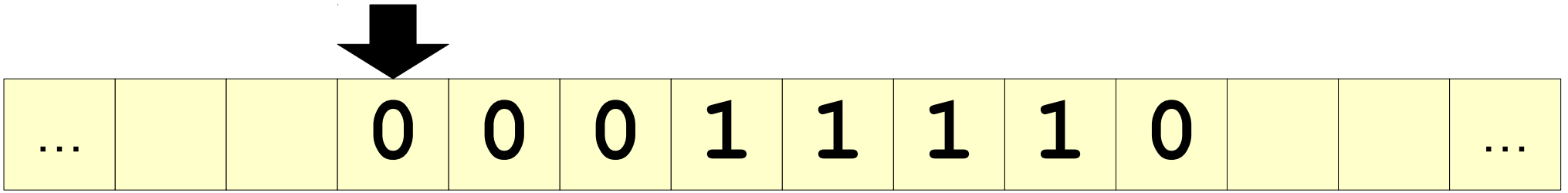
Another TM Design

- We just designed a TM for this language over $\Sigma = \{$**0**, **1**$\}$:

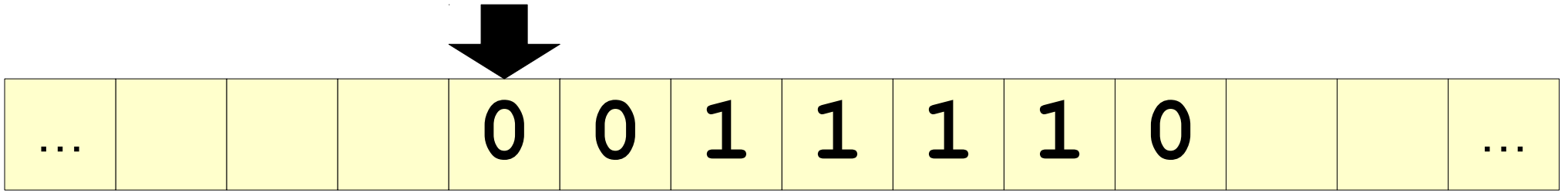
  $L = \{ w \in \Sigma^* \mid w$ has the same number of **0**s and **1**s $\}$

- Let's do a quick review of how it worked.

