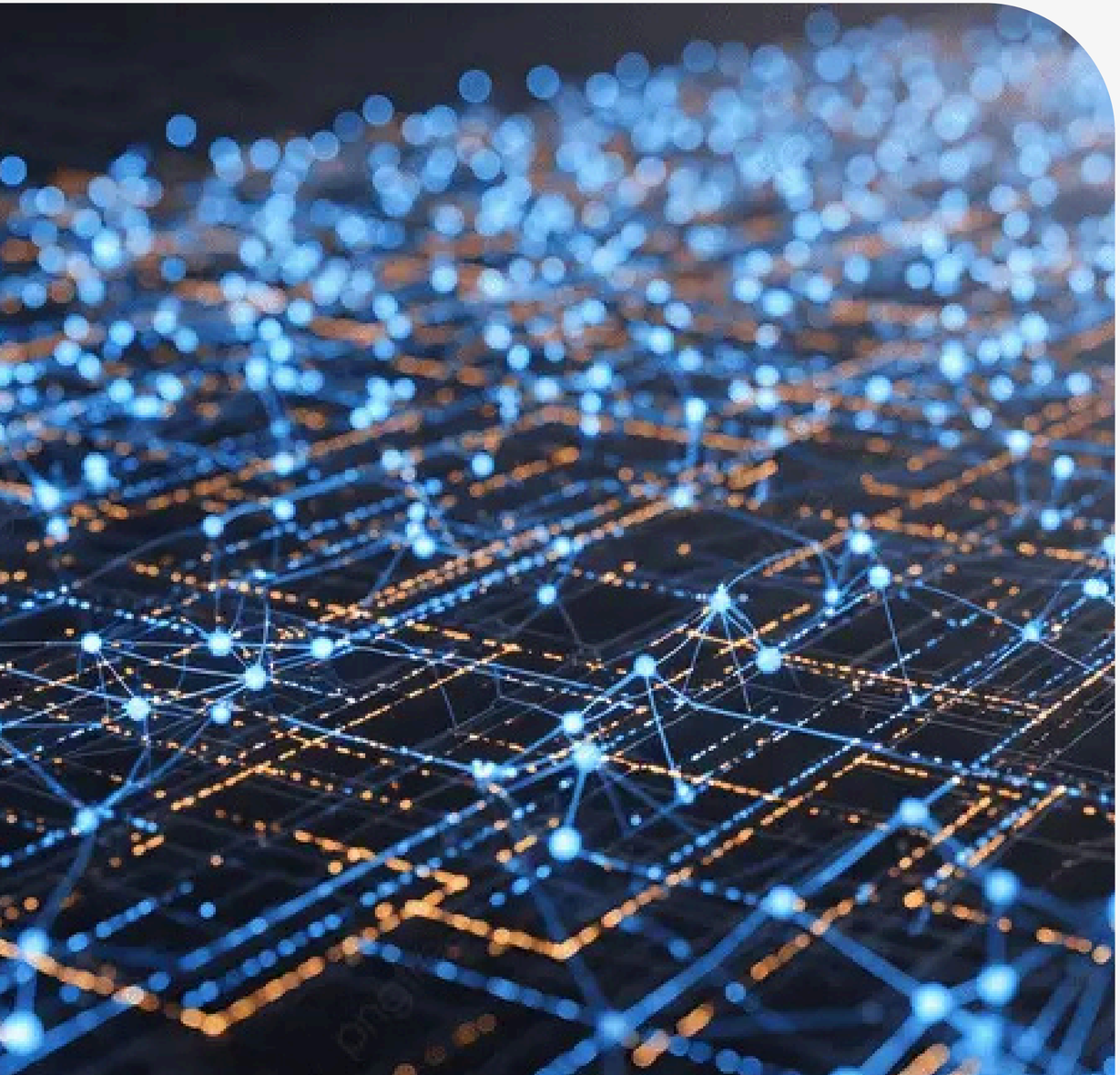
Bilgisayar Mühendisliğine
Giriş Dersi Dönem Projesi

Veri Depolama ve Sıkıştırma Algoritmaları

VELİ EMRE ERSOY 24360859076

İçindekiler

Veri Temsili Nedir ?



Veri temsili, gerçek dünyadaki bilgilerin (metin, sayı, resim, ses vb.) bilgisayar tarafından anlaşılabilir ve işlenebilir hale getirilmesi sürecidir.

