作业四

李宁　2016050201017

3、编程练习：读取一幅 512×512×8 比特的单色 Lena 图像，完成以下步骤：

1）、统计该图像的概率直方图，并画出直方图；

2）、计算该图像的熵；

3）、对其进行霍夫曼编码；

4）、分别计算压缩率和冗余度。

代码：

```
clc,clear;
close all;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
f=imread('lena512.jpg');
[height,width]=size(f);
a=min(min(f));%The minimum gray value of image.
b=max(max(f));%The maximum gray value of image.
x=unique(f);
y=zeros(size(x));
for i=1:length(x)%The frequency of the corresponding gray value.
    y(i)=length(find(f==x(i)));
end
frequ=y./(height*width);
%sum(frequ)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure,subplot(1,2,1),imshow(f),title('原始图像')%Original image.
subplot(1,2,2),bar(x,frequ),title('频率直方图')%Frequency histogram.
Hs=-(sum(frequ.*log2(frequ)));%Entropy.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fre_sort=sort(frequ,'descend');
[m,n]=size(x);
x_sort=zeros(m,n);
len_t=1;
for i=1:m
    if len_t==1
        len_t=length(find(frequ==fre_sort(i)));
        x_sort(i:i+len_t-1)=x(frequ==fre_sort(i));
    else
        len_t=len_t-1;
        continue;
    end
end
```

```
B=zeros(m,m);%Encoder matrix.
for i=1:m
    B(:,i)=fre_sort;%Initialize the first column of encoder matrix.
end
temp_fre_sort=fre_sort;
for i=2:m%Encoder matrix
    if i==2
        temp_s=B(end-1,i-1)+B(end,i-1);
        B(end-1,i)=temp_s;
        B(end,i:end)=0;
        temp_fre_sort=sort(B(:,i),'descend');
        B(:,i)=temp_fre_sort;
        B(:,i+1)=temp_fre_sort;
        temp_addr=find(temp_fre_sort==temp_s,m);
        B(end,i)=temp_addr(end);
    else
        temp_s=B(end-i+1,i-1)+B(end-i+2,i-1);
        B(end-i+1,i)=temp_s;
        B(end-i+2,i:end)=0;
        temp_fre_sort=sort([B(1:end-1,i);0],'descend');
        if i==m
            B(:,i)=temp_fre_sort;
        else
            B(:,i)=temp_fre_sort;
            B(:,i+1)=temp_fre_sort;
        end
        temp_addr=find(temp_fre_sort==temp_s,m);
        B(end,i)=temp_addr(end);
    end
end
H_code=char(zeros(m,m));%The initial coding.
for i=m:-1:2%Produce Hoffman coding.
    if i==m
        H_code(1,1)='0';
        H_code(2,1)='1';
    else
        temp_H_code=H_code;
        K=B(end,i);
        for
j=1:m-i%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            if
j>1&&B(j,i-1)==B(j-1,i-1)%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                continue;
            else
```

```
                B(K,i)=-1;
                temp_fre=B(:,i);
                temp_addr=find(temp_fre==B(j,i-1));
                if length(temp_addr)==1
                    H_code(j,:)=temp_H_code(temp_addr,:);
                else
                    temp_L=length(temp_addr);
                    H_code(j:j+temp_L-1,:)=temp_H_code(temp_addr,:);
                end
            end
        end
        code_length=length(find(temp_H_code(K,:)=='1'))+...
            length(find(temp_H_code(K,:)=='0'));
        H_code(m-i+1,1:code_length+1)...
            =[temp_H_code(K,1:code_length),'0'];
        H_code(m-i+2,1:code_length+1)...
            =[temp_H_code(K,1:code_length),'1'];
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
len_average=0;
for i=1:m
    len_subcode=length(find(H_code(i,:)=='1'))+...
        length(find(H_code(i,:)=='0'));
    len_average=len_average+len_subcode*fre_sort(i);
end
[num2str(x_sort),char(ones(m,n).*32),char(ones(m,n).*32),H_code]
Hs
len_average
y=Hs/len_average
C=8/len_average
R=1-y
```