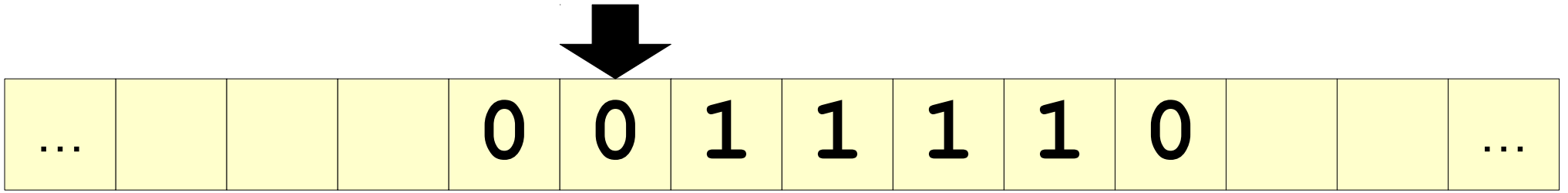
A Leap of Faith



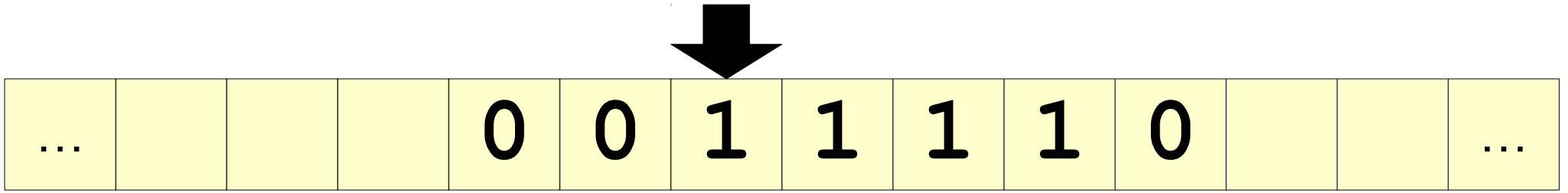
A Leap of Faith



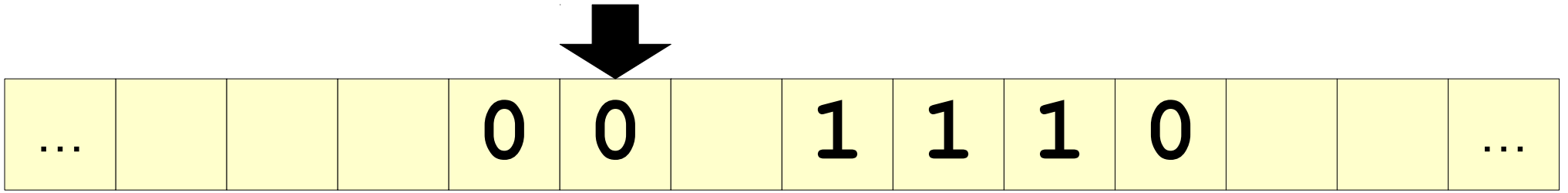
A Leap of Faith



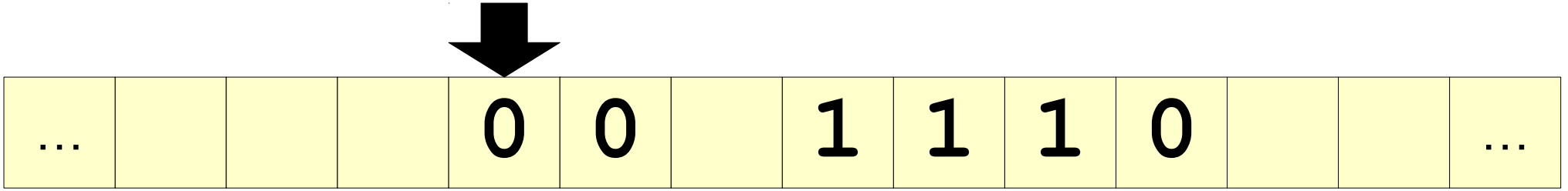
A Leap of Faith



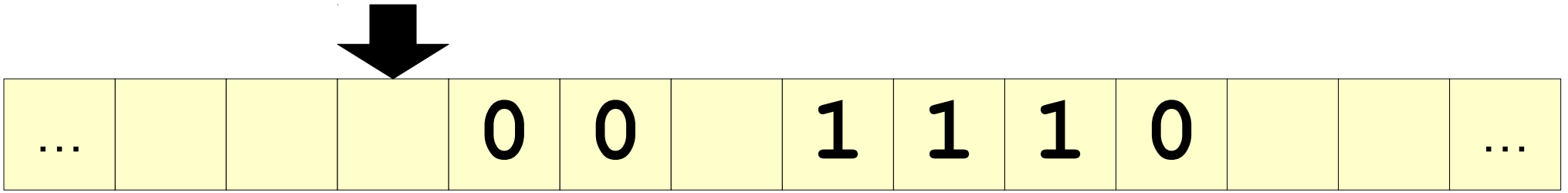
A Leap of Faith



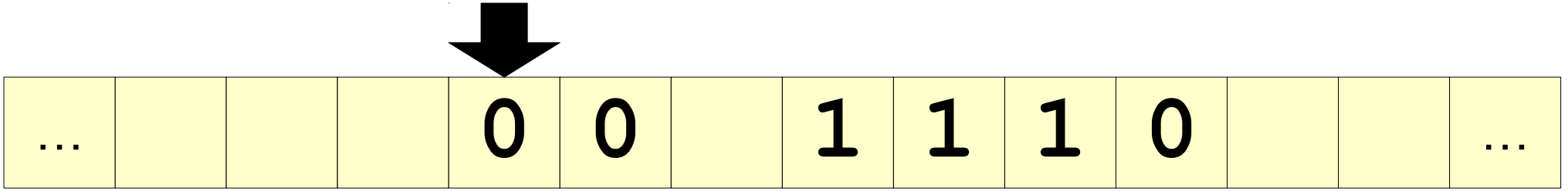
A Leap of Faith



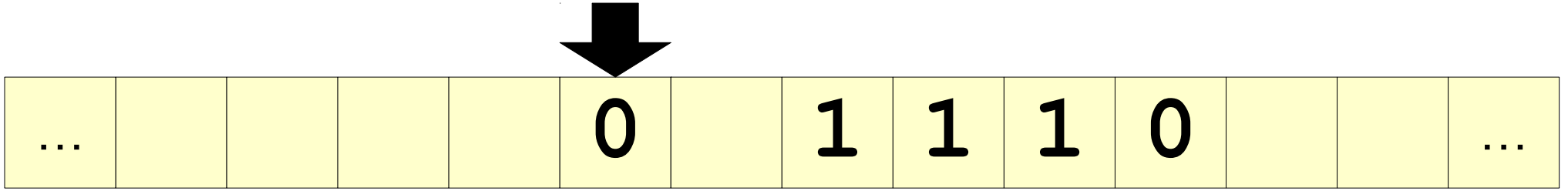
A Leap of Faith



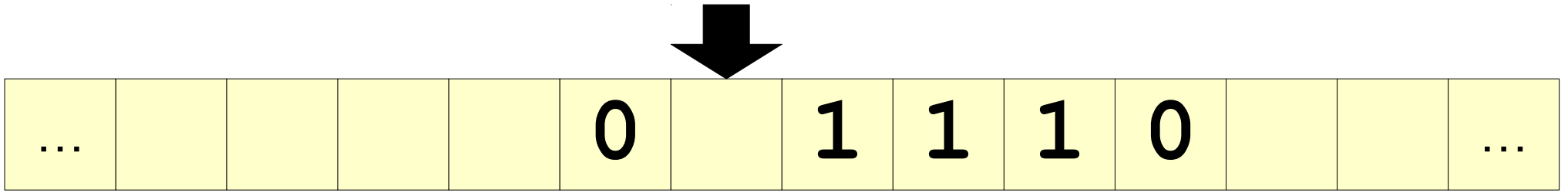
A Leap of Faith



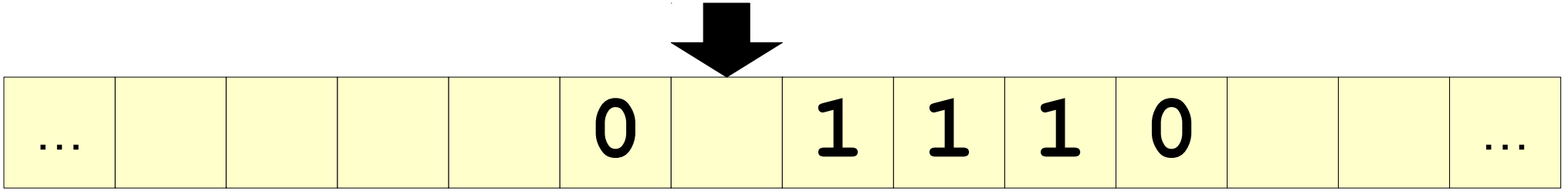
A Leap of Faith



A Leap of Faith

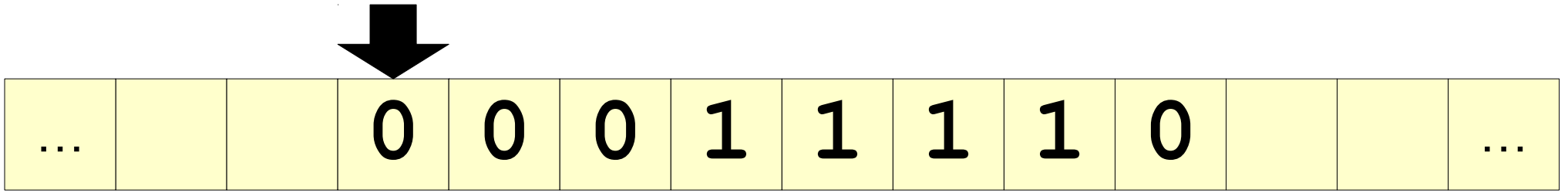


A Leap of Faith

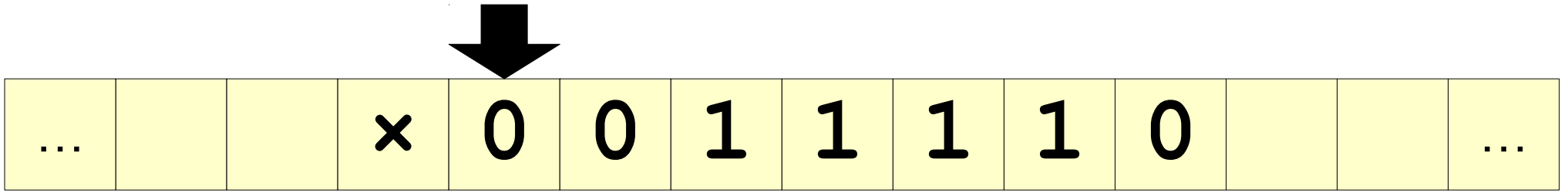


How do we know that this blank isn't one of the infinitely many blanks after our input string?

