BLM101 Dönem Projesi

Veri Depolama ve Sıkıştırma Algoritmaları - RLE (Run-Length Encoding)

Bilgiler

Öğrenci: Suat Samet Kömürcü

Öğrenci No: 24360859016

Bölüm: Bilgisayar Mühendisliği

İçindekiler

RLE (Run-Length Encoding) - Kayıpsız Sıkıştırma

Veri ve Depolama Problemi

RLE (Run-Length Encoding) - Kayıpsız Sıkıştırma

- Günlük hayatta metin, görüntü ve ses gibi büyük miktarda veri üretilir.
- Ham (sıkıştırılmamış) veri boyutları hızla artar; depolama ve aktarım maliyeti yükselir.
- Sıkıştırma, aynı bilgiyi daha az bit ile temsil etmeyi amaçlar.

Hedef: Veri kaybı olmadan (kayıpsız) boyutu azaltmak.

Bit - Byte - Temsil

RLE (Run-Length Encoding) - Kayıpsız Sıkıştırma

- Bit: 0 veya 1 (en küçük bilgi birimi).
- Byte: 8 bit. Dosya boyutları genellikle byte ve katlarıyla ölçülür (KB, MB, GB).
- Bir veri türünü temsil etmek için belirli kodlama (encoding) kullanılır.

Örnek

1 KB = 1024 byte
1 MB = 1024 KB

Metin Verisi

RLE (Run-Length Encoding) - Kayıpsız Sıkıştırma

- Metin verisi karakterlerden oluşur ve sayısal kodlarla temsil edilir.
- ASCII: 7-bit/8-bit karakter kodları (temel İngilizce).
- Unicode (UTF-8): farklı diller ve semboller için geniş kapsam.

Not

Türkçe karakterler UTF-8 ile doğru temsil edilir.

Görüntü Verisi

RLE (Run-Length Encoding) - Kayıpsız Sıkıştırma

- Görüntü, piksel (pixel) adı verilen noktalardan oluşur.
- Her piksel renk kanallarıyla temsil edilir (ör. RGB).
- Bit derinliği arttıkça renk hassasiyeti artar, dosya boyutu büyür.

Örnek Hesap

1920x1080 RGB, 24-bit -> yaklaşık 1920*1080*3 = 6.2 MB (sıkıştırmasız)

Ses Verisi

RLE (Run-Length Encoding) - Kayıpsız Sıkıştırma

- Ses dalgası belirli aralıklarla örneklenir (sampling).
- Örnekleme frekansı (Hz) ve bit derinliği dosya boyutunu belirler.
- Stereo ses iki kanal içerir (sol/sağ).

Örnek

44.1 kHz, 16-bit stereo -> saniyede 44,100*16*2 bit

Neden Sıkıştırma?

RLE (Run-Length Encoding) - Kayıpsız Sıkıştırma

- Depolama maliyetini azaltır (disk/SSD/bulut).
- İnternet üzerinden daha hızlı aktarım sağlar.
- Yedekleme sürelerini kısaltır, bant genişliğini verimli kullanır.

Kayıpsız ve Kayıplı Sıkıştırma

RLE (Run-Length Encoding) - Kayıpsız Sıkıştırma

Kayıpsız (Lossless)

- Orijinal veri tamamen geri elde edilir.
- Metin, kaynak kod, bazı görüntüler için uygundur.
- Ör: RLE, Huffman, LZW, PNG

Kayıplı (Lossy)

- Bazı bilgiler atılır, tam geri dönüş yoktur.
- İnsan algısına göre önemsiz kısımlar silinir.
- Ör: JPEG, MP3, AAC

Sıkıştırma Ölçütleri

RLE (Run-Length Encoding) - Kayıpsız Sıkıştırma

- Sıkıştırma oranı: boyuttaki azalma (veya artış) yüzdesi.
- Sıkıştırma/açma hızı: gerçek zamanlı kullanım için önemli.
- Bellek kullanımı: özellikle büyük verilerde kritik.
- Veri türüne göre yöntem seçilir (tek yöntem her yerde iyi değildir).

Temel Fikir: Tekrarları Yakalama

RLE (Run-Length Encoding) - Kayıpsız Sıkıştırma

- Birçok veride tekrar eden semboller veya örüntüler bulunur.
- RLE, art arda tekrar eden karakterleri 'adet + sembol' olarak yazar.
- Bu yaklaşım basit, hızlı ve uygulaması kolaydır.

RLE, özellikle uzun tekrar serileri olan verilerde çok etkilidir.

RLE Nedir?

