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 karşılaşıyoruz.
- 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 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 Ayrıştırma (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 ayrıştırma 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 önişlem adımı olarak kullanılır.

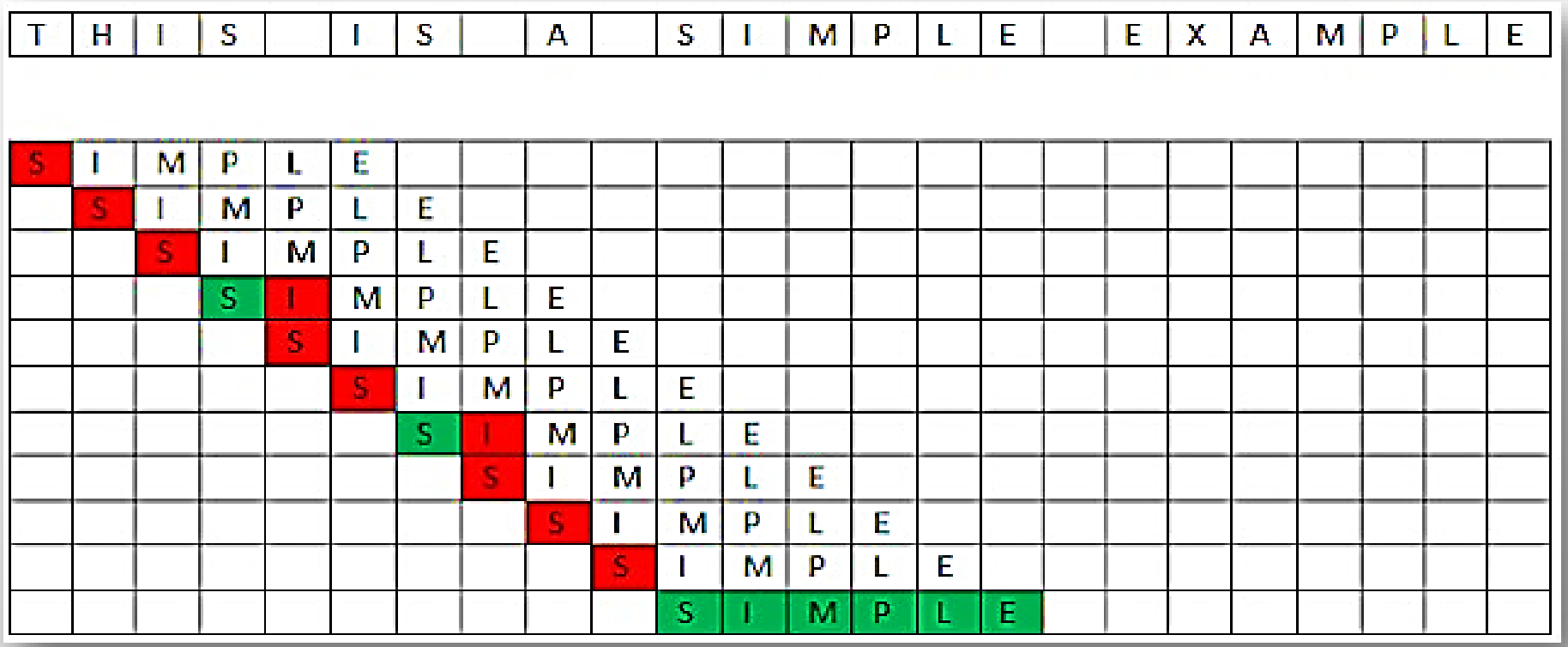
String Naive Algoritması

- Bir metin içinde belirli bir örüntüyü (*pattern*) arayan basit bir algoritma.
- Naive (Saf) olarak adlandırılır çünkü basit bir yaklaşım kullanır.
- Ortalama ve en kötü durumda *O(m n)* zaman karmaşıklığına sahiptir.
  - (m: örüntü uzunluğu, n: metin uzunluğu)

İşleyiş

- Metindeki her konum için örüntünün ilk karakterinin eşleşip eşleşmediğini kontrol edilir.
- Eşleşen karakterlerin tümü için örüntünün tam olarak eşleşip eşleşmediğini kontrol edilir.
  - Eşleşme Durumu: eşleşme pozisyonu rapor edilir.
  - Eşleşmeme Durumu: sonraki konumlar kontrol edilir.
- Metindeki tüm konumlar için adımlar tekrarlanır.

Brute Force - Naive



Brute Force - Naive

Brute Force - Naive

Brute Force - Naive

Brute Force - Naive

Brute Force - Naive

Brute Force - Naive

Brute Force - Naive

Brute Force - Naive

Brute Force - Naive

Brute Force - Naive

Knuth-Morris-Pratt (KMP) Algoritması

- Bir metin içinde belirli bir örüntüyü bulmak için kullanılır.
- *Donald Knuth, Vaughan Pratt* ve *James H. Morris* tarafından geliştirilmiştir.
- Ortalama ve en kötü durumda *O(m + n)* zaman karmaşıklığına sahiptir.
  - (m: örüntü uzunluğu, n: metin uzunluğu)

İşleyiş

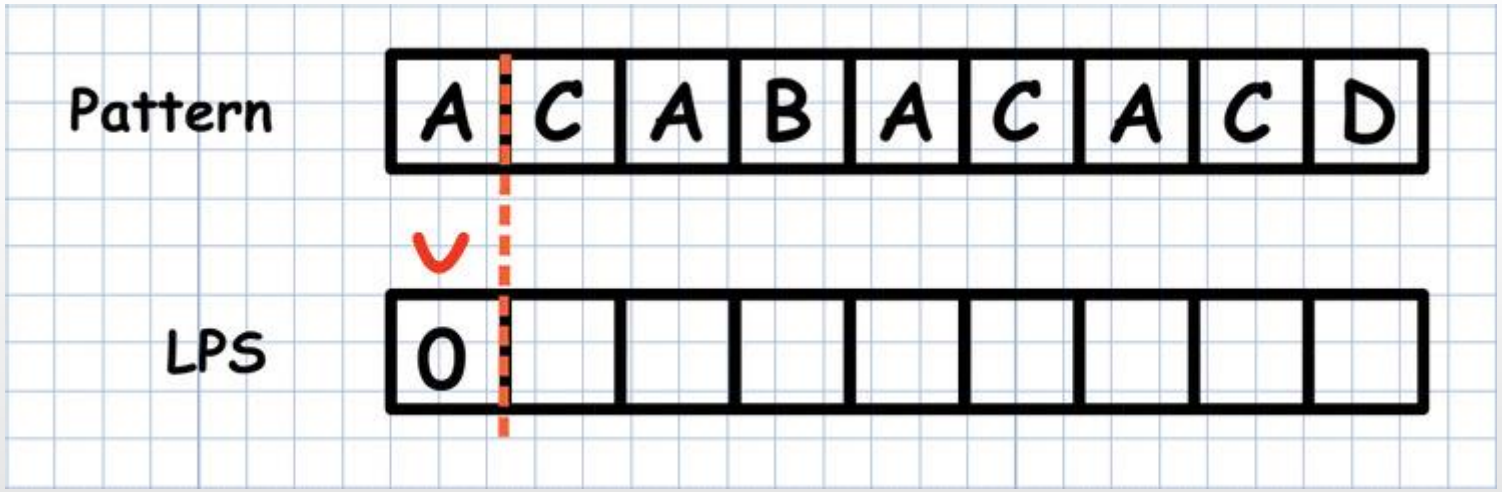
- Ön İşleme: Örüntü içindeki her karakter için,
  - eşleşme durumunda geri dönülecek pozisyonları belirleyen,
  - en uzun önek-suffix eşleşmesini bulan bir tablo oluşturulur.
- Örüntü Arama: Metin içinde arama yapılırken,
  - örüntü ile eşleşmeyen karakterlerde geri dönülecek pozisyonlar,
  - tablodan elde edilen geri dönüş değerleri kullanılarak hesaplanır.
- Eşleşme Kontrolü: Eşleşen karakterlerin tümü için örüntünün tam olarak eşleşip eşleşmediği kontrol edilir.

Örnek

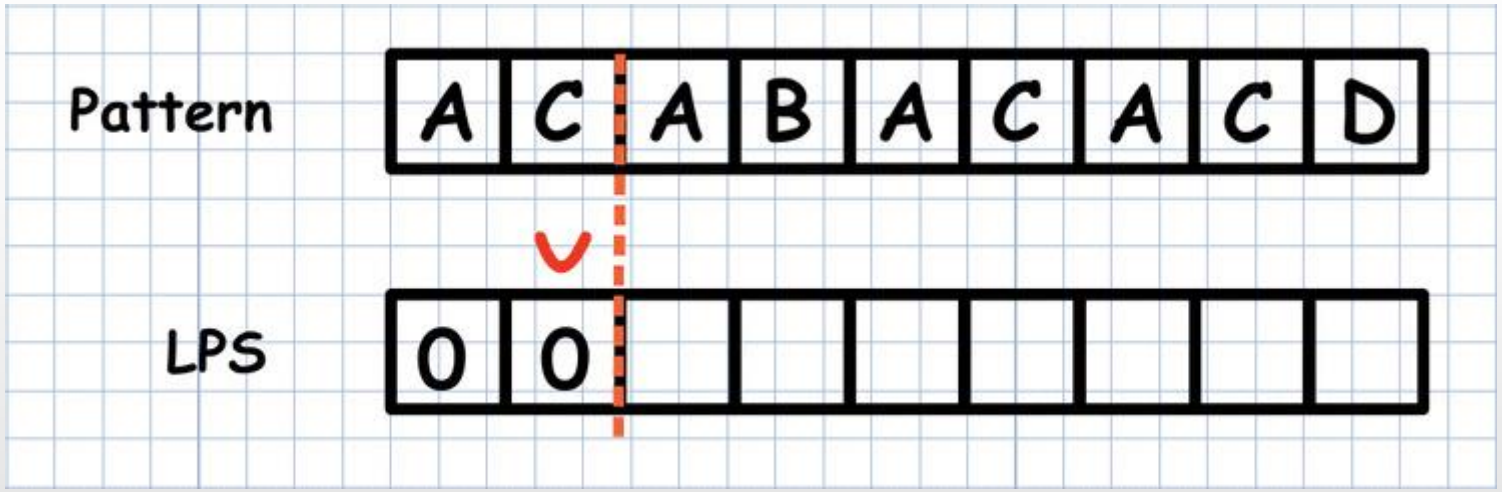
- Metin: "ababcababcababababd"
- Örüntü: "ababd"
- Örüntü Tablosu:
  - a: 0, b: 0, a: 1, b: 2, d: 0

