ΑΣΚΗΣΗ 1 (ΕΠΙΔΕΙΞΗ ΣΥΜΠΙΕΣΗΣ)
Αρκεί να βρούμε παρχ,nhuf,ndict, για τη συμπλήρωση των πινάκων

```
function [] = compressionDemo( )
A=imread( 'C:\Users\theodoros\Desktop\askisi1\plane1024.bmp'   );
Ad=double(A);
narxiko=(length(Ad)^2)*8;

foteinotitesused=unique(Ad(:));
foteinotitesall=Ad(:);
pithanotita=zeros(1,length(foteinotitesused));
for i=1:length(foteinotitesused
pithanotita(i)=length(foteinotitesall(foteinotitesall==foteinotitesused
(i)))/(length(foteinotitesall));
end
[Dict] = huffmandict(foteinotitesused,pithanotita);
%-------------------------------------------------------------------------------
ginomenodictbit=zeros(1,length(foteinotitesused));
for i=1:length(foteinotitesused)

ginomenodictbit(i)=length(foteinotitesall(foteinotitesall==Dict{i,1}))*l
ength(Dict{i,2});
end
nhuffman=sum(ginomenodictbit);
%-------------------------------------------------------------------------------
alldictionarybits=zeros(1,length(foteinotitesused));
for i=1:length(foteinotitesused)
    alldictionarybits(i)=length(Dict{i,2})+8;
end
ndict=sum(alldictionarybits);
%-------------------------------------------------------------------------------
pithanotitadict=(ndict/nhuffman)*100;
pithanotitahuffman=(nhuffman/narxiko)*100;
pithanotitall=((nhuffman+ndict)/narxiko)*100;
X = ['Τα χαρακτηριστικά της φωτογραφίας με μέγεθος',
num2str(length(A)),'x',num2str(length(A)),'είναι:'];
disp(X)
a=['Το παρχικό είναι:', num2str(narxiko) ];
disp(a)
b=['Το nHuffman είναι:',num2str(nhuffman)];
disp(b)
```

```
c=['Το ndictionary είναι:',num2str(ndict)];
disp(c)
pinakas2=['Για τον πίνακα 2 έχουμε:', '(ndict/nhuf)*100%
:',num2str(pithanotitadict)  '(nhuf/nαρχ)*100%  :',
num2str(pithanotitahuffman) '((nhuf+ndict)/(nαρχ))*100%  :'
,num2str(pithanotitall) ];
disp(pinakas2);
end
```

Πίνακας 1

Εικόνα	$n_{αρχ}$ (bit)	n_{huf} (bit)	n_{dict} (bit)	$n_{huf} + n_{dict}$ (bit)
1024x1024	8388608	5952458	3430	5955888
512x512	**2097152**	**1489013**	**3293**	**1492306**
256x256	**524288**	**372295**	**3270**	**375565**
128x128	**131072**	**93173**	**3124**	**96297**
64x64	**32768**	**23200**	**2782**	**25982**
32x32	**8192**	**5707**	**1887**	**7594**

Πίνακας 2

Εικόνα	$\frac{n_{dict}}{n_{huf}} \cdot 100\%$	$\frac{n_{huf}}{n_{αρχ}} \cdot 100\%$	$\frac{n_{huf}+n_{dict}}{n_{αρχ}} \cdot 100\%$
1024x1024	0.0576	70.9588	70.9997
512x512	**0.22115**	**71.0017**	**71.1587**
256x256	**0.87834**	**71.0096**	**71.6333**
128x128	**3.3529**	**71.0854**	**73.4688**
64x64	**11.914**	**70.8008**	**79.2908**
32x32	**33.0647**	**69.6655**	**97.7002**

