ECE 231A Shengze Ye HW3

1. code Analysis

a	0
b	01
c	001
d	0011001
e	00100100

(a) Is this code instantaneous? Does an instantaneous binary source code exist with these codeword length?

No, it is not instantaneous

Since some of the codes are prefix of other codes

"001" is prefix of "0011001"

"001" is also prefix of "001001001"

The code lengths are 1, 2, 3, 6, 9, if it satisfies the kraft inequality, we can find an instantaneous binary code

$$\sum_i D^{-L_i} = 2^{-1} + 2^{-2} + 2^{-3} + 2^{-6} + 2^{-9}$$

$$= \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{64} + \frac{1}{512} = \frac{457}{512} \approx 0.89 < 1$$

So, according to the kraft inequality, there exists an instantaneous code.

For example

a	0
b	10
c	110
d	111110
e	11111110

(b) Is this code uniquely decodeable? Does a uniquely decodeable binary source code exist with these codeword lengths?

No, it is not uniquely decodeable

since "001001" can be decoded as "d" or "ee"

"00100100" can be decoded as

"e". or "dc" or "cd" or "ccc"

The uniquely decodeable source code exist with these code length.

According to the McMillan Inequality, all uniquely decodeable codes satisfy the kraft inequality. and the codeword lengths satisfy kraft inequality, we can find a uniquely decodable code

2. Reversal

a	0
b	01
c	011
d	0111
e	01111
f	011111
g	111111

1a) Is this code instantaneous? Does an instantaneous code exist with these code lengths?

No, this code is not instantaneous. because some of the codes are prefix of other codes.

Such as "0", "01"

"011", "0111"

...

The lengths of the codes are: 1, 2, 3, 4, 5, 6, 6

$$\sum_i D^{-l_i} = 2^{-1} + 2^{-2} + 2^{-3} + 2^{-4} + 2^{-5} + 2^{-6} + 2^{-6}$$

$$= 1$$

The lengths satisfy the kraft inequality, so we could find an instantaneous source code

For example

a	0
b	10
c	110
d	1110
e	11110
f	111110
g	111111

(b) Is this code uniquely decodable? Does a uniquely decodable source code exists with these codeword length?

Yes, it is uniquely decodable. Except for last word, each word can be recognized by the number of "1"s after one "0".
And if a word starts with "1", we know it is "g" (the last symbol)

An uniquely decodable source code exists since the lengths satisfy the kraft inequality, according to the McMillan Inequality, a uniquely decodeable source code exists

3. Slackness in the Kraft Inequality

$$\sum_{i=1}^{m} D^{-l_i} < 1$$

The code alphabet is $D = \{0, 1, 2, \cdots, D-1\}$, show that there exist sequences of code symbols in D^* which cannot be decoded into sequences of codewords

let $l_{max} = \max\{l_1, l_2, \cdots l_m\}$, and there are

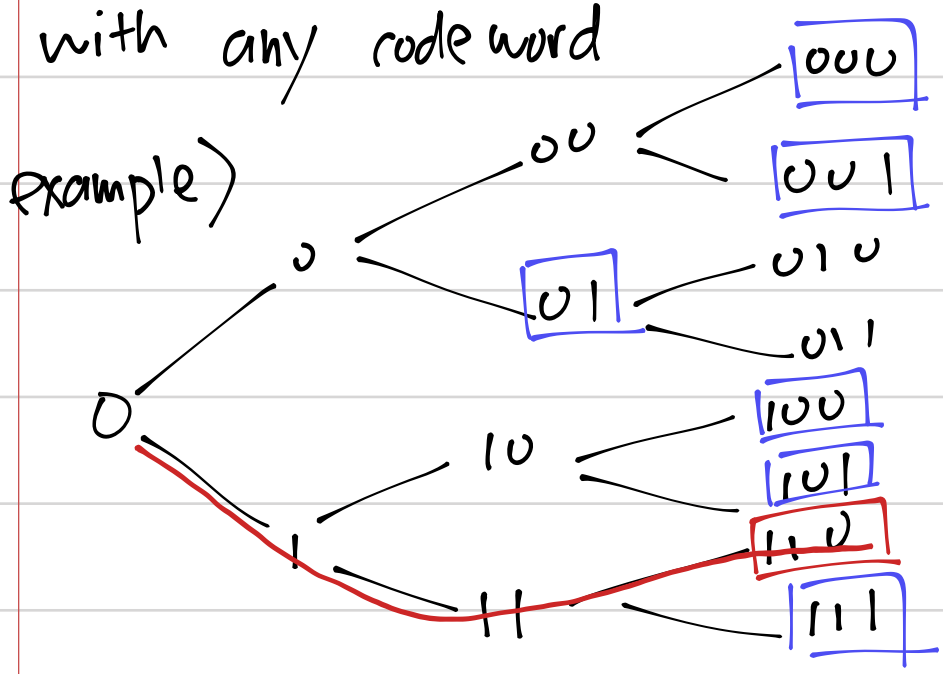
$D^{l_{max}}$ sequences of length l_{max}

