

Longest Prefix Suffix Matching

Shusen Wang

Prefix

- If string X can be written as $X = PS$ for a nonempty string S , then P is a prefix of X .

Example string: $X = \text{"algorithm"}$

- `prefixes = {"algorithm", "algorit", "algori", "algor", "algo", "alg", "al", "a"}.`

Suffix

- If string X can be written as $X = PS$ for a nonempty string P , then S is a suffix of X .

Example string: $X = \text{"algorithm"}$

- `suffixes = {"lgorithm", "gorithm", "orithm", "rithm", "ithm", "thm", "hm", "m"}.`

Matching Prefixes and Suffixes

Example string: **X** = **"aba"**

- **prefixes** = { **"ab"**, **"a"** }.
- **suffixes** = { **"ba"**, **"a"** }.

Matching Prefixes and Suffixes

Example string: **X** = "aba"

- **prefixes** = { "ab", "a" } .
- **suffixes** = { "ba", "a" } .
- Their intersection: { "a" } .
- The longest matching: "a" .

Matching Prefixes and Suffixes

Example string: **X** = "ababa"

- **prefixes** = { "abab", "aba", "ab", "a" } .
- **suffixes** = { "baba", "aba", "ba", "a" } .

Matching Prefixes and Suffixes

Example string: **X** = "ababa"

- **prefixes** = { "abab", "aba", "ab", "a" } .
- **suffixes** = { "baba", "aba", "ba", "a" } .
- Their intersection: { "aba", "a" } .

Matching Prefixes and Suffixes

Example string: **X** = "ababa"

- **prefixes** = { "abab", "aba", "ab", "a" } .
- **suffixes** = { "baba", "aba", "ba", "a" } .
- Their intersection: { "aba", "a" } .
- The longest matching: "aba".

Longest Prefix Suffix Array

Longest Prefix Suffix Array

String: X =

a	b	a	b	a	b	c	a
?	?	?	?	?	?	?	?

Lengths: L =

Longest Prefix Suffix Array

String: X =	a	b	a	b	a	b	c	a
Lengths: L =	?							

- `prefixes = { }.`
- `suffixes = { }.`
- Their intersection: `{ }.`
- The longest matching: `empty string.` (Length = 0)

Longest Prefix Suffix Array

String: X =

a	b	a	b	a	b	c	a
0							

Lengths: L =

- `prefixes = { }.`
- `suffixes = { }.`
- Their intersection: `{ }.`
- The longest matching: `empty string.` (Length = 0)

Longest Prefix Suffix Array

String: X =	a	b	a	b	a	b	c	a
Lengths: L =	0	?						

- `prefixes = {"a"}.`
- `suffixes = {"b"}.`
- Their intersection: `{ }.`
- The longest matching: `empty string.` (Length = 0)

Longest Prefix Suffix Array

String: X =

a	b	a	b	a	b	c	a
0	0						

Lengths: L =

- `prefixes = {"a"}.`
- `suffixes = {"b"}.`
- Their intersection: `{ }.`
- The longest matching: `empty string.` (Length = 0)

Longest Prefix Suffix Array

String: X =	a	b	a	b	a	b	c	a
Lengths: L =	0	0	?					

- `prefixes = {"ab", "a"}`.
- `suffixes = {"ba", "a"}`.
- Their intersection: `{"a"}`.
- The longest matching: `"a"`. (**Length = 1**)

Longest Prefix Suffix Array

String: X =

a	b	a	b	a	b	c	a
0	0	1					

Lengths: L =

- `prefixes = {"ab", "a"}`.
- `suffixes = {"ba", "a"}`.
- Their intersection: `{"a"}`.
- The longest matching: `"a"`. (**Length = 1**)

Longest Prefix Suffix Array

String: X =	a	b	a	b	a	b	c	a
Lengths: L =	0	0	1	?				

- `prefixes = {"aba", "ab", "a"}`.
- `suffixes = {"bab", "ab", "a"}`.
- Their intersection: `{"ab", "a"}`.
- The longest matching: `"ab"`. (**Length = 2**)

Longest Prefix Suffix Array

String: X =	a	b	a	b	a	b	c	a
Lengths: L =	0	0	1	2				

- `prefixes = {"aba", "ab", "a"}`.
- `suffixes = {"bab", "ab", "a"}`.
- Their intersection: `{"ab", "a"}`.
- The longest matching: `"ab"`. (**Length = 2**)

Longest Prefix Suffix Array

String: X =	a	b	a	b	a	b	c	a
Lengths: L =	0	0	1	2	?			

- `prefixes` = {"abab", "aba", "ab", "a"}.
- `suffixes` = {"baba", "aba", "ba", "a"}.
- Their intersection: {"aba", "a"}.
- The longest matching: "aba". (Length = 3)

Longest Prefix Suffix Array

String: X =

a	b	a	b	a	b	c	a
0	0	1	2	3			

Lengths: L =

- `prefixes` = {"abab", "aba", "ab", "a"}.
- `suffixes` = {"baba", "aba", "ba", "a"}.
- Their intersection: {"aba", "a"}.
- The longest matching: "aba". (Length = 3)

Longest Prefix Suffix Array

String: X =	a	b	a	b	a	b	c	a
Lengths: L =	0	0	1	2	3	?		

- `prefixes` = {"ababa", "abab", "aba", "ab", "a"}.
- `suffixes` = {"babab", "abab", "bab", "ab", "b"}.
- Their intersection: {"abab", "ab"}.
- The longest matching: "abab". (Length = 4)

Longest Prefix Suffix Array

String: **X** =

a	b	a	b	a	b	c	a
0	0	1	2	3	4		

Lengths: **L** =

- **prefixes** = {"ababa", "abab", "aba", "ab", "a"}.
- **suffixes** = {"babab", "abab", "bab", "ab", "b"}.
- Their intersection: {"abab", "ab"}.
- The longest matching: "abab". (**Length = 4**)

Longest Prefix Suffix Array

String: X =	a	b	a	b	a	b	c	a
Lengths: L =	0	0	1	2	3	4	?	

- `prefixes` = {"ababab", "ababa", "abab", "aba", "ab", "a"}.
- `suffixes` = {"bababc", "ababc", "babc", "abc", "bc", "c"}.
- Their intersection: { }.
- The longest matching: **empty string**. (Length = 0)

Longest Prefix Suffix Array

String: X =

a	b	a	b	a	b	c	a
0	0	1	2	3	4	0	

Lengths: L =

- `prefixes` = {"ababab", "ababa", "abab", "aba", "ab", "a"}.
- `suffixes` = {"bababc", "ababc", "babcb", "abc", "bc", "c"}.
- Their intersection: { }.
- The longest matching: **empty string**. (Length = 0)

Longest Prefix Suffix Array

String: X =

a	b	a	b	a	b	c	a
0	0	1	2	3	4	0	?

Lengths: L =

- `prefixes` = {`"abababc"`, `"ababab"`, `"ababa"`, `"abab"`, `"aba"`, `"ab"`, `"a"`}.
- `suffixes` = {`"bababca"`, `"ababca"`, `"babca"`, `"abca"`, `"bca"`, `"ca"`, `"a"`}.
- Their intersection: { `"a"` }.
- The longest matching: `"a"`. (Length = 1)

Longest Prefix Suffix Array

String: X =

a	b	a	b	a	b	c	a
0	0	1	2	3	4	0	1

Lengths: L =

- `prefixes` = {"abababc", "ababab", "ababa", "abab", "aba", "ab", "a"}.
- `suffixes` = {"bababca", "ababca", "babca", "abca", "bca", "ca", "a"}.
- Their intersection: { "a" }.
- The longest matching: "a". (Length = 1)

Longest Prefix Suffix Array

String: **X** =

a	b	a	b	a	b	c	a
0	0	1	2	3	4	0	1

Lengths: **L** =

What does the number mean?

Longest Prefix Suffix Array

		The first 3 chars			The last 3 chars					
String: X =		a	b	a	b	a	b	c	a	
Lengths: L =		0	0	1	2	3	4	0	1	

What does the number mean?

Application

Why is the longest prefix suffix array interesting?

- The array is used by the Knuth–Morris–Pratt (KMP) algorithm [\[1\]](#).
- KMP algorithm solves the string matching problem.

Reference:

1. Knuth, Morris, & Pratt. [Fast pattern matching in strings](#). *SIAM Journal on Computing*, 6 (2): 323–350, 1977.

Linear-Time Algorithm

Longest Prefix Suffix Array

String: X =

a	b	c	...	a	b
0	0	0	...	1	2

Lengths: L =

Longest Prefix Suffix Array

String: X =	a	b	c	...	a	b
Lengths: L =	0	0	0	...	1	2

What does the value mean?

Longest Prefix Suffix Array

	Prefix				Suffix	
String: $X =$	a	b	c	...	a	b
Lengths: $L =$	0	0	0	...	1	2

What does the value mean?

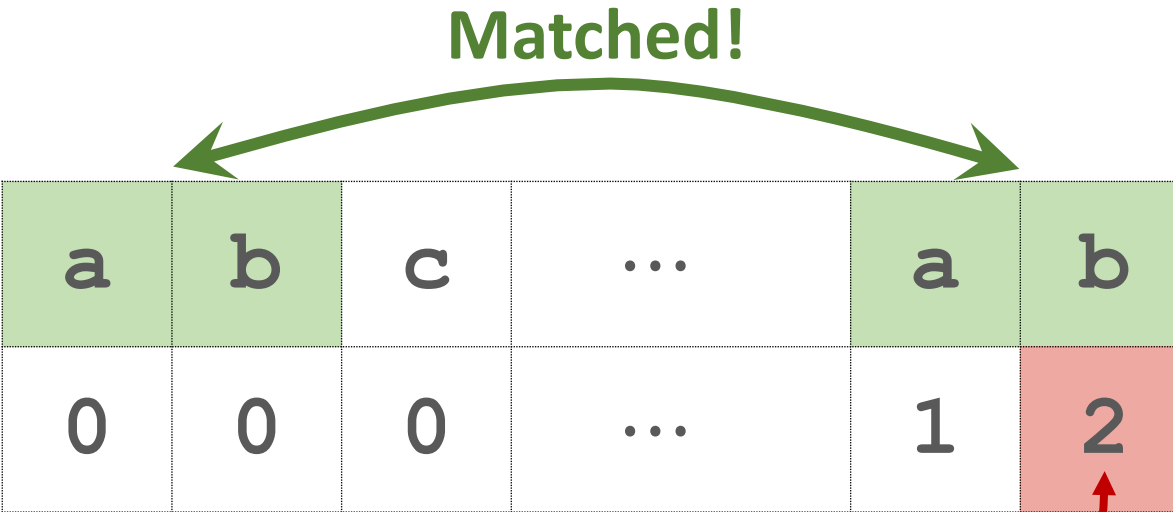
Longest Prefix Suffix Array

String: $X =$

Lengths: $L =$

a	b	c	...	a	b
0	0	0	...	1	2

Matched!



What does the value mean?

What is the next element in the array?

String: X =

a	b	c	...	a	b
0	0	0	...	1	2

Lengths: L =

What is the next element in the array?

Append a new character to **X**

String: X =	a	b	c	...	a	b	c
Lengths: L =	0	0	0	...	1	2	

What is the next element in the array?