Ενδεικτική εκτέλεση Demo για την 1024x1024 default αριθμών

```
>> compressionDemo
Τα χαρακτηριστικά της φωτογραφίας με μέγεθος1024x1024είναι:
Το παρχικό είναι:8388608
Το nHuffman είναι:5952458
Το ndictionary είναι:3430
Για τον πίνακα 2 έχουμε:(ndict/nhuf)*100%  :0.057623(nhuf/nαρχ)*100%  :70.9588((nhuf+ndict)/(nαρχ))*100%  :70.9997
```

Ποια η επίπτωση της αποθήκευσης του λεξικού μαζί με την εικόνα για μικρές και ποια για μεγαλύτερες εικόνες;
Καλύτερη συμπίεση στις μεγάλες εικόνες έναντι των μικρών αφού στις μικρές το λεξικό είναι συγκρίσιμο με το αρχικό μήκος των bits (συγκρίνω narxiko με nhuf+ndict)

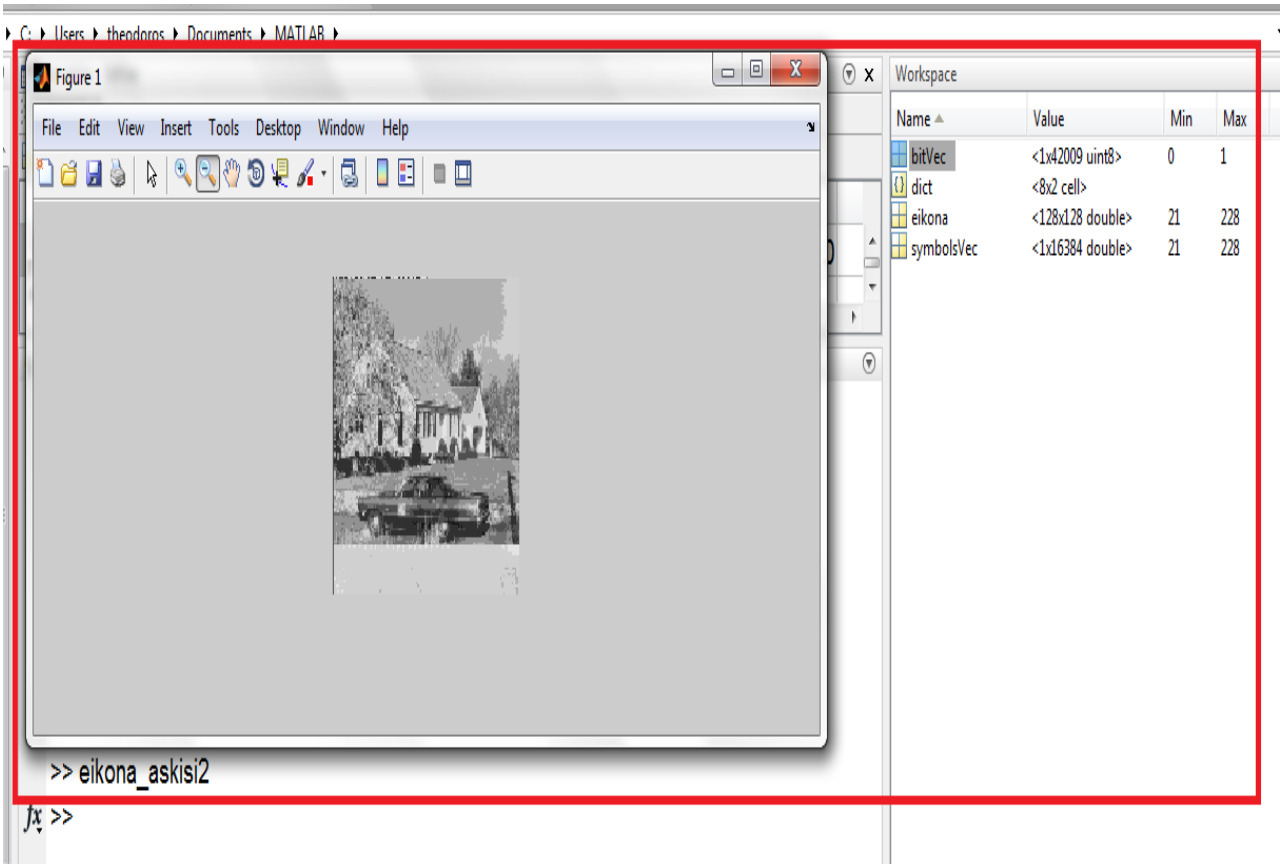
ΑΣΚΗΣΗ 2 (ΕΠΙΔΕΙΞΗ ΑΝΑΚΑΤΑΣΚΕΥΗΣ)
Αρχικά κατασκευάζουμε την Huffman Encode για την ανακατασκευή της εικόνας δοθέντος του λεξικού dict και της μεταβλητής bitVec

```
function [symbolsVec] = huffmanDecode(bitVec,dict)

symbolsVec=(zeros(1,16384));
```

```
n=0;
while size(bitVec) ~=0
n=n+1;
 for i =1:length(dict(:,2))
     if  length(bitVec)>=length(dict{i,2})
     if dict{i,2} == bitVec(1:length(dict{i,2}))
  symbolsVec(n)=dict{i,1};        bitVec(1:length(dict{i,2}))=[];
                 break;
     end;
     end; end; end
```

