VERİ DEPOALMA VE SIKIŞTIRMA ALGORİTMALARI

ÖĞRENCİ İSMİ :
Ali Osman Karaaslan

ÖĞRENCİ NUMARASI :
25360859431

Bursa Teknik Üniversitesi Bilgisayar Mühendisliği BMG Projesi

İçindekiler

3.....Metin Verisinin Bit Düzeyinde Temsili
4.....Resim Verisinin Bit Düzeyinde Temsili
6.....Ses Verisinin Bit Düzeyinde Temsili
9.....Veri Sıkıştırma
10.....Veri Sıkıştırma Neden Gerekli
12.....Veri Sıkıştırma Algoritmaları
14.....Kayıplı Sıkıştırma Algoritmaları
17.....Kayıpsız Sıkıştırma Algoritmaları
24.....Kaynaklar
25.....Son

Metin (Text) Verisinin Bit Düzeyinde Temsili

Bilgisayarlar metni karakter kodlama sistemleri ile saklar.

ASCII :

Her karakter 1 byte (8 bit) ile temsil edilir.

"Ali" kelimesi :

A → 01000001
l → 01101100
i → 01101001

KOD	KARAKTER	KOD	KARAKTER	KOD	KARAKTER	KOD	KARAKTER	KOD	KARAKTER
00	NULL(Null character)	51	3	103	g	156	£	208	º
01	SOH(Start of Header)	52	4	104	h	157	Ø	209	ª
02	STX(Start of Text)	53	5	105	i	158	Ş	210	Ê
03	ETX(End of Text)	54	6	106	j	159	ş	211	Ë
04	EOT(End of Trans.)	55	7	107	k	160	á	212	È
05	ENQ(Enquiry)	56	8	108	l	161	í	213	?
06	ACK(Acknowledgement)	57	9	109	m	162	ó	214	Í
07	BEL(Bell)	58	:	110	n	163	ú	215	Î
08	BS(Backspace)	59	;	111	o	164	ñ	216	Ï
09	HT(Horizontal Tab)	60	<	112	p	165	Ñ	217	┘
10	LF(Line feed)	61	=	113	q	166	Ğ	218	┌
11	VT(Vertical Tab)	62	>	114	r	167	ğ	219	▉
12	FF(Form feed)	63	?	115	s	168	¿	220	▄
13	CR(Carriage return)	64	@	116	t	169	®	221	¦
14	SO(Shift Out)	65	A	117	u	170	¬	222	Ì
15	SI(Shift In)	66	B	118	v	171	½	223	▀
16	DLE(Data link escape)	67	C	119	w	172	¼	224	Ó
17	DC1(Device control 1)	68	D	120	x	173	¡	225	ß

Resim (Image) Verisinin Bit Düzeyinde Temsili

Renk derinliği, diğer adıyla bit derinliği , tek bir pikselin rengini belirtmek için kullanılan bit sayısı veya tek bir pikselin her renk bileşeni için kullanılan bit sayısıdır.

Siyah-Beyaz (1 bit) :

0 → Siyah
1 → Beyaz

tr.wikipedia.org

Resim (Image) Verisinin Bit Düzeyinde Temsili

Gri Seviye (8 bit) :

Her piksel 0–255 arası değer alır.

Renkli (RGB – 24 bit) :

Her piksel:

- 8 bit Kırmızı
- 8 bit Yeşil
- 8 bit Mavi

tr.wikipedia.org

Ses (Audio) Verisinin Bit Düzeyinde Temsili

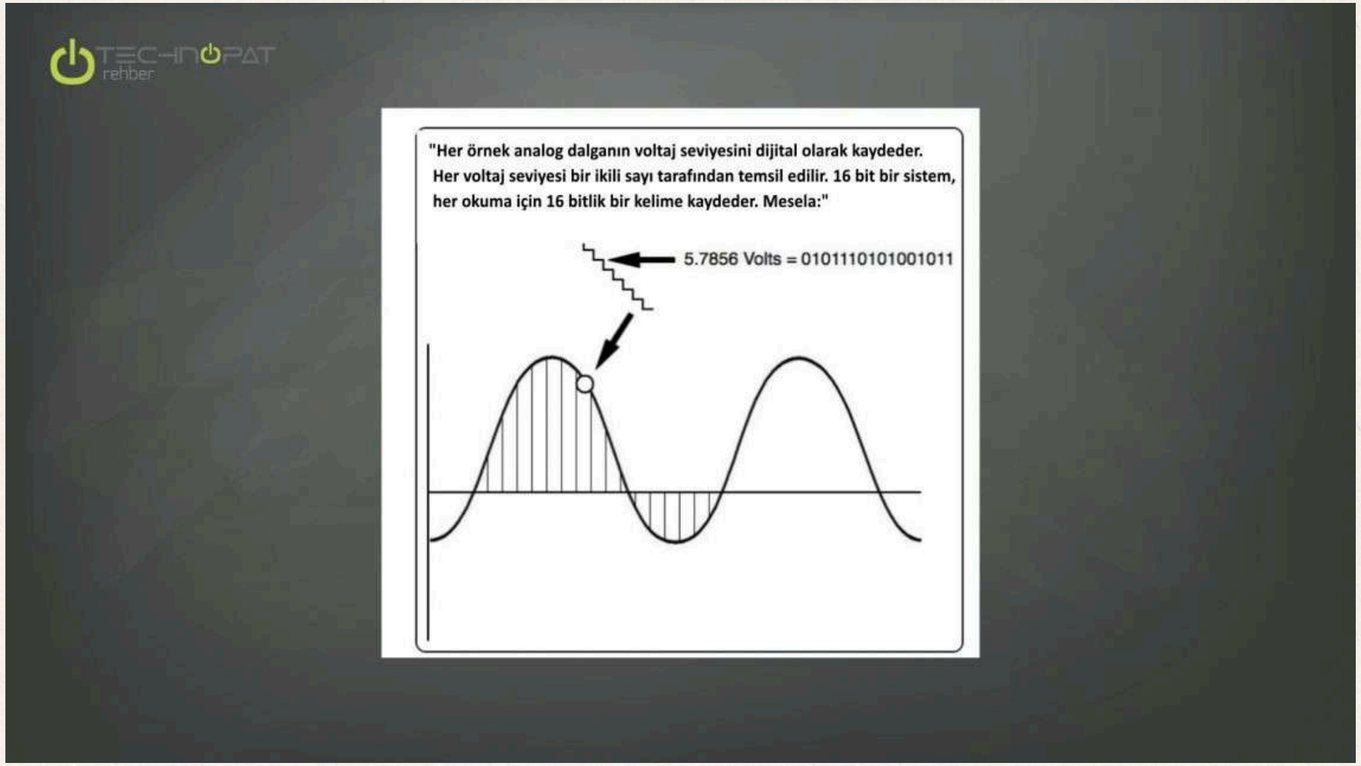
Ses, analog dalgadır → dijitalleştirilir.

