

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN – ĐIỆN TỬ
BỘ MÔN ĐIỆN TỬ

-----o0o-----



BÁO CÁO BÀI TẬP LỚN
XÁC SUẤT THỐNG KÊ
ĐỀ TÀI: MÃ HOÁ SHANNON – FANO

GVHD: ĐẶNG NGUYỄN CHÂU
NHÓM 2
THÀNH VIÊN:

STT	Họ và Tên	MSSV
1	Trần Ngọc Đan Thanh	1912039
2	Huỳnh Ngọc Thanh Thảo	1912069
3	Phí Tuệ Thư	1912179
4	Ngô Đức Thuận	1912157
5	Văn Thành Thuận	1912162
6	Võ Duy Thuận (nhóm trưởng)	1912163

TP. HỒ CHÍ MINH, NGÀY 06 THÁNG 12 NĂM 2020

Bảng chấm điểm chéo hoạt động nhóm 2:

<i>Thành viên</i>	Trần Ngọc Đan Thanh	Huỳnh Ngọc Thanh Thảo	Phí Tuệ Thư	Ngô Đức Thuận	Văn Thành Thuận	Võ Duy Thuận
Trần Ngọc Đan Thanh	9	9	9	9	9	10
Huỳnh Ngọc Thanh Thảo	9	9	8	9	8	10
Phí Tuệ Thư	9	9	8	9	8	10
Ngô Đức Thuận	9	9	7	9	8	10
Văn Thành Thuận	9	9	9	8	7	10
Võ Duy Thuận	9	9	9	8	8	10

Bảng mô tả hoạt động làm việc nhóm 2:
(hiệu quả đóng góp được đánh giá theo mức 1-4)

<i>Thành viên</i>	Nghiên cứu đề tài	Lý thuyết	Code	Hiệu suất	Báo cáo	Power
Trần Ngọc Đan Thanh	4	4	1	1	1	1
Huỳnh Ngọc Thanh Thảo	4	1	1	4	4	1
Phí Tuệ Thu	4	2	1	1	3	1
Ngô Đức Thuận	4	1	2	4	1	4
Văn Thành Thuận	4	1	2	1	1	4
Võ Duy Thuận	4	1	4	1	1	1

Mục lục

I.CƠ SỞ LÝ THUYẾT:	5
1. Mã hóa Shannon – Fano là gì?	5
2. Các bước mã hóa Shannon – Fano:	6
a. Mã hóa Shanon:	6
b. Mã hóa Fano:	6
3. Nhận xét:	7
II.CHƯƠNG TRÌNH DÙNG ĐỂ ĐỌC VÀ HIỂN THỊ ẢNH LENA-GRAY.....	8
III.MÃ HÓA ẢNH LENA DỰA THEO PHƯƠNG PHÁP SHANNON-FANO	9
IV.HIỆU SUẤT MÃ HÓA	18
Phương pháp Shannon:	18
Phương pháp Fano:	19

