GÖRÜNTÜ SIKIŞTIRMA

Görüntü Sıkıştırma Nedir

- Sayısal görüntünün hafızada kapladığı alanı azaltan, kayıplı ve kayıpsız yapılabilen bir işlemdir.
- Sıkıştırılmış görüntü tekrar açıldığında orijinal görüntüden farklı ise kayıplı değilse kayıpsız sıkıştırmadır.
- Kayıpsız sıkıştırma genellikle pikseller arası korelasyonun düşük olduğu şekil gibi görüntülerde, kayıplı sıkıştırma ise fotoğraf görüntüleri gibi karmaşık ve yüksek korelasyona sahip görüntülerde kullanılır.

Run Length Encoding

- Hem kayıplı hem kayıpsız uygulanabilir.
- Uygulaması en kolay algoritmalardan biridir.
- Aynı sembolün ardışık olarak çok defa tekrar etmesi durumunda iyi bir sıkıştırma oranı sağladığı için daha çok siyah beyaz görüntülerin sıkıştırılmasında kullanılır.
- Run başlangıç pikseli olup, length o pikselden yan yana kaç adet olduğunu belirtir.
- Length için başlangıçta kaç bit yer ayrılacağı belirlenir.
- Görüntü satır satır işlenir.

Run Length Encoding – Kayıpsız

- 001111000000011111111111111000000001111000000000000000000 görüntüsünü 4 bitlik uzunluk kullanarak sıkıştıralım.

- Toplam 60 piksel=60 bit
- Sıkıştırılmış dosya: 0(2)-1(4)-0(7)-1(15)-0(8)-1(4)-0(15)-0(5)

00001 10011 00110 11110 00111 10011 01110 00100

R A R A R A R A R A R A R A R A

Toplam bit sayısı : 40

Sıkıştırma oranı =Orjinal bit sayısı/Sıkışmış bit sayısı =60/40= 1.5:1

Not: 4 bit ile 16 ifade edilemediğinden 1 düşürülerek adet belirlenir, açarken de 1 ilave edilir.

Run Length Encoding – Kayıpsız

- Çok yüksek frekanslı görüntülerde sıkıştırma oranları düşer hatta orijinal boyutu geçebilir. Bu durumu optimize etmek için maksimum uzunluk ve onun için gerekli bit sayısı azaltılmalıdır.

- 00101101001010100101 – 20 piksel, maksimum uzunluk 4 bit olursa

0(2)-1(1)-0(1)-1(2)-0(1)-1(1)-0(2)-1(1)-0(1)-1(1)-0(1)-1(1)-0(2)-1(1)-0(1)-1(1)

00001-10000-00000-10001-00000-10000-00001-10000-00000-10000-00000-10000-00001-10000-00000-10000

Sıkışmış dosyadaki bit sayısı: 16*5=80

Sıkıştırma oranı =Orijinal bit sayısı/Sıkışmış bit sayısı =20/80= 1:4

Run Length Encoding – Kayıpsız

- Aynı örnek optimize edilsin. Maksimum uzunluk 1 bit olsun.

0(2)-1(1)-0(1)-1(2)-0(1)-1(1)-0(2)-1(1)-0(1)-1(1)-0(1)-1(1)-0(2)-1(1)-0(1)-1(1)

01-10-00-11-00-10-01-10-00-10-00-10-01-10-00-10

Sıkışmış dosyadaki bit sayısı: 16*2=32

Sıkıştırma oranı =Orijinal bit sayısı/Sıkışmış bit sayısı =20/32= 0.625:1

Bu görüntü için uygun bir yöntem olmadığı söylenebilir.

Run Length Encoding – Kayıplı

- Başlangıç piksel değerinden bir eşik ton değeri aralığında kalan piksel adedi sayılır.
- Yeniden oluşturmada referans piksel kullanılır.
- Maksimum adet baştan sınırlanır.
- Kayıplı olduğu için yeniden oluşturulan görüntüde hata oluşur.
- Gri veya renkli görüntülerde tercih edilir.

