Veri Sıkıştırma ve RLE Projesi

Bilgisayar Mühendisliği'ne Giriş

İsim: MAHMOUD ESAM I ALFALAH

Öğrenci No: 24360859821

Sunum İçeriği

1. Veri Nasıl Saklanır? (Bölüm 1.4)
2. Veri Sıkıştırma Yöntemleri (Bölüm 1.9)
3. Kayıplı ve Kayıpsız Sıkıştırma
4. Benim Projem: RLE (Run-Length Encoding)
5. Kod İncelemesi ve Demo

Bölüm 1.4: Bilginin Gösterimi

1. Bilgisayarlar sadece 0 ve 1 (Bit) kullanır.

2. Gerçek dünyadaki verileri 'Bit Desenlerine' çevirmeliyiz:

3. Metin, Görüntü ve Ses

Metin Gösterimi (Text)

1. Her karakterin bir sayısal karşılığı vardır.
2. ASCII: 7-bit (Sadece İngilizce)
3. Unicode (UTF-8): Tüm Diller ve Emojiler

ASCII Örneği: 'Hello.'

01001000	01100101	01101100	01101100	01101111	00101110
H	e	l	l	o	.

Görüntülerin Gösterimi

1. Bit Eşlem (Bitmap) Teknikleri
2. Piksel: En küçük görüntü elemanı
3. Yaklaşınca kareleri görürüz.

Renkler: RGB Modeli

1. Her piksel 3 renkten oluşur:
2. R (Kırmızı)
3. G (Yeşil)
4. B (Mavi)
5. Örnek: (255, 0, 0) = Saf Kırmızı

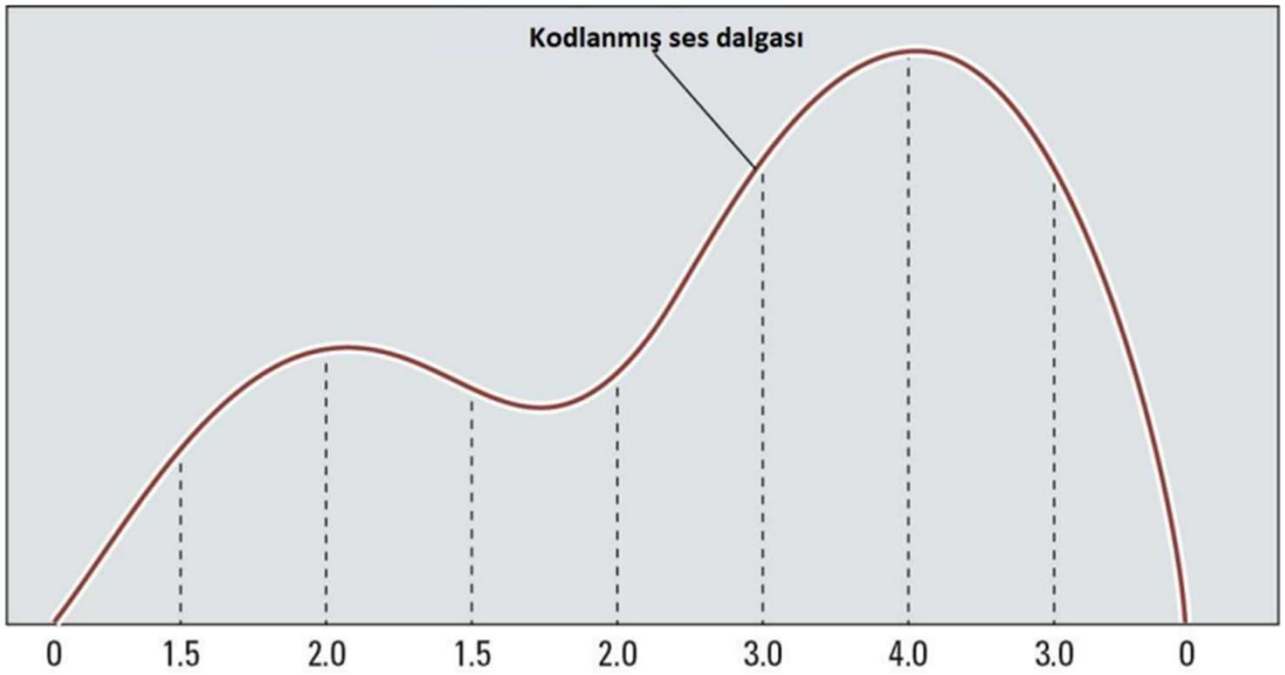
Sesin Gösterimi

1. Ses = Titreşim (Dalga)
2. Bilgisayar dalgayı saklayamaz.
3. Çözüm: Ölçüm Yapmak (Sampling).