RLE (Run-Length Encoding) - Kayıpsız Sıkıştırma

- Run-Length Encoding (RLE) = Çalışma Uzunluğu Kodlama.
- Art arda gelen aynı karakterlerin sayısını ve karakteri tutar.
- Kodlama biçimi (bu projede): <sayı><karakter> tekrarları.

Örnek

AAAAA -> 5A
BBB -> 3B

RLE Ne Zaman İyi Çalışır?

RLE (Run-Length Encoding) - Kayıpsız Sıkıştırma

- Uzun tekrarlar içeren metinler (ör. 'AAAAA...').
- Tek renk blokları olan basit görüntüler (ikon, maske, tarama).
- Bazı sensör/telemetri verileri (uzun sabit değer aralıkları).

RLE Ne Zaman Kötü Çalışır?

RLE (Run-Length Encoding) - Kayıpsız Sıkıştırma

- Karakterler sık değişiyorsa: ABCDEF... gibi.
- Tekrar uzunluğu çoğunlukla 1 ise: 1A1B1C... (boyutu artırabilir).
- Bu yüzden sıkıştırma sonrası oran hesaplanıp değerlendirilir.

RLE her zaman küçültmez; bazen veri büyüyebilir.

Encode Algoritması (Adım Adım)

RLE (Run-Length Encoding) - Kayıpsız Sıkıştırma

- Metni soldan sağa dolaş.
- Bir karakterin ardışık tekrar sayısını (count) artır.
- Karakter değişince 'count + karakter' çıktıya ekle ve sayacı sıfırla.
- Metin bitince son grubun çıktısını da ekle.

Decode Algoritması (Adım Adım)

RLE (Run-Length Encoding) - Kayıpsız Sıkıştırma

- Sıkıştırılmış metni soldan sağa oku.
- Rakamları biriktirerek sayıyı (count) oluştur.
- Bir harf/karakter görünce o karakteri count kez yaz.
- Sonda sayı kalırsa veya sayı olmadan karakter gelirse hata say.

Örnek Encode

RLE (Run-Length Encoding) - Kayıpsız Sıkıştırma

Girdi / Çıktı

Girdi : AAAAABBBCCDAA

Çıktı : 5A3B2C1D2A

Yorum : A(5), B(3), C(2), D(1), A(2)

- Her grup için tekrar sayısı ve karakter yazılır.
- Bu örnekte orijinal uzunluk 13, encoded uzunluk 10.

Örnek Decode

RLE (Run-Length Encoding) - Kayıpsız Sıkıştırma

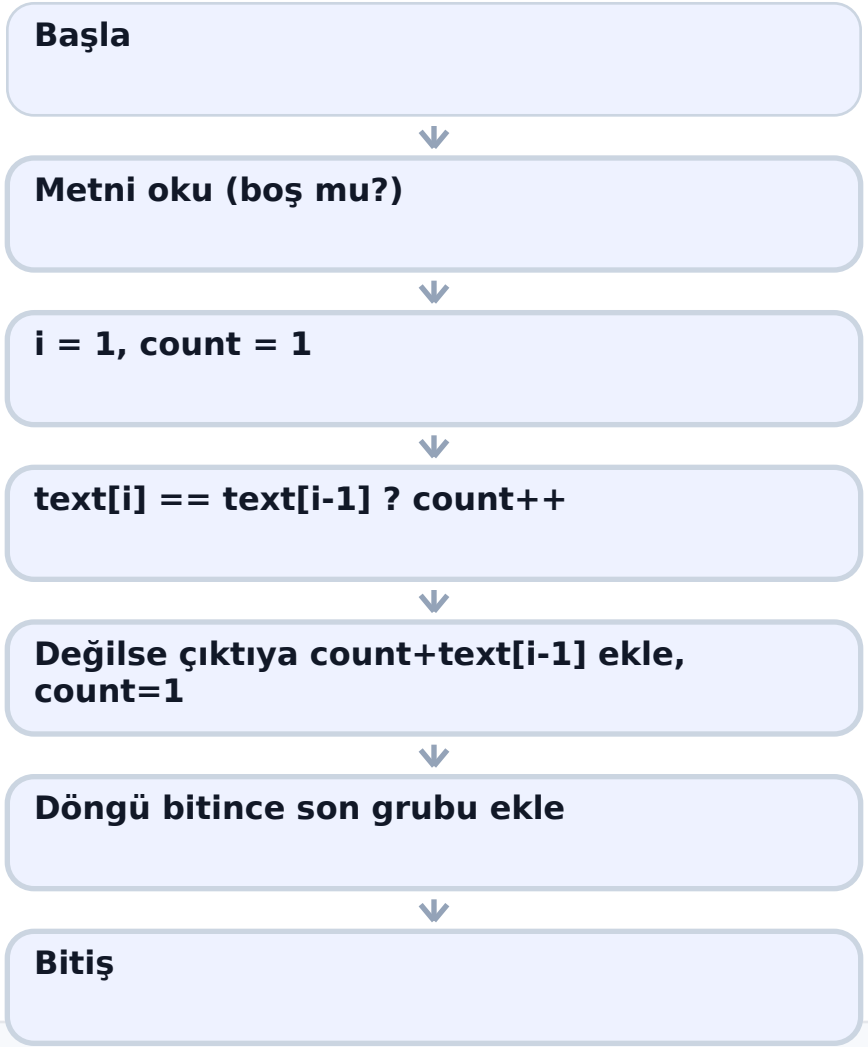
Girdi / Çıktı
Girdi : 5A3B2C1D2A
Çıktı : AAAAABBBCCDAA

