Lecture 19

Golomb Coding Procedure (cont'd)

What if m is not a power of 2?

Code the non-negative integers n = mq + r, where m is a coding parameter (positive integer).

Split the integer n into two parts:
(1) Code q with unary code. Here q is the quotient of (n/m).
    Unary code: q 1's, followed by one '0'. Codeword length of this unary code: (q + 1) bits
(2) Code r using **truncated** binary code. Here r is the remainder of (n/m).

Let $2^B < m < 2^A$, where $A = B+1$

For example, if m = 3, $2^1 < m = 3 < 2^2$, then $B = 1$, and $A = 2$.

if m = 14, $2^3 < m = 14 < 2^4$, then $B = 3$, and $A = 4$

- **Truncated** binary code to represent r:
  (1) Use B-bit binary representation for r values in [0, 1, ..., $2^A - m - 1$].

    For example, if m = 3, use 1-bit binary code for r values in [0], as $(0)_2$.

    where $2^A - m - 1 = 2^2 - 3 - 1 = 0$

    if m = 14, use 3-bit binary code for r values in [0, 1]

    where $2^4 - m - 1 = 16 - 14 - 1 = 1$

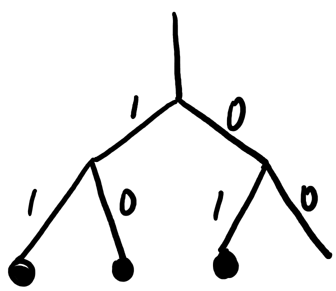
  (2) Use A-bit binary representation for coding r values in $[2^A - m, 2^A - m + 1, \ldots, m-1]$,

    as binary code for $(r + \underbrace{2^A - m}_{\text{offset}})$.

    For example, if m = 3, use 2-bit binary code for r values in [1, 2], where $2^A - m = 2^2 - 3 = 1$
    '1' coded as $(10)_2$, and '2' coded as $(11)_2$.

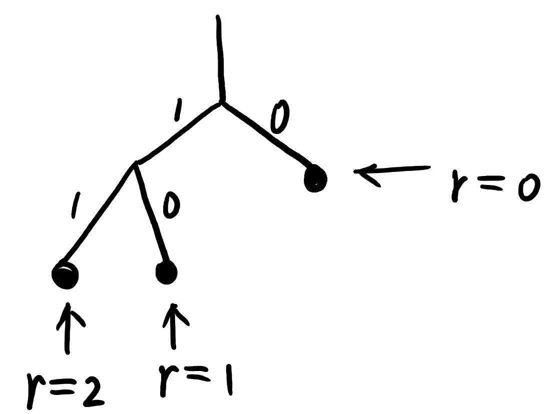
	$m = 3$	
n	$G(n)$	Codeword
0	0.206	00
1	0.164	010
2	0.130	011
3	0.103	100
4	0.081	1010
5	0.064	1011
6	0.051	1100
→ 7	0.041	11010
8	0.032	11011
9	0.026	11100
10	0.021	111010

$$\rightarrow \frac{n}{m} = \frac{7}{3} \Rightarrow \begin{cases} q = 2 \Rightarrow \text{'110'} \\ r = 1 \Rightarrow \text{'10'} \end{cases}$$



Incomplete Binary Code

$\Rightarrow$



Truncated Binary Code

Another example, m = 5, $2^2 < m = 5 < 2^3$

$$\underset{B=2}{\downarrow} \qquad \underset{A=3}{\downarrow}$$

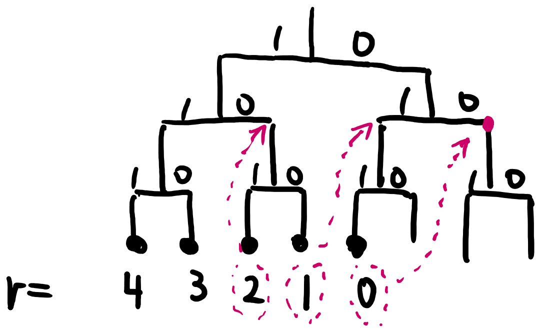
- **Truncated** binary code to represent r:
  (1) Use B-bit binary representation for r values in [0, 1, …, $2^A - m - 1$].
  ↓
  2-bit

$r \in [0, 1, \dots, 2^3 - 5 - 1 = 2]$

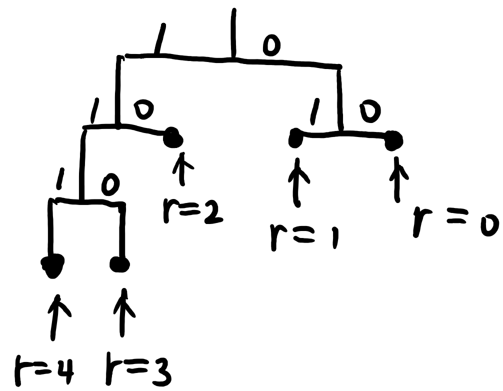
(2) Use A-bit binary representation for coding r values in $[3, 4]$