Demo επιδειξής ανακατασκευής eikona_askisi2.m

```
load ('C:\Users\theodoros\Desktop\askisi2\askisi2.mat');
[symbolsVec] = huffmanDecode(bitVec,dict);
eikona=reshape(symbolsVec,[128 128]);
imshow(uint8(eikona));
```



Επειτά καλούμαστε να δημιουργήσουμε την Huffman Encode και την reconstruction με σκοπό την συμπίεση και την αποσυμπίεση χωρίς απώλειες εικόνας 128x128

```
function [bitVec] = huffmanEncode(A,dict)
Ad=double(A);
```

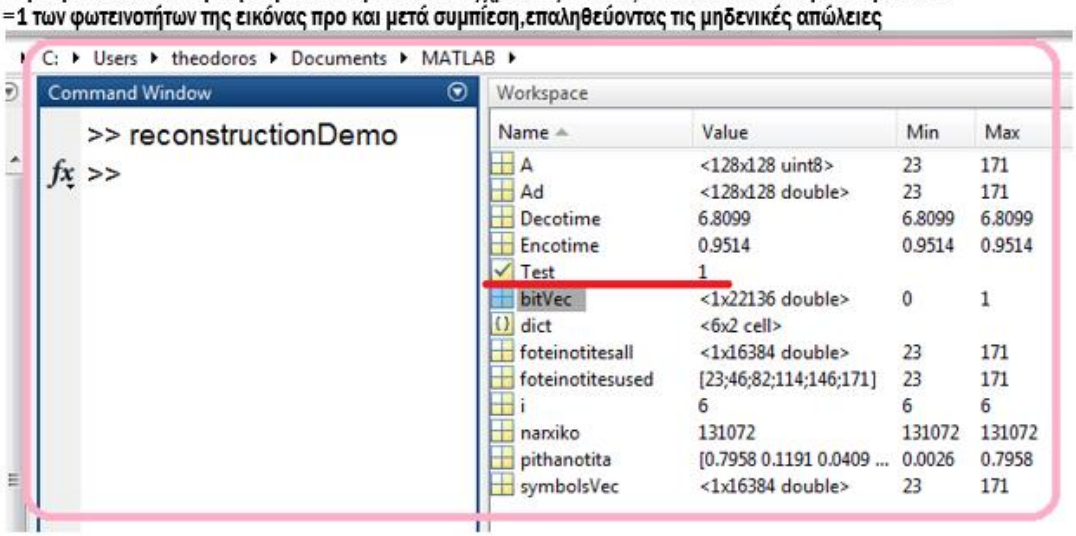
```
v=unique(Ad(:));
foteinotitesused=zeros(1,length(v));
ginomenodictbit=zeros(1,length(foteinotitesused));
for i=1:length(foteinotitesused)
ginomenodictbit(i)=length(foteinotitesused(foteinotitesused==dict{i,
1}))*length(dict{i,2});
end
nhuf=sum(ginomenodictbit);
%----------------------------------
bitVec=cell(1,nhuf);z=1;
for i=1:length(A)
    for j=1:length(v)
if A(i)==dict{j,1}
  bitVec{z}=dict{j,2};
     z=z+1;
          break;
 end
    end
end
bitVec=cell2mat(bitVec);
 end
```