Run Length Encoding – Kayıplı

- 50 52 47 49 45 48 48 62 61 57 53 69 59 65 30 30 40 30 35 39 45 45 33 32 42 49 54 51 50 50

görüntüsü verilsin.

256 ton değeri mevcut.

Maksimum uzunluk 8 olsun.

Eşik değeri 10 için sonuç ne olur bakalım.

Run Length Encoding – Kayıplı

Sıkışmış dosya: 50(7)-62(7)-30(6)-45(2)-33(3)-49(5)

Orijinal dosyanın bit sayısı : 30 piksel* 8 bit = 240 bit

Sıkıştırılmış dosyanın bit sayısı : 6* 11 bit = 66 bit

11 bit =(8 bit pikselden, 3 bit ise sayıdan)

Sıkıştırma oranı =Orijinal bit sayısı/Sıkışmış bit sayısı =240/66=3.64:1

Açılan görüntü : 50 50 50 50 50 50 50 62 62 62 62 62 62 62 30 30 30 30 30 30 30 45 45 33 33 33 49 49 49 49 49

Huffman Kodlama

- Huffman kodlama, kayıpsız bir sıkıştırma algoritmasıdır.
- Bu kodlama değişken uzunlukta bir kodlama olup, veri setindeki karakterlerin frekansına bağlıdır.
- Algoritma, bir veri setinde daha çok rastlanan bir sembolün daha düşük uzunluktaki kodla, daha az rastlanan sembolün ise daha büyük uzunluktaki kodla temsil edilmesine dayanmaktadır.
- Bir veri kümesini sıkıştırabilmek için bu küme içerisindeki sembollerin tekrar etme sıklıklarının bilinmesi gerekmektedir. Her sembolün ne kadar sıklıkta tekrar ettiğini gösteren tabloya frekans tablosu denir.

Huffman Kodlama

- Veri setine ait frekans tablosu aşağıdaki gibi olsun.

Sembol	**Frekans**
A	**60**
B	**40**
C	**25**
D	**20**
E	**10**

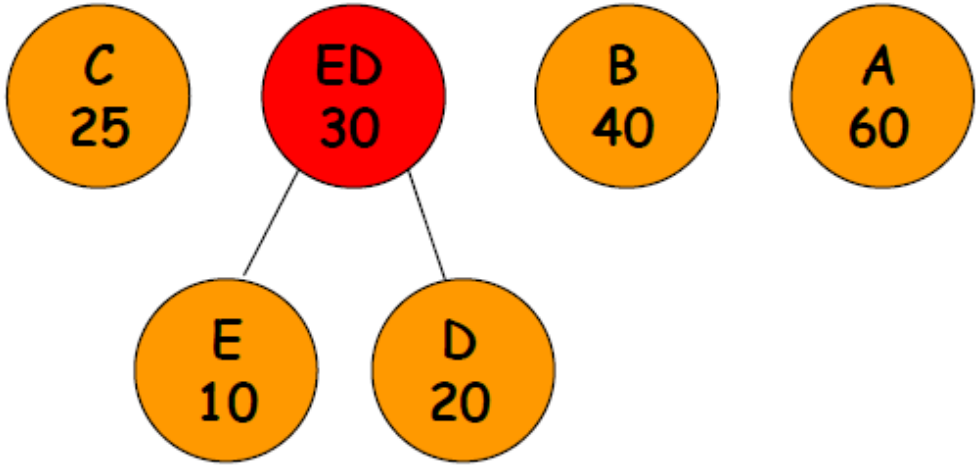
Huffman Kodlama

- Huffman ağacındaki en son düğümleri oluşturacak olan bütün semboller frekanslarına göre küçükten büyüğe doğru sıralanırlar.

