ECE401/8001
VISUAL SIGNAL PROCESSING AND COMMUNICATIONS

Winter 2005

Prof.: Dr. Zhihai (Henry) He

Project

Phase I: Huffman Coding

02/03/2005

Luis M Vicente
Student number: 945-995
E-mail: lmvy8c@mizzou.edu

Luis M Vicente Student# 945-995 2/5/2005
lmvy8c@mizzou.edu

Purpose
Design a Huffman code with Matlab or C.
Procedure
Step 1: Write a function to find the probability of English characters a-z, A-Z, etc… in the article "state.txt"
Step 2: Construct the Huffman code word table.
Step 3: Compute the average length

Step 1: (approach of the Problem)

First of all I have to read the "state.txt" file into Matlab. (Using *fopen* and *fscanf*).
The whole training text is stored in a string variable I named *sffile***.** The number of characters is stored in a variable named *N***.** In this particular file there are **45592** symbols.

Next, we have to count the symbols, and what is their frequency in the training text to find the probability distribution.

Before counting the symbols, I have to create an alphabet. I created an alphabet inside the function *enumsymbols()* where I stored all characters in a string vector. I included the "special" characters inside a symbol called '§' that accounts for all special symbols. The total number of symbols in my alphabet is 96.

Originally I used a "for loop" with a "switch" statement to find all symbols and their occurrence. However, I found a problem reading special characters as new line etc.
Trying to optimize the code, I used the function *findstr* along with the function *length* to find the occurrence of each symbol. I had the same problem counting the special characters. Therefore as per Dr. He recommendation, all unknown symbols were assigned to a special character '§'. I implemented function *countocurrence(s,Nsym,sffile)*; to count all symbols. The occurrence and the relative frequency (we will use this number as the probability) is shown in the appropriate columns of Table 1.

I found that state.txt does not have all the 96 symbols from the alphabet created; therefore the occurrence, and hence, probability of some symbols is zero, leading the length to "infinite" if we use the Shannon formula $l_i = \left\lceil \log_2 \frac{1}{p_i} \right\rceil$. One form of dealing with this problem is to add at least one character to the training text. However I do not know what would be the optimization of doing this. I will investigate this at the end of my project.

order	symbol	Occur.	Rel. Freq (prob)		order	symbol	Occur.	Rel. Freq (prob)
1	a	2817	0.0618		49	W	42	0.0009
2	b	374	0.0082		50	X	1	2.2e-005
3	c	1121	0.0246		51	Y	19	0.0004
4	d	1364	0.0299		52	Z	0	0
5	e	4498	0.0987		53	Space	7561	0.1658
6	f	761	0.0167		54	.	519	0.0114
7	g	728	0.016		55	,	456	0.01
8	h	1575	0.0345		56	/	2	4.4e-005
9	i	2645	0.058		57	<	0	0
10	j	23	0.0005		58	>	0	0
11	k	180	0.0039		59	?	9	0.0002
12	l	1406	0.0308		60	;	10	0.0002
13	m	913	0.02		61	'	53	0.0012
14	n	2725	0.0598		62	:	37	0.0008
15	o	2788	0.0612		63	"	83	0.0018
16	p	709	0.0156		64	[	61	0.0013
17	q	55	0.0012		65	]	61	0.0013
18	r	2388	0.0524		66	{	0	0
19	s	2390	0.0524		67	}	0	0
20	t	3158	0.0693		68	\	0	0
21	u	943	0.0207		69	|	0	0
22	v	384	0.0084		70	!	4	0.0001
23	w	606	0.0133		71	@	0	0
24	x	59	0.0013		72	#	0	0
25	y	515	0.0113		73	$	2	4.4e-005
26	z	33	0.0007		74	%	0	0
27	A	168	0.0037		75	^	0	0
28	B	28	0.0006		76	&	0	0
29	C	36	0.0008		77	*	2	4.4e-005
30	D	28	0.0006		78	(	2	4.4e-005
31	E	17	0.0004		79	)	2	4.4e-005
32	F	18	0.0004		80	-	31	0.0007
33	G	16	0.0004		81	=	0	0
34	H	51	0.0011		82	_	0	0
35	I	139	0.003		83	+	0	0
36	J	7	0.0002		84	`	0	0
37	K	11	0.0002		85	~	0	0
38	L	9	0.0002		86	0	72	0.0016
39	M	27	0.0006		87	1	35	0.0008
40	N	44	0.001		88	2	16	0.0004
41	O	27	0.0006		89	3	7	0.0002
42	P	39	0.0009		90	4	7	0.0002
43	Q	10	0.0002		91	5	11	0.0002
44	R	15	0.0003		92	6	6	0.0001
45	S	108	0.0024		93	7	2	4.4e-005
46	T	101	0.0022		94	8	4	0.0001
47	U	35	0.0008		95	9	17	0.0004
48	V	3	0.0001		96	Schar	363	0.008

Table 1: Symbol Table, with occurrences and relative frequency

With this table, and the function in Appendix 1, **step 1** of the project is concluded.