reconstructionDemo

```
A=imread( 'C:\Users\theodoros\Desktop\askisi2\plane128_alt.bmp' ,'bmp' );
Ad=double(A);
narxiko=(length(Ad)^2)*8;

foteinotitesused=unique(Ad(:));
foteinotitesall=Ad(:)';
pithanotita=zeros(1,length(foteinotitesused));
for i=1:length(foteinotitesused)

pithanotita(i)=length(foteinotitesall(foteinotitesall==foteinotitesused(
i)))/(length(foteinotitesall));
end
[dict] = huffmandict(foteinotitesused,pithanotita);



tic;
[bitVec] = huffmanEncode(foteinotitesall,dict);
Encotime=toc;
tic;
[symbolsVec] =  huffmanDecode(bitVec,dict);
```

```
Decotime=toc;
Test=isequal(foteinotitesall,symbolsVec);
```

Στη παρακάτω εικόνα μπορούμε να δούμε τόσο τους χρόνους Encode/Decode όσο και την ισότητα Test =1 των φωτεινοτήτων της εικόνας προ και μετά συμπίεση,επαληθεύοντας τις μηδενικές απώλειες



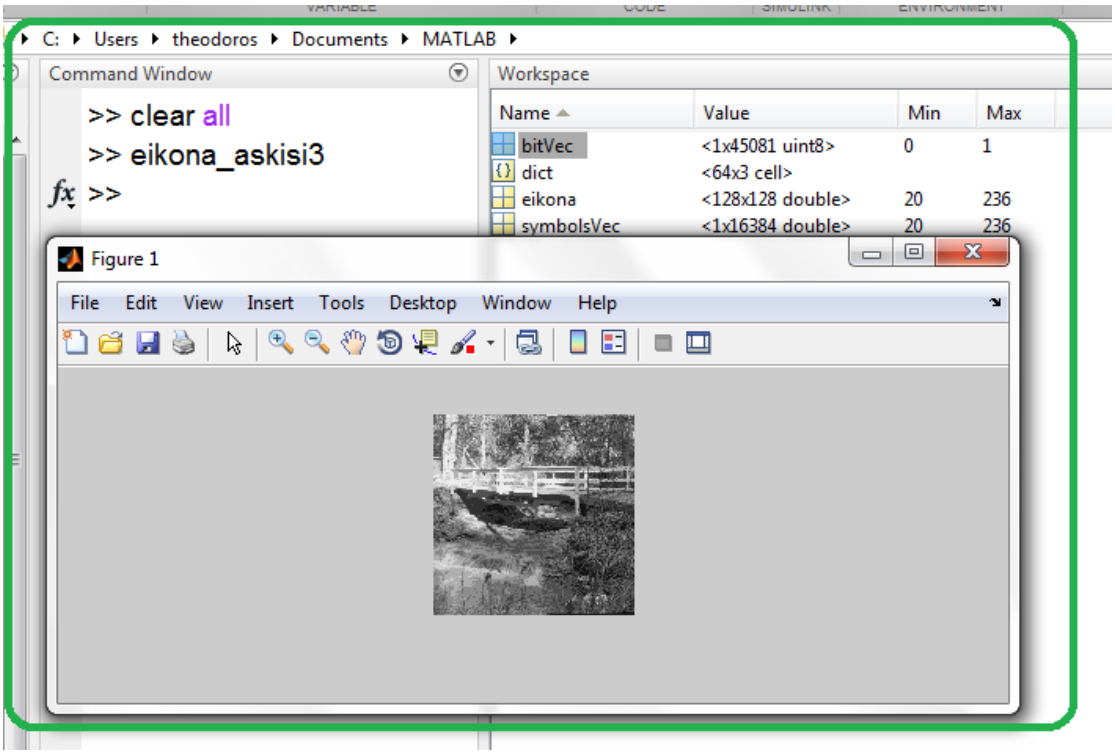
ΑΣΚΗΣΗ 3 (ΕΠΙΔΕΙΞΗ ΣΥΜΠΙΕΣΗΣ ΜΕ ΕΠΕΚΤΑΣΗ ΠΗΓΗΣ &ΑΝΑΚΑΤΑΣΚΕΥΗ)
Ομοία με τη προηγούμενη άσκηση δημιουργούμε μια νέα decode με επέκταση 2ης πηγής

```
function [symbolsVec] = huffman2Decode(bitVec,dict)
symbolsVec=(zeros(1,16384));
z=1;
while size(bitVec) ~=0

 for i =1:length(dict(:,2))
    if  length(bitVec)>=length(dict{i,2})
    if dict{i,3} == bitVec(1:length(dict{i,3}))
  symbolsVec(z)=dict{i,1};
   symbolsVec(z+1)=dict{i,2};
                bitVec(1:length(dict{i,3}))=[];
   z=z+2;
                break;
     end
    end
 end
end
end
```