Temel Adımlar :

1- Örnekleme (Sampling)

2- Nicemleme (Quantization)

3- Kodlama (Encoding)



Örnekleme (Sampling Rate):

Saniyede kaç örnek alındığıdır.

CD Kalitesi → 44.100 Hz

Ses (Audio) Verisinin Bit Düzeyinde Temsili

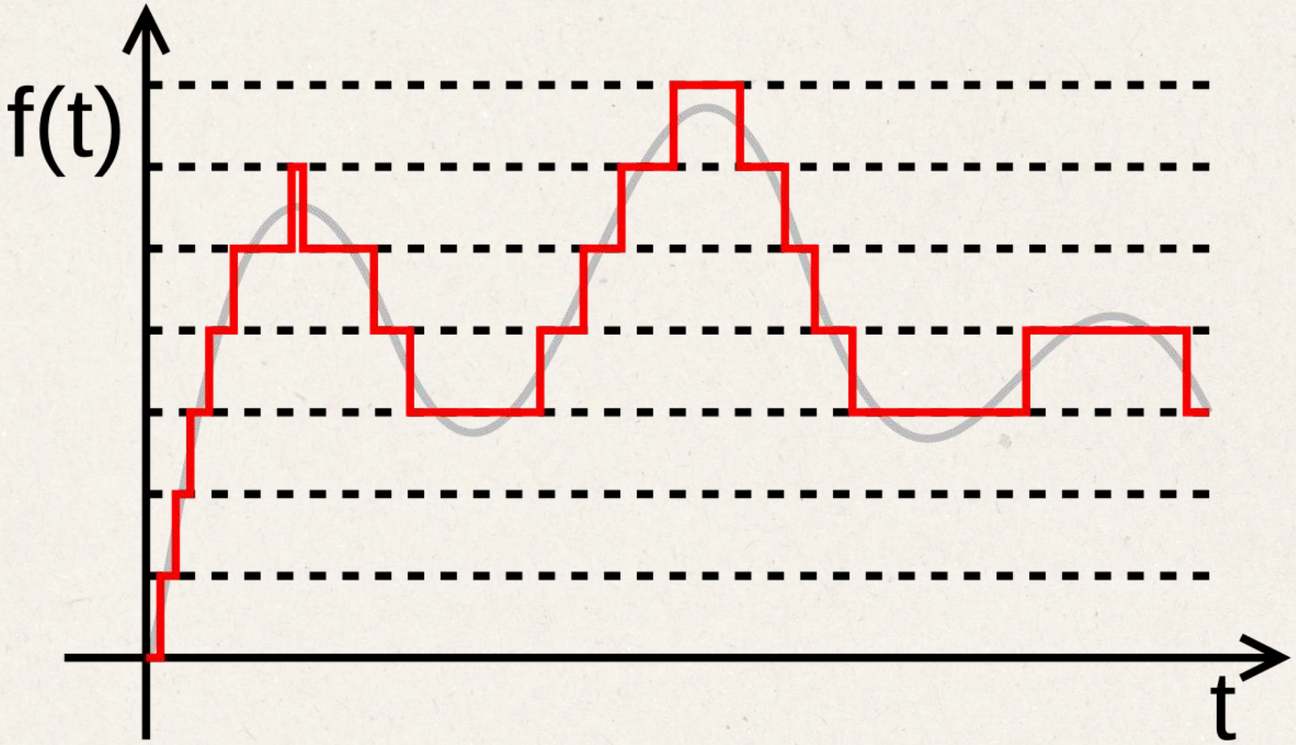
Nicemleme (Bit Derinliği) :

Her örneğin kaç bit ile tutulduğu

8 bit → 256

16 bit → 65.536

Nicemleme, örneklenmiş (sample edilmiş) analog genlik değerlerinin, belirli sayıda dijital seviyeye yuvarlanmasıdır (ADC kullanılır).



