Fundamentos de Algoritmos e Estrutura de Dados – Lecture 07 – Pattern Matching

Prof. André Gustavo Hochuli

gustavo.hochuli@pucpr.br
aghochuli@ppgia.pucpr.br

Plano de Aula

- **Lecture 06**
  - **Fluxo Máximo: Ford-Fulkerson**

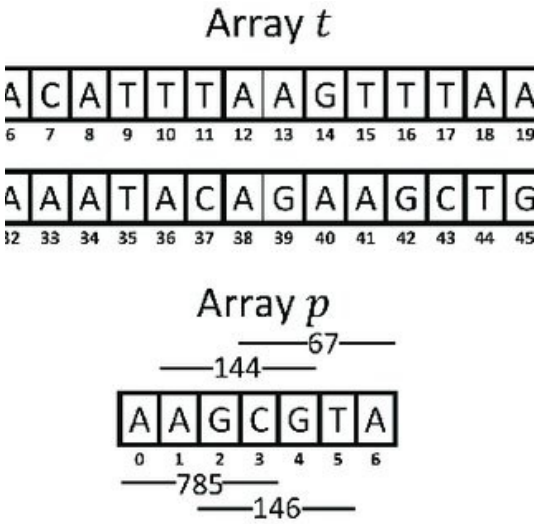
- **Lecture 06**
  - **Pattern Matching (correspondência de cadeias)**
    - **Exato (Força Bruta, KMP, Rabin-Karp)**

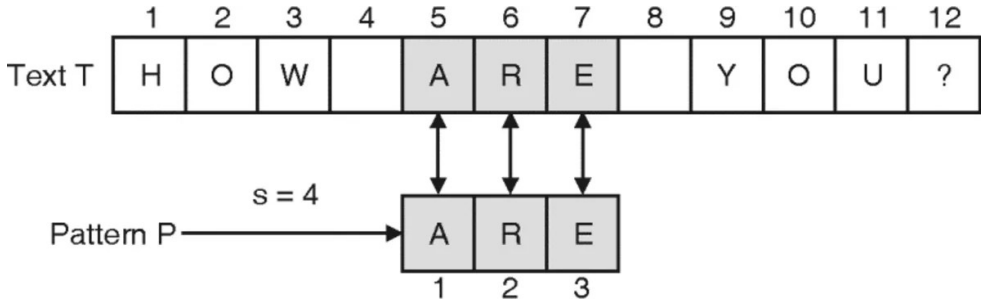
Pattern Matching
Correspondência de Cadeias

Correspondência de Cadeias

- **Problema:**
  - **Encontrar todas as ocorrências de um padrão dentro de uma sequência**

- **Aplicações**
  - **Busca de Texto**
  - **DNA, RNA, proteínas**
  - **Compiladores**
  - **Compressão de dados**
  - **CiberSegurança**



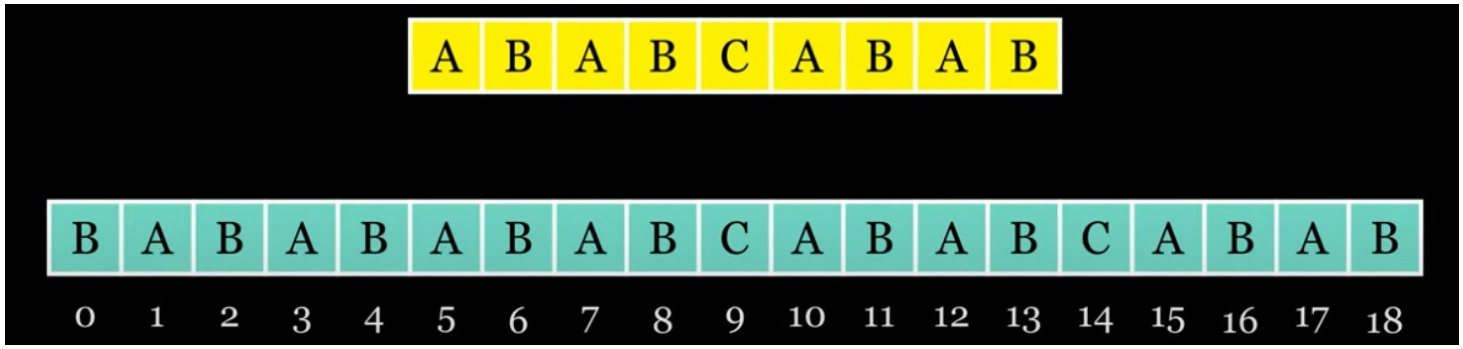


Correspondência de Cadeias

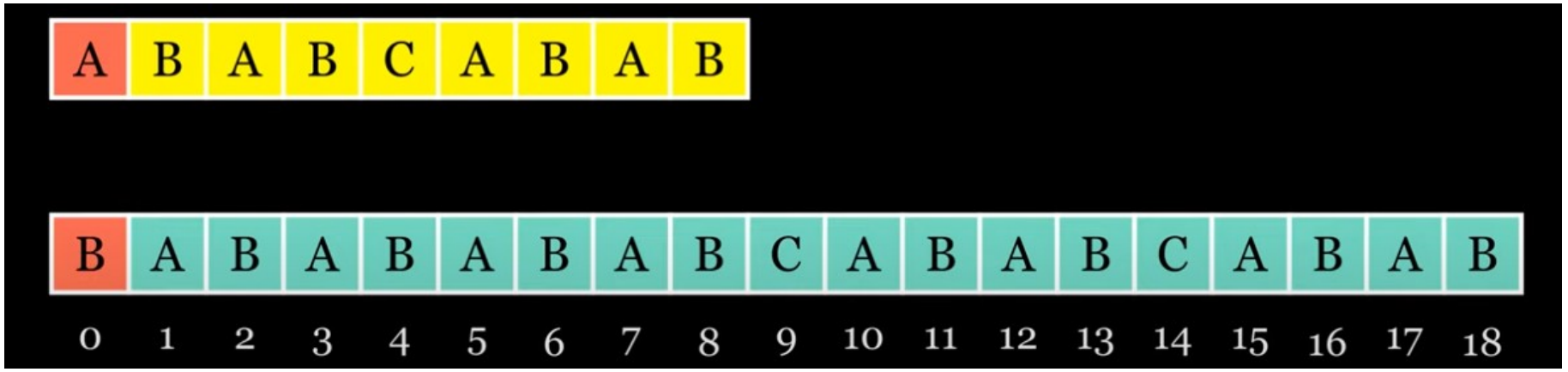
- **Algoritmos:**
  - **Correspondência Exata**
    - **Força – Bruta**
    - **KMP**
    - **Rabin-Karp**
  - **Aproximação**
    - **Edit Distance**

Força Bruta

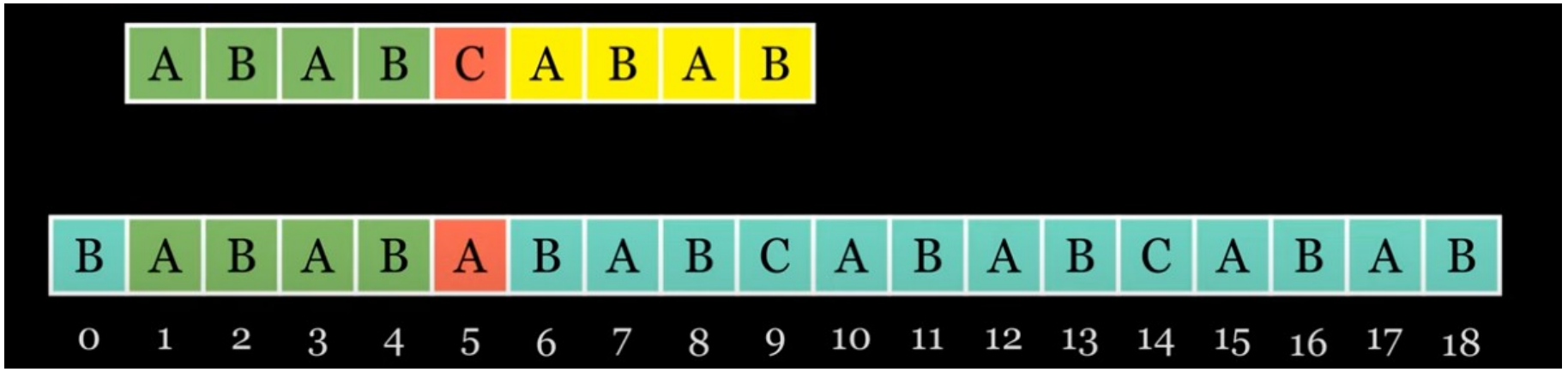
- **Compara o padrão P com todas as substrings possíveis do texto T de mesmo comprimento que P, posição por posição.**
  - **Percorrer o texto do índice 0 até n−m (onde n= tamanho do texto, m = tamanho do padrão).**
  - **Para cada posição i do texto:**
  - **Comparar T[i..i+m−1] com P[0..m−1], caractere por caractere.**
  - **Se todos os caracteres coincidem, registrar que o padrão foi encontrado na posição i**
  - **Repetir até o final da sequência**



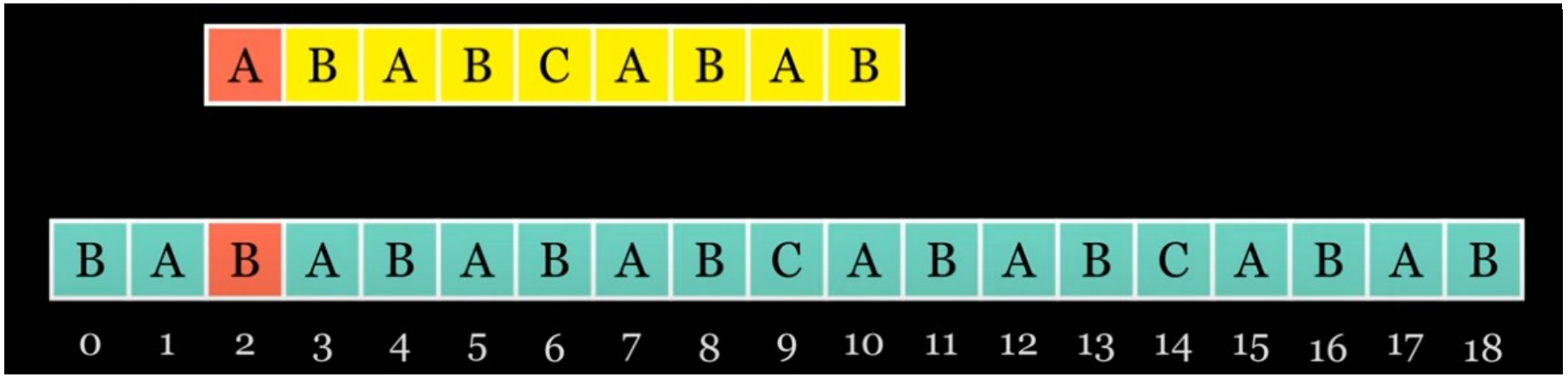
Força Bruta


A B A B C A B A B
B A B A B A B A B C A B A B C A B A B
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18

Força Bruta

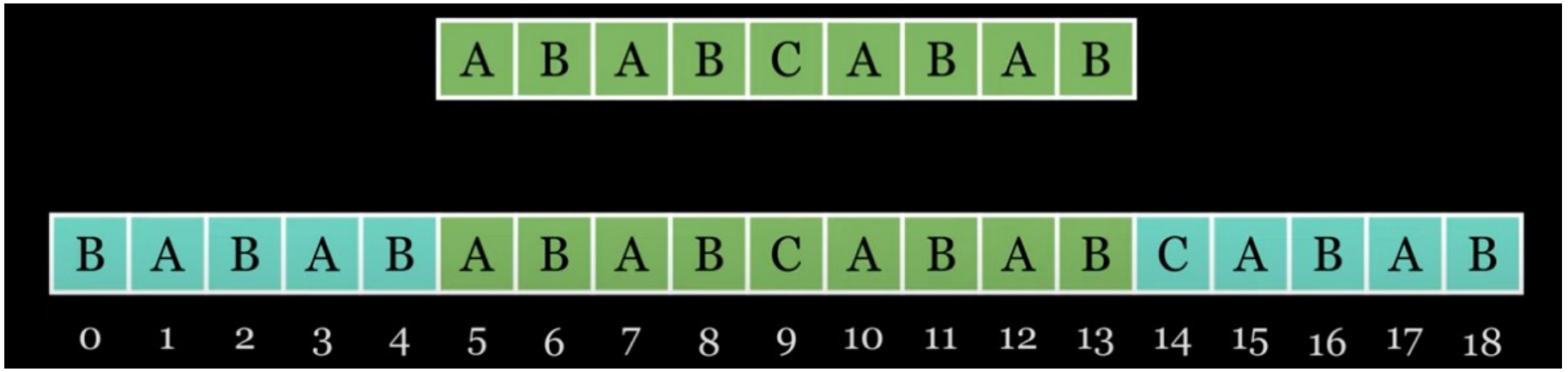

A B A B C A B A B
B A B A B A B A B C A B A B C A B A B
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18

Força Bruta


A B A B C A B A B
B A B A B A B A B C A B A B C A B A B
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18

Força Bruta

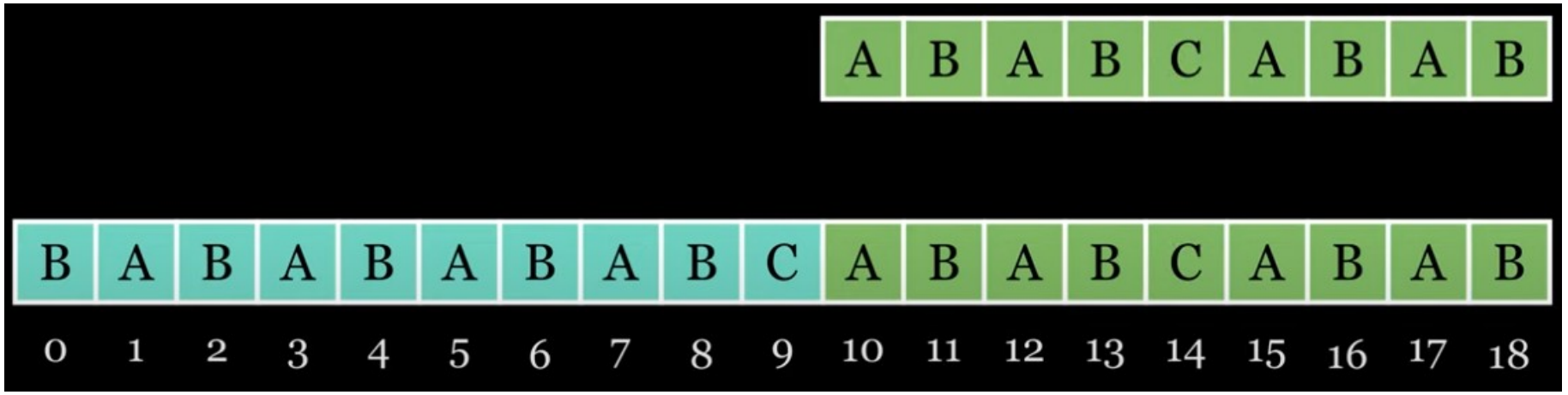
.....