Εκτελώντας το 2ο scriptaki eikona_askisi3 παιρνούμε την ανακατασκευασμένη εικόνα της 3ης άσκησης βάση λεξικού 3ης επέκτασης και της γνωστής μεταβλητής bitVec

```
load ('C:\Users\theodoros\Desktop\askisi3\askisi3.mat');
[symbolsVec] = huffman2Decode(bitVec,dict);
```

```
eikona=reshape(symbolsVec,[128 128]);
imshow(uint8(eikona));
```



Επείτα πρέπει να φτιάξουμε και μια Encode κατάλληλη για τη 2η επέκταση,όπως και μια νέα reconstruction για τη συμπλήρωση των πινάκων ένα και δύο

Huffman Encode-2

```
function [bitVec] = huffman2Encode(A,dict)
Ad=double(A);
foteinotitesall=Ad(:)';
nhuf2=0;
for i=0:2:length(foteinotitesall)-2
    for k=1:length(dict(:,1))
if foteinotitesall(i+1)==dict{k,1}&&foteinotitesall(i+2)==dict{k,2}
    nhuf2=nhuf2+length(dict{k,3});
    break
end
    end
end
%----------------------------------
bitVec=cell(1,nhuf2);
q=1;

for i=0:2:length(foteinotitesall)-2
    for j=1:length(dict(:,3))
```

```
if A(i+1)==dict{j,1} && A(i+2)==dict{j,2}
  bitVec{q}=dict{j,3};
     q=q+1;
            break;

end
    end
end
bitVec=cell2mat(bitVec);
end
```