Kontrol : Encode(Çıktı) tekrar 5A3B2C1D2A verir.

- Decode işlemi encode’un tersidir.
- Kayıpsız sıkıştırmada orijinal veri tamamen geri gelir.

Akış Diyagramı - Encode

RLE (Run-Length Encoding) - Kayıpsız Sıkıştırma

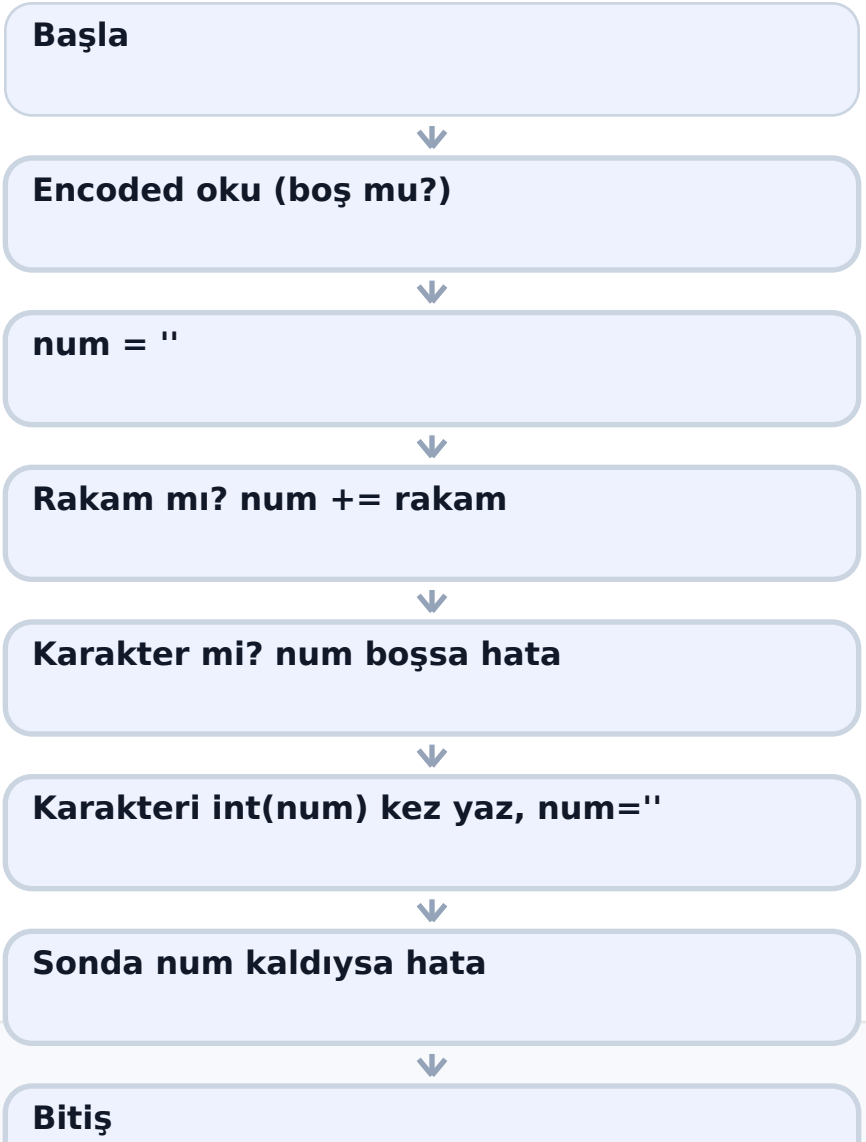


Notlar

- Zaman karmaşıklığı: O(n)
- Ek bellek: O(1) (çıktı hariç)