Bilgisayarlar insan gibi harfleri veya sesleri doğrudan anlayamaz.
Bu nedenle tüm veriler sayılara, sayılar da bitlere (0 ve 1) dönüştürülür.

Cloud
PC
Mobile
Remote Server
Notebook
Tablet PC
Database
Remote

Bit Kavramı

- Bit (Binary Digit), bilgisayarda kullanılan en küçük veri birimidir.
- Bir bit yalnızca iki değer alabilir: 0 ve 1.
- Bu değerler, donanımda elektrik sinyalinin yokluğu (0) veya varlığı (1) ile temsil edilir.
- Bilgisayarlar bu nedenle ikilik (binary) sayı sistemini kullanır.
- 8 bit = 1 byte
- Veriler bellek üzerinde byte’lar halinde saklanır.

“Bit, bilgisayarda bilginin temel yapı taşıdır ve veri sıkıştırma, kullanılan bit sayısını azaltmayı hedefler.”

Metin Verisinin Temsili

- **Bilgisayarda metinler karakter dizileri (string) olarak saklanır.**
- **Her karakter, önceden tanımlanmış bir sayısal kod ile temsil edilir.**
- **Bu sayısal kodlar ikilik (binary) sisteme çevrilerek bellekte tutulur.**
- **Metin kodlama sistemleri: ASCII ve Unicode**

“Kullanılan karakter kodlama sistemi, veri boyutunu ve sıkıştırma verimini doğrudan etkiler.”

ASCII ve Unicode Arasındaki Fark

ASCII (American Standard Code for Information Interchange)

- İlk metin kodlama standartlarından biridir
- 7 bit kullanır ve 128 karakter içerir
- İngilizce harfler, rakamlar ve temel semboller bulunur
- Türkçe karakterleri desteklemez
- Daha basit ve az yer kaplar

Unicode

- Tüm dünya dillerini desteklemek için geliştirilmiştir
- Milyonlarca karakteri kapsar
- Türkçe, Arapça, Çince, emoji gibi karakterleri içerir
- UTF-8, UTF-16 gibi farklı kodlama biçimleri vardır
- Unicode daha kapsamlı ve evrenseldir

Metin (Karakter) Verisinin Temsili

- Metin verileri, bilgisayarda karakter dizileri (string) olarak saklanır.
- Her karakter, bir karakter kümesine (character set) ait olacak şekilde temsil edilir.
- Bu karakter kümeleri, her karaktere benzersiz bir sayısal değer atar.
- Her karakter bellekte belirli sayıda bit kullanır.
- Metin verisinin bilgisayarda doğru görüntülenmesi:
- Doğru karakter kodlamasına
- Uygun bit uzunluğuna bağlıdır

Resim Verisinin Temsili

- Dijital resimler, piksellerden (picture element) oluşur.
- Her piksel, resimdeki en küçük görsel birimi temsil eder.
- Her pikselin bir renk değeri vardır.
- Bu renk bilgisi sayısal olarak saklanır ve ikilik (binary) sisteme çevrilir.
- Resmin dosya boyutu:

  Piksel sayısına (çözünürlük)