prob_source_ext εργαστηρίου

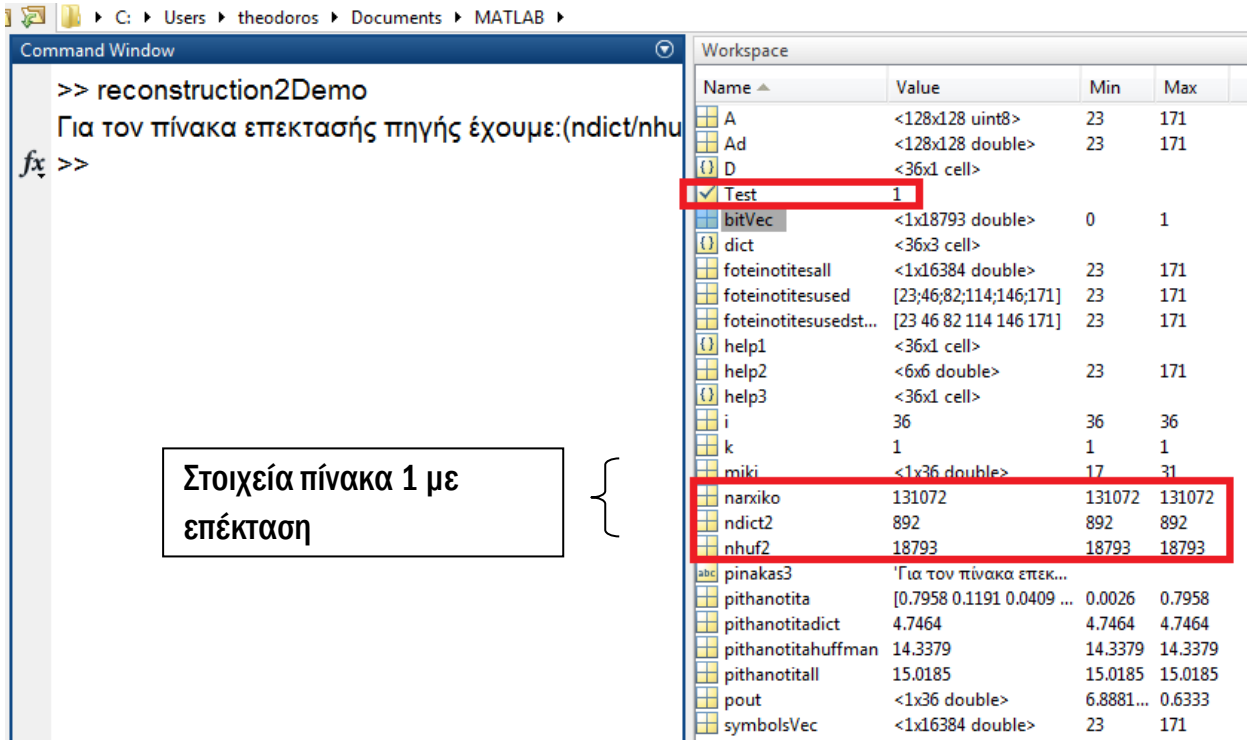
```
function pout = prob_source_ext(N,pin)
if N==1
pout = pin;
return
end
base = length(pin);
A = zeros( base^N , N);
basic_col = (1:base).';
for i=1:N
B = repmat( basic_col, 1, base^(i-1) );
B = B.';
v = B(:);
reps = base^(N-i);
A(:,i) = repmat(v, reps ,1);
end
pMat = pin(A);
pout = prod(pMat,2).';
end
```

