



# **Bölüm 5: Dizgi Algoritmaları**

## **Algoritmalar**



# Dizgi Algoritmaları

- Metinlerle dolu bir dünyada yaşıyoruz.
- E-postalar, mesajlar, sosyal medya paylaşımları, haber metinleri...
- Bilgisayarlarımızda her gün sayısız metinle karşılaşırız.
- Peki, bu metinler nasıl düzenlenir ve analiz edilir?
- Dizgi (String) algoritmaları,
  - metinlerde arama,
  - değiştirme,
  - karşılaştırma gibi işlemleri gerçekleştirir.



# Dizgi Eşleştirme Algoritmaları

- Brute Force (Kaba Kuvvet):
  - Metindeki her konum örüntü ile eşleştirmek için kontrol edilir.
  - Maksimum sayıda karşılaştırma gerektirebilir.
- Knuth-Morris-Pratt (KMP)
  - Başlangıçta tablo oluşturularak arama süresi azaltılır,
  - Karakter karşılaştırmalarını azaltarak hızlı çalışır.
- Boyer-Moore
  - Uzun aramalarda etkili. Kök bulma ve kaydırma stratejisi kullanır.
- Rabin-Karp Algoritması
  - Olasılıksal bir algoritma. Hashing kullanır.



# Dizgi Sıralama Algoritmaları

- Sözlüksel Sıralama (Lexicographic Order)
  - Dizgiler, alfabetik sıraya benzer sıralanır.
  - Her karakterin ASCII değeri karşılaştırılarak sıralama yapılır.
- Radix Sıralama
  - Karşılaştırmalı olmayan bir tam sayı sıralama algoritmasıdır.
  - Veriler tamsayı anahtarlarına sahiptir.
  - Aynı konumda aynı değeri paylaşan verileri gruplandırarak sıralar.
  - Her basamak için ayrı ayrı işlem yapılır.



# Dizgi Ayırıştırma (Parsing) Algoritmaları

- Düzenli İfadeler (Regular Expressions)
  - Bir arama örüntüsünü tanımlayan karakter dizisi,
  - Belirli bir örüntüye uyan tüm dizgileri bulmak için kullanılır
- Sonlu Durum Makineleri (Finite State Machines - FSM)
  - Dizgi içindeki örüntüleri tanımak için kullanılan hesaplama modelleri,
  - Belirli bir girdi dizisindeki geçişlerin durumlarını izleyen bir otomat,
  - Karmaşık ayırıştırma ve analiz işlemlerinde kullanılır.



# Dizgi Düzenleme Mesafesi Algoritmaları

- Levenshtein Mesafesi
  - İki dizgi arasındaki benzerliği ölçen bir metrik,
  - Bir dizgiden diğerine dönüştürmek için gereken minimum tek karakterli düzenleme sayısı olarak tanımlanır.
- En Uzun Ortak Alt Dizi (Longest Common Subsequence - LCS)
  - İki dizginin ortak olan en uzun alt dizisi,
  - Karakterlerin sıralı olmasını gerektirmez, ancak sıra korunmalıdır.
  - Dizgiler arasındaki benzerlik veya farkı belirlemek için kullanılır.



# Dizgi Dönüşüm Algoritmaları

- Sonek Dizisi (Suffix Array)
  - Bir dizginin tüm son eklerinin bir dizisi.
  - Dizgi içindeki alt dizgilerin bir temsili olarak kullanılır.
- Burrows-Wheeler Dönüşümü (BWT)
  - Bir dizginin tersine dönüştürülmesiyle elde edilen yeni bir form,
  - Bzip2 gibi sıkıştırma algoritmaları için ön işlem adımı olarak kullanılır.



# Dizgi Sıkıştırma Algoritmaları

- Sıralı Sıkıştırma Kodlaması (Run Length Encoding)
  - Aynı veri değerleri tek bir değer ve sayı olarak saklanır.
  - Tekrar eden değerler yerine tekrar eden veri sayısı saklanır.
- Lempel-Ziv-Welch (LZW)
  - GIF gibi formatlarda kullanılan sözlük tabanlı sıkıştırma algoritması.
  - Tekrar eden örüntüleri sözlük oluşturarak kısa sembollerle temsil eder.
  - Dinamik bir sözlük kullanarak sıkıştırma sağlar.