Step 2 and 3: (Huffman Algorithm – average length)

The proposed Huffman algorithm works as follows: from top to bottom we search for the two minimum symbols with less probability. We mark those symbols as counted and create a new super symbol with both symbols and probability the sum of the probabilities. Then repeat the process until completion of the unmarked symbols. If there are 5 symbols we have to do 4 comparisons.

Example:

symbols	prob
a	.25
b	.25
c	.20
d	.15
e	.15

Table 2: example Symbol Table, with probabilities

The algorithm finds **d** and then finds **e**. Assigns 0 to d and 1 to e
Then, update the symbol table with a super symbol with probability the sum of the probabilities of the found symbols. In order to make the algorithm work, I set the probabilities of the found symbols an impossible number (like 2) in order to be larger than the maximum probability (this is a way of marking them as counted).

symbols	prob	Code
a	.25	
b	.25	
c	.20	
d	**2**	0
e	**2**	1
de	.30	

Table 3: example Symbol Table, with probabilities and code

Now it repeats itself. At the second round we find **c** and **a**. Assigns 0 to c and 1 to a

symbols	prob	Code
a	**2**	1
b	.25	
c	**2**	0
d	**2**	0
e	**2**	1
de	.30	
ca	.45	

Table 4: example Symbol Table, with probabilities and code

At the third round it finds **b** and **de**. Assigns 0 to b and 1 to both d and e.

symbols	prob	Code
a	**2**	1
b	**2**	0
c	**2**	0
d	**2**	10
e	**2**	11
de	**2**	
ca	.45	
bde	.55	

Table 5: example Symbol Table, with probabilities and code

At the last round (N-1), it finds **ca** and **bde**. Assigns 0 to c and a and 1 to b, d and e

symbols	prob	Code
a	**2**	01
b	**2**	10
c	**2**	00
d	**2**	110
e	**2**	111
de	**2**	
ca	**2**	
bde	**2**	
cabde	1	

Table 6: example Symbol Table, with probabilities and code

The final code is shown next:

symbols	code
a	01
b	10
c	00
d	110
e	111

Table 7: Huffman code for this example.

I coded this algorithm, and came out with the Huffman code. The Matlab code it is written in Appendix 1 (look for fhuffman function). The Huffman code for this project is written in Table 8. The hexadecimal representation of each codeword is displayed in Appendix 1.

Observations: Analyzing the code, I found that the maximum length is 21. It seems is a pretty big number for just 96 symbols. Let's find the average length or bit rate, defined as the summation of the product of the probability of a symbol with the length of the code for that symbol.

$$L = \sum_{i=1}^{N} p_i l_i = 4.4563 \quad \text{(Step 3)}$$

This seems ok with me, since if we look in Table 8, the symbols with more occurrences (space, e) have code length of 3. Other frequent symbols (a, i, n, o etc) have length of 4. The symbols with less frequency (c, f) have lengths of 5 and 6. Finally, symbols that do not appear (frequency 0) have lengths of 20 and 21.

I added to the training file the missing characters (Z, <, >, {, }, \, |, @, #, ^, &, =, _, +, `, ~), to see if we improved the efficiency and I found that in this case even when the maximum length reduces from 21 to 16; the bit rate or average length raises from 4.4563 to 4.4607. Therefore I think there is not optimization in adding the missing symbols to the training file.