Akış Diyagramı - Decode

RLE (Run-Length Encoding) - Kayıpsız Sıkıştırma



Notlar

- Zaman karmaşıklığı: O(m) (encoded uzunluğu)
- Hatalı biçim kontrolü yapılır

Sıkıştırma Oranı

RLE (Run-Length Encoding) - Kayıpsız Sıkıştırma

Oran(%) = (1 - encoded_len / original_len) * 100

- Pozitif değer: veri küçüldü.
- 0'a yakın: fark yok.
- Negatif değer: veri büyüdü (RLE için mümkün).

Örnek

original=13, encoded=10 -> (1-10/13)*100 = 23.08%

Uygulama Tasarımı

RLE (Run-Length Encoding) - Kayıpsız Sıkıştırma

- Dil: Python 3
- Dosya yapısı: src/rle.py + sunum.pdf + README.md
- Kullanıcı arayüzü: basit menü (encode / decode / çıkış)
- Çıktılar: encoded string, decoded string, oran (%)

Fonksiyonlar

RLE (Run-Length Encoding) - Kayıpsız Sıkıştırma

Fonksiyon İmzaları

```
rle_encode(text: str) -> str
rle_decode(encoded: str) -> str
compression_ratio_percent(original: str, encoded: str) -> float
main() -> None
```

- Modüler yapı: test etmek ve anlatmak kolaylaşır.
- Encode/Decode ayrı yazıldığı için terslenebilirlik net görülür.

Hata Durumları ve Edge-Case'ler

RLE (Run-Length Encoding) - Kayıpsız Sıkıştırma

- Boş metin: encode/decode sonucu boş olur.
- Encoded metin sayı ile bitiyorsa: hata (örn: '12A3').
- Sayı olmadan karakter gelirse: hata (örn: 'A3B').
- Çok basamaklı sayılar desteklenir (örn: '12A').

Demo - Encode

RLE (Run-Length Encoding) - Kayıpsız Sıkıştırma

Örnek Çalıştırma

```
=== RLE Sıkıştırıcı ===
1) Encode (Sıkıştır)
2) Decode (Aç)
3) Çıkış
Seçim: 1
Metin: AAAAABBBCCDAA
Encoded: 5A3B2C1D2A
Sıkıştırma oranı: 23.08%
```

Demo - Decode

RLE (Run-Length Encoding) - Kayıpsız Sıkıştırma

Örnek Çalıştırma
Seçim: 2
Encoded: 5A3B2C1D2A
Decoded: AAAAABBBCCDAA

Not

Decode sonrası verinin doğruluğu encode ederek tekrar kontrol edilebilir.

Demo - Veri Büyümesi Örneği

RLE (Run-Length Encoding) - Kayıpsız Sıkıştırma

RLE Her Zaman Küçültmez
Metin : ABCDEF
Encoded: 1A1B1C1D1E1F
Oran   : negatif (veri büyüdü)

- Bu tür verilerde RLE uygun bir seçim olmayabilir.
- Bu yüzden gerçek dünyada veri türüne göre yöntem seçilir.

Sonuç

RLE (Run-Length Encoding) - Kayıpsız Sıkıştırma

- RLE, tekrarları sayarak kodlayan basit bir kayıpsız sıkıştırma yöntemidir.
- Uygulaması hızlıdır ve düşük ek bellek gerektirir.
- En iyi performans uzun tekrar serilerinde elde edilir.
- Proje kapsamında encode, decode ve oran hesaplama başarıyla gerçekleştirildi.

Gelecek geliştirme: dosya okuma/yazma, farklı biçimler, daha güçlü yöntemlerle karşılaştırma.

Kaynakça

RLE (Run-Length Encoding) - Kayıpsız Sıkıştırma

- BLM101 Dönem Projesi yönergesi (ders dokümanı).
- Veri Depolama ve Sıkıştırma Algoritmaları - RLE Sıkıştırıcı (ders dokümanı).
- Salomon, D. (2007). Data Compression: The Complete Reference. (Genel kaynak).
- Wikipedia: Run-length encoding (genel bilgi).
    - Brookshear, J. G. & Brylow, D. (2014). Computer Science: An Overview (12th Global Edition), Bölüm 1.4 (Representing Information as Bit Patterns) ve 1.9 (Data Compression).

Not

Sunumda kullanılan örnekler ve kod, proje gereksinimlerine göre hazırlanmıştır.