I. CƠ SỞ LÝ THUYẾT:

1. Mã hóa Shannon – Fano là gì?

- **Nén dữ liệu**, hay còn được gọi là mã hóa nguồn, là quá trình mã hóa hay chuyển hóa dữ liệu sao cho không gian bộ nhớ được sử dụng là ít nhất có thể. Nén dữ liệu giúp giảm lượng tài nguyên cần thiết để lưu trữ và chuyển hóa dữ liệu.
- Trong lĩnh vực nén dữ liệu, **mã hóa Shannon – Fano**, được đặt tên theo hai nhà khoa học là Claude Shannon và Robert Fano, tuy là hai nhà khoa học độc lập nhưng thực chất của hai phương pháp Shannon và Fano chỉ là một, là đều mã hóa dữ liệu dựa trên tập hợp các ký hiệu và xác suất xuất hiện của chúng:
 - Phương pháp của Shannon: chọn mã tiền tố trong đó ký hiệu nguồn i được cung cấp độ dài từ mã $l_i = \lceil -\log_2 p_i \rceil$. Một cách phổ biến để chọn các từ mã là sử dụng sự mở rộng nhị phân của các xác suất tích lũy. Phương pháp này được đề xuất trong bài báo “Lý thuyết toán học trong truyền thông” của Shannon (1948), giới thiệu lĩnh vực lý thuyết thông tin.
 - Phương pháp của Fano: chia các ký hiệu nguồn thành hai tập hợp “0” và “1” (gán giá trị “0” cho tập hợp thứ nhất và “1” cho tập hợp thứ hai) với xác suất của mỗi tập hợp gần bằng nhau nhất có thể. Sau đó, mỗi bộ đó lại tiếp tục được chia làm hai, một cách tương tự và tiếp tục cho đến khi mỗi bộ chỉ chứa một ký hiệu. Phương pháp này được đề xuất trong một báo cáo kỹ thuật sau đó của Fano (1949).
 - Mã Shannon Fano là mã hóa không tối ưu vì nó không luôn đạt được độ dài từ mã được mong đợi thấp nhất có thể như mã hóa Huffman, nên hầu như không được sử dụng trong thực tế. Tuy nhiên, mã Shannon Fano có độ dài từ mã dự kiến trong khoảng tối ưu là 1 bit. Phương pháp của Fano thường tạo ra từ mã có độ dài dự kiến ngắn hơn phương pháp của Shannon nhưng phương pháp của Shannon dễ phân tích hơn về mặt lý thuyết.
- Thuật toán Shannon Fano là một kỹ thuật mã hóa entropy (entropy là khái niệm trong lý thuyết thông tin do Shannon đưa ra vào năm 1948, là đại lượng đo thông tin hay còn gọi là độ bất định, nó được tính như một hàm phân bố xác suất) để nén dữ liệu mà không mất dữ liệu của đa phương tiện, nó gán mã cho mỗi ký hiệu dựa trên xác suất xuất hiện của chúng. Đây là một lược đồ mã hóa có độ dài thay đổi, tức là các mã được gán cho các ký hiệu khác nhau thì có độ dài khác nhau, các ký hiệu có tần suất xuất hiện nhiều sẽ có độ dài bit nhỏ và ngược lại.

2. Các bước mã hóa Shannon – Fano:

a. Mã hóa Shanon:

- Bước 1: Tạo danh sách xác suất (hoặc tần suất) các màu xuất hiện trong mỗi ô nhớ.
- Bước 2: Sắp xếp danh sách theo thứ tự xác suất giảm dần. Giả sử $p_1 > p_2 > p_3 > \dots > p_n$.
- Bước 3: Tính độ dài bit của từng xác suất $l_i = -\log_2 p_i$.
- Bước 4: Tính xác suất tích lũy $c_i = \sum_{j=1}^{i-1} p_j$ với $i \geq 2$.
- Bước 5: Đổi c_i sang hệ nhị phân và lấy l_i số sau dấu phẩy.

Ví dụ: Mã hóa nguồn $S=\{a_1, a_2, a_3, a_4, a_5\}$ với các xác suất lần lượt là 0,385; 0,179; 0,154; 0,154; 0,128.

a_i	p_i	l_i	c_i	Nhị phân	Từ mã
a_1	0,385	2	0	0,00000	00
a_2	0,179	3	0,385	0,01100	011
a_3	0,154	3	0,564	0,10010	100
a_4	0,154	3	0,718	0,10110	101
a_5	0,128	3	0,872	0,11011	110

b. Mã hóa Fano:

- Bước 1: Tạo danh sách xác suất hoặc tần suất.
- Bước 2: Sắp xếp danh sách theo thứ tự xác suất giảm dần. Giả sử $p_1 > p_2 > p_3 > \dots > p_n$.
- Bước 3: Phân các xác suất thành hai nhóm có tổng gần bằng nhau.
- Bước 4: Gán cho các nhóm ký hiệu 0 và 1.
- Bước 5: Lặp lại bước 3 cho các nhóm con đến khi kết thúc.
- Bước 6: Từ mã ứng với mỗi xác suất là chuỗi bao gồm các ký hiệu theo thứ tự được gán.