order	symbol	Occur.	Huffman Code.	length		order	symbol	Occur.	Huffman Code.	length
1	a	2817	'1001'	4		49	W	42	'0110001011'	10
2	b	374	'1011000'	7		50	X	1	'1101110101000001'	16
3	c	1121	'00100'	5		51	Y	19	'01100010001'	11
4	d	1364	'01101'	5		52	Z	0	'110111010100000011110'	21
5	e	4498	'000'	3		53	Space	7561	'111'	3
6	f	761	'101101'	6		54	.	519	'001010'	6
7	g	728	'101011'	6		55	,	456	'1101001'	7
8	h	1575	'10111'	5		56	/	2	'110111010100001'	15
9	i	2645	'0101'	4		57	<	0	'110111010100000011111'	21
10	j	23	'10101000110'	11		58	>	0	'11011101010000000000'	20
11	k	180	'10101001'	8		59	?	9	'011000100001'	12
12	l	1406	'10100'	5		60	;	10	'011000101010'	12
13	m	913	'110101'	6		61	'	53	'1101000011'	10
14	n	2725	'0111'	4		62	:	37	'0110000111'	10
15	o	2788	'1000'	4		63	"	83	'101010000'	9
16	p	709	'011001'	6		64	[	61	'1101110011'	10
17	q	55	'1101000111'	10		65	]	61	'1101110100'	10
18	r	2388	'0011'	4		66	{	0	'11011101010000000001'	20
19	s	2390	'0100'	4		67	}	0	'11011101010000000010'	20
20	t	3158	'1100'	4		68	\	0	'11011101010000000011'	20
21	u	943	'110110'	6		69	|	0	'11011101010000000100'	20
22	v	384	'1011001'	7		70	!	4	'11011101010001'	14
23	w	606	'001011'	6		71	@	0	'11011101010000000101'	20
24	x	59	'1101110001'	10		72	#	0	'11011101010000000110'	20
25	y	515	'1101111'	7		73	$	2	'110111010100110'	15
26	z	33	'11011101011'	11		74	%	0	'11011101010000000111'	20
27	A	168	'01100011'	8		75	^	0	'11011101010000001000'	20
28	B	28	'11011100000'	11		76	&	0	'11011101010000001001'	20
29	C	36	'0110000110'	10		77	*	2	'110111010100111'	15
30	D	28	'11011100001'	11		78	(	2	'01100010100000'	14
31	E	17	'110111011001'	12		79	)	2	'01100010100001'	14
32	F	18	'110111011111'	12		80	-	31	'11011100101'	11
33	G	16	'110111010101'	12		81	=	0	'11011101010000001010'	20
34	H	51	'1101000010'	10		82	_	0	'11011101010000001011'	20
35	I	139	'01100000'	8		83	+	0	'11011101010000001100'	20
36	J	7	'1010100011111'	13		84	`	0	'11011101010000001101'	20
37	K	11	'011000101011'	12		85	~	0	'11011101010000001110'	20
38	L	9	'011000100000'	12		86	0	72	'011000010'	9
39	M	27	'11010001100'	11		87	1	35	'11011101110'	11
40	N	44	'1010100010'	10		88	2	16	'110111011000'	12
41	O	27	'11010001101'	11		89	3	7	'1101110010000'	13
42	P	39	'0110001001'	10		90	4	7	'1101110010001'	13
43	Q	10	'011000101001'	12		91	5	11	'101010001110'	12
44	R	15	'110111001001'	12		92	6	6	'1010100011110'	13
45	S	108	'110100010'	9		93	7	2	'01100010100010'	14
46	T	101	'110100000'	9		94	8	4	'11011101010010'	14
47	U	35	'11011101101'	11		95	9	17	'110111011110'	12
48	V	3	'01100010100011'	14		96	Schar	363	'1010101'	7

Table 8: Huffman code for the project showing the codeword and the length of each codeword.

APPENDIX 1

This is the Matlab program I wrote for this project.
It consists of several functions I will summarize here:
main() is the main function, it calls the other functions and perform the display and final computations.
readfile() this function read the characters of the training text file
enumsymbols() this function creates the dictionary of symbols
countocurrence(s,Nsym,sffile) this function counts the symbols from the training file and gives how many symbols of the dictionary are in the file.

fhuffman(st,p) this function performs the Huffman algorithm

```
%mreadf05 (The last number is the version of this code)
%read all characters in a file, perform the Huffman code and display
%Created 2/3/2005 by Luis M Vicente SN: 945-995
%Deficiencies in version 04: I am going to do a new version because I want
%to read the special characters in a supper symbol. I can not use n(i) =
%length(findstr(sffile,s(i))); because it does not account for unknown
%symbols. I will use a switch statement or something else and all unnkwon
%characters will be stored in a special character symbol
%Creating version mreadf05

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function []=main()
clc        %clear screen
close all   %close all figures
clear all   %cean workspace
disp(['START OF MAIN %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%']);

%Function Read file and store it in array sffile. N is the # of characters
[sffile,N] = readfile();

%Function to create a look up table of all symbols
[s,Nsym]=enumsymbols();

%Function to count the occurrence of each symbol in the training set
%store it in a vector that is sincronized with the symbol vector
n = countocurrence(s,Nsym,sffile);

nt=n'; %column vector
st=s'; %column vector

%Find the probabilities
p = nt/N;

%This is just to display the symbol table, the occurrence, and the
%probability
mcell=cell(length(st),2);
mcell(:,1)=cellstr(st);
mcell(:,2)=num2cell(nt);
mcell(:,3)=num2cell(p);

%Display symbol, occurrence and probability
disp(' ');
disp('   Symbol, [Occurrence], [Probability]');
mcell

disp('End of Step 1 of the project. ');
disp('...To continue with Step 2, please press "Enter"');
pause

%Step 2 Function to create Huffmann code
```