  Her piksel için kullanılan bit miktarına bağlıdır

  Yüksek çözünürlüklü ve renkli resimler,

  Daha fazla bit kullanır Daha fazla depolama alanı gerektirir

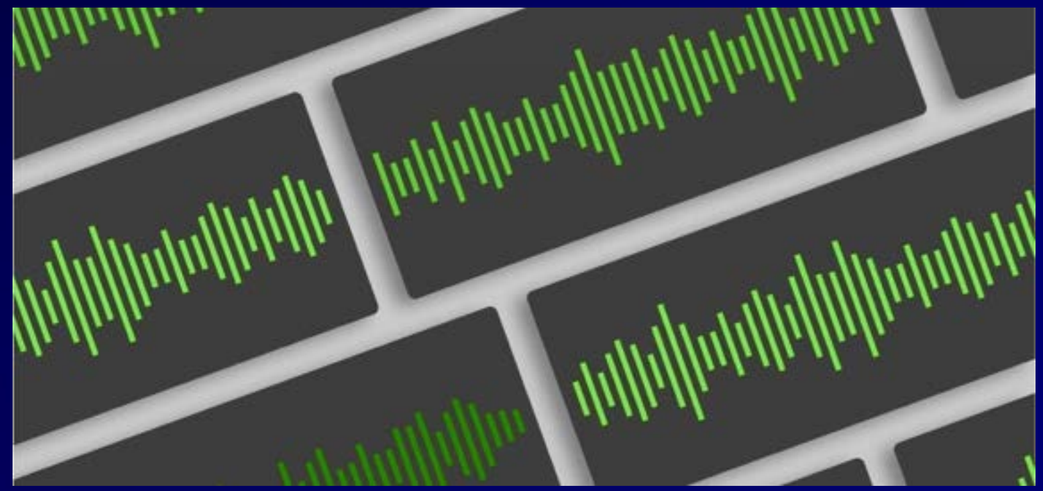
RGB Renk Modeli

- **Dijital resimlerde renkler RGB (Red, Green, Blue) modeli ile temsil edilir.**
- **Her renk, kırmızı, yeşil ve mavi bileşenlerinin birleşimiyle oluşur.**
- **Her RGB bileşeni sayısal bir değere sahiptir.**
- **Genellikle her bileşen 8 bit ile temsil edilir.**

“RGB renk modeli, modern dijital görüntüleme sistemlerinin temelini oluşturur.”

Ses Verisinin Bit Düzeyinde Temsili

- Ses, fiziksel olarak sürekli (analog) bir sinyaldir.
- Bilgisayarlarda sesler dijital olarak temsil edilir.
- Dijital ses elde etmek için örnekleme (sampling) yapılır.
- Örnekleme, ses sinyalinin belirli zaman aralıklarıyla ölçülmesidir.
- Örnekleme hızı Hz (Hertz) ile ifade edilir.
- Her örnek, belirli bir bit sayısı ile temsil edilir.
- Örnekleme hızı ve bit derinliği, dijital sesin kalitesini belirleyen temel faktörlerdir.