Ses (Audio) Verisinin Bit Düzeyinde Temsili

Kodlama (Encoding) :

Kodlama, nicemleme sonucu elde edilen sayısal seviyelerin, ikili (binary) bit dizilerine dönüştürülmesidir.

Analog Ses

↓

Örnekleme → zaman ayrık (Sinyalin sadece belirli zaman anlarında ölçülmes)

↓

Nicemleme → genlik ayrık(Ölçülen genliğin, belirli basamaklara yuvarlanması)

↓

Kodlama → bit dizisi (0–1)

Veri Sıkıştırma

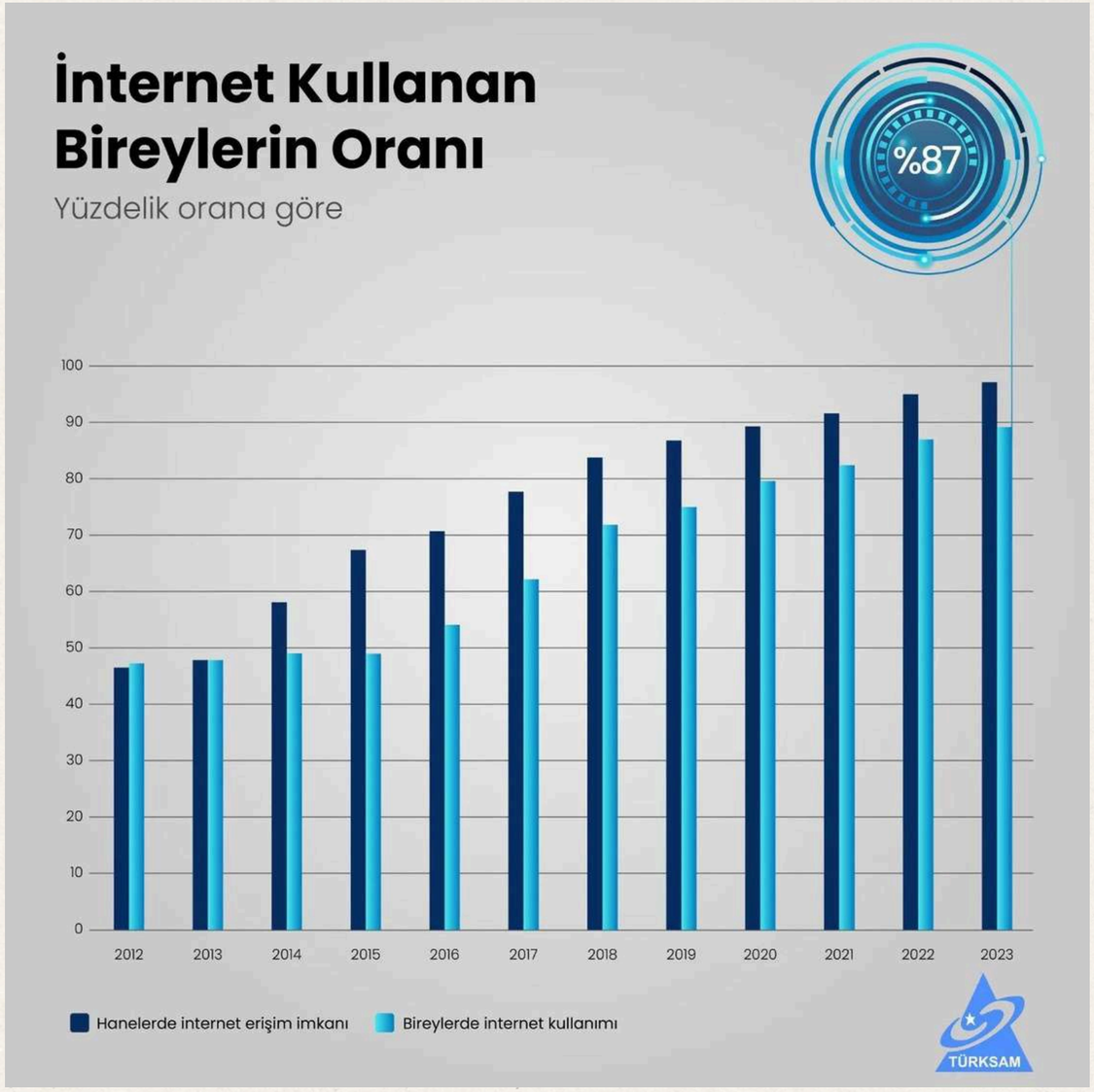
Bilgisayardaki veya belleği olan herhangi bir elektronik cihazdaki verilerin daha az yer kaplaması amacıyla sıkıştırılması anlamına gelir.

tr.wikipedia.org

Bilgisayarda, Windows işletim sistemi altında en yaygın olan veri (dosya) sıkıştırma programları WinZip ve WinRAR iken Linux sisteminde ise Gzip, Bzip2'dır.

Veri Sıkıştırma Neden Gerekli ?