$A = 3$

$$\text{offset} = 2^A - m = 2^3 - 5 = 3$$



Incomplete Binary Code



$$(110)_2 = (6)_{10} = 3 + \underbrace{\text{offset}}_{3}$$

Truncated Binary Code

$m = 14$			
n	Codeword	n	Codeword
0	0000	24	101100
1	0001	25	101101
2	00100	26	101110
3	00101	27	101111
4	00110	28	110000
5	00111	29	110001
6	01000	30	1100100
7	01001	31	1100101
8	01010	32	1100110
9	01011	33	1100111
10	01100	34	1101000
11	01101	35	1101001
12	01110	36	1101010
13	01111	37	1101011
14	10000	38	1101100
15	10001	39	1101101
16	100100	40	1101110
17	100101	41	1101111
18	100110	42	1110000
19	100111	43	1110001
20	101000	44	11100100
21	101001	45	11100101
22	101010	46	11100110
23	101011	47	11100111

$2^3 < m = 14 < 2^4$

$\downarrow \quad B = 3 \qquad \downarrow \quad A = 4$

(1) Use B-bit binary representation for r values in $[0, 1, \ldots, \underbrace{2^A - m - 1}]$

$16 - 14 - 1 = 1$

(2) Use $\overset{\nearrow 4}{A}$-bit to represent $r = 2$, with offset $2^A - m = 2^4 - 14 = 2$

$r + \text{offset} = 2 + 2 = 4$.

$\frac{n}{m} = \frac{44}{14} \Rightarrow \begin{cases} q = 3 \Rightarrow '1110' \\ r = 2 \Rightarrow '0100' \end{cases}$

- Golomb Decoding

Case 1: m is a power of two

Code the non-negative integers n = mq + r, where m is a coding parameter (positive integer).

Split the integer n into two parts:

(1) Code q with unary code. Here q is the quotient of (n/m).

Unary code: q 1's, followed by one '0'. Codeword length of this unary code: (q + 1) bits

(2) Code r using binary code. Here r is the remainder of (n/m). Binary code has $\underbrace{(\log_2 m)}_{s = \log_2 m}$ bits

$(\log_2 m)$ bits for binary representation of r.

Decoding

- Count from the left the number (q) of 1's preceding the first 0;
- Scan the next (s + 1) bits to reconstruct the r value.
- The codeword can be decoded as n = mq + r.

$(0110)_2 = 6 = r$

For example, for m = 16, decode codeword (100110) Thus $n = mq + r = 16 \times 1 + 6 =$

- The codeword can be decoded as n = mq + r.

$(0110)_2 = 6 = r$

For example, for m = 16, decode codeword (100110)

$s = \log_2 m = 4$

$q = 1$

Thus $n = mq + r = 16 \times 1 + 6 = 22.$

n

22 | 101010

Case 2: m is NOT a power of two

Review the encoding procedure:
Split the integer n into two parts:
(1) Code q with unary code. Here q is the quotient of (n/m).
Unary code: q 1's, followed by one '0'. Codeword length of this unary code: (q + 1) bits
(2) Code r using **truncated** binary code. Here r is the remainder of (n/m).

- **Truncated** binary code to represent r:
  (1) Use B-bit binary representation for r values in [0, 1, ..., $2^A - m - 1$].
  (2) Use A-bit binary representation for coding r values in $[2^A - m, 2^A - m + 1, \ldots, m-1]$,
  as binary code for $(r + \underbrace{2^A - m}_{\text{offset}})$.

Decoding Procedure:

- Count from the left the number (q) of 1's preceding the first 0;
- Skip the next bit ('0').
- First assume that r is represented by the next B-bit binary code, then check if $0 \leq r \leq 2^A - m - 1$.

  If "Yes", then the assumption is valid;
  If "No", decode the next A-bit binary representation:

$$r = (\quad) - \underbrace{\text{offset}}_{2^A - m}$$

- The codeword can be decoded as n = mq + r.

http://www.ece.uah.edu/~dwpan/course/ee614/code/golomb_deco.m

```
function n_rec = golomb_deco (codeword, m)
%   Function n_rec = golomb_deco (codeword, m)
%   golomb_deco decodes the input golomb codeword using parameter m;
```

```
%   Input: The codeword (1-D vector) for a non-negative integer n;
%          m is the coding parameter.
%   Output: n_rec is the reconstructed symbol.
%   Example: n_rec = golomb_deco([0 1 0], 5)
```

```
len = length(codeword);
% Count the number of 1's followed by the first 0
q = 0;
for i = 1: len
    if codeword(i) == 1
        q = q + 1;
    else
        ptr = i;   % first 0
        break;
    end
end
if (m == 1)
    n_rec = q;   % special case for m = 1
else
    A = ceil(log2(m));
    B = floor(log2(m));
    bcode = codeword((ptr+1): (ptr + B));
    r = bi2de(bcode,'left-msb');
    if r < (2^A - m)
        ptr = ptr + B;
    else
        % r is A-bit represtation of (r + (2^A - m))
        bcode = codeword((ptr+1): (ptr + A));
        r = bi2de(bcode,'left-msb') - (2^A - m);
        ptr = ptr + A;
    end
    n_rec = q * m + r;
end
if ~isequal(ptr, len)
    error('Error: More than one codeword detected!');
end
```