Ví dụ: Mã hóa nguồn $S=\{a_1, a_2, a_3, a_4, a_5\}$ với các xác suất lần lượt là 0,385; 0,179; 0,154; 0,154; 0,128.

a_i	p_i	Lần 1	Lần 2	Lần 3	Từ mã
a_1	0,385	0	0		00
a_2	0,179	0	1		01
a_3	0,154	1	0		10
a_4	0,154	1	1	0	110
a_5	0,128	1	1	1	111

3. Nhận xét:

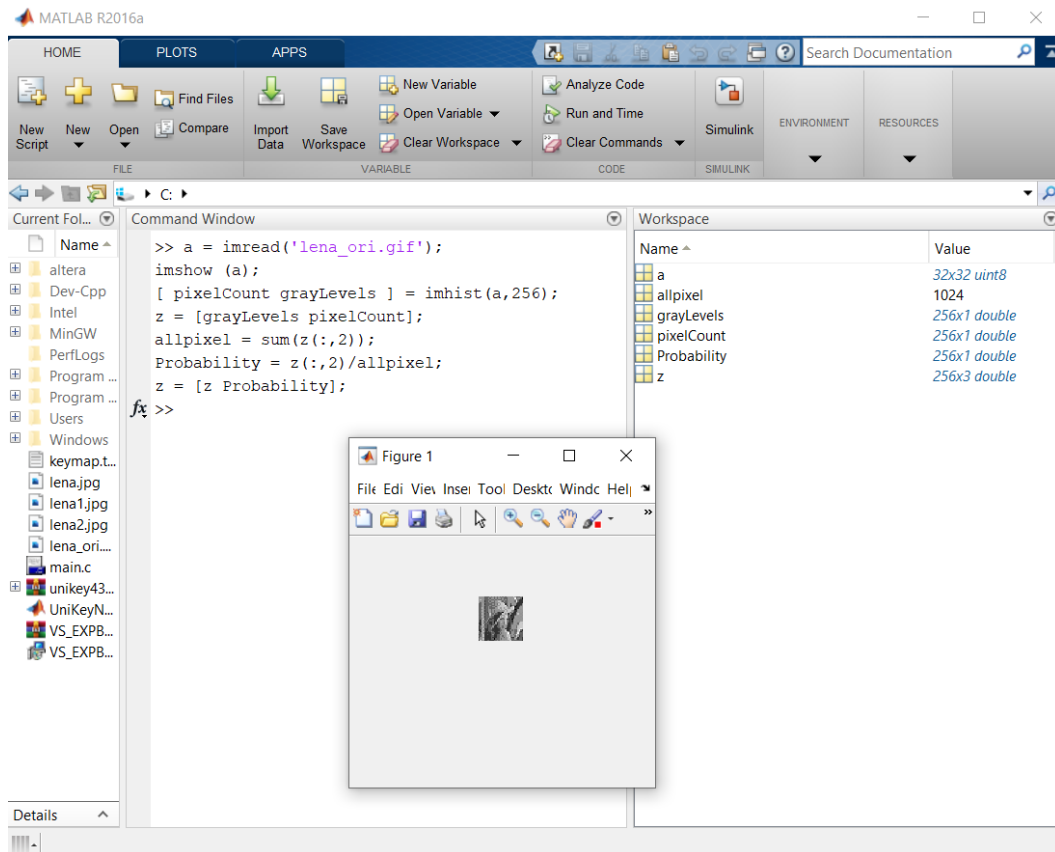
- Hai phương pháp Shannon – Fano thực chất là một, đều xây dựng trên cùng một cơ sở độ dài từ mã tỉ lệ nghịch với xác suất xuất hiện, không cho phép lập mã một cách duy nhất vì sự chia nhóm trên cơ sở đồng đều và tổng xác suất nên có thể có nhiều cách chia.
- Bên cạnh đó, hai phương pháp trên có thể khác nhau về độ dài bit sau khi mã hóa. Cụ thể, phương pháp Fano sẽ cho ra kết quả mã hóa với chiều dài bit ngắn hơn phương pháp Shannon.

II. CHƯƠNG TRÌNH DÙNG ĐỂ ĐỌC VÀ HIỂN THỊ ẢNH LENA-GRAY

- Dưới đây là chương trình được viết trên Matlab để đọc và hiển thị ảnh Lenagray 32*32 pixels

```
a = imread('lena_ori.gif');  
imshow (a);  
[ pixelCount grayLevels ] = imhist(a,256);  
z = [grayLevels pixelCount];  
allpixel = sum(z(:,2)) ;  
Probability = z(:,2)/allpixel;  
z = [z Probability];
```

- Ta nhận được kết quả xuất ra màn hình như sau:

