Solutions to MATH3411 Problems 27-34

27a) $\frac{1}{2^1}+\frac{1}{2^2}+\frac{3}{2^3}=\frac{9}{8}>1$, so there is no such code, by the Kraft-McKillan Theorem.

b) 00, 01, 100, 101, 1100, 1101, 1110 (this last codeword can be shortened to 111).

c) 0, 1, 200, 201, 202, 210, 211, 212, 220

d) $\frac{2}{3^1}+\frac{2}{3^2}+\frac{4}{3^3}=\frac{28}{27}>1$, so there is no such code, by the Kraft-McKillan Theorem.

28a) $K=\frac{2}{r^1}+\frac{3}{r^2}+\frac{2}{r^3}+\frac{1}{r^4}$.
Substituting $r=2,3,\ldots$, we see that if $K\le 1$, then $r>3$.
The Kraft-McKillan Theorem implies that the minimal radix for such a UD-code to exist is $r=4$.

b) $K=\frac{3}{r^2}+\frac{4}{r^4}+\frac{1}{r^5}$.
Substituting $r=2,3,\ldots$, we see that if $K\le 1$, then $r>2$.
The Kraft-McKillan Theorem implies that the minimal radix for such a UD-code to exist is $r=3$.

29. Here, it is a good idea to draw the decision tree arising from the binary Huffman algorithm. However, I am going to be lazy and just write up the steps without drawing anything.

a) The combining phase of the Huffman algorithm (with place-high strategy) goes as follows:

Source symbols	Step 0	Step 1	Step 2
s_1	$p_1=\frac{1}{2}$	$p_{23}=\frac{1}{2}$	$p_{231}=1$
s_2	$p_2=\frac{1}{3}$	$p_1=\frac{1}{2}$	
s_3	$p_3=\frac{1}{6}$		

Going backwards, the splitting phase of the Huffman algorithm gives the codeword symbols as follows:

Source symbols	Step 0	Step 1	Step 2
s_1	p_1 : 1	p_{23} : 0	p_{231} : $\emptyset$
s_2	p_2 : 00	p_1 : 1	
s_3	p_3 : 01		

In other words, s_1, s_2, s_3 are encoded as $1, 00, 01$, respectively.
The expected codeword length is then:

$$L=\frac{1}{2}\times 1+\frac{1}{3}\times 2+\frac{1}{6}\times 2=\frac{3}{2}$$

You can reduce these calculations with Knuth's theorem: $L=1+\frac{1}{2}=\frac{3}{2}$.

b) The combining phase of the Huffman algorithm (with place-high strategy) goes as follows:

Source symbols	Step 0	Step 1	Step 2	Step 3	Step 4
s_1	$p_1 = \frac{1}{3}$	$p_1 = \frac{1}{3}$	$p_{453} = \frac{5}{12}$	$p_{12} = \frac{7}{12}$	$p_{12453} = 1$
s_2	$p_2 = \frac{1}{4}$	$p_2 = \frac{1}{4}$	$p_1 = \frac{1}{3}$	$p_{453} = \frac{5}{12}$	
s_3	$p_3 = \frac{1}{5}$	$p_{45} = \frac{13}{60}$	$p_2 = \frac{1}{4}$		
s_4	$p_4 = \frac{1}{6}$	$p_3 = \frac{1}{5}$			
s_5	$p_5 = \frac{1}{20}$				

Going backwards, the splitting phase of the Huffman algorithm gives the codeword symbols as follows:

Source symbols	Step 0	Step 1	Step 2	Step 3	Step 4
s_1	$p_1 : 00$	$p_1 : 00$	$p_{453} : 1$	$p_{12} : 0$	$p_{45312} : \emptyset$
s_2	$p_2 : 01$	$p_2 : 01$	$p_1 : 00$	$p_{453} : 1$	
s_3	$p_3 : 11$	$p_{45} : 10$	$p_2 : 01$		
s_4	$p_4 : 100$	$p_3 : 11$			
s_5	$p_5 : 101$				

In other words, $s_1, \ldots, s_5$ are encoded as $00, 01, 11, 100, 101$, respectively.
The expected codeword length is then:

$$L = \frac{1}{3} \times 2 + \frac{1}{4} \times 2 + \frac{1}{5} \times 2 + \frac{1}{6} \times 3 + \frac{1}{20} \times 3 = \frac{133}{60} \approx 2.127$$

