

# Exact Pattern Matching on Strings: Boyer-Moore

Chapter 2 of Dan Gusfield: *Algorithms on strings, trees, and sequences*

Giulia Bernardini  
[giulia.bernardini@units.it](mailto:giulia.bernardini@units.it)

Algorithmic Design  
a.y. 2022/2023

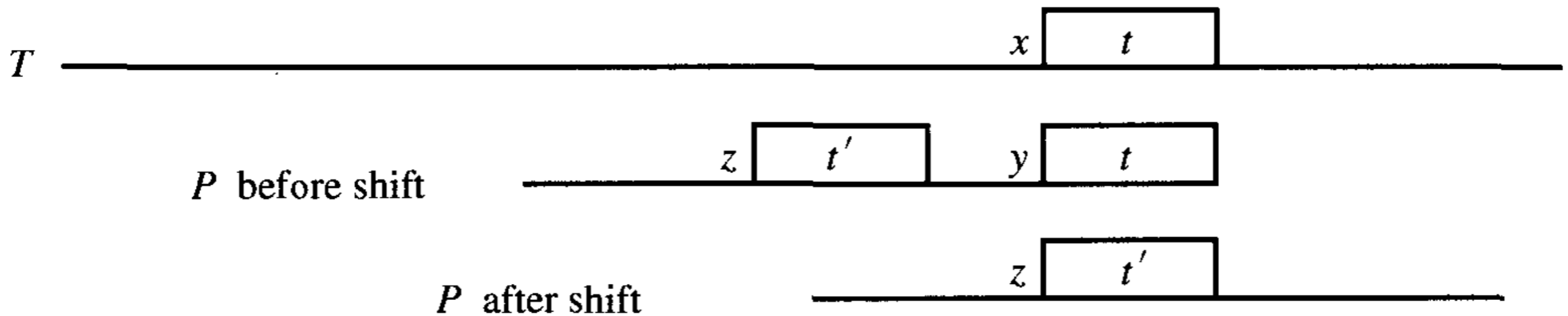
# The Boyer-Moore algorithm

Boyer-Moore is the practical method of choice for exact matching: it typically examines less than  $|P|+|T|$  characters, so it has an **expected sublinear running time** and a **linear worst-case time**.

It uses three clever ideas:

1. The characters of the pattern are scanned from right to left
2. It uses the bad character shift rule
3. It uses the good suffix shift rule

# The good suffix rule



Preprocessing  $P$  for the good suffix rule requires  $O(|P|)$  time.

# Preprocessing P for the bad character rule

Let  $\Sigma$  be the alphabet of T (note that we can assume  $|\Sigma| \leq |T|$ ).

- Initialise an array of zeroes R of length  $|\Sigma| \leq |T|$
- For each  $i=1, \dots, |P|$ ,  $R[P[i]] \leftarrow i$
- At the end,  $R[x]$  contains the rightmost position of P where character x occurs; or 0 if x does not occur in P.
- This preprocessing requires  $\Theta(|\Sigma| + |P|)$  time

# Comparison between Knuth-Morris-Pratt and Boyer-Moore-Galil

- Both use a **sliding window** of the same length as the pattern. The window delimits a factor of the text to be examined, and slides along the text from left to right. Not all existing pattern matching algorithms use this framework.

## KMP

- $\Theta(|P|+|T|)$  worst-case running time
- P is scanned from left to right

## BMG

- $O(|P|+|T|)$  worst-case running time; sublinear expected time
- P is scanned from right to left