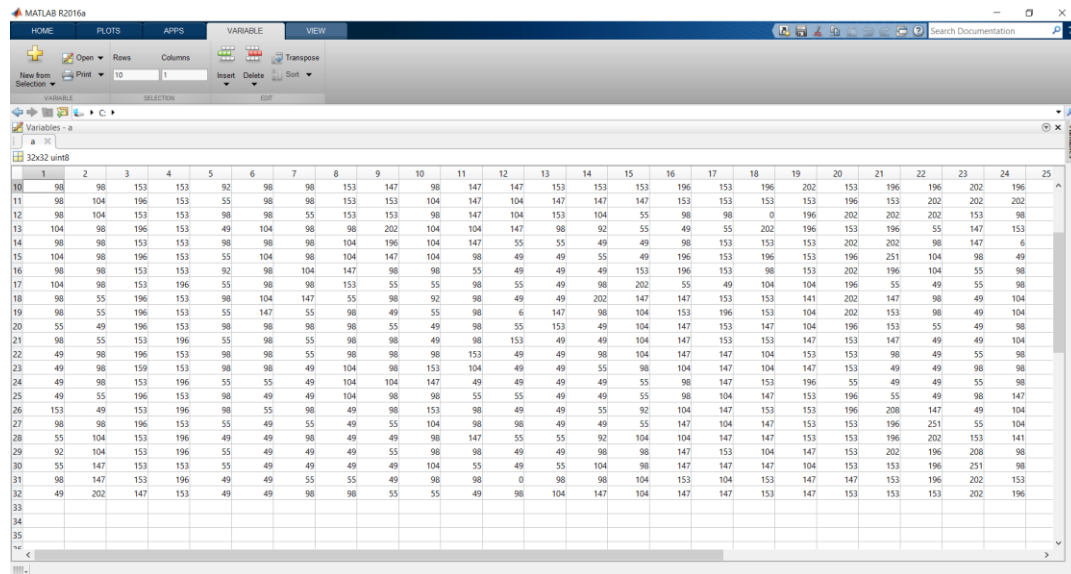


Hình 1: Kết quả xuất ra màn hình

III. MÃ HÓA ẢNH LENA DỰA THEO PHƯƠNG PHÁP SHANNON-FANO

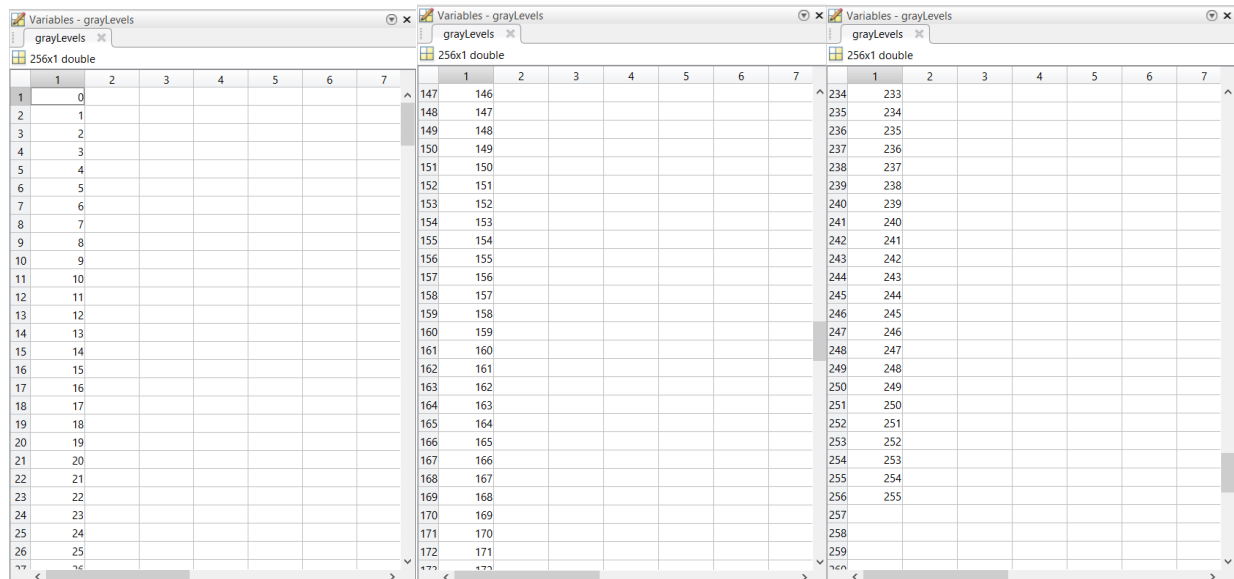
Đầu tiên, ta cần một số dữ liệu thống kê về tần suất xuất hiện của các bit màu trong ảnh Lena. Chúng ta có thể tiếp tục sử dụng kết quả của đoạn chương trình trên. Như đã trình bày ở Hình 1, ta có ‘Bảng giá trị’ bên phải màn hình. Từng giá trị có ý nghĩa như sau:

- Với giá trị ‘a’, ta nhận được bảng hiển thị các bit màu của ảnh Lena 32*32 pixels.



The image shows the MATLAB R2016a interface with the 'Variables' window displaying a 32x32 matrix of grayscale values. The matrix is titled 'a' and is of type 'uint8'. The values range from 0 to 255, representing the intensity of each pixel in the Lena image.

- Với giá trị ‘allpixel’, ta nhận giá trị 1024, tức là tổng số pixel có trong ảnh Lena.
- Với giá trị ‘grayLevels’, ta nhận các giá trị biểu diễn cho các mức xám (từ 0 đến 255).



The image shows three MATLAB R2016a windows, each displaying a 256x1 vector of grayscale values. The first window is titled 'grayLevels' and shows values from 0 to 255. The second window is titled 'grayLevels' and shows values from 147 to 172. The third window is titled 'grayLevels' and shows values from 234 to 255. These values represent the frequency of each grayscale level in the Lena image.

- Với giá trị 'pixelCount', ta nhận các giá trị là số lần xuất hiện của từng mức xám trong ảnh xám (ví dụ: mức xám số 0 xuất hiện 5 lần trong ảnh xám).

pixelCount		
256x1 double		
	1	
1	5	
2	0	
3	0	
4	0	
5	0	
6	0	
7	5	
8	0	

pixelCount		
256x1 double		
	1	
43	0	
44	2	
45	0	
46	0	
47	0	
48	0	
49	0	
50	118	
51	0	
52	0	
53	0	
54	0	
55	0	
56	82	
57	0	

pixelCount		
256x1 double		
	1	
92	0	
93	7	
94	0	
95	0	
96	0	
97	0	
98	0	
99	180	
100	0	
101	0	
102	0	
103	0	
104	0	
105	125	

- Với giá trị 'Probability', ta có xác suất xuất hiện của từng mức xám trong ảnh xám.

Probability		
256x1 double		
	1	
1	0.0049	
2	0	
3	0	
4	0	
5	0	
6	0	
7	0.0049	
8	0	

Probability		
256x1 double		
	1	2
44	0.0020	
45	0	
46	0	
47	0	
48	0	
49	0	
50	0.1152	
51	0	
52	0	
53	0	
54	0	
55	0	
56	0.0801	

Probability		
256x1 double		
	1	2
92	0	
93	0.0068	
94	0	
95	0	
96	0	
97	0	
98	0	
99	0.1758	
100	0	
101	0	
102	0	
103	0	
104	0	
105	0.1221	

- Với giá trị 'z', ta có bảng mức xám, tần số và cả xác suất xuất hiện của các mức xám được xuất ra màn hình như hình minh họa bên dưới:

z				z			
256x3 double				256x3 double			
	1	2	3		1	2	3
1	0	5	0.0049	233	232	0	0
2	1	0	0	234	233	0	0
3	2	0	0	235	234	0	0
4	3	0	0	236	235	0	0
5	4	0	0	237	236	0	0
6	5	0	0	238	237	0	0
7	6	5	0.0049	239	238	0	0
8	7	0	0	240	239	0	0
9	8	0	0	241	240	0	0
10	9	0	0	242	241	0	0
11	10	0	0	243	242	0	0
12	11	0	0	244	243	0	0
13	12	0	0	245	244	0	0
14	13	0	0	246	245	5	0.0049
15	14	0	0	247	246	0	0
16	15	0	0	248	247	0	0
17	16	0	0	249	248	0	0
18	17	0	0	250	249	0	0
19	18	0	0	251	250	0	0
20	19	0	0	252	251	8	0.0078
21	20	0	0	253	252	0	0
22	21	0	0	254	253	0	0
23	22	0	0	255	254	0	0
24	23	0	0	256	255	0	0
25	24	0	0	257			
26	25	0	0				

