ITCT Lecture 6.2:
Research Topics on Huffman Codes

- Huffman Codes with Constrained Length

What we have done!!

- Optimal Huffman Codes for Arbitrary-side Growing Trees
- Reversible Variable Length Codes- Huffman code based Approach

Huffman codes with constrained length

- Due to lookup table size constraints, no codeword length exceeds L bits is allowed.

$$\begin{Bmatrix} S_1, & S_2, & \cdots, & S_N \\ P_1, & P_2, & \cdots, & P_N \end{Bmatrix}, \quad \text{where } P_1 \geq P_2 \geq \cdots \geq P_N$$

the design procedure for a shortened Huffman code:

1. Partition S into two sets S_1 and S_2 as

$$S_1 = s_i \mid p_i > \frac{1}{2^L}$$

$$S_2 = s_i \mid p_i \leq \frac{1}{2^L}$$

2. Create a special symbol Q such that its freq. of occurrence is

$$q = \sum_{i \in S_2} p_i$$

3. Augment S_1 by Q to form a new set W. Construct an optimal prefix code for W using the design procedure for unconstrained length Huffman codewords.
   $\rightarrow$ codewords c_s, for symbols in the set S_1
   codeword c_q for the symbols Q.
   c_q is the shortened prefix-code for symbols in S_2

   If l_i is the length of the i-th codeword of S_1, then

$$\max_{s_i \in S_1} l_i = \max_{s_i \in S_1} \left\lceil \log_2 \frac{1}{p_i} \right\rceil \leq L$$

   ($\lceil x \rceil$ denotes the smallest integer larger or equal to x)

Encoding : input message string $m_1, m_2, \ldots, m_k$

- For all $m_i \in S_1$, output the corresponding codeword from C_{s_1} .
- For all $m_i \in S_2$, output the codeword c_q followed by an L-bit fixed-length binary representation for m_i. (Actually, one can use fewer than L bits, since if there are N_{s_2} symbols in S_2 and $N_{s_2} < N$, then the fixed-length binary representation for each m_i is $\lceil \log_2 N_{s_2} \rceil$ bits).

Let l_{sh} be the average codeword length for the shortened codes.
l_w be the average codeword length for W.

$$H(w) = \sum_{i \in S_1} p_i \log_2 \frac{1}{p_i} + q \log_2 \frac{1}{q}$$

$$H(s) = \sum_{i \in S_1} p_i \log_2 \frac{1}{p_i} + \sum_{i \in S_2} p_i \log_2 \frac{1}{p_i} \le H(w)$$

$$l_{sh} = l_w + qL$$

$$H(w) \le l_w \le H(w) + 1$$

$$H(s) \le l_{sh} \le H(w) + qL + 1$$

$$qL = \sum_{i \in S_2} p_i L = \sum_{i \in S_2} p_i \log_2 \frac{1}{2^{-L}} \le \sum_{i \in S_2} p_i \log_2 \frac{1}{p_i}$$

$$H(s) \le l_{sh} \le H(w) + \sum_{i \in S_2} p_i \log_2 \frac{1}{p_i} + 1 = \sum_{i \in S_1} p_i \log_2 \frac{1}{p_i} + \sum_{i \in S_2} p_i \log_2 \frac{1}{p_i} + q \log_2 \frac{1}{q} + 1$$

$$= H(s) + q \log_2 \frac{1}{q} + 1$$

$$H(s) \le lave \le H(s) + 1.5$$

The worst-case increase in the average codeword length for the shortened code is $q \log_2 \frac{1}{q}$ bits per symbol (this function attains maximum value of ½).

- Example:

Symbol S_i	p_i	l_i	Codeword
0	0.2820	2	11
1	0.2786	2	10
2	0.1419	3	011
3	0.1389	3	010
4	0.0514	4	0011
5	0.0513	4	0010
6	0.0153	5	00011
7	0.0153	5	00010
8	0.0072	6	000011
9	0.0068	6	000010
10	0.0038	7	0000011
11	0.0032	7	0000010
12	0.0019	7	0000001
13	0.0013	8	00000001
14	0.0007	9	000000001
15	0.0004	9	000000000

$$lave = \sum_{i=0}^{15} l_i p_i = 2.694 \text{ bits}$$

The longest codeword is 9 bits
$2^9 = 512$ - entry table is needed for lookup-table-based decoding