$D^{l_{max} - l_i}$ sequences start with i-th codeword

Because it is an instantaneous code, no two sequences can start with the same codeword

$$\sum_{i=1}^{m} D^{l_{max} - l_i} = D^{l_{max}} \underbrace{\sum_{i=1}^{m} D^{-l_i}}_{(<1)} < D^{l_{max}}$$

, since the strict inequality, there are sequences do not start with any codeword



example)

The codeword in blue box is the codeword we choose

and $\sum_{i=1}^{m} D^{-l_i} < 1$ (strict inequality)

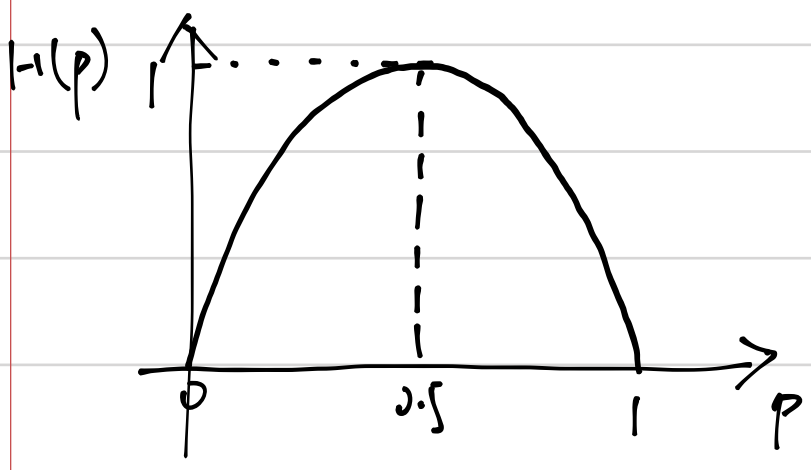
So. there are sequences that do not start with any codeword. Like "110" is this example, and it cannot be decoded into sequences of codewords

4. Optimal code lengths that require one bit above entropy

let X be a binary random variable

$p(X=1) = p \quad p(X=0) = 1-p$

So, the entropy for X is $H(X) = H(p)$



H(p) is a concave function and we can choose p close to 0 to make H(p) as small as possible

For any $\varepsilon > 0$, we can choose a small p, such that $H(p) < \varepsilon$

The length of optimal code is $L = 1$, since we need at least one bit to code it

$1 + H(X) - \varepsilon < 1 = L$

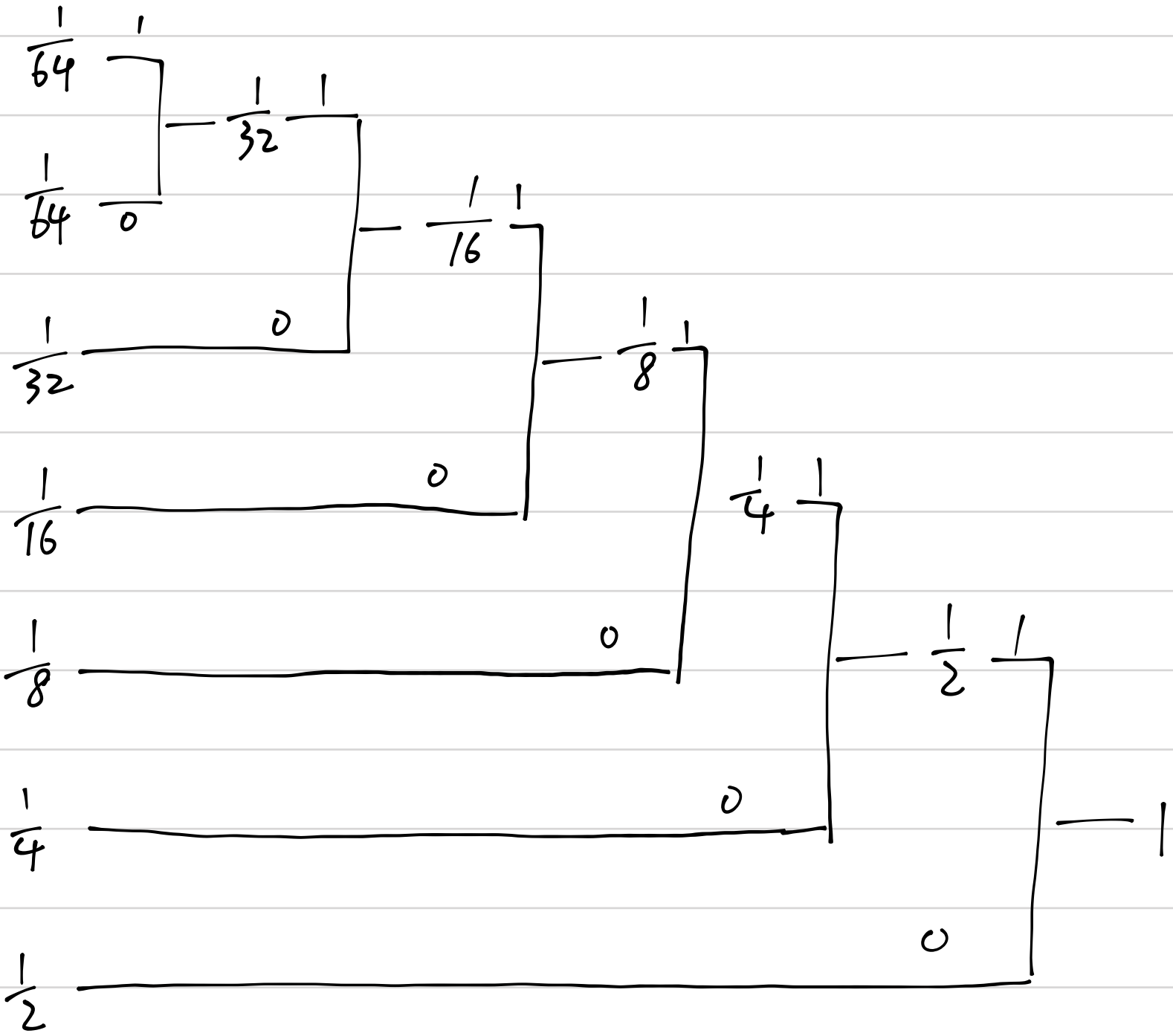
$\therefore L > H(X) + 1 - \varepsilon$ for any $\varepsilon > 0$