结果：

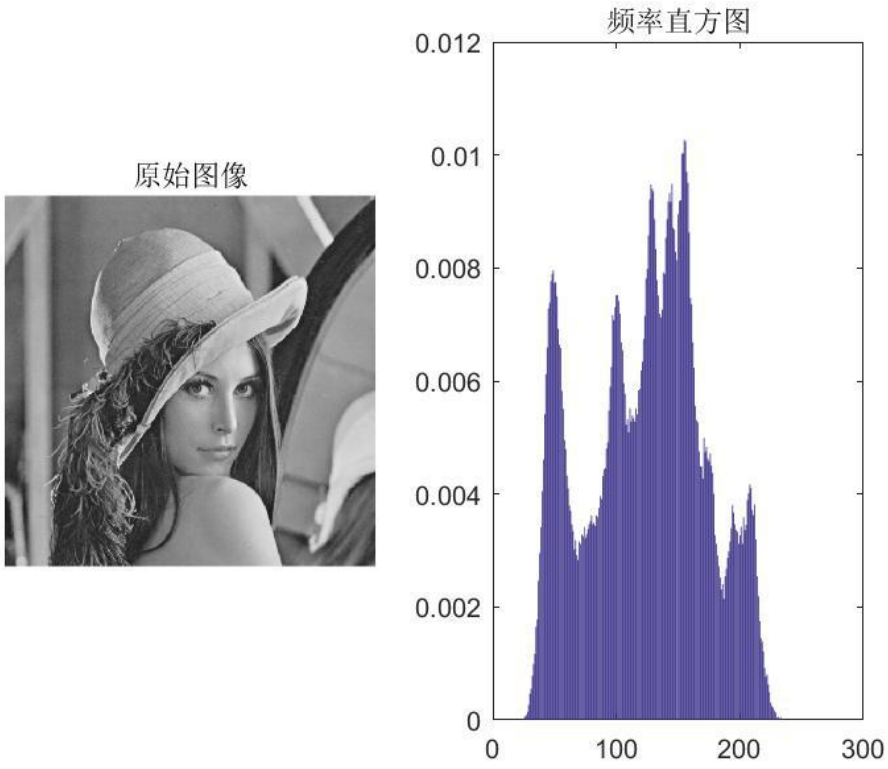

图 1

熵 Hs：7.4471
编码效率 y：0.9969
压缩比 C：1.0709
冗余度 R：0.0031
霍夫曼编码结果：

155	0000110
156	0000111
154	0001010
153	0001011
157	0001101
158	0010000
145	0010010
128	0010011
129	0010100
130	0010110
143	0010111
127	0011001
152	0011010
151	0011011
142	0011100
144	0011101
150	0100000
131	0100010

141	0100011
146	0100100
147	0100110
159	0101000
126	0101001
140	0101010
132	0101100
148	0101101
149	0101111
125	0110001
49	0110010
138	0110100
48	0110101
139	0110110
124	0111000
133	0111001
47	0111010
51	0111011
50	0111100
101	0111111
100	1000000
134	1000001
52	1000010
160	1000100
102	1000101
46	1000110
161	1001000
99	1001001
45	1001010
137	1001011
103	1001110
135	1001111
97	1010000
123	1010001
136	1010010
98	1010100
53	1010111
122	1011100
104	1011111
162	1100000
54	1100001
44	1100011
55	1100100
105	1100110

121	1101010
96	1101100
163	1101110
120	1110000
43	1110001
106	1110011
95	1110101
119	1110110
118	1110111
56	1111001
107	1111010
164	1111100
111	1111101
116	1111110
57	1111111
94	00000000
42	00000001
117	00000010
113	00000100
114	00000101
109	00000110
165	00000111
112	00001000
115	00001001
166	00001010
108	00001011
110	00010001
171	00010011
58	00011000
93	00011001
173	00011101
92	00011110
59	00011111
167	00100010
172	00100011
176	00101010
174	00110000
41	00110001
175	00111100
177	00111101
168	00111110
169	00111111
91	01000010
90	01000011

178	01001010
60	01001110
89	01001111
170	01010110
208	01010111
209	01011100
61	01011101
40	01100000
207	01100001
179	01100110
87	01100111
88	01101111
206	01111010
212	01111011
194	01111100
62	01111101
86	10000110
210	10000111
63	10001110
195	10001111
211	10011000
80	10011001
84	10011010
85	10011011
203	10100110
79	10101010
82	10101011
81	10101100
64	10101101
196	10110000
78	10110001
205	10110010
83	10110011
201	10110100
74	10110101
39	10110110
193	10110111
77	10111010
197	10111011
204	10111100
65	10111101
198	11000100
72	11000101
76	11001010

180	11001011
200	11001110
73	11010000
75	11010001
213	11010010
67	11010011
192	11010110
70	11010111
181	11011010
199	11011011
202	11011110
71	11011111
66	11100100
191	11101000
38	11101001
68	11110000
182	11110001
190	11110110
69	11110111
183	000000110
189	000000111
188	000100001
214	000100100
184	000100101
37	000111000
186	000111001
185	001010111
215	010010110
187	010010111
36	101001110
216	101001111
35	110011110
217	111001011
218	0001000000
219	0001000001
34	0010101101
33	0110111001
220	0110111011
222	1100111111
32	1110010100
221	1110010101
223	00101011000
31	01101110000
224	01101110100

30	01101110101
225	001010110010
29	001010110011
226	011011100011
227	110011111001
228	110011111011
28	0110111000100
229	0110111000101
27	1100111110101
26	11001111100001
231	11001111100011
233	11001111101000
230	11001111101001
25	110011111000001
232	110011111000100
234	1100111110000000
24	1100111110001011
23	11001111100000011
22	110011111000000101
236	1100111110000001000
239	1100111110000001001
240	1100111110001010101
242	110011111000101011
245	110011111000101000
248	110011111000101001