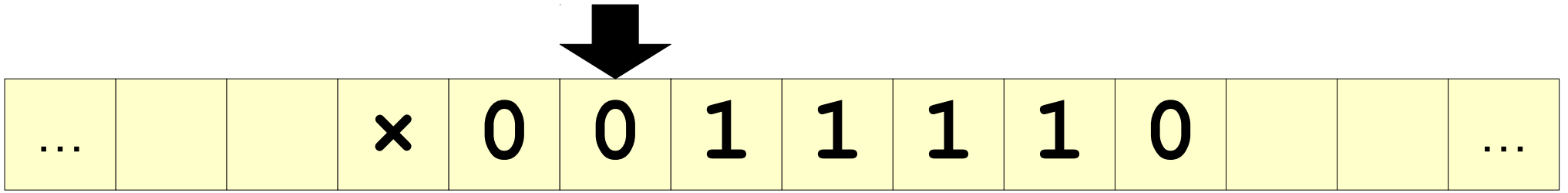
The Solution



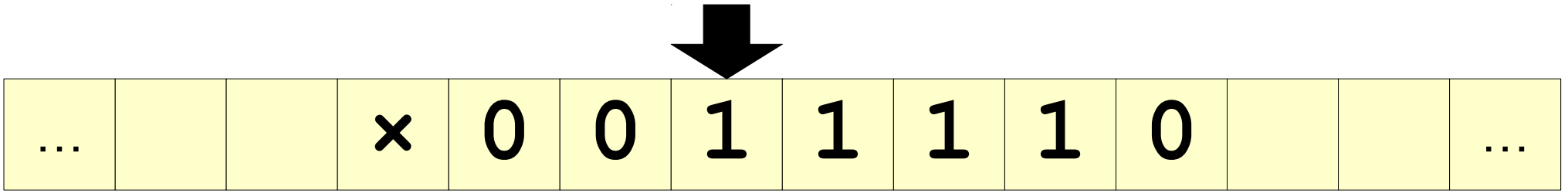
The Solution



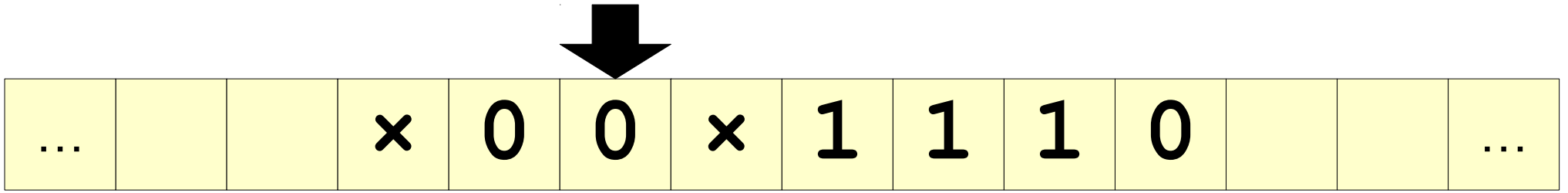
The Solution



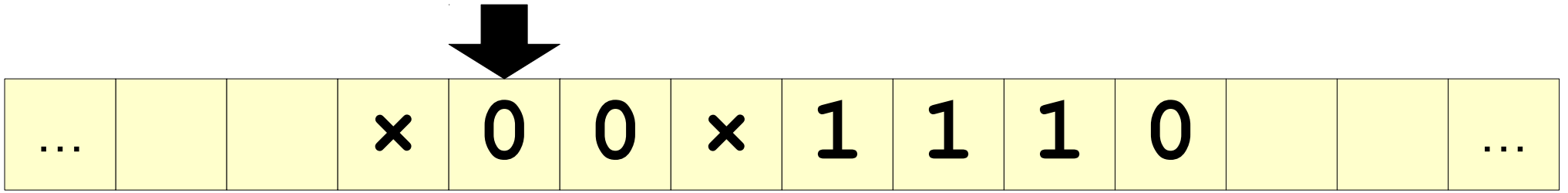
The Solution



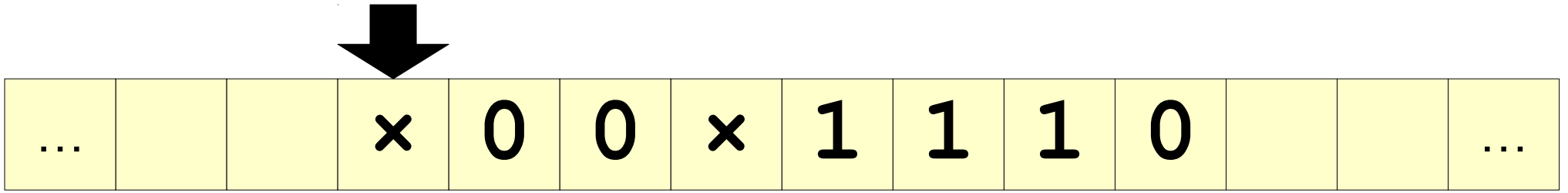
The Solution



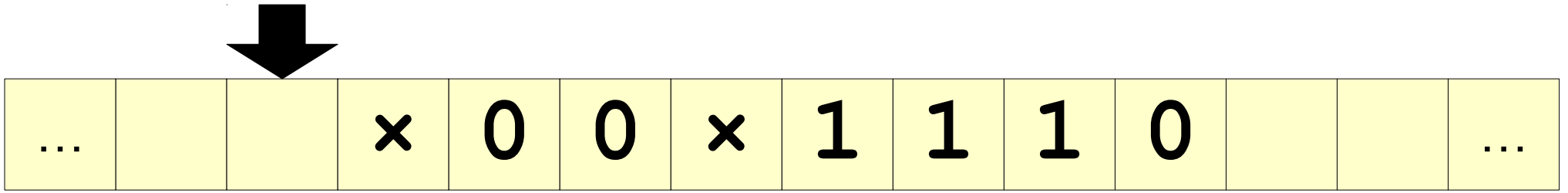
The Solution



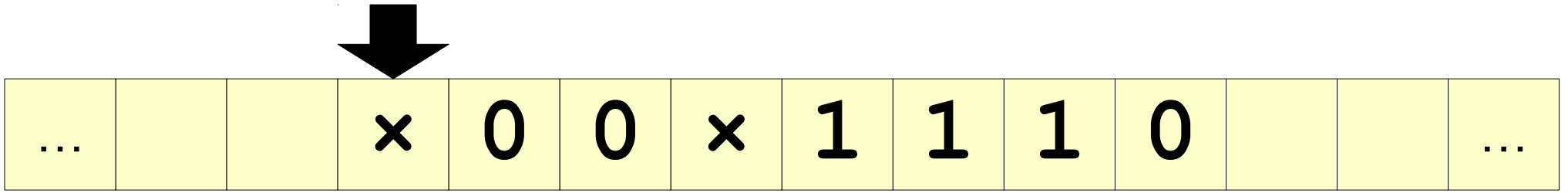
The Solution



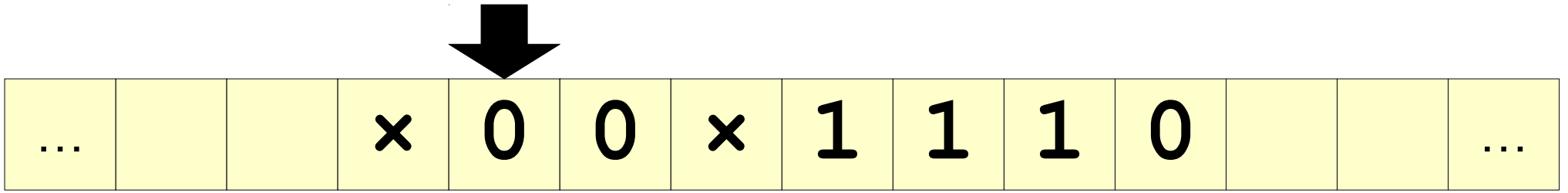
The Solution



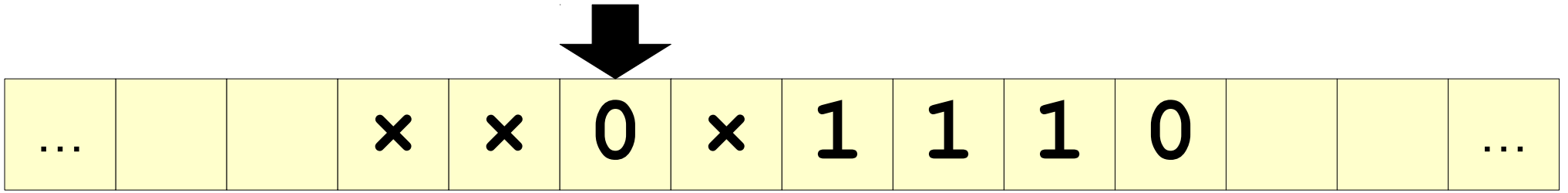
The Solution



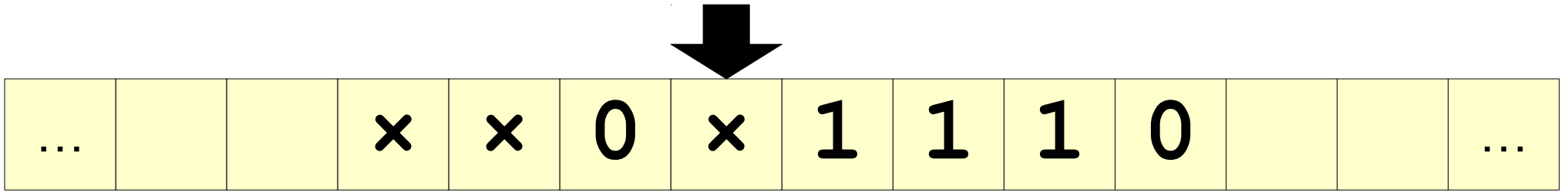
The Solution