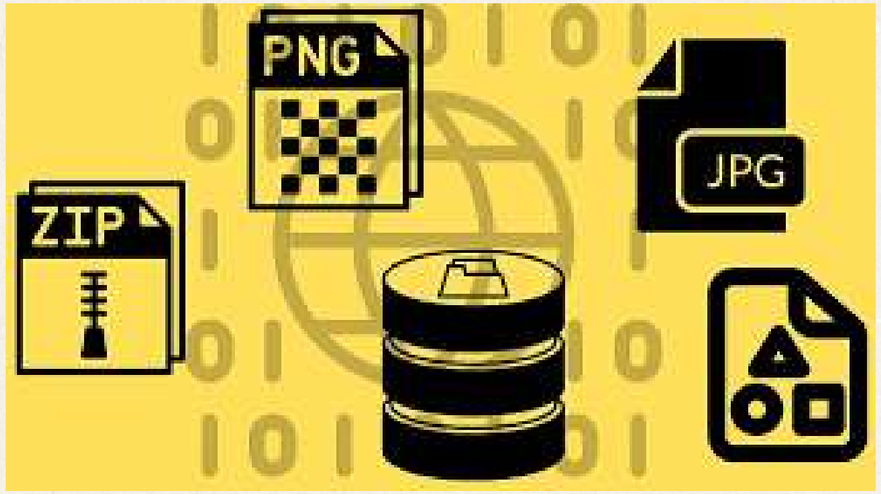
2023 yılı itibarıyla Türkiye’de internet kullanım oranı %83,4 ile dünyada 37.sırada yer almıştır. (We Are Social 2023 Global Raporu). IOT (Nesnelerin İnterneti) teknolojisi gelişimiyle beraber internete erişim olanakları inanılmaz bir boyuta geldi.



www.medium.com

Veri Sıkıştırma Neden Gerekli ?

Dünya üzerinde üretilen verilerin %90'ı son 5 yılda oluştu ve birgün içinde üretilen toplam veri miktarı 2,5 milyar GB seviyelerine geldi. Ön görülemeyen veri artışı, veri depolama ve sıkıştırılmış veri algoritmalarının hayati önemini ortaya çıkardı.



Veri sirkülasyonunun bu boyutlarda ve hızda olması, dosya transferlerinde verilerin mümkün olduğunca küçük olmasıyla sağlanabilmektedir.

Sıkıştırma algoritmaları bu noktada devreye girer.

Veri Sıkıştırma Algoritmaları

Kayıplı mı Kayıpsız mı?
İlk ve en kritik karar budur.

🔹 Kayıpsız (Lossless)
- Orijinal veri birebir geri elde edilir
- Metin, kaynak kodu, veri tabanı için zorunludur
- Örnekler:
  - RLE
  - Huffman
  - LZW
  - ZIP, PNG

🔹 Kayıplı (Lossy)
- Bir miktar bilgi bilinçli olarak atılır
- İnsan algısının fark edemeyeceği veriler silinir
- Görsel / ses için uygundur
- Örnekler:
  - JPEG
  - MP3
  - MP4

📌 Metin ve sayısal veriler → Kayıpsız şart

Veri Sıkıştırma Algoritmaları

Önemi :

Sıkıştırma Algoritmalarının Önemi
Depolama alanından tasarruf sağlar
İnternet üzerinden veri aktarımını hızlandırır
Bant genişliği maliyetini düşürür
Mobil ve gömülü sistemlerde performansı artırır

Sonuçları :

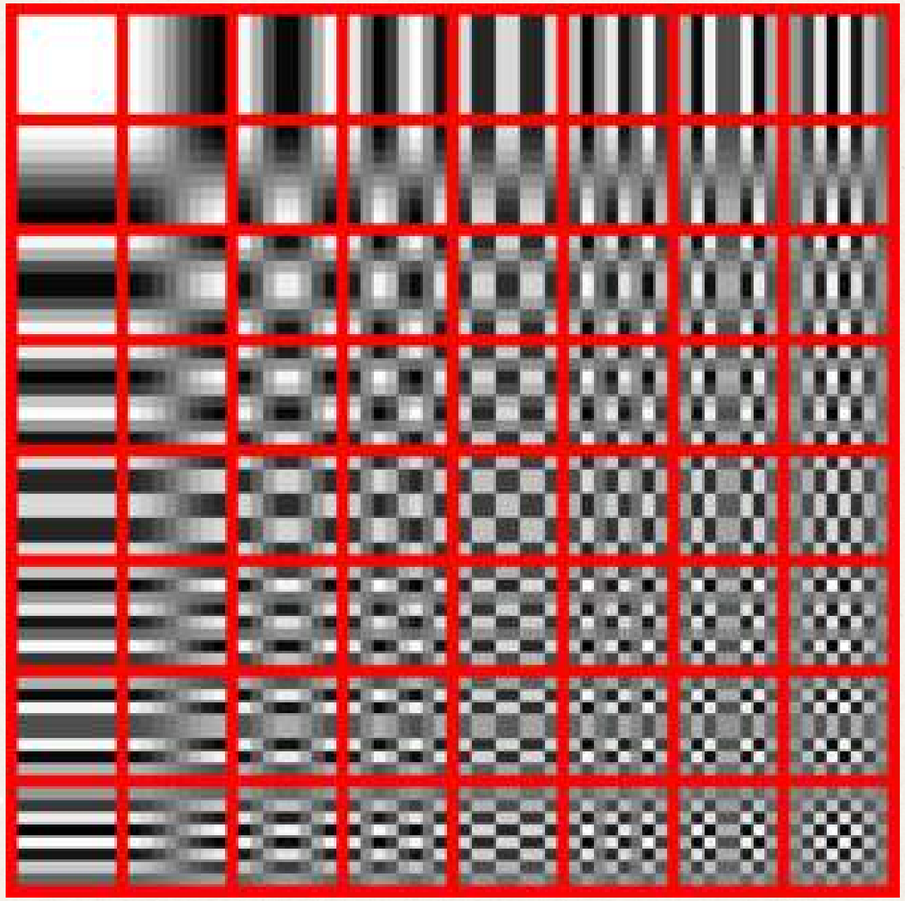
Veri sıkıştırma algoritmaları, modern bilgisayar sistemlerinin vazgeçilmez bir parçasıdır. Kullanılacak algoritma; verinin türüne, hız gereksinimine ve kayıplı ya da kayıpsız olma durumuna göre seçilmelidir. Doğru algoritma seçimi, hem performans hem de verimlilik açısından büyük önem taşır.

