Solutions to Homework # 4

ECE 6605 Information Theory 10/15/03
Prof. Steven W McLaughlin

1. Following are the tables for each of the three cases, the calculated entropy $H(X) = 2.5546$. The code that generated the results in the tables is added at the end of the solutions.

   a) n=1, calculated $l_x = 2.5833$

   b) n=2, calculated $l_x = 2.5712$

   c) n=3, calculated $l_x = 2.5614$

Label	$C(x)$	l_x
0	10	2
1	11	2
2	000	3
3	001	3
4	010	3
5	011	3

Label	$C(x)$	l_x	Label	$C(x)$	l_x
00	00000	5	30	01010	5
01	00001	5	31	01011	5
02	01100	5	32	10000	5
03	01101	5	33	10001	5
04	10110	5	34	001000	6
05	10111	5	35	001001	6
10	1110	4	40	10100	5
11	1111	4	41	10101	5
12	01000	5	42	000100	6
13	01001	5	43	000101	6
14	11010	5	44	011110	6
15	11011	5	45	011111	6
20	00110	5	50	11000	5
21	00111	5	51	11001	5
22	10010	5	52	001010	6
23	10011	5	53	001011	6
24	000110	6	54	011100	6
25	000111	6	55	011101	6

Label	$C(x)$	l_x	Label	$C(x)$	l_x	Label	$C(x)$	l_x	Label	$C(x)$	l_x	Label	$C(x)$	l_x
000	0011100	7	112	1000000	7	224	10110010	8	340	01010100	8	452	000110010	9
001	0011101	7	113	1000001	7	225	10110011	8	341	01010101	8	453	000110011	9
002	1001000	7	114	00001110	8	230	1111110	7	342	10100110	8	454	011011110	9
003	1001001	7	115	00001111	8	231	1111111	7	343	10100111	8	455	011011111	9
004	00010110	8	120	1000100	7	232	00100110	8	344	000111010	9	500	00000100	8
005	00010111	8	121	1000101	7	233	00100111	8	345	000111011	9	501	00000101	8
010	0011010	7	122	1110110	7	234	10110000	8	350	01000000	8	502	01101000	8
011	0011011	7	123	1110111	7	235	10110001	8	351	01000001	8	503	01101001	8
012	1001010	7	124	01000100	8	240	01100100	8	352	10101000	8	504	11001100	8
013	1001011	7	125	01000101	8	241	01100101	8	353	10101001	8	505	11001101	8
014	00010010	8	130	0111000	7	242	10110110	8	354	000111000	9	510	00001100	8
015	00010011	8	131	0111001	7	243	10110111	8	355	000111001	9	511	00001101	8
020	1001110	7	132	1111010	7	244	001000000	9	400	00000000	8	512	01011000	8
021	1001111	7	133	1111011	7	245	001000001	9	401	00000001	8	513	01011001	8
022	1111000	7	134	00111110	8	250	01001110	8	402	01011010	8	514	11000110	8
023	1111001	7	135	00111111	8	251	01001111	8	403	01011011	8	515	11000111	8
024	01100110	8	140	00010000	8	252	10101010	8	404	11000000	8	520	01010010	8
025	01100111	8	141	00010001	8	253	10101011	8	405	11000001	8	521	01010011	8
030	1001100	7	142	01100010	8	254	001000100	9	410	00000110	8	522	10101100	8
031	1001101	7	143	01100011	8	255	001000101	9	411	00000111	8	523	10101101	8
032	1111100	7	144	11001000	8	300	1000110	7	412	01011100	8	524	000111100	9
033	1111101	7	145	11001001	8	301	1000111	7	413	01011101	8	525	000111101	9
034	01001010	8	150	00010100	8	302	1110000	7	414	10111010	8	530	01010000	8
035	01001011	8	151	00010101	8	303	1110001	7	415	10111011	8	531	01010001	8
040	00001010	8	152	01001000	8	304	01000110	8	420	01101010	8	532	10100000	8
041	00001011	8	153	01001001	8	305	01000111	8	421	01101011	8	533	10100001	8
042	01001100	8	154	11001110	8	310	0111010	7	422	10100100	8	534	000110000	9
043	01001101	8	155	11001111	8	311	0111011	7	423	10100101	8	535	000110001	9
044	11000010	8	200	0111110	7	312	1101010	7	424	001000010	9	540	10111100	8
045	11000011	8	201	0111111	7	313	1101011	7	425	001000011	9	541	10111101	8
050	00000010	8	202	1101110	7	314	01011110	8	430	01100000	8	542	000110100	9
051	00000011	8	203	1101111	7	315	01011111	8	431	01100001	8	543	000110101	9
052	00111100	8	204	01010110	8	320	1101000	7	432	10110100	8	544	011011000	9
053	00111101	8	205	01010111	8	321	1101001	7	433	10110101	8	545	011011001	9
054	11001010	8	210	0111100	7	322	00101010	8	434	001000110	9	550	10111110	8
055	11001011	8	211	0111101	7	323	00101011	8	435	001000111	9	551	10111111	8
100	0011000	7	212	1110010	7	324	10101110	8	440	11000100	8	552	000110110	9
101	0011001	7	213	1110011	7	325	10101111	8	441	11000101	8	553	000110111	9
102	1000010	7	214	01000010	8	330	1101100	7	442	000111110	9	554	011011100	9
103	1000011	7	215	01000011	8	331	1101101	7	443	000111111	9	555	011011101	9
104	00001000	8	220	1110100	7	332	00101000	8	444	011011010	9			
105	00001001	8	221	1110101	7	333	00101001	8	445	011011011	9			
110	0010110	7	222	00100100	8	334	10100010	8	450	10111000	8			
111	0010111	7	223	00100101	8	335	10100011	8	451	10111001	8			