E	**D**	**C**	**B**	**A**
10	**20**	**25**	**40**	**60**

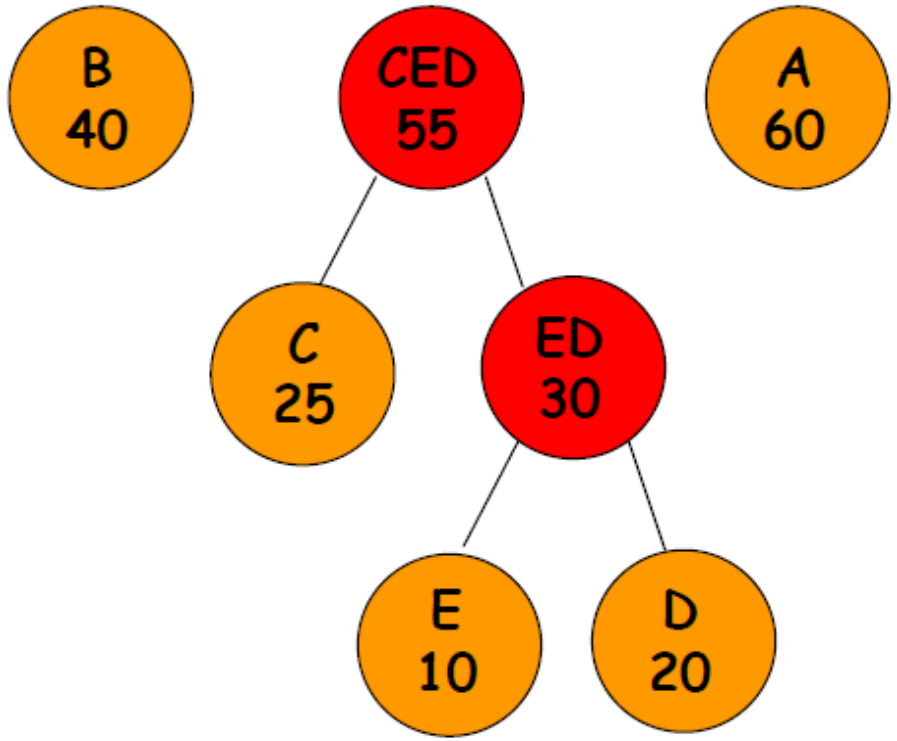
Huffman Kodlama

- En küçük frekansa sahip olan iki sembolün frekansları toplanarak yeni bir düğüm oluşturulur. Oluşturulan bu yeni düğüm var olan düğümler arasında uygun yere yerleştirilir. Bu yerleştirme frekans bakımından küçüklük veya büyüklüğe göredir.