Found at: 5

Força Bruta

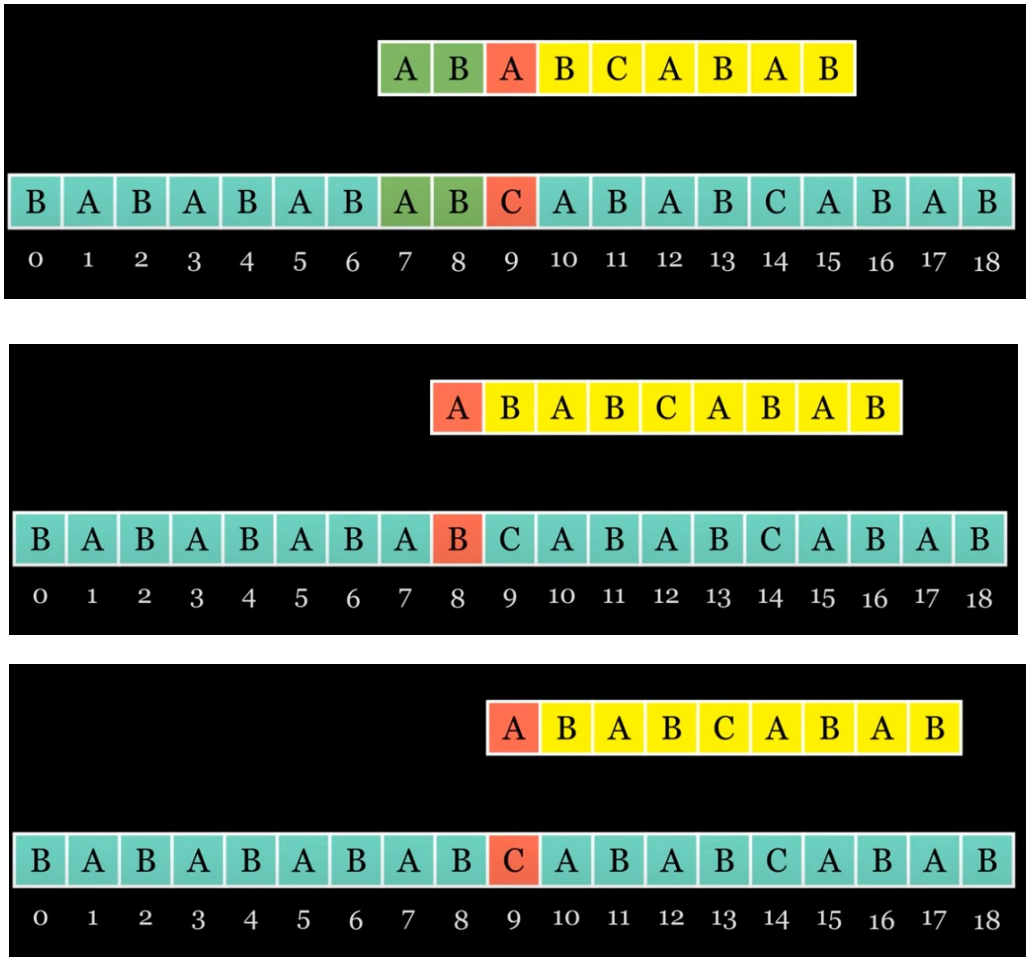
.....



Found at: 5 10

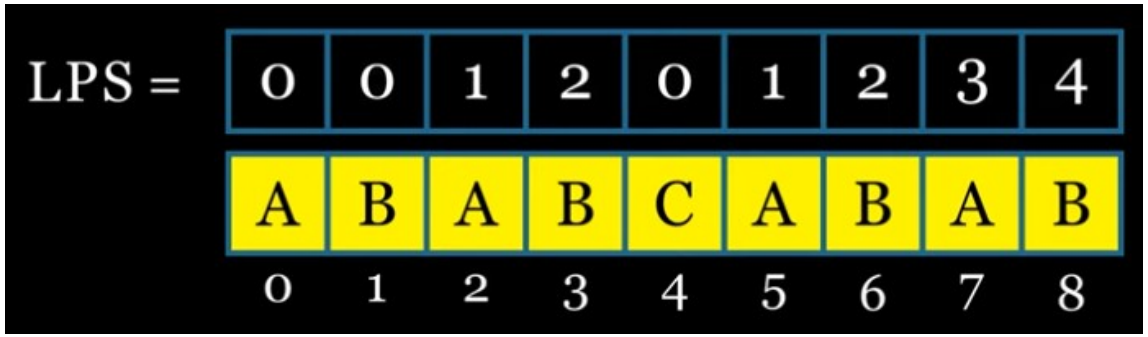
Força Bruta

- **Simples e Sem Pré-Processamento**
- **Extremamente Custoso**
- **Comparações Desnecessárias**



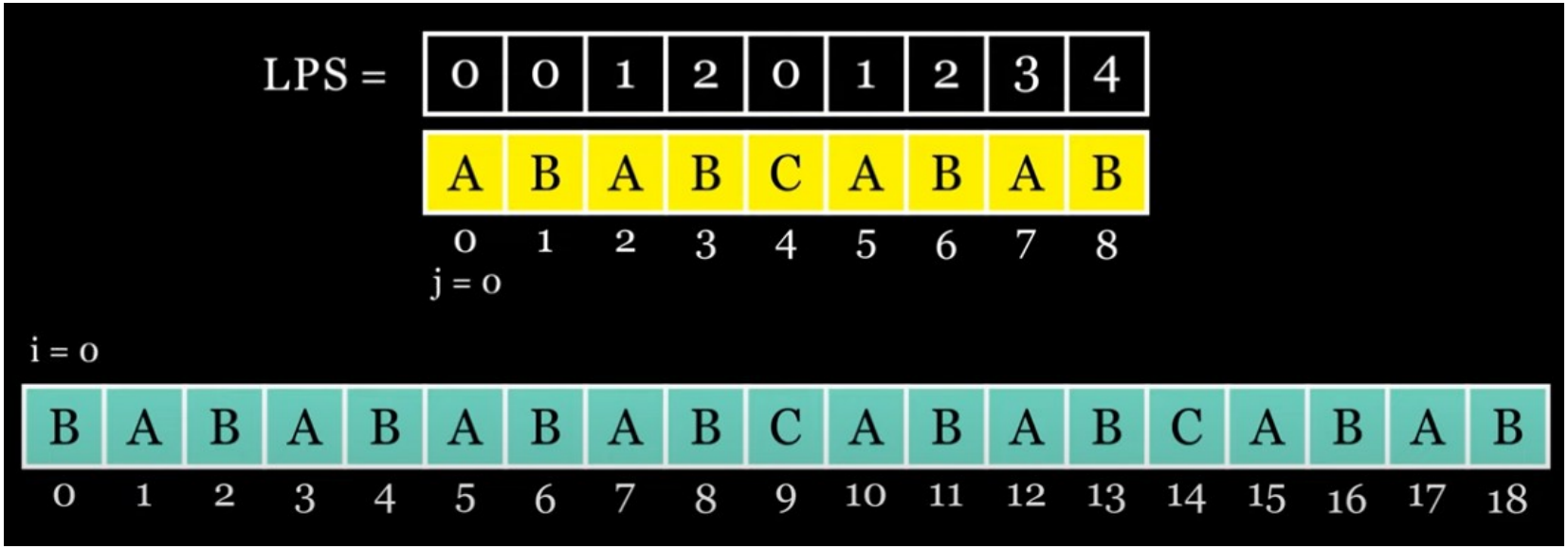
KMP

- **Donald Knut, James Morris e Vaughan Pratt**
- **Passo 1: vetor de prefixo (Longest Proper Prefix)**

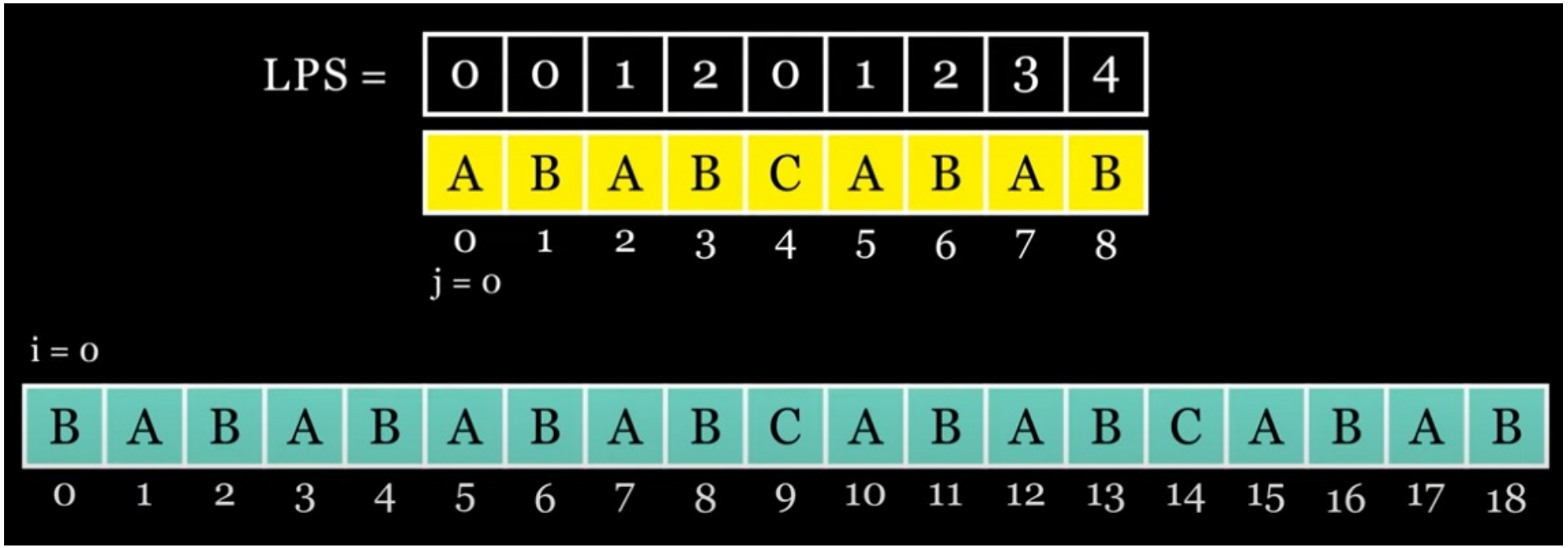


KMP

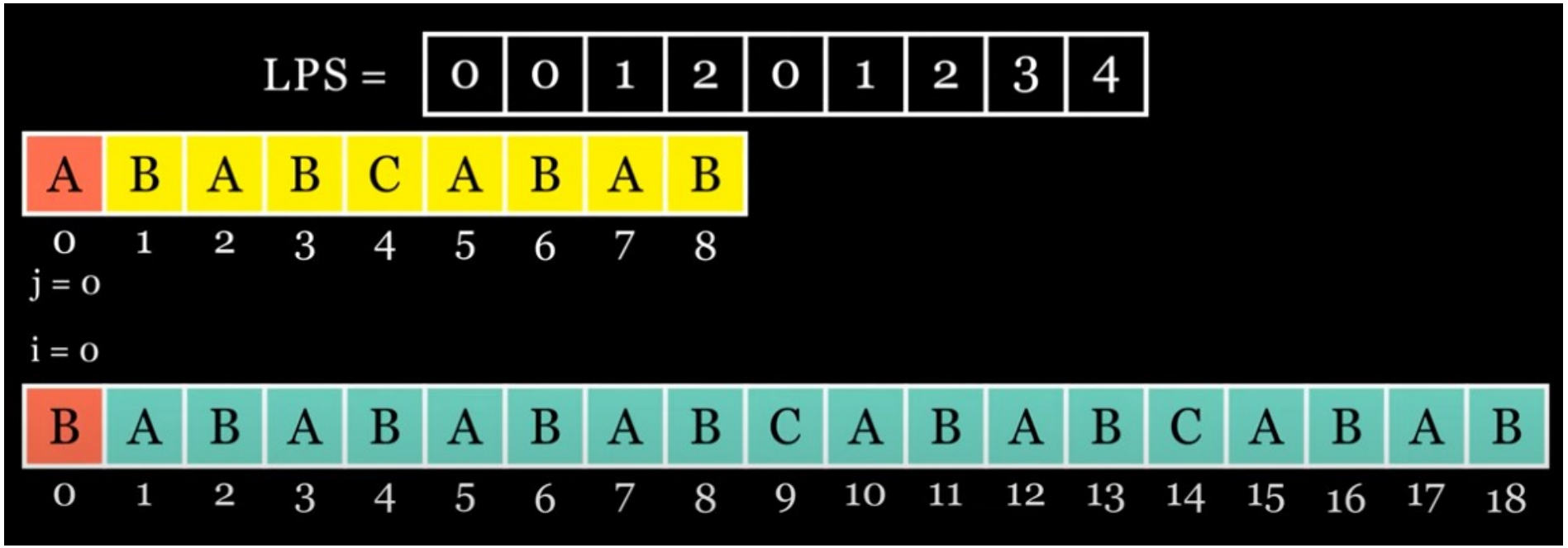
- **Donald Knut, James Morris e Vaughan Pratt**
  - **Passo 2 – Buscar o padrão no texto usando a tabela LPS**
  - **Se houver match, avançar nos índices do texto e do padrão.**
  - **Se houver mismatch, usar a tabela LPS para ajustar o índice do padrão, sem retroceder no texto.**