Now suppose only a 128-entry lookup table can be permitted.
$\rightarrow$ 7-bit shortened Huffman code

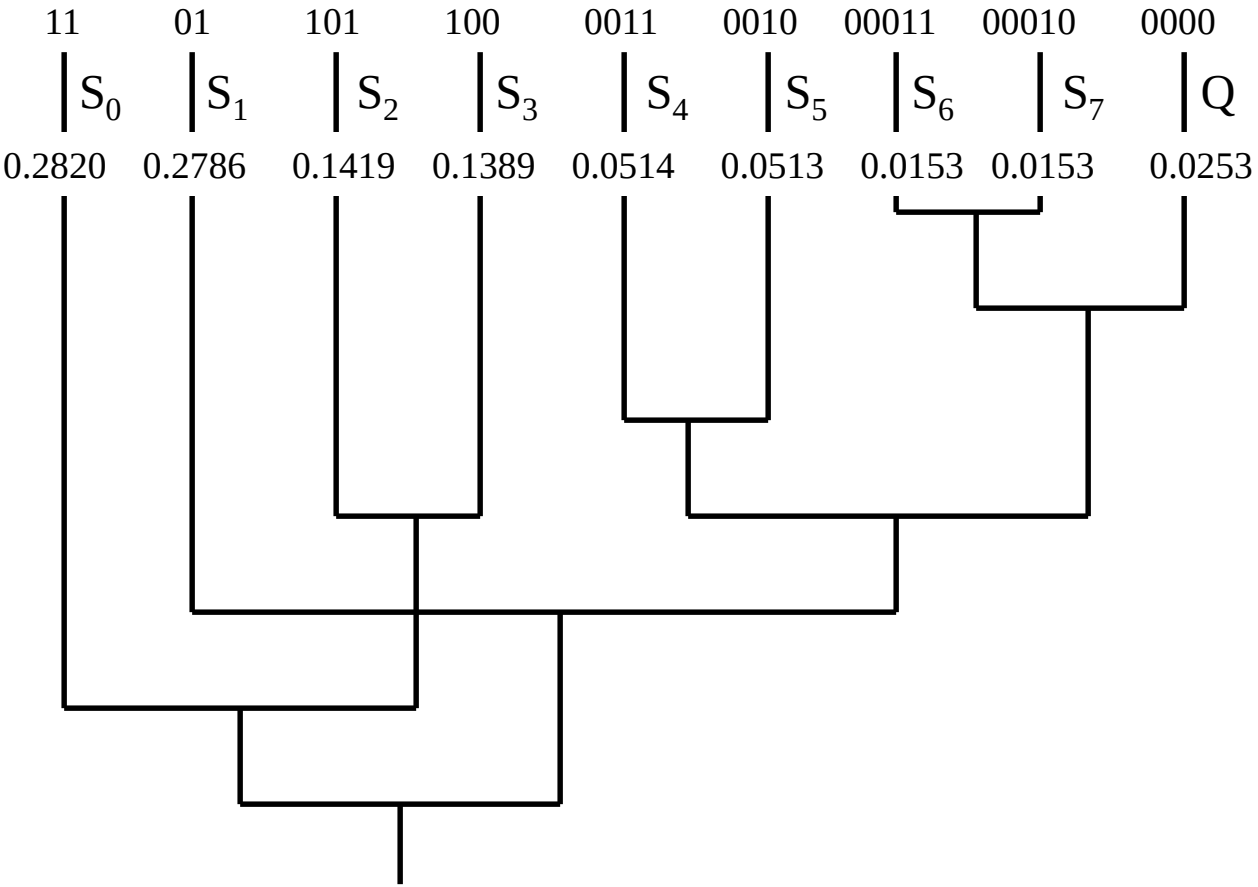
Code construction

1.