Append a new character to **X**

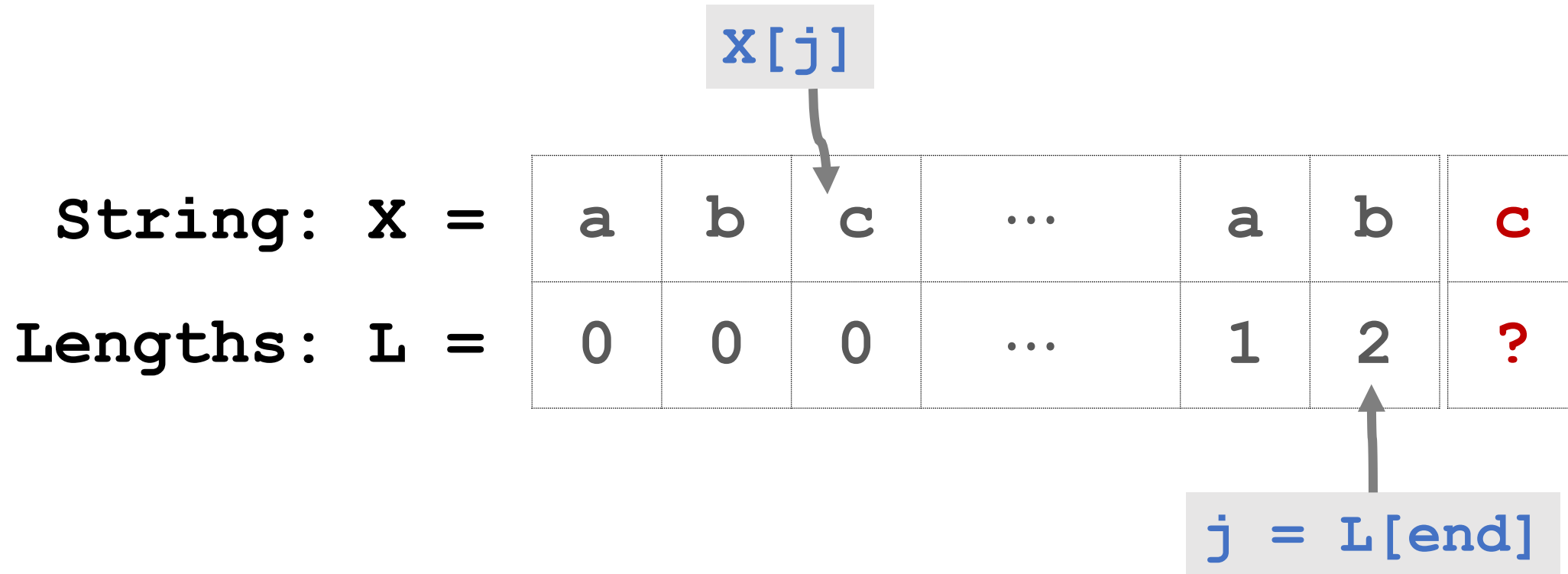
String: **X** =

Lengths: **L** =

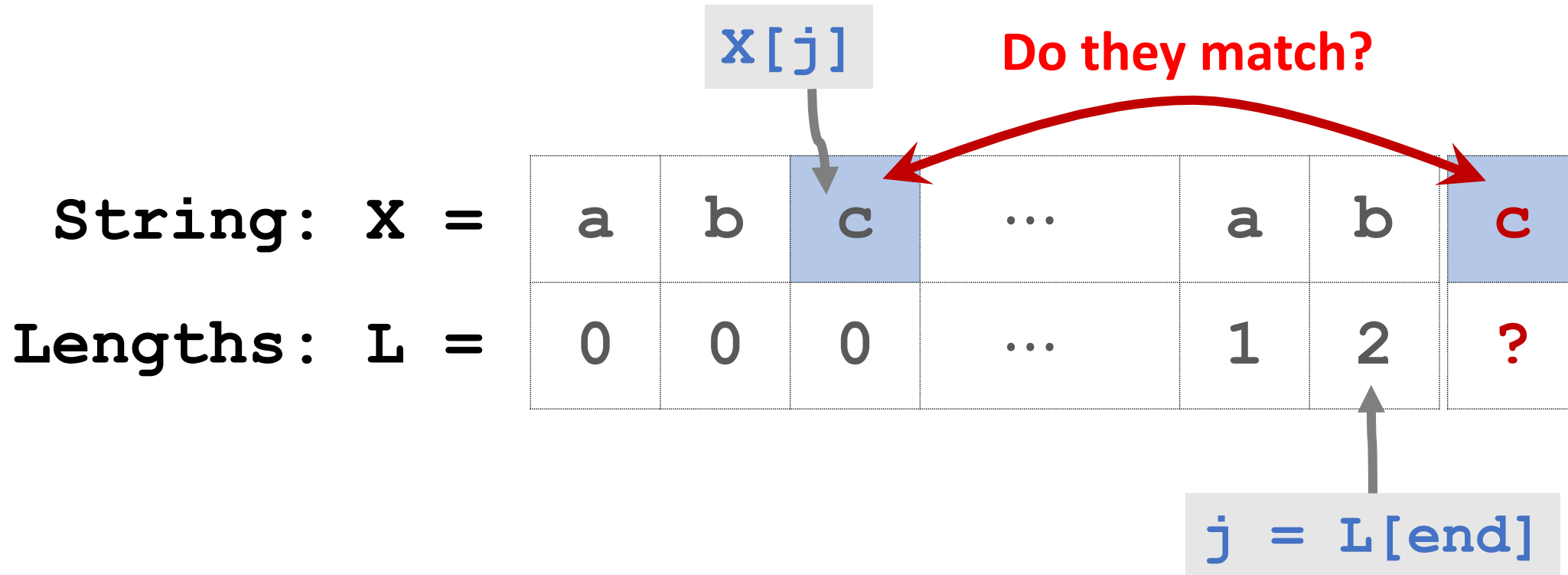
a	b	c	...	a	b	c
0	0	0	...	1	2	?

What is the new value?

Three Cases



Three Cases

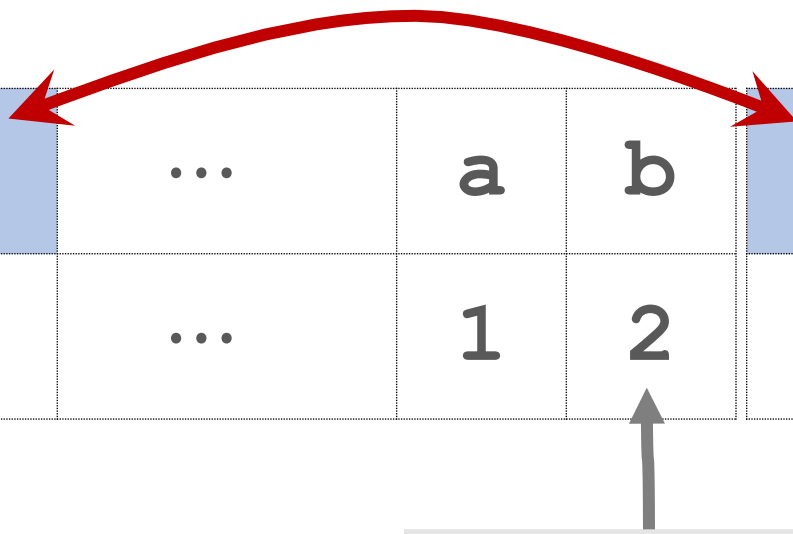


Three Cases

Do they match?

String: X =	a	b	c	...	a	b	c
Lengths: L =	0	0	0	...	1	2	?

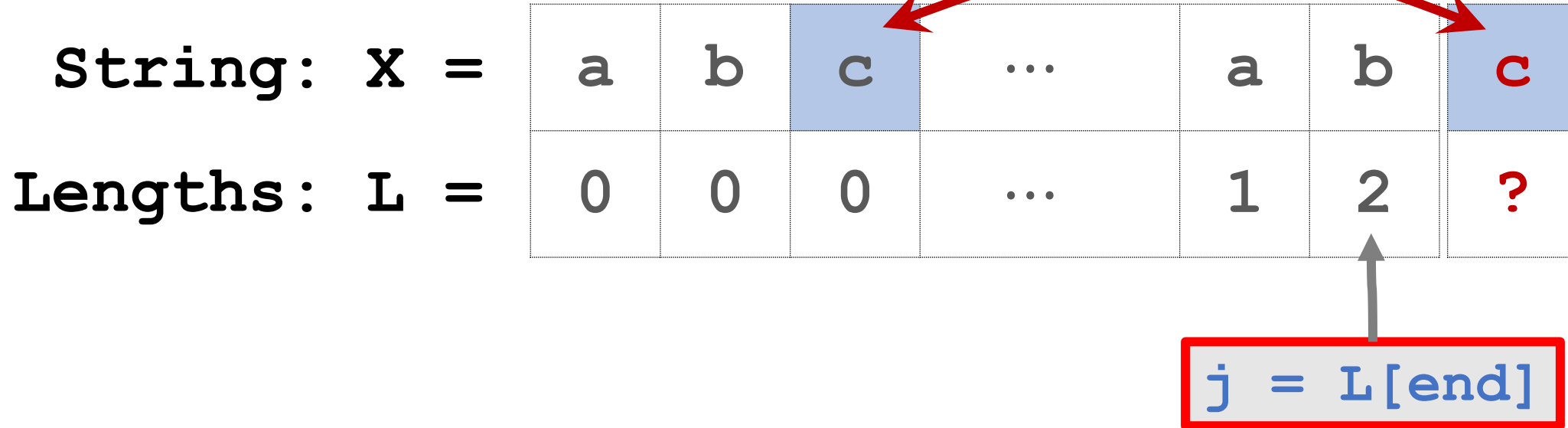
$j = L[\text{end}]$



- Match ==> Case 1;

Three Cases

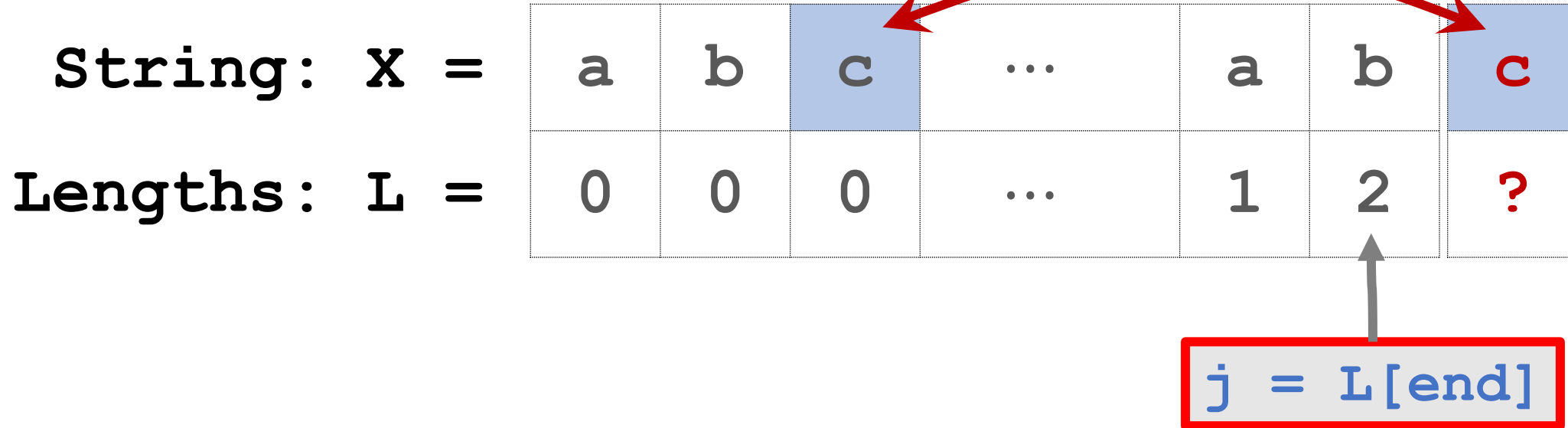
Do they match?



- Match ==> Case 1;
- Mismatch ==>

Three Cases

Do they match?

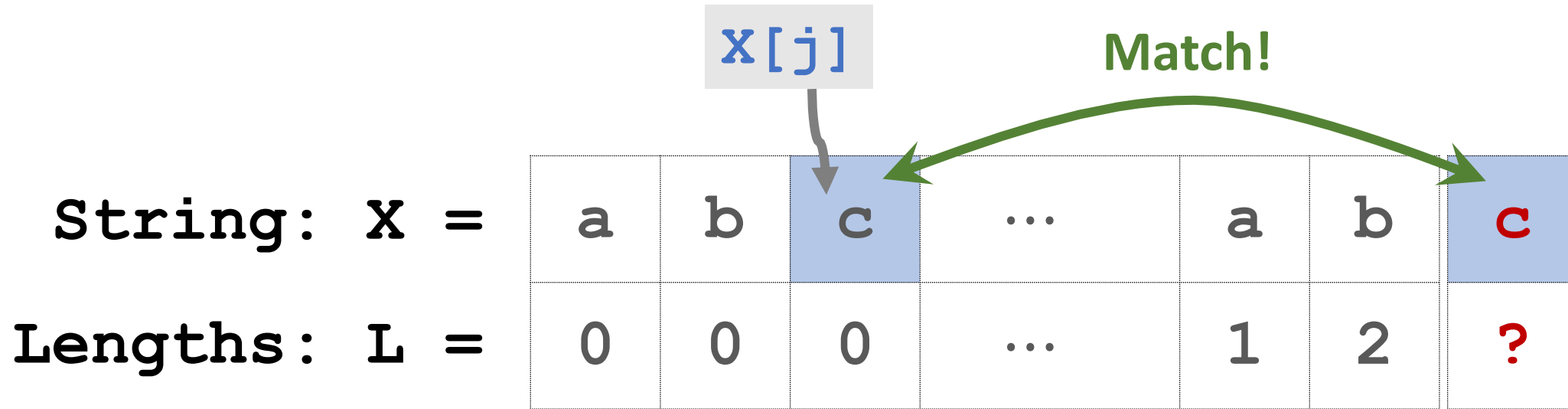


- Match ==> Case 1;

- Mismatch ==>

j = 0 ==> Case 2;
j ≠ 0 ==> Case 3.

