Lecture 5: Introduction to Entropy Coding

Thinh Nguyen
Oregon State University

Codes

- Definitions:
  - Alphabet: is a collection of symbols.
  - Letters (symbols): is an element of an alphabet.
  - Coding: the assignment of binary sequences to elements of an alphabet.
  - Code: A set of binary sequences.
  - Codewords: Individual members of the set of binary sequences.

Examples of Binary Codes

- English alphabets:
  - 26 uppercase and 26 lowercase letters and punctuation marks.

  - ASCII code for the letter “a” is 1000011
  - ASCII code for the letter “A” is 1000001
  - ASCII code for the letter “,” is 0011010

Note: all the letters (symbols) in this case use the same number of bits (7). These are called fixed length codes.

Examples of Binary Codes

- English alphabets:
  - 26 uppercase and 26 lowercase letters and punctuation marks.
  - ASCII code for the letter "a" is 1000011
  - ASCII code for the letter "A" is 1000001
  - ASCII code for the letter "," is 0011010

Note: all the letters (symbols) in this case use the same number of bits (7). These are called fixed length codes.

The average number of bits per symbol (letter) is called the rate of the code.

Code Rate

- Average length of the code is important in compression.

- Suppose our source alphabet consists of four letters a_1, a_2, a_3, and a_4 with probabilities $P(a_1) = 0.5$ $P(a_2) = 0.25$, and $P(a_3) = P(a_4) = 0.125$.

- The average length of the code is given by

$$l = \sum_{i=1}^{4} P(a_i) n(a_i)$$

- $n(a_i)$ is the number of bits in the codeword for letter a_i

Uniquely Decodable Codes

Letters	Probabilitity	Code 1	Code 2	Code 3	Code 4
a_1	0.5	0	0	0	0
a_2	0.25	0	1	10	01
a_3	0.125	1	00	110	011
a_4	0.125	10	11	111	0111
Average Length		1.125	1.25	1.75	1.875

Code 1: not unique a_1 and a_2 have the same codeword

Code 2: not uniquely decodable: 100 could mean a_2a_3 or $a_2a_1a_1$

Codes 3 and 4: uniquely decodable: What are the rules?

Code 3 is called instantaneous code since the decoder knows the codeword the moment a code is complete.

How do we know a uniquely decodable code?

- Consider two codewords: 011 and 011101
  - Prefix: 011
  - Dangling suffix: 101
- Algorithm:
  1. Construct a list of all the codewords.
  2. Examine all pairs of codewords to see if any codeword is a prefix of another codeword. If there exists such a pair, add the dangling suffice to the list unless there is one already.
  3. Continue this procedure using the larger list until:
     1. Either a dangling suffix is a codeword -> not uniquely decodable.
     2. There are no more unique dangling suffixes -> uniquely decodable.

Examples of Unique Decodability

- Consider {0,01,11}
  - Dangling suffix is 1 from 0 and 01
  - New list: {0,01,11,1}
  - Dangling suffix is 1 (from 0 and 01, and also 1 and 11), and is already included in previous iteration.
  - Since the dangling suffix is not a codeword, {0,01, 11} is uniquely decodable.

Examples of Unique Decodability

- Consider {0,01,10}
  - Dangling suffix is 1 from 0 and 01
  - New list: {0,01,10,1}
  - The new dangling suffix is 0 (from 10 and 1).
  - Since the dangling suffix 0 is a codeword, {0,01, 10} is not uniquely decodable.

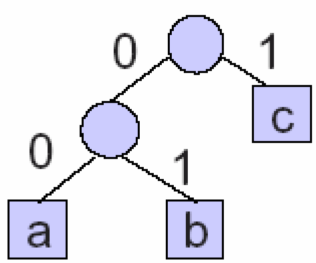
Prefix Codes

- Prefix codes: A code in which no codeword is a prefix to another codeword.

- A prefix code can be defined by a binary tree

  Example:

tree



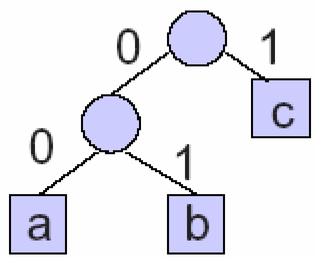
input	output
a	00
b	01
c	1

code

c c a b c c b c c c

1 1 00 01 1 1 01 1 1 1

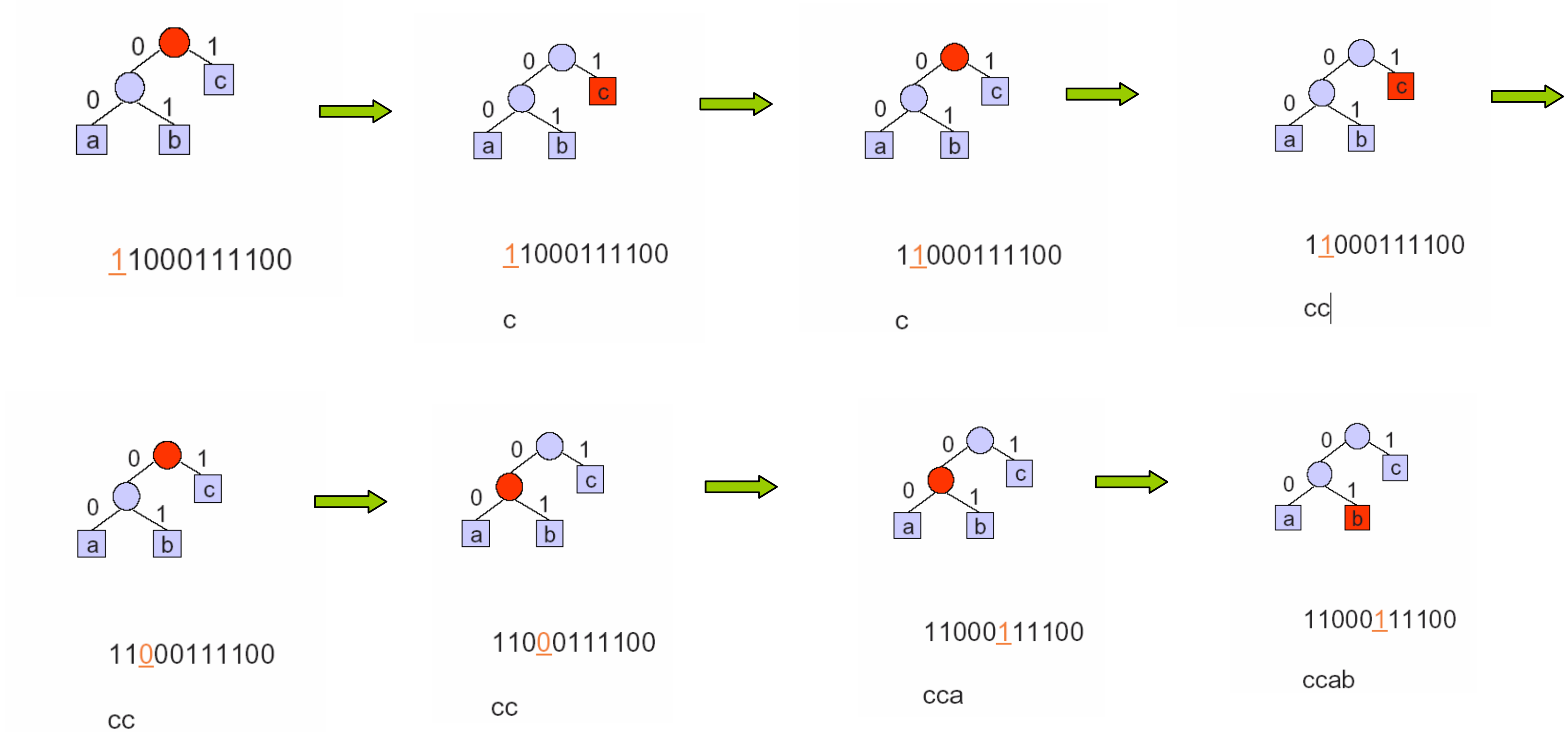
Decoding a Prefix Codeword




```
repeat
start at root of tree
   repeat
      if read bit = 1 then go right
      else go left
   until node is a leaf
   report leaf
until end of the code
```

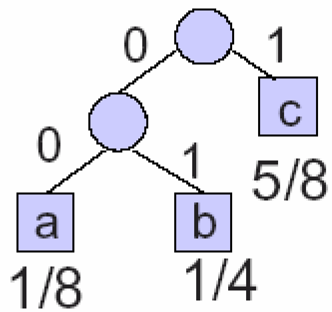
11000111100

Decoding a Prefix Codeword



How good is the code?

Suppose a, b, and c occur with probabilities 1/8, 1/4, and 5/8, respectively.



bit rate = (1/8)2 + (1/4)2 + (5/8)1 = 11/8 = 1.375 bps
Entropy = 1.3 bps
Standard code = 2 bps

(bps = bits per symbol)

Are we losing any efficiency by using prefix code?