5. Huffman code

X is distributed according to the following probability mass function:

$\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \frac{1}{64}, \frac{1}{64}$

Find a Huffman code



prob	codeword	length
$\frac{1}{64}$	111111	6
$\frac{1}{64}$	111110	6
$\frac{1}{32}$	11110	5
$\frac{1}{16}$	1110	4
$\frac{1}{8}$	110	3
$\frac{1}{4}$	10	2
$\frac{1}{2}$	0	1

$$L[c] = \sum_x p(x) l(x)$$

$$= \frac{1}{64}\times 6 + \frac{1}{64}\times 6 + \frac{1}{32}\times 5 + \frac{1}{16}\times 4 + \frac{1}{8}\times 3 + \frac{1}{4}\times 2 + \frac{1}{2}\times 1$$

$$= \frac{126}{64} = 1.96875 \text{ bit}$$

$$H(x) = -\sum p(x) \log p(x)$$

$$= -\frac{1}{64}\log\frac{1}{64} - \frac{1}{64}\log\frac{1}{64} - \frac{1}{32}\log\frac{1}{32} - \frac{1}{16}\log\frac{1}{16} - \frac{1}{8}\log\frac{1}{8} - \frac{1}{4}\log\frac{1}{4} - \frac{1}{2}\log\frac{1}{2}$$

$$= \frac{6}{64} + \frac{6}{64} + \frac{5}{32} + \frac{4}{16} + \frac{3}{8} + \frac{2}{4} + \frac{1}{2}$$