Case 1: A New Match



- Case 1: the new char is equal to $x[j]$.

Case 1: A New Match

String: X =	a	b	c	...	a	b	c
Lengths: L =	0	0	0	...	1	2	?

$j = L[\text{end}]$

- Case 1 \implies Let the new value in L be $j+1$.

Case 1: A New Match

String: X =	a	b	c	...	a	b	c
Lengths: L =	0	0	0	...	1	2	3


$j = L[\text{end}]$

- Case 1 ==> Let the new value in L be $j+1$.

Case 1: A New Match

String: X =	a	b	c	...	a	b	c
Lengths: L =	0	0	0	...	1	2	3

$j = L[\text{end}]$



Question: Why is the new value equal to $j+1$?

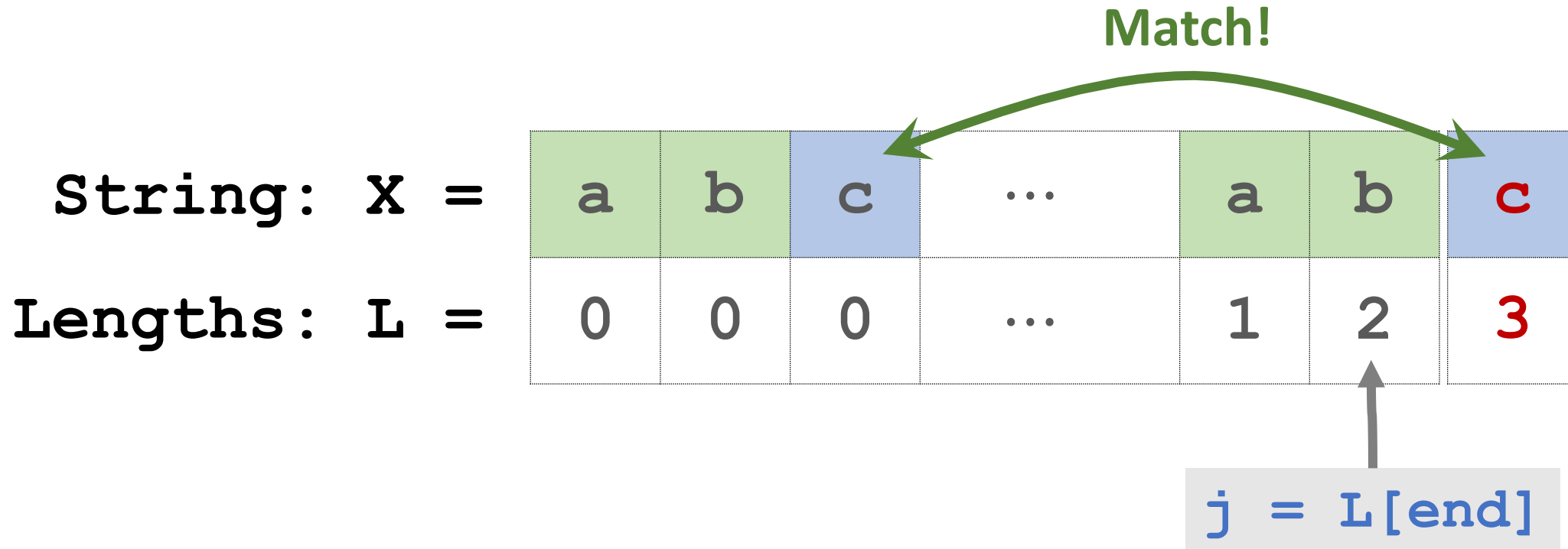
Case 1: A New Match

String: X =	a	b	c	...	a	b	c
Lengths: L =	0	0	0	...	1	2	3

$j = L[\text{end}]$

Question: Why is the new value equal to $j+1$?

Case 1: A New Match



Question: Why is the **new value** equal to $j+1$?

Case 1: A New Match

Another Example:

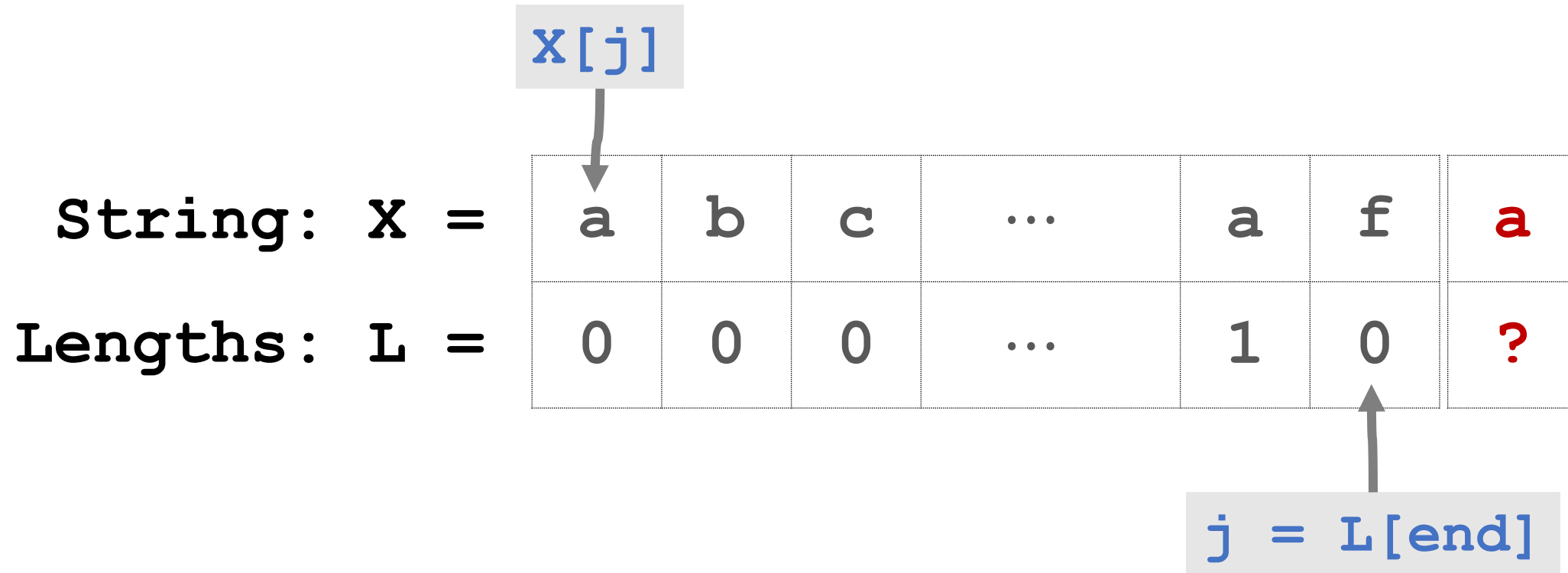
String: X =	a	b	c	...	a	f
Lengths: L =	0	0	0	...	1	0

Case 1: A New Match

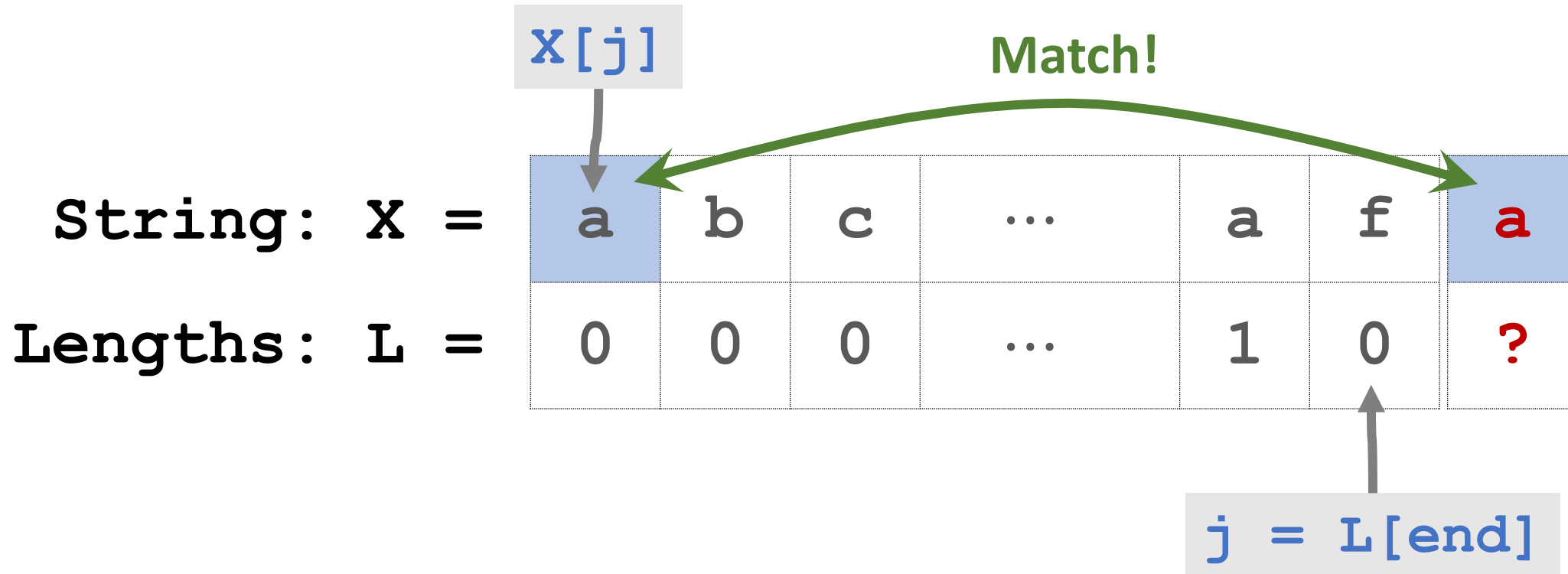
Append a new character to **X**

String: X =	a	b	c	...	a	f	a
Lengths: L =	0	0	0	...	1	0	?

Case 1: A New Match



Case 1: A New Match



- Case 1: the new char is equal to $x[j]$.

Case 1: A New Match

String: X =	a	b	c	...	a	f	a
Lengths: L =	0	0	0	...	1	0	1

$j = L[\text{end}]$

- Case 1 ==> Let the new value of L be $j+1$.

Case 2: Mismatch and $L[end]==0$

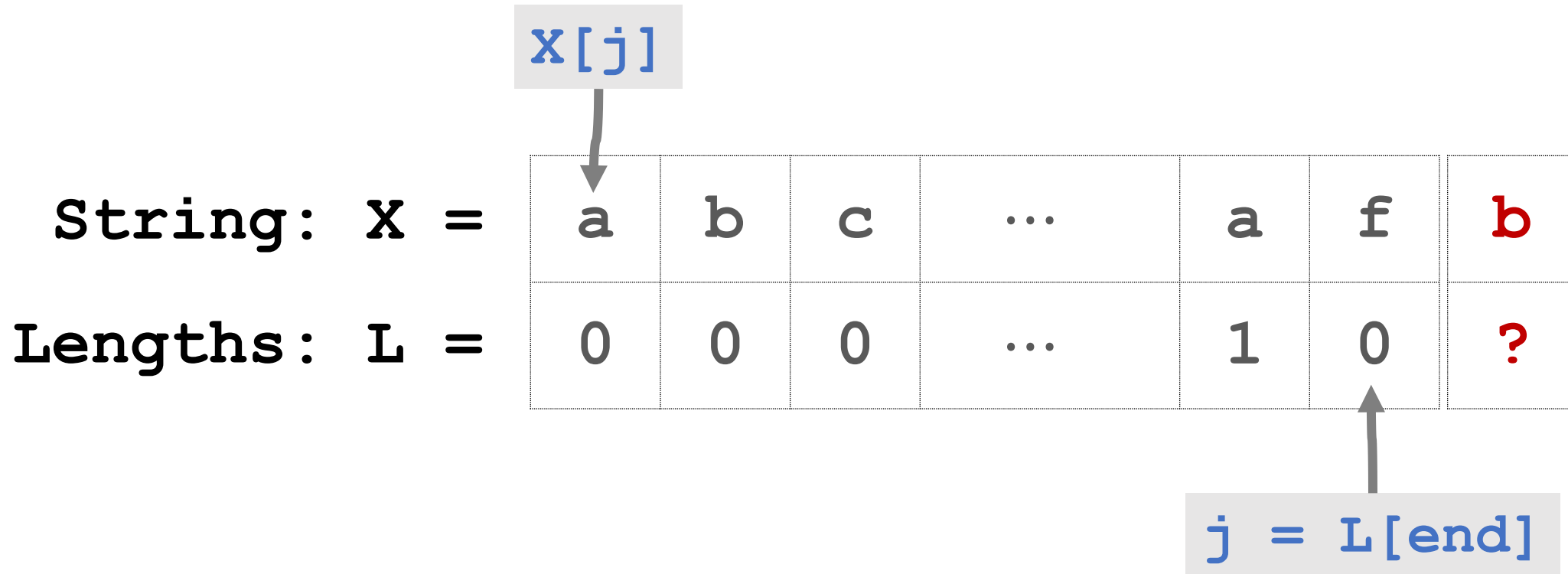
String: X =	a	b	c	...	a	f
Lengths: L =	0	0	0	...	1	0