Kayıplı Sıkıştırma Algoritmaları

Genellikle ses ve görüntü uygulamalarında kullanılır. Çok büyük ses ve görüntü dosyalarında çok iyi sıkıştırma yapar ve kullanıcı kalitedeki azalmayı fark etmez.

Ayrık Kosinüs Dönüşümü

Ayrık Kosinüs Dönüşümü (DCT), farklı frekanslarda dalgalanan bir kosinüs fonksiyonunun toplamı açısından sınırlı bir veri noktası dizisidir. JPEG, HEIF, J2K, EXIF ve DNG gibi dijital görüntüler dahil olmak üzere çoğu dijital ortamda kullanılır.



Dalgacık Sıkıştırma

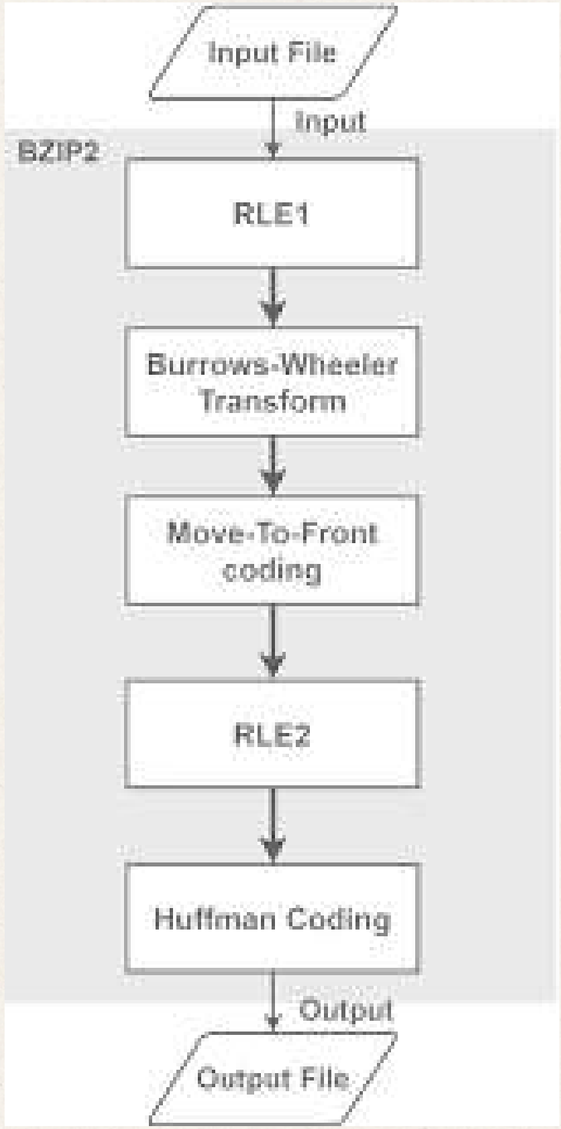
Dalgacık sıkıştırma, en çok görüntü sıkıştırmasında kullanılan kayıplı bir sıkıştırma algoritmasıdır. Bu algoritma, bir dalgacık dönüşümünün başlangıçta uygulandığı dönüşüm kodlaması adı verilen bir prensip kullanır. Bu, görüntüde piksel olduğu kadar katsayılar oluşturur. Bilgi istatistiksel olarak sadece birkaç katsayıyla konsantre edildiğinden, bu katsayılar daha kolay sıkıştırılabilir. Dikkate değer uygulamalar hareketsiz görüntüler için JPEG 2000, DJVU ve ECW'dir.

Kayıpsız Sıkıştırma Algoritmaları

Veri sıkıştırıldıktan sonra orijinaline çevrilirken kayıp olmayan algoritmalardır.