$$= \frac{126}{64} = 1.96875 \text{ bit}$$

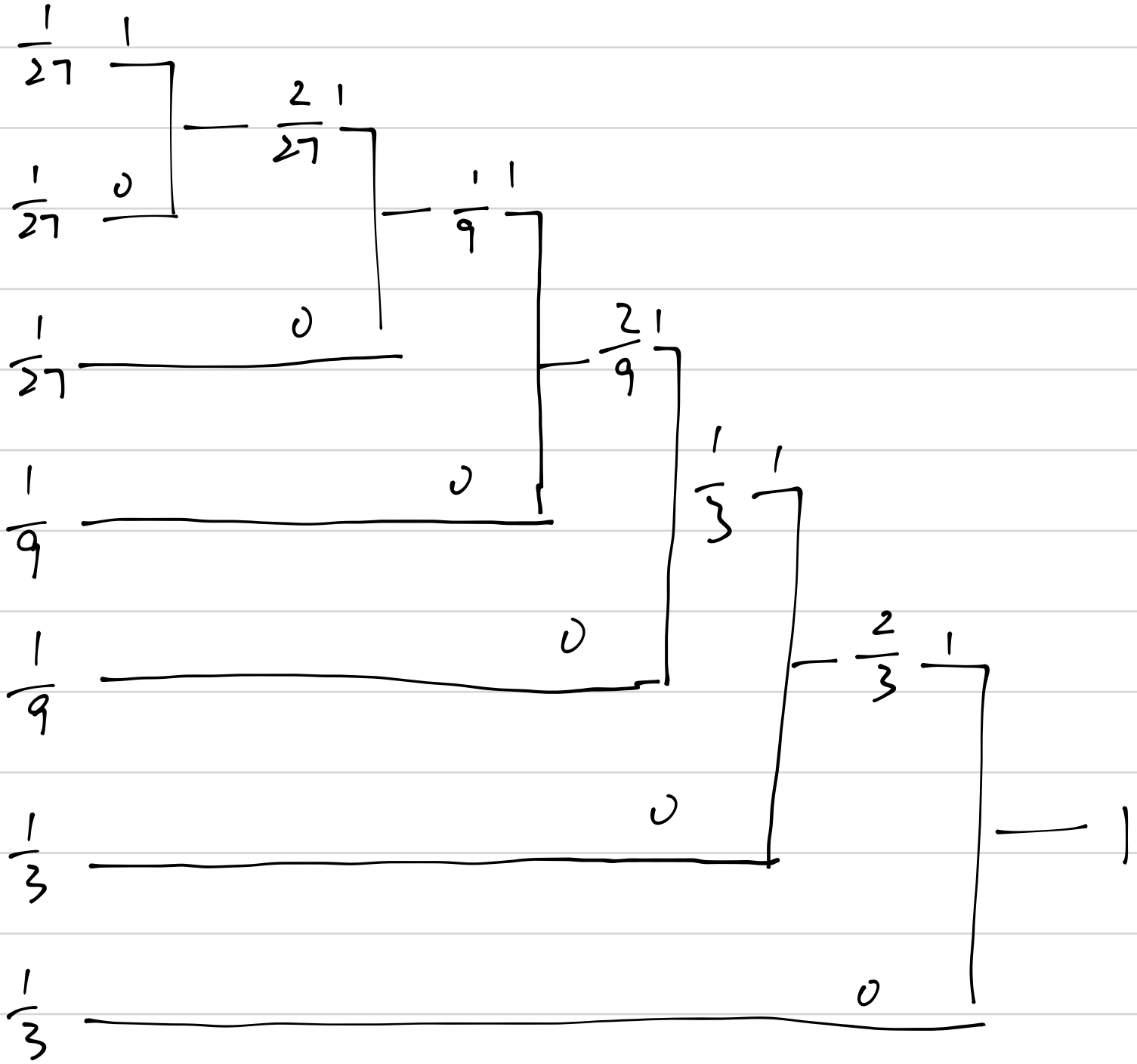
$H(x) = L(c)$

$\therefore$ we have already achieve the corresponding entropy limit on compression

6. Non-binary Huffman code ?

X is distributed acrording to the following probability mass function:

$\frac{1}{3}$ $\frac{1}{3}$ $\frac{1}{9}$ $\frac{1}{9}$ $\frac{1}{27}$ $\frac{1}{27}$ $\frac{1}{27}$



prob	codeword	length
$\frac{1}{27}$	111111	6
$\frac{1}{27}$	111110	6
$\frac{1}{27}$	11110	5
$\frac{1}{9}$	1110	4
$\frac{1}{9}$	110	3
$\frac{1}{3}$	10	2
$\frac{1}{3}$	0	1