$$S_1 = \{s_0, s_1, \cdots s_7\}$$

$$S_2 = \{s_8, s_9, \cdots s_{15}\}, \text{ since } P_i \le \tfrac{1}{128} \quad \text{for i} = 8 \text{ to} 15.$$

$$q = \sum_{i=8}^{15} P_i = 0.0253$$



Symbol i	p_i	l_i	Codeword	Additional
0	0.2820	2	11	
1	0.2786	2	01	
2	0.1419	3	101	
3	0.1389	3	100	
4	0.0514	4	0011	
5	0.0513	4	0010	
6	0.0153	5	00011	
7	0.0153	5	00010	
8	0.0072	11	0000	0001000
9	0.0068	11	0000	0001001
10	0.0038	11	0000	0001010
11	0.0032	11	0000	0001011
12	0.0019	11	0000	0001100
13	0.0013	11	0000	0001101
14	0.0007	11	0000	0001110
15	0.0004	11	0000	0001111

2. For all symbols in S_2 we need a prefix code of 4 bits followed by a 7-bit representation for the specific symbol in S_2

$$l_{sh} = \sum_{i=0}^{7} l_i p_i + \sum_{i=8}^{15} 11 p_i = 2.8057 \quad \text{bits}$$

Decoding:

1. We first construct a lookup table as described above.
2. From the input bit stream, we fetch bits into a buffer until the buffer has 7 bits. We access the lookup table location, using the 7 bits as an address.
   This lookup table location contains (m_k, l_k)
3. The first l_k bits in the buffer are discarded by shifting the buffer contents to the left by l_k bits positions.
   - If $m_k \neq Q$, $m_k \in \{S_0, S_1, \ldots S_7\}$, and thus we have correctly decoded this symbol
   - If $m_k = Q$, additional bits from the input bit stream are needed for decoding. We fetch l_k bits from the bit stream to fill up the buffer. The buffer now contains the binary representation for one of the symbols S_8, $S_9, \ldots S_{15}$, and thus we have correctly decoded a symbol from S_2.
4. Repeat steps 2 and 3 until the complete message has been decoded.

Lookup table size (entries)	Worst case l_{sh}-l_{ave} (bits/symbol)
16	0.4213
32	0.2326
64	0.2326
128	0.1342
256	0.0731
512	0.0338

- The key disadvantage of two-level decoding is that we cannot guarantee a constant symbol rate at the Huffman decoder output.

Constrained-length Huffman codes: prefix-free constant output decoding rate with a table-lookup decoder

- For a maximum codeword length of L bits, we define a threshold $T = 2^{-L}$
- Sort s_i, i = 1, 2, …, N so that $p_k \geq p_{k+1}$
- For each p_i, if $p_i \leq T$, set $p_i = T$
- Design the codebook using the modified p_i values and the unconstrained-length Huffman code table design approach
- Since p_i is restricted to at most 2^{-L}, no codeword length will exceed L bits.
- Codeword length $\propto 1/ p_i$: not guarantee
  This is due to the fact that some of the properties were set to the threshold T and hence the ordering among the properties is obscured.
- Rearranging is done by simply sorting the codeword lengths in ascending order of magnitude and associating this sorted list to the corresponding list of codewords.

Symbol i	p_i	l	Codeword	Reordered	
				l	codeword
0	0.2820	2	11	2	11
1	0.2786	2	01	2	01
2	0.1419	3	101	3	101
3	0.1389	3	100	3	100
4	0.0514	4	0010	4	0010
5	0.0513	4	0001	4	0001
6	0.0153	6	001100	6	001100
7	0.0153	6	001101	6	001101
8	0.0072	7	0011110	6	000010
9	0.0068	7	0011111	6	000011
10	0.0038	7	0011100	6	000000
11	0.0032	7	0011101	6	000001
12	0.0019	6	000010	7	0011110
13	0.0013	6	000011	7	0011111
14	0.0007	6	000000	7	0011100
15	0.0004	6	000001	7	0011101
l_{ave}			2.7308		2.7141

: single-layer decoding

Constrained-length Huffman codes

: The Voorhis method [1974, IEEE Trans. IT]: near optimum codeword length

$$\begin{Bmatrix} S_1, & S_2, & \cdots, & S_N \\ P_1, & P_2, & \cdots, & P_N \end{Bmatrix}, \quad \text{and} \quad P_1 \geq P_2 \geq \cdots \geq P_N$$

- Determine code lengths $l_1, l_2, \ldots, l_N$ that minimize $\sum_i^N l_i p_i$ subject to the constraints $1 \leq l_i \leq L$. For unique decodable codes, we also require that $\sum_j^N 2^{-l_i} \leq 1$. The resulting codeword lengths will be such that $1 \leq l_1 \leq l_2 \leq \ldots \leq l_N \leq L$
- The I-th codeword is the first l_i bits of the fraction computed by $\sum_{k=1}^{i-1} 2^{-l_k}$.