2. We have alphabet $A = \{a,b,c\}$ and $p(a) = 1/12$, $p(b) = 1/3$ and $p(c) = 7/12$. Therefore we can calculate the lengths using $\lceil -\log_2 p(x) \rceil + 1$ and we get $\{5,3,2\}$. We then obtain the expansions using the function dectobin.m at the end of the solutions and we find $a \to 00010$, $b \to 010$ and $c \to 10$. We find that $H(X) = 1.2807$ and the average codeword length is 2.5833

3. Yes, $L = (1,2,2)$ can be the word lengths of a binary Huffman code, in fact they are the lengths for the code in which the probability vector is $\{1/2, 1/4, 1/4\}$.

   No, $L = (2,2,3,3)$ can not be the set of word lengths for a binary Huffman code. Since the last two words differ only in the last bit, then we have only used one of four possible two bit sequences, assume without loss of generality that we used 00, then we can use 01 for one of the other two codewords. This leaves us with the choices 11,10 for the other codeword, but this means we can choose just 1 to represent the codeword and still guarantee the code is prefix free. In fact one can show that the only possible codeword sets of codeword lengths for binary Huffman codes with four codewords are $L = (2,2,2,2)$ or $L = (1,2,3,3)$.

   Yes, this is true. To see this we realize that a binary Huffman code is built on a binary tree in such a way that one of two things happen a) the codewords are all the branches at one level l_{max} in the tree or b) there are at least two codewords at level l_{max} on the tree and all other brances at that level are either codewords (always in pairs) or descendants of codewords. The key to the matter here is that codewords at level l_{max} most occur in pairs. For any tree and any l_{max} we have $\sum_{i=1}^{2^{l_{max}}} 2^{l_i} = 1$. Therefore from what was described above for a Huffman code $\sum_{i=1}^{n} 2^{l_i} = \sum_{i=1}^{2^{l_{max}}} 2^{l_i} = 1$.

4. a) First notice that the X_i's are i.i.d. therefore $H(\mathscr{X}) = H(X_1)$ and we have $H(\mathscr{X}) = H(\frac{1}{2},\frac{1}{4},\frac{1}{4}) = \frac{1}{2}\log_2 2 + \frac{2}{4}\log_2 2 = \frac{1}{2} + 1 = \frac{3}{2}$. Now we also notice that the average codeword length $E[L(C(X))] = \sum_{i=1}^{3} p_i l_i = \frac{1}{2} + 2*\frac{1}{4} + 2*\frac{1}{4} = \frac{3}{2} = H(\mathscr{X})$ so the source can't be compressed any further. This in turn should intituively tell us that the Z_i's are then an i.i.d. Bernoulli process with $p = 1/2$ so that $H(\mathscr{Z}) = 1$.