You can reduce these calculations with Knuth's theorem: $L = 1 + \frac{7}{12} + \frac{5}{12} + \frac{13}{60} = \frac{133}{60}$.

c) The combining phase of the Huffman algorithm (with place-high strategy) goes as follows:

Source symbols	Step 0	Step 1	Step 2	Step 3	Step 4
s_1	$p_1 = \frac{1}{2}$	$p_1 = \frac{1}{2}$	$p_1 = \frac{1}{2}$	$p_{4532} = \frac{1}{2}$	$p_{45321} = 1$
s_2	$p_2 = \frac{1}{4}$	$p_2 = \frac{1}{4}$	$p_{453} = \frac{1}{4}$	$p_1 = \frac{1}{2}$	
s_3	$p_3 = \frac{1}{8}$	$p_{45} = \frac{1}{8}$	$p_2 = \frac{1}{4}$		
s_4	$p_4 = \frac{1}{16}$	$p_3 = \frac{1}{8}$			
s_5	$p_5 = \frac{1}{16}$				

Going backwards, the splitting phase of the Huffman algorithm gives the codeword symbols as follows:

Source symbols	Step 0	Step 1	Step 2	Step 3	Step 4
s_1	$p_1 : 1$	$p_1 : 1$	$p_1 : 1$	$p_{4532} : 0$	$p_{45321} : \emptyset$
s_2	$p_2 : 01$	$p_2 : 01$	$p_{453} : 00$	$p_1 : 1$	
s_3	$p_3 : 001$	$p_{45} : 000$	$p_2 : 01$		
s_4	$p_4 : 0000$	$p_3 : 001$			
s_5	$p_5 : 0001$				

In other words, $s_1, \ldots, s_5$ are encoded as $1, 01, 001, 0000, 0001$, respectively.
The expected codeword length is then:

$$L = \frac{1}{2} \times 1 + \frac{1}{4} \times 2 + \frac{1}{8} \times 3 + \frac{1}{16} \times 4 + \frac{1}{16} \times 4 = \frac{15}{8} = 1.875$$

You can reduce these calculations with Knuth's theorem: $L = 1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} = \frac{15}{8}$.

d) The combining phase of the Huffman algorithm (with place-high strategy) goes as follows:

Source symbols	Step 0	Step 1	Step 2	Step 3
s_1	$p_1 = \frac{27}{40}$	$p_1 = \frac{27}{40}$	$p_1 = \frac{27}{40}$	$p_{1234} = 1$
s_2	$p_2 = \frac{9}{40}$	$p_2 = \frac{9}{40}$	$p_{234} = \frac{13}{40}$	
s_3	$p_3 = \frac{3}{40}$	$p_{34} = \frac{1}{10}$		
s_4	$p_4 = \frac{1}{40}$			

Going backwards, the splitting phase of the Huffman algorithm gives the codeword symbols as follows:

Source symbols	Step 0	Step 1	Step 2	Step 3	Step 4
s_1	$p_1 : 0$	$p_1 : 0$	$p_1 : 0$	$p_{1234} : \emptyset$	
s_2	$p_2 : 10$	$p_2 : 10$	$p_{234} : 1$		
s_3	$p_3 : 110$	$p_{34} : 11$			
s_4	$p_4 : 111$				

In other words, $s_1, \ldots, s_5$ are encoded as $0, 10, 110, 111$, respectively.
The expected codeword length is then:

$$L = \frac{27}{40} \times 1 + \frac{9}{40} \times 2 + \frac{3}{40} \times 3 + \frac{1}{40} \times 3 = \frac{57}{40} = 1.425$$

You can reduce these calculations with Knuth's theorem: $L = 1 + \frac{13}{40} + \frac{1}{10} = \frac{57}{10}$.