Knuth Morris Pratt

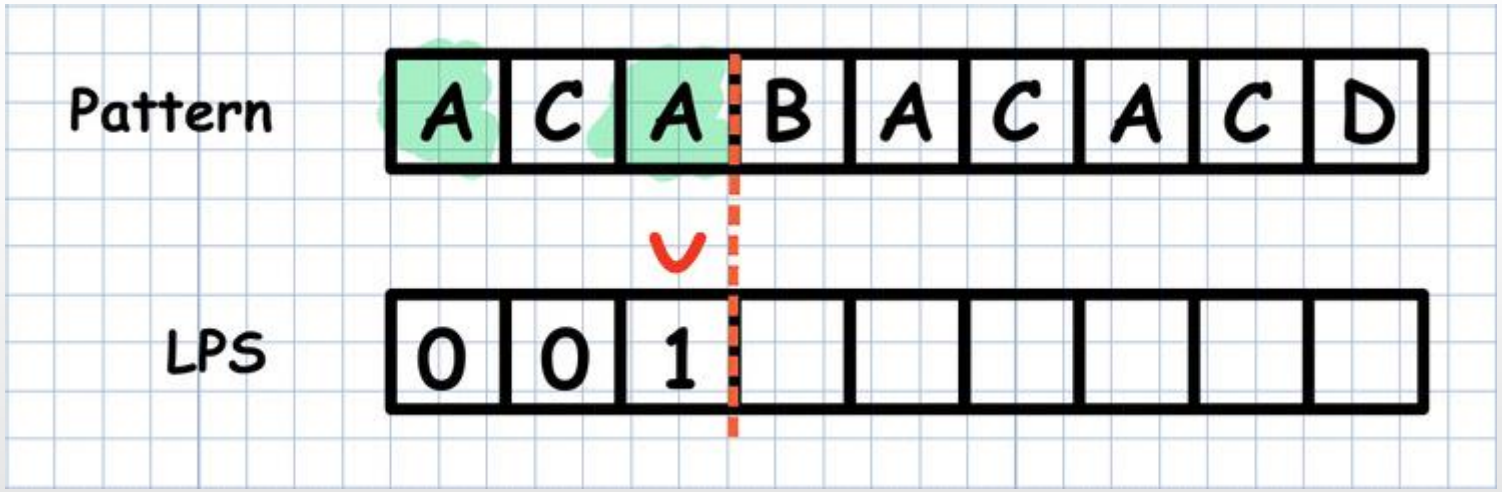
- *Longest Proper Prefix*



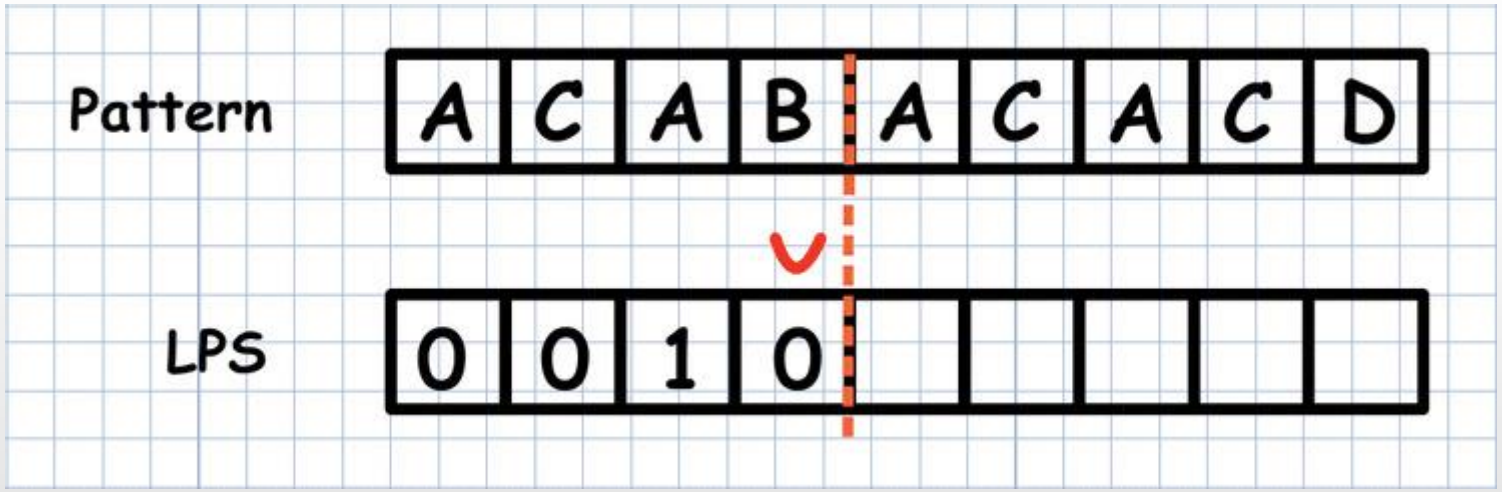
Knuth Morris Pratt



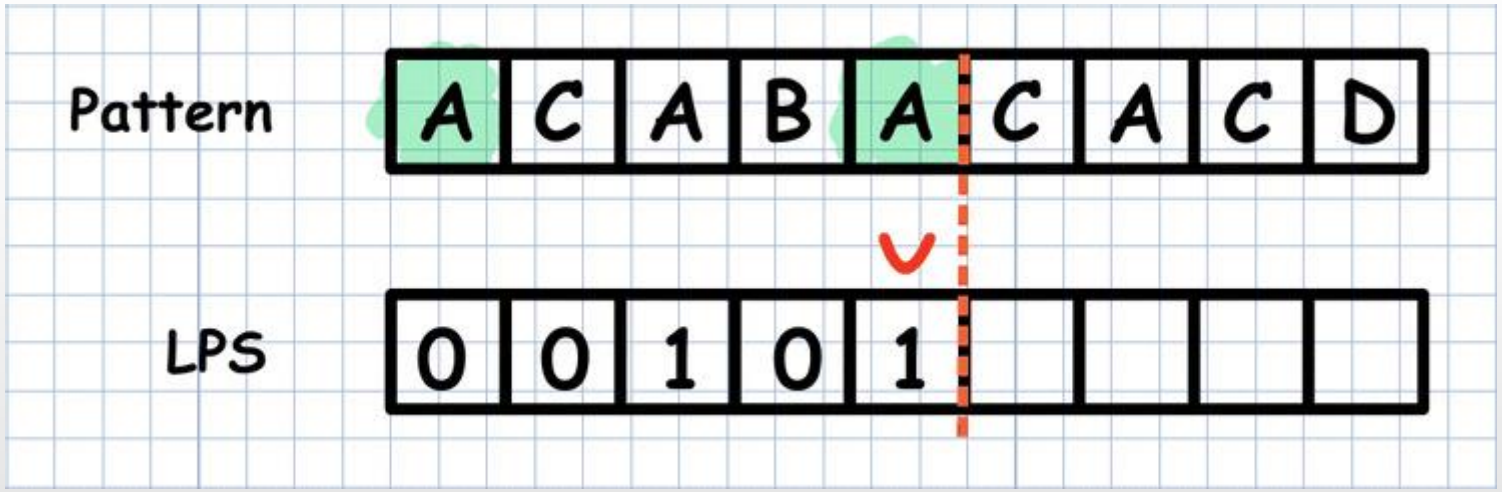
Knuth Morris Pratt



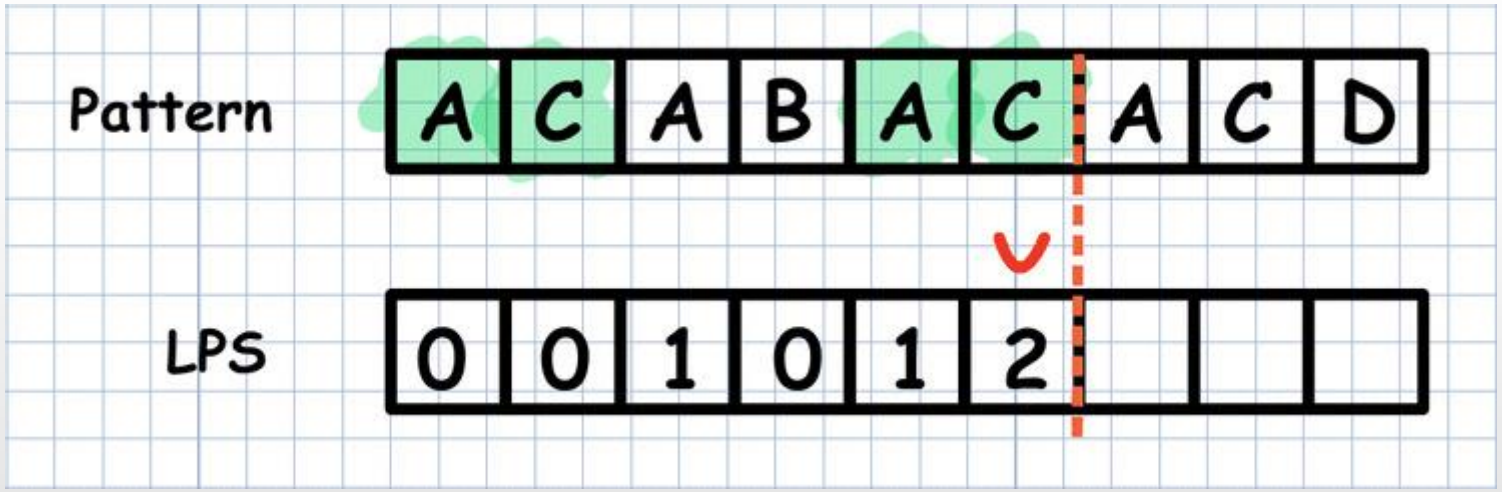
Knuth Morris Pratt



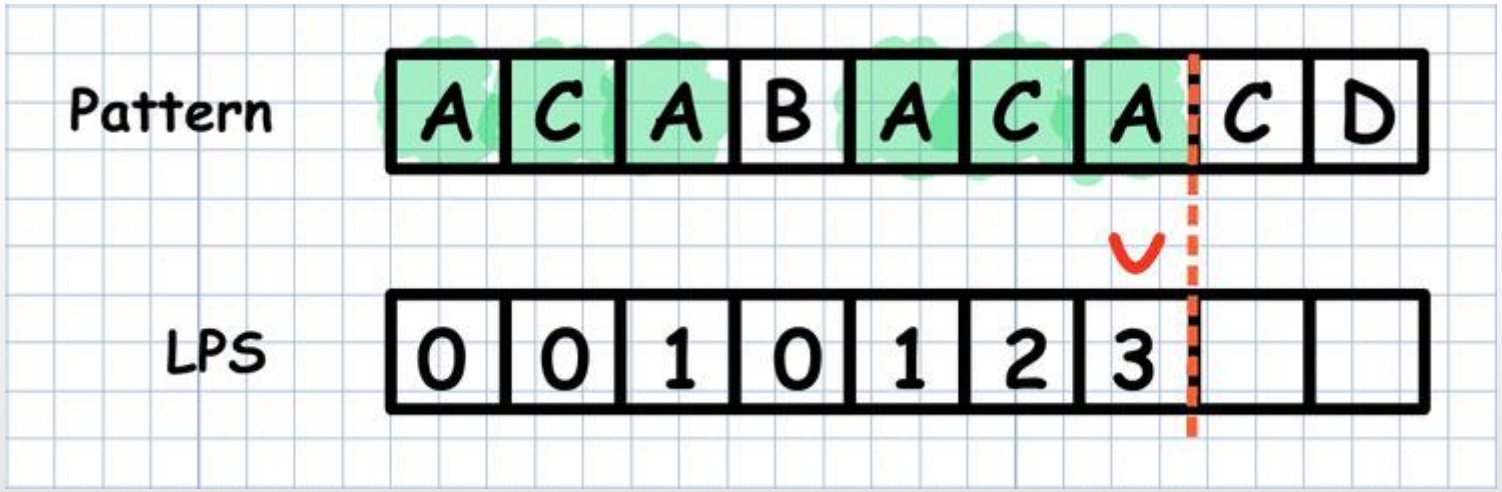
Knuth Morris Pratt



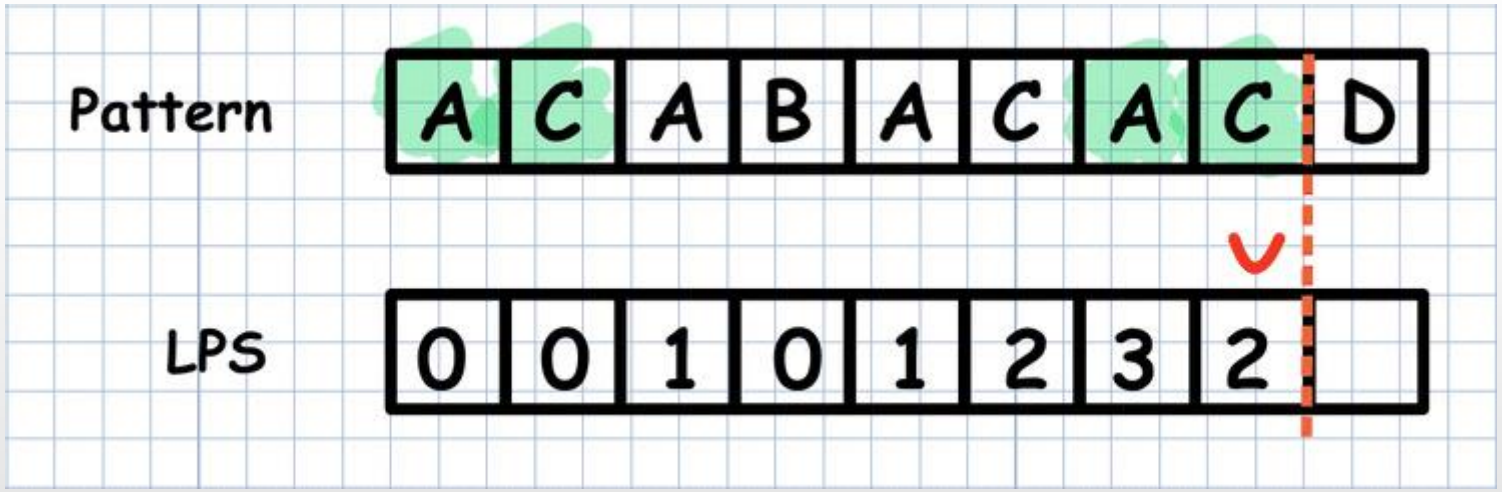
Knuth Morris Pratt



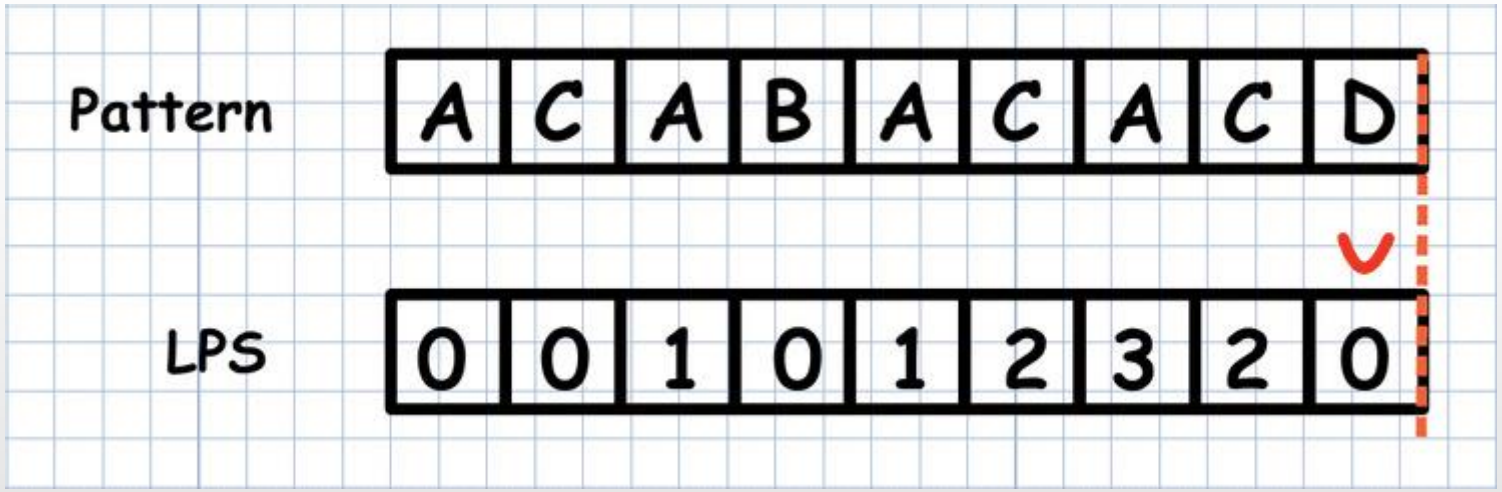
Knuth Morris Pratt



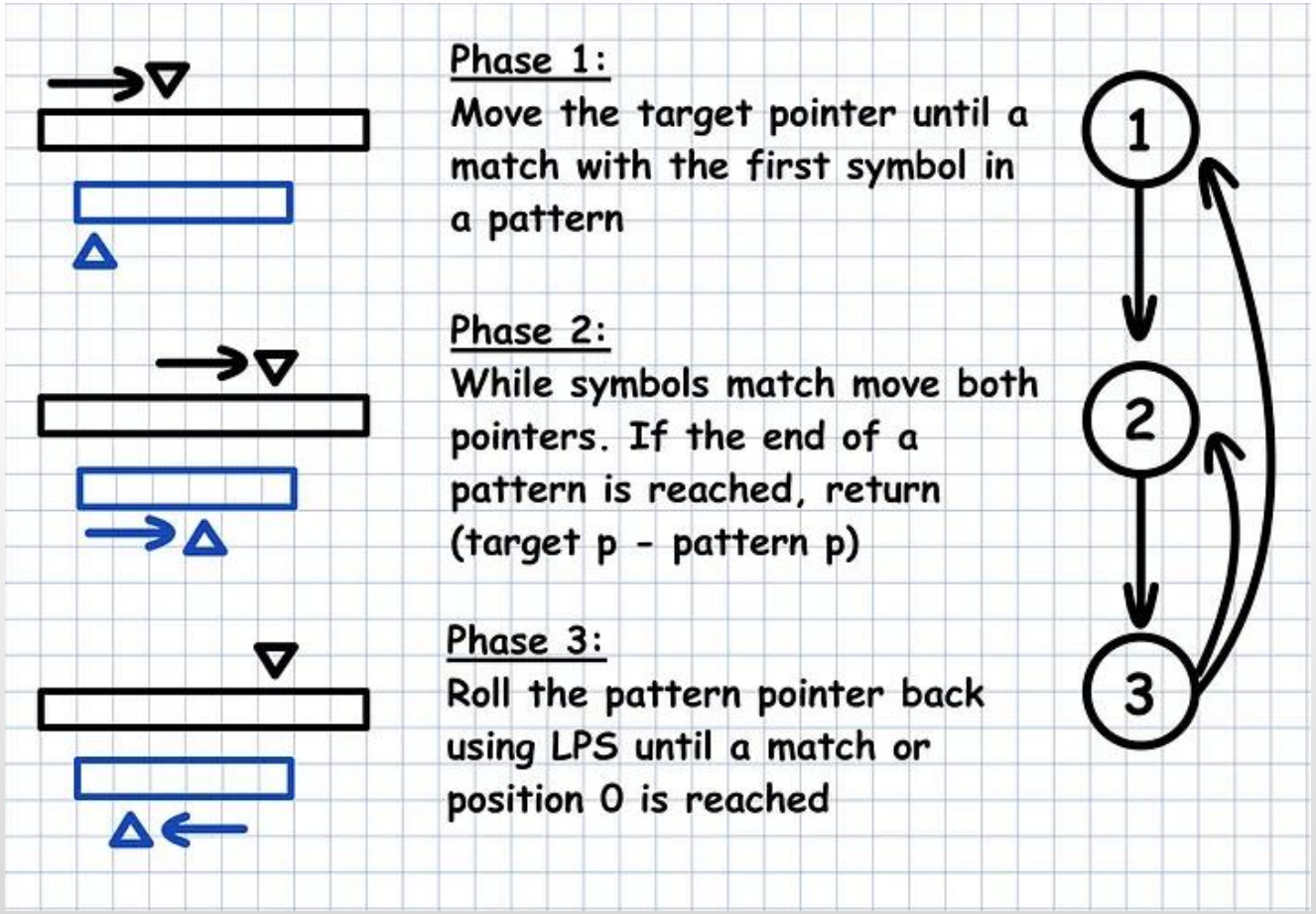
Knuth Morris Pratt



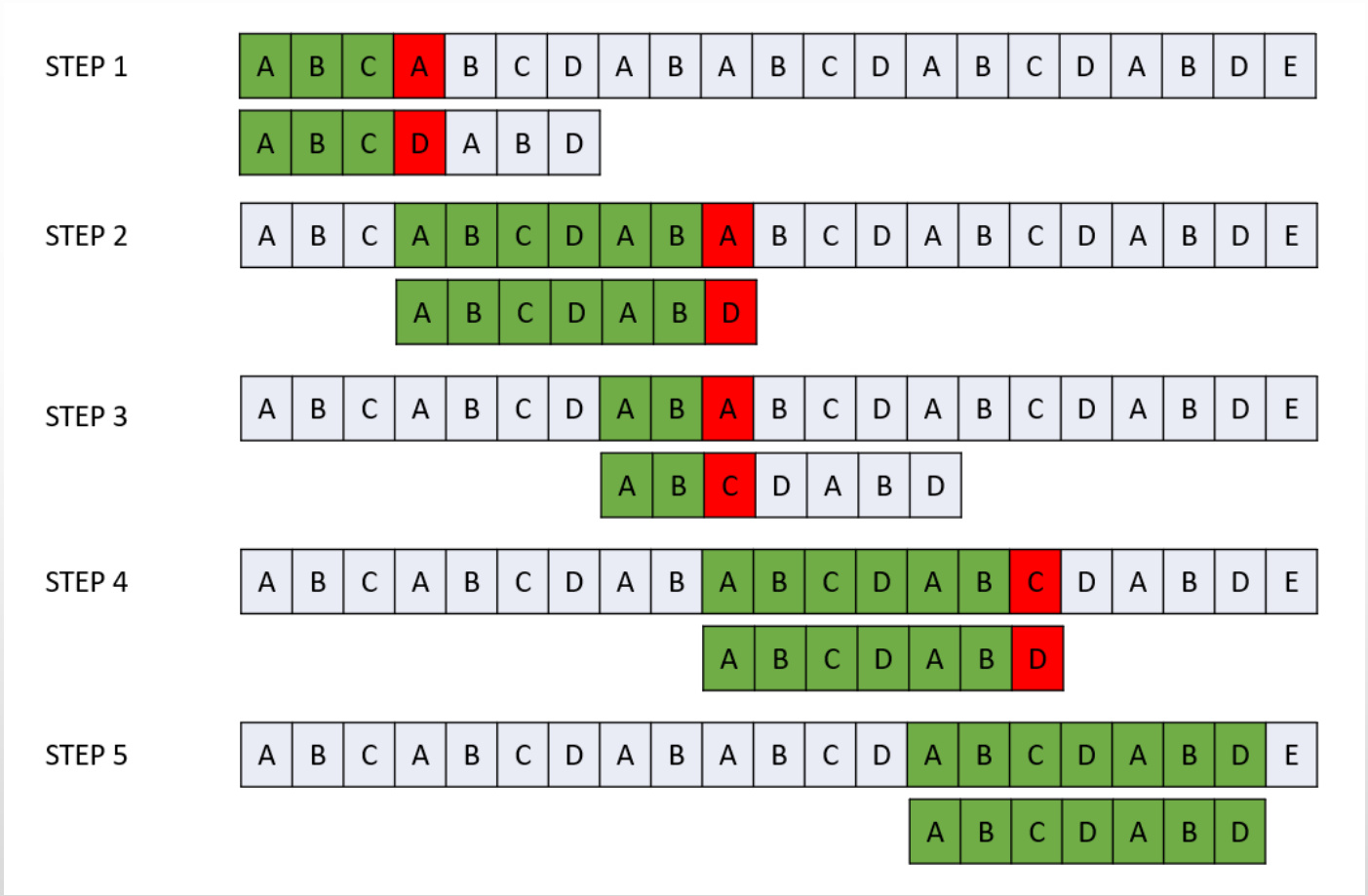
Knuth Morris Pratt



Knuth Morris Pratt



Knuth Morris Pratt



Boyer-Moore Algoritması

- Bir metin içinde belirli bir örüntüyü arar.
- *Robert S. Boyer* ve *J Strother Moore* tarafından geliştirilmiştir.
- Ortalama ve en kötü durumda *O(n/m)* zaman karmaşıklığına sahiptir.
  - (n: metin uzunluğu, m: desen uzunluğu)

İşleyiş

- Ön İşleme: Örüntü içindeki her karakter için eşleşme durumunda geri dönülecek pozisyonları belirleyen bir tablo oluşturulur.
- Arama: Metin içinde örüntüyü ararken, eşleşmeyen karakterlerde tablodan yararlanarak geri dönülecek pozisyonlar hesaplanır.
- Eşleşme Kontrolü: Eşleşen karakterlerin tümü için örüntünün tam olarak eşleşip eşleşmediği kontrol edilir.
- Kötü Karakter Kaydırma Kuralı: Eşleşmeyen bir karakter varsa, örüntüdeki bu karakterin metindeki en sağdaki konumu baz alınarak kaydırma yapılır.
- İyi Sone Kuralı: Eşleşmeyen bir alt-dizgi varsa, örüntüdeki bu alt-dizginin metindeki en sağdaki konumu baz alınarak kaydırma yapılır.

Örnek

- Metin: "abccbaabccbaabcbcabbabcabc"
- Desen: "abcbcabbabcabc"
- Desen Tablosu:
  - a: 10, b: 8, c: 7
- Sonuç:
  - Pozisyon 12: "abcbcabbabcabc"