- The answer is NO!

- Theorem 1: Let C be a code with N code words with lengths l_1, l_2, ... l_N . If C is uniquely decodable, then

$$K(C) = \sum_{i=1}^{N} 2^{-l_i} \leq 1$$

- Theorem 2: Given a set of integers l_1, l_2, ... l_N that satisfy the inequality

$$\sum_{i=1}^{N} 2^{-l_i} \leq 1$$

we can always find a prefix code with codeword lengths l_1, l_2, ... l_N .

Proof of Theorem 1

$$K(C)=\sum_{i=1}^{N}2^{-l_i}\le 1$$

$$\left[\sum_{i=1}^{N}2^{-l_i}\right]^n=\left(\sum_{i=1}^{N}2^{-l_{i1}}\right)\left(\sum_{i=1}^{N}2^{-l_{i2}}\right)\cdots\left(\sum_{i=1}^{N}2^{-l_{i3}}\right)=\sum_{i1=1}^{N}\sum_{i2=1}^{N}\cdots\sum_{in=1}^{N}2^{-(l_{i1}+l_{i2}+\ldots+l_{in})}$$

The exponent $k=(l_{i1}+l_{i2}+\ldots+l_{in})$ is simply the length of n codewords

Smallest value of k is n and largest value is

So,

$$[K(C)]^n=\sum_{k=n}^{nl}A_k 2^{-k}$$

A_k is the number of combinations of n codewords that have a combined length of k

$A_k\le 2^k$ Since for a uniquely decodable code, each sequence can represent one and only one sequence of codewords. This implies

Growth linearly!!!!

$$[K(C)]^n=\sum_{k=n}^{nl}A_k 2^{-k}\le\sum_{k=n}^{nl}2^k 2^{-k}=nl-n+1$$

Thus, $K(C)\le 1$

Proof of Theorem 2: If $\sum_{i=1}^{N} 2^{-l_i} \le 1$ we can always find a prefix codes with the length $l_1, l_2 \ldots l_N$

Assume: $l_1 \le l_2 \le \ldots \le l_N$

Define: $w_1 = 0, w_j = \sum_{i=1}^{j-1} 2^{l_j - l_i} \qquad j > 1$

Fact 1: binary representation of w_j would take up $ceil[\log_2(w_j + 1)]$

Fact 2: The number of bits in the binary representation of w_j is less than l_j

$$\log_2(w_j + 1) = \log_2\left(\sum_{i=1}^{j-1} 2^{l_j - l_i} + 1\right) = \log_2\left(2^{l_j}\left[\sum_{i=1}^{j-1} 2^{-l_i} + 2^{-l_j}\right]\right)$$

$$= l_j + \log_2\left(\sum_{i=1}^{j} 2^{-l_i}\right) \le l_j$$

Proof of Theorem 2: If $\sum_{i=1}^{N} 2^{-l_i} \le 1$ we can always find a prefix codes with the length $l_1, l_2 \ldots l_N$

Now using the binary representation of w_j , we define the codeword as:

If $ceil(\log_2(w_j+1)) = l_j$, then the jth codeword c_j is the binary representation of w_j

If $ceil(\log_2(w_j+1)) \le l_j$, then the jth codeword c_j is the binary representation of w_j *with* $l_j - ceil(\log_2(w_j+1))$ zeros

This is clearly a decodable code (w_j are all different since $\sum_{i=1}^{j-1} 2^{l_j - l_i}$ is an increased function, each w_j also has length l_j)

Proof of Theorem 2: If $\sum_{i=1}^{N} 2^{-l_i} \le 1$ we can always find a prefix codes with the length $l_1, l_2 \ldots l_N$

Suppose the claim is not true, then for some $j < k$, c_j is the prefix of c_k

This means l_j most significant bits fo w_k form the binary represention of w_j

$$w_j = \left\lfloor \frac{w_k}{2^{l_k - l_j}} \right\rfloor$$, However $$w_k = \sum_{i=1}^{k-1} 2^{l_k - l_j}$$

Therefore,

$$\frac{w_k}{2^{l_k - l_j}} = \sum_{i=1}^{k-1} 2^{l_j - l_i} = w_j + \sum_{i=j}^{k-1} 2^{l_j - l_i} = w_j + 1 + \sum_{i=j+1}^{k-1} 2^{l_j - l_i} \ge w_j + 1$$

That is the smallest value for $\frac{w_k}{2^{l_k - l_j}}$ is $w_j + 1$

Hence, contradicts!