# Sıralı Sıkıştırma Kodlaması (RLE)

- RLE, metin verilerini sıkıştırmak için kullanılır.
- Ardışık tekrar eden karakterleri bir kodla yer değiştirir.
- Metin sıkıştırmada kullanılan basit ve etkili bir yöntemdir.
- Tekrar eden karakterlerin sayısı az ise etkisiz olabilir.



# RLE Sıkıştırma

- Sıkıştırma Süreci:
  - Metin: "AAAAABBBCCCCDDD"
  - Sıkıştırılmış Hali: "5A3B4C3D"
  - Ardışık tekrar eden karakterlerin sayısı ve kendisi ile temsil edilmesi.



# Sıkıştırma Adımları

- Metin soldan sağa taranır.
- Ardışık tekrar eden karakterler sayılır.
- Tekrar eden karakterler için sayı ve karakter bir araya getirilir.
- Bu bilgiler sıkıştırılmış metin olarak depolanır.



# Algoritma Karmaşıklığı

- RLE algoritması, girdi metni tek bir kez tarar.
- Ardışık tekrar eden karakterlerin sayısını kaydeder.
- Bu nedenle, algoritmanın zaman karmaşıklığı,  $O(n)$ .



# RLE Sıkıştırma


W1B1W3B1W1

B3W1B3

B7

W1B5W1

W2B3W2

W3B1W3



# RLE Sıkıştırma

- **Metin:**

- aaaaaaaaaabbbbbbbececececececdccccccccccccccccccdec (46 bytes)

- **RLE sayı karakter çiftleri:**

- (10, a)(6, b)(1, e)(1, c)(1, e)(1, c)(1, e)(1, c)(1, e) (1, c)(1, e)(1, c)(1, e)(1, c)(15, d)(1, e)(1, c)(1, b)

- **Bayrak olmadan RLE:**

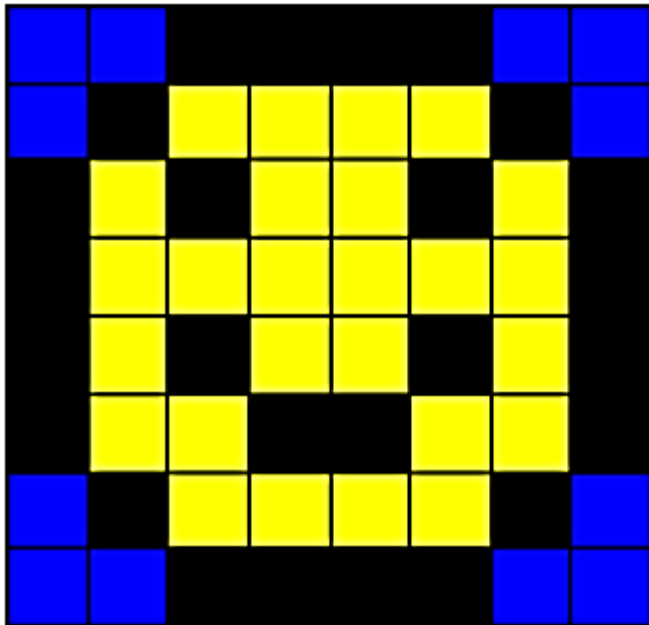
- 10 97 06 98 01 101 01 99 01 101 01 99 01 101 01 99 01 101 01 99 01 101 01 99 01 101 01 99 15 100 01 101 01 99 01 98(36 bytes)




- **Bayrak kullanılarak (255) RLE:**

- 255 10 97 255 06 98 101 99 101 99 101 99 101 99 101 99 101 99 255 15 100 101 99 98 (24 bytes)



# Pixel level RLE



RGB colour values			
	R	G	B
	0	0	255
	0	0	0
	255	255	0



# Pixel level RLE

Count	R	G	B	Count	R	G	B	Count	R	G	B
2	0	0	255	4	0	0	0	3	0	0	255
1	0	0	0	4	255	255	0	1	0	0	0
1	0	0	255	1	0	0	0	1	255	255	0
1	0	0	0	2	255	255	0	1	0	0	0
1	255	255	0	2	0	0	0	6	255	255	0
2	0	0	0	1	255	255	0	1	0	0	0
2	255	255	0	1	0	0	0	1	255	255	0
2	0	0	0	2	255	255	0	2	0	0	0
2	255	255	0	1	0	0	0	1	0	0	255
1	0	0	0	4	255	255	0	1	0	0	0
3	0	0	255	4	0	0	0	2	0	0	255







# Lempel-Ziv-Welch (LZW) Sıkıştırma Algoritması

- LZW, metin sıkıştırma için kullanılan etkili bir algoritmadır.
- Tekrar eden dizgileri tanımlayarak sıkıştırma yapar.
- Metin sıkıştırmada yaygın olarak kullanılan bir algoritmadır.
- Sözlük boyutu büyüdükçe performansı düşer.



