Veri Depolama ve Sıkıştırma Algoritmaları

Bu sunum, dijital dünyanın temelini oluşturan bilginin bit düzeyinde nasıl temsil edildiğini ve modern bilişimin vazgeçilmezi olan veri sıkıştırma tekniklerini derinlemesine incelemektedir. Metin, resim ve ses gibi farklı veri türlerinin bilgisayar ortamında nasıl ikili sayılara dönüştürüldüğünü ve bu büyük veri setlerini daha verimli bir şekilde depolamak ve iletmek için neden sıkıştırma yöntemlerine ihtiyaç duyulduğunu keşfedeceğiz. Ayrıca, Run-Length Encoding (RLE) gibi temel sıkıştırma algoritmalarının çalışma mantığını ve gerçek dünyadaki uygulamalarını detaylı bir şekilde ele alacağız.

Bitler ve Bilginin Temeli

Dijital dünyada her şey bitler aracılığıyla temsil edilir. Bir bit (binary digit), bilgisayarların anlayabileceği en küçük bilgi birimidir ve yalnızca iki durumu temsil edebilir: O veya 1. Bu ikili sistem, elektrik sinyallerinin açık/kapalı durumu veya manyetik kutupların yönü gibi fiziksel karşılıklara sahiptir. Sekiz bit bir araya gelerek bir baytı oluşturur ve bir bayt 2^8 = 256 farklı değeri temsil edebilir.

Metin Gösterimi (Text Representation)

Metin gösterimi, her harf veya sembolün bilgisayar tarafından saklanabilmesi ve işlenebilmesi için 0 ve 1'lerden oluşan bit dizilerine dönüştürülmesi işlemidir.

ASCII, İngilizce harfleri ve temel sembolleri 7 veya 8 bit kullanarak temsil eder.

Unicode (UTF-8) ise değişken uzunlukta bitler kullanarak çok dilli metinlerin doğru şekilde gösterilmesini sağlar.

Bu kodlamalar **sayesinde metinler dosyalar içinde doğru biçimde saklanabilir ve okunabilir**.

Örneğin, 'A' harfi ASCII' de 01000001 olarak temsil edilirken, 'B' harfi 01000010 olarak temsil edilir. Daha geniş karakter setlerini desteklemek için Unicode ve UTF-8 gibi kodlama standartları geliştirilmiştir. UTF-8, fark dillerdeki karakterleri ve sembolleri kapsayan milyarlarca farkli karakteri temsil edebilen değişken uzunluklu bir kodlama sistemidir. Her bir metin belgesi, bu kodlama standartlan sayesinde bit dizilerine dönüştürülerek depolanır ve işlenir. Bu, bilgisayarların metinleri okumasını, yazmasını ve düzenlemesini mümkün klar.

Converting the text "hope" into binary

Characters:	h	o	p	e
ASCII Values:	104	111	112	101
Binary Values:	01101000	01101111	01110000	01100101
Bits:	8	8	8	8

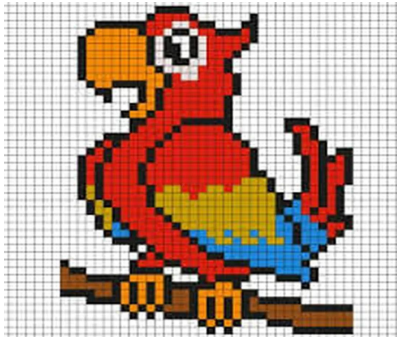
Resim Verilerinin Bit Temsili

Dijital resimler, pikseller adı verilen küçük renkli noktalardan oluşur. Her pikselin rengi, belirli bir bit dizisiyle temsil edilir. En yaygın renk temsil modellerinden biri RGB (Kırmızı, Yeşil, Mavi) modelidir. Bu modelde, her bir pikselin rengi, kırmızı, yeşil ve mavi bileşenlerinin yoğunluğuyla belirlenir.

Her bir renk bileşeni (Kırmızı, Yeşil, Mavi) genellikle 8 bit ile temsil edilir, bu da her bir bileşen için 256 farklı yoğunluk seviyesi olduğu anlamına gelir. Böylece, bir pikselin rengi toplamda 24 bit (8 bit Kırmızı + 8 bit Yeşil + 8 bit Mavi) kullanılarak temsil edilir ve bu da 2^24 (yaklaşık 16.7 milyon) farklı rengin oluşturulmasını sağlar. Bu derinlik, "gerçek renk olarak adlandırılır.