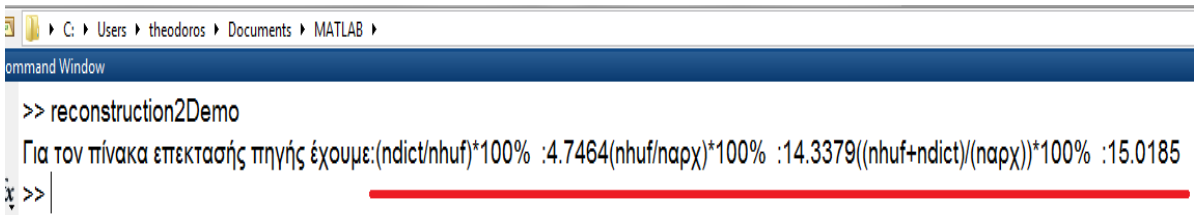
reconstruction2Demo

```
A=imread( 'C:\Users\theodoros\Desktop\askisi3\plane128_alt.bmp' ,'bmp' );
Ad=double(A);
narxiko=(length(Ad)^2)*8;
foteinotitesused=unique(Ad(:));
foteinotitesusedsthlh=foteinotitesused';
foteinotitesall=Ad(:)';
pithanotita=zeros(1,length(foteinotitesused));
for i=1:length(foteinotitesused)
```

```
pithanotita(i)=length(foteinotitesall(foteinotitesall==foteinotitesused(
i)))/(length(foteinotitesall));
end
pout = prob_source_ext(2,pithanotita);
dict = huffmandict((1:length(pout)),pout);
%--------------------------------------
nhuf2=0;
help1=repmat(foteinotitesused,length(foteinotitesused),1);

help2=repmat(foteinotitesusedsthlh,length(foteinotitesused),1);
help3=reshape(help2,36,1);

help1=num2cell(help1);
help3=num2cell(help3);
D=dict(:,2);
dict(:,1)=help1;
dict(:,2)=help3;
dict(:,3)=D;

for i=0:2:length(foteinotitesall)-2
    for k=1:length(dict(:,1))
if foteinotitesall(i+1)==dict{k,1}&&foteinotitesall(i+2)==dict{k,2}
    nhuf2=nhuf2+length(dict{k,3});
    break
end
    end
end

miki=zeros(1,length(dict(:,1)));
for i=1:length(dict(:,1))
    miki(i)=length(dict{i,3})+16;
end
ndict2=sum(miki);
pithanotitadict=(ndict2/nhuf2)*100;
pithanotitahuffman=(nhuf2/narxiko)*100;
pithanotitall=((nhuf2+ndict2)/narxiko)*100;

pinakas3=['Για τον πίνακα επεκτασής πηγής έχουμε:',
'(ndict/nhuf)*100%  :',num2str(pithanotitadict)  '(nhuf/ναρχ)*100%
:', num2str(pithanotitahuffman) '((nhuf+ndict)/(ναρχ))*100%  :'
,num2str(pithanotitall) ];

disp(pinakas3);
```

```
[bitVec]=huffman2Encode(A,dict);
[symbolsVec] =  huffman2Decode(bitVec,dict);
Test=isequal(foteinotitesall,symbolsVec);
```





Συμπλήρωση των ζητούμενων πινάκων

Τρέχουμε τη νέα εικόνα 128x128_alt στο compressionDemo και το reconstruction2Demo για τη συμπλήρωση των πινάκων

```
>> compressionDemo
Τα χαρακτηριστικά της φωτογραφίας με μέγεθος128x128είναι:
Το ναρχικό είναι:131072
Το nHuffman είναι:22136
Το ndictionary είναι:68
Για τον πίνακα 2 έχουμε:(ndict/nhuf)*100% :0.30719(nhuf/naρχ)*100% :16.8884((nhuf+ndict)/(naρχ))*100% :16.9403
>> reconstruction2Demo
Για τον πίνακα επεκτασής πηγής έχουμε:(ndict/nhuf)*100% :4.7464(nhuf/naρχ)*100% :14.3379((nhuf+ndict)/(naρχ))*100% :15.0185
fx >>
```

Πίνακας 1

Εικόνα	$n_{αρχ}$ (bit)	n_{huf} (bit)	n_{dict} (bit)	$n_{huf} + n_{dict}$ (bit)
128 x 128 **χωρίς** επέκταση της πηγής	131072	**22136**	**68**	**22204**
128 x 128 **με** επέκταση της πηγής **2ης τάξης**	131072	**18793**	**892**	**19685**

Πίνακας 2

Εικόνα	$\frac{n_{dict}}{n_{huf}} \cdot 100\%$	$\frac{n_{huf}}{n_{αρχ}} \cdot 100\%$	$\frac{n_{huf}+n_{dict}}{n_{αρχ}} \cdot 100\%$
128 x 128 **χωρίς** επέκταση της πηγής	**0.30719**	**16.8884**	**16.9403**
128 x 128 **με** επέκταση της πηγής **2ης τάξης**	**4.7464**	**14.3379**	**15.0185**