Case 2: Mismatch and $L[end] == 0$

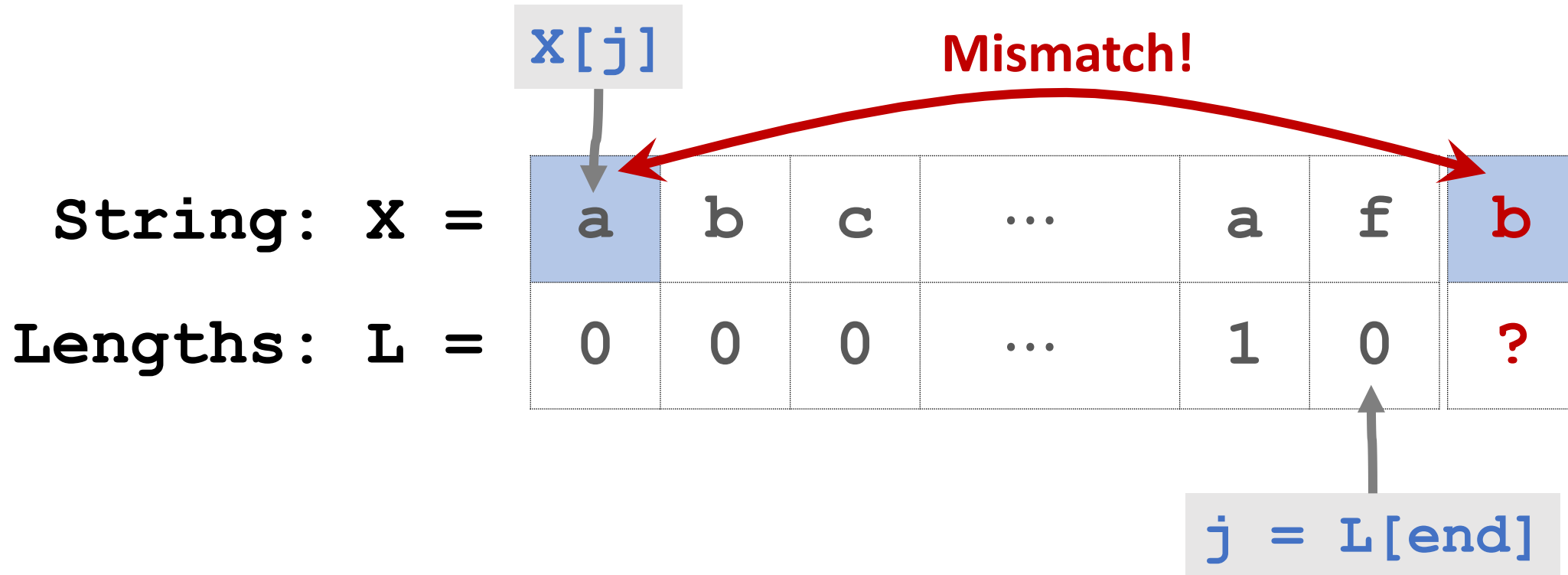
Append a new character to **X**

String: X =	a	b	c	...	a	f	b
Lengths: L =	0	0	0	...	1	0	?

Case 2: Mismatch and $L[\text{end}] == 0$



Case 2: Mismatch and $L[\text{end}] == 0$



- Mismatch $\implies \begin{cases} j = 0 & \implies \text{Case 2;} \\ j \neq 0 & \implies \text{Case 3.} \end{cases}$

Case 2: Mismatch and $L[\text{end}] == 0$

String: X =	a	b	c	...	a	f	b
Lengths: L =	0	0	0	...	1	0	?

$j = L[\text{end}]$

- Mismatch $\implies \begin{cases} j = 0 \implies \text{Case 2;} \\ j \neq 0 \implies \text{Case 3.} \end{cases}$

Case 2: Mismatch and $L[end]==0$

String: X =	a	b	c	...	a	f	b
Lengths: L =	0	0	0	...	1	0	?

- Case 2 \implies Let the new value be 0.

Case 2: Mismatch and $L[end]==0$

String: X =	a	b	c	...	a	f	b
Lengths: L =	0	0	0	...	1	0	0

- Case 2 \implies Let the new value be 0.

Case 3: Mismatch and $L[end] \neq 0$

String: X =	a	b	a	f	...	a	b	a
Lengths: L =	0	0	1	0	...	1	2	3

Case 3: Mismatch and $L[end] \neq 0$

Append a new character to **X**

String: X =	a	b	a	f	...	a	b	a	b
Lengths: L =	0	0	1	0	...	1	2	3	?

Case 3: Mismatch and $L[\text{end}] \neq 0$

$x[j]$



String: $X =$

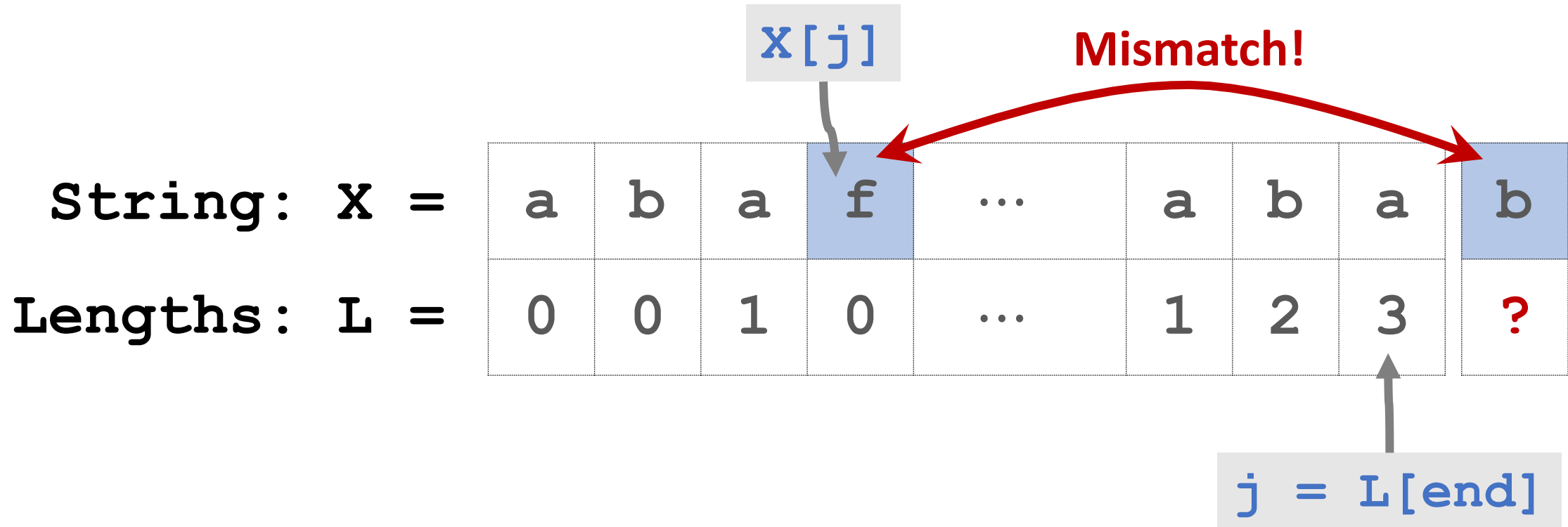
a	b	a	f	...	a	b	a	b
0	0	1	0	...	1	2	3	?

Lengths: $L =$

$j = L[\text{end}]$



Case 3: Mismatch and $L[\text{end}] \neq 0$



- Mismatch $\implies \begin{cases} j = 0 & \implies \text{Case 2;} \\ j \neq 0 & \implies \text{Case 3.} \end{cases}$

Case 3: Mismatch and $L[\text{end}] \neq 0$

String: X =	a	b	a	f	...	a	b	a	b
Lengths: L =	0	0	1	0	...	1	2	3	?

$j = L[\text{end}]$

- Mismatch \Rightarrow

$\begin{cases} j = 0 \Rightarrow \text{Case 2;} \\ j \neq 0 \Rightarrow \text{Case 3.} \end{cases}$

Case 3: Mismatch and $L[\text{end}] \neq 0$

String: X =	a	b	a	f	...	a	b	a	b
Lengths: L =	0	0	1	0	...	1	2	3	?

↑
 $j = L[\text{end}]$

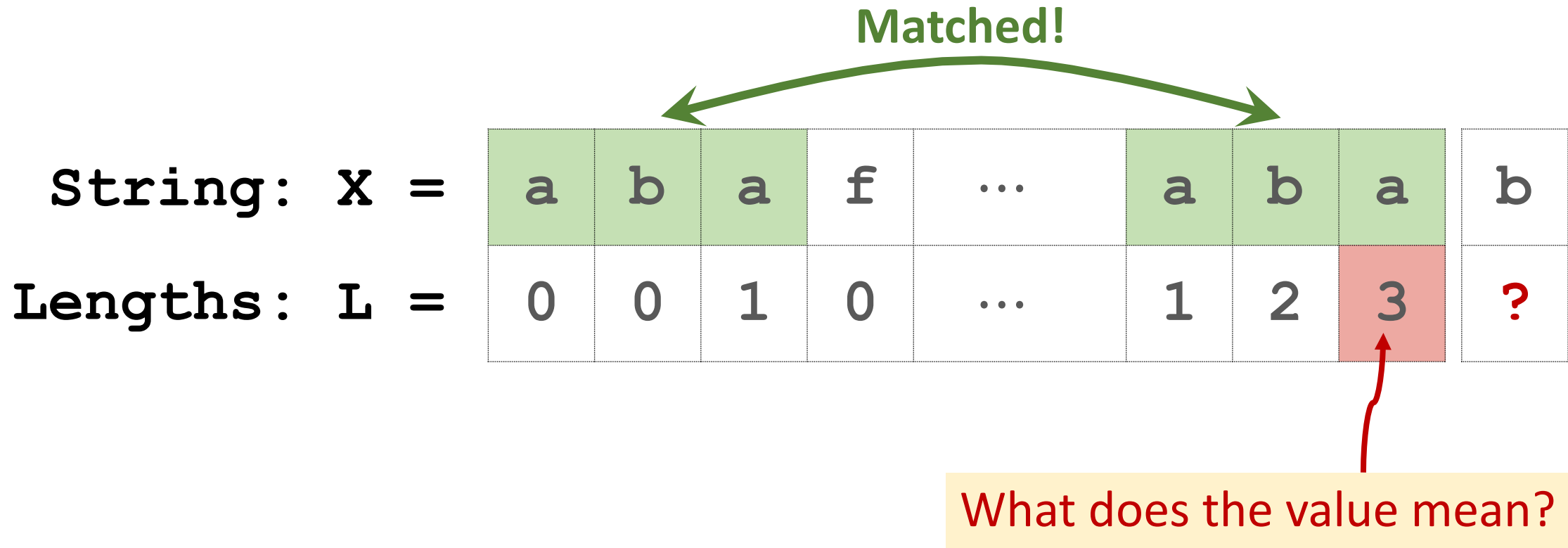
- **Case 3** ==> Reduce the big problem to a smaller one.

Case 3: Mismatch and $L[\text{end}] \neq 0$

String: X =	a	b	a	f	...	a	b	a	b
Lengths: L =	0	0	1	0	...	1	2	3	?

What does the value mean?

Case 3: Mismatch and $L[\text{end}] \neq 0$



Case 3: Mismatch and $L[\text{end}] \neq 0$

String: X =	a	b	a	f	...	a	b	a	b
Lengths: L =	0	0	1	0	...	1	2	3	?

String: X' =

Lengths: L' =

Case 3: Mismatch and $L[\text{end}] \neq 0$

String: X =	a	b	a	f	...	a	b	a	b
Lengths: L =	0	0	1	0	...	1	2	3	?