❖ Chương trình mã hóa ảnh Lena theo tần suất xuất hiện (viết dựa trên phương pháp Fano)

```
1  #include<stdio.h>
2
3  struct node {
4      int color;
5      float pro;
6      int arr[100];
7      int top;
8  } p[256];
9      float size,H;
10
11 void Fano_Function(int l, int h,node p[]){
12     float pack1 = 0, pack2 = 0, diff1 = 0, diff2 = 0;
13     int i,j;
14     if((l+1)==h || l>=h){
15         if(l>=h){
16             return;
17         }
18         p[l].arr[++(p[l].top)]=0;
19         p[h].arr[++(p[h].top)]=1;
20         return;
21     }
22     else{
23         pack1=p[l].pro;
24         for(i=l+1;i<=h;i++){
25             pack2+=p[i].pro;
26         }
27         diff1= pack1 -pack2;
28         if (diff1<0) {diff1=-diff1;}
29
30         j=l+1;
31         while(j<=h-1){
32             pack1 = pack2 = 0;
33             for (i=l;i<=j;i++){
34                 pack1+=p[i].pro;
35             }
36             for (i=j+1;i<=h;i++){
37                 pack2+=p[i].pro;
38             }
```

```
37         pack2+=p[i].pro;
38     }
39     diff2=pack1- pack2;
40     if(diff2<0){diff2=-diff2;}
41
42     if(diff2>=diff1){
43         break;
44     }
45     diff1=diff2;
46     j++;
47 }
48
49 for (int a=1;a<=j-1;a++){
50     p[a].arr[++(p[a].top)]=0;
51 }
52 for (int a=j;a<=h;a++){
53     p[a].arr[++(p[a].top)]=1;
54 }
55 Fano_Function(1,j-1,p);
56 Fano_Function(j,h,p);
57 }
58 }
59
60 void DecreasebyProbability(int n, node p[]){
61     int i,j;
62     node temp;
63     for (i=0;i<=n-2;i++){
64         for(j=i+1;j<=n-1;j++){
65             if(p[j].pro>=p[i].pro){
66                 temp = p[i];
67                 p[i] = p[j];
68                 p[j] = temp;
69             }
70         }
71     }
72 }
73
```

```

73
74 void display_binary(int n,node p[]){
75     int i,j;
76     printf("\n\t\tColor\tProbability\tBit-Length\tBinary");
77     for (i=0;i<=n-1;i++){
78         if(p[i].pro>0){
79             printf("\n\t\t %d \t %f \t   %d \t\t",p[i].color,p[i].pro,p[i].top+1);
80             for (j=0;j<=p[i].top;j++){
81                 printf("%d",p[i].arr[j]);
82             }
83         }
84     }
85 }
86
87 void Func_after_compression(int n, node p[]){
88     int length[n-1];
89     for(int i=0;i<=n-1;i++){
90         length[i]=p[i].top+1;
91     }
92
93     for(int i=0;i<=n-1;i++){
94         size+=p[i].pro*length[i]*32*32;
95     }
96
97     H=size/(32*32*8);
98
99     printf("\n\n\tHieu suat ma hoa = (%.2f/%d)*100% = %f%c",size,32*32*8,H*100,'%');
100 }
101
102 int main(){
103     int n, i, j;
104     float tong = 0;
105
106     printf("Nhap so mau:");
107     scanf("%d",&n);
108
109     printf("Input the intensity of grayscale\n");
110     for (int i=0;i<=n-1;i++){