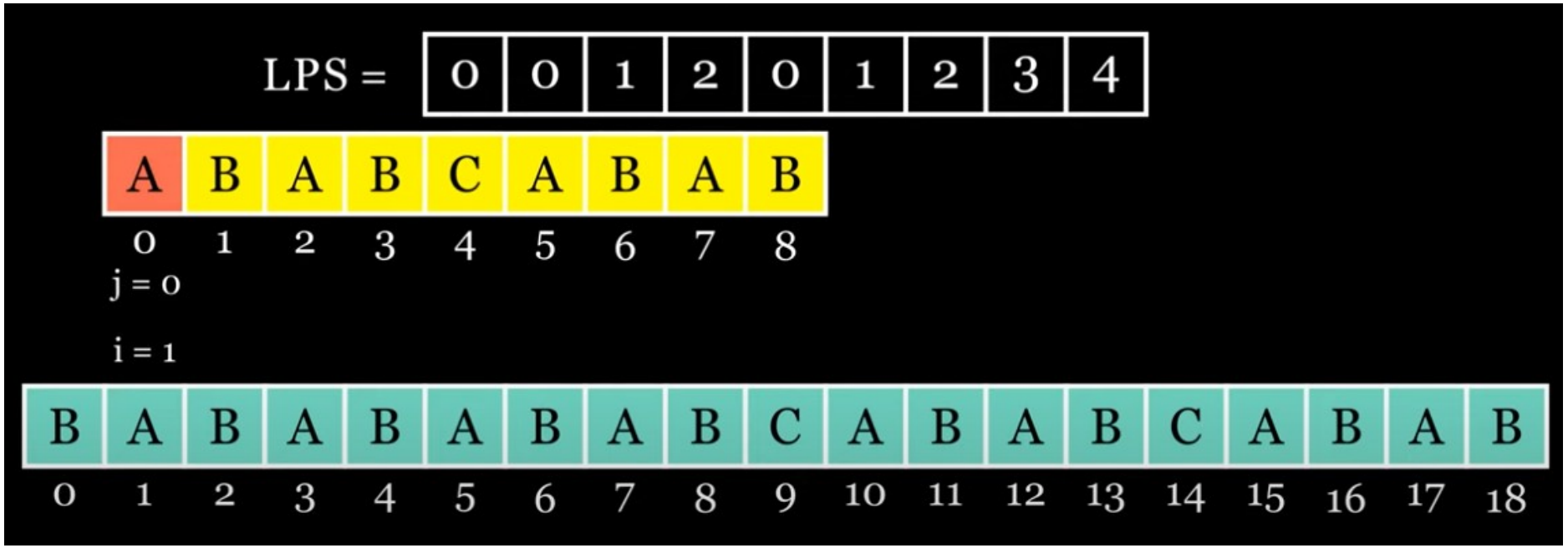
KMP


LPS =
0 0 1 2 0 1 2 3 4
A B A B C A B A B
0 1 2 3 4 5 6 7 8
j = 0
i = 0
B A B A B A B A B C A B A B C A B A B
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18

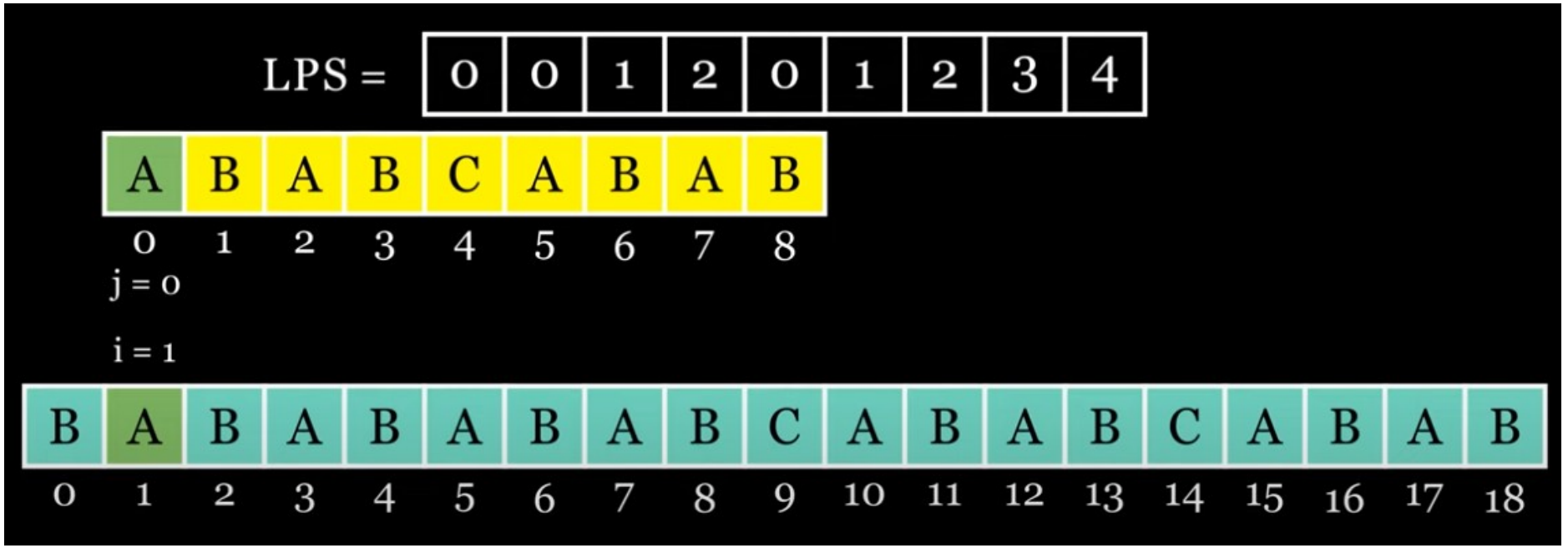
KMP


LPS = 0 0 1 2 0 1 2 3 4
A B A B C A B A B
0 1 2 3 4 5 6 7 8
j = 0
i = 0
B A B A B A B A B C A B A B C A B A B
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18

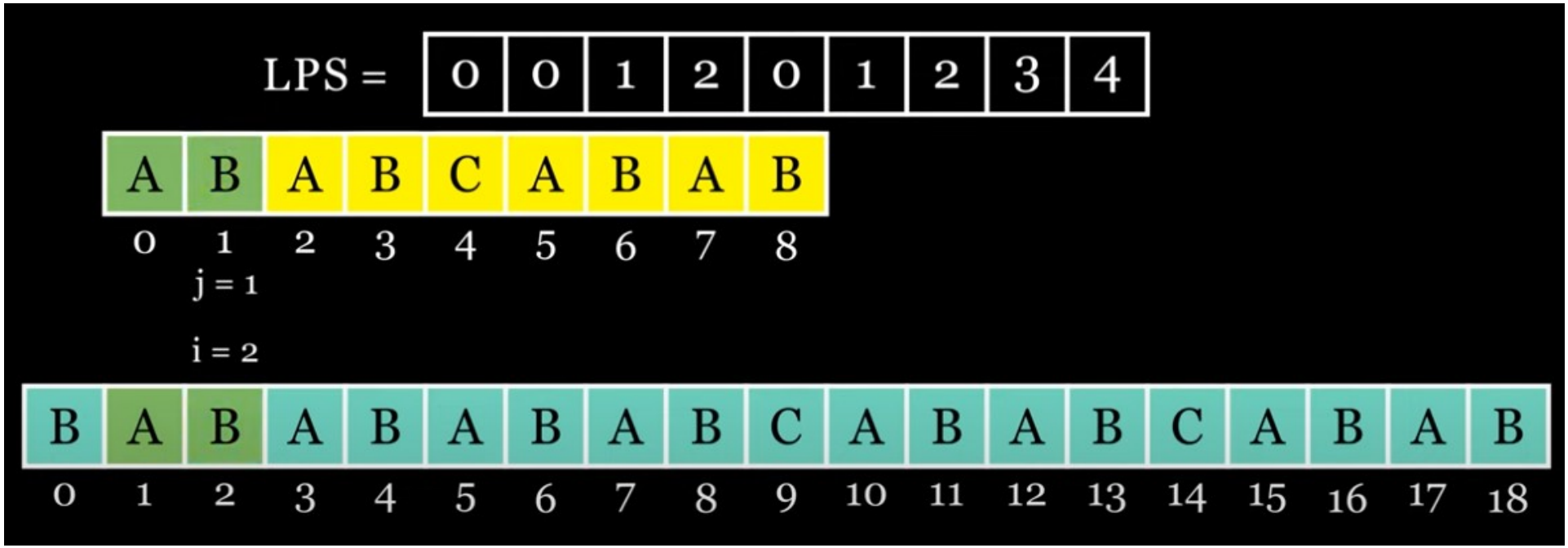
KMP


LPS = 0 0 1 2 0 1 2 3 4
A B A B C A B A B
0 1 2 3 4 5 6 7 8
j = 0
i = 1
B A B A B A B A B C A B A B C A B A B
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18

KMP

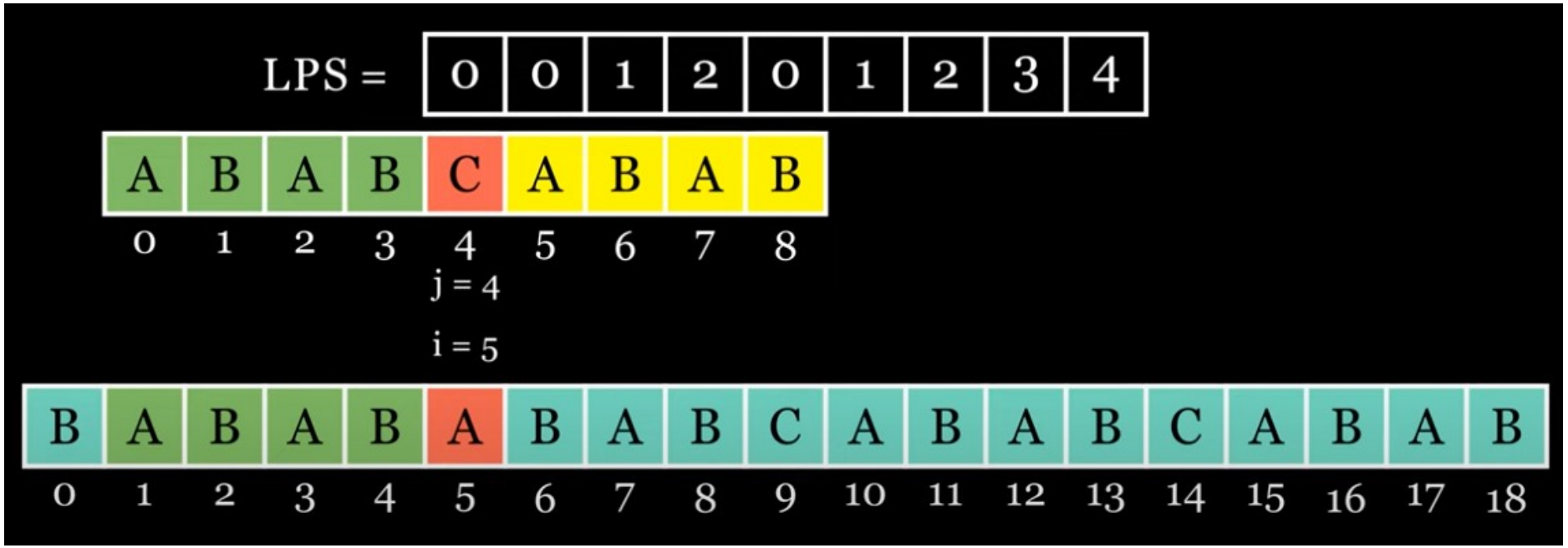

LPS =
0 0 1 2 0 1 2 3 4
A B A B C A B A B
0 1 2 3 4 5 6 7 8
j = 0
i = 1
B A B A B A B A B C A B A B C A B A B
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18