Boyer Moore

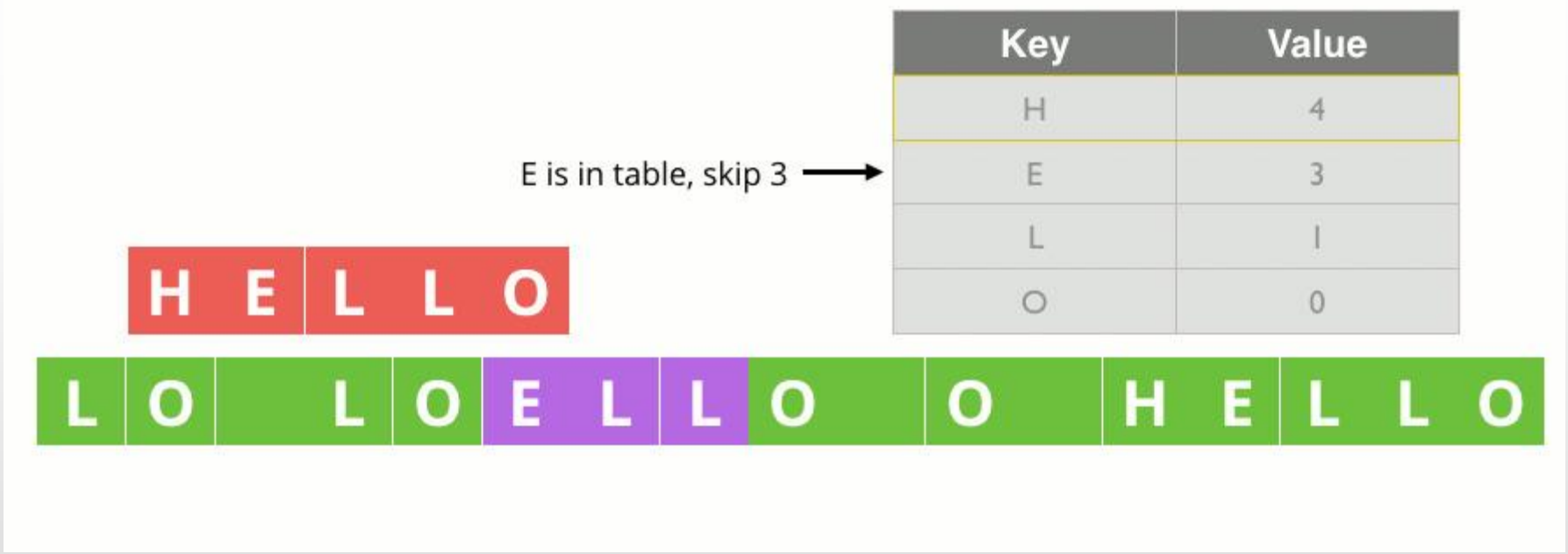
Boyer Moore

Boyer Moore

Boyer Moore

Boyer Moore

Boyer Moore

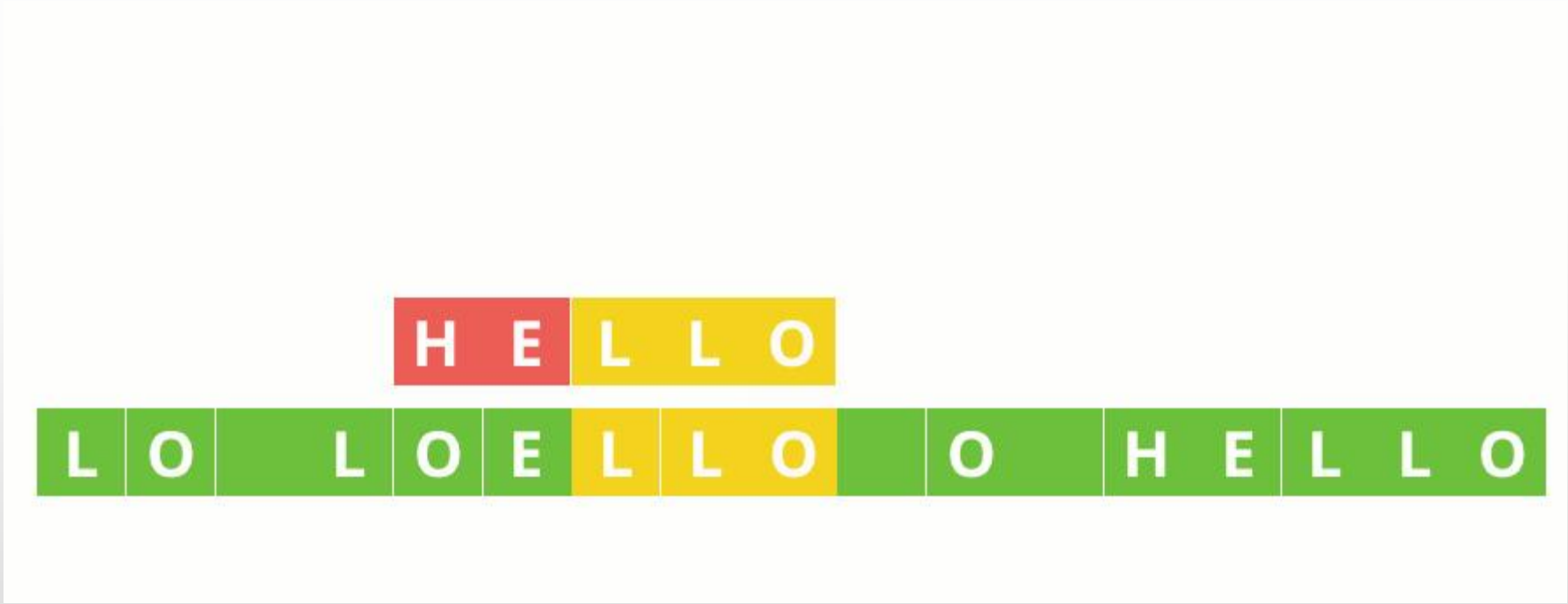


Boyer Moore

Boyer Moore

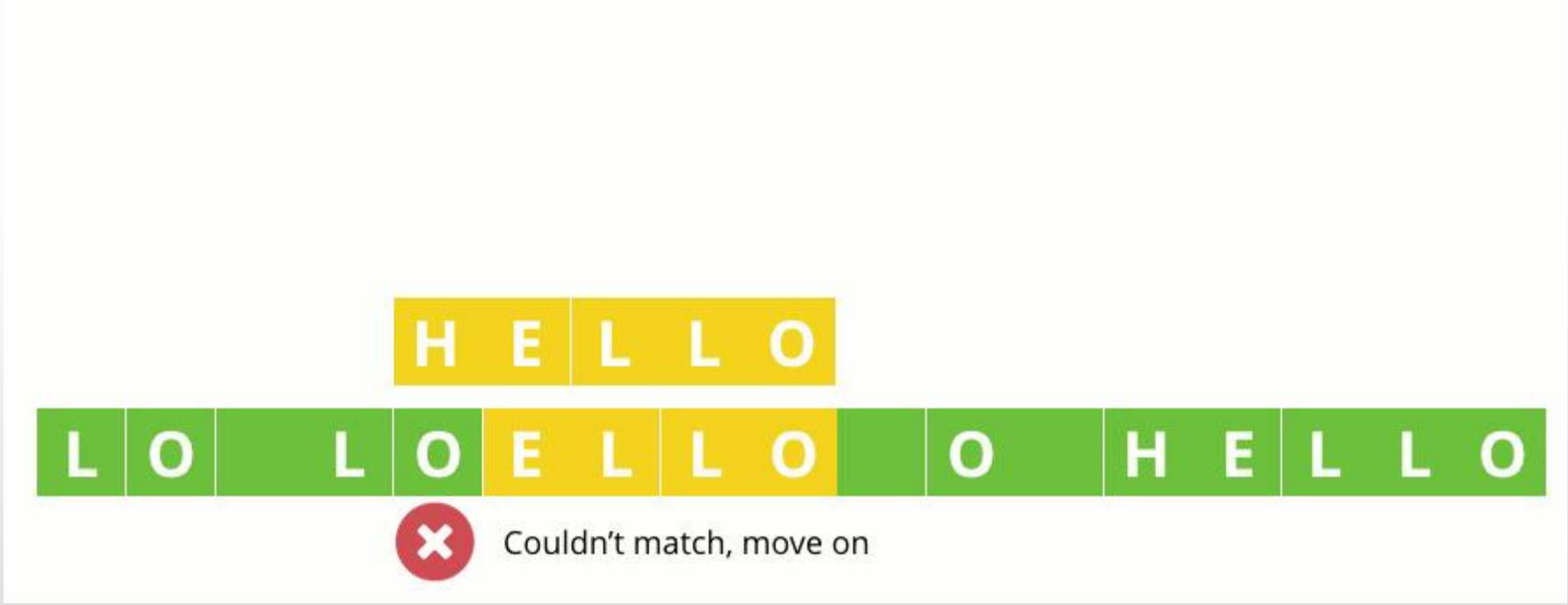
Boyer Moore

Boyer Moore



Boyer Moore

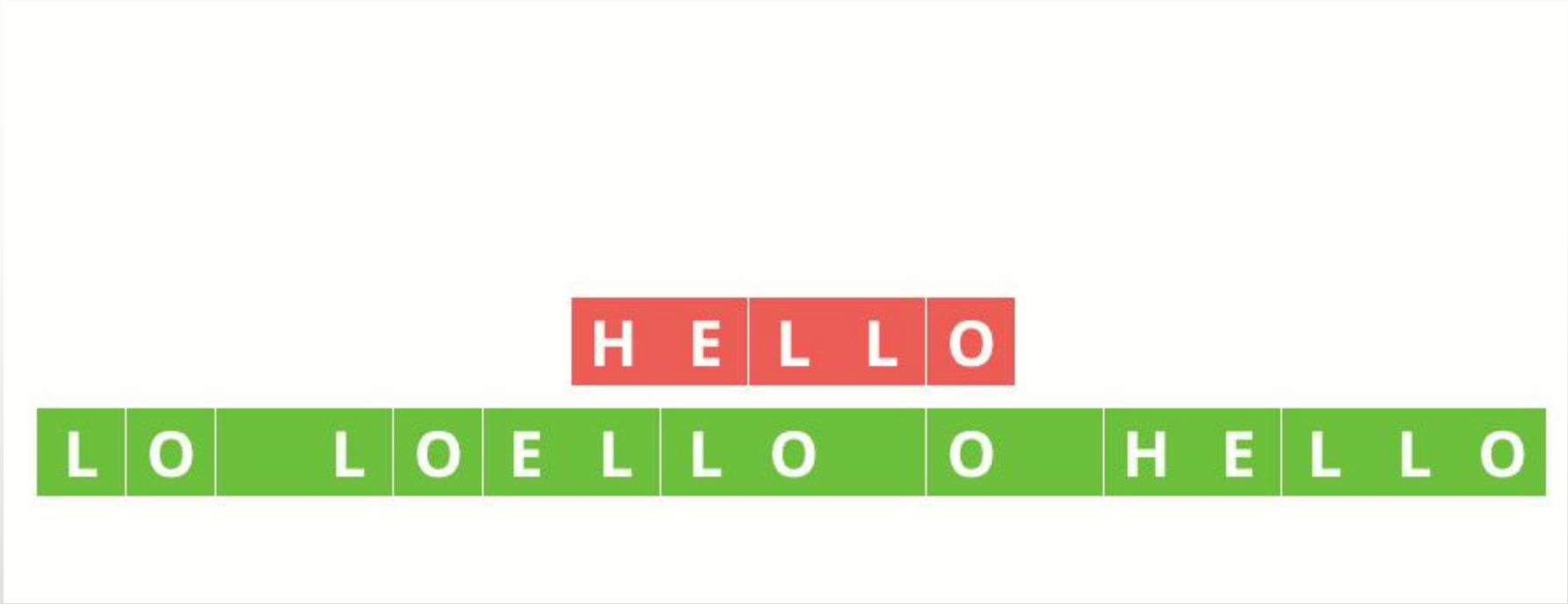
Boyer Moore