30. Here, it is a good idea to draw the decision tree arising from the Huffman algorithm. However, I am going to be lazy and just write up the steps without drawing anything.

a) We must find the binary Huffman code for the seven probabilities and also the ternary Huffman code. Let us first find the binary Huffman code:

Source	Step 0	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6
s_1	$p_1 = \frac{1}{3}$ **00**	$p_1 = \frac{1}{3}$ **00**	$p_1 = \frac{1}{3}$ **00**	$p_1 = \frac{1}{3}$ **00**	$p_{34675} = \frac{1}{3}$ **1**	$p_{12} = \frac{2}{3}$ **0**	$p_{1234675} = 1$ $\emptyset$
s_2	$p_2 = \frac{1}{3}$ **01**	$p_2 = \frac{1}{3}$ **01**	$p_2 = \frac{1}{3}$ **01**	$p_2 = \frac{1}{3}$ **01**	$p_1 = \frac{1}{3}$ **00**	$p_{34675} = \frac{1}{3}$ **1**	
s_3	$p_3 = \frac{1}{9}$ **100**	$p_3 = \frac{1}{9}$ **100**	$p_{675} = \frac{1}{9}$ **11**	$p_{34} = \frac{2}{9}$ **10**	$p_2 = \frac{1}{3}$ **01**		
s_4	$p_4 = \frac{1}{9}$ **101**	$p_4 = \frac{1}{9}$ **101**	$p_3 = \frac{1}{9}$ **100**	$p_{675} = \frac{1}{9}$ **11**			
s_5	$p_5 = \frac{1}{27}$ **111**	$p_{67} = \frac{2}{27}$ **110**	$p_4 = \frac{1}{9}$ **101**				
s_6	$p_6 = \frac{1}{27}$ **1100**	$p_5 = \frac{1}{27}$ **111**					
s_7	$p_7 = \frac{1}{27}$ **1101**						

In other words, the binary Huffman code is 00, 01, 100, 101, 111, 1100, 1101, respectively.
By Knuth's theorem, the expected codeword length is

$$L = 1 + \frac{2}{3} + \frac{1}{3} + \frac{2}{9} + \frac{1}{9} + \frac{2}{27} = \frac{65}{27}$$

The average codeword cost with the binary Huffman code is then $\frac{65}{27} \times \$2.00 \approx \$4.81$.

Now let us find the ternary Huffman code. Since $7 \equiv 1 \pmod{(3-1)}$, no dummy symbols are needed.

Source	Step 0	Step 1	Step 2	Step 3
s_1	$p_1 = \frac{1}{3}$ **1**	$p_1 = \frac{1}{3}$ **1**	$p_{56734} = \frac{1}{3}$ **0**	$p_{5673412} = 1$ $\emptyset$
s_2	$p_2 = \frac{1}{3}$ **2**	$p_2 = \frac{1}{3}$ **2**	$p_1 = \frac{1}{3}$ **1**	
s_3	$p_3 = \frac{1}{9}$ **01**	$p_{567} = \frac{1}{9}$ **00**	$p_2 = \frac{1}{3}$ **2**	
s_4	$p_4 = \frac{1}{9}$ **02**	$p_3 = \frac{1}{9}$ **01**		
s_5	$p_5 = \frac{1}{27}$ **000**	$p_4 = \frac{1}{9}$ **02**		
s_6	$p_6 = \frac{1}{27}$ **001**			
s_7	$p_7 = \frac{1}{27}$ **002**			

In other words, the ternary Huffman code is 1, 2, 01, 02, 000, 001, 002, respectively.
By Knuth's theorem, the expected codeword length is

$$L = 1 + \frac{1}{3} + \frac{1}{9} = \frac{13}{9} \approx 1.44$$

The average codeword cost with the ternary Huffman code is then $\frac{13}{9} \times \$3.25 \approx \$4.69$.
It is therefore cheaper to select the ternary service.

b) The binary service would be cheaper if the ternary digit unit price is $\frac{\frac{65}{27} \times \$2.00}{\left(\frac{13}{9}\right)} = \$3.33$ or more.