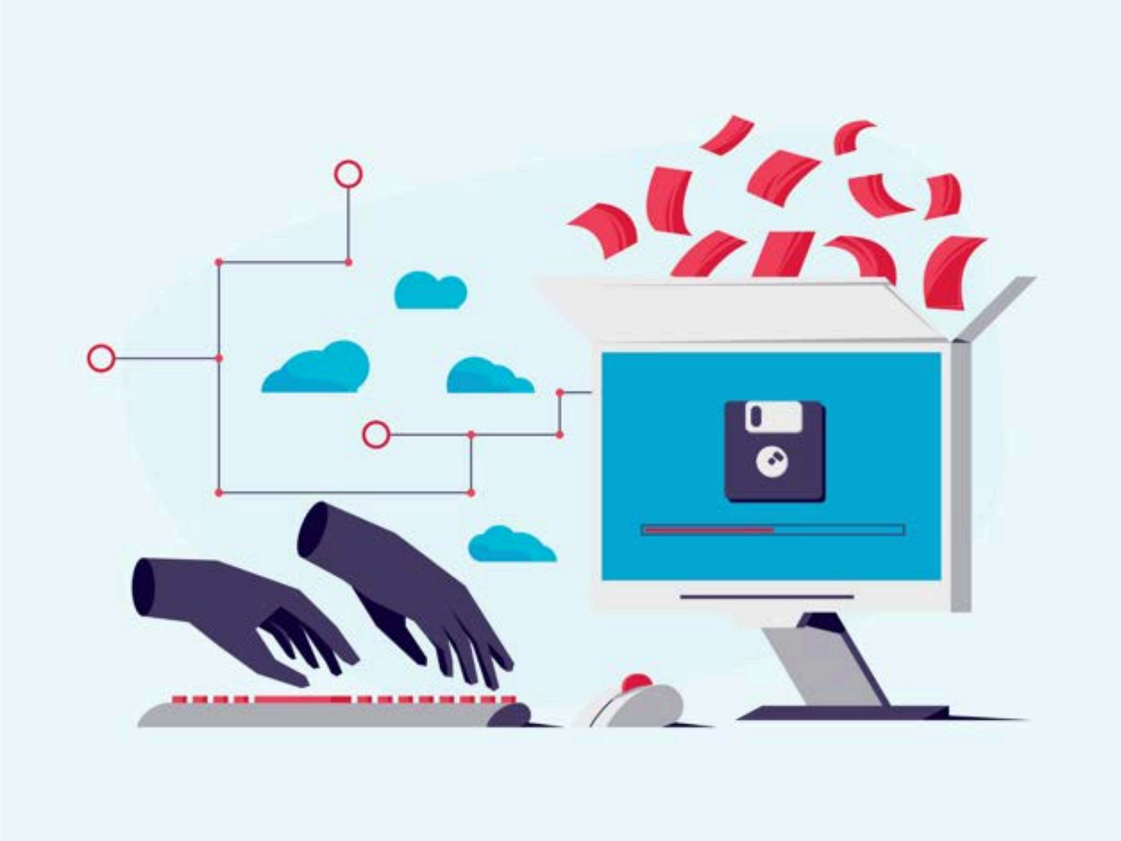


Veri Sıkıştırma (Data Compression)

- Veri sıkıştırma, dosya boyutunu azaltma işlemidir.
- Amaç, daha az depolama alanı kullanmak ve daha hızlı iletim sağlamaktır.
- Sıkıştırma, verideki tekrar eden veya gereksiz bilgilerin azaltılmasıyla yapılır.
- Veri sıkıştırma ikiye ayrılır:
- Kayıpsız (Lossless): Veri tamamen geri elde edilir
- Kayıplı (Lossy): İnsan tarafından fark edilmeyen bilgiler atılır

Neden Veri Sıkıştırma ?

- Dijital dünyada üretilen veri miktarı çok hızlı artmaktadır
- Büyük dosyalar:
- Daha fazla depolama alanı kaplar
- Daha uzun aktarılma süresi gerektirir
- Veri sıkıştırma sayesinde:
- Dosya boyutu küçülür
- İnternet üzerinden veri daha hızlı gönderilir
- Bant genişliği daha verimli kullanılır
- Özellikle şu alanlarda gereklidir:
- Web siteleri
- Mobil uygulamalar
- Ses ve video iletimi
- Sıkıştırma kullanılmazsa:
- Depolama maliyeti artar
- Sistem performansı düşer



Sıkıştırma Türleri

Veri sıkıştırma iki ana gruba ayrılır

Kayıpsız sıkıştırma:
Orijinal veri birebir geri elde edilir
Bilgi kaybı yoktur
Metin ve program dosyalarında kullanılır

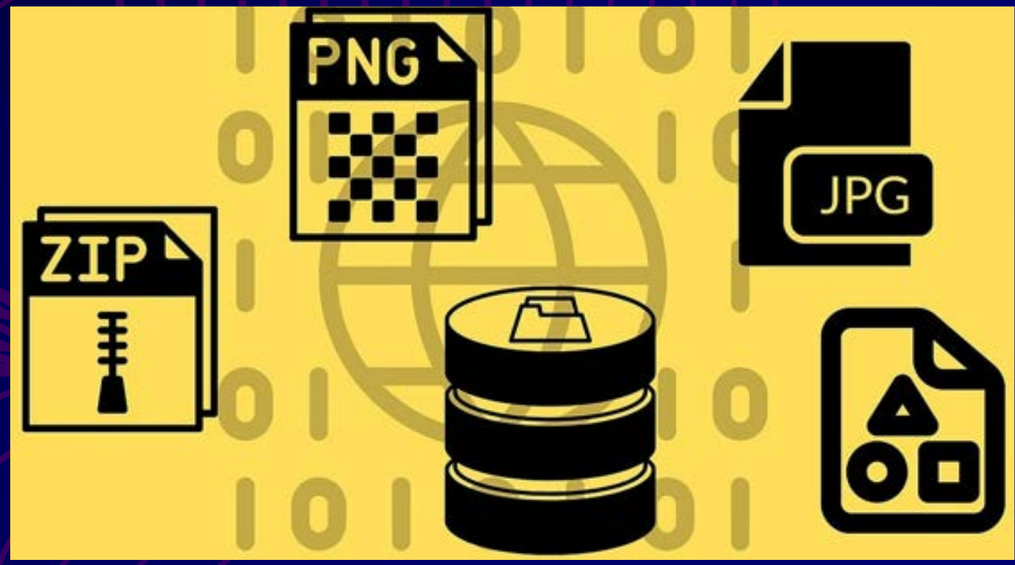
Kayıplı sıkıştırma:
Verinin bir kısmı bilinçli olarak silinir
İnsan tarafından fark edilmesi zor bilgiler atılır
Resim, ses ve video dosyalarında kullanılır