Örnekleme Oranları

1. Telefon: 8000 örnek/saniye
2. CD Sesi: 44.100 örnek/saniye
3. Daha çok örnek = Daha iyi ses kalitesi

Ses Dalgası Grafiği



Sorun: Büyük Dosyalar

1. 1 Resim = Milyonlarca Piksel
2. 1 Şarkı = Milyonlarca Örnek
3. Sonuç: Yığın Depolama dolar, internet yavaşlar.

Çözüm: Veri Sıkıştırma

1. Bölüm 1.9
2. Amaç: Veriyi daha az bit ile saklamak.
3. Verimlilik ve Hız artar.

1.9 Veri Sıkıştırma Konuları

1. • Kayıplı ve Kayıpsız
2. • Kodlama Yöntemleri:

    - İşlem uzunluğu kodlama (RLE)
    - Frekans-bağımlı kodlama (Huffman)
    - Göreceli kodlama
    - Sözlük kodlama

1. Kayıplı (Lossy) Sıkıştırma

1. Bazı veriler silinir.
2. Geri getirilemez.
3. Örnek: JPEG (Foto), MP3 (Müzik)
4. İnsan gözü fark etmez.

2. Kayıpsız (Lossless) Sıkıştırma

1. Hiçbir veri silinmez.
2. Orijinal dosya %100 geri gelir.
3. Örnek: ZIP, RLE (Benim projem).

Bavul Analojisi

1. Kıyafetleri katlamak = Sıkıştırma
2. Bavulu açmak = Decompress
3. Kıyafetleri kesip atmıyoruz!

Benim Projem: RLE

1. Run-Length Encoding
2. (İşlem Uzunluğu Kodlama)
3. Tekrar eden verileri sayar.

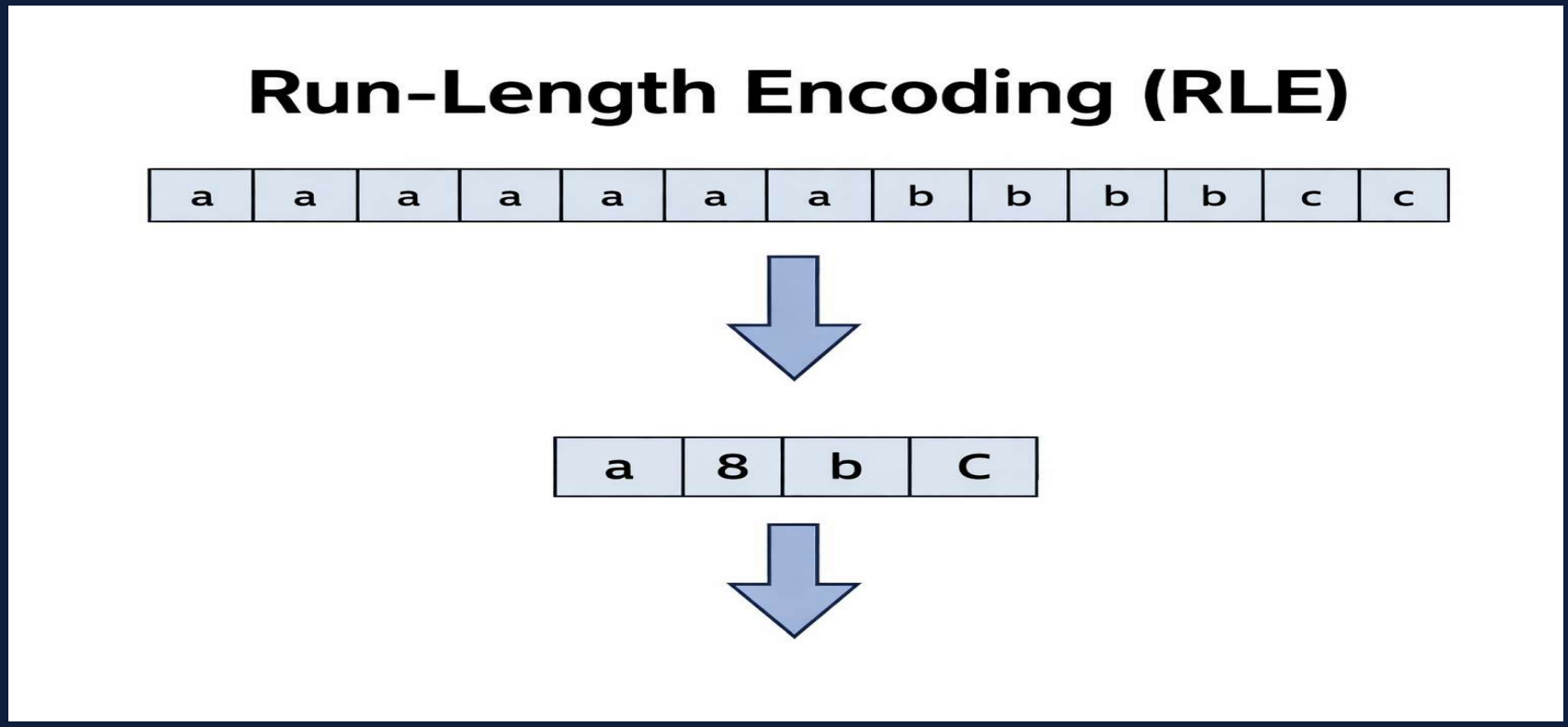
RLE Nasıl Çalışır?