KMP

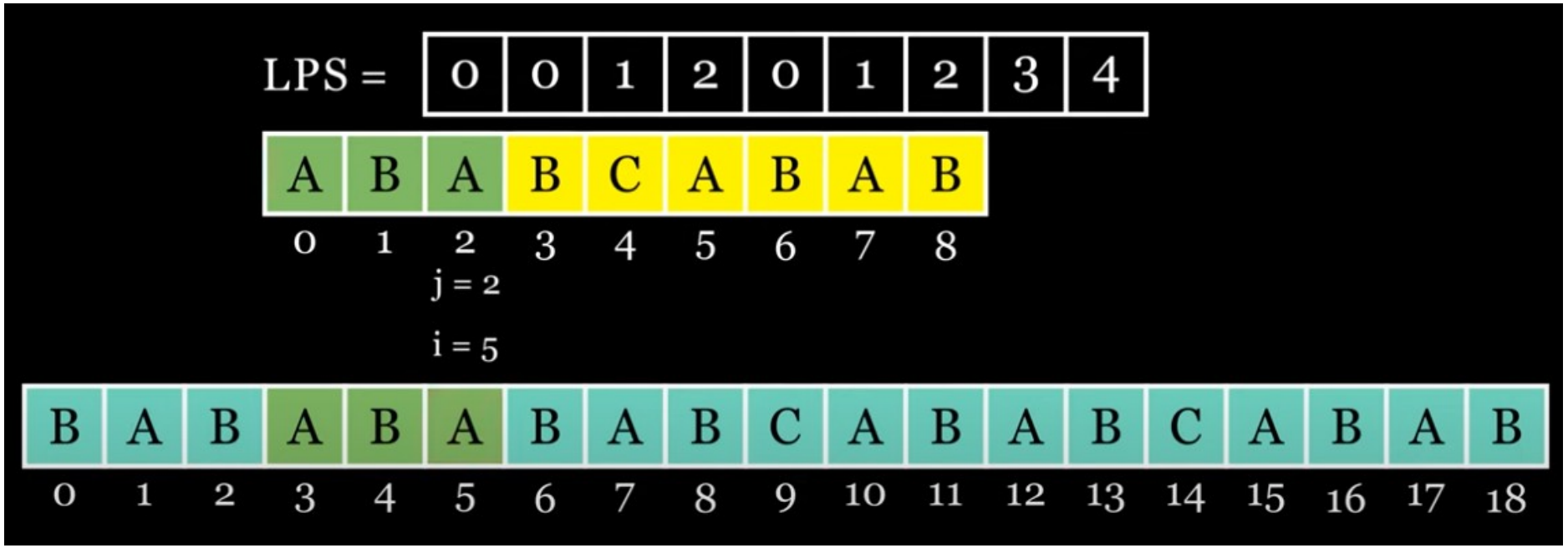

LPS =
0 0 1 2 0 1 2 3 4
A B A B C A B A B
0 1 2 3 4 5 6 7 8
j = 1
i = 2
B A B A B A B A B C A B A B C A B A B
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18

KMP

.....

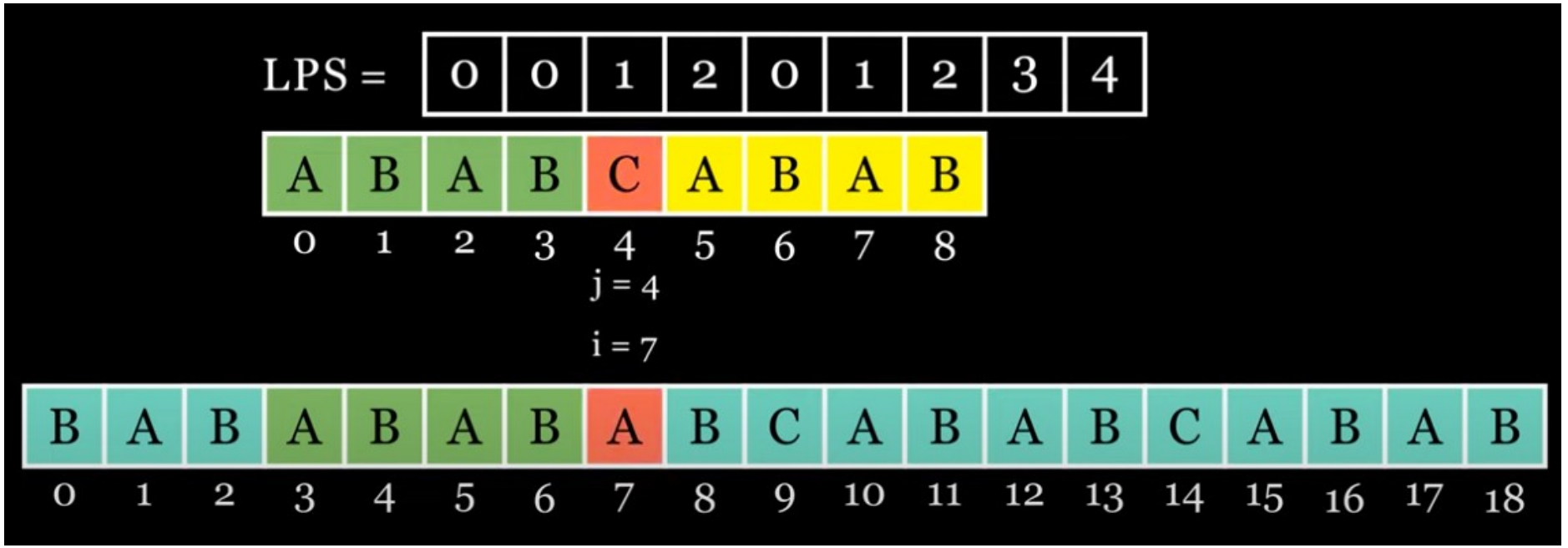

LPS =
0 0 1 2 0 1 2 3 4
A B A B C A B A B
0 1 2 3 4 5 6 7 8
j = 4
i = 5
B A B A B A B A B C A B A B C A B A B
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18

KMP

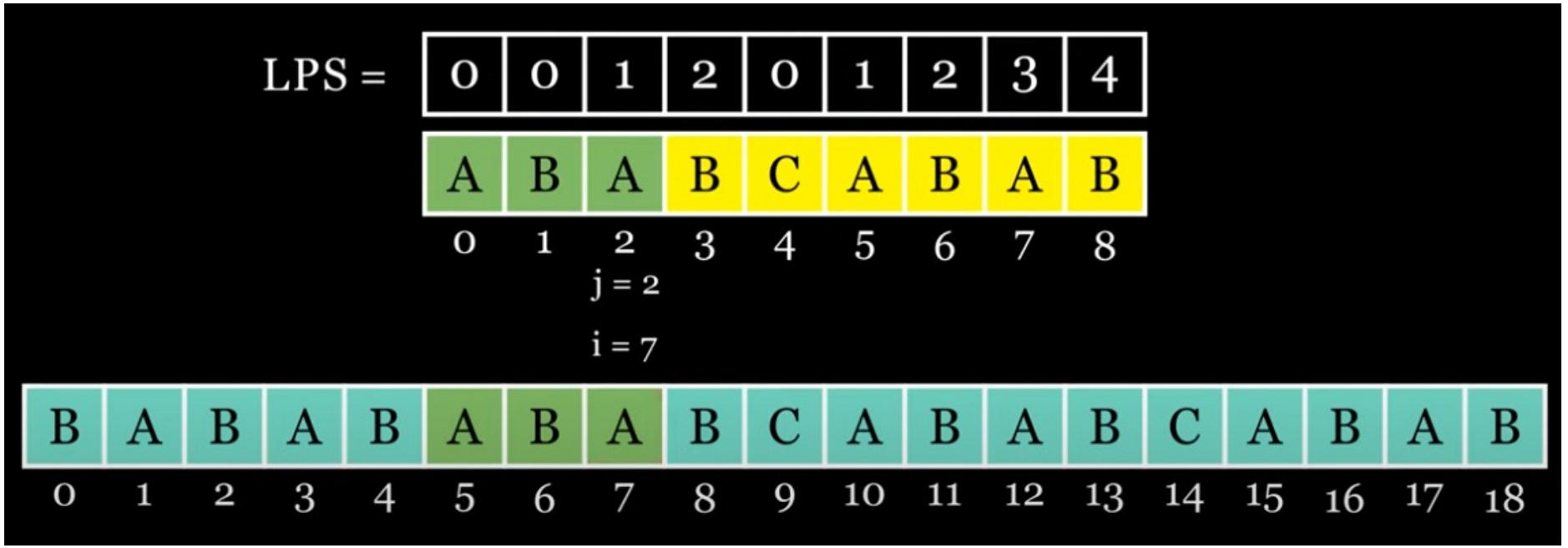

LPS =
0 0 1 2 0 1 2 3 4
A B A B C A B A B
0 1 2 3 4 5 6 7 8
j = 2
i = 5
B A B A B A B A B C A B A B C A B A B
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18

KMP

.....

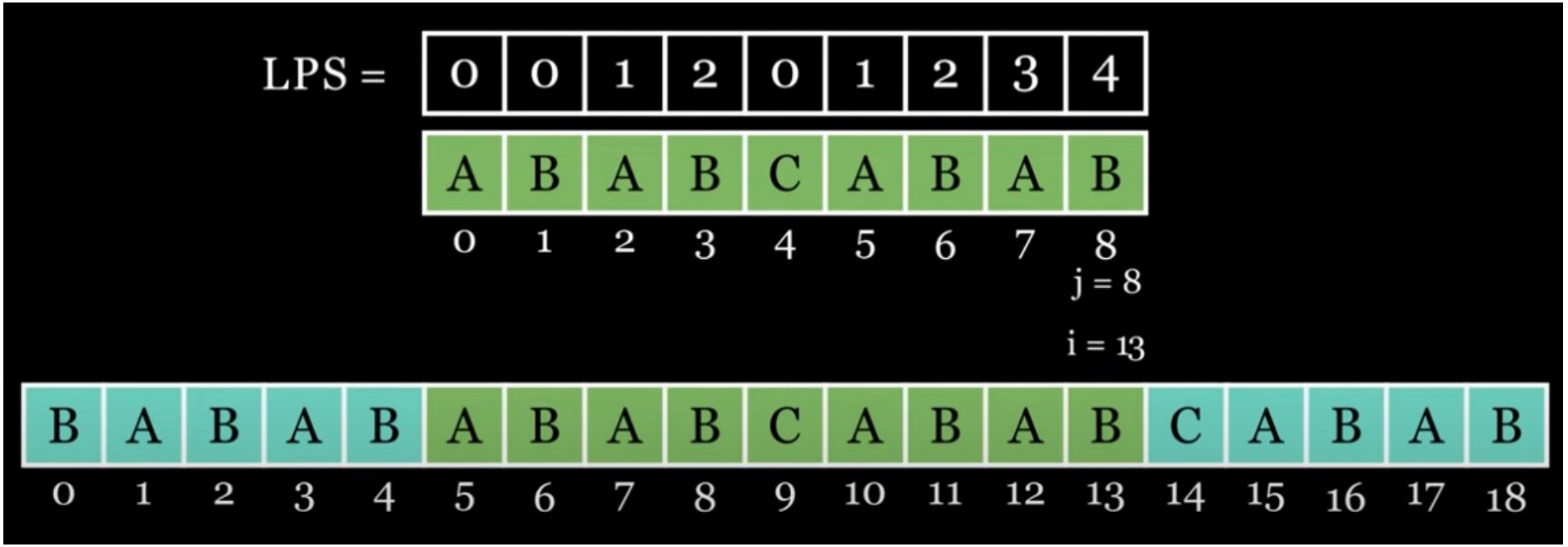

LPS = 0 0 1 2 0 1 2 3 4
A B A B C A B A B
0 1 2 3 4 5 6 7 8
j = 4
i = 7
B A B A B A B A B C A B A B C A B A B
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18

KMP


LPS =
0 0 1 2 0 1 2 3 4
A B A B C A B A B
0 1 2 3 4 5 6 7 8
j = 2
i = 7
B A B A B A B A B C A B A B C A B A B
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18

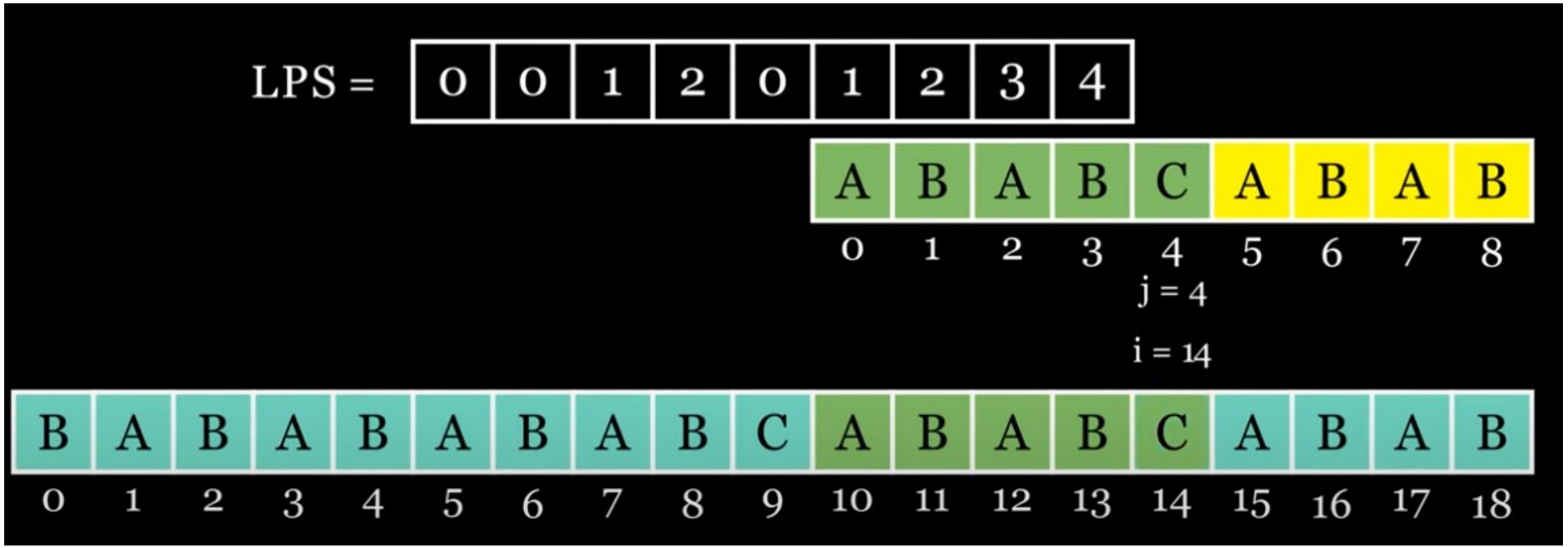
KMP

.....