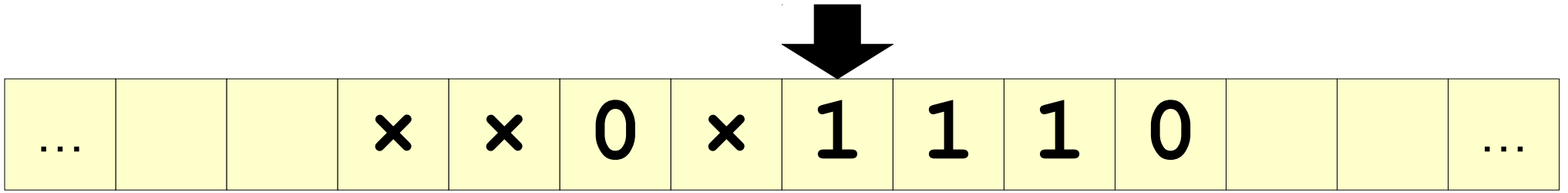
The Solution



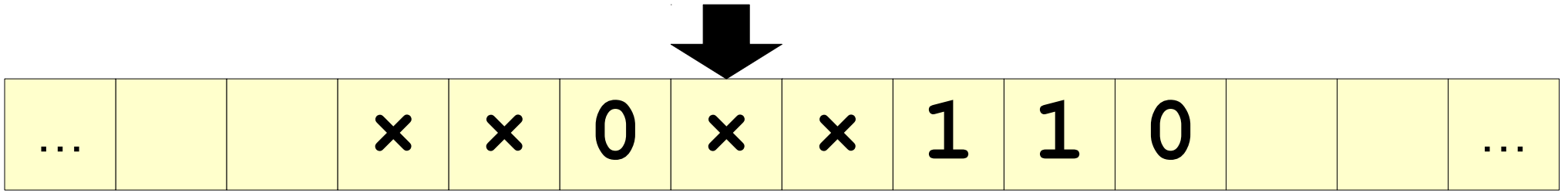
The Solution



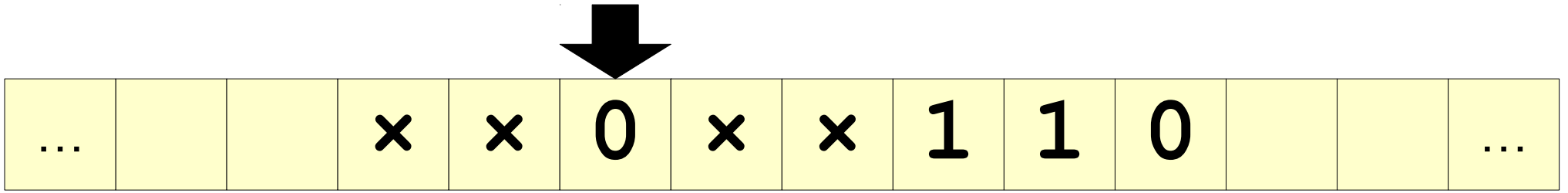
The Solution



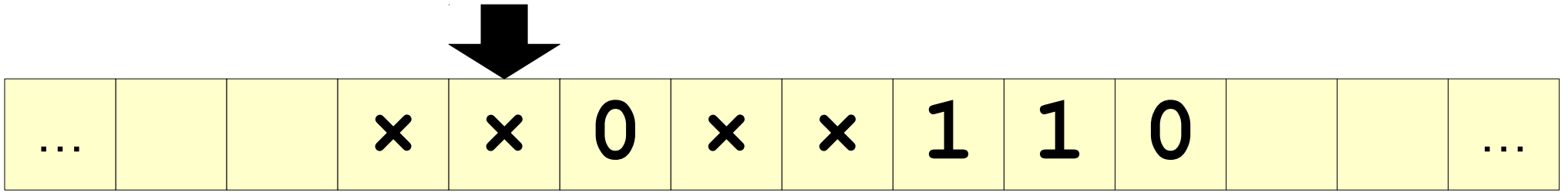
The Solution



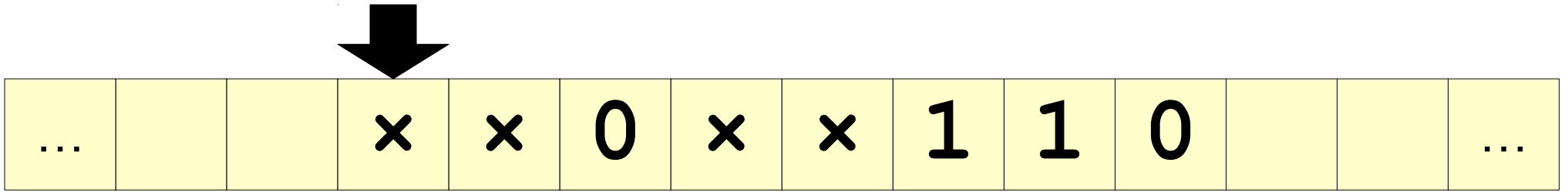
The Solution



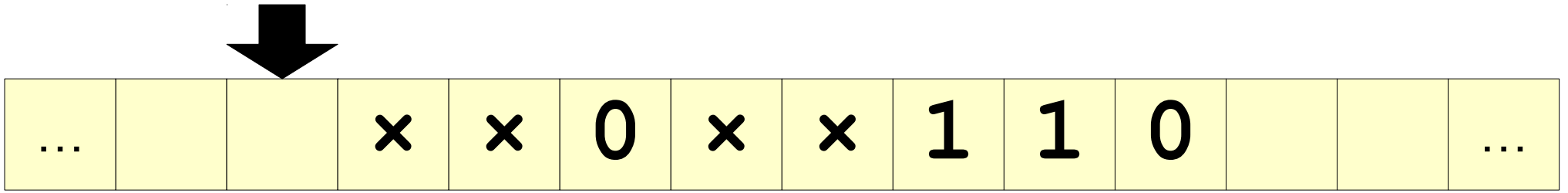
The Solution



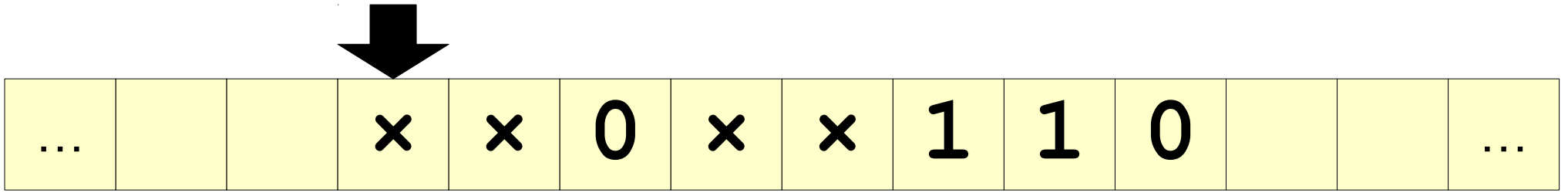
The Solution



The Solution

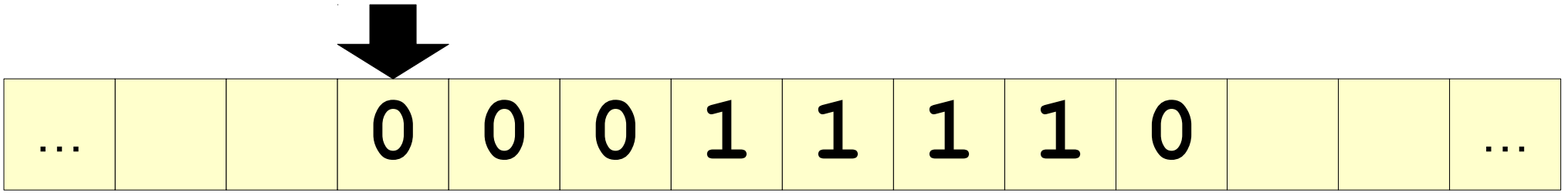


The Solution

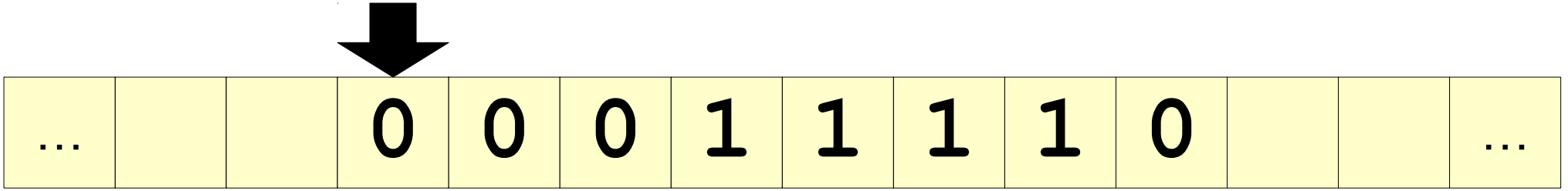


A Different Idea

A Different Strategy