1. Girdi: A A A A A B B B

2. Çıktı: 5A 3B

3. Kazanç: 8 karakter -> 4 karakter

Kitaptaki Örnek (Sayfa 61)



Python Kodum: Encoder

```python
# Kodlama yapan fonksyinu, ornek: AAABB -> 3A2B
def encoder(text):
    # su andkai sayildigimiz harfi
    current = ''
    # her harf kac kere yazildigi sayan degiskeni
    counter = 0
    # sonuc tutan degiskeni
    result = ""
    # kullancinin girdigi metni tek tek dolasi artiyoruz
        if i == current:
    for i in text:
        # current harfi i ile ayni ise, demek hala sayac
            counter += 1
        else:
            # farkli ise, iki durumu olabilir
            # ya counter > 0, demek ki daha once saydigimiz harf var, onu sonucu ekle
            if (counter > 0):
                """
                    countr > 0  ise, demek ki daha once saydigimiz harf var, ve bu
                    itersayonda farkliu bir harf ile karsilastik, dolaysiyla sonuca ekleme
                    yapmaliyiz
                """
                result += f"{counter}{current}"
```

Python Kodum: Encoder

```
            # yoksaa birnci harf demektir ve hala her hangi bir sayma islemi yapmadik

            # (i == current) degil ise, demek yeni bir harf ile karsilastik
            # counter 1 olur, simdiki harfi sayilmak icin
            counter = 1
            # current i olur, yani bit harf ile sayma islemi basladik
            current = i
        """
            bu satir debugging icin kullanildi
            print(f"i is {i}, current is {current}, counter is {counter}")
        """
    """
        son harftan sonra baska bir iterasyonu olmayacagi icin, son harfi ve sayisini
        eklememiz lazim, cunku her zaman ekleme islemi sonraki iterasyonda olur
        kullandigim deekleme yontem f-string ile yapildi
    """
    result += f"{counter}{current}"
    return result
```

Python Kodum: Decoder

```python
# Kod cozme fonksiyonu, ornek: 3A2B -> AAABB
def decoder(text):
    # sonuc tutan degiskeni
    result = ""
    """
        sayac degiskeni, ve string olarak baslatildi cunku birden fazla basamakli sayilar
        olabilir, ve o  durumda sayiyi string olarak tutup sonra inte cevircegiz
    """
    count = ""
    # kullanicinin girdigi metni tek tek dolas
    for i in text:
        """
            isDigit fonksyonu bir karakter sayi olup olmadigini belirtiren bir methodtur
            Ayrica kednimiz fonksiyonumuz da olusturabilirz try, except ve int() kullnarak
        """
        if i.isdigit():
            # i'ninci karakteri sayi ise, count degiskenine ekle (string concatenation)
            count += i
        else:
            """
                yoksa, count degiskenini inte cevir ve i karakteri ile carp,
                ama burada try, except kullaildi, cunku brinici itersayonda eger i'ninci
                karakter sayi degil ise, count hala bos string olacak ve int("") hata
                verecek
            """
```