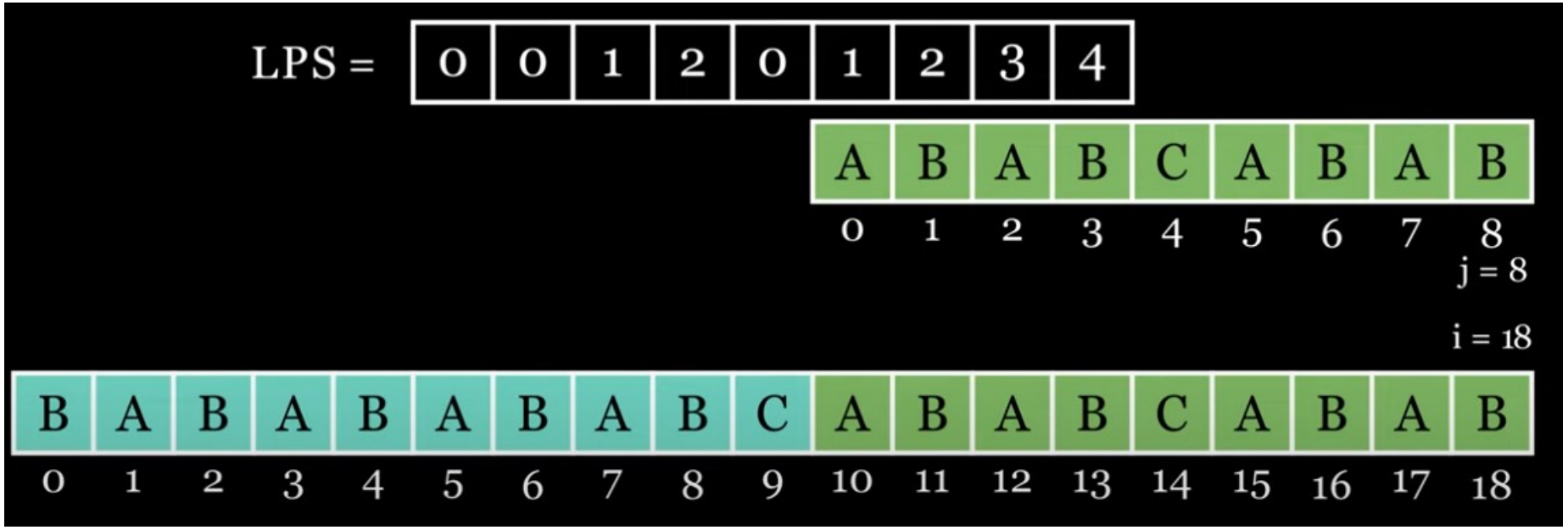


Found at: 5

KMP


LPS =
0 0 1 2 0 1 2 3 4
A B A B C A B A B
0 1 2 3 4 5 6 7 8
j = 4
i = 14
B A B A B A B A B C A B A B C A B A B
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18

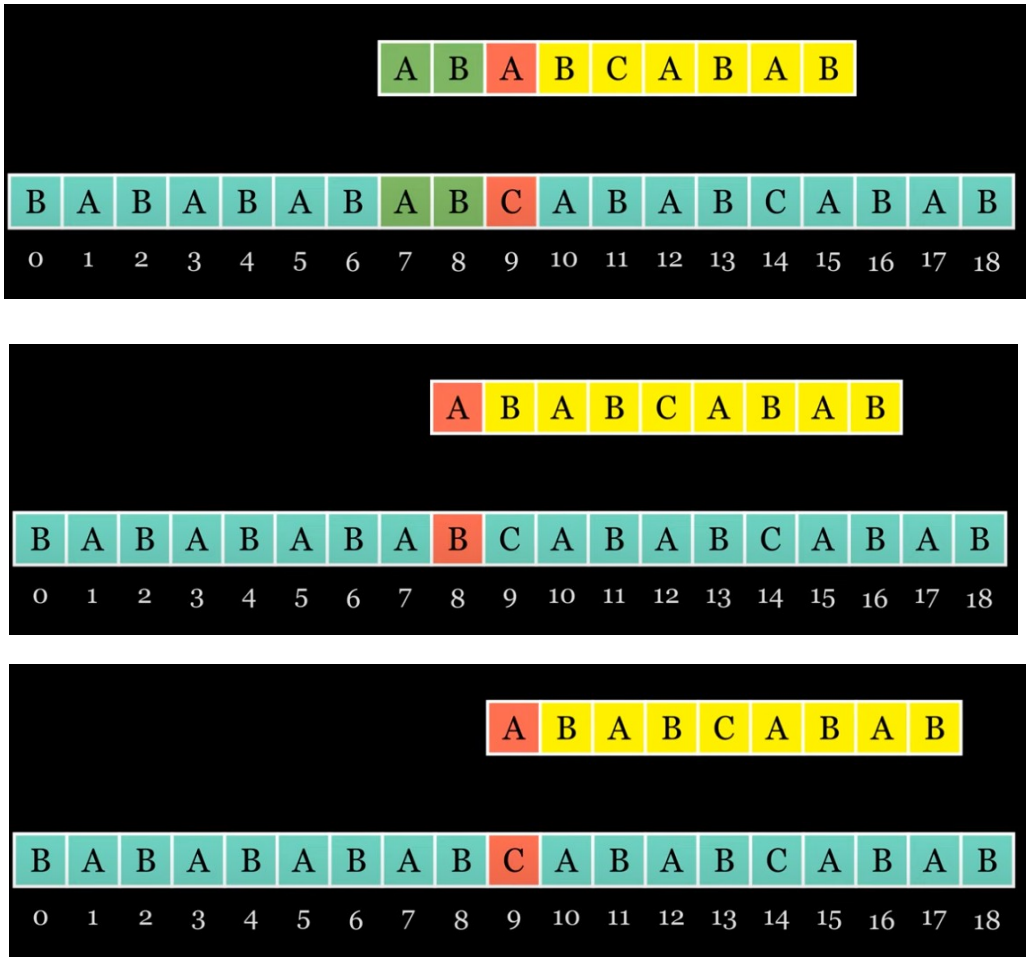
KMP



Found at: 5 10

KMP

- **Evita comparações repetidas**
- **Necessita pré-processamento**
  - **LPS (Vetor)**



Rabin-Karp

- **Eficiente**
- **Hash-Based**
- **Rolling Hash**
  - **Comprimento do padrão (janela)**
  - **Base**
  - **Primo (hash)**

Rolling Hash Function

$$H = c_1 \times b^{n-1} + c_2 \times b^{n-2} + \cdots + c_n \times b^0 \mod p$$

Hash Update Formula

$$H_{\text{new}} = (b \times (H_{\text{old}} - c_{\text{old}} \times b^{n-1}) + c_{\text{new}}) \mod p$$

Rabin-Karp

- **Deslizar a janela sobre o texto**
- **Para cada posição da janela:**
- **Aplicar a fórmula de rolling hash para remover o caractere mais à esquerda e adicionar o novo caractere à direita**
- **Comparar o hash da janela com o hash do padrão**
- **Se houver igualdade, comparar caractere a caractere para confirmar correspondência**

Rolling Hash Function

$$H = c_1 \times b^{n-1} + c_2 \times b^{n-2} + \cdots + c_n \times b^0 \mod p$$

Hash Update Formula

$$H_{\text{new}} = (b \times (H_{\text{old}} - c_{\text{old}} \times b^{n-1}) + c_{\text{new}}) \mod p$$

Rabin-Karp

Rolling Hash Function

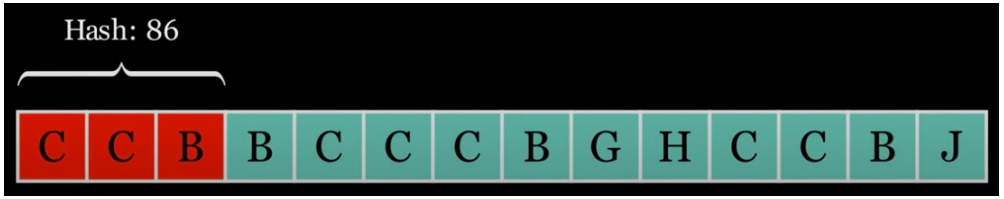
$$H = c_1 \times b^{n-1} + c_2 \times b^{n-2} + \cdots + c_n \times b^0 \mod p$$

Hash Update Formula

$$H_{\text{new}} = (b \times (H_{\text{old}} - c_{\text{old}} \times b^{n-1}) + c_{\text{new}}) \mod p$$

- Comprimento da janela $m = 3$
- Base $b = 256$ (ASCII)
- Módulo primo $p = 101$

$$H(\text{"CCB"}) = (67 \cdot 256^2 + 67 \cdot 256 + 66) \bmod 101$$



Rabin-Karp

Rolling Hash Function

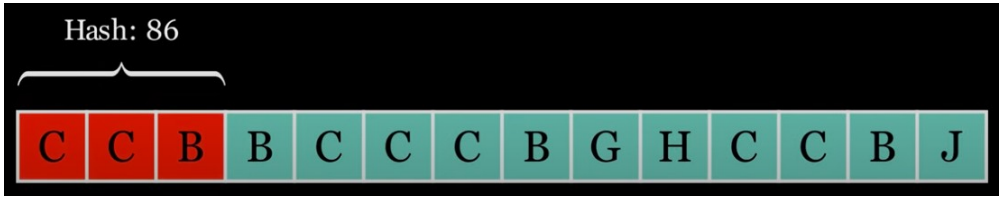
$$H = c_1 \times b^{n-1} + c_2 \times b^{n-2} + \cdots + c_n \times b^0 \mod p$$

Hash Update Formula

$$H_{\text{new}} = (b \times (H_{\text{old}} - c_{\text{old}} \times b^{n-1}) + c_{\text{new}}) \mod p$$

- Comprimento da janela $m = 3$
- Base $b = 256$ (ASCII)
- Módulo primo $p = 101$

$$H(\text{"CCB"}) = (67 \cdot 256^2 + 67 \cdot 256 + 66) \bmod 101$$



Rabin-Karp

Rolling Hash Function

$$H = c_1 \times b^{n-1} + c_2 \times b^{n-2} + \cdots + c_n \times b^0 \mod p$$

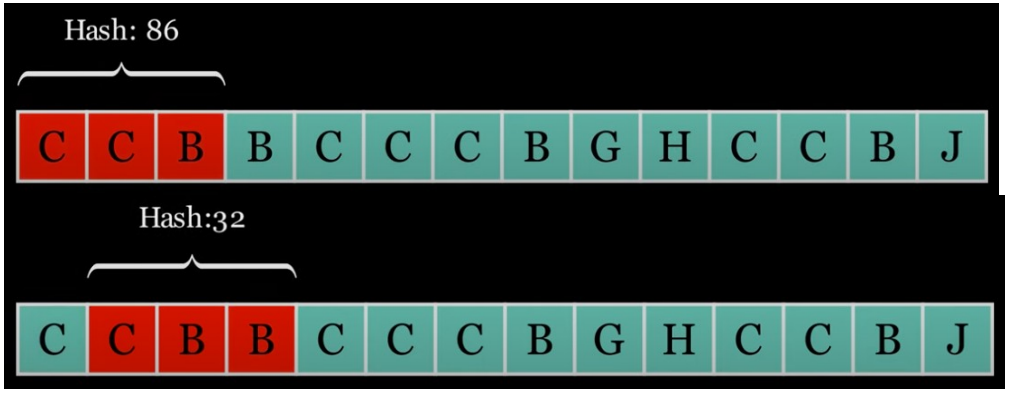
Hash Update Formula

$$H_{\text{new}} = (b \times (H_{\text{old}} - c_{\text{old}} \times b^{n-1}) + c_{\text{new}}) \mod p$$

