



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,
 - deęiştirme,
 - 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 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.



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 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 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.



Levenshtein Mesafesi

- İki dizgi arasındaki farkı nicel olarak ölçen etkili bir metriktir.
- Otomatik düzeltme ve tahmin sistemlerinde kullanılabilir.
- İki dizgi arasındaki minimum düzenleme (ekleme, çıkarma veya değiştirme) işlem sayısını belirtir.



Temel İşlemler

- Ekleme (Insertion): Bir karakterin eklenmesi.
- Çıkarma (Deletion): Bir karakterin çıkarılması.
- Değişirme (Substitution): Bir karakterin başka bir karakterle değiştirilmesi.



Levenshtein Mesafesi Hesaplama

- İki dizgi arasındaki minimum düzenleme işlemi sayısı olarak hesaplanır.
- Dinamik programlama yöntemiyle hesaplanır.
- İki dizgi arasındaki karakterlerin karşılaştırılması ve işlem maliyetlerinin belirlenmesi ile yapılır.

Levenshtein



H	O		N	D	A	
H	Y	U	N	D	A	I

H	Y	U	N	D	A	I
H		O	N	D	A	

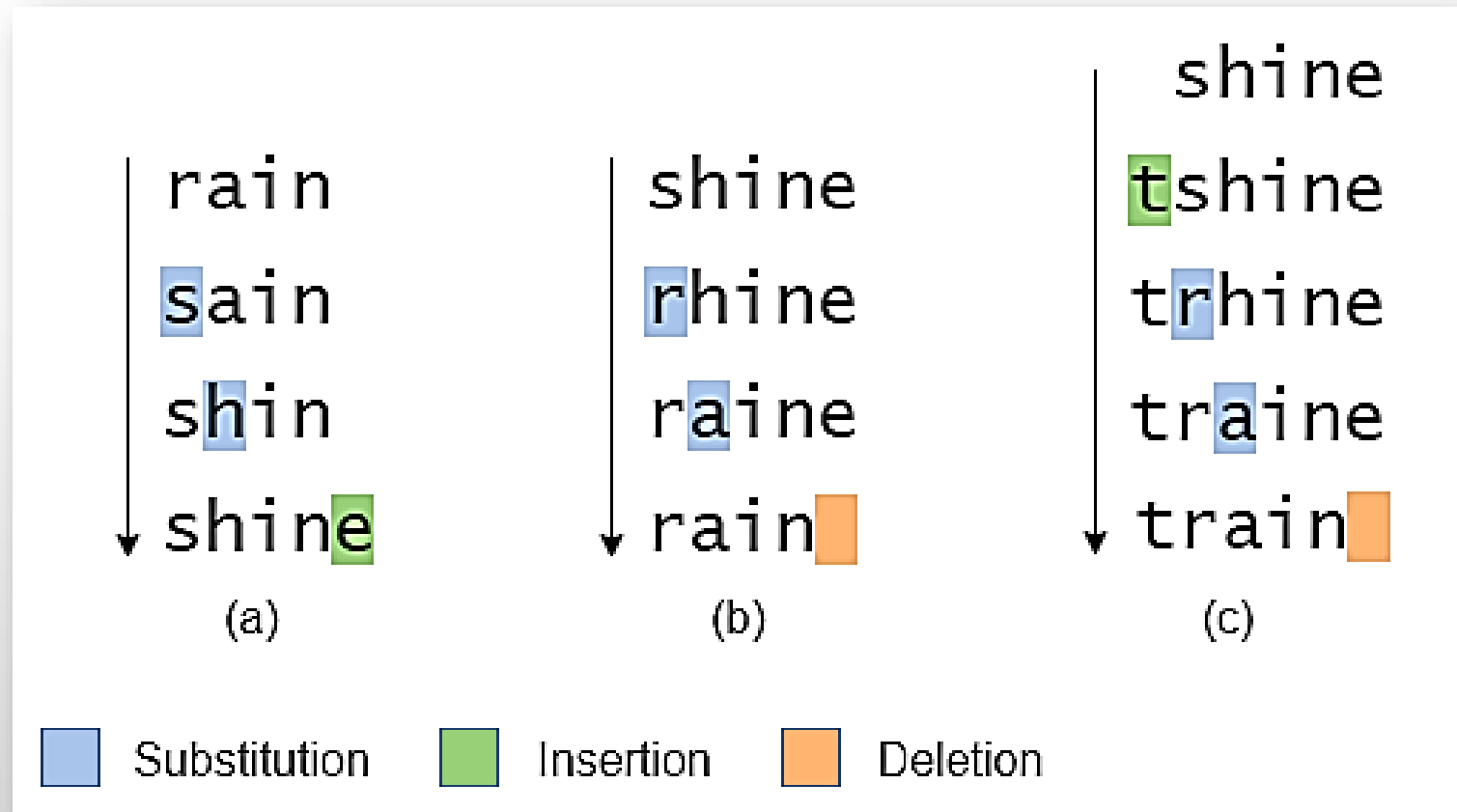
Levenshtein



	""	P	A	I	R	S
""	0	1	2	3	4	5
C	1	1	2	3	4	5
A	2	2	1	2	3	4
R	3	3	2	2	2	3
S	4	4	3	3	3	2



Levenshtein







En Uzun Ortak Alt Dizgi (LCS)

- İki dizgi içinde sıralı olarak bulunan ve mümkün olan en uzun dizgidir.
- Bu alt dizgi, dizgilerin karakter sırasını bozmaz ancak ardışık olmak zorunda değildir.
- İki dizgi arasındaki benzerlik seviyesini ölçer.
- Dinamik programlama yöntemiyle hesaplanır.
- İki dizgi arasındaki karakterlerin sıralı şekilde eşleştirilmesiyle elde edilir.