Luis M Vicente Student# 945-995
lmvy8c@mizzou.edu


```
[codem] = fhuffman(st,p);

%Adding the code to display later
mcell(:,4)=codem(:,2);

%sum of all probabilities, =1??
disp(' ');
disp(['Sum of all probabilities : ', num2str(sum(p,1))]);

%computing the length of each symbol
for i=1:Nsym
   len(i,1) = length(char(codem(i,2)));
end

%Display symbol, occurrence, probability, codeword and length
disp(' ');
disp('   Symbol, [Occurrence], [Probability] [codeword]        [hex]     [length]');

mcell(:,5)=cellstr(dec2hex(bin2dec(char(mcell(:,4)))));
mcell(:,6)=num2cell(len)

%Bit rate or Average Length
Lengthaverage = sum(p.*len ,1);
disp(['Bit rate or Average Length : ', num2str(Lengthaverage)]);

%Max length
maxlength = max(len);
disp(['Max codeword Length : ', num2str(maxlength)]);


%Lets find how many words have length of 16 15 14 etc.
for i=1:maxlength
   maxl(i,1)=0;
   for j=1:Nsym
      if len(j)==i
         maxl(i,1)=maxl(i,1)+1;
      end
   end
end

%Take out the semicolon if you want to display how many words have lengths
%of 1 2 3 up to max length
maxl;

%This is a checkup to see if the characters read are the same as the
%total characters in the training file
if sum(n,2) ~= N
   disp('Error counting characters in this file');
else
   disp(['Characters read: ', num2str(N)]);
end

disp(['END OF MAIN %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%']);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [sffile,N] = readfile()
%Read the Training set,
%OUTPUT: sffile (string) containing all symbols read and N (integer)
%with number of characters read in

%Filename
strf = 'state.txt';

%Open the file
fid = fopen(strf);
if(fid ~= -1)
   disp(['File ',strf,' succesfully opened']);
else
   disp(['ERROR OPENING FILE ',strf]);
   return
```

```
end

%Read text from file and count characters
[sffile,N] = fscanf(fid,'%c');

%closing the file
if(fclose(fid) ==0)
   disp(['File ',strf,' succesfully closed']);
else
   disp(['ERROR CLOSING File ',strf]);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [s,Nsym]=enumsymbols()
%Enumeration of symbols/ Creation of the alphabet
%OUTPUT: s (is a cell of strings) and Nsym (integer) the number of
%symbols in the alphabet

%to save space I wrote them in the way it is shown.
%Orthodox programmers forgive me!!
s(1) =  {'a'};s(2) =  {'b'};s(3) =  {'c'};s(4) =  {'d'};
s(5) =  {'e'};s(6) =  {'f'};s(7) =  {'g'};s(8) =  {'h'};
s(9) =  {'i'};s(10) = {'j'};s(11) = {'k'};s(12) = {'l'};
s(13) = {'m'};s(14) = {'n'};s(15) = {'o'};s(16) = {'p'};
s(17) = {'q'};s(18) = {'r'};s(19) = {'s'};s(20) = {'t'};
s(21) = {'u'};s(22) = {'v'};s(23) = {'w'};s(24) = {'x'};
s(25) = {'y'};s(26) = {'z'};s(27) = {'A'};s(28) = {'B'};
s(29) = {'C'};s(30) = {'D'};s(31) = {'E'};s(32) = {'F'};
s(33) = {'G'};s(34) = {'H'};s(35) = {'I'};s(36) = {'J'};
s(37) = {'K'};s(38) = {'L'};s(39) = {'M'};s(40) = {'N'};
s(41) = {'O'};s(42) = {'P'};s(43) = {'Q'};s(44) = {'R'};
s(45) = {'S'};s(46) = {'T'};s(47) = {'U'};s(48) = {'V'};
s(49) = {'W'};s(50) = {'X'};s(51) = {'Y'};s(52) = {'Z'};
s(53) = {' '}; %Space character
s(54) = {'.'};s(55) = {','};s(56) = {'/'};s(57) = {'<'};
s(58) = {'>'};s(59) = {'?'};s(60) = {';'};s(61) = {''''};
s(62) = {':'};s(63) = {'"'};s(64) = {'['};s(65) = {']'};
s(66) = {'{'};s(67) = {'}'};s(68) = {'\'};s(69) = {'|'};
s(70) = {'!'};s(71) = {'@'};s(72) = {'#'};s(73) = {'$'};