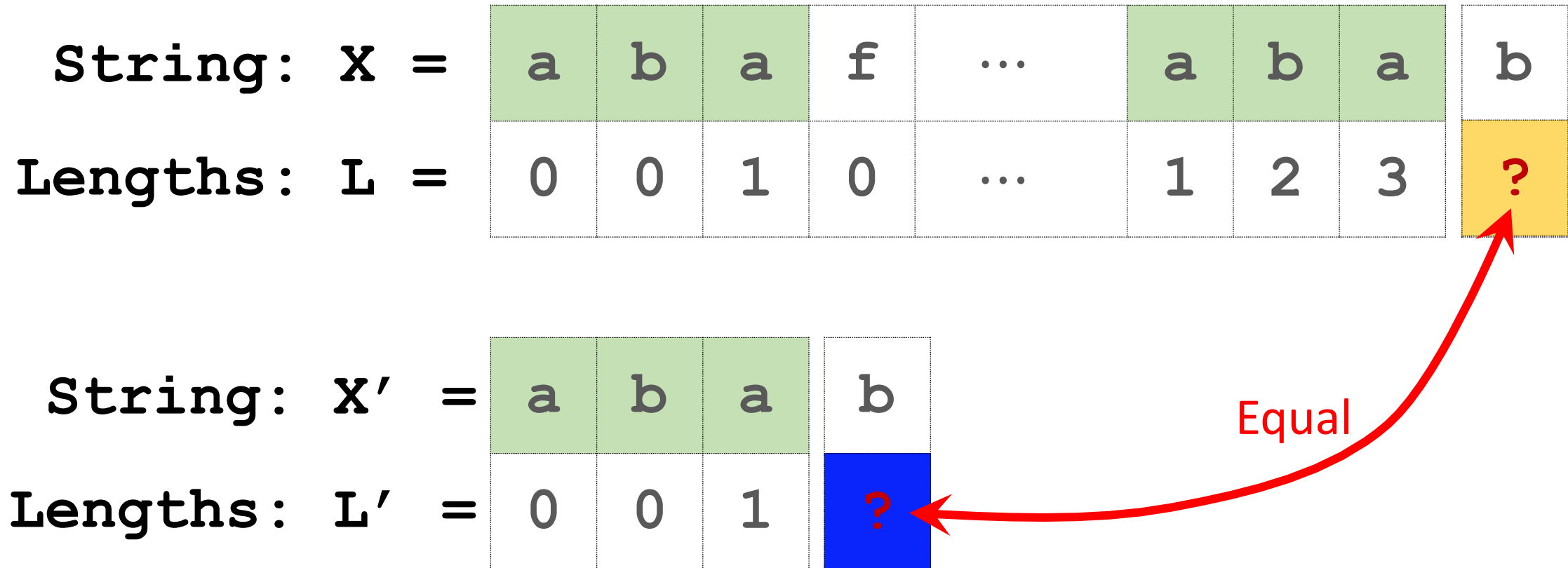
String: X' =	a	b	a
Lengths: L' =	0	0	1

Case 3: Mismatch and $L[\text{end}] \neq 0$

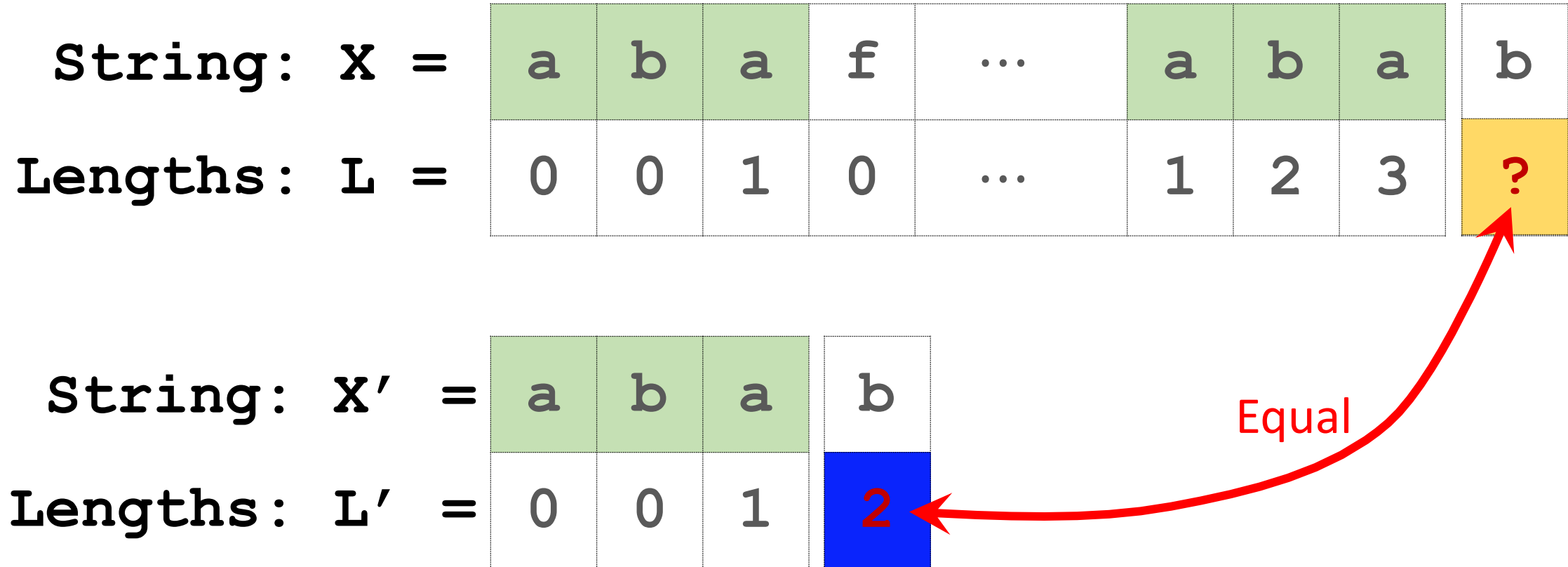
String: X =	a	b	a	f	...	a	b	a	b
Lengths: L =	0	0	1	0	...	1	2	3	?

String: X' =	a	b	a	b
Lengths: L' =	0	0	1	?

Case 3: Mismatch and $L[\text{end}] \neq 0$



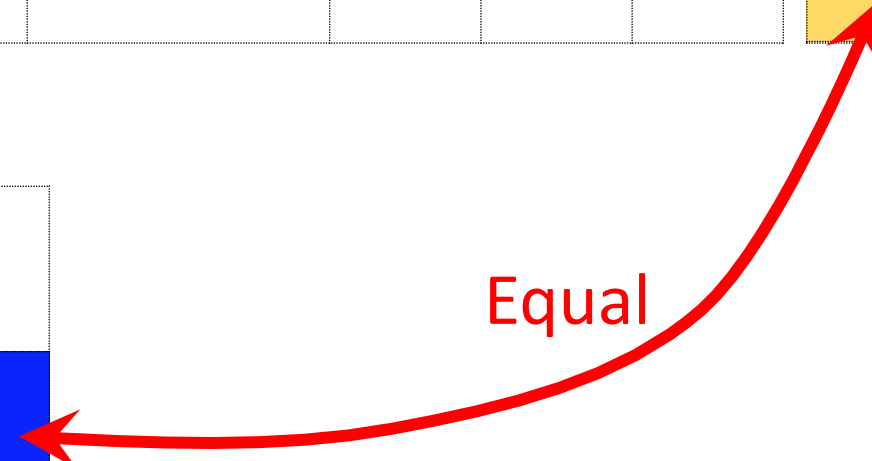
Case 3: Mismatch and $L[\text{end}] \neq 0$



Case 3: Mismatch and $L[\text{end}] \neq 0$

String: $X =$	a	b	a	f	...	a	b	a	b
Lengths: $L =$	0	0	1	0	...	1	2	3	2

String: $X' =$	a	b	a	b
Lengths: $L' =$	0	0	1	2



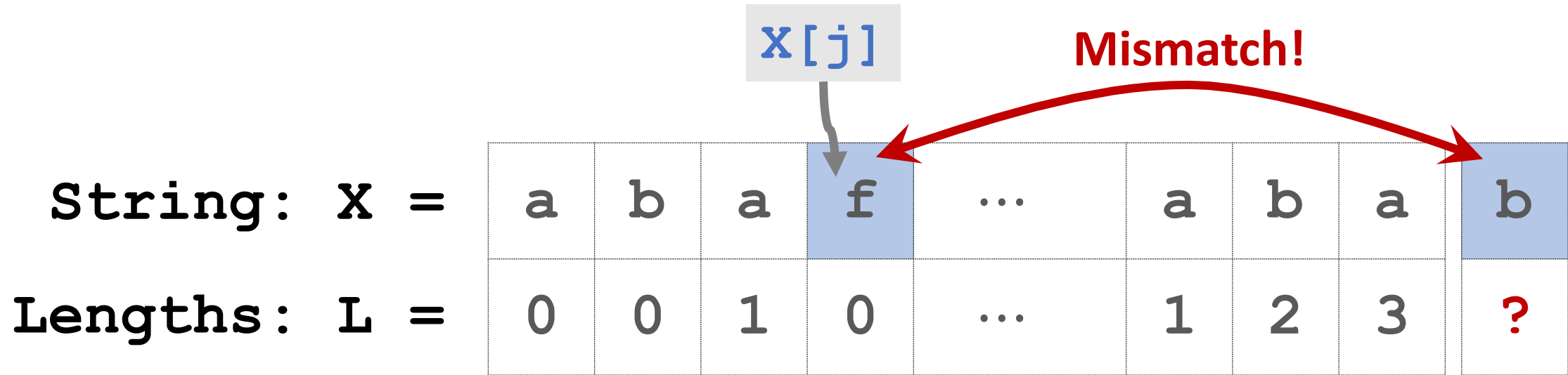
Equal

Case 3: Mismatch and $L[\text{end}] \neq 0$

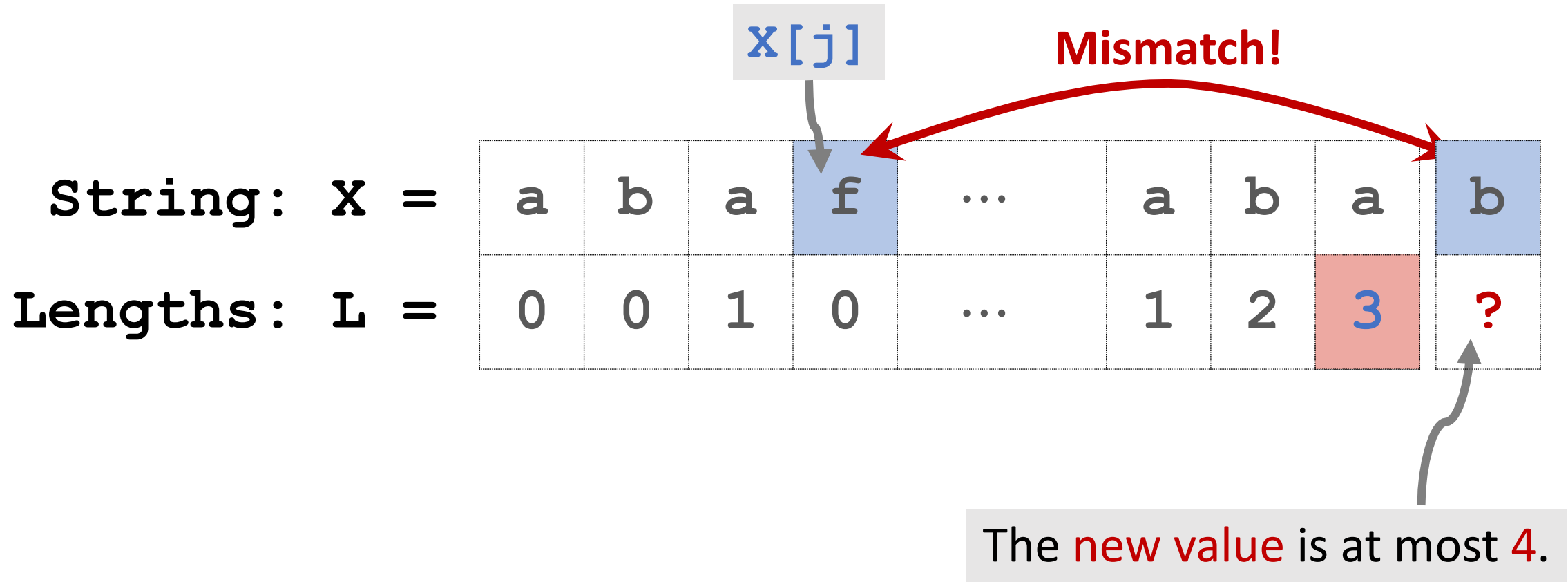
String: X =	a	b	a	f	...	a	b	a	b
Lengths: L =	0	0	1	0	...	1	2	3	?

Question: Why can Case 3 be solved in this way?