- Comprimento da janela $m = 3$
- Base $b = 256$ (ASCII)
- Módulo primo $p = 101$

$$H(\text{"}CCB\text{"}) = (67 \cdot 256^2 + 67 \cdot 256 + 66) \bmod 101$$

$$H(\text{CBB}) = (256 \cdot (86 - 67 \cdot 256^2) + 66) \mod 101$$



Rabin-Karp

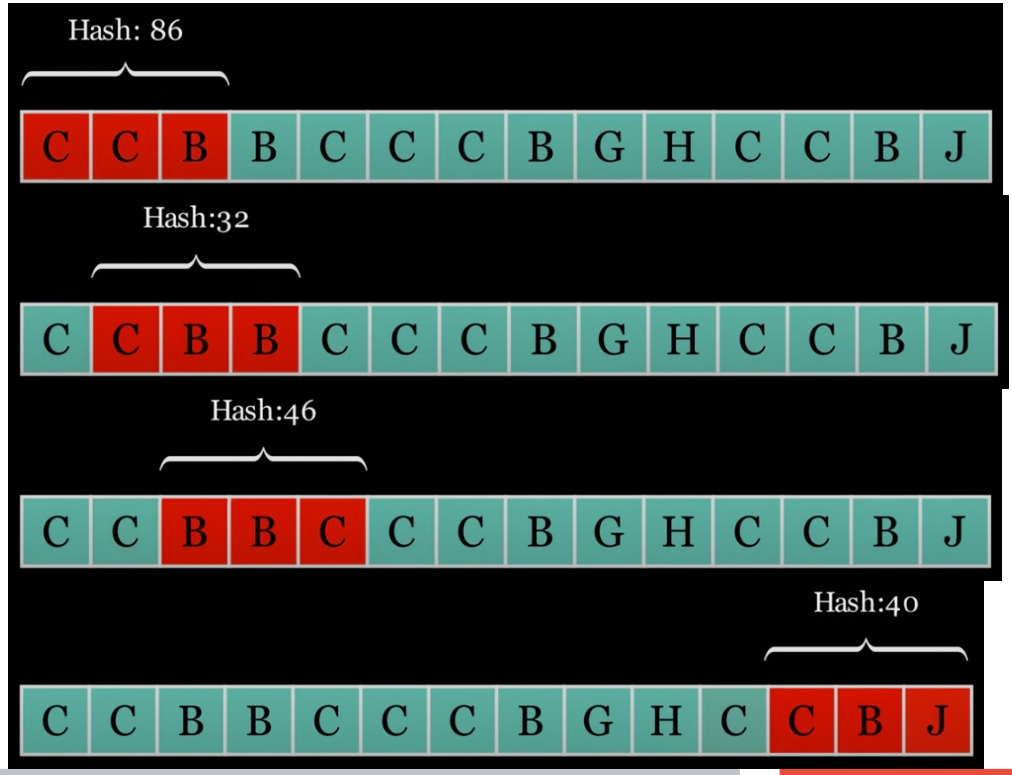
Rolling Hash Function

$$H = c_1 \times b^{n-1} + c_2 \times b^{n-2} + \cdots + c_n \times b^0 \mod p$$

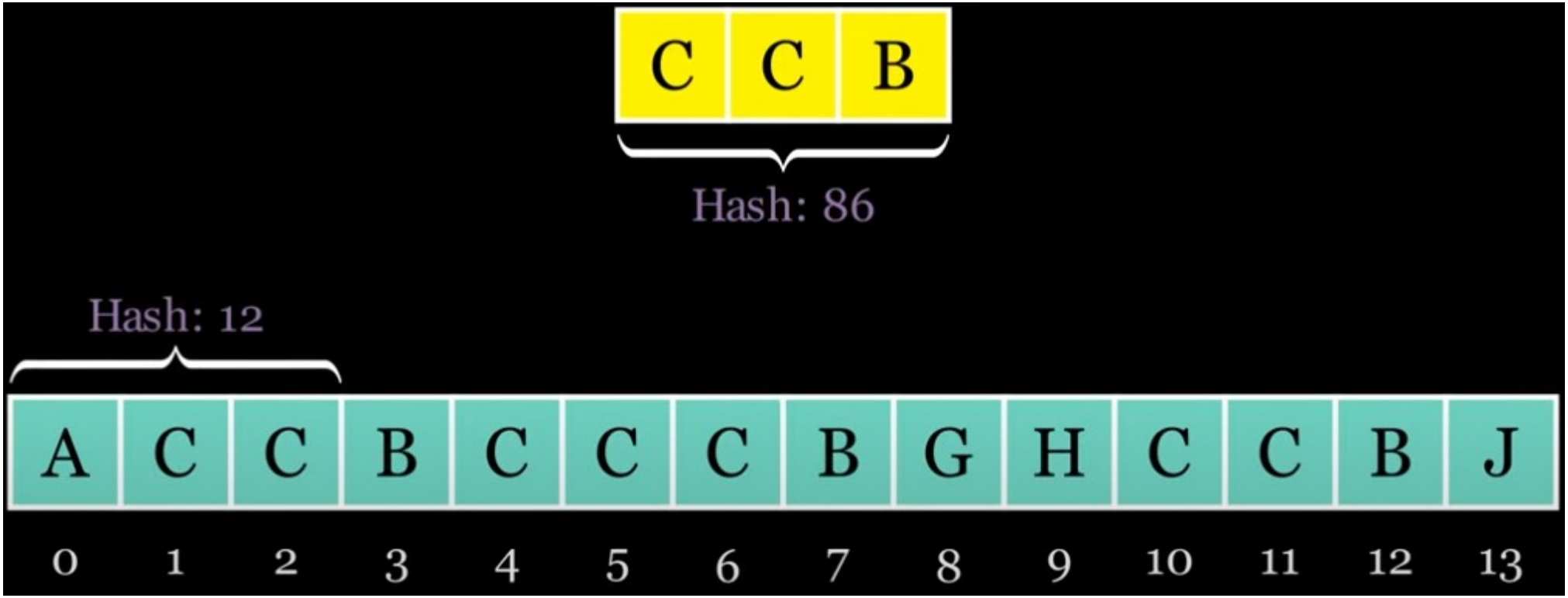
Hash Update Formula

$$H_{\text{new}} = (b \times (H_{\text{old}} - c_{\text{old}} \times b^{n-1}) + c_{\text{new}}) \mod p$$

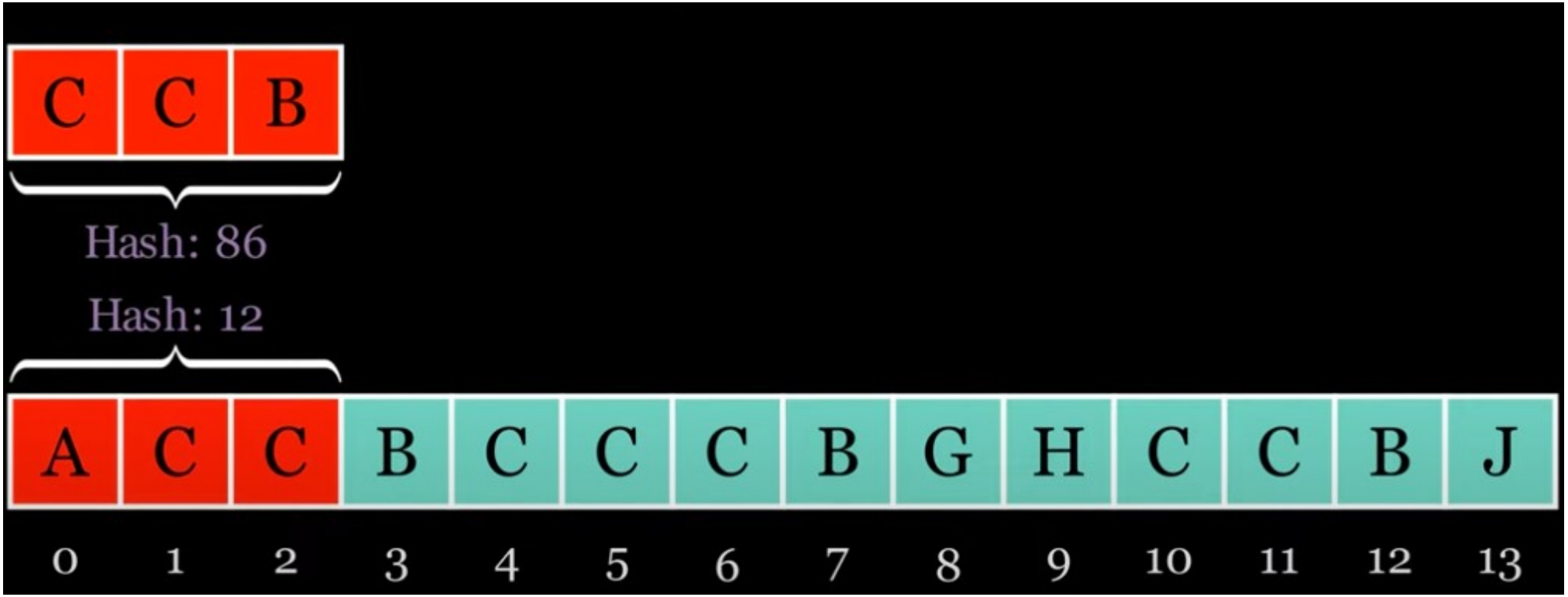
- Comprimento da janela $m = 3$
- Base $b = 256$ (ASCII)
- Módulo primo $p = 101$