# LZW Sıkıştırma

- Sıkıştırma Süreci:
  - Metin: "ABABCABABCA"
  - Sıkıştırılmış Hali: "0 1 2 4 3 5"
  - Metin içindeki tekrar eden dizgilerin bir indeksle temsil edilmesi.



# Sıkıştırma Adımları

- Metin soldan sağa taranır.
- İndekslerle tanımlanan dizgilerin bulunduğu bir sözlük oluşturulur.
- Sözlükte olmayan yeni dizgiler eklenir.
- Dizgiler sıkıştırılmış metin olarak depolanır.



# Lempel-Ziv-Welch

A T G A T C A T G A G

code	word	out
------	------	-----

0	A	
1	T	
2	G	
3	C	
4	AT	0
5	TG	1
6	GA	2
7	ATC	4
8	CA	3
9	ATG	4
10	GAG	6
		2



# Lempel-Ziv-Welch

	Output	Dict.													
<table><tr><td>a</td><td>a</td><td>b</td><td>a</td><td>a</td><td>c</td><td>a</td><td>b</td><td>c</td><td>a</td><td>b</td><td>c</td><td>b</td></tr></table>	a	a	b	a	a	c	a	b	c	a	b	c	b	(0, a)	1 = a
a	a	b	a	a	c	a	b	c	a	b	c	b			
<table><tr><td>a</td><td>a</td><td>b</td><td>a</td><td>a</td><td>c</td><td>a</td><td>b</td><td>c</td><td>a</td><td>b</td><td>c</td><td>b</td></tr></table>	a	a	b	a	a	c	a	b	c	a	b	c	b	(1, b)	2 = ab
a	a	b	a	a	c	a	b	c	a	b	c	b			
<table><tr><td>a</td><td>a</td><td>b</td><td>a</td><td>a</td><td>c</td><td>a</td><td>b</td><td>c</td><td>a</td><td>b</td><td>c</td><td>b</td></tr></table>	a	a	b	a	a	c	a	b	c	a	b	c	b	(1, a)	3 = aa
a	a	b	a	a	c	a	b	c	a	b	c	b			
<table><tr><td>a</td><td>a</td><td>b</td><td>a</td><td>a</td><td>c</td><td>a</td><td>b</td><td>c</td><td>a</td><td>b</td><td>c</td><td>b</td></tr></table>	a	a	b	a	a	c	a	b	c	a	b	c	b	(0, c)	4 = c
a	a	b	a	a	c	a	b	c	a	b	c	b			
<table><tr><td>a</td><td>a</td><td>b</td><td>a</td><td>a</td><td>c</td><td>a</td><td>b</td><td>c</td><td>a</td><td>b</td><td>c</td><td>b</td></tr></table>	a	a	b	a	a	c	a	b	c	a	b	c	b	(2, c)	5 = abc
a	a	b	a	a	c	a	b	c	a	b	c	b			
<table><tr><td>a</td><td>a</td><td>b</td><td>a</td><td>a</td><td>c</td><td>a</td><td>b</td><td>c</td><td>a</td><td>b</td><td>c</td><td>b</td></tr></table>	a	a	b	a	a	c	a	b	c	a	b	c	b	(5, b)	6 = abcb
a	a	b	a	a	c	a	b	c	a	b	c	b			



# Lempel-Ziv-Welch

	Output	Dict.
a a b a a c a b a b a c b	112	256=aa
a a b a a c a b a b a c b	112	257=ab
a a b a a c a b a b a c b	113	258=ba
a a b a a c a b a b a c b	256	259=aac
a a b a a c a b a b a c b	114	260=ca
a a b a a c a b a b a c b	257	261=aba
a a b a a c a b a b a c b	261	262=abac
a a b a a c a b a b a c b	114	263=cb





SON