   We can show it another way basically it is not hard to show that $P(Z_n|Z_1,\ldots,Z_{n-1}) = P(Z_n)$ and that $P(Z_n = 0) = 1/2$ and $P(Z_n = 1) = 1/2$, therefore $H(\mathscr{Z}) = 1$. Intituitevely one can see this by considering two cases, if we are starting a new word, then $P(Z_n = 0) = P(A) = 1/2$, if we are not starting a new word, then $P(Z_n = 0) = 1/2$ since we are either on the second bit of C(B) or of C(A) and both occur with equal probability.

   b) We can find $H'(\mathscr{Z}) = \lim_{n\to\infty} H(Z_n|Z_{n-1},\ldots,Z_1)$ and by the Cesaro mean theorem this $H'(\mathscr{Z}) = H(\mathscr{Z})$ if the above limit exists. To find $H'(\mathscr{Z})$ we will define the variable Y_n as follows:

$$Y_n = \begin{cases} 0 & \text{If } Z_1,\ldots,Z_{n-1} \text{ end a codeword, i.e. } Z_n \text{ is the beginning of a codeword} \\ 1 & \text{If } Z_1,\ldots,Z_{n-1} \text{ are in the middle of a codeword} \end{cases}$$

   Then Y_n is a function of $Z_1,\ldots,Z_{n-1}$ and given Y_n we have that Z_n and $Z_1,\ldots,Z_{n-1}$ are conditionally independent. Therefore

$$\begin{aligned} H(Z_n|Z_{n-1},\ldots,Z_1) &= H(Z_n|Y_n) \\ &= \sum_y H(Z_n|Y_n = y)P(Y_n = y) \\ &= H(Z_n|Y_n = 0)P(Y_n = 0) + H(Z_n|Y_n = 1)P(Y_n = 1) \end{aligned}$$

$$= \quad H(\frac{1}{4},\frac{3}{4})P(Y_n=0)+H(\frac{1}{3},\frac{2}{3})P(Y_n=1)$$

Where the entropies are found based on the probabilites of $Z_n=\{0,1\}$ given Y_n.
Basically we can think of Y_n as a Markov chain with transition probability matrix given by $T=\begin{bmatrix} 1/4 & 3/4 \\ 1 & 0 \end{bmatrix}$, therefore since the MC is aperiodic and irreceducible we can find the stationary distribution μ. It turns out that $\mu=(4/7,3/7)$. Then we can use this result as $P(Y_n=0)$ and $P(Y_n=1)$ respectively. Then

$$\begin{aligned}
\lim_{n\to\infty} H(Z_n|Z_1,\ldots,Z_{n-1}) &= H(\frac{1}{4},\frac{3}{4})P(Y_n=0)+H(\frac{1}{3},\frac{2}{3})P(Y_n=1) \\
&= H(\frac{1}{4},\frac{3}{4})\frac{4}{7}+H(\frac{1}{3},\frac{2}{3})\frac{3}{7} \\
&= (2-\frac{3}{4}\log_2 3)\frac{4}{7}+(\log_2 3-\frac{2}{3})\frac{3}{7} \\
&= \frac{6}{7}
\end{aligned}$$

Therefore $H(\mathscr{Z})=\frac{6}{7}$. Notice that in this case $E[L(C(X))]=7/4$ and therefore $H(\mathscr{X})=E[L(C(X))]H(\mathscr{Z})$ this is always true as long as X has a finite alphabet and $C(X)$ is uniquely decodable and with finite length, i.e. the longest $C(X)$ is not infinite.
Intituitevely we can show this using the fact that $C(X)$ is uniquely decodable and $(X_1,\ldots,X_n)$ is mapped to $(Z_1,\ldots,Z_M)$ where $M=\sum_{i=1}^{n} L(C(X_i))$. It might then be plausible that, by the stron law of large numbers

$$\begin{aligned}
\frac{H(X_1,\ldots,X_n)}{n} &= \frac{H(Z_1,\ldots,Z_M)}{n} \\
&= \frac{M}{n}\frac{H(Z_1,\ldots,Z_M)}{M} \\
&\to E[L(C(X))]H(\mathscr{Z}) \text{as } n\to\infty
\end{aligned}$$

Note that here M is actually a random variable, therefore the above argument does not constitute a rigorous proof, such a proof is involved and is therefore not included, but the conclusion reached is true, even though the steps might be misleading.