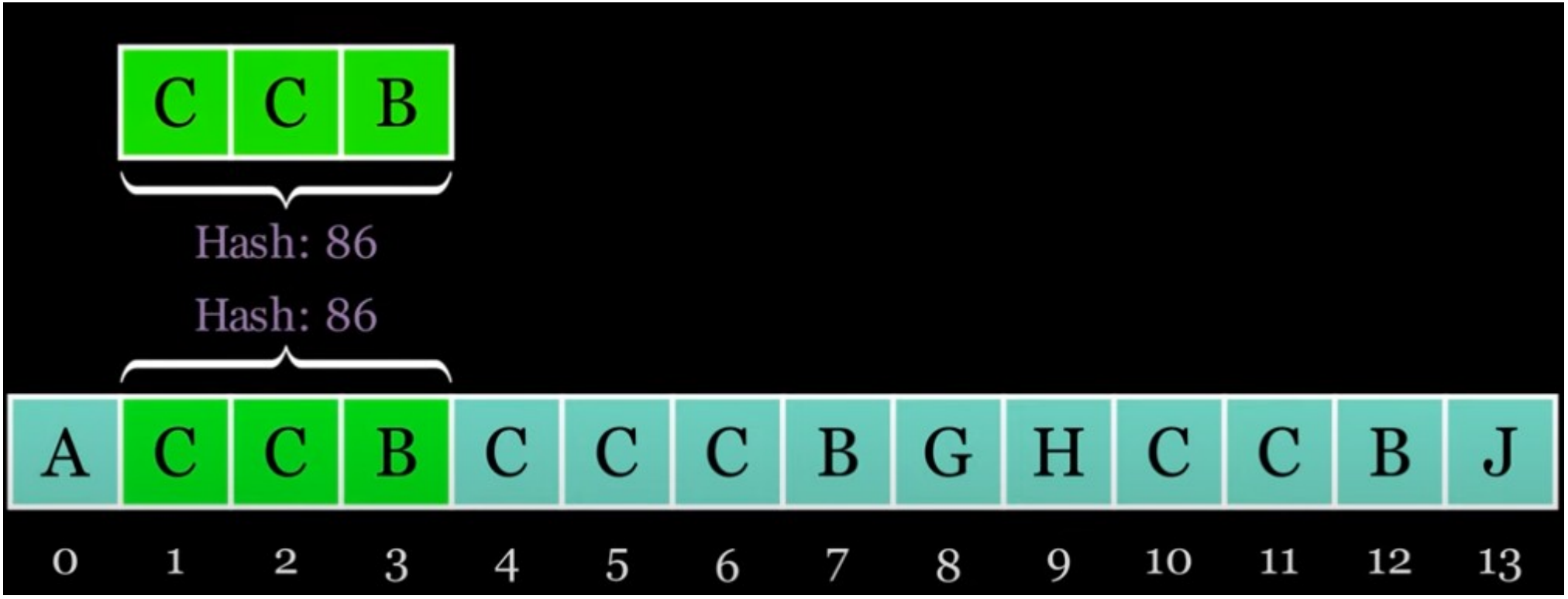
Rabin-Karp



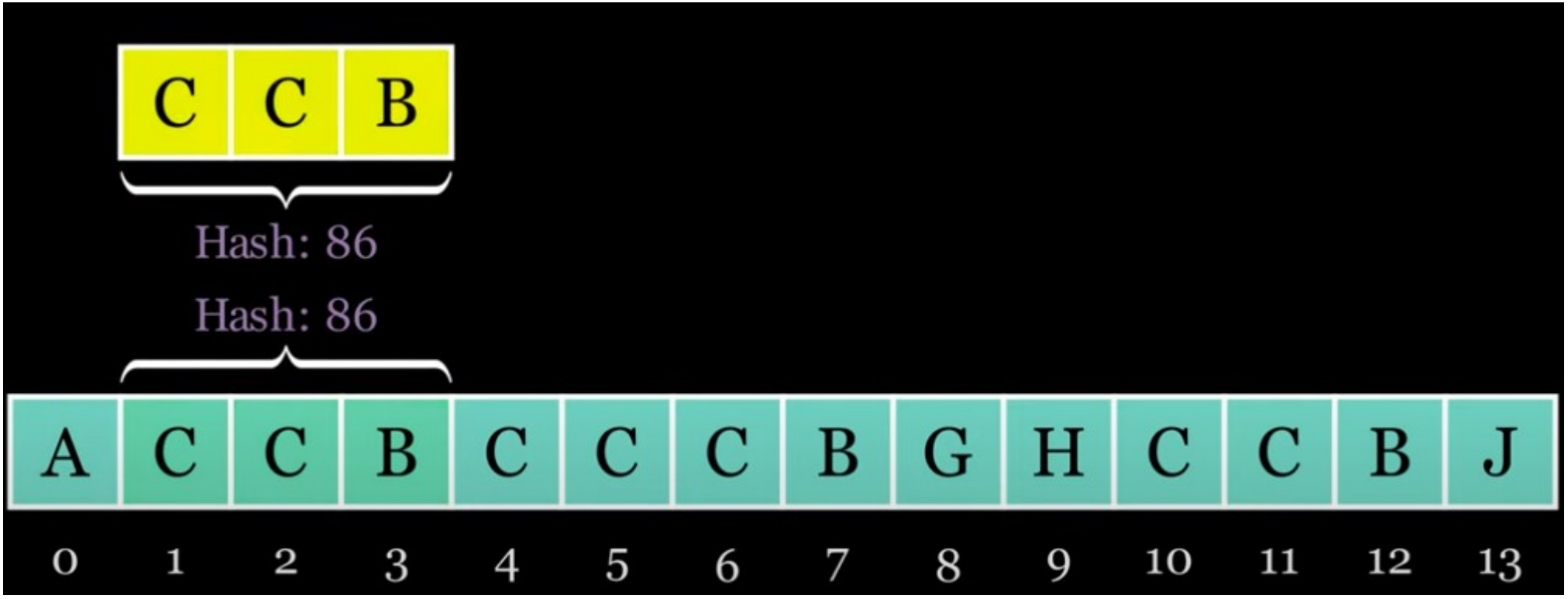
Rabin-Karp



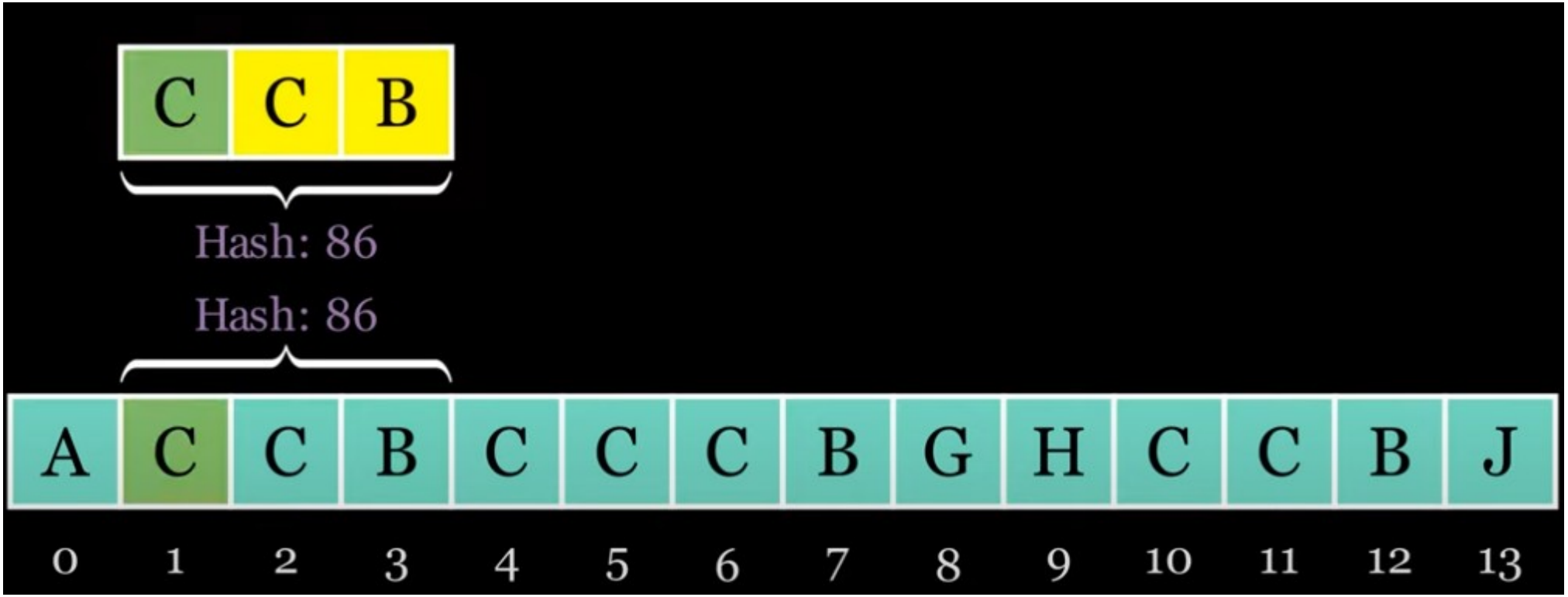
Rabin-Karp



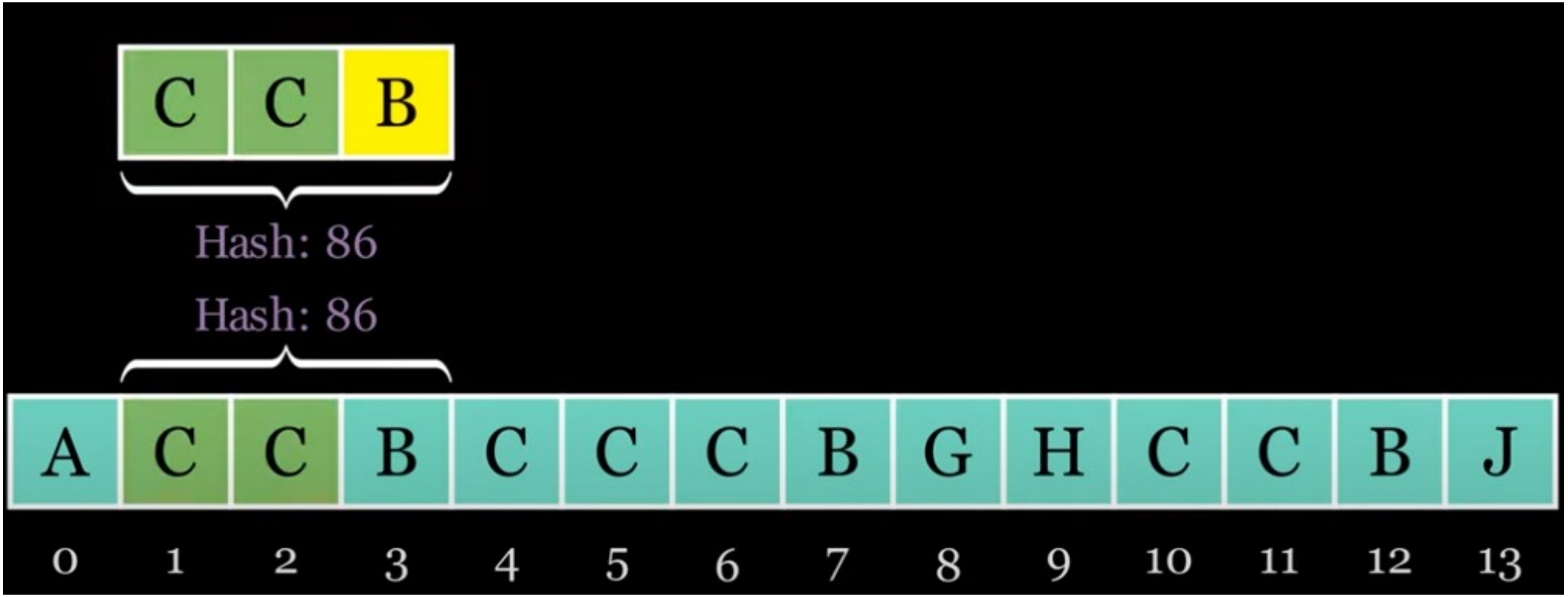
Rabin-Karp



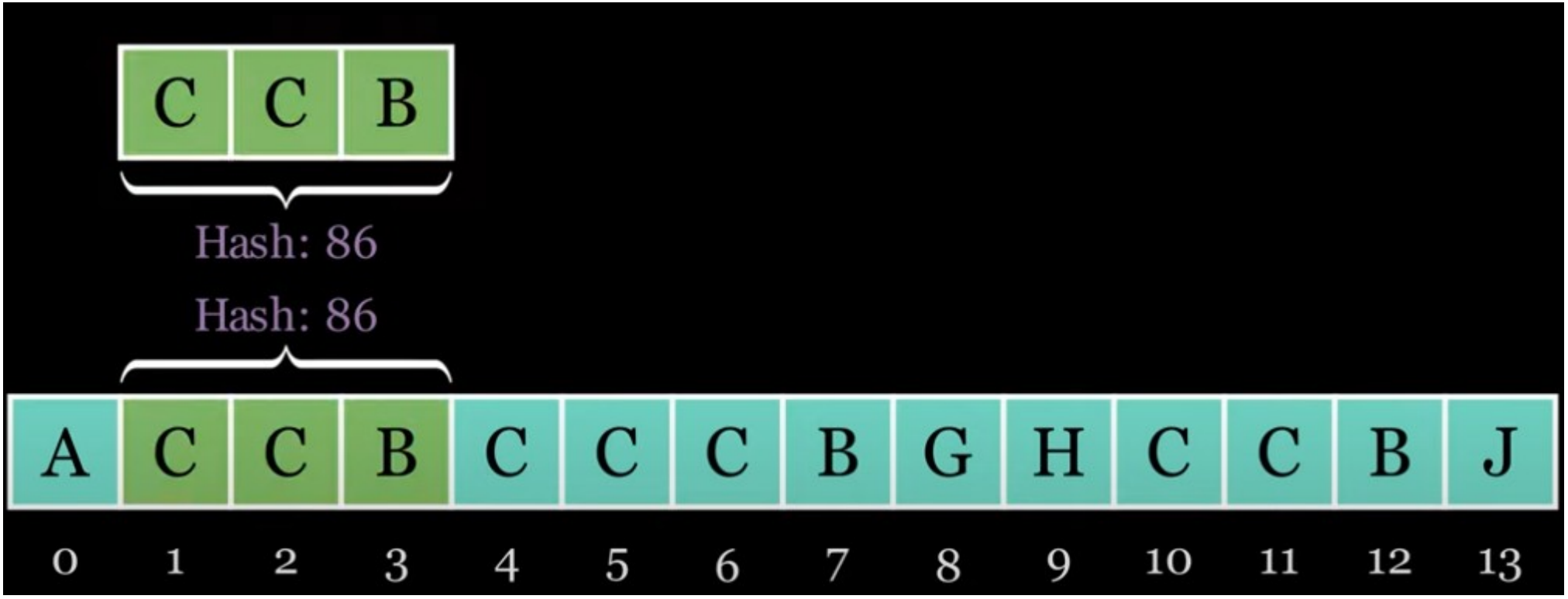
Rabin-Karp



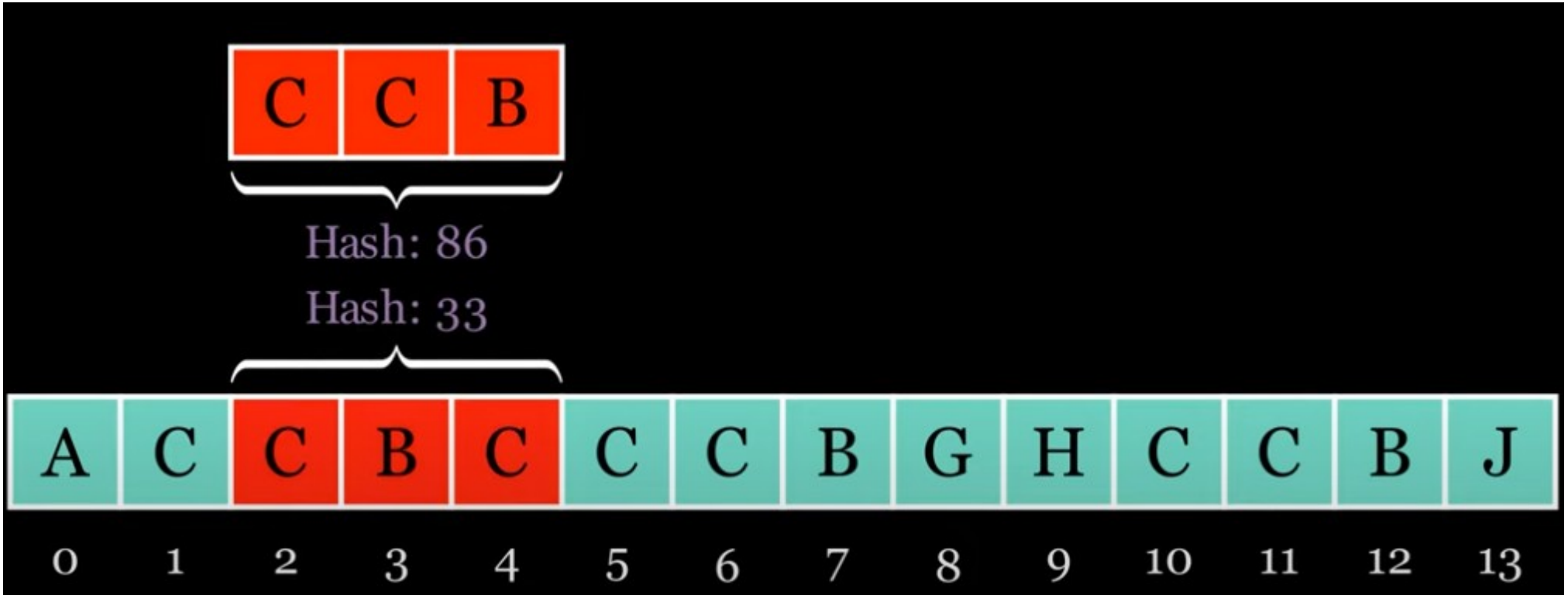
Rabin-Karp



Rabin-Karp

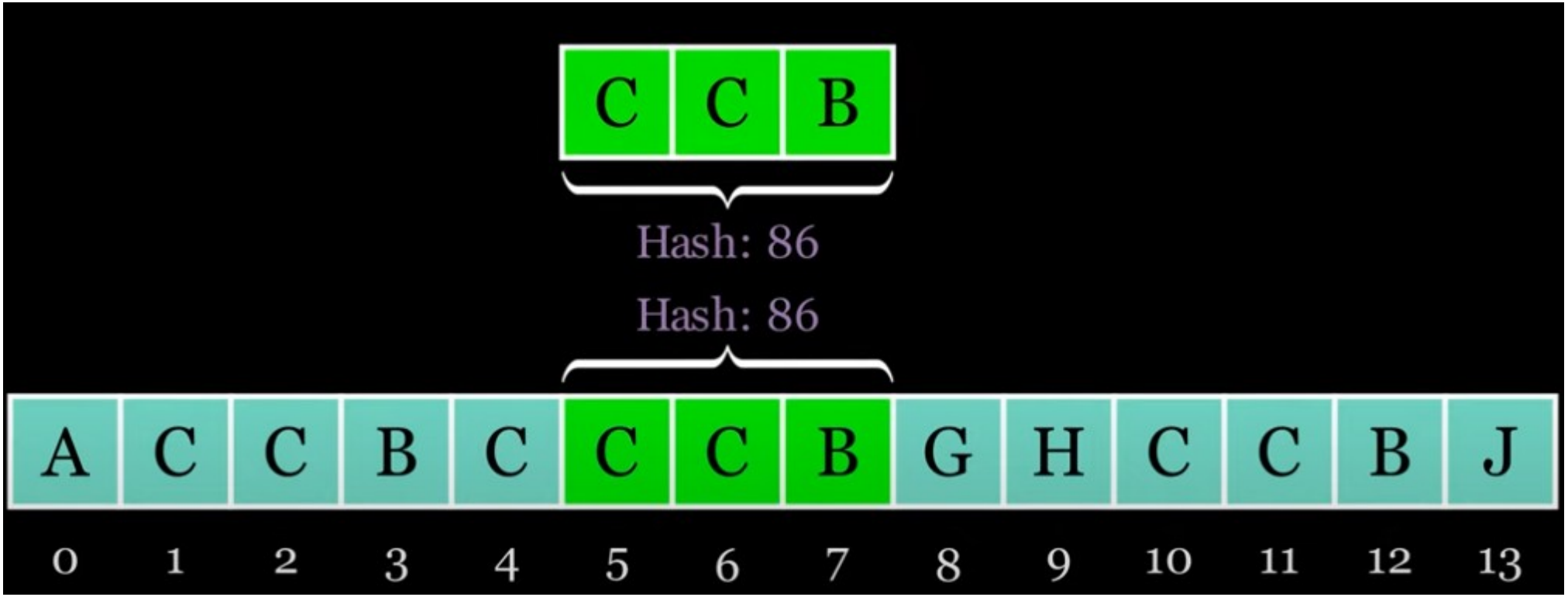


Rabin-Karp



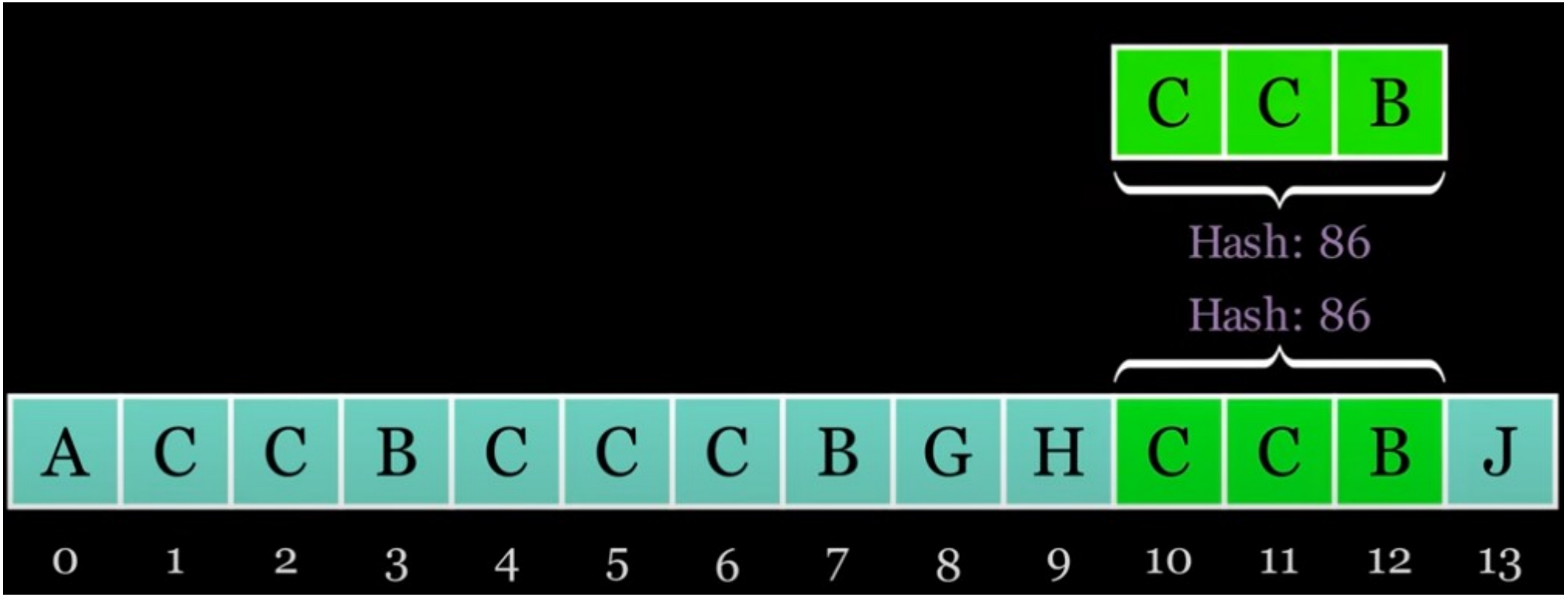
Rabin-Karp

.....