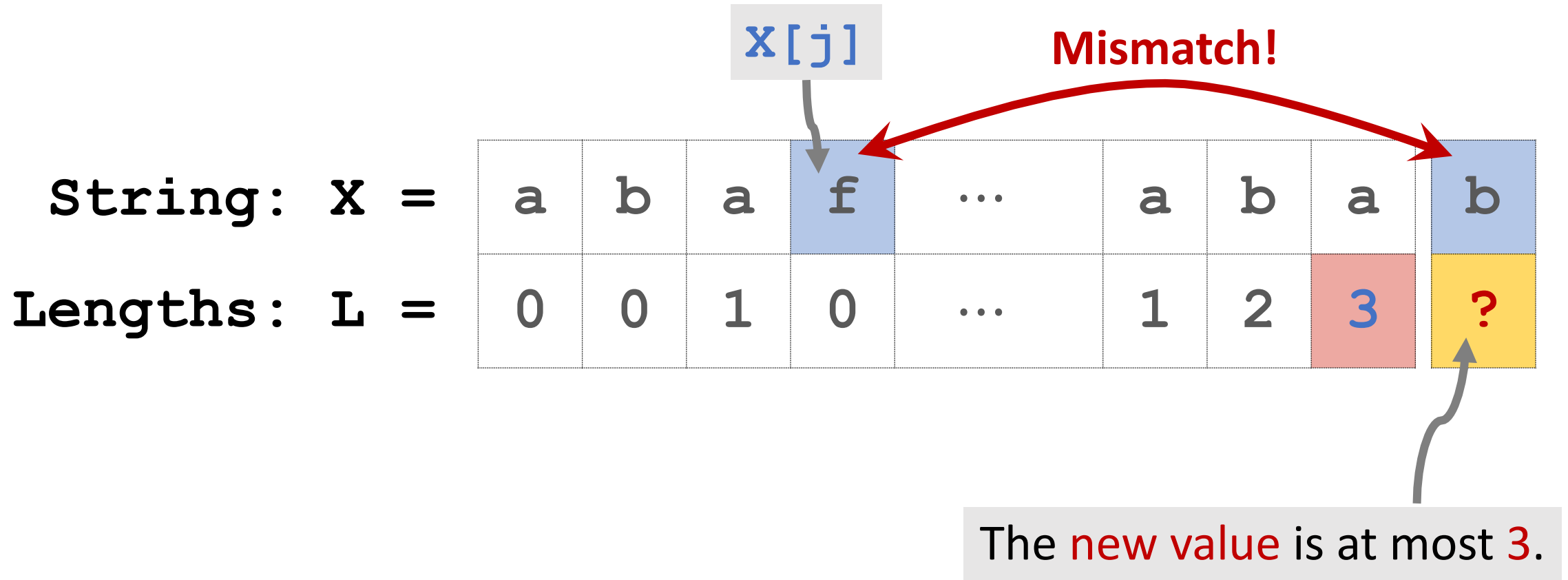
Case 3: Mismatch and $L[\text{end}] \neq 0$



Case 3: Mismatch and $L[\text{end}] \neq 0$



Case 3: Mismatch and $L[\text{end}] \neq 0$



Case 3: Mismatch and $L[\text{end}] \neq 0$

String: X =	a	b	a	f	...	a	b	a	b
Lengths: L =	0	0	1	0	...	1	2	3	?

The new value is at most 3.

Case 3: Mismatch and $L[\text{end}] \neq 0$

String: X =	a	b	a	f	...	a	b	a	b
Lengths: L =	0	0	1	0	...	1	2	3	?

Case 3: Mismatch and $L[\text{end}] \neq 0$

Question: Why are the yellow entry and blue entry equal?

String: $X =$	a	b	a	f	...	a	b	a	b
Lengths: $L =$	0	0	1	0	...	1	2	3	2

String: $X' =$	a	b	a	b
Lengths: $L' =$	0	0	1	?

Equal

Case 3: Mismatch and $L[\text{end}] \neq 0$

Question: Why are the yellow entry and blue entry equal?

→ String: $X =$

a	b	a	f	...	a	b	a	b
0	0	1	0	...	1	2	3	2

Lengths: $L =$

X and X' have the same prefixes.

→ String: $X' =$

a	b	a	b
0	0	1	?

Lengths: $L' =$

Case 3: Mismatch and $L[\text{end}] \neq 0$

Question: Why are the yellow entry and blue entry equal?

→ String: $X =$

a	b	a	f	...	a	b	a	b
---	---	---	---	-----	---	---	---	---

Lengths: $L =$

0	0	1	0	...	1	2	3	2
---	---	---	---	-----	---	---	---	---

X and X' have the same prefixes.

→ String: $X' =$

a	b	a	b
---	---	---	---

Lengths: $L' =$

0	0	1	?
---	---	---	---

Case 3: Mismatch and $L[\text{end}] \neq 0$

Question: Why are the yellow entry and blue entry equal?

→ String: $X =$

a	b	a	f	...	a	b	a	b
0	0	1	0	...	1	2	3	2

Lengths: $L =$

X and X' have the same prefixes.

→ String: $X' =$

a	b	a	b
0	0	1	?

Lengths: $L' =$

Case 3: Mismatch and $L[\text{end}] \neq 0$

Question: Why are the yellow entry and blue entry equal?

➔ String: $X =$

a	b	a	f	...	a	b	a	b
0	0	1	0	...	1	2	3	2

Lengths: $L =$

X and X' have the same suffixes.

➔ String: $X' =$

a	b	a	b
0	0	1	?

Lengths: $L' =$

Case 3: Mismatch and $L[\text{end}] \neq 0$

Question: Why are the yellow entry and blue entry equal?

➔ String: $X =$

a	b	a	f	...	a	b	a	b
0	0	1	0	...	1	2	3	2

Lengths: $L =$

X and X' have the same suffixes.

➔ String: $X' =$

a	b	a	b
0	0	1	?

Lengths: $L' =$

Case 3: Mismatch and $L[\text{end}] \neq 0$

Question: Why are the yellow entry and blue entry equal?

➔ String: $X =$

a	b	a	f	...	a	b	a	b
0	0	1	0	...	1	2	3	2

Lengths: $L =$

X and X' have the same suffixes.

➔ String: $X' =$

a	b	a	b
0	0	1	?

Lengths: $L' =$

Case 3: Mismatch and $L[\text{end}] \neq 0$

Question: Why are the yellow entry and blue entry equal?

String: $X =$	a	b	a	f	...	a	b	a	b
Lengths: $L =$	0	0	1	0	...	1	2	3	2

String: $X' =$	a	b	a	b
Lengths: $L' =$	0	0	1	?

Equal

Summary

Longest Prefixes Suffixes Matching

Example string: $X = \text{"ababa"}$

- `prefixes` = { `"abab"`, `"aba"`, `"ab"`, `"a"` }.
- `suffixes` = { `"baba"`, `"aba"`, `"ba"`, `"a"` }.

Longest Prefixes Suffixes Matching

Example string: $X = \text{"ababa"}$

- `prefixes` = { `"abab"`, `"aba"`, `"ab"`, `"a"` } .
- `suffixes` = { `"baba"`, `"aba"`, `"ba"`, `"a"` } .
- Their intersection: { `"aba"`, `"a"` } .

Longest Prefixes Suffixes Matching

Example string: $X = \text{"ababa"}$

- `prefixes` = { `"abab"`, `"aba"`, `"ab"`, `"a"` }.
- `suffixes` = { `"baba"`, `"aba"`, `"ba"`, `"a"` }.
- Their intersection: { `"aba"`, `"a"` }.
- The longest matching: `"aba"`.