CODE FOR PROBLEMS 1 and 2
The only functions that you need to use are huffman.m, lx_calc.m, disp_code.m, disp_code2.m and prob_n_cr.m. Use prob_n_cr.m to create the probability vector for a specific n from the vector for n=1, then use huffman.m to create the code, lx_calc.m to calculate the average codeword length and disp_code.m or disp_code2.m to display the results.
For problem 2 use dectobin.m at the end of this section

```
function out=add_value_struct(in,st,len)

vec=st.label;
```

```
t=length(vec);

out=in;
for i=1:len
  out(vec(i)).value=['0' out(vec(i)).value ];
end
for i=len+1:t
  out(vec(i)).value=['1' out(vec(i)).value];

---------------------------------

function vec=allperm(l,a);
%given alphabet a and length l, generates all posible
%vectors of length l with elements from a
%returns a^l vectors

a=a(:);
if (l==1)
  vec=a;
  return
else
  tv=allperm(l-1,a);
  vec=[];
  for i=1:length(a)
    vec=[vec; tv a(i)*ones(size(tv,1),1)];
  end;
end;

---------------------------------

function [comb_st,n]=comb_struct(init_st)

l=length(init_st);
if (l>2)
  for i=1:l-2
    comb_st(i)=init_st(i);
  end;
  i=i+1;
else
  i=1;
end;
comb_st(i)=struct('label',[init_st(i).label init_st(i+1).label],'value',...
   init_st(i).value+init_st(i+1).value);
n=length(init_st(i).label);
```

```
----------------------------------

function disp_code(code)

lt=length(code)
for i=1:lt
   t=length(str2mat(code(i).value));
   disp(['Label=',int2str(code(i).label),' Codeword= ',code(i).value,...
       ' Length= ',int2str(t)]);
end;

----------------------------------

function disp_code2(code,n)

if nargin<2
  n=1;
end
lt=length(code);
for i=1:lt
   t=length(str2mat(code(i).value));
%   disp([int2str(code(i).label),\t,code(i).value,\t,int2str(t)]);
   vec=de2bi(code(i).label-1,n,6,'left-msb');
   l=' ';
   for j=1:n
     l=[l int2str(vec(j))];
   end
    fprintf('%3s & %10s & %2d \\\\ \n',l,code(i).value,t);

-------------------------------------

function filled=fill_struct(vec,value);

s=struct('label',vec(1),'value',value);
t=s;
for i=2:length(vec)
    s=struct('label',vec(i),'value',value);
    t=[t s];
end
filled=t;

-------------------------------------

function filled=fill_vec_struct(vec,value);
```

```
s=struct('label',vec(1),'value',value(1));
t=s;
for i=2:length(vec)
    s=struct('label',vec(i),'value',value(i));
    t=[t s];
end
filled=t;

-----------------------------------

function lx=lx_calc(code,probs)

lt=length(code)
lx=0;
for i=1:lt
   t=length(str2mat(code(i).value));
   lx=lx+t*probs(i);
end;

-----------------------------------

function probs=mult_probs(matr,prob)

[m,n]=size(matr);
probs=zeros(1,m);
for i=1:m
  prod=1;
  for j=1:n
    prod=prod*prob(matr(i,j));
  end
  probs(i)=prod;
end

-----------------------------------

function [probs,vec]=prob_n_cr(prob,n)

lt=0:length(prob)-1;
allp=allperm(n,lt);
vec=bi2de(allp,length(prob))+1;
vec=vec';
probs=mult_probs(allp+1,prob);

-------------------------------
```

```
function [y,ind]=sort_slow(t)

i=max(t);
indp=find(t==i);
ind=indp;
y=t(indp);
np=t;
pos=1:length(t);
np(indp)=[];
pos(indp)=[];
while (~isempty(np))
   i=max(np);
   indp=find(np==i);
   ind=[ind pos(indp)];
   y=[y t(pos(indp))];
   np(indp)=[];
   pos(indp)=[];
end;

----------------------------

function ind=sort_struct(name)

t=zeros(1,length(name));
for i=1:length(t)
    t(i)=name(i).value;
end;
[y,ind]=sort_slow(t);

-----------------------------------

function code=huffman(prob)
%gives huffman code of n symbols with the symbol probabilities
%specified by prob

lab=1:length(prob);
code=fill_struct(lab,'');
init_st=fill_vec_struct(lab,prob);
ind=sort_struct(init_st);
sort_st(ind)=init_st;
comb_st=comb_struct(sort_st);
code=add_value_struct(code,comb_st(end),1);
i=1;
clear sort_st;
while (length(comb_st)>1)
```

```
   ind=sort_struct(comb_st);
   sort_st=comb_st(ind);
   [comb_st,len]=comb_struct(sort_st);
   code=add_value_struct(code,comb_st(end),len);
   i=i+1
   clear sort_st;
end

-----------------------------

function bina=dectobin(dec,l)
%comments are not included

temp=dec;
bina=zeros(1,l);
for i=1:l
  if (temp>(1/(2^i)))
     bina(i)=1;
     temp=temp-(1/(2^i));
  else
     bina(i)=0;
  end
  if (temp==0)
    break
  end
end
```