$$L(C) = \sum_{i} p(x)\, l(x)$$

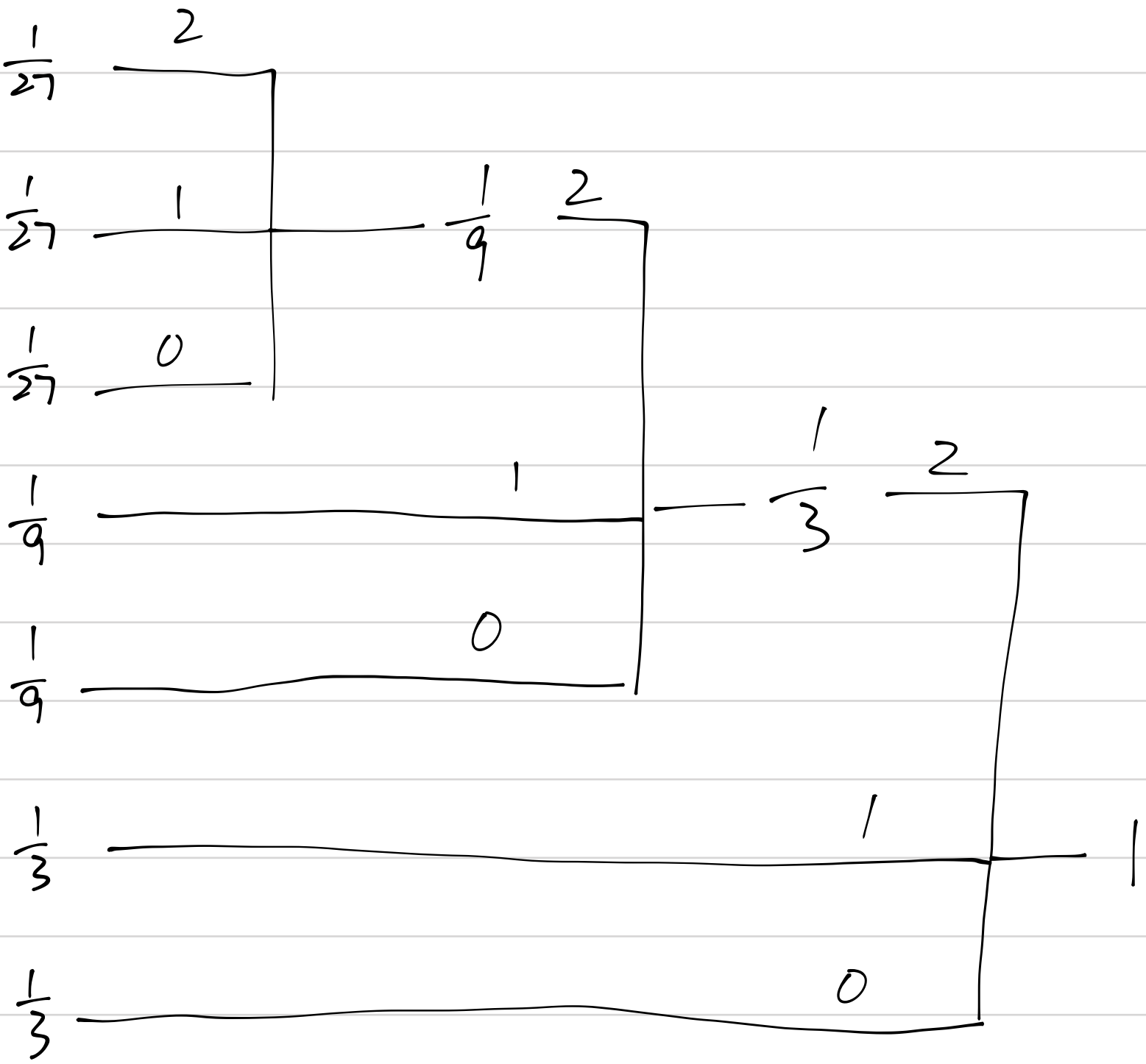
$$= \frac{1}{27}\times 6 + \frac{1}{27}\times 6 + \frac{1}{27}\times 5 + \frac{1}{9}\times 4 + \frac{1}{9}\times 3 + \frac{1}{3}\times 2 + \frac{1}{3}$$

$$= \frac{17}{27} + \frac{7}{9} + 1$$

$$= \frac{65}{27} = 2.4074 \text{ bits}$$

The average length in bits is 2.4074 bits

(b) Find a ternary Huffman code (for which there are three symbols instead of two, three branches in every Huffman step instead of two). Compute it's average length.



prob	codeword	length
$\frac{1}{27}$	222	3
$\frac{1}{27}$	221	3
$\frac{1}{27}$	220	3
$\frac{1}{9}$	21	2
$\frac{1}{9}$	20	2
$\frac{1}{3}$	1	1
$\frac{1}{3}$	0	1

$$L(C) = \sum_x p(x)\, l(x)$$

$$= \frac{1}{27} \times 3 + \frac{1}{27} \times 3 + \frac{1}{27} \times 3 + \frac{1}{9} \times 2 + \frac{1}{9} \times 2 + \frac{1}{3} + \frac{1}{3}$$

$$= \frac{13}{9} = 1.444 \text{ trits}$$

The average length is 1.444 trits

(c) $H_3(x) = -\sum_x p(x) \log_3 p(x)$

$$= -\frac{1}{27}\log_3\frac{1}{27} - \frac{1}{27}\log_3\frac{1}{27} - \frac{1}{27}\log_3\frac{1}{27} - \frac{1}{9}\log_3\frac{1}{9} - \frac{1}{9}\log_3\frac{1}{9} - \frac{1}{3}\log_3\frac{1}{3} - \frac{1}{3}\log_3\frac{1}{3}$$

$$= \frac{1}{9} + \frac{1}{9} + \frac{1}{9} + \frac{2}{9} + \frac{2}{9} + \frac{1}{3} + \frac{1}{3}$$

$$= \frac{13}{9} = 1.444 \text{ trits}$$

$\therefore H_3(x) = L(C)$

The ternary Huffman code achieves the corresponding ternary entropy limit on compression.

To make the comparison which code is more efficient we need to normalize them. 1 trits $= \log_2 3$ bits

1.444 trits $\times \log_2 3 = 2.2887$ bits < 2.4074 bits

From this point of view, I think ternery code is more efficient.