s(74) = {'%'};s(75) = {'^'};s(76) = {'&'};s(77) = {'*'};
s(78) = {'('};s(79) = {')'};s(80) = {'-'};s(81) = {'='};
s(82) = {'_'};s(83) = {'+'};s(84) = {'`'};s(85) = {'~'};
s(86) = {'0'};s(87) = {'1'};s(88) = {'2'};s(89) = {'3'};
s(90) = {'4'};s(91) = {'5'};s(92) = {'6'};s(93) = {'7'};
s(94) = {'8'};s(95) = {'9'};
s(96) = {'§'}; %This accounts for all special characters

%For the near future, to add new characters to the alphabet
% s(96) = (char(10)); %New Line?
% s(97) = (char(13)); %New Line?
% s(98) = (char(151)); %Strange square

Nsym = length(s);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [n] = countocurrence(s,Nsym,sffile)
%function [n] = countocurrence(s,Nsym,sffile)
%Count the occurrence of each symbol in the training set except the special
%character symbol
%store it in a vector that is sincronized with the symbol vector
%INPUTS: s (cell of characters - symbols), Nsym ( integer - number of
%symbols), sffile (char array - text having the symbols to count)
%OUTPUT: n (array of integers -Count of each symbol indexed)
for i=1:Nsym-1
   %Count the known symbols
   n(i) = length(findstr(sffile,char(s(i))));
end

%Unknown symbols = total symbols - known symbols
n(Nsym)=length(sffile)-sum(n,2);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [codem] = fhuffman(s,p)
%function [codem] = fhuffman(s,p)
%This algoritm creates a Huffman code from s (a cell of strings) and a
%p (vector of floats) which is synchronized with s. meaning p(1) is the
%probability of symbol s(1).
%This algorithm uses string comparison, therefore the condition for vector
%s is that all characters should be SINGLE CHARACTER and different from
%each other.
%The outpus is a cell with two columns, first has the symbols and the
%second column has the codeword
%Example fhuffman({'a';'b';'c';'d';'e'},[.25;.25;.2;.15;.15])


%length of the diccionary
len = length(s);

%CodeMatrix. First column is initialized to the symbols in s,
%second column is emptied will be used to store the Huffman code as a
%binary string (string of 0's and 1's)
codem=cell(len,2);
codem(:,1)=cellstr(s);
codem(:,2)={''};

%We will modify the p vector, so we create a copy to work with it
pwork = p;
%We will modify the s vector, so we create a copy to work with it
swork = s;

%Find first two symbols with the smallest probability
%take that value from the vector, and repeat
%until we have counted all probabilities.
for i=1:len-1
    %first minimum symbol
    [pmin1,ind0]=min(pwork);
    pwork(ind0)=2; %Replace with an impossible probability
    %second minimum symbol
    [pmin2,ind1]=min(pwork);
    pwork(ind1)=2; %Replace with an impossible probability

    %Add probabilities
    pnew = pmin1+pmin2;

    %update pwork adding at the end of the vector the new
    %supersymbol and the new probability
    lsw = length(swork);
    pwork(lsw+1,1)=pnew;

    %Assign bits to the selected symbols.
    for j=1:len
        if double(findstr(char(swork(ind0)),char(s(j)))) >=1
            %disp(['Assign 0 to ',s(j)])
            codem(j,2)={['0',char(codem(j,2))]};
        end

        if double(findstr(char(swork(ind1)),char(s(j)))) >=1
            %disp(['Assign 1 to ',s(j)])
            codem(j,2)={['1',char(codem(j,2))]};
        end
    end

    %Create new super symbol
    ssym = {[char(swork(ind0)),char(swork(ind1))]};

    swork(lsw+1,1)=ssym;

end

%EOM
```