Sıkıştırma türü seçimi verinin önemine kullanım amacına göre yapılır

Kayıpsız Sıkıştırma (Lossless Compression)

- Sıkıştırılan veri eksiksiz geri elde edilir
- Hiçbir bilgi kaybı yoktur
- Özellikle metin verileri için uygundur
- Program dosyaları ve veri dosyalarında kullanılır
- Run-Length Encoding (RLE) bu gruba girer

Örnek:ZIP,RAR,GZIP,RLE



Kayıplı Sıkıştırma (Lossy Compression)

- Sıkıştırma sırasında bazı bilgiler silinir
- Orijinal veri tam olarak geri alınamaz
- İnsan tarafından fark edilmesi zor veriler atılır
- Daha yüksek sıkıştırma oranı sağlar
- Resim, ses ve video dosyalarında kullanılır

Örnek: JPEG, MP3, MP4

RLE Nedir? (Run-Length Encoding)

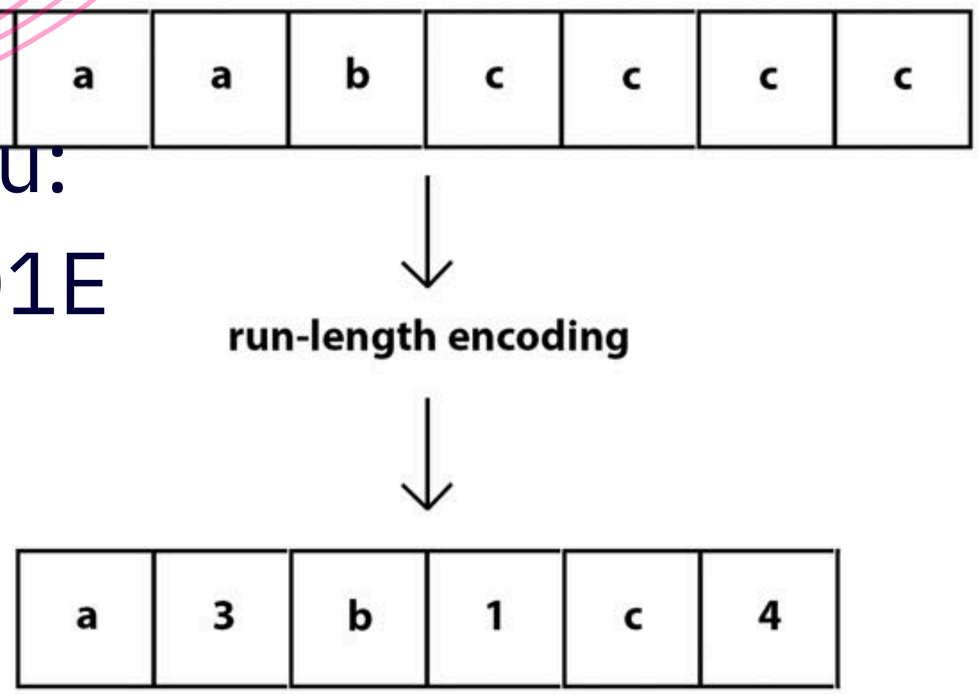
- Run-Length Encoding (RLE),
- basit bir kayıpsız veri sıkıştırma yöntemidir
- Ardışık olarak tekrar eden verileri:
- Sayarak aha kısa biçimde temsil eder
- Temel mantık:
- Tekrar eden karakter → sayı + karakter
- Amaç:
- Kullanılan bit sayısını azaltmak
- Veri boyutunu küçültmek
- En iyi çalıştığı durum:
- Tekrar oranı yüksek veriler

Orijinal veri:

ABCDE

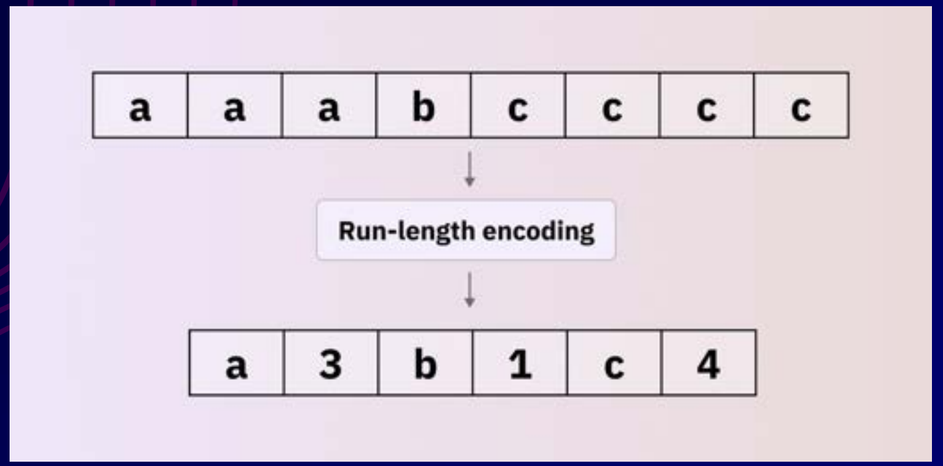
RLE sonucu:

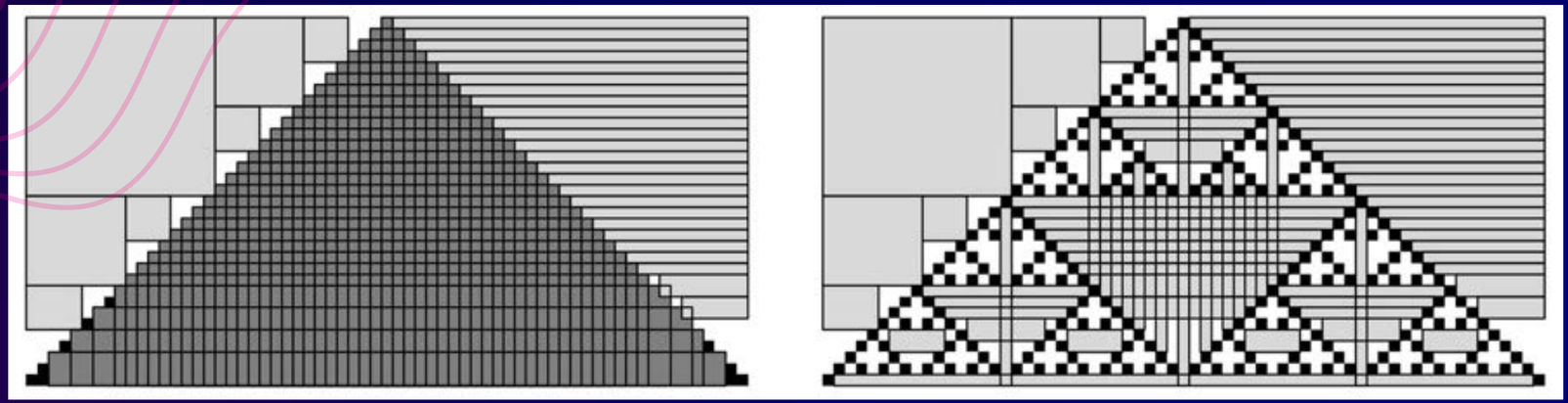
1A1B1C1D1E



RLE Mantığı (Nasıl Çalışır?)

1. Veri soldan sağa doğru okunur
2. İlk karakter referans olarak alınır
3. Ardışık tekrar eden karakterler sayılır
4. Aynı karakter devam ederse:
5. Sayaç arttırılır
6. Karakter değiştiğinde:
7. Sayı + karakter çıktıya eklenir
8. Sayaç sıfırlanır (1’den başlar)
9. Veri bittiğinde:
10. Son karakter grubu da çıktıya eklenir
11. Genel çıktı formatı:
12. Tekrar Sayısı + Karakter





RLE – Başarılı Örnek

Orijinal veri: AAAABBBCCCC
RLE sonucu: 4A3B4C

Sonuç:
Veri boyutu küçülür
Sıkıştırma başarılıdır

Karşılaştırma:
Orijinal: 11 karakter
Sıkıştırılmış: 6 karakter

Tekrar oranı yüksek olduğundan dolayı veri boyutu küçülür

RLE – Başarısız Örnek

Orijinal veri: ABCDE

RLE sonucu:1A1B1C1D1E

Karşılaştırma:

Orijinal: 5 karakter

Sıkıştırılmış: 10 karakter

Sonuç:

Veri boyutu artar

Sıkıştırma başarısızdır

Tekrar eden veri olmadığından dolayı veri boyutu artar

Önemli Not (RLE Kullanımı)

RLE her veri türü için uygun değildir

En iyi sonucu:

Tekrar eden verilerde verir

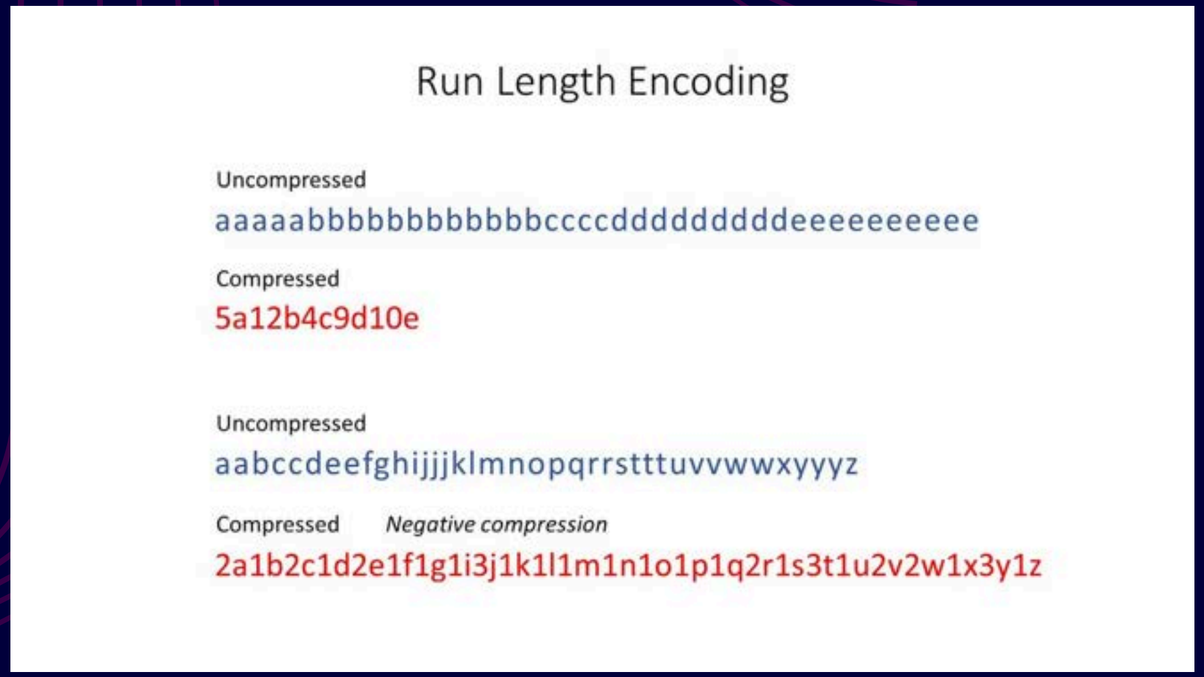
Karmaşık ve düzensiz verilerde:

Veri boyutu artabilir

RLE: Basit, Hızlı Öğretici bir algoritmadır

Gerçek hayatta:

Daha gelişmiş sıkıştırma yöntemleri tercih edilir



Bit Düzeyinde RLE

Her karakter 8 bit (1 byte) ile temsil edilir

Örnek metin:

“AAA”

Orijinal veri:

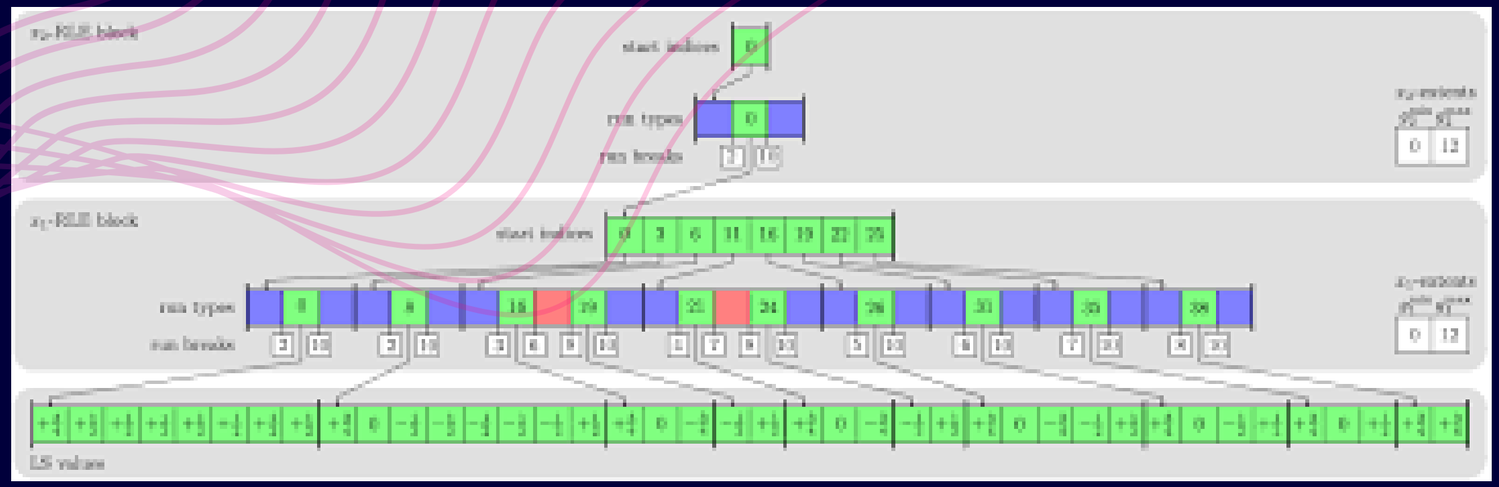
3 karakter × 8 bit = 24 bit

RLE sonrası:

“3A”

2 karakter × 8 bit = 16 bit

- Daha az bit kullanılır
- Veri boyutu küçülür



Python Uygulaması (RLE Programı)

Program Run-Length Encoding (RLE) algoritmasını uygular

Kullanıcıya menü sunar:

- Veri sıkıştırma (Encode)
- Veri çözme (Decode)

Programın yaptığı işlemler:

- Girilen metni RLE ile sıkıştırır
- Sıkıştırılmış veriyi tekrar orijinale çevirir
- Sıkıştırma oranını hesaplar

Amaç:RLE algoritmasının pratikte nasıl çalıştığını göstermek

Kod Yapısı ve Fonksiyonlar

rle_encode()

- Veriyi sıkıştırır
- Tekrar eden karakterleri sayar
- “Sayı + Karakter” formatında çıktı üretir

rle_decode()

- Sıkıştırılmış veriyi çözer
- Sayıları okuyarak karakterleri çoğaltır
- Orijinal veriyi geri oluşturur

oran_hesapla()

- Orijinal ve sıkıştırılmış veri boyutlarını karşılaştırır
- Sıkıştırma başarısını yüzde olarak hesaplar

Programın amacı:Kullanıcıdan seçim alır Encode / Decode işlemlerini çalıştırır

Genel Değerlendirme

- Bilgisayarlar tüm verileri bit düzeyinde işler
- Metin, resim ve ses verileri farklı şekillerde temsil edilir
- Veri temsili, verinin doğru saklanması ve iletilmesi için kritiktir

Veri sıkıştırma:

- Dosya boyutunu azaltır
- Depolama ve aktarım verimliliğini artırır
- Modern sistemlerde zorunludur

RLE algoritması:

- Kayıpsız bir sıkıştırma yöntemidir
- Uygulaması basit ve hızlıdır
- Tekrar içeren verilerde etkilidir
- Her veri türü için uygun değildir

Python uygulaması sayesinde:

RLE algoritmasının çalışma mantığı net olarak görülmüştür

Teorik bilginin pratiğe dökülmesi sağlanmıştır

Kaynaklar

Computer Science – Chapter 1

- 1.4 Representing Information
- 1.9 Data Compression

Akademik & Eğitim Kaynakları

- Veri Temsili ve Bit Düzeyi Gösterimler – Ders Notları
- Veri Sıkıştırma Temelleri – Üniversite Sunumları

Algoritma Kaynakları

- Run-Length Encoding (RLE) Algoritması – Temel Algoritma Açıklamaları
- Kayıpsız Veri Sıkıştırma Yöntemleri
- Python Resmi Dokümantasyonu
- Basit Veri Sıkıştırma Uygulamaları (RLE Örnekleri)