However, in practice, we normally use binary code since our computer is binary system, it is more convenient.

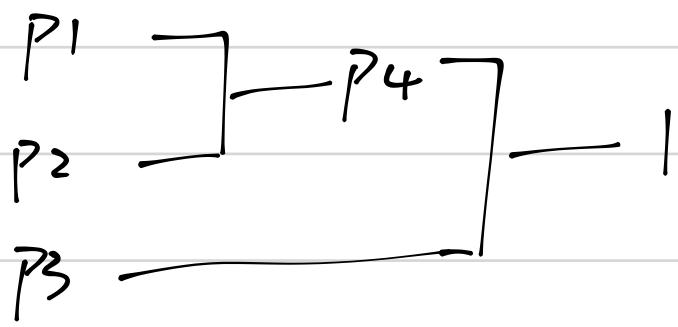
7. A sufficient set of Huffman code

(a) How many distinct binary Huffman codes does it take to handle all possible PMF's that have exactly three distinct outcomes with non-zero probability?

one.

We can assume the distribution is p_1, p_2, p_3.

For Huffman coding, we always start with two outcomes with smallest probability. without losing generology we can assume that $p_1 \le p_2 \le p_3$



we first merge p_1 and p_2 and then merge the two left

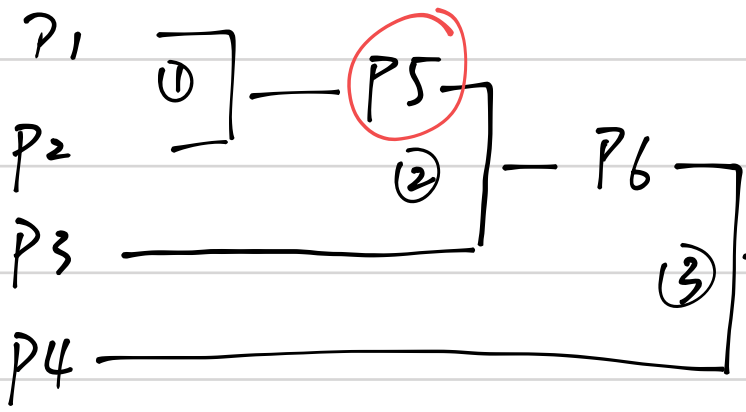
So we only have one tree structure

The smallest number of Huffman code is 1.

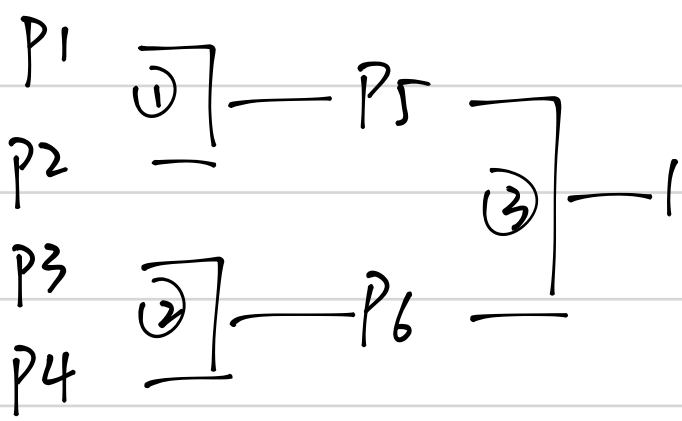
(b) How many distinct binary Huffman codes does it take to handle all possible PMF's that have exactly four distinct outcomes with non-zero prob?
two

Same with the previous problem, we could assume the distribution is p_1, p_2, p_3, p_4, we can also assume that $p_1 \le p_2 \le p_3 \le p_4$

And there are two kind of tree structures



At first, as the p_1, p_2 are the smallest prob, we should merge them, if P_5 is not the largest prob among P_3, P_4, P_5, P_5 should merge with P_3 or P_4, this is first case.



other wise, if P_5 is the largest prob among P_3, P_4, P_5, we then should merge P_3 and P_4, and this corresponds to the second tree.

So, we need 2 distinct Huffman codes to handle

8. codeword length

Consider a loseless compression code that uses five binary codewords with lengths $\{1, 2, 3, 4, 5\}$

(a) Prove that a uniquely decodeable code exists with these codewords, While you are at it. prove that pre-fix free code exists with these codeword lengths.

Arrording to McMillan Inequality, and Kraft inequality

$$\sum_i D^{-l_i} = 2^{-1} + 2^{-2} + 2^{-3} + 2^{-4} + 2^{-5} = \frac{31}{32} < 1$$

Therefore, a uniquely decodeable code exists with these codewords length,

And for any uniquely decodeable code, there is an instantaneous code with the same word lengths. So that pre-fix free code exists

(b) exhibit a prefix free code with these codelngt

(code)	(Length)
0	1
10	2
110	3
1110	4
11110	5

(c) Either give a PMF for which your code is a Huffman code, or explain why your code cannot be a Huffman code for any PMF?