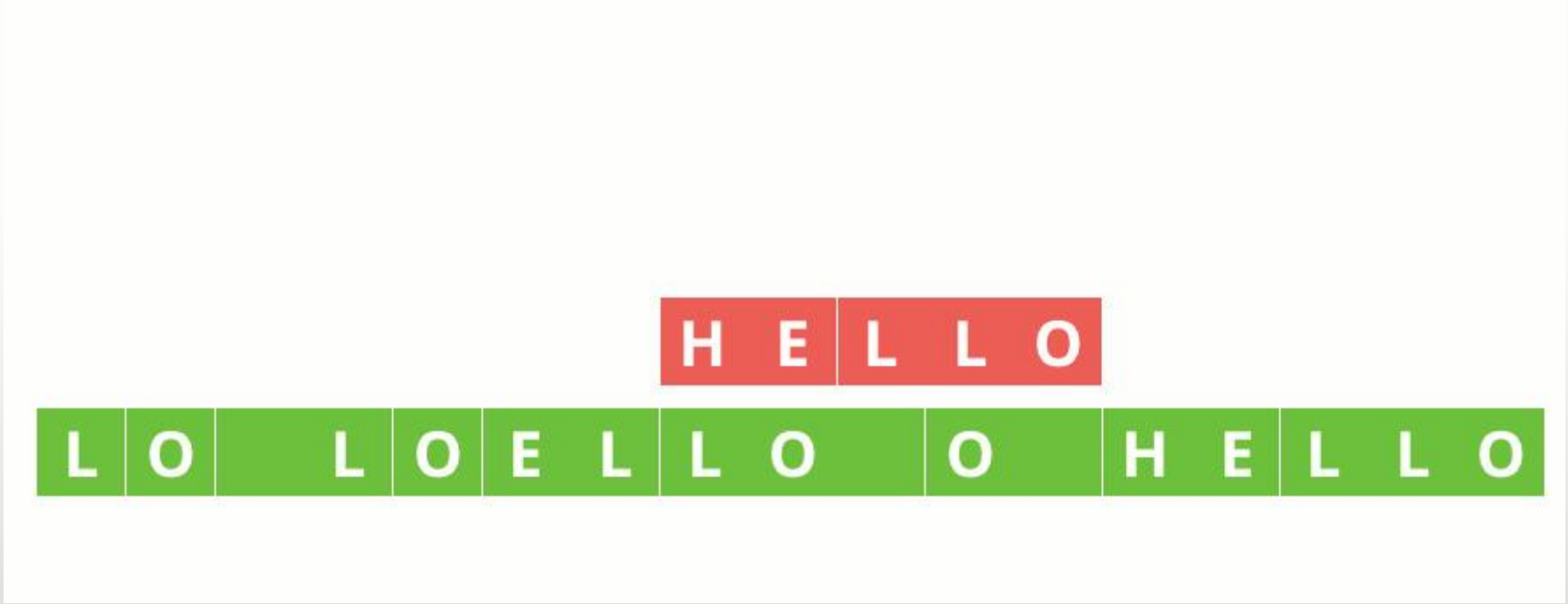
Boyer Moore

Boyer Moore

Boyer Moore

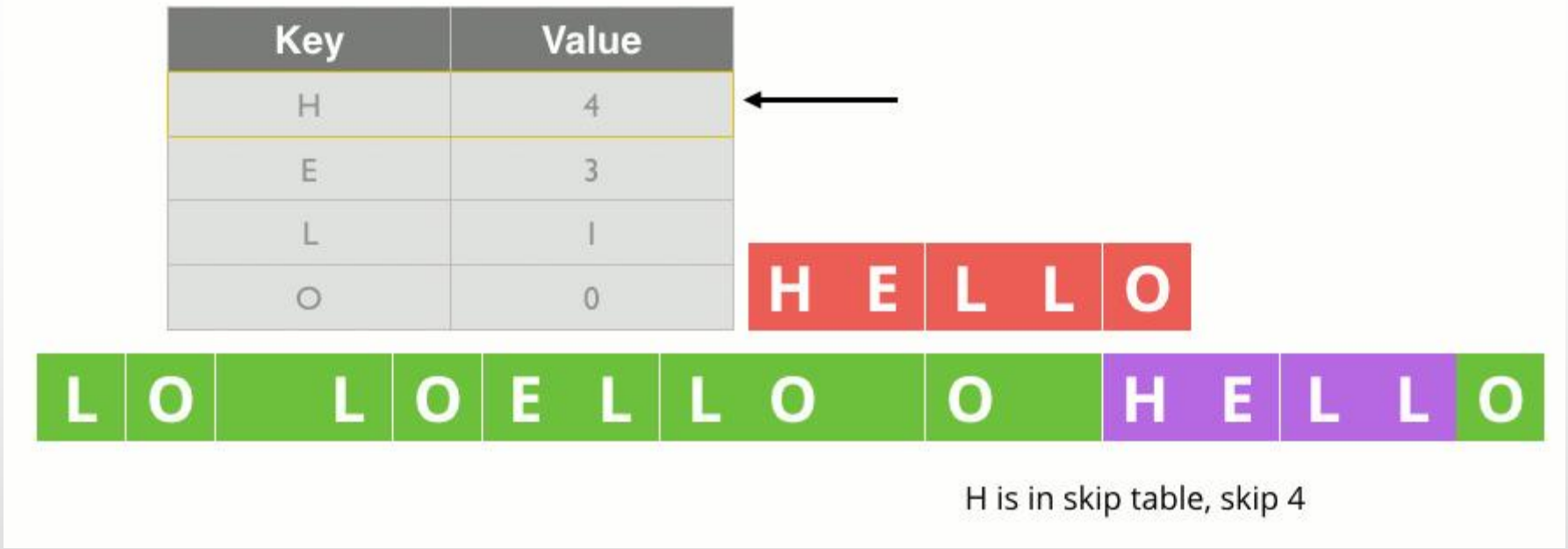


Boyer Moore



Boyer Moore

Boyer Moore



Boyer Moore

Boyer Moore

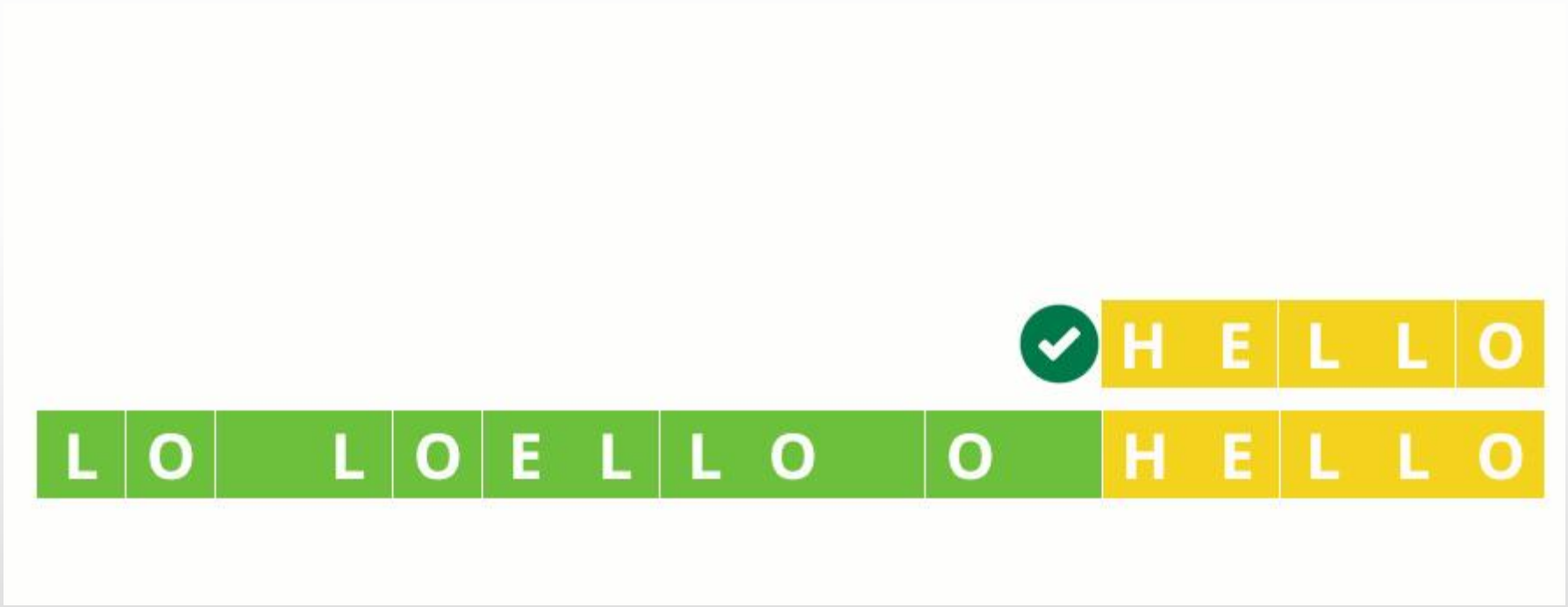
Boyer Moore

Boyer Moore

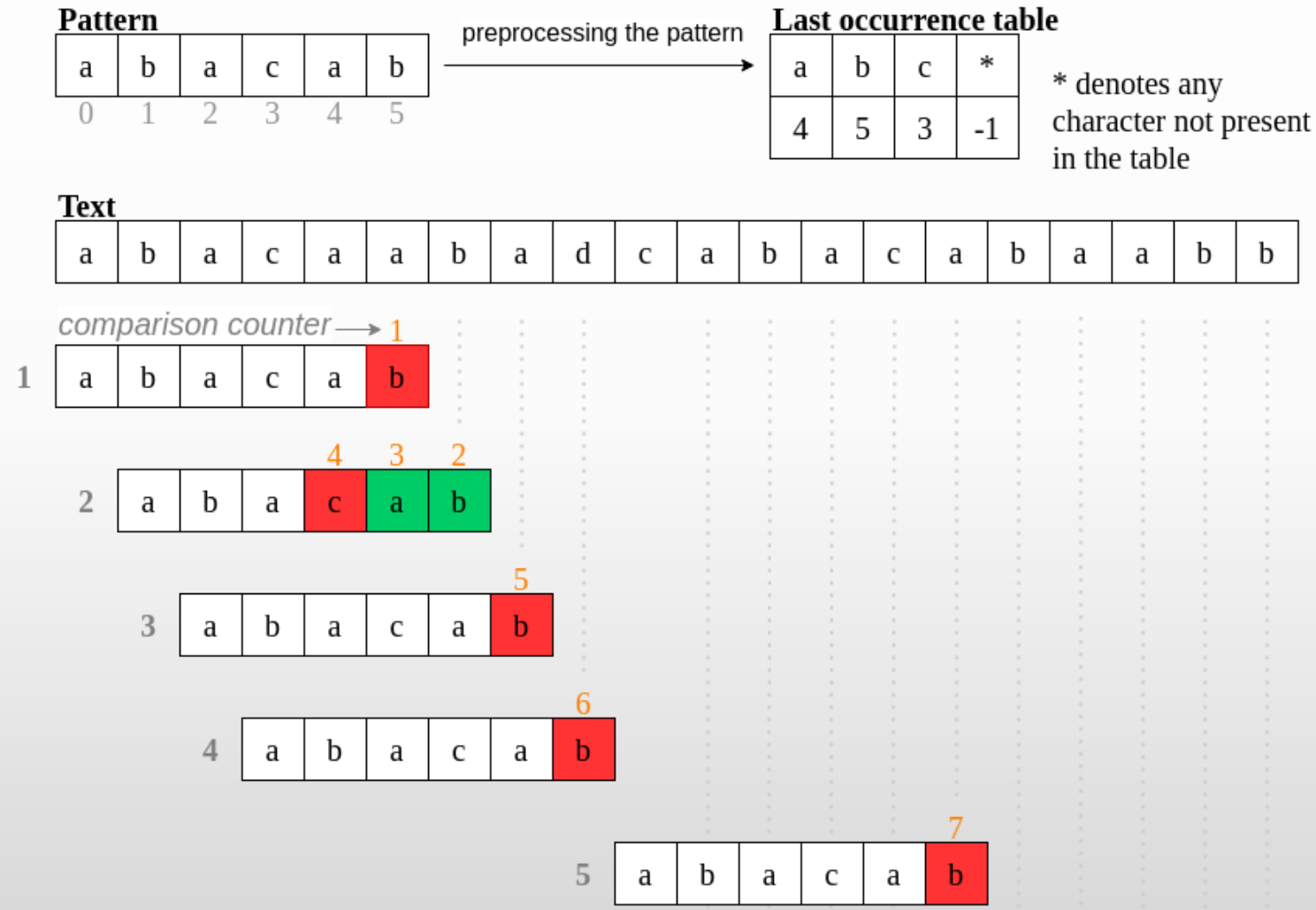
Boyer Moore

Boyer Moore

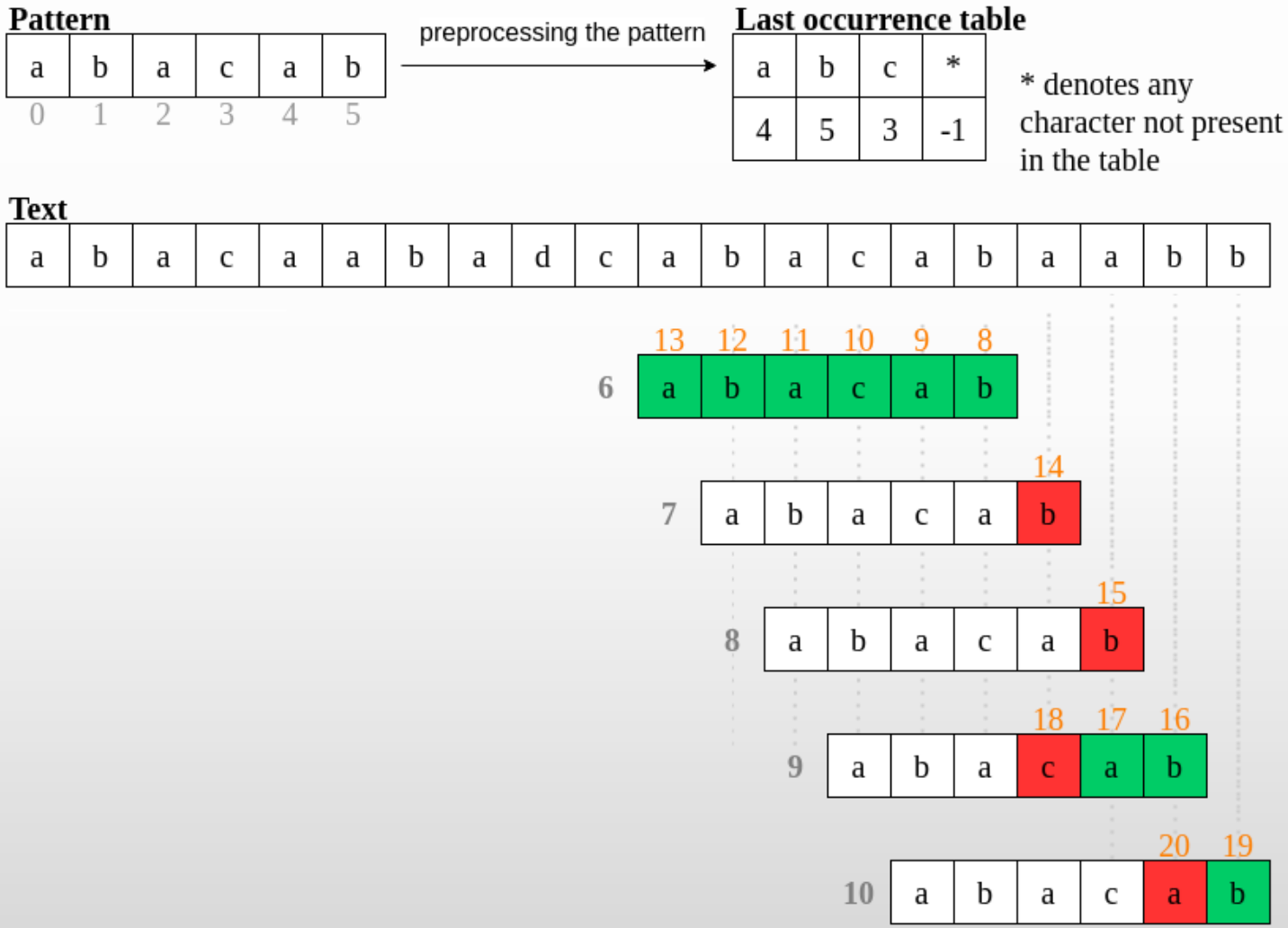
Boyer Moore



Boyer Moore



Boyer Moore



Rabin-Karp Algoritması

- Bir metin içinde belirli bir örüntüyü bulmak için kullanılır.
- *Michael O. Rabin* ve *Richard M. Karp* tarafından geliştirilmiştir.
- Ortalama ve en kötü durumda $O(n + m)$ zaman karmaşıklığına sahiptir.
  - (n: metin uzunluğu, m: örüntü uzunluğu)