Örnek

- Dizi 1: AGGTAB
- Dizi 2: GXTXAYB
- LCS: GTAB



Dinamik Programlama Yaklaşımı

- İki boyutlu bir tablo oluşturulur.
- Her hücre, alt dizinin o noktaya kadar olan LCS uzunluğunu temsil eder.
- Tablonun sonunda LCS uzunluğu bulunur.



Algoritma

- İki dizinin uzunlukları m ve n olsun.
- $(m+1) \times (n+1)$ boyutunda bir tablo oluşturulur.
- İlk satır ve sütun sıfır ile doldurulur.
- $A[i] == B[j]$ ise, hücre değeri üst-sol köşedeki değerin 1 fazlasıdır.
- $A[i] != B[j]$ ise, hücre değeri üst veya sol hücrenin maksimum değeri olur.
- Tablonun son hücresi, LCS uzunluğunu verir.



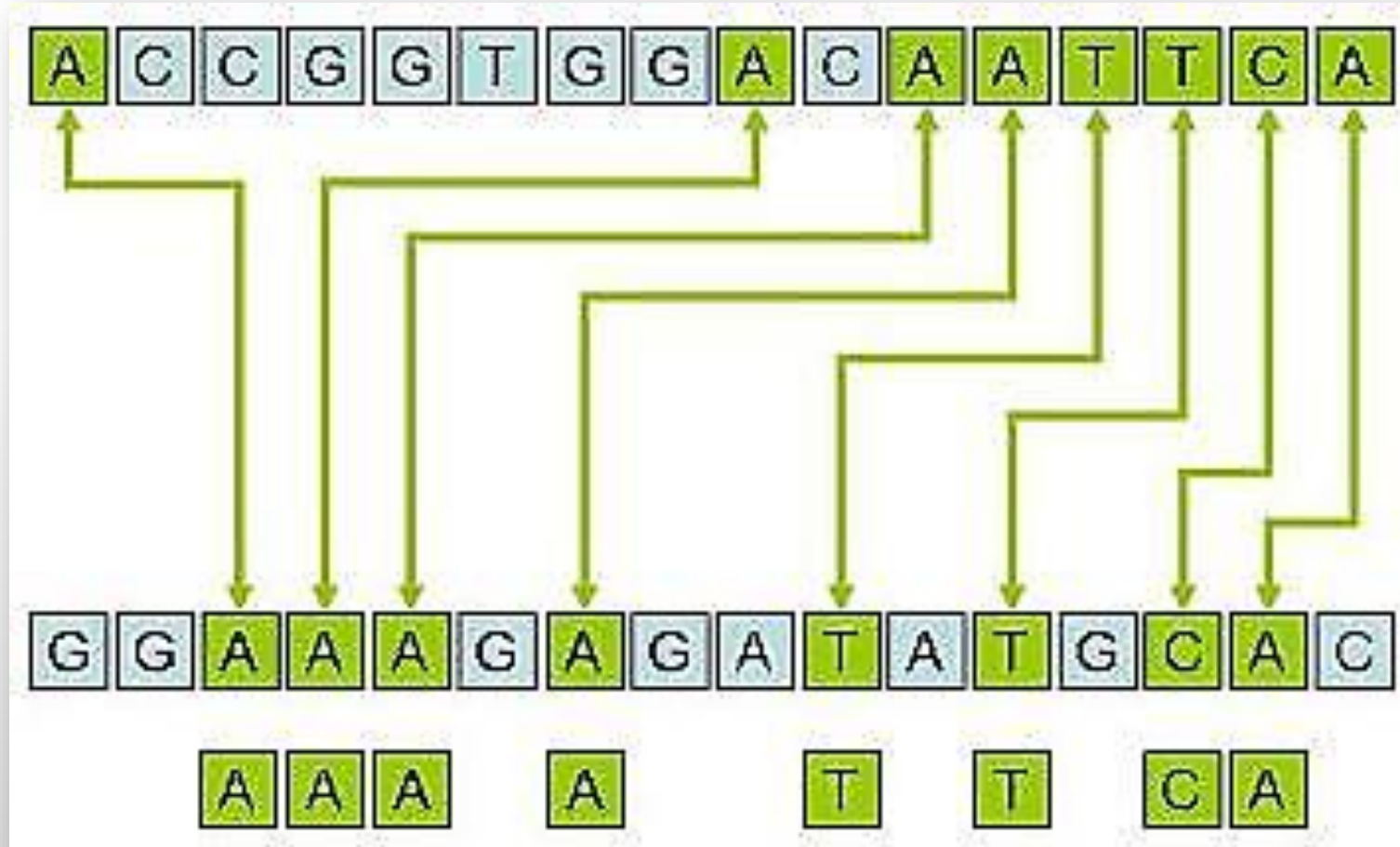
En Uzun Ortak Alt Dizi (LCS)

string 1	a	c	b	a	e	d
string 2	a	b	c	a	d	f

LCS: "acad" with length 4



En Uzun Ortak Alt Dizi (LCS)





SON