, set bit = 1.}