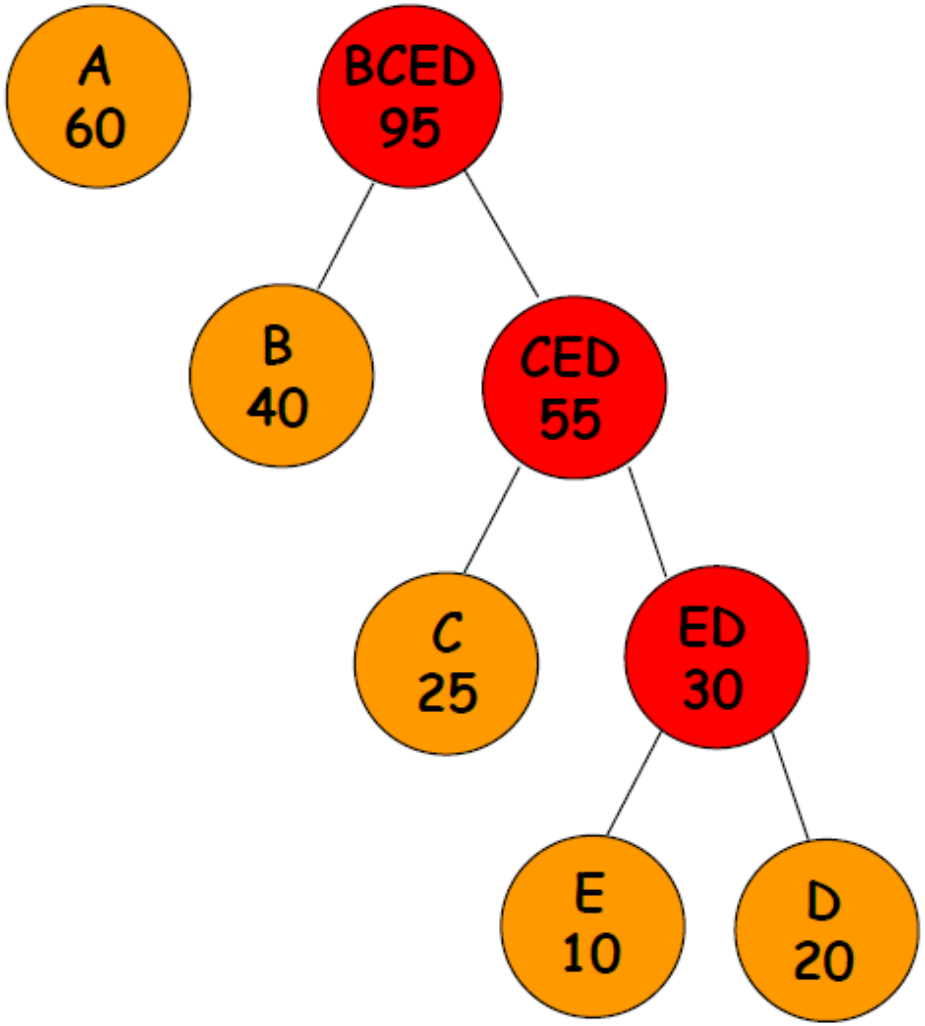


Huffman Kodlama

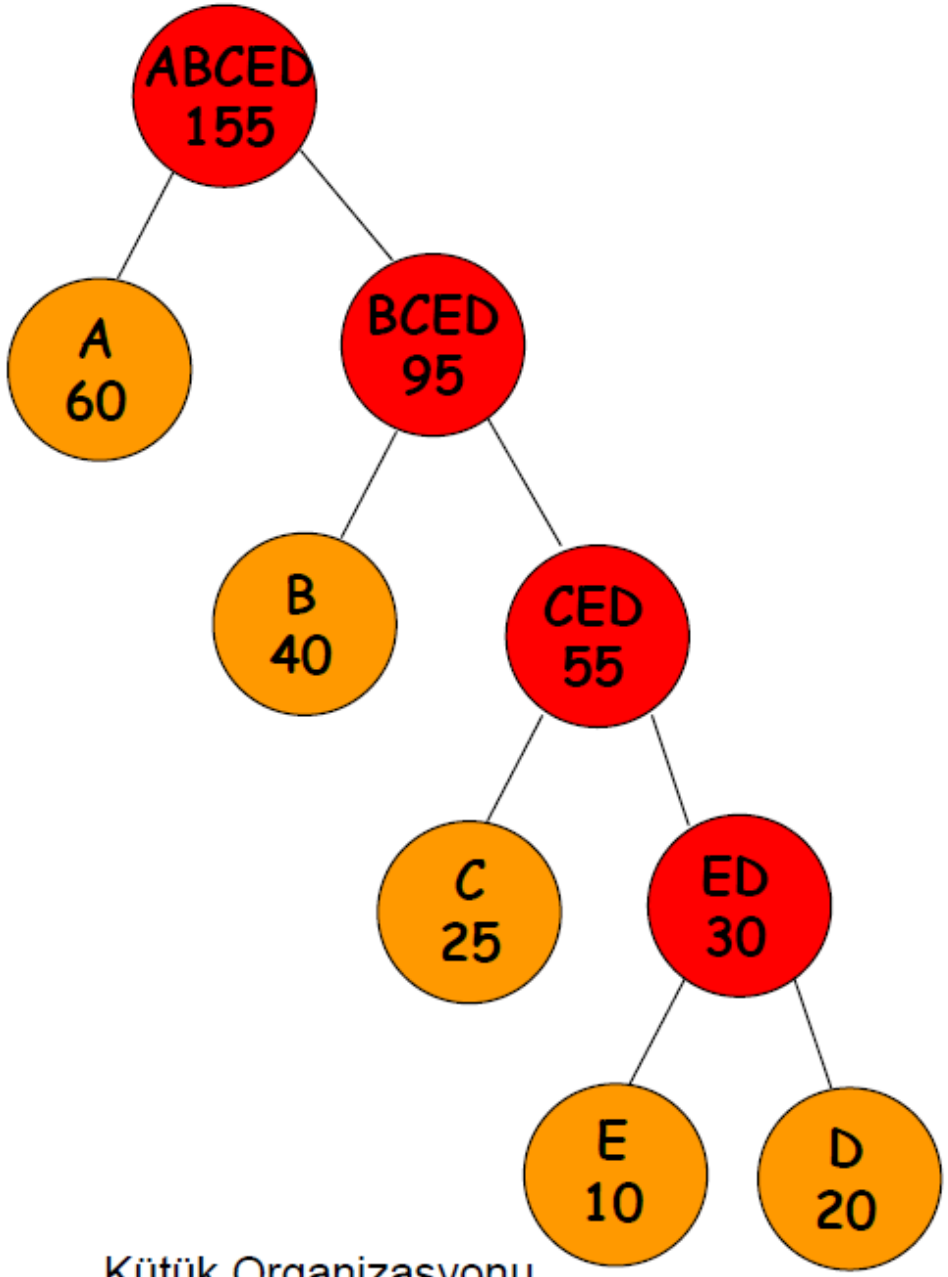
- Bir önceki adımdaki işlem tekrarlanarak en küçük frekanslı 2 düğüm tekrar toplanır ve yeni bir düğüm oluşturulur.