Bir resmin çözünürlüğü (genişlik x yükseklik), piksel sayısını belirler. Örneğin, 1920×1080 çözünürlüğündeki bir resim, 2.073.600 piksel içerir. Her pikselin renk bilgisi bitlere dönüştürüldüğünde, büyük bir resim dosyasının neden bu kadar çok depolama alanı kapladığı anlaşılır. Dijital kameralar ve tarayıcılar, optik bilgiyi elektrik sinyallerine ve ardından bit dizilerine dönüştürerek bu süreci gerçekleştirir.

Piksel (Bitmap) Görsel Örneği:



Görsel, küçük karelerden (piksel) oluşur.
Her piksel farklı bir rengi temsil eder.
Renkler harflerle kodlanabilir:
R = Kırmızı, Y = Sarı, B = Mavi, K = Siyah, W = Beyaz, Br = Kahverengi

Her renk, bilgisayarda saklanmak üzere özel bir sayıyla temsil edilir. Genellikle bu sayı RGB kodu ya da basitleştirilmiş bir numara olur.

Görsel, bir renk matrisi olarak dijital ortamda saklanabilir.
Aynı renklerin tekrarını sayarak RLE sıkıştırma yöntemiyle daha az yer kaplaması sağlanabilir.

Ses Verilerinin Bit Temsili

Ses verileri bilgisayarlarda bitler halinde temsil edilir. Bu işlem analog-dijital dönüşüm (ADC) ile gerçekleştirilir. ADC süreci, ses dalgasından örnekler alınması ve bu örneklerin dijital değerlere dönüştürülmesini kapsar. Böylece ses, 0 ve 1'lerden oluşan bit dizileri şeklinde saklanabilir. Sesin temsilinde kalite arttıkça veri miktarı da artar, bu nedenle ses verileri genellikle sıkıştırma teknikleriyle daha küçük boyutlarda saklanır(MP3 gibi)

Neden Veri Sıkıştırmaya İhtiyaç Duyarız?

Günumür nijital çağında veri üretimi inaminaz boyutlara ulaşmıştır. Yüksek çhziiniriozu fotoğraflar, 4K videolar, büyük yazılım paketteri ve karmaş veritabanlan, depolama ve iletim kaynakları üzerinde büyük bir yük oluşturmaktadır. Bu noktada veri sikiştima, vazgeçimez bir teknoloji ha ine gelmiştir.

1. Depolama Alanından Tasarruf

Sıkıştırılmamış veriler, sabit disklerde, SSD'lerde ve bulut depolama hizmetlerinde hızla yer kaplar. Sıkıştırma, ayni miktarda verinin daha az fiziksel alanda depolanmasını sağlayarak maliyetleri düşürür ve depolama kapasitesini optimize eder.

2.Daha Hızlı Veri İletimi

İnternet bant genişliği ve ağ hızları artsa da, büyük dosyaların iletimi hala zaman alıcı olabilir. Sıkıştırılmış dosyalar, ağ üzerinden daha hızlı transfer edilebilir, bu da indirme sürelerini kısaltır ve gerçek zamanlı uygulamaların (video akışı gibi) performansını artırır.

3.Arşivleme ve Yedekleme

Uzun süreli depolama ve yedekleme stratejilerinde sıkıştırma, veri bütünlüğünü korurken depolama maliyetlerini düşürmek için kullanılır. Eski veya nadiren erişilen veriler genellikle sıkıştırılarak arşivlenir.

4.Bant Genişliği Kullanımını Azaltma

Sıkıştırma, özellikle sınırlı bant genişliğine sahip mobil ağlar veya uzak bağlantılar için kritik öneme sahiptir. Daha az veri gönderilmesi, ağ tıkanıklığını azaltır ve daha verimli bir ağ kullanımı sağlar.

Sıkıştırma Mantıklarına Giriş

Veri sıkıştırma algoritmaları, temelde verilerdeki tekrarları veya gereksiz bilgileri tespit ederek bunları daha kısa bir formda temsil etme prensibine dayanır.

Sıkıştırma algoritmaları, verilerin iç yapısındaki istatistiksel düzenlilikleri, tekrarları veya anlamsal ilişkileri kullanarak verimli kodlamalar oluşturur. Her algoritmanın kendine özgü bir çalışma prensibi ve farklı veri türleri için optimize edilmiş performans karakteristikleri vardır.

İki ana sıkıştırma türü vardır:

1.Kayıpsız Sıkıştırma (Lossless Compression)

Bu yöntemlerde, sıkıştırma sonrası veriler, orijinal verilerle tamamen aynıdır. Hiçbir bilgi kaybı yaşanmaz. Metin belgeleri, program kodları veya veri tabanı kayıtları gibi verilerin mutlak doğruluğunun kritik olduğu durumlarda kullanılır. Örnekler: ZIP, PNG, GIF, RLE.