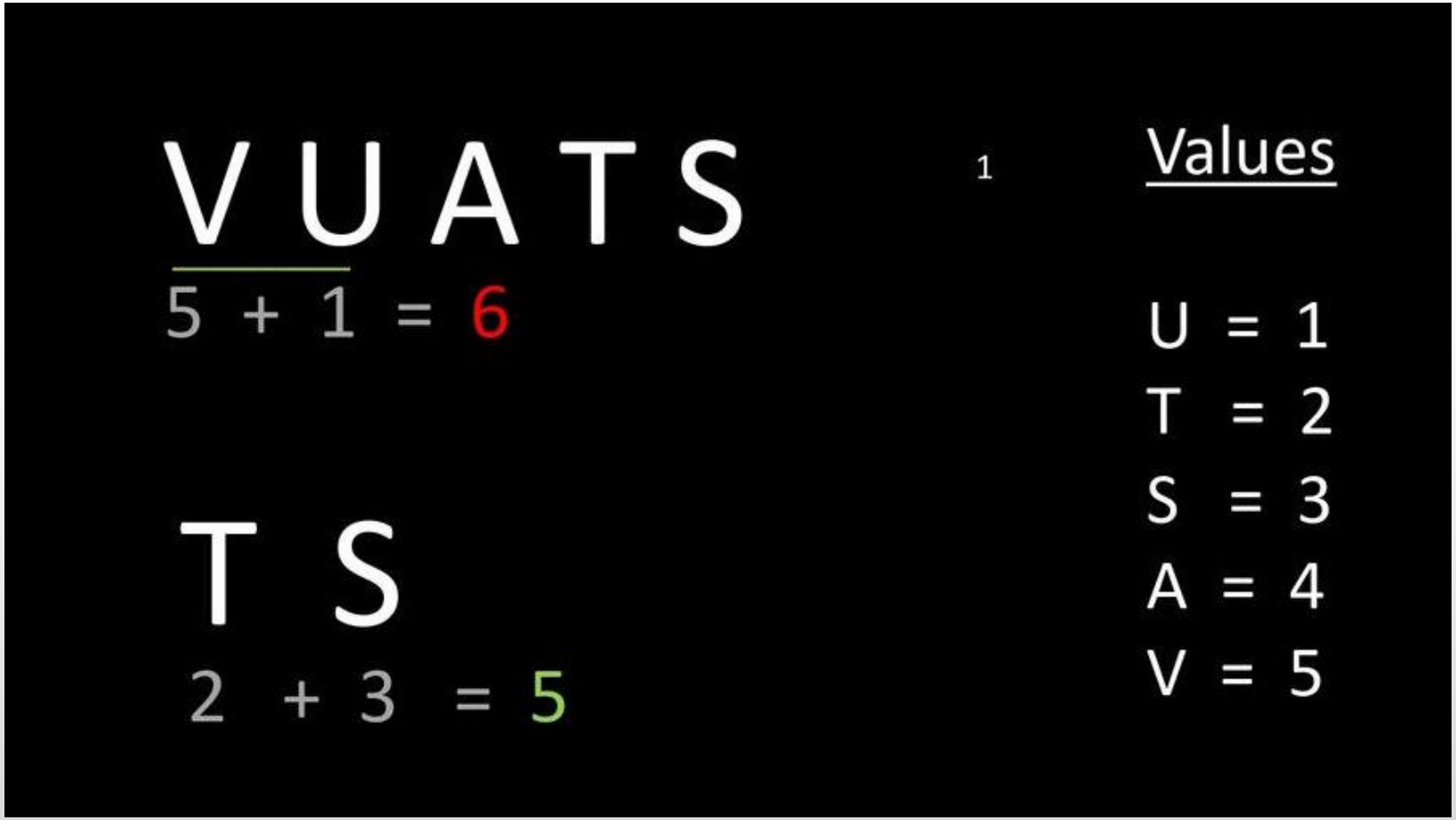
İşleyiş

- Hash: Örüntü ve metin içindeki alt dizgelerin hash değerleri hesaplanır.
- Eşleşme Kontrolü: Hash değerleri eşleşen alt dizgeler karşılaştırılır.
- Doğrulama: Eşleşme olduğunda, karakter bazında doğrulanır.
- Kaydırma ve Yeniden Hesaplama: Eşleşme olmadığında, yeni bir alt dizge seçilir ve hash değeri yeniden hesaplanır.
- Tekrarlama: Tüm metin boyunca adımlar tekrarlanır.

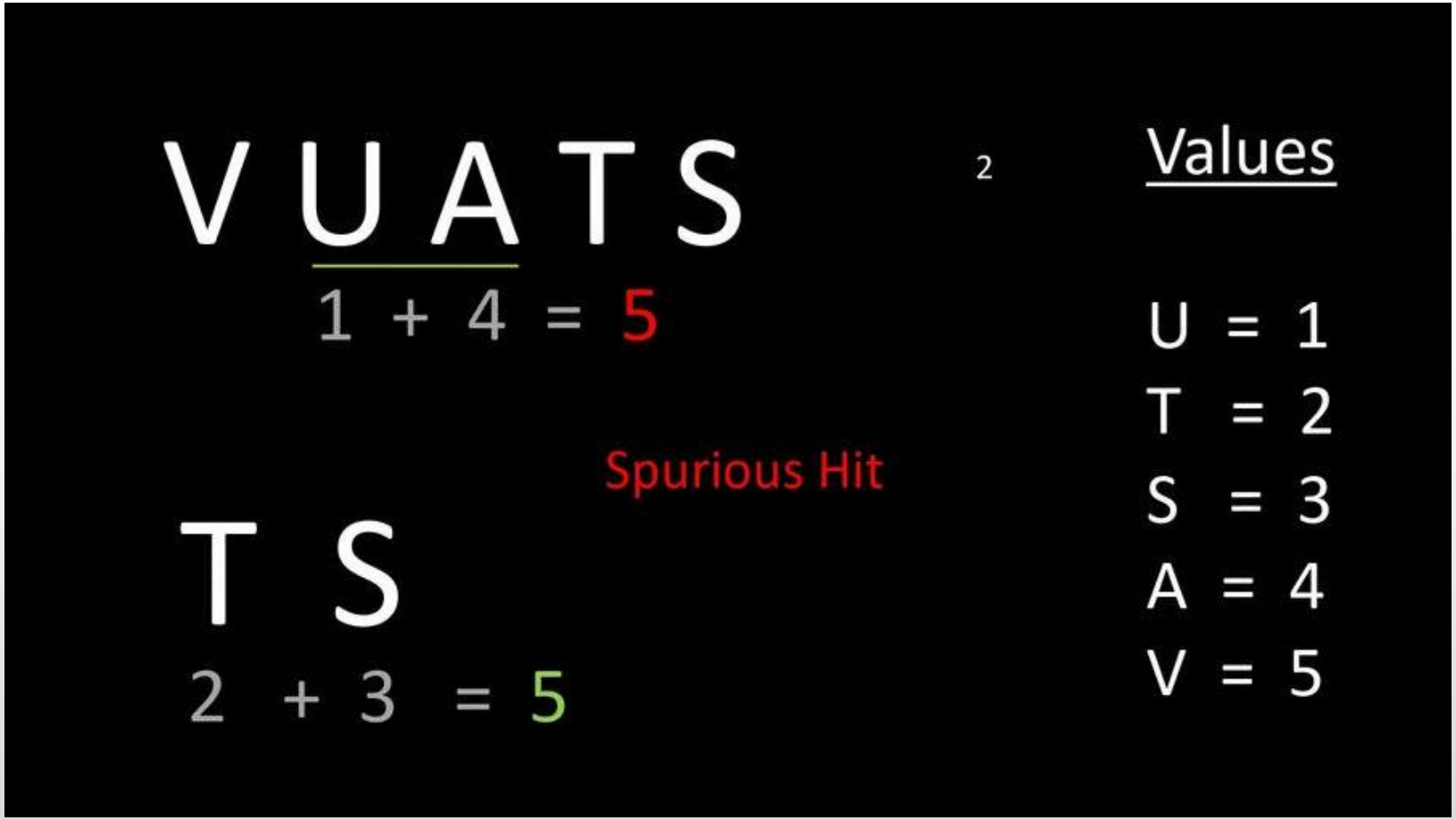
Örnek

- Metin: "abracadabra"
- Örüntü: "cad"
- İşleyiş:
  - Hash Değerleri: Metin: "abr", Örüntü: "cad"
  - Eşleşme Kontrolü: Hash değerleri eşleşmez.
  - Kaydırma ve Yeniden Hesaplama: Yeni alt dizge seçilir: "bra"