31. Proof. For $n = 1$, the binary Huffman code for a $2^n = 2$ symbol source, with every symbol having equal probabilities, is certainly a block code of length n: it is just $\{0, 1\}$.
Assume now that this statement is true for some arbitrary positive integer n and consider a 2^{n+1} symbol having equal probabilities. Listing the probabilities as $p_1, \ldots, p_{2^n}, p'_1 \ldots, p'_{2^n}$ (all equal to $\frac{1}{2^{n+1}}$), the Huffman algorithm will first combine the probabilities $p'_1 \ldots, p'_{2^n}$, replacing them by the probability $p' = \frac{1}{2}$, to form the list of probabilities $p', p_1 \ldots, p_{2^n}$. The Huffman algorithm will then proceed to combine $p_1, \ldots, p_{2^n}$, replacing them by the probability $p = \frac{1}{2}$, to form the list of probabilities p, p', both equal to $\frac{1}{2}$. The last combination step produces the root with probability 1.
The first splitting step labels p by 0 and p' by 1. After this, the algorithm labels $p_1, \ldots, p_{2^n}$ first and then labels $p'_1, \ldots, p'_{2^n}$ after that. By our induction assumption, each of these two sub-labellings will result in a block code of length n; together with the initial 0 (or 1), we now have a block code of length $n + 1$. $\square$

32. The problem is slightly unfortunately phrased since the order of probabilities is *non-increasing*, so we have to reverse the order of the probabilities: the ith symbol *from the bottom* has frequency f_i. Therefore *from the top*, the symbols have probabilities $p_i = \frac{f_{n+1-i}}{f_1 + \cdot + f_n} = \frac{f_{n+1-i}}{c}$ where $c = f_{n+2} - 1$.
Let us just use the frequencies $f_1, \ldots, f_n$ from the bottom instead, noting that this is equivalent to looking at $p_1, \ldots, p_n$ from the top. In this light, the Huffman algorithm first adds $f_1 = 1 = f_2$ and $f_2 = 1 = f_3 - 1$, to give $f_1 + f_2 = 2 = f_4 - 1$, and is therefore placed just above f_3 but below $f_4 = 3$. At the next step, $f_4 - 1 = 2$ and $f_3 = 3$ and added together to give $f_3 + f_3 = 4 = f_5 - 1$, and is therefore placed just above f_4 but below f_5. Repeating this, f_{i+1} and $f_{i+2} - 1$ are added to give $f_{i+3} - 1$ which is placed just below f_{i+3}, and so forth. Finally, f_n is added in the very last step.
Going backwards in the splitting phase of the algorithm, we therefore get

$$1,\ 01,\ 001,\ 0001, \ldots, \underbrace{0 \cdots 0}_{n-1}1, \underbrace{0 \cdots 0}_{n}$$

which we recognise as the standard comma code with 0 and 1s swapped.

33. We first use the dummy method with the Huffman algorithm.
Since $8 \equiv -1 \equiv 1-2 \pmod{(4-1)}$, we need two dummy symbols s_9 and s_{10}:

Source symbols	Step 0	Step 1	Step 2	Step 3
s_1	$p_1 = 0.22$ **1**	$p_1 = 0.22$ **1**	$p_{45678910} = 0.40$ **0**	$p_{45678910123} = 1$ $\emptyset$
s_2	$p_2 = 0.20$ **2**	$p_2 = 0.20$ **2**	$p_1 = 0.22$ **1**	
s_3	$p_3 = 0.18$ **3**	$p_3 = 0.18$ **3**	$p_2 = 0.20$ **2**	
s_4	$p_4 = 0.15$ **00**	$p_4 = 0.15$ **00**	$p_3 = 0.18$ **3**	
s_5	$p_5 = 0.10$ **01**	$p_5 = 0.10$ **01**		
s_6	$p_6 = 0.08$ **02**	$p_6 = 0.08$ **02**		
s_7	$p_7 = 0.05$ **030**	$p_{78910} = 0.07$ **03**s_8	$p_8 = 0.02$ **031**	
s_9	$p_9 = 0$ **032**			
s_{10}	$p_{10} = 0$ **033**			

Excluding the last two codewords, the quaternary Huffman code is here 1, 2, 3, 00, 01, 02, 030, 031.
By Knuth's theorem, the expected codeword length is

$$L = 1 + 0.40 + 0.07 = 1.47$$

We now use the Huffman algorithm with "combine 4 symbols as longs as possible" variation:

Source symbols	Step 0	Step 1	Step 2	Step 3
s_1	$p_1 = 0.22$ **10**	$p_{5678} = 0.25$ **0**	$p_{5678} = 0.75$ **0**	$p_{56781234} = 1$ $\emptyset$
s_2	$p_2 = 0.20$ **11**	$p_1 = 0.22$ **10**	$p_{1234} = 0.25$ **1**	
s_3	$p_3 = 0.18$ **12**	$p_2 = 0.20$ **11**		
s_4	$p_4 = 0.15$ **13**	$p_3 = 0.18$ **12**		
s_5	$p_5 = 0.10$ **00**	$p_4 = 0.15$ **13**		
s_6	$p_6 = 0.08$ **01**			
s_7	$p_7 = 0.05$ **02**			
s_8	$p_8 = 0.02$ **03**			

The quaternary Huffman code is now 10, 11, 12, 13, 00, 01, 02, 03, respectively.
By Knuth's theorem, the expected codeword length is

$$L = 1 + 0.25 + 0.25 = 2$$

We see that the dummy method gives the shortest (indeed minimal) expected codeword length.

34. Here, it is a good idea to draw the decision tree arising from the Huffman algorithm. However, I am going to be lazy and just write up the steps without drawing anything.

a) Let us first find the binary Huffman code:

Source	Step 0	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6	Step 7
$\sigma_1 = s_1 s_1 s_1$	$p_1 = \frac{27}{64}$ **1**	$p_1 = \frac{27}{64}$ **1**	$p_1 = \frac{27}{64}$ **1**	$p_1 = \frac{27}{64}$ **1**	$p_1 = \frac{27}{64}$ **1**	$p_1 = \frac{27}{64}$ **1**	$p_{5678234} = \frac{37}{64}$ **0**	$p_{56782341} = 1$ $\emptyset$
$\sigma_2 = s_1 s_1 s_2$	$p_2 = \frac{9}{64}$ **001**	$p_2 = \frac{9}{64}$ **001**	$p_2 = \frac{9}{64}$ **001**	$p_{5678} = \frac{10}{64}$ **000**	$p_{34} = \frac{18}{64}$ **01**	$p_{56782} = \frac{19}{64}$ **00**	$p_1 = \frac{27}{64}$ **1**	
$\sigma_3 = s_1 s_2 s_1$	$p_3 = \frac{9}{64}$ **010**	$p_3 = \frac{9}{64}$ **010**	$p_3 = \frac{9}{64}$ **010**	$p_2 = \frac{9}{64}$ **001**	$p_{5678} = \frac{10}{64}$ **000**	$p_{34} = \frac{18}{64}$ **01**		
$\sigma_4 = s_1 s_2 s_2$	$p_4 = \frac{9}{64}$ **011**	$p_4 = \frac{9}{64}$ **011**	$p_4 = \frac{9}{64}$ **011**	$p_3 = \frac{9}{64}$ **010**	$p_2 = \frac{9}{64}$ **001**			
$\sigma_5 = s_2 s_1 s_1$	$p_5 = \frac{3}{64}$ **00000**	$p_{78} = \frac{4}{64}$ **0001**	$p_{56} = \frac{6}{64}$ **0000**	$p_4 = \frac{9}{64}$ **011**				
$\sigma_6 = s_2 s_1 s_2$	$p_6 = \frac{3}{64}$ **00001**	$p_5 = \frac{3}{64}$ **00000**	$p_{78} = \frac{4}{64}$ **0001**					
$\sigma_7 = s_2 s_2 s_1$	$p_7 = \frac{3}{64}$ **00010**	$p_6 = \frac{3}{64}$ **00001**						
$\sigma_8 = s_2 s_2 s_2$	$p_8 = \frac{1}{64}$ **00011**							

In other words, the binary Huffman code for $\sigma_1, \ldots, \sigma_8$ is

$$1,\ 001,\ 010,\ 011,\ 00000,\ 00001,\ 00010,\ 00011$$

By Knuth's theorem, the expected codeword length is

$$L = 1 + \frac{37}{64} + \frac{19}{64} + \frac{18}{64} + \frac{10}{64} + \frac{6}{64} + \frac{4}{64} = \frac{158}{64} \approx 2.47$$

The average codeword length per binary symbol is $\frac{158}{64}/3 \approx 0.82$.

b) $s_1 s_1 s_2\ s_1 s_2 s_1\ s_1 s_1 s_1\ s_2 s_1 s_1 = \sigma_2 \sigma_3 \sigma_1 \sigma_5 \to 001\,010\,1\,00000$