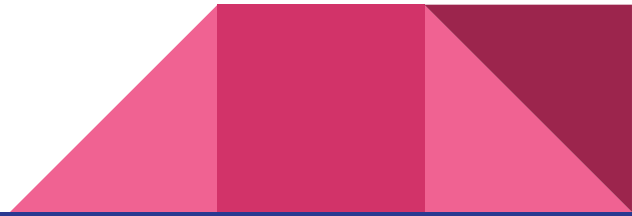
Kayıplı Sıkıştırma (Lossy Compression)

Bu yöntemlerde, sıkıştırma sırasında bazı önemsiz veya insan algısı için fark edilemez bilgiler kalıcı olarak atılır. Bu, çok daha yüksek sıkıştırma oranları sağlar ancak orijinal veriye tam olarak geri dönülemez. Özellikle multimedya verileri (resim, ses, video) için yaygın olarak kullanılır. Örnekler: JPEG, MP3, MPEG.

Veri sıkıştırma neden gereklidir?

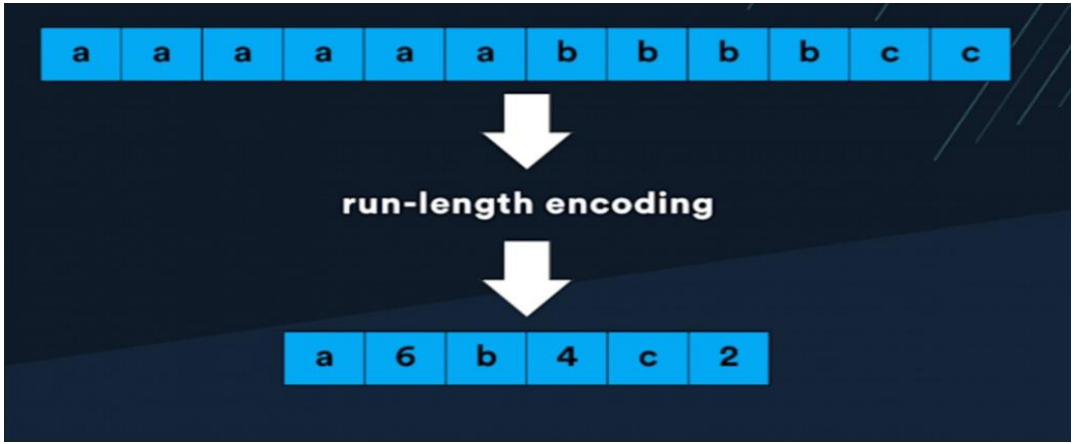
Veri sıkıştırma, depolama alanından tasarruf etmemizi, veri iletimini hızlandırmamızı ve ağ bant genişliğini daha verimli kullanmamızı sağlar.

Yüksek çözünürlüklü multimedya ve karmaşık veriler, önemli depolama alanı ve bant genişliği gerektirir. Bu durum, veri sıkıştırmayı vazgeçilmez kılar.



Veri sıkıştırmanın en önemli yöntemlerinden biri RLE(Run Length Encoding)dir.

RLE, kullanılan en basit sıkıştırma algoritmalarından biridir.



(RLE) Ne demek?

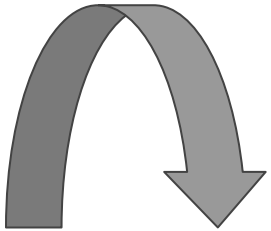
Run-Length Encoding, ardışık olarak tekrar eden verileri, verinin kendisi ve tekrar sayısı ile temsil eden basit bir veri sıkıştırma yöntemidir.

- Örneğin, "AAAAABBBCCDAA" gibi bir veri dizisinde, RLE bu diziyi "5A3B2C1D2A" olarak sıkıştırabilir. Burada, '5A' beş adet 'A' karakterini, '3B' üç adet 'B' karakterini temsil eder. Bu yöntem, özellikle bitmap resimler veya basit animasyonlar gibi yoğun tekrar eden desenlere sahip verilerde oldukça etkilidir.

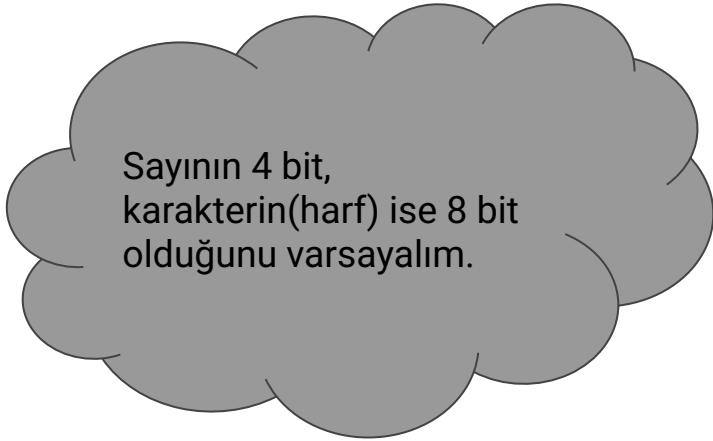
- RLE kullanıldığında, tekrar eden karakterlerin ve sayıların temsil edilmesiyle veri boyutu azaltılır.