1. BZIP2 Algoritması :

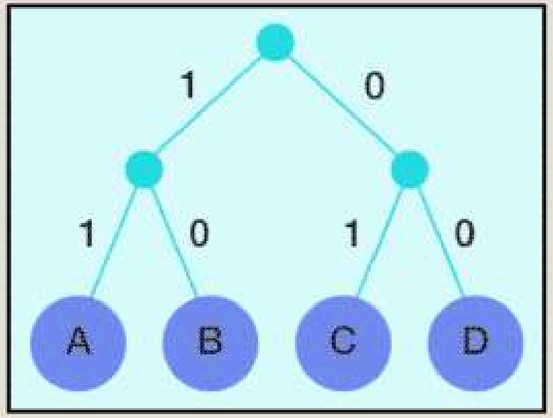
Bu algoritma, verileri sıkıştırmak için RLE ve Huffman kodlamalı Burrows-Wheeler algoritmasını kullanır. Dosyaları yalnızca arşivlemeden sıkıştırmak için kullanılır. Sıkıştırılmış dosyalar genellikle .bz2 uzantısı ile kaydedilir.



Kayıpsız Sıkıştırma Algoritmaları

2.Huffman Kodlama :

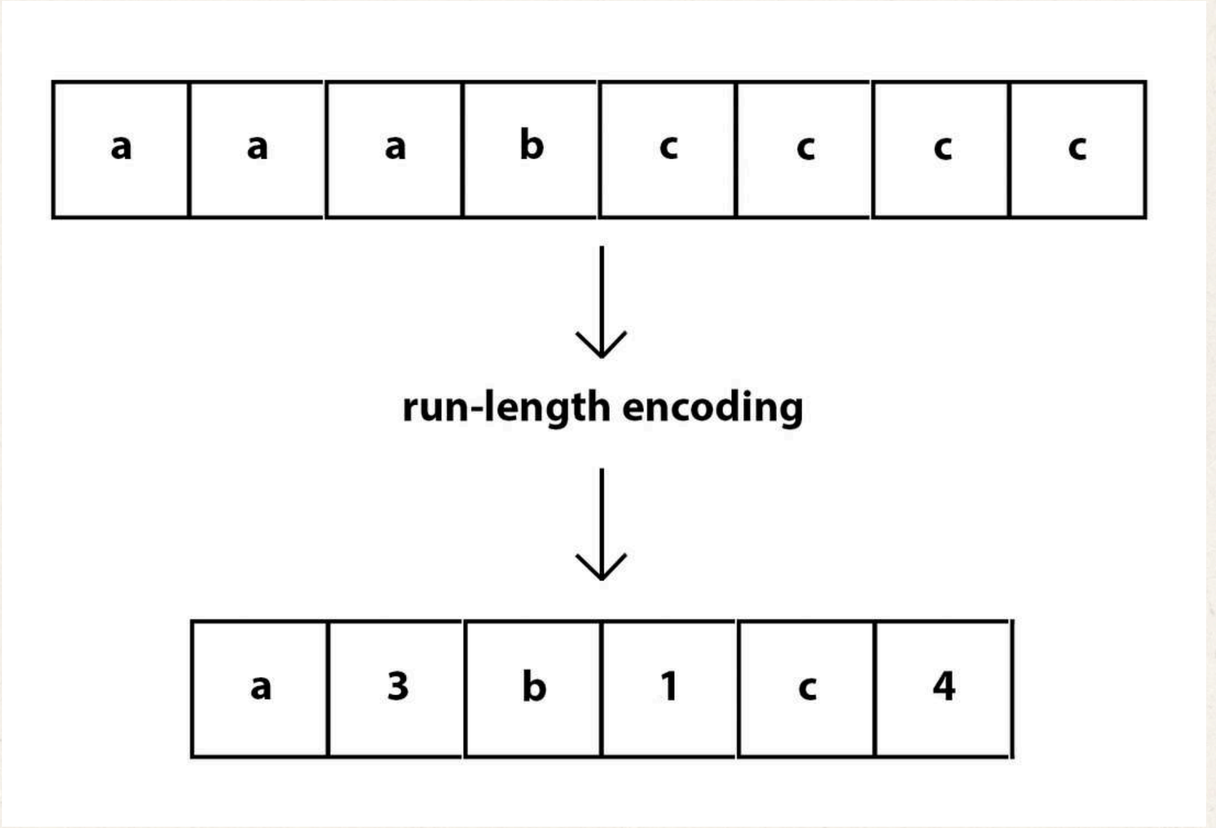
Kayıpsız (lossless) olarak veriyi sıkıştırıp tekrar açmak için kullanılır. Huffman kodlamasının en büyük avantajlarından birisi kullanılan karakterlerin frekanslarına göre bir kodlama yapması ve bu sayede sık kullanılan karakterlerin daha az, nadir kullanılan karakterlerin ise daha fazla yer kaplamasını sağlamasıdır.



Kayıpsız Sıkıştırma Algoritmaları

3. Çalışma uzunluğu kodlama (RLE) :

Bu algoritma, birçok bitişik veri öğesinde meydana gelen aynı veri değerini içeren sekanslara dayanan RLE kayıpsız sıkıştırma algoritması olarak da bilinir. Bu dizilere çalışma denir. RLE, her çalışmayı tek bir veri değeri ve sayımı olarak sakladı. Bu, basit grafik görüntüler gibi birçok çalışma içeren verilerde yararlıdır, ör. Çizimler, simgeler, çizgiler ve animasyonlar.