The code cannot be a Huffman code for any PMF.
For any distribution, there is an instantaneous code with minimum length that satisfies the following properties:

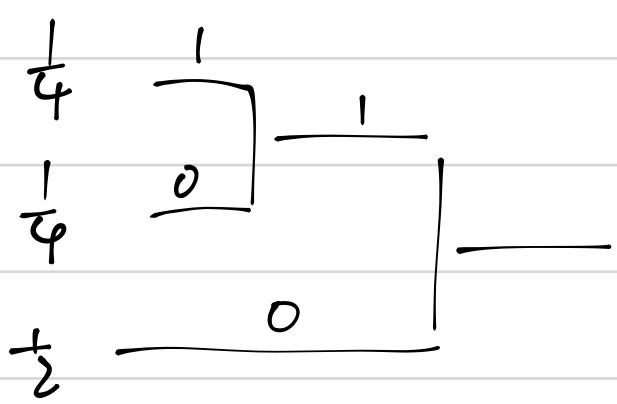
1. $p_j > p_k \Rightarrow l_j \leq l_k$
2. Two longest words have the same length
3. Two of the longest codewords differ only in the last bit

And the Huffman codes are the most efficient codes that obey rule 1, 2 and 3.
But in this case, the two longest codeword do not have the same length (4 and 5)
So it can't be a Huffman code

9. Bad codes.
Which of these codes cannot be Huffman codes for any probability assignment?

(a) $\{0, 10, 11\}$

It could be the Huffman code of $\{\frac{1}{2}, \frac{1}{4}, \frac{1}{4}\}$



$\frac{1}{4} \rightarrow 11$

$\frac{1}{4} \rightarrow 10$

$\frac{1}{2} \rightarrow 0$

(b) $\{00, 01, 10, 110\}$

The code can be shortened to $\{00, 01, 10, 11\}$ and reserve its pre-fix free property, it is not optimal so it cannot be Huffman code

(c) $\{01, 10\}$

The code can be shortened to $\{0, 1\}$ and reserve its pre-fix free property, it is not optimal, So it cannot be Huffman code.

10. Arithmetic coding

The following probability model should be used for both encoding and decoding.

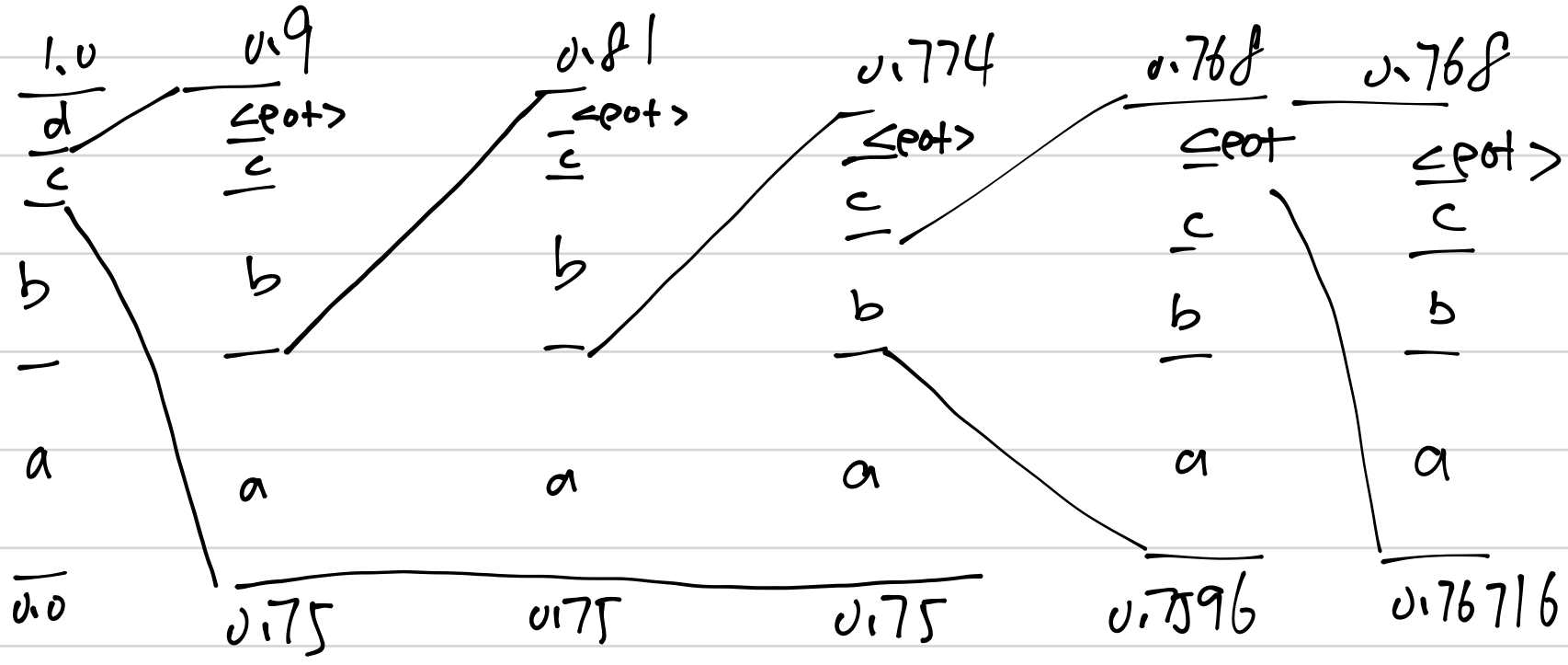
Symbol	Probability	Range
a	0.4	[0.0, 0.4)
b	0.35	[0.4, 0.75)
c	0.15	[0.75, 0.9)
<eot>	0.1	[0.9, 1)