For an N-symbol alphabet, if $L=\log_2 N+d$, then this method has a complexity of $O(dN^2)$.

Symbol S_i	Voorhis code
0	11
1	10
2	011
3	010
4	0011
5	0010
6	00011
7	00010
8	0000111
9	0000110
10	0000101
11	0000100
12	0000011
13	0000010
14	0000001
15	0000000
l_{ave}	2.7045

Optimal Huffman Codes for Arbitrary-Side Growing Trees

- Memory Efficient Hierarchical Lookup Tables for Massive Arbitrary-Side Growing Huffman Tree Decoding, IEEE Trans. On Circuits and Systems for Video Technology, Vol. 18, pp. 1335-1346, Oct. 2008.

Reversible Variable Length Codes- Huffman code based Approach

- On Constructing the Huffman-code-based Reversible Variable-Length Codes, IEEE T-Comm, Sept. 01.
- Modified Symmetrical Reversible Variable-Length Code and Its Theoretical Bounds, IEEE T-IT, Sept., 01.
- On Error Detection and Error Synchronization of Reversible Variable-Length Codes, IEEE T-Comm, May 05.
- Two Algorithms for Constructing Efficient Huffman-Code Based Reversible Variable Length Codes, IEEE T-Comm, Jan. 08.

A different view for constructing Huffman Codes

- H.C. Huang, & Ja-Ling Wu, “Windowed Huffman coding algorithm with size adaptation,” IEE Proceeding-I, pp. 109-113, April 1993.

- Home Work:

1. Consider codes that satisfy the suffix condition, which says that no codeword is a suffix of any other codeword. Show that a suffix condition code is uniquely decodable, and show that the minimum average length over all codes satisfying the suffix condition is the same as the average length of the Huffman code for that random variable.

   Suffix code

2. Suppose that X=i with probability P_i, i=1, 2, …m. Let l_i be the number of binary symbols in the codeword associated with X=i, and let c_i denote the cost per letter of the codeword when X=i. Thus the average cost C of the description of X is

$$C = \sum_{i-1}^{m} p_i c_i l_i$$

a) Minimize C over all $l_1, l_2, \ldots, l_m$ such that $\Sigma 2^{-li} \leq 1$. Ignore any implied integer constraints on l_i. Exhibit the minimizing $l_1^*, l_2^*, \ldots, l_m^*$ and the associated minimum value C^*.

b) How would you use the Huffman code procedure to minimize C over all uniquely decodable codes? Let $C_{Huffman}$ denote this minimum. Show that

$$C^* \leq C_{Huffman} \leq C^* + \sum_{i=1}^{m} p_i c_i$$

Huffman codes with costs.

3. A computer generates a number X according to a known prob. mass function p(x), $x \in \{1,2,\ldots,100\}$. The player asks arbitrary Yes-No questions sequentially until X is determined. If he is right (i.e., X is determined), he receives a prize of value v(x).

   a) How should the player proceed to maximize his expected winnings? What is his expected return?

   b) Continuing (a), what if v(x) is fixed, but p(x) can be chosen by the computer (and then announced to the player)? The computer wishes to minimize the player's expected return. What should p(x) be? What is the expected return to the player?

   — The game of Hi-Lo.

4. Although the codeword lengths of an optimal variable length code are complicated functions of the message probabilities $\{p_1,p_2,\ldots,p_m\}$, it can be said that less probable symbols are encoded into longer codewords. Suppose that the message probabilities are given in decreasing order $p_1 \geq p_2 \geq \ldots \geq p_m$.

   a) Prove that for any binary Huffman code, if the most probable message symbol has probability $p_1 > 2/5$, then that symbol must be assigned a codeword of length 1.

   b) Prove that for any binary Huffman code, if the most probable message symbol has probability $p_1 < 1/3$, then that symbol must be assigned a codeword of length ≥ 2.