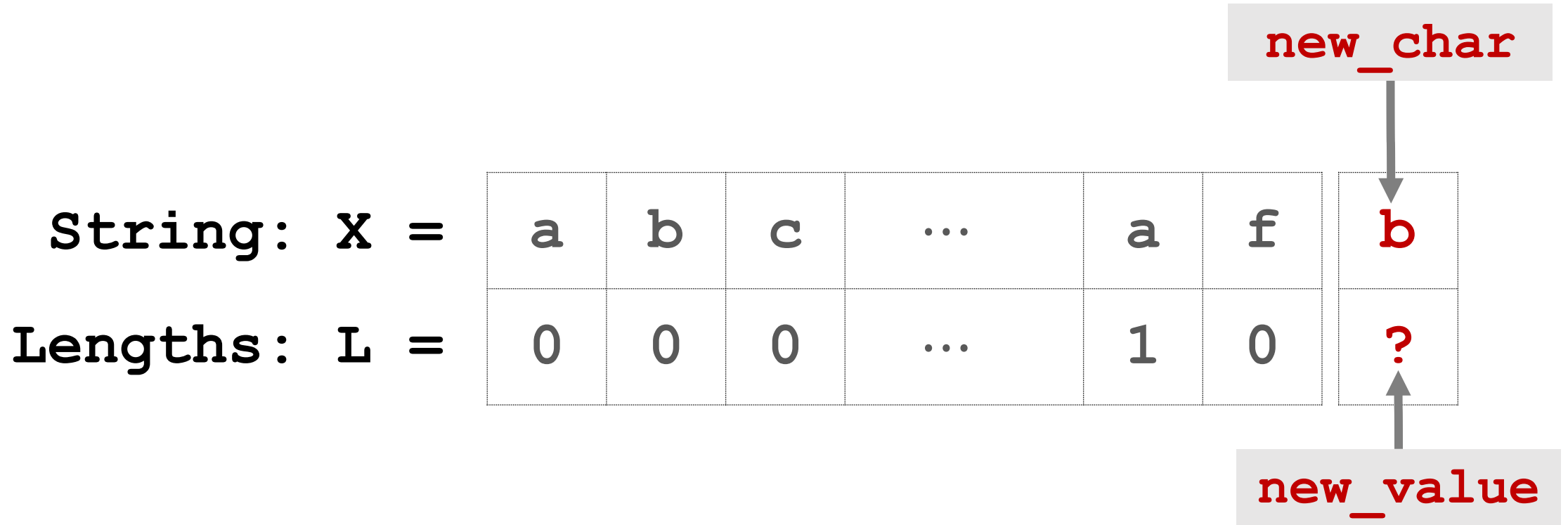
Longest Prefix Suffix Array

String: X =

a	b	c	...	a	f
0	0	0	...	1	0

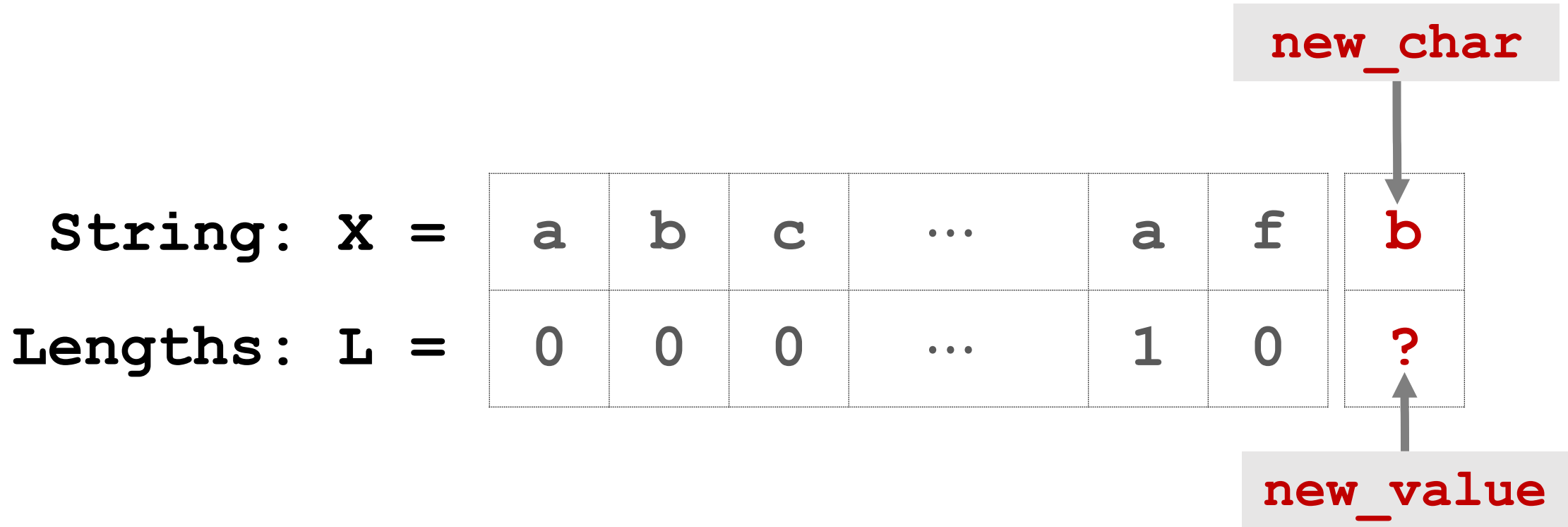
Lengths: L =

Longest Prefix Suffix Array



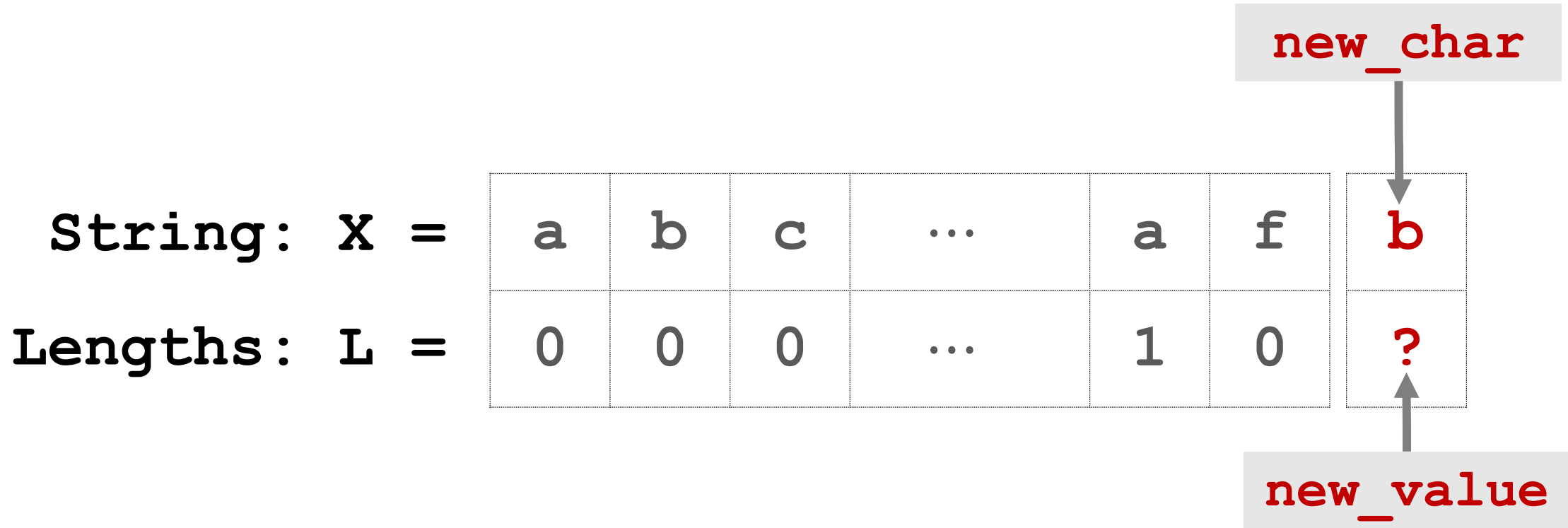
Efficient Algorithm

Function: **new_value** = f(X, L, **new_char**)



Efficient Algorithm

Function: **new_value** = $f(X, L, \text{new_char})$



Time complexity: $O(1)$ time (amortized) for running the function once.

Efficient Algorithm

Function: `new_value = f(X, L, new_char)`

Step 1: Decide the 3 cases

- Let `j = L[end]`.

Efficient Algorithm

Function: `new_value = f(X, L, new_char)`

Step 1: Decide the 3 cases

- Let `j = L[end]`.
- **Match** (i.e., `new_char == X[j]`) or **mismatch**?

Efficient Algorithm

Function: `new_value = f(X, L, new_char)`

Step 1: Decide the 3 cases

- Let `j = L[end]`.
- Match (i.e., `new_char == X[j]`) or mismatch?
- If Match ==> Case 1;

Efficient Algorithm

Function: `new_value = f(X, L, new_char)`

Step 1: Decide the 3 cases

- Let `j = L[end]`.
- Match (i.e., `new_char == X[j]`) or mismatch?
- If Match ==> Case 1;
- If Mismatch ==>

<code>i = 0</code>	==>	Case 2;
<code>j ≠ 0</code>	==>	Case 3.

Efficient Algorithm

Function: `value = f(X, L, new_char)`

Step 2: Different solutions to the 3 cases

- ➡ • Case 1 ==> Return `j + 1`.
- ➡ • Case 2 ==> Return `0`.
- ➡ • Case 3 ==> Return `f(X', L', new_char)`.

$X' = X[0:j]$ and $L' = L[0:j]$

Questions

Build Partial Matching Table

Question 1:

X =	a	b	a	a	b	b	a	b	a	a	b
L =	0	0	1	?	?	?	1	2	3	4	?

Question 2:

X =	a	a	a	a	b	...	a	a	a	a	a
L =	0	1	2	3	?	...	1	2	3	4	?

Build Partial Matching Table

Question 3:

X =	a	b	a	b	c	a	b	a	b	a	b
L =	0	0	1	2	?	?	2	3	4	?	?

Question 4:

X =	b	a	b	a	a	...	a	b	a	b	a	c	a
L =	0	0	1	2	?	...	2	3	?	?	?	0	?

Thank You!

Solution to Question 1

Solution to Question 1

X =	a	b	a	a	b	b	a	b	a	a	b
L =	0	0	1	?	?	?	1	2	3	4	?

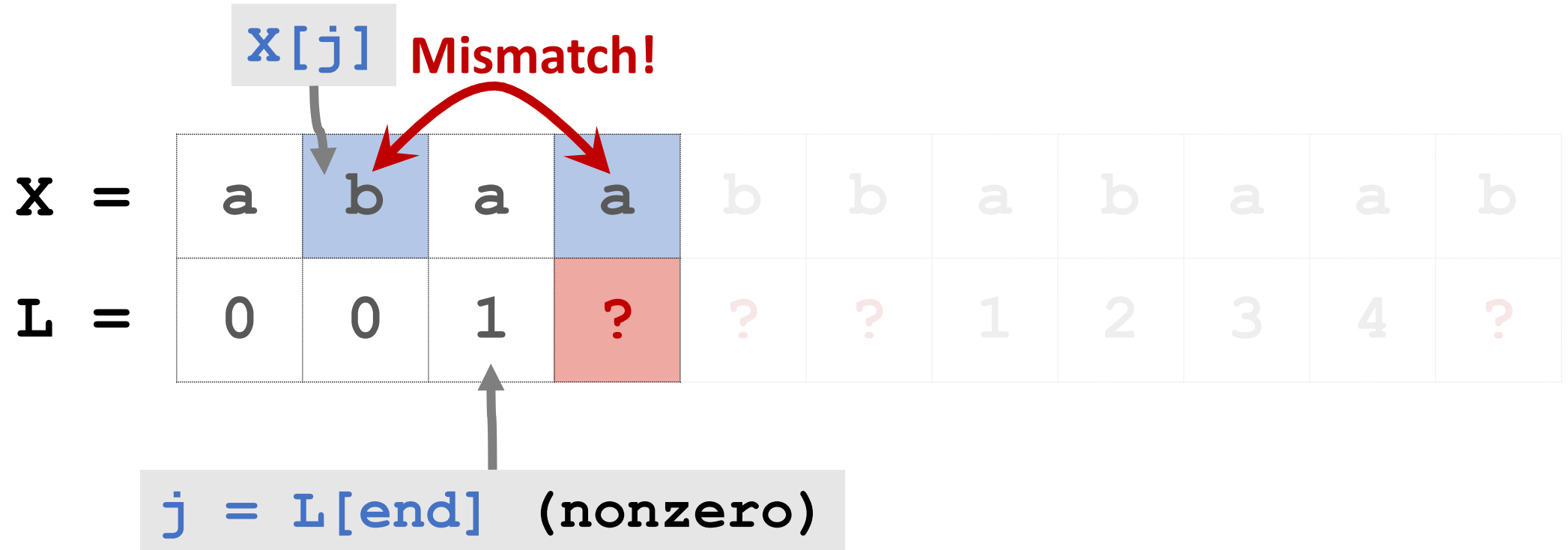
Solution to Question 1(i)

X =	a	b	a	a	b	b	a	b	a	a	b
L =	0	0	1	?	?	?	1	2	3	4	?

`j = L[end] (nonzero)`



Solution to Question 1(i)



- This is **Case 3**.
- Reduce the problem to a smaller problem.

Solution to Question 1(i)

X =	a	b	a	a	b	b	a	b	a	a	b
L =	0	0	1	?	?	?	1	2	3	4	?

`j = L[end] (nonzero)`

X =	a	a
L =	0	?

Solution to Question 1(i)

X =	a	b	a	a	b	b	a	b	a	a	b
L =	0	0	1	?	?	?	1	2	3	4	?

X =	a	a
L =	0	?

Equal



Solution to Question 1(i)

X =	a	b	a	a	b	b	a	b	a	a	b
L =	0	0	1	?	?	?	1	2	3	4	?

X =	a	a
L =	0	1

Equal



Solution to Question 1(i)

X =	a	b	a	a	b	b	a	b	a	a	b
L =	0	0	1	1	?	?	1	2	3	4	?

X =	a	a
L =	0	1

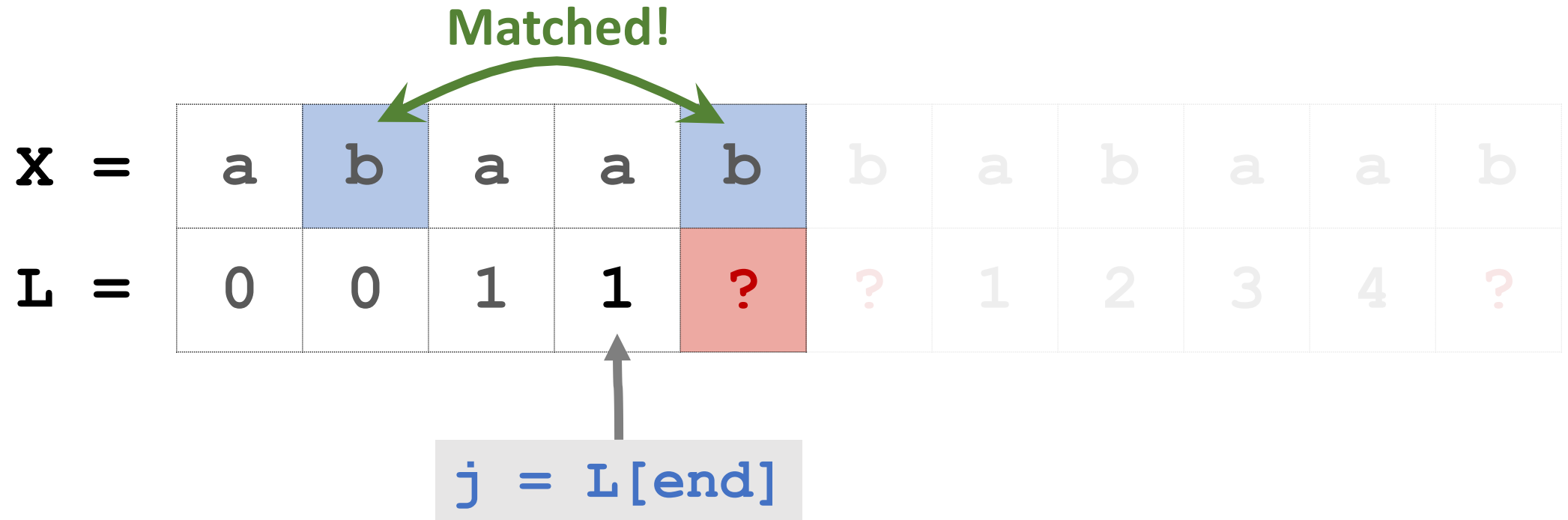
Equal



Solution to Question 1(ii)

X =	a	b	a	a	b	b	a	b	a	a	b
L =	0	0	1	1	?	?	1	2	3	4	?


Solution to Question 1(ii)



- **Case 1:** the new char is equal to $X[j]$.
- Then the new value is $j+1$.

Solution to Question 1(ii)

X =	a	b	a	a	b	b	a	b	a	a	b
L =	0	0	1	1	2	?	1	2	3	4	?