The actual screenshot when running this program in Matlab is shown next:

```
START OF MAIN %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
File state.txt succesfully opened
File state.txt succesfully closed

   Symbol, [Occurrence], [Probability]

mcell =

    'a'    [2817]    [     0.0618]
    'b'    [ 374]    [     0.0082]
    'c'    [1121]    [     0.0246]
    'd'    [1364]    [     0.0299]
    'e'    [4498]    [     0.0987]
    'f'    [ 761]    [     0.0167]
    'g'    [ 728]    [     0.0160]
    'h'    [1575]    [     0.0345]
    'i'    [2645]    [     0.0580]
    'j'    [  23]    [5.0447e-004]
    'k'    [ 180]    [     0.0039]
    'l'    [1406]    [     0.0308]
    'm'    [ 913]    [     0.0200]
    'n'    [2725]    [     0.0598]
    'o'    [2788]    [     0.0612]
    'p'    [ 709]    [     0.0156]
    'q'    [  55]    [     0.0012]
    'r'    [2388]    [     0.0524]
    's'    [2390]    [     0.0524]
    't'    [3158]    [     0.0693]
    'u'    [ 943]    [     0.0207]
    'v'    [ 384]    [     0.0084]
    'w'    [ 606]    [     0.0133]
    'x'    [  59]    [     0.0013]
    'y'    [ 515]    [     0.0113]
    'z'    [  33]    [7.2381e-004]
    'A'    [ 168]    [     0.0037]
    'B'    [  28]    [6.1414e-004]
    'C'    [  36]    [7.8961e-004]
    'D'    [  28]    [6.1414e-004]
    'E'    [  17]    [3.7287e-004]
    'F'    [  18]    [3.9481e-004]
    'G'    [  16]    [3.5094e-004]
    'H'    [  51]    [     0.0011]
    'I'    [ 139]    [     0.0030]
    'J'    [   7]    [1.5354e-004]
    'K'    [  11]    [2.4127e-004]
    'L'    [   9]    [1.9740e-004]
    'M'    [  27]    [5.9221e-004]
    'N'    [  44]    [9.6508e-004]
    'O'    [  27]    [5.9221e-004]
    'P'    [  39]    [8.5541e-004]
    'Q'    [  10]    [2.1934e-004]
    'R'    [  15]    [3.2901e-004]
    'S'    [ 108]    [     0.0024]
    'T'    [ 101]    [     0.0022]
    'U'    [  35]    [7.6768e-004]
    'V'    [   3]    [6.5801e-005]
    'W'    [  42]    [9.2121e-004]
    'X'    [   1]    [2.1934e-005]
    'Y'    [  19]    [4.1674e-004]
    'Z'    [   0]    [          0]
    ' '    [7561]    [     0.1658]
    '.'    [ 519]    [     0.0114]
    ','    [ 456]    [     0.0100]
    '/'    [   2]    [4.3867e-005]
    '<'    [   0]    [          0]
    '>'    [   0]    [          0]
```

```
    '?'    [   9]    [1.9740e-004]
    ';'    [  10]    [2.1934e-004]
    '''    [  53]    [     0.0012]
    ':'    [  37]    [8.1155e-004]
    '"'    [  83]    [     0.0018]
    '['    [  61]    [     0.0013]
    ']'    [  61]    [     0.0013]
    '{'    [   0]    [          0]
    '}'    [   0]    [          0]
    '\'    [   0]    [          0]
    '|'    [   0]    [          0]
    '!'    [   4]    [8.7735e-005]
    '@'    [   0]    [          0]
    '#'    [   0]    [          0]
    '$'    [   2]    [4.3867e-005]
    '%'    [   0]    [          0]
    '^'    [   0]    [          0]
    '&'    [   0]    [          0]
    '*'    [   2]    [4.3867e-005]
    '('    [   2]    [4.3867e-005]
    ')'    [   2]    [4.3867e-005]
    '-'    [  31]    [6.7994e-004]
    '='    [   0]    [          0]
    '_'    [   0]    [          0]
    '+'    [   0]    [          0]
    '`'    [   0]    [          0]
    '~'    [   0]    [          0]
    '0'    [  72]    [     0.0016]
    '1'    [  35]    [7.6768e-004]
    '2'    [  16]    [3.5094e-004]
    '3'    [   7]    [1.5354e-004]
    '4'    [   7]    [1.5354e-004]
    '5'    [  11]    [2.4127e-004]
    '6'    [   6]    [1.3160e-004]
    '7'    [   2]    [4.3867e-005]
    '8'    [   4]    [8.7735e-005]
    '9'    [  17]    [3.7287e-004]
    '§'    [ 363]    [     0.0080]

End of Step 1 of the project.
...To continue with Step 2, please press "Enter"

Sum of all probabilities : 1

   Symbol, [Occurrence], [Probability] [codeword]         [hex]     [length]