Huffman Kodlama



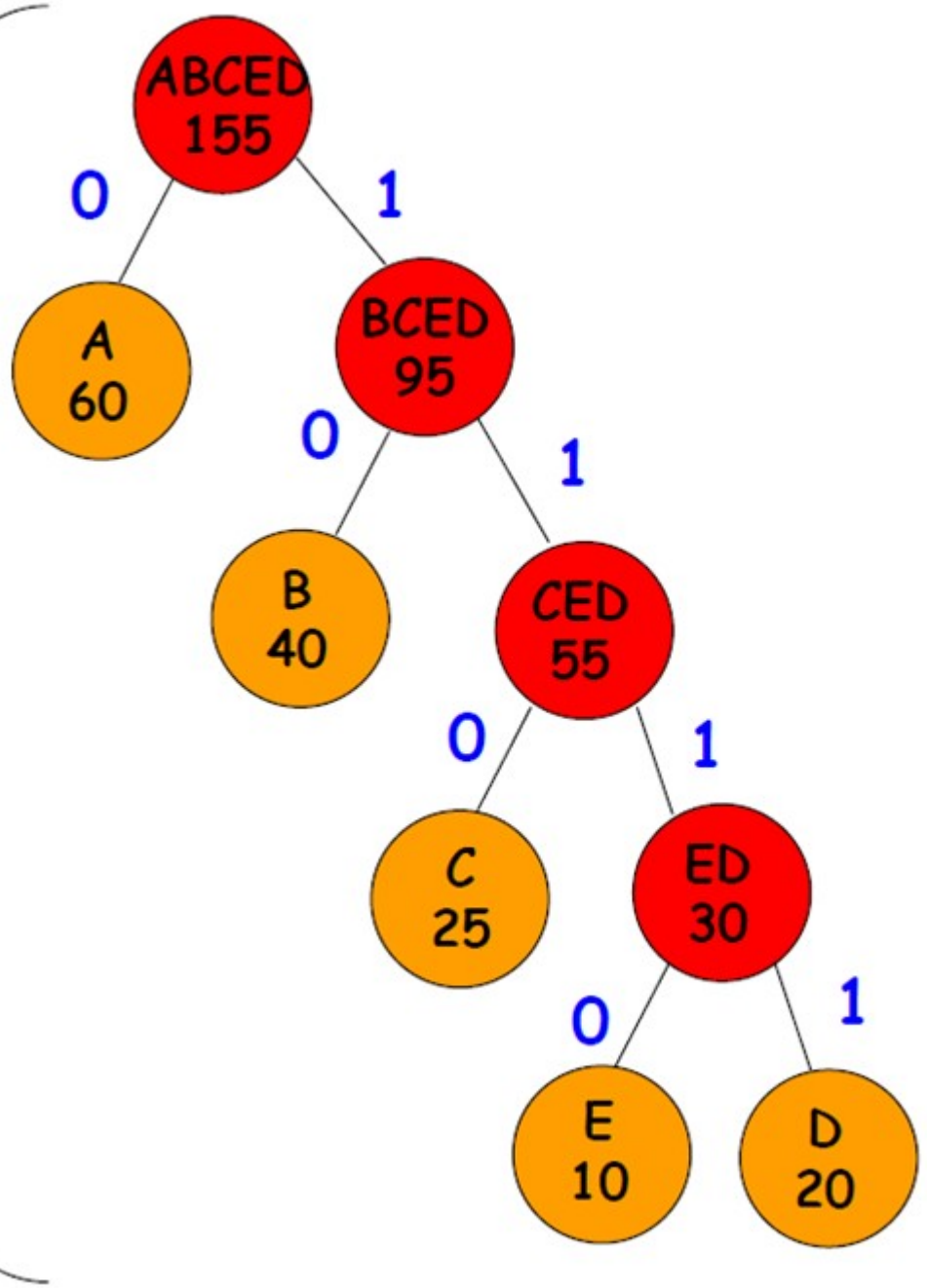
Huffman Kodlam



Huffman Kodlama

Huffman ağacının kodu oluşturulurken ağacın en tepesindeki kök düğümden başlanır.

Kök düğümün soluna ve sağına giden dallara sırasıyla 0 ve 1 yazılır. Sırası seçimliktir.



Huffman Kodlama

Sembol	Huffman Kodu	Bit Sayısı	Frekans
A	0	1	60
B	10	2	40
C	110	3	25