Kayıpsız Sıkıştırma Algoritmaları

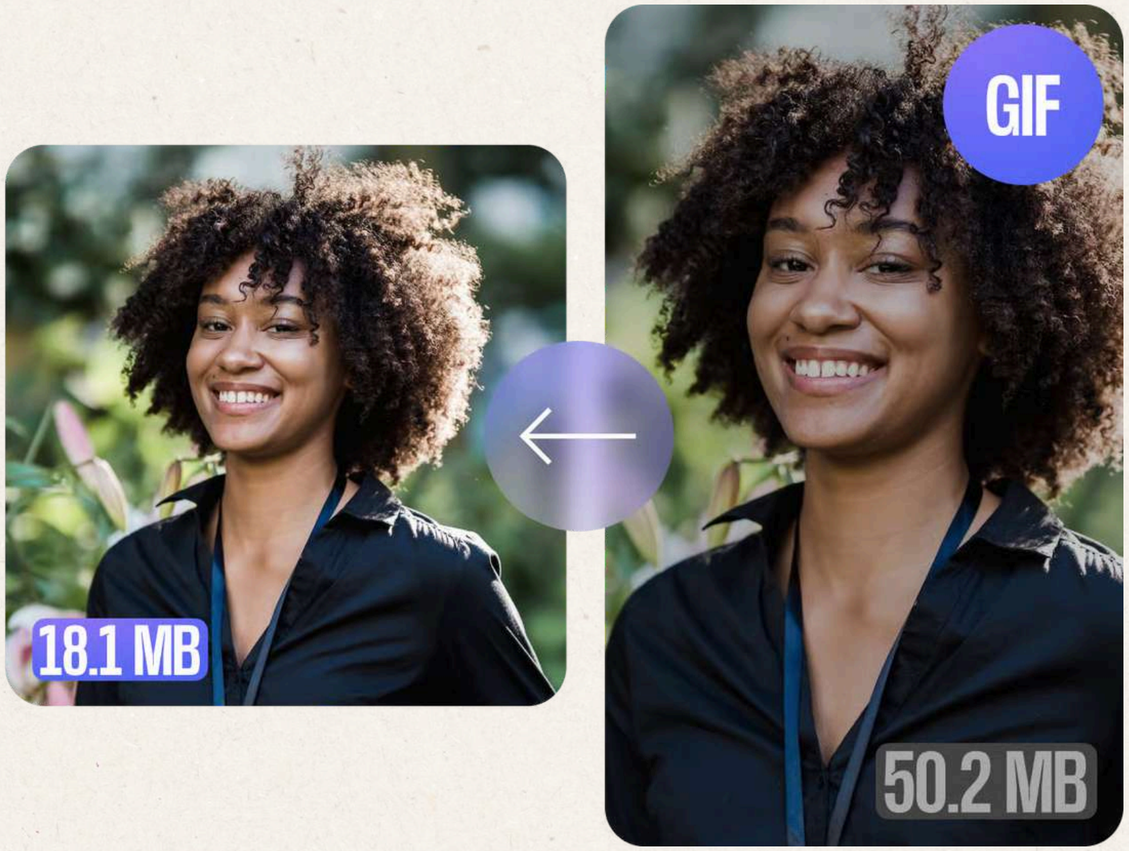
4.Lempel-ziv sıkıştırma

Lempel–Ziv–Welch (LZW), Abraham Lempel , Jacob Ziv ve Terry Welch tarafından oluşturulan evrensel bir kayıpsız veri sıkıştırma algoritmasıdır . 1978'de Lempel ve Ziv tarafından yayınlanan LZ78 algoritmasının geliştirilmiş bir uygulaması olarak Welch tarafından 1984'te yayınlandı.

Algoritmanın uygulanması basittir ve donanım uygulamalarında çok yüksek verim potansiyeline sahiptir. Unix dosya sıkıştırma yardımcı programının sıkıştırma algoritmasıdır ve GIF görüntü formatında kullanılır.

Kayıpsız Sıkıştırma Algoritmaları

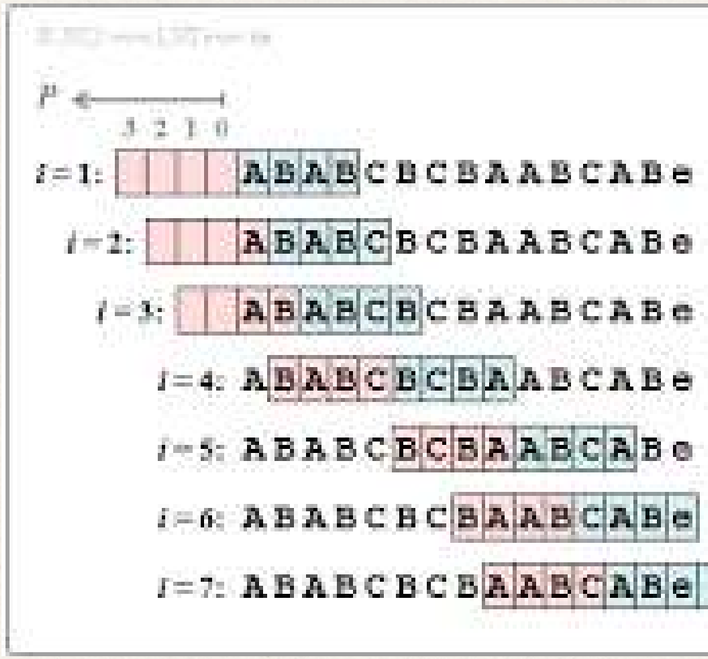
4.Lempel-ziv sıkıştırma



Kayıpsız Sıkıştırma Algoritmaları

4.Lempel-ziv sıkıştırma

Index *I* decimal	Index *I* binary	Content	Entry in step	LZ78 encoding output formally	LZ78 encoding output binary
0	0000	ε (empty)	$i = 0$	—	—
1	0001	A	$i = 1$	(0, A)	000000
2	0010	B	$i = 2$	(0, B)	000001
3	0011	AB	$i = 3$	(1, B)	000101
4	0100	C	$i = 4$	(0, C)	000010
5	0101	BC	$i = 5$	(2, C)	001010
6	0110	BA	$i = 6$	(2, A)	001000
7	0111	ABC	$i = 7$	(3, C)	001110
8	1000	ABe	$i = 8$	(3, e)	001111
9	1001	· · · · ·			