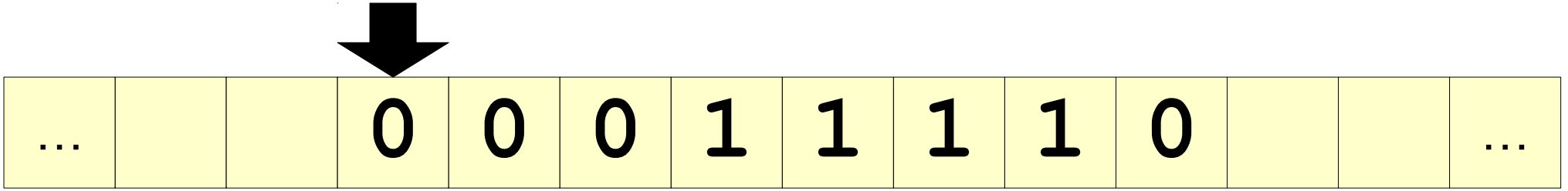


A Different Strategy



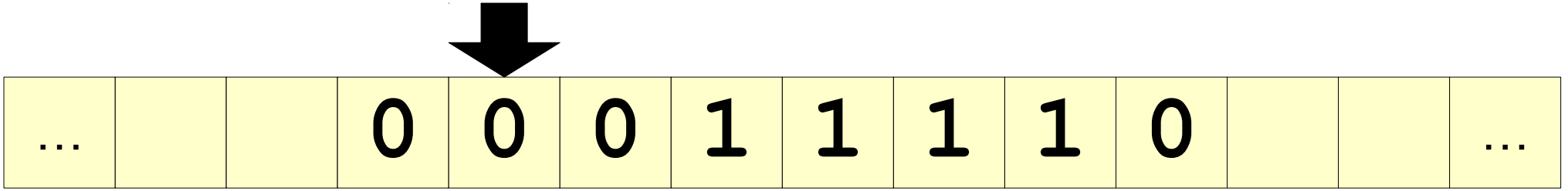
Could we sort the characters of this string?

A Different Strategy



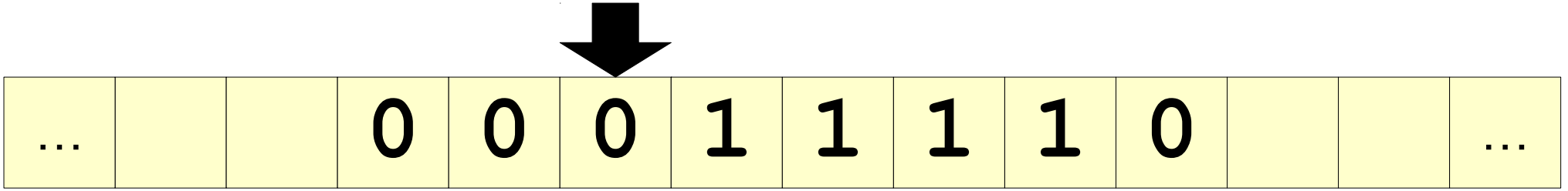
Observation 1: A string of 0s and 1s is sorted if it matches the regex 0^*1^*.

A Different Strategy



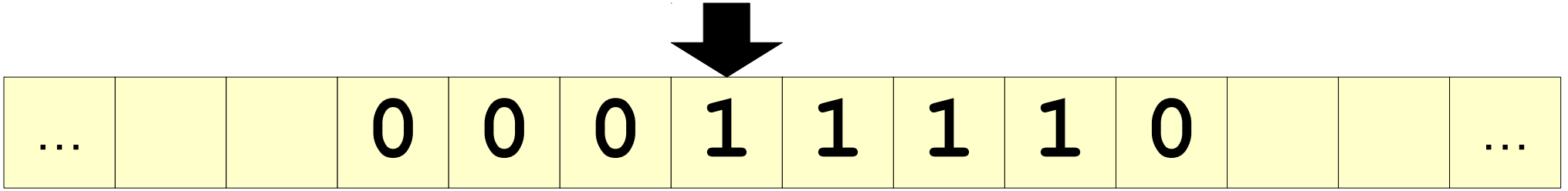
Observation 1: A string of 0s and 1s is sorted if it matches the regex 0^*1^*.

A Different Strategy



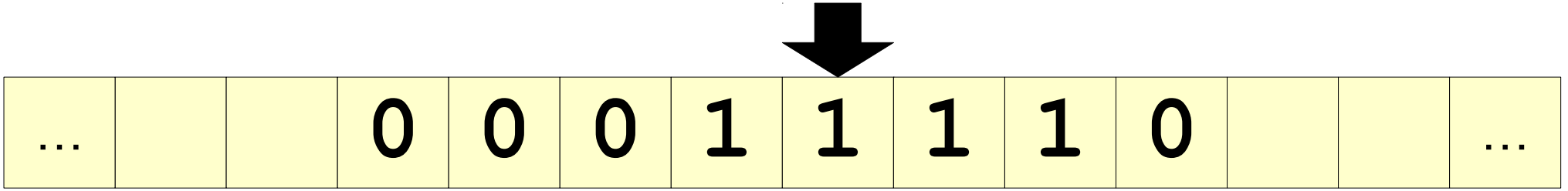
Observation 1: A string of 0s and 1s is sorted if it matches the regex 0^*1^*.