D	1111	4	20
E	1110	4	10

60x1 + 40x2 + 25x3 + 20x4 +10x4= 335 bit

Eğer veri seti ASCII kodu ile kodlansaydı, her karakter için 1 byte (8 bit)'lık alan gerektiğinden toplamda 155 byte(1240 bit) gerekecekti.

Huffman Kodlama

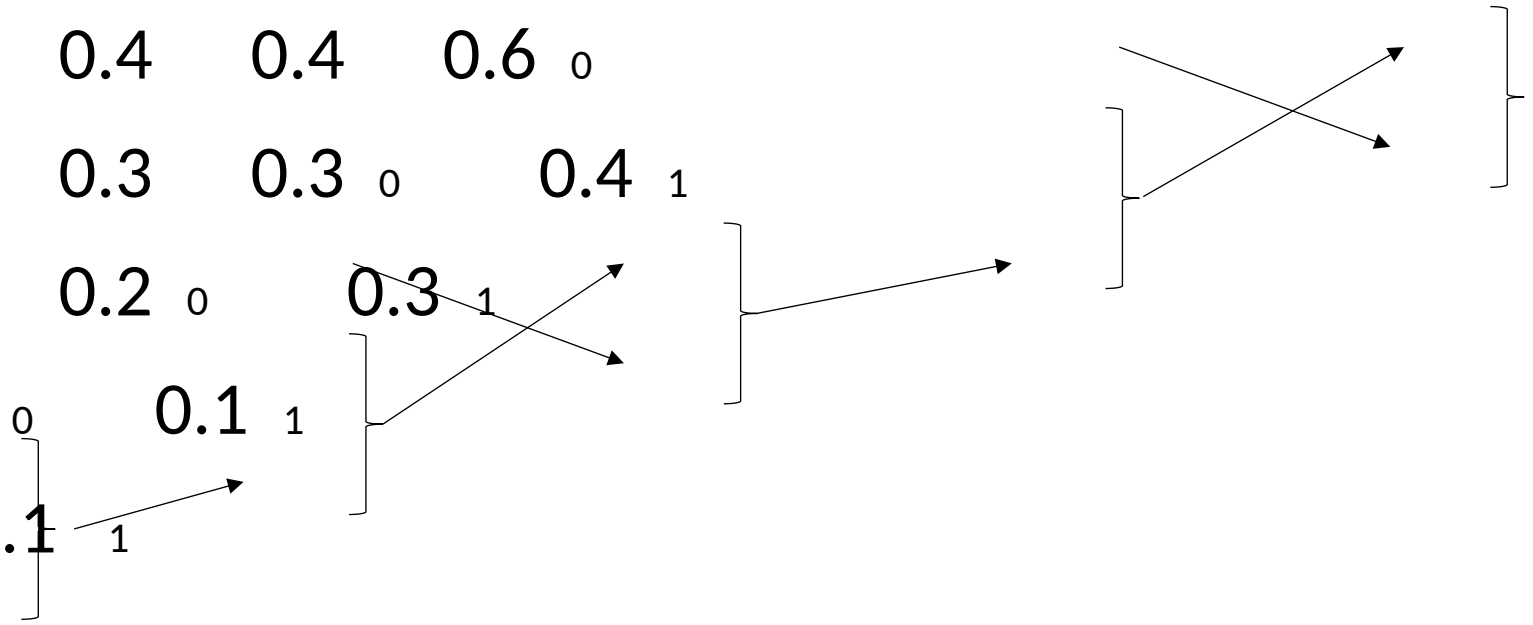
- 6 gri tonlamaya sahip bir görüntü içinde her bir tonun bulunma olasılıkları şöyledir :