RLE’nin boyutu nasıl azalttığına dair bir örnek:

[aaaabbfcc] bunlar(9h)×8=72bit



[4a 2b 1f 2c] (4h×8)+(4×4)=48bit



Burada sıkıştırma sonrasında hacmin azaldığını gözlemliyoruz.

Peki sıkıştırma oranını(%) nasıl hesaplamadır?

$$\left(1 - \frac{\text{Sıkıştırma sonrası boyut}}{\text{Sıkıştırma öncesi boyut}}\right) \times 100$$

$$\left(1 - \frac{48}{72}\right) \times 100. = 33.33\%$$

Ancak tekrar olmaması durumunda ne olurdu?

Veride çok fazla tekrar yoksa veya veri tamamen rastgele ise, RLE sıkıştırma oranını artırmak yerine dosya boyutunu artırabilir, çünkü her tekil karakter için bile bir '1' sayısıyla kodlama yapmak gerekir!!

Örnek

[abcdefgh] 8seyı×8bit =64 bit

[1a 1b 1c 1d 1e 1f 1g 1h] (8sayı×8bit)+(8harf×4bit)=96

Bu örnekte, boyutun arttığını görüyoruz.

Neden RLE kullanıyoruz?

- Veri boyutunu küçültmek için.

- Verinin aktarımını veya depolanmasını hızlandırmak için.

- Çünkü daha karmaşık sıkıştırma algoritmalarına göre basit ve hızlıdır.

Ne zaman etkili olur?

- Verilerde büyük ardışık tekrarlar olduğunda, örneğin:
  - Tek renkli görüntüler.
  - Tekrarlayan ikili (0 ve 1) veriler.
  - Tekrarlayan harfler içeren metinler.

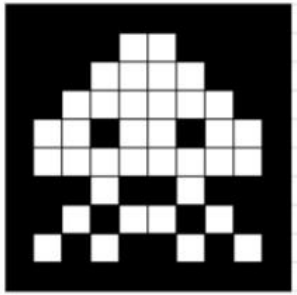
En yaygın kullanımları

En yaygın kullanımları:

1. Bitmap görüntüler gibi:
   - BMP veya TIFF formatları.
2. Faks cihazlarındaki eski görüntüler
   - Tekrarlayan siyah çizgilerin sıkıştırılması için.
3. Basit metin dosyaları
   - İçinde tekrar eden basit verilerin saklanması için.
4. Basit dijital görüntüleme sistemleri
   - Hesap makineleri veya eski cihazlar gibi.

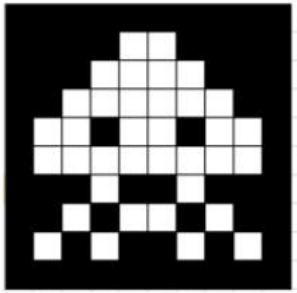
RLE algoritması ile bitmap görüntü sıkıştırma örneği

Sıkıştırılmamış görüntü



1111111111
1111001111
1110000111
1100000011
1001001001
1000000001
1110110111
1101001011
1010110101
1111111111

Görüntü RLE ile sıkıştırılmıştır.



10 1
4 1 2 0 4 1
3 1 4 0 3 1
2 1 6 0 2 1
1 1 2 0 1 1 2 0 1 1 2 0 1 1
1 1 8 0 1 1
3 1 1 0 2 1 1 0 3 1
2 1 1 0 1 1 2 0 1 1 1 0 2 1
1 1 1 0 1 1 1 0 2 1 1 0 1 1 1 0 1 1
10 1

RLE Gibi Temel Mantıklar

Run-Length Encoding IRLE) gibi basit kayıpsız algoritmalar, tekrarlayan veri dinlerini daha kısa formda temsil ederek verirsi sıkıştırma sağlar ve belei alanlarda hata kullarsimaktadır

Kaynaklar

MrDoranComputing.com .
Bilgisayar Bilimine Giriş kitabı .
Yazarlar: J. Glenn Brookshear, Dennis Brylow.
W3Schools Online Web Tutorials.

Dinlediğiniz için teşekkür ederim.

FATIMA EIDOU Prof.Dr.TURGAY BiLGiN