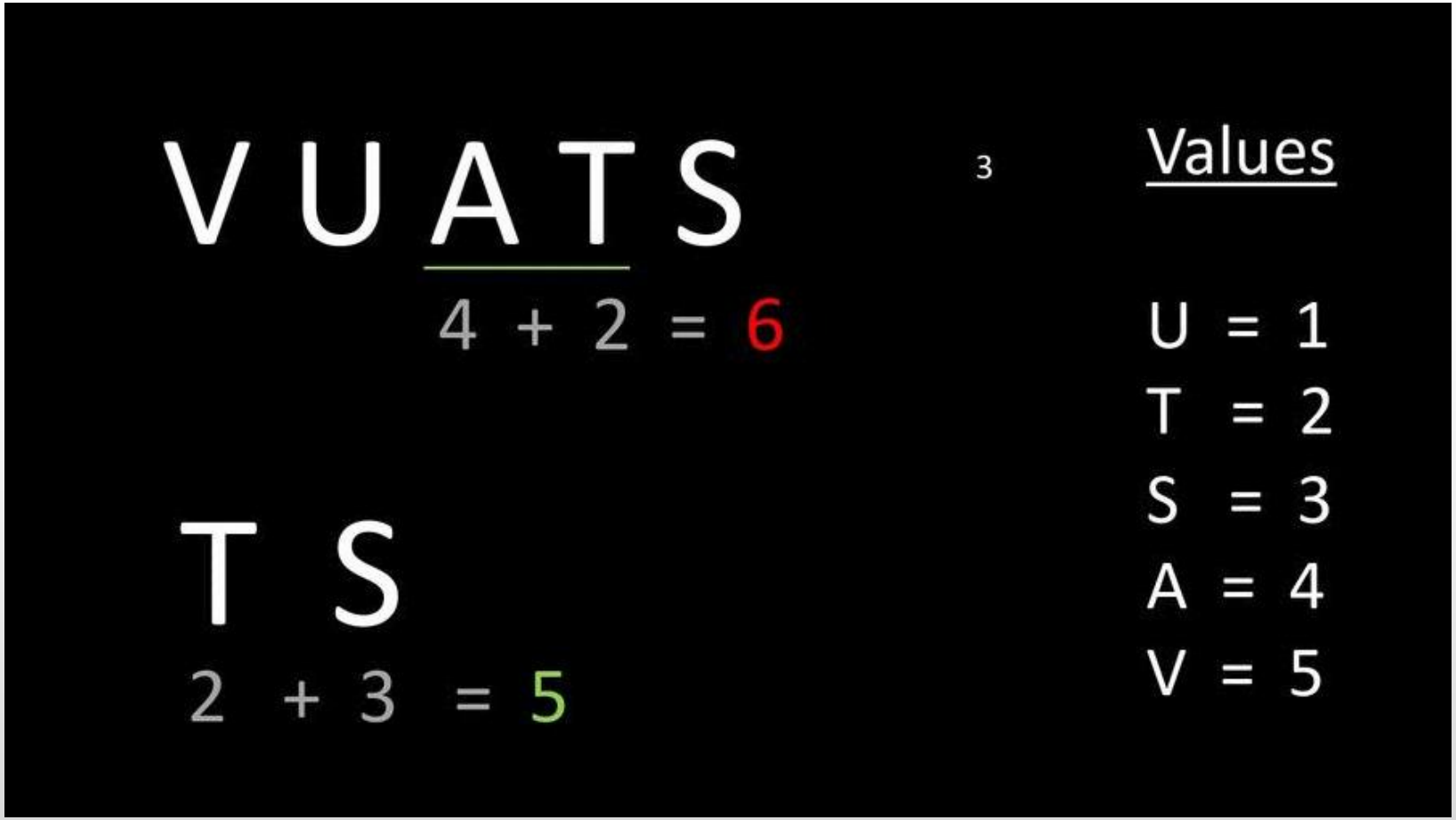
Rabin Karp



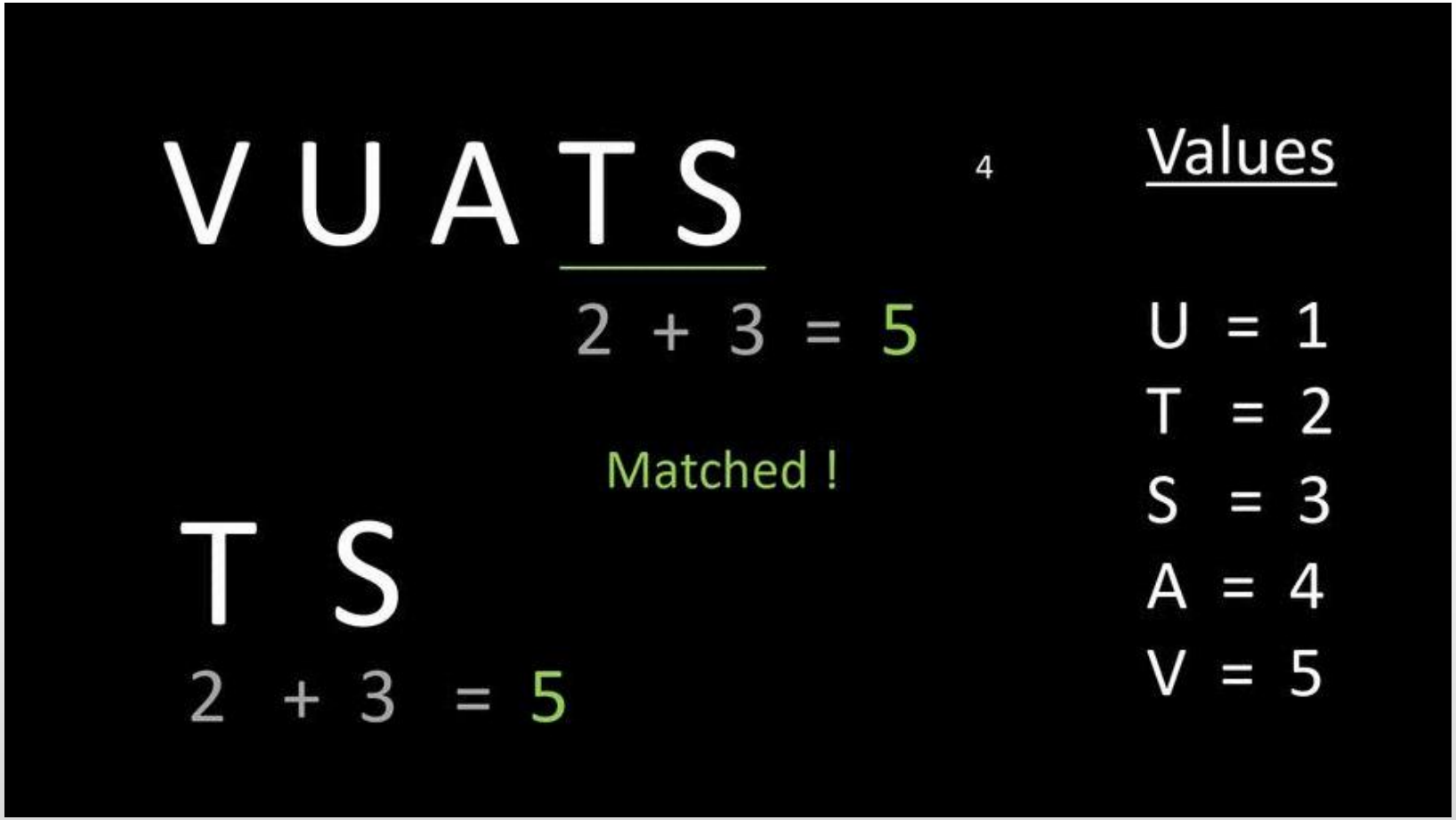
Rabin Karp



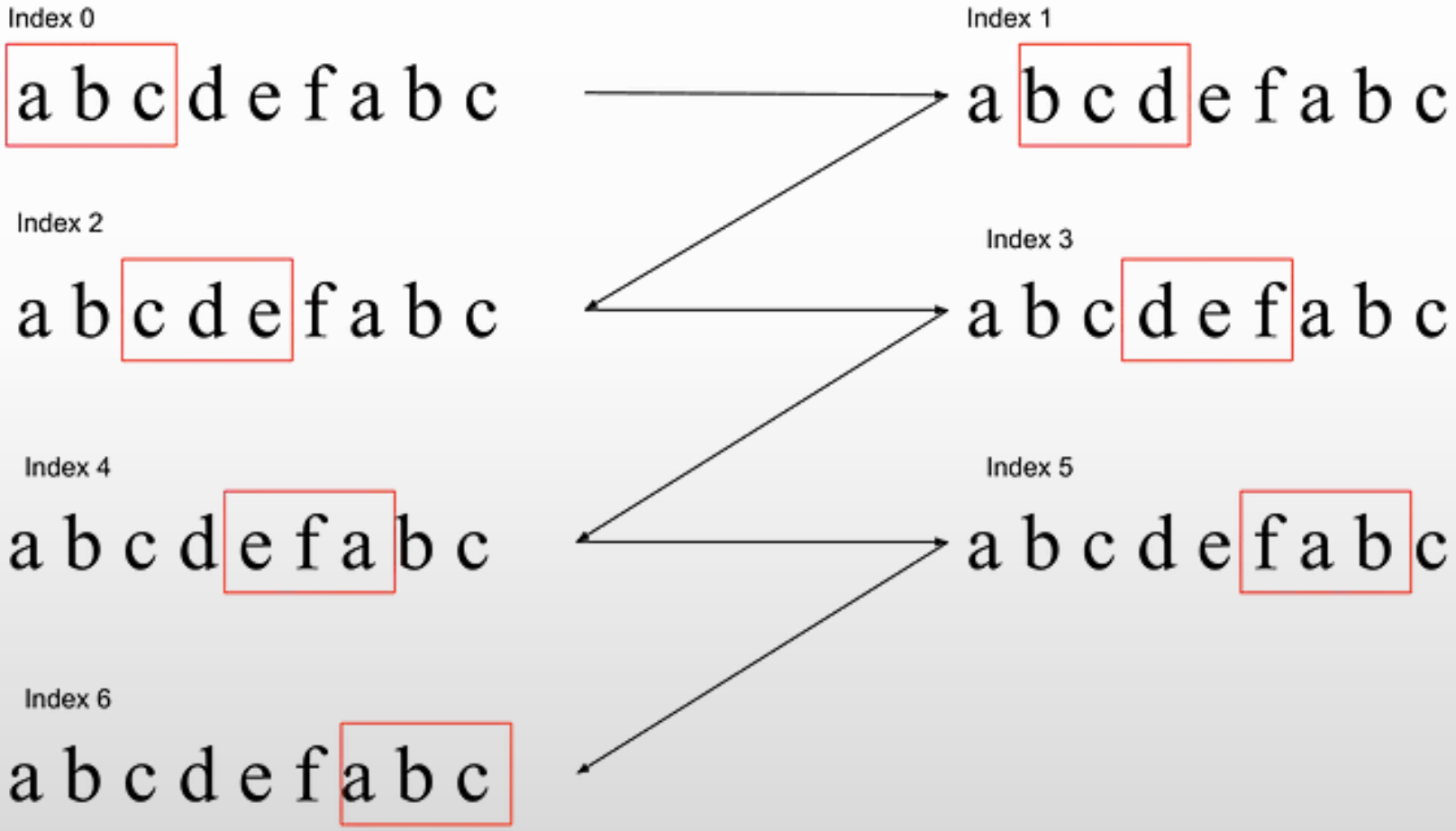
Rabin Karp



Rabin Karp



Rabin Karp



SON