t_1=0.1 t_2=0.4 t_3=0.06 t_4=0.1 t_5=0.04 t_6=0.3

- Görüntü 256*256 pikseldir. Huffman kodlama uygulanırsa,

Huffman Kodlama

Sembol	Olasılık	Adım1	Adım2	Adım3	Adım4
t_2	0.4	0.4	0.4	0.4	0.6 0
t_6	0.3	0.3	0.3	0.3 0	0.4 1
t_1	0.1	0.1	0.2 0	0.3 1	
t_4	0.1	0.1 0	0.1 1		
t_3	0.06 0	0.1 1			
t_5	0.04 1				



Huffman Kodlama

Sembol	Olasılık	Kod
t_2	0.4	1
t_6	0.3	00
t_1	0.1	011
t_4	0.1	0100
t_3	0.06	01010
t_5	0.04	01011

Ortalama uzunluk = 0.4*1+0.3*2+0.1*3+0.1*4+0.06*5+0.04*5 = 2.2 bit/piksel

Sıkıştırma oranı = 3/2.2 = 1.36:1

(6 gri ton 3 bit ile ifade edilebilir 2^3=8 durum)

Huffman Kodlama

- Frekans tablosu ve sıkıştırılmış veri mevcutsa bahsedilen işlemlerin tersini yaparak kod çözülür. Diğer bir değişle:
- Sıkıştırılmış verinin ilk biti alınır. Eğer alınan bit bir kod sözcüğüne karşılık geliyorsa, ilgili kod sözcüğüne denk düşen karakter yerine konulur.
- Eğer alınan bit bir kod sözcüğü değil ise sonraki bit ile birlikte alınır ve yeni dizinin bir kod sözcüğü olup olmadığına bakılır. Bu işlem dizinin sonuna kadar yapılır. Böylece huffman kodu çözülerek karakter dizisi elde edilmiş olur.

LZW (Lempel-Ziv-Welch) Kodlama

- Kayıpsız bir sıkıştırma algoritmasıdır.
- İşlemin başında kodlanacak kaynak sembolleri içeren bir kod kitabı ya da sözlük oluşturulur. 8 bitlik bir gri-seviyeli görüntü için her bir gri değeri (0,1,…,255) sözlüğe atanır.
- Algoritmanın değerleri ardışıl olarak incelemesi nedeniyle, sözlükte bulunmayan gri değer dizileri sözlüğe eklenir. Örneğin, görüntünün ilk iki pikseli beyaz ise, 255-255 dizisi sözlüğün 256. konumuna atanabilir. İki ardışık beyaz pikselle karşılan daha sonraki bir durumda bu diziyi ifade etmek için 256 değeri kullanılır.
- Sözlüğün boyutu önemli bir parametredir. Eğer çok küçükse eşleşen gri seviye dizilerinin algılanması çok az olacaktır. Eğer çok büyükse kod kelimelerinin büyüklüğü sıkıştırma performansını ters etkileyecektir.