A Different Strategy



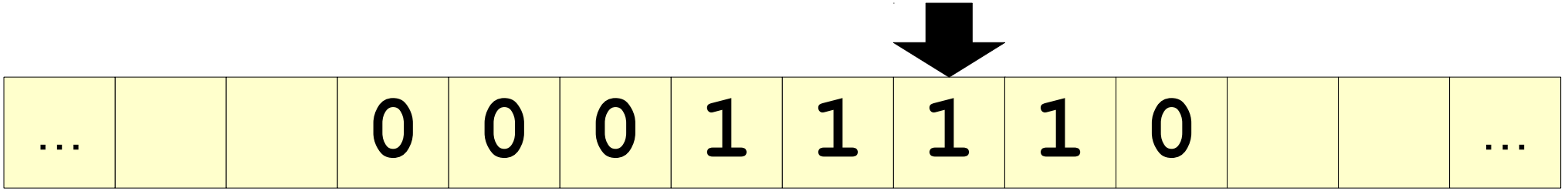
Observation 1: A string of 0s and 1s is sorted if it matches the regex 0^*1^*.

A Different Strategy



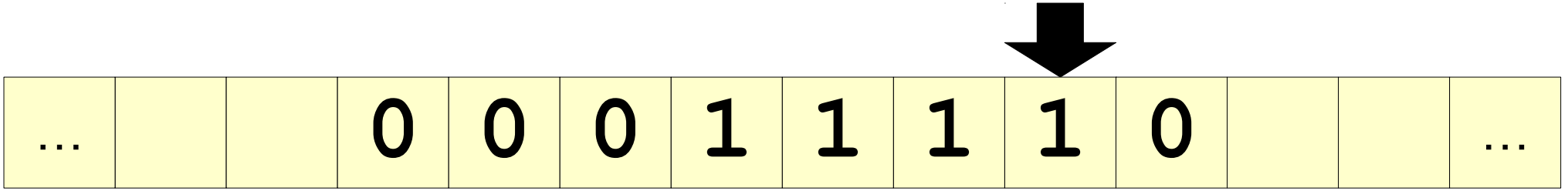
Observation 1: A string of 0s and 1s is sorted if it matches the regex 0^*1^*.

A Different Strategy



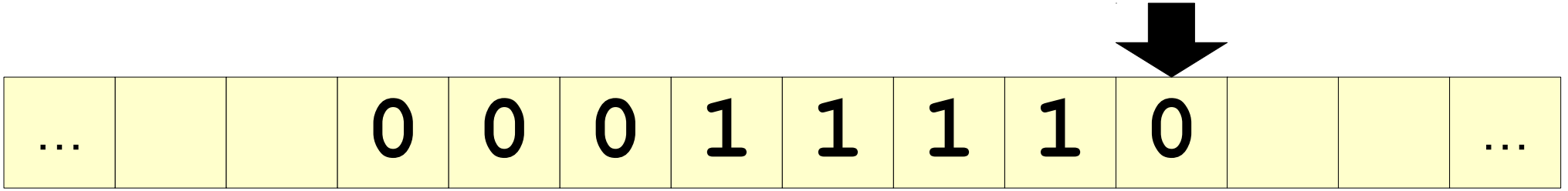
Observation 1: A string of 0s and 1s is sorted if it matches the regex 0^*1^*.

A Different Strategy



Observation 1: A string of 0s and 1s is sorted if it matches the regex 0^*1^*.

A Different Strategy



Observation 1: A string of 0s and 1s is sorted if it matches the regex 0^*1^*.

A Different Strategy

Observation 1: A string of 0s and 1s is sorted if it matches the regex 0^*1^*.

A Different Strategy

Observation 2: A string of 0s and 1s is **not** sorted if it contains 10 as a substring.

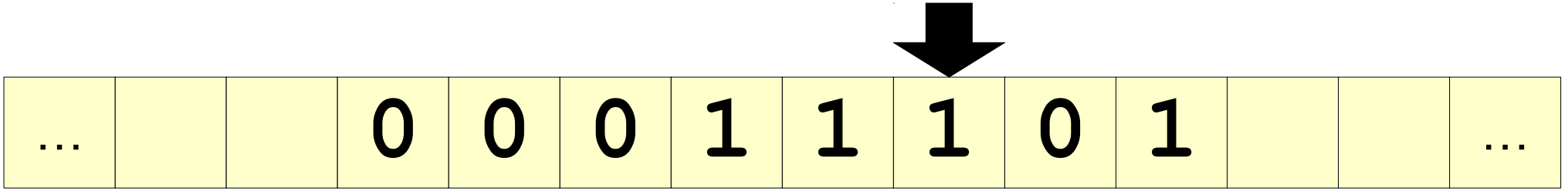
A Different Strategy

Observation 2: A string of 0s and 1s is **not** sorted if it contains 10 as a substring.

A Different Strategy

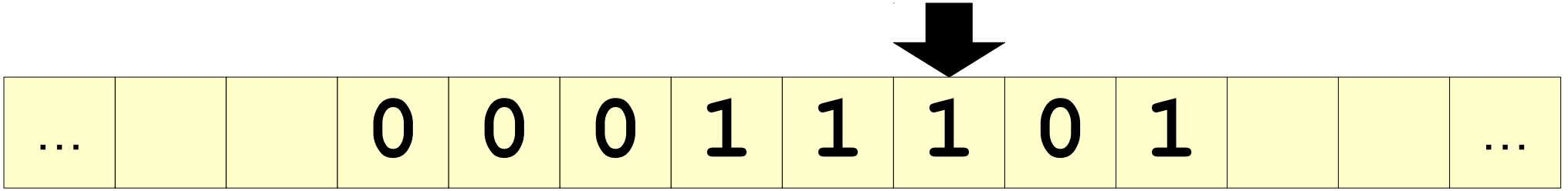
Observation 2: A string of 0s and 1s is **not** sorted if it contains 10 as a substring.

A Different Strategy



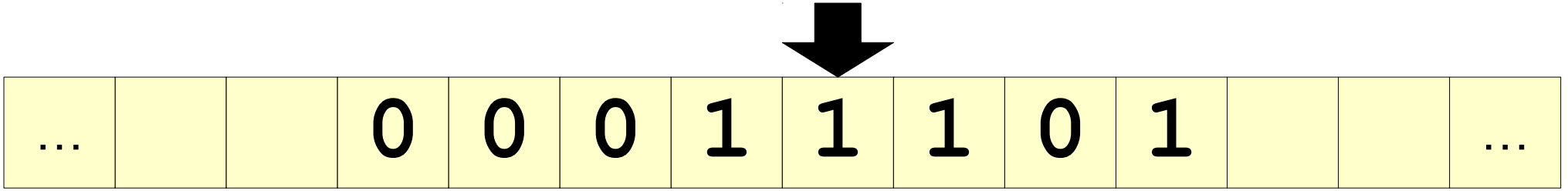
Observation 2: A string of 0s and 1s is **not** sorted if it contains 10 as a substring.

A Different Strategy



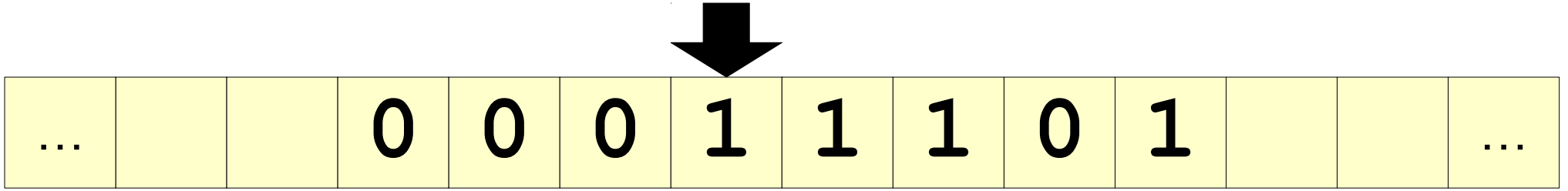
Idea: Repeatedly find a copy of 10 and replace it with 01.