mcell =

    'a'    [2817]    [     0.0618]    '1001'                 '000009'    [ 4]
    'b'    [ 374]    [     0.0082]    '1011000'              '000058'    [ 7]
    'c'    [1121]    [     0.0246]    '00100'                '000004'    [ 5]
    'd'    [1364]    [     0.0299]    '01101'                '00000D'    [ 5]
    'e'    [4498]    [     0.0987]    '000'                  '000000'    [ 3]
    'f'    [ 761]    [     0.0167]    '101101'               '00002D'    [ 6]
    'g'    [ 728]    [     0.0160]    '101011'               '00002B'    [ 6]
    'h'    [1575]    [     0.0345]    '10111'                '000017'    [ 5]
    'i'    [2645]    [     0.0580]    '0101'                 '000005'    [ 4]
    'j'    [  23]    [5.0447e-004]    '10101000110'          '000546'    [11]
    'k'    [ 180]    [     0.0039]    '10101001'             '0000A9'    [ 8]
    'l'    [1406]    [     0.0308]    '10100'                '000014'    [ 5]
    'm'    [ 913]    [     0.0200]    '110101'               '000035'    [ 6]
    'n'    [2725]    [     0.0598]    '0111'                 '000007'    [ 4]
    'o'    [2788]    [     0.0612]    '1000'                 '000008'    [ 4]
    'p'    [ 709]    [     0.0156]    '011001'               '000019'    [ 6]
    'q'    [  55]    [     0.0012]    '1101000111'           '000347'    [10]
    'r'    [2388]    [     0.0524]    '0011'                 '000003'    [ 4]
    's'    [2390]    [     0.0524]    '0100'                 '000004'    [ 4]
    't'    [3158]    [     0.0693]    '1100'                 '00000C'    [ 4]
    'u'    [ 943]    [     0.0207]    '110110'               '000036'    [ 6]
    'v'    [ 384]    [     0.0084]    '1011001'              '000059'    [ 7]
    'w'    [ 606]    [     0.0133]    '001011'               '00000B'    [ 6]
```

```
    'x'    [  59]    [     0.0013]    '1101110001'          '000371'    [10]
    'y'    [ 515]    [     0.0113]    '1101111'             '00006F'    [ 7]
    'z'    [  33]    [7.2381e-004]    '11011101011'         '0006EB'    [11]
    'A'    [ 168]    [     0.0037]    '01100011'            '000063'    [ 8]
    'B'    [  28]    [6.1414e-004]    '11011100000'         '0006E0'    [11]
    'C'    [  36]    [7.8961e-004]    '0110000110'          '000186'    [10]
    'D'    [  28]    [6.1414e-004]    '11011100001'         '0006E1'    [11]
    'E'    [  17]    [3.7287e-004]    '110111011001'        '000DD9'    [12]
    'F'    [  18]    [3.9481e-004]    '110111011111'        '000DDF'    [12]
    'G'    [  16]    [3.5094e-004]    '110111010101'        '000DD5'    [12]
    'H'    [  51]    [     0.0011]    '1101000010'          '000342'    [10]
    'I'    [ 139]    [     0.0030]    '01100000'            '000060'    [ 8]
    'J'    [   7]    [1.5354e-004]    '1010100011111'       '00151F'    [13]
    'K'    [  11]    [2.4127e-004]    '011000101011'        '00062B'    [12]
    'L'    [   9]    [1.9740e-004]    '011000100000'        '000620'    [12]
    'M'    [  27]    [5.9221e-004]    '11010001100'         '00068C'    [11]
    'N'    [  44]    [9.6508e-004]    '1010100010'          '0002A2'    [10]
    'O'    [  27]    [5.9221e-004]    '11010001101'         '00068D'    [11]
    'P'    [  39]    [8.5541e-004]    '0110001001'          '000189'    [10]
    'Q'    [  10]    [2.1934e-004]    '011000101001'        '000629'    [12]
    'R'    [  15]    [3.2901e-004]    '110111001001'        '000DC9'    [12]
    'S'    [ 108]    [     0.0024]    '110100010'           '0001A2'    [ 9]
    'T'    [ 101]    [     0.0022]    '110100000'           '0001A0'    [ 9]
    'U'    [  35]    [7.6768e-004]    '11011101101'         '0006ED'    [11]
    'V'    [   3]    [6.5801e-005]    '01100010100011'      '0018A3'    [14]
    'W'    [  42]    [9.2121e-004]    '0110001011'          '00018B'    [10]