LZW Kodlama

- 39 39 126 126 39 39 126 126 39 39 126 126 39 39 126 126 şeklinde 8 bitlik bir görüntü verilsin. 512 bitlik bir sözlük kullanılacağı varsayılsın.

Şu anda tanınan dizi	İşlenecek piksel	kodlanmış çıkış	sözlük konumu (kod kelimesi)	sözlük girişi
	39			
39	39	39	256	39-39
39	126	39	257	39-126
126	126	126	258	126-126
126	39	126	259	126-39
39	39			
39-39	126	256	260	39-39-126
126	126			
126-126	39	258	261	126-126-39
39	39			
39-39	126			
39-39-126	126	260	262	39-39-126-126
126	39			
126-39	39	259	263	126-39-39
39	126			
39-126	126	257	264	39-126-126
126		126		

LZW Kodlama

- 32 32 34 32 34 32 32 33 32 32 32 34 şeklinde bir kod dizisi verilsin.

Şu anda tanınan dizi	İşlenecek piksel	kodlanmış çıkış	sözlük konumu (kod kelimesi)	sözlük girişi
	32			
32	32	32	256	32-32
32	34	32	257	32-34
34	32	34	258	34-32
32	34			
32-34	32	257	259	32-34-32
32	32			
32-32	33	256	260	32-33
33	32	33	261	33-32
32	32			
32-32	32	256	262	32-32-32
32	34			
32-34		257		

Kayıpsız Sıkıştırma Kullanan Görüntü Formatları

GIF (Graphics Interchange Format)

- En fazla 8-bit renk derinliğine (28 = 256 renk) sahip görüntülere izin verir.
- Bu nedenle fotoğraf görüntülerinin sıkıştırılmasında yetersiz kalsa da, birkaç rengin çoğunlukta olduğu grafiksel gösterimler ve basit şekiller gibi görüntülerin kayıpsız olarak sıkıştırılmasında halen kullanılmaktadır.
- LZW (Lempel Ziv Welch) tabanlıdır.

Kayıpsız Sıkıştırma Kullanan Görüntü Formatları

TIFF (Tagged Image File Format)

- GIF den farklı olarak 24-bit RGB (8-bit Red + 8-bit Green + 8-bit Blue) ile de çalışabilmektedir.
- 24-bit renk derinliğini desteklemesi sayesinde fotoğrafların kayıpsız olarak sıkıştırılmasında kullanılır.

Kayıpsız Sıkıştırma Kullanan Görüntü Formatları

PNG (Portable Network Graphics)

- 48-bit gerçek renk, 16-bit gri tonlama desteği, ve yüksek oranda sıkıştırma becerisi sayesinde GIF ve TIFF standartlarından daha iyi bir formattır.
- DEFLATE veri sıkıştırma yöntemini kullanır.

JPEG

- DCT (Discrete Cosine Transform) tabanlıdır.
- 24-bit RGB görüntüler üzerinde çalışabilir. 24-bit renk derinliğine sahip fotoğraflarda çok iyi sonuçlar verdiği için sayısal fotoğrafçılıkta çok kullanılır.
- Fakat bir rengin ağırlıkta olduğu resimlerde sıkıştırma oranı azalır.
- JPEG, tek bir algoritmadan ibaret olmayıp, birçok görüntü sıkıştırma yönteminin bir araya getirilmesi ile oluşturulmuş bir standart olduğu için, sıkıştırma ve açma süreleri tek algoritmaya sahip olan kayıpsız sıkıştırma yöntemlerine göre daha fazladır.