```

```
95     }
96
97     H=size/(32*32*8);
98
99     printf("\n\n\tHieu suat ma hoa = (%.2f/%d)*100% = %f%c",size,32*32*8,H*100,'%');
100 }
101
102 int main(){
103     int n, i, j;
104     float tong = 0;
105
106     printf("Nhap so mau:");
107     scanf("%d",&n);
108
109     printf("Input the intensity of grayscale\n");
110     for (int i=0;i<=n-1;i++){
111         scanf("%d",&p[i].color);
112     }
113
114     for(int i=0;i<=n-1;i++){
115         printf("Tan suat cua mau %d la: ",p[i].color);
116         scanf("%f",&p[i].pro);
117         tong+=p[i].pro;
118         if(tong>1){
119             tong-=p[i].pro;
120             printf("nhap gia tri sai\tTong xac suat = %f\n",tong);
121             i--;
122         }
123     }
124     for (i = 0; i < n; i++)
125         p[i].top = -1;
126
127     DecreasebyProbability(n,p);
128     Fano_Function(0,n-1,p);
129     display_binary(n,p);
130     Func_after_compression(n,p);
131     return 0;
132 }
```

✓ **Kết quả minh họa:**

```
C:\Users\PC\Documents\DevC++data\BTL2_xstk.exe
Nhap so mau:18
Input the intensity of grayscale
0
6
43
49
55
92
98
104
110
141
147
153
159
196
202
208
245
251
Tan suat cua mau 0 la: 0.004882813
Tan suat cua mau 6 la: 0.004882813
Tan suat cua mau 43 la: 0.001953125
Tan suat cua mau 49 la: 0.115234375
Tan suat cua mau 55 la: 0.080078125
Tan suat cua mau 92 la: 0.006835938
Tan suat cua mau 98 la: 0.17578125
Tan suat cua mau 104 la: 0.122070313
Tan suat cua mau 110 la: 0.000976563
Tan suat cua mau 141 la: 0.001953125
Tan suat cua mau 147 la: 0.1484375
Tan suat cua mau 153 la: 0.204101563
Tan suat cua mau 159 la: 0.000976563
Tan suat cua mau 196 la: 0.05859375
Tan suat cua mau 202 la: 0.057617188
Tan suat cua mau 208 la: 0.002929688
Tan suat cua mau 245 la: 0.004882813
Tan suat cua mau 251 la: 0.0078125

      Color    Probability    Bit-Length    Binary
      153      0.204102        2            00
      98       0.175781        3            010
      147      0.148438        3            011
```



```

245
251
Tan suat cua mau 0 la: 0.004882813
Tan suat cua mau 6 la: 0.004882813
Tan suat cua mau 43 la: 0.001953125
Tan suat cua mau 49 la: 0.115234375
Tan suat cua mau 55 la: 0.080078125
Tan suat cua mau 92 la: 0.006835938
Tan suat cua mau 98 la: 0.17578125
Tan suat cua mau 104 la: 0.122070313
Tan suat cua mau 110 la: 0.000976563
Tan suat cua mau 141 la: 0.001953125
Tan suat cua mau 147 la: 0.1484375
Tan suat cua mau 153 la: 0.204101563
Tan suat cua mau 159 la: 0.000976563
Tan suat cua mau 196 la: 0.05859375
Tan suat cua mau 202 la: 0.057617188
Tan suat cua mau 208 la: 0.002929688
Tan suat cua mau 245 la: 0.004882813
Tan suat cua mau 251 la: 0.0078125

      Color  Probability  Bit-Length  Binary
      153    0.204102      2           00
      98     0.175781      3           010
     147     0.148438      3           011
     104     0.122070      3           100
      49     0.115234      3           101
      55     0.080078      4           1100
     196     0.058594      4           1101
     202     0.057617      4           1110
     251     0.007813      6           111100
      92     0.006836      7           1111010
     245     0.004883      7           1111011
       6     0.004883      7           1111100
       0     0.004883      7           1111101
     208     0.002930      8           11111100
     141     0.001953      8           11111101
      43     0.001953      8           11111110
     159     0.000977      9           111111110
     110     0.000977      9           111111111

Hieu suat ma hoa = (3223.00/8192)*100= 39.343262%
-----

```

IV. HIỆU SUẤT MÃ HÓA

- Dung lượng ảnh ban đầu = $32 \times 32 \times 8 = 8192$ (bit)

Phương pháp Shannon:

Màu	Số lần xuất hiện	Xác suất	Binary	Độ dài bit
153	209	0.204101563	000	3
98	180	0.17578125	010	3
147	152	0.1484375	011	3
104	125	0.122070313	1000	4
49	118	0.115234375	1010	4
55	82	0.080078125	1100	4
196	60	0.05859375	11011	5
202	59	0.057617188	11100	5
251	8	0.0078125	1111011	7
92	7	0.006835938	11111000	8
0	5	0.004882813	11111010	8
6	5	0.004882813	11111011	8
245	5	0.004882813	11111100	8
208	3	0.002929688	111111011	9
43	2	0.001953125	111111101	9
141	2	0.001953125	111111110	9
110	1	0.000976563	1111111101	10
159	1	0.000976563	1111111110	10