A Different Strategy



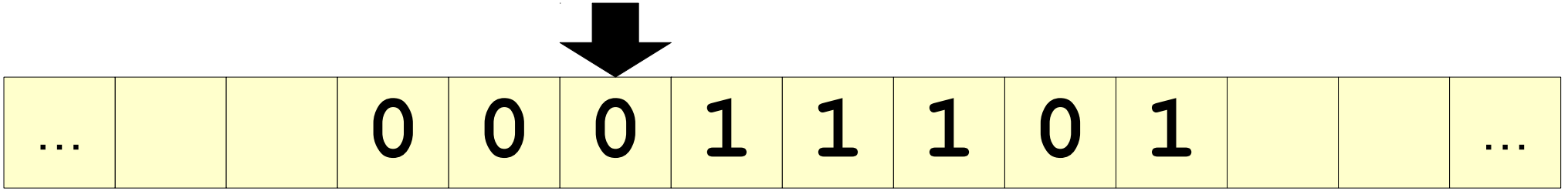
Idea: Repeatedly find a copy of 10 and replace it with 01.

A Different Strategy



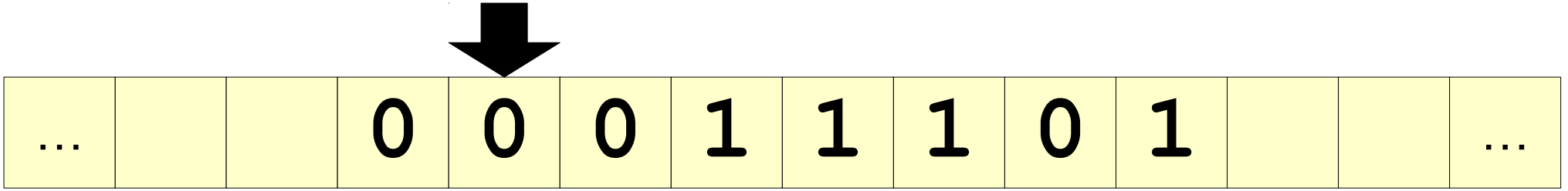
Idea: Repeatedly find a copy of 10 and replace it with 01.

A Different Strategy



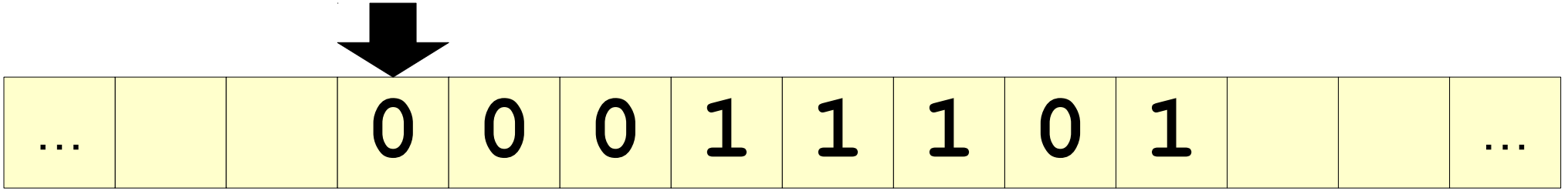
Idea: Repeatedly find a copy of 10 and replace it with 01.

A Different Strategy



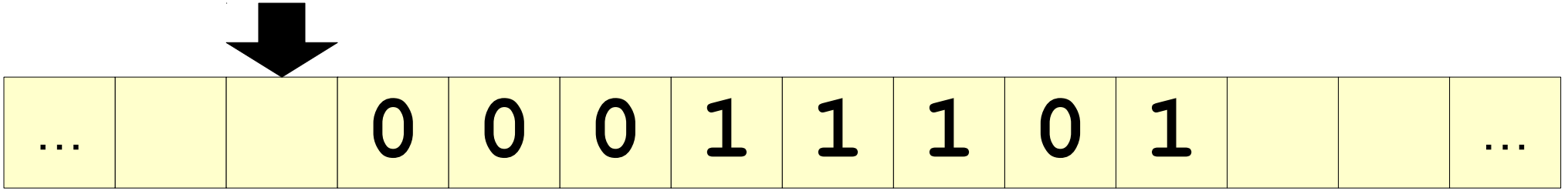
Idea: Repeatedly find a copy of 10 and replace it with 01.

A Different Strategy



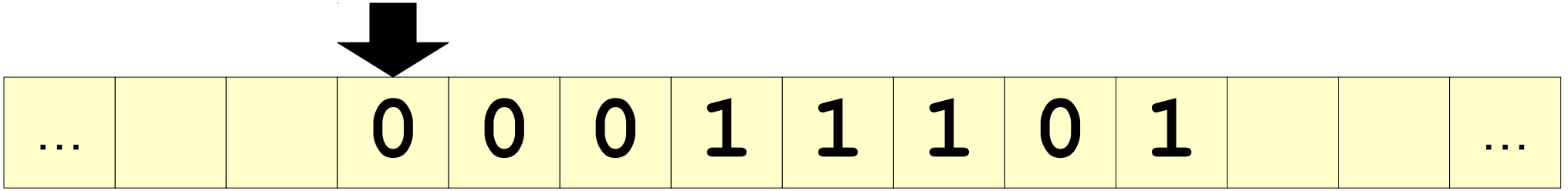
Idea: Repeatedly find a copy of 10 and replace it with 01.

A Different Strategy



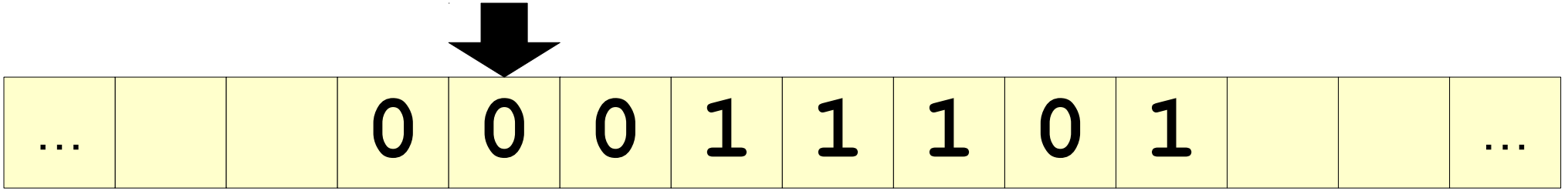
Idea: Repeatedly find a copy of 10 and replace it with 01.

A Different Strategy



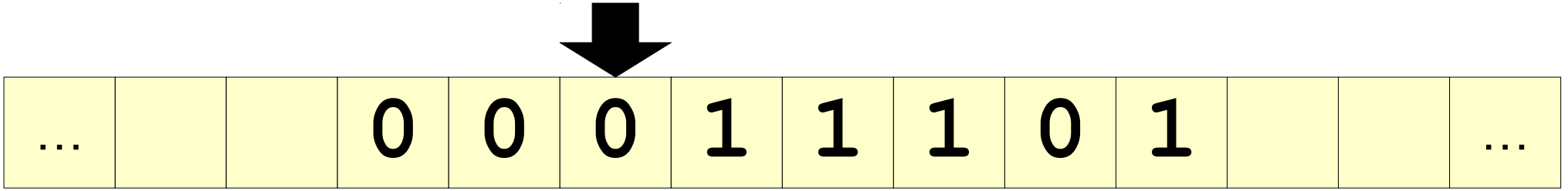
Idea: Repeatedly find a copy of 10 and replace it with 01.