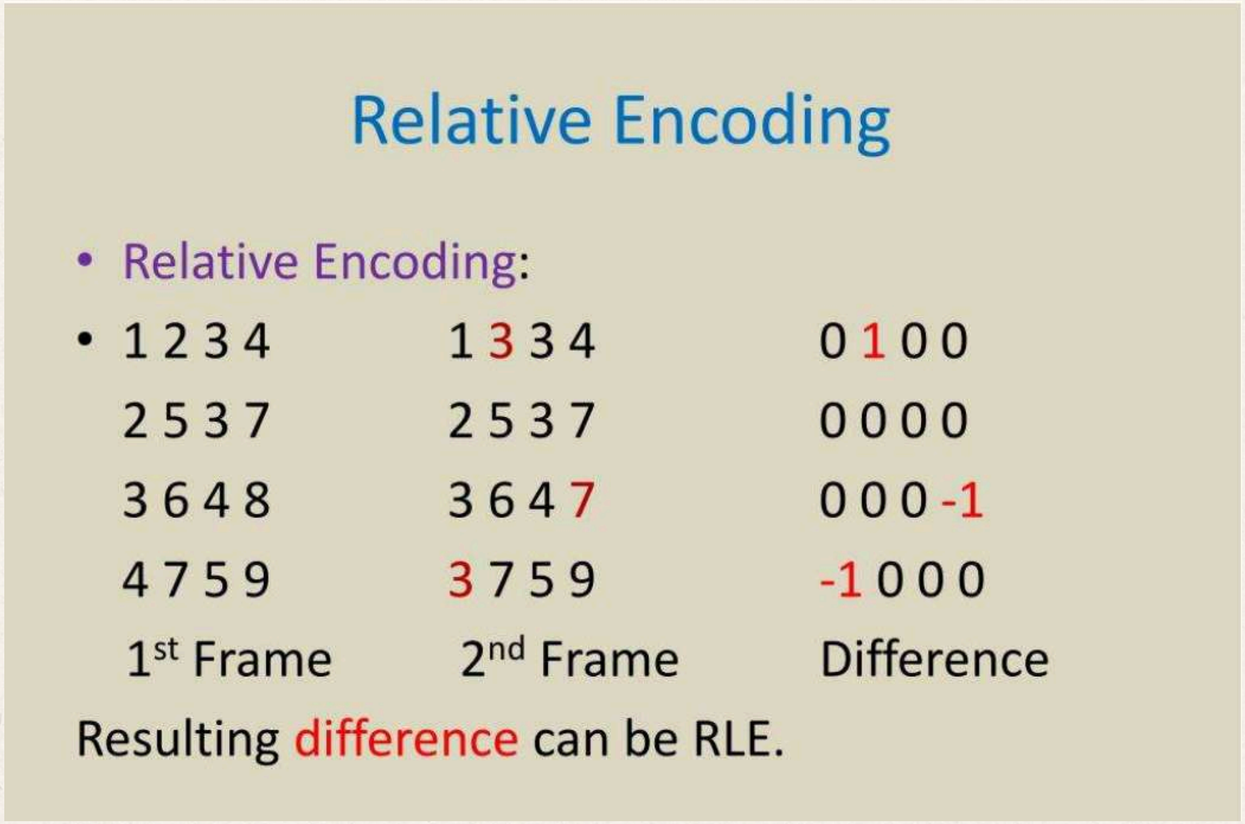


LZ77 encoding output formally	LZ77 encoding output binary
(0, 0, A)	0000000
(0, 0, B)	0000001
(1, 2, C)	0101010
(1, 3, A)	0101100
(0, 1, B)	0000101
(0, 0, C)	0000010
(2, 2, e)	1001011

Kayıpsız Sıkıştırma Algoritmaları

4. Relative Encoding (Göreli Kodlama) :

bir veriyi mutlak değerleriyle değil, bir önceki değere göre farklarını kodlama yöntemidir.



Kaynaklar

1)
https://blog.fileformat.com/tr/compression/lossy-and-lossless-compression-algorithms/

2)
https://www.clicksus.com/we-are-social-2023-global-ve-turkiye-raporu#:~:text=Dijital%202023%3A%20%C3%9Clkelere%20G%C3%B6re%20%C4%B0nternet%20Kullan%C4%B1m%C4%B1&text=T%C3%BCrkiye%202023%20y%C4%B1l%C4%B1%20itibariyle%20internet,d%C3%BCnya%20s%C4%B1ralamas%C4%B1na%2037.%20s%C4%B1rada%20bulunuyor.

Dinlediğiniz için teşekkürler