Độ dài trung bình của từ mã: $\sum P_i \times n$

(với P_i là xác suất xuất hiện của màu i , n là số bit mã hóa)

$$0.204101563 \times 3 + 0.17578125 \times 3 + 0.1484375 \times 3 + 0.122070313 \times 4 + 0.115234375 \times 4 + 0.080078125 \times 4 + 0.05859375 \times 5 + 0.057617188 \times 5 + 0.0078125 \times 7 + 0.006835938 \times 8 + 0.004882813 \times 8 + 0.004882813 \times 8 + 0.004882813 \times 8 + 0.002929688 \times 9 + 0.001953125 \times 9 + 0.001953125 \times 9 + 0.000976563 \times 10 + 0.000976563 \times 10 = 3.743164099 \text{ (bit)}$$

Dung lượng ảnh lúc sau = $3.743164099 \times 32 \times 32 = 3833$ (bit)

$$\text{Hiệu suất mã hóa} = \frac{\text{Dung lượng ảnh lúc sau}}{\text{Dung lượng ảnh ban đầu}} \times 100\% = 46.76\%$$

Phương pháp Fano:

	A	B	C	D	E	F	G	H	I	J	K	L
1	Color	Occurences	Xác suất xuất hiện	xấp xỉ 0.5	xấp xỉ 0.25	xấp xỉ 0.125	xấp xỉ 0.0625 ...					
2	153	209	0.204101563	0	0							
3	98	180	0.17578125	0	1	0						
4	147	152	0.1484375	0	1	1						
5	104	125	0.122070313	1	0	0						
6	49	118	0.115234375	1	0	1						
7	55	82	0.080078125	1	1	0	0					
8	196	60	0.05859375	1	1	0	1					
9	202	59	0.057617188	1	1	1	0					
10	251	8	0.0078125	1	1	1	1	0	0			
11	92	7	0.006835938	1	1	1	1	0	1	0		
12	0	5	0.004882813	1	1	1	1	0	1	1		
13	6	5	0.004882813	1	1	1	1	1	0	0		
14	245	5	0.004882813	1	1	1	1	1	0	1		
15	208	3	0.002929688	1	1	1	1	1	1	0	0	
16	43	2	0.001953125	1	1	1	1	1	1	0	1	
17	141	2	0.001953125	1	1	1	1	1	1	1	0	
18	110	1	0.000976563	1	1	1	1	1	1	1	1	0
19	159	1	0.000976563	1	1	1	1	1	1	1	1	1

Độ dài trung bình của từ mã: $\sum P_i \times n$

(với P_i là xác suất xuất hiện của màu i , n là số bit mã hóa)

$$0.204101563*2 + 0.17578125*3 + 0.1484375*3 + 0.122070313*3 + 0.115234375*3 + 0.080078125*4 + 0.05859375*4 + 0.057617188*4 + 0.0078125*6 + 0.006835938*7 + 0.004882813*7 + 0.004882813*7 + 0.004882813*7 + 0.002929688*8 + 0.001953125*8 + 0.001953125*8 + 0.000976563*9 + 0.000976563*9 = 3.147460969 \text{ (bit)}$$

$$\text{Dung lượng ảnh lúc sau} = 3.147460969*32*32 = 3223 \text{ (bit)}$$

$$\text{Hiệu suất mã hóa} = \frac{\text{Dung lượng ảnh lúc sau}}{\text{Dung lượng ảnh ban đầu}} * 100\% = 39.34\%$$

Nhận xét: Cả hai phương pháp Shannon và Fano đều cho kết quả ảnh sau khi mã hóa có độ tối ưu cao hơn, tiết kiệm dung lượng hơn (chỉ bằng khoảng 40% so với dung lượng ảnh ban đầu). Và dựa vào hiệu suất mã hóa, ta còn có thể thấy phương pháp mã hóa Fano có hiệu quả cao hơn phương pháp Shannon.