A Different Strategy



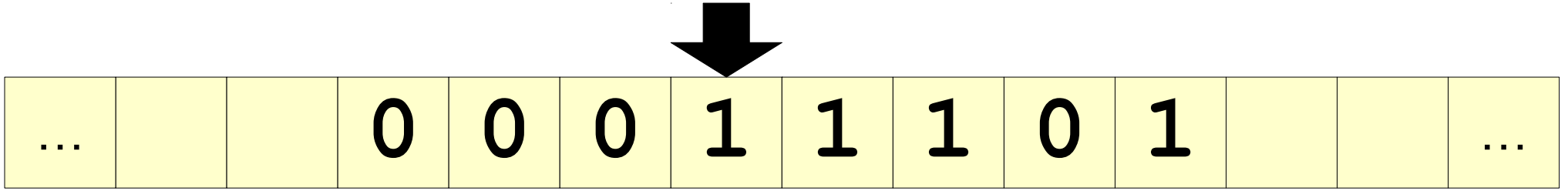
Idea: Repeatedly find a copy of 10 and replace it with 01.

A Different Strategy



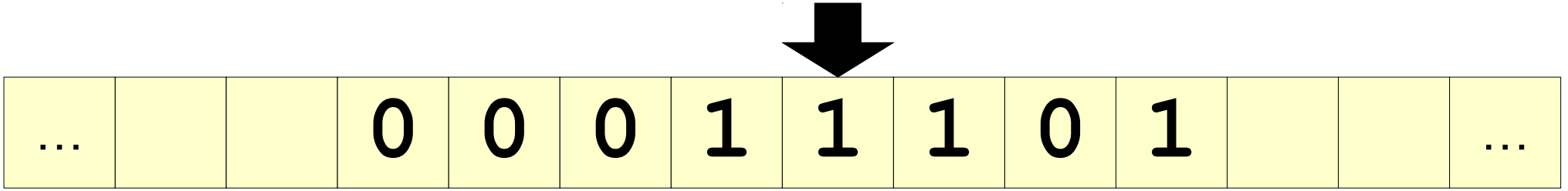
Idea: Repeatedly find a copy of 10 and replace it with 01.

A Different Strategy



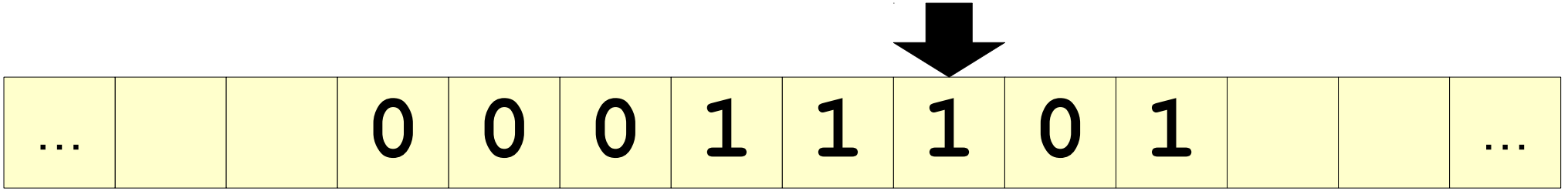
Idea: Repeatedly find a copy of 10 and replace it with 01.

A Different Strategy



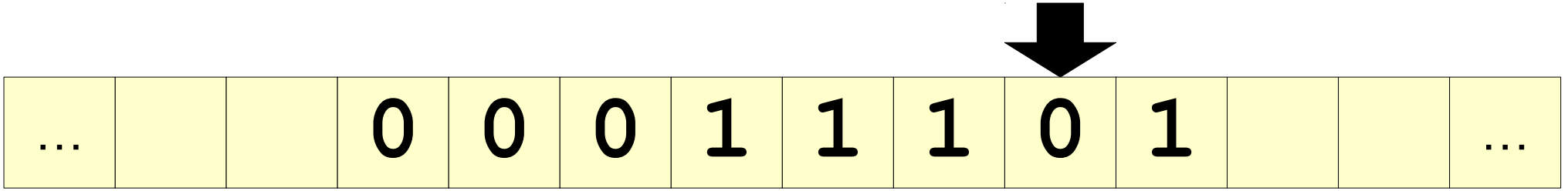
Idea: Repeatedly find a copy of 10 and replace it with 01.

A Different Strategy



Idea: Repeatedly find a copy of 10 and replace it with 01.

A Different Strategy



Idea: Repeatedly find a copy of 10 and replace it with 01.

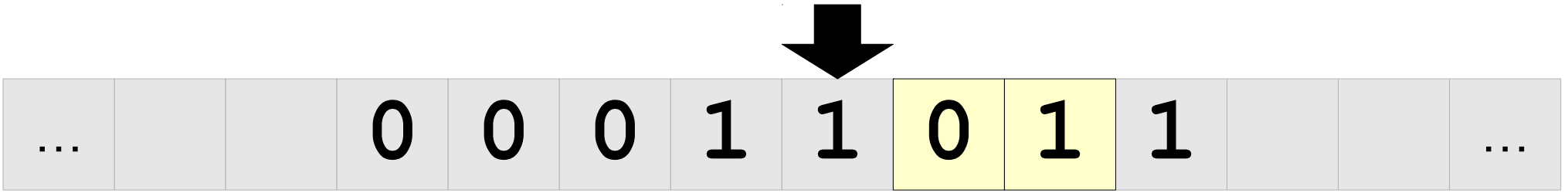
A Different Strategy

Idea: Repeatedly find a copy of 10 and replace it with 01.

A Different Strategy

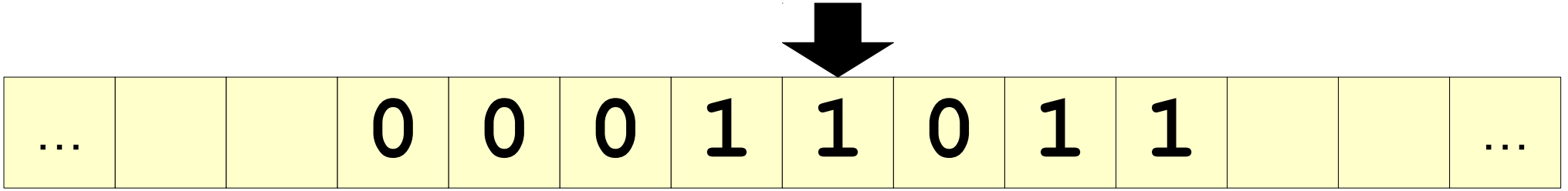
Idea: Repeatedly find a copy of 10 and replace it with 01.

A Different Strategy



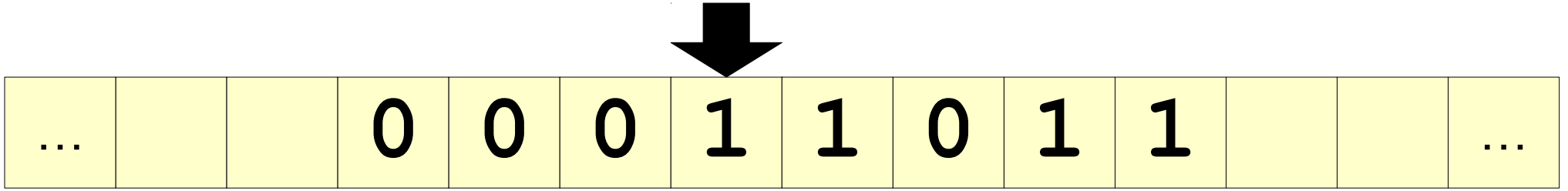
Idea: Repeatedly find a copy of 10 and replace it with 01.

A Different Strategy



Idea: Repeatedly find a copy of 10 and replace it with 01.

A Different Strategy



Idea: Repeatedly find a copy of 10 and replace it with 01.

Let's Build It!