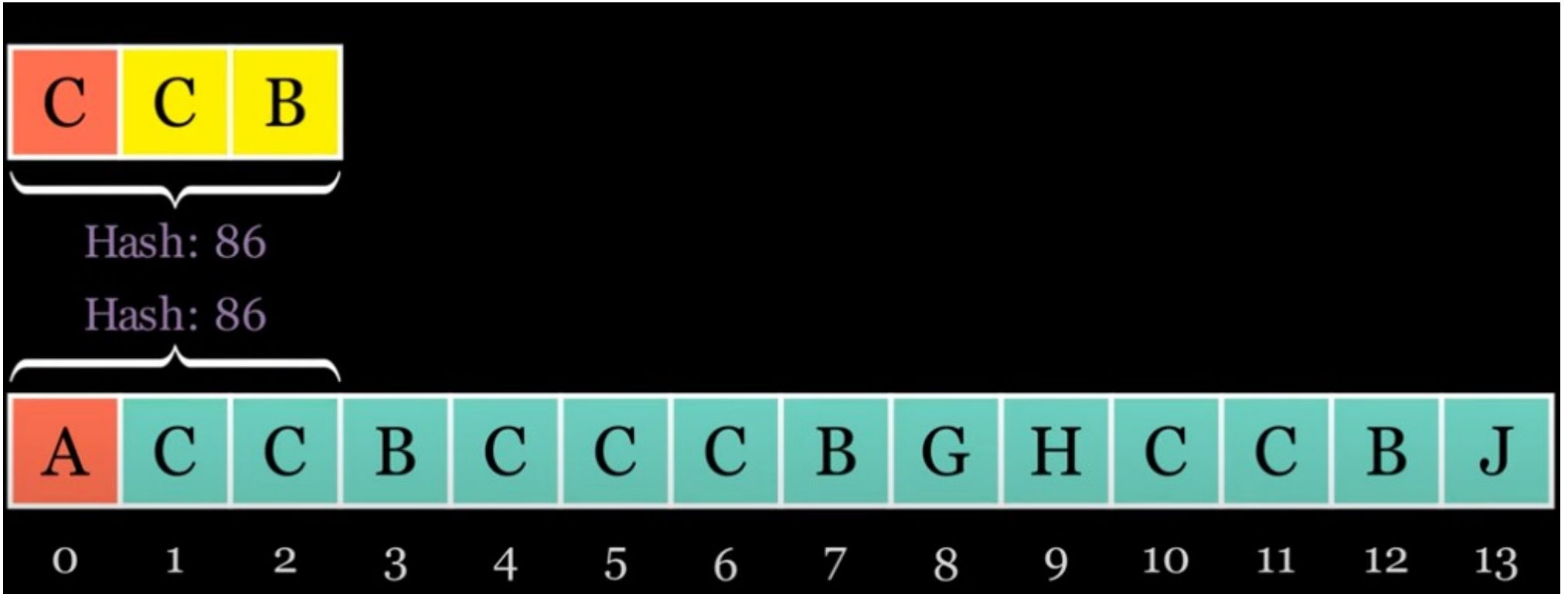
Rabin-Karp

.....



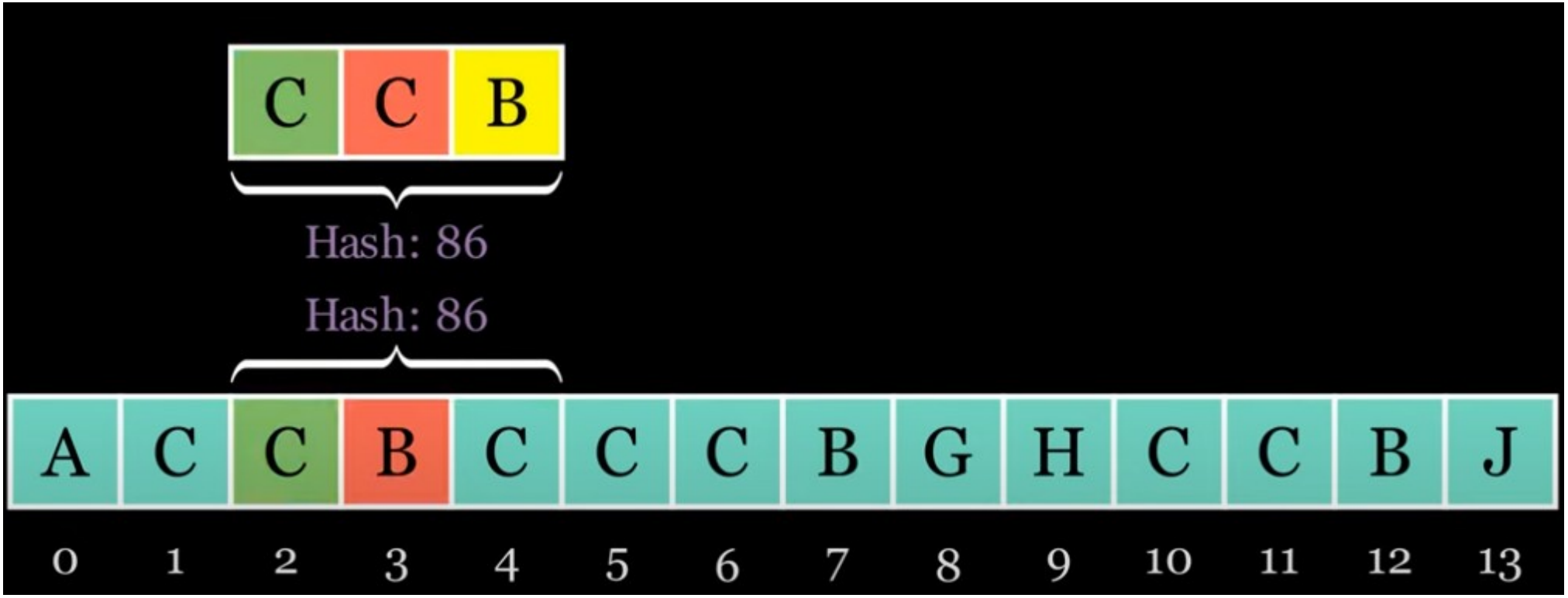
Rabin-Karp

Falso Positivo



Rabin-Karp

Falso Positivo

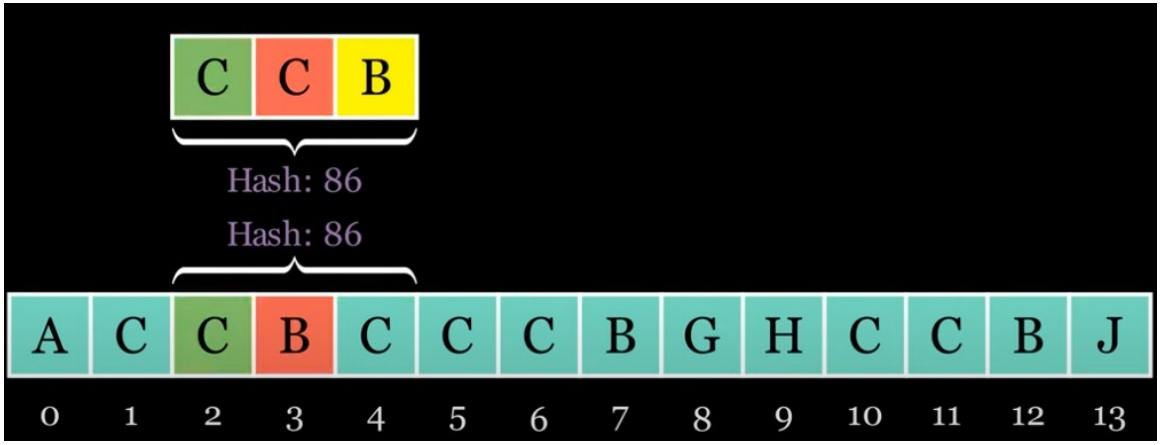


Pior Caso == Força Bruta

Rabin-Karp

- **Eficiente e Escalonável**
  - **Permite comparar múltiplos padrões (janelas)**
  - **Comparação por caracteres só quando hash colide**

- **Falsos Positivos**
  - **Pior Caso == Força Bruta**
  - **Módulo (p)**



Lets Code!!