    'X'    [   1]    [2.1934e-005]    '1101110101000001'    '00DD41'    [16]
    'Y'    [  19]    [4.1674e-004]    '01100010001'         '000311'    [11]
    'Z'    [   0]    [          0]           [1x21 char]    '1BA81E'    [21]
    ' '    [7561]    [     0.1658]    '111'                 '000007'    [ 3]
    '.'    [ 519]    [     0.0114]    '001010'              '00000A'    [ 6]
    ','    [ 456]    [     0.0100]    '1101001'             '000069'    [ 7]
    '/'    [   2]    [4.3867e-005]    '110111010100001'     '006EA1'    [15]
    '<'    [   0]    [          0]           [1x21 char]    '1BA81F'    [21]
    '>'    [   0]    [          0]           [1x20 char]    '0DD400'    [20]
    '?'    [   9]    [1.9740e-004]    '011000100001'        '000621'    [12]
    ';'    [  10]    [2.1934e-004]    '011000101010'        '00062A'    [12]
    ''''   [  53]    [     0.0012]    '1101000011'          '000343'    [10]
    ':'    [  37]    [8.1155e-004]    '0110000111'          '000187'    [10]
    '"'    [  83]    [     0.0018]    '101010000'           '000150'    [ 9]
    '['    [  61]    [     0.0013]    '1101110011'          '000373'    [10]
    ']'    [  61]    [     0.0013]    '1101110100'          '000374'    [10]
    '{'    [   0]    [          0]           [1x20 char]    '0DD401'    [20]
    '}'    [   0]    [          0]           [1x20 char]    '0DD402'    [20]
    '\'    [   0]    [          0]           [1x20 char]    '0DD403'    [20]
    '|'    [   0]    [          0]           [1x20 char]    '0DD404'    [20]
    '!'    [   4]    [8.7735e-005]    '11011101010001'      '003751'    [14]
    '@'    [   0]    [          0]           [1x20 char]    '0DD405'    [20]
    '#'    [   0]    [          0]           [1x20 char]    '0DD406'    [20]
    '$'    [   2]    [4.3867e-005]    '110111010100110'     '006EA6'    [15]
    '%'    [   0]    [          0]           [1x20 char]    '0DD407'    [20]
    '^'    [   0]    [          0]           [1x20 char]    '0DD408'    [20]
    '&'    [   0]    [          0]           [1x20 char]    '0DD409'    [20]
    '*'    [   2]    [4.3867e-005]    '110111010100111'     '006EA7'    [15]
    '('    [   2]    [4.3867e-005]    '01100010100000'      '0018A0'    [14]
    ')'    [   2]    [4.3867e-005]    '01100010100001'      '0018A1'    [14]
    '-'    [  31]    [6.7994e-004]    '11011100101'         '0006E5'    [11]
    '='    [   0]    [          0]           [1x20 char]    '0DD40A'    [20]
    '_'    [   0]    [          0]           [1x20 char]    '0DD40B'    [20]
    '+'    [   0]    [          0]           [1x20 char]    '0DD40C'    [20]
    '`'    [   0]    [          0]           [1x20 char]    '0DD40D'    [20]
    '~'    [   0]    [          0]           [1x20 char]    '0DD40E'    [20]
    '0'    [  72]    [     0.0016]    '011000010'           '0000C2'    [ 9]
    '1'    [  35]    [7.6768e-004]    '11011101110'         '0006EE'    [11]
    '2'    [  16]    [3.5094e-004]    '110111011000'        '000DD8'    [12]
    '3'    [   7]    [1.5354e-004]    '1101110010000'       '001B90'    [13]
    '4'    [   7]    [1.5354e-004]    '1101110010001'       '001B91'    [13]
    '5'    [  11]    [2.4127e-004]    '101010001110'        '000A8E'    [12]
    '6'    [   6]    [1.3160e-004]    '1010100011110'       '00151E'    [13]
    '7'    [   2]    [4.3867e-005]    '01100010100010'      '0018A2'    [14]
    '8'    [   4]    [8.7735e-005]    '11011101010010'      '003752'    [14]
```

```
    '9'    [  17]    [3.7287e-004]    '110111011110'        '000DDE'    [12]
    '§'    [ 363]    [     0.0080]    '1010101'             '000055'    [ 7]

Bit rate or Average Length : 4.4563
Max codeword Length : 21
Characters read: 45592
END OF MAIN %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

End Of Project