Python Kodum: Decoder

```
            """
            try:
                count = int(count)
            except ValueError:
                """
                hata verdi ise, demek ki bu birinci karakter ve count hala bos ve sadece
                bir kes yazmak istemis, once 1 yazmaya untulmus ve o andda count 1
                tamamlaniyoruz
                ornek, girdigi metin: A3B, sonuc ABBB olmali
                """
                count = 1
            # sonucu guncelle
            result += i * count
            count = ""
    return result
```

RLE Dezavantajı

1. Her zaman çalışmaz!
2. Örnek: 'A B C D' (Tekrar yok)
3. RLE: '1A 1B 1C 1D' (Dosya Büyür!)

Diğer Yöntem: Huffman

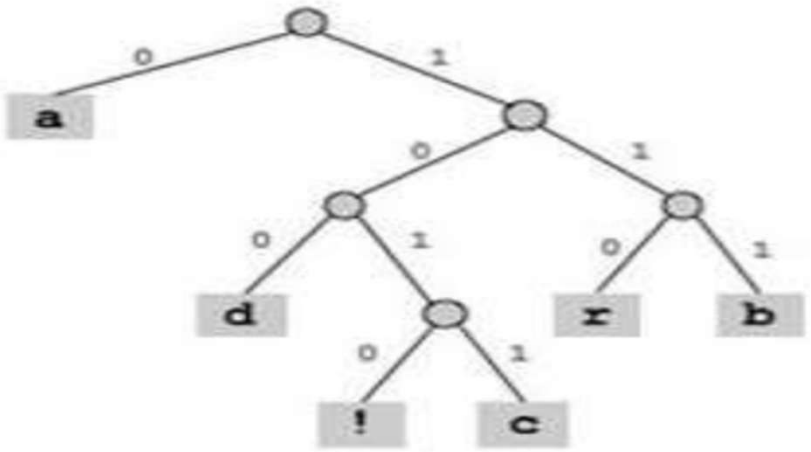
1. Frekans-Bağımlı Kodlama
2. Sık kullanılan harf = Kısa Kod
3. Az kullanılan harf = Uzun Kod

Örnek: Mors Alfabesi

1. 'E' çok sık kullanılır -> .
2. 'Q' nadir kullanılır -> --.-
3. Huffman mantığı aynıdır.

Huffman Ağacı

HUFFMAN CODE
DATA COMPRESSION



char	encoding
a	0
b	111
c	1011
d	100
r	110
!	1010

Diğer Yöntem: Göreceli Kodlama

Relative Encoding

- Relative Encoding:

1st Frame	2nd Frame	Difference
1 2 3 4	1 3 3 4	0 1 0 0
2 5 3 7	2 5 3 7	0 0 0 0
3 6 4 8	3 6 4 7	0 0 0 -1
4 7 5 9	3 7 5 9	-1 0 0 0

Resulting difference can be RLE.

Örnek: Film Kareleri

1. Arka plan sabittir.
2. Sadece oyuncu hareket eder.
3. Bilgisayar sadece farkı kaydeder.

Diğer Yöntem: Sözlük Kodlama

Dictionary-based encoding example

- Dictionary:

1. ASK
2. NOT
3. WHAT
4. YOUR
5. COUNTRY
6. CAN
7. DO
8. FOR
9. YOU

- Original text:
- ASK NOT WHAT YOUR COUNTRY CAN DO FOR YOU ASK WHAT YOU CAN DO FOR YOUR COUNTRY

- Encoded based on dictionary :
- 1 2 3 4 5 6 7 8 9 1 3 9 6 7 8 4 5

Programlama Analojisi

1. Sözlük = Fonksiyonlar
2. Kodu bir kere yaz (Define).
3. İsmini çağır (Call).

Sonuç

1. Veriler 0 ve 1'dir.
2. RLE basit ve etkilidir.
3. Modern yöntemler daha karmaşıktır.

Teşekkürler

1. Dinlediğiniz için teşekkür ederim.

2. Kaynak: Bilgisayar Bilimine Giriş, 13. Baskı, Bölüm 1