(a) Encode caab <eot> using arithmetic coding.

We can obtain the code using MATLAB

```
l1 = [0.0, 0.4, 0.75, 0.9];
u1 = [0.4, 0.75, 0.9, 1.0];
% a, b, c, <eot> -> 1,2,3,4
source = [3,1,1,2,4]; % caab<eot>
low = 0.0;
high = 1.0;
for i=1:length(source)
    [low,high] = encode_symbol(source(i), low, high, l1, u1);
end
n_low = low;
n_high = high;
code_bits = [];
while true
    [n_low, n_high, bit] = send_bit(n_low, n_high);
    if bit == -1
        break
    end
    code_bits = [code_bits, bit];
end
code_bits = [code_bits, 1]; % send 1 at last to guarantee that <eot> is received
value = binary2real(code_bits);
```

So we use function encode_symbol to find the interval for caab<eot>, then encoded it to binary bit.



The final value is

0.7675781125

The code is 1 1 0 0 0 1 0 0 1 1

send 1 to guarantee <eot> will be received

we assume that all bits to the right of the last transmitted bit are zeros

The code is 1 1 0 0 0 1 0 0 1 1

(b) Decode 0 0 1 1 0 1 0 1 1, Assume that all bits to the right of the last transmitted bit are zeros.

First, we can calculate the value for 0.0 0 1 1 0 1 0 1 1

$\Rightarrow$ value = 0.209960975

```
function [symbol, new_value] = decode_symbol(value, ll, ul)
symbol = -1;
for i=1:length(ul)
    if value < ul(i)
        symbol = i; % find the right interval of symbol
        break;
    end
end
low = ll(symbol); % find the lower bound of that interval
high = ul(symbol); % find the upper bound of that interval
new_value = (value - low)/(high-low);

end
```

This the decode_symbol function, it decodes the symbol one by one

```
decode_bits = [];
code_bits = [0, 0, 1, 1, 0, 1, 0, 1, 1, 1];
value = binary2real(code_bits);
while true
    [symbol, value] = decode_symbol(value, ll, ul);
    if symbol == 4
        decode_bits = [decode_bits, 4];
        break
    end
    decode_bits = [decode_bits, symbol];
end
```

This the code for decoding sequences. we end decoding as soon as we meet <eot>. which represents the end

The decodes symbols are 1, 2, 1, 3, 4
which is abac<eot>
The decoded word is abac<eot>

II. L&W

In this problem, you will encode and decode sequences using the Limpel - Ziv - Welch algorithm L&W discussed in lecture
The following code book should be used for both coding and decoding

Phrase Number	Phrase
1	a
2	b
3	c

(a) Encode cabcbcbcb using LZW

	cabcbcbcb		# phrase	phrase
1.	3	"c"	1	a
2	1	"a"	2	b
3.	2	"b"	3	c
4.	3	"c"	4	ca
5.	6	"bc"	5	ab
6.	8	"bcb"	6	bc
			7	cb
			8	bcb

The code is "3 1 2 3 6 8"

(b) Decode 3, 4, 5, 6, 7, 8, 9, 1

Initial codebook

1	a
2	b
3	c

1.	3	"c"
2.	4	"cc"
3.	5	"ccc"
4.	6	"cccc"
5.	7	"ccccc"
6.	8	"cccccc"
7.	9	"ccccccc"
8	1	"a"

#phrase	phrase
1.	a
2.	b
3.	c
4.	cc
5.	ccc
6.	cccc
7.	ccccc
8	cccccc
9.	ccccccc
10	cccccca

Therefore the code word is

c.cc.ccc.cccc.ccccc.cccccc.ccccccc.a