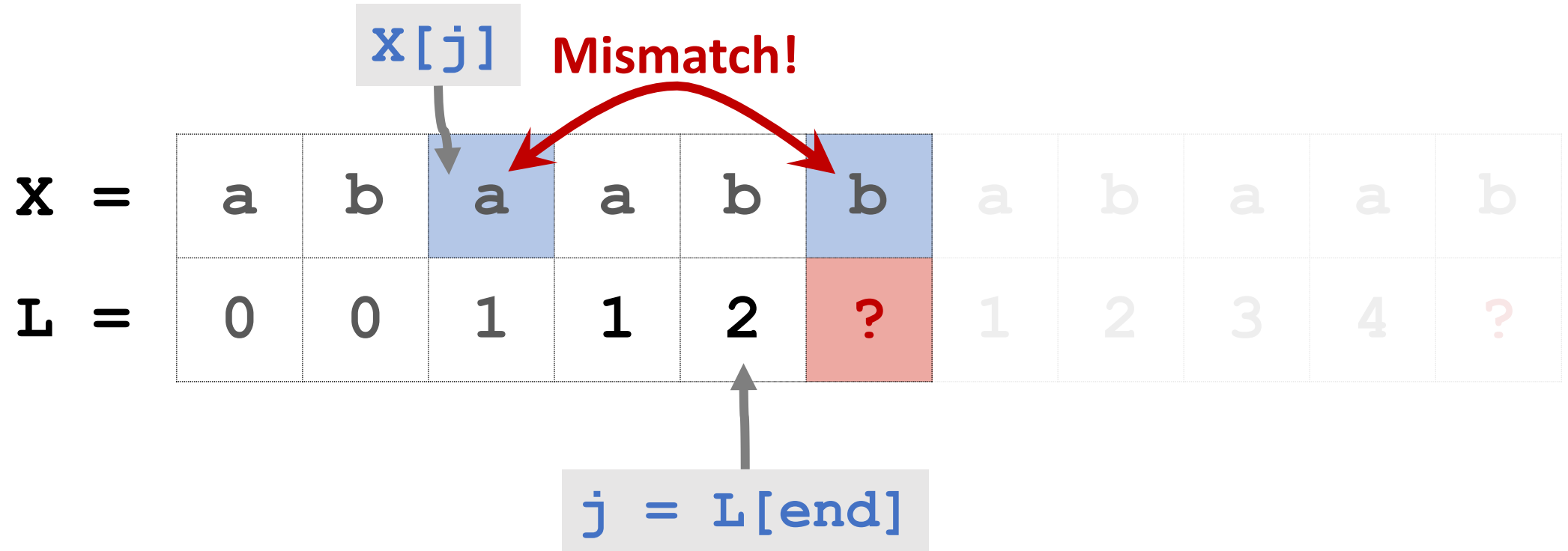

j = L[end]

- **Case 1:** the new char is equal to **X[j]**.
- Then the new value is **j+1**.

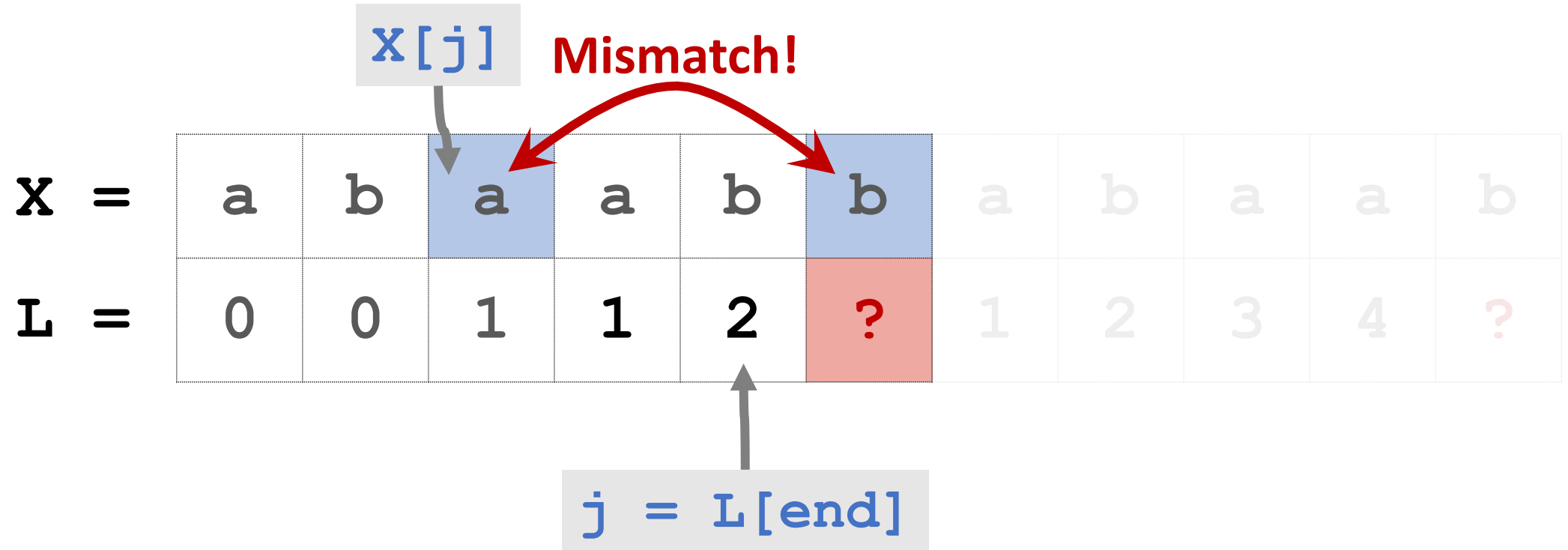
Solution to Question 1(iii)

X =	a	b	a	a	b	b	a	b	a	a	b
L =	0	0	1	1	2	?	1	2	3	4	?

Solution to Question 1(iii)



Solution to Question 1(iii)



- This is **Case 3**.
- Reduce the problem to a smaller problem.

Solution to Question 1(iii)

X =	a	b	a	a	b	b	a	b	a	a	b
L =	0	0	1	1	2	?	1	2	3	4	?

$j = L[\text{end}]$

- This is **Case 3**.
- Reduce the problem to a smaller problem.

Solution to Question 1(iii)

X =	a	b	a	a	b	b	a	b	a	a	b
L =	0	0	1	1	2	?	1	2	3	4	?

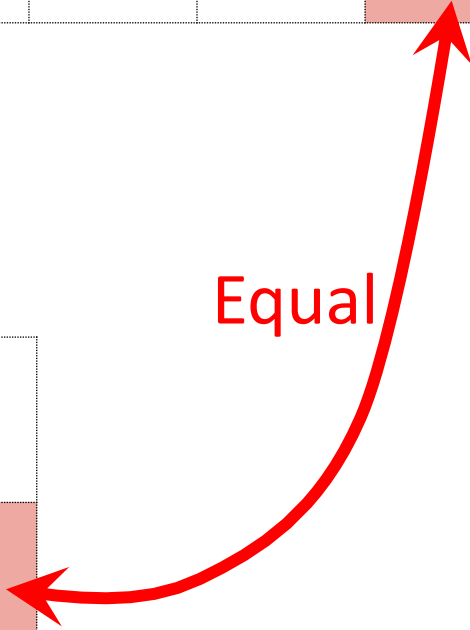
X =	a	b
L =	0	0

Solution to Question 1(iii)

X =	a	b	a	a	b	b	a	b	a	a	b
L =	0	0	1	1	2	?	1	2	3	4	?

X =	a	b	b
L =	0	0	?

Equal

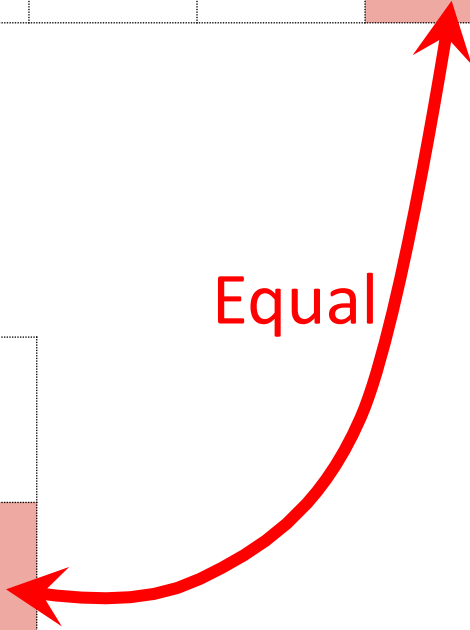


Solution to Question 1(iii)

X =	a	b	a	a	b	b	a	b	a	a	b
L =	0	0	1	1	2	?	1	2	3	4	?

X =	a	b	b
L =	0	0	0

Equal

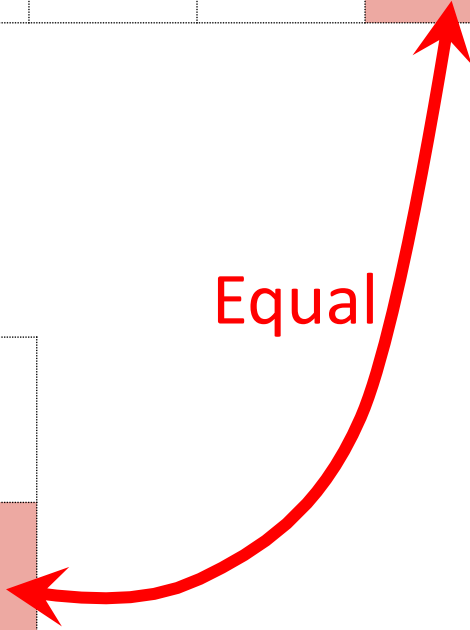


Solution to Question 1(iii)

X =	a	b	a	a	b	b	a	b	a	a	b
L =	0	0	1	1	2	0	1	2	3	4	?

X =	a	b	b
L =	0	0	0

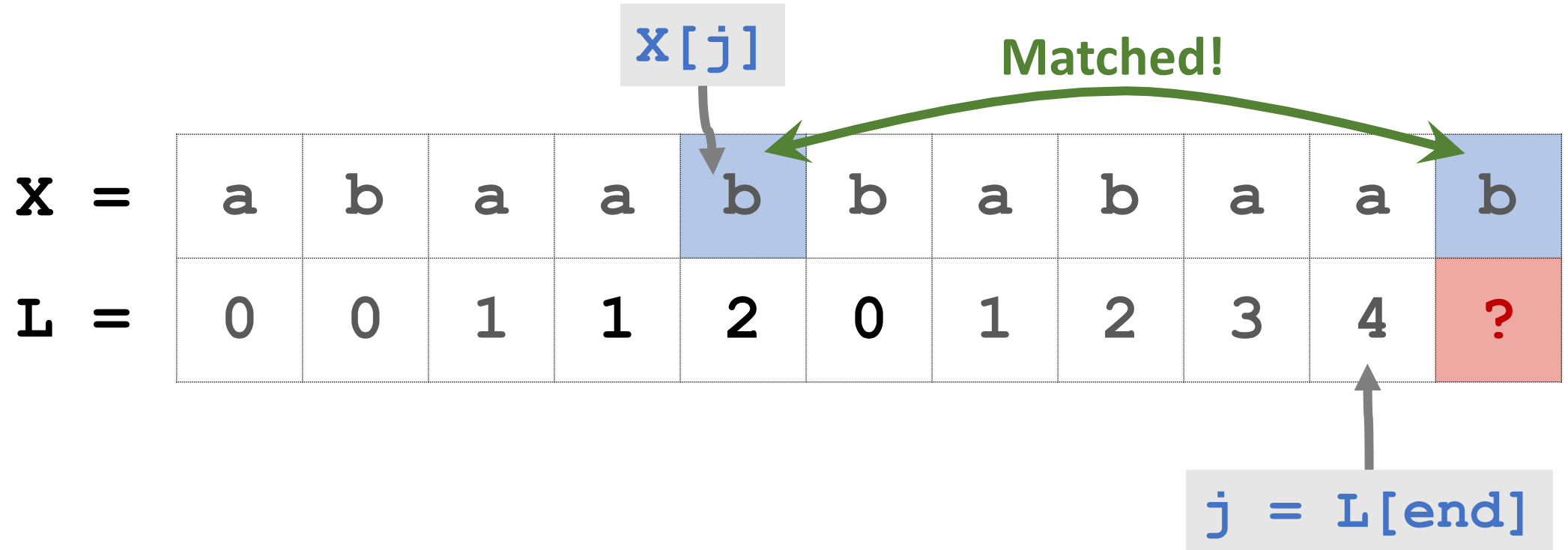
Equal



Solution to Question 1(iv)

X =	a	b	a	a	b	b	a	b	a	a	b
L =	0	0	1	1	2	0	1	2	3	4	?

Solution to Question 1(iv)



- **Case 1:** the new char is equal to $x[j]$.
- Then the new value is $j+1$.

Solution to Question 1(iv)

X =	a	b	a	a	b	b	a	b	a	a	b
L =	0	0	1	1	2	0	1	2	3	4	5

$j = L[\text{end}]$

- **Case 1:** the new char is equal to $X[j]$.
- Then the new value is $j+1$.