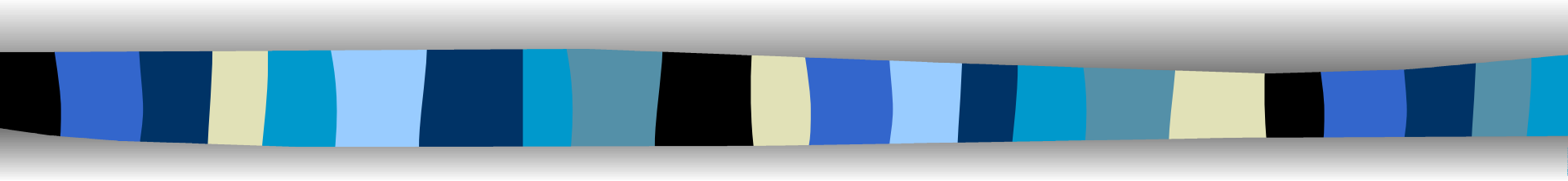
String Matching



including Horspool's algorithm

by Saadiq Moolla

Introduction

- Given the problem:

  find the pattern "*not*" in a substring of the text "*nobody noticed him*"

- How could we do this?

Brute Force Solution

1. Align the pattern against the first letters of the text.
2. Compare the characters from left to right.
3. If a mismatch occurs, shift the pattern to the right and compare the characters.
4. If no mismatch occurs, a substring is found.
5. The algorithm can stop or continue searching for more substrings.

NOBODY NOTICED HIM NOT

Evaluation

- Worst case solution is not very efficient.
- O (nm)

- Typical word search is much more efficient because shifting occurs must sooner.
- Search time is linear…

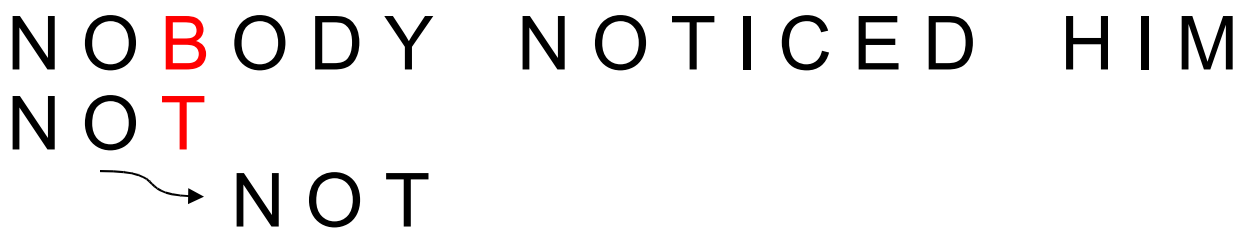
- … but there are more efficient algorithms.

Horspool's Algorithm

- Starting at the beginning of the text, compare the last letter of the pattern with the corresponding letter in the text.
- Continue matching characters, but if a mismatch is found, we want to make as large a shift as possible without missing any possibilities.
- Horspool's Algorithm determines the size of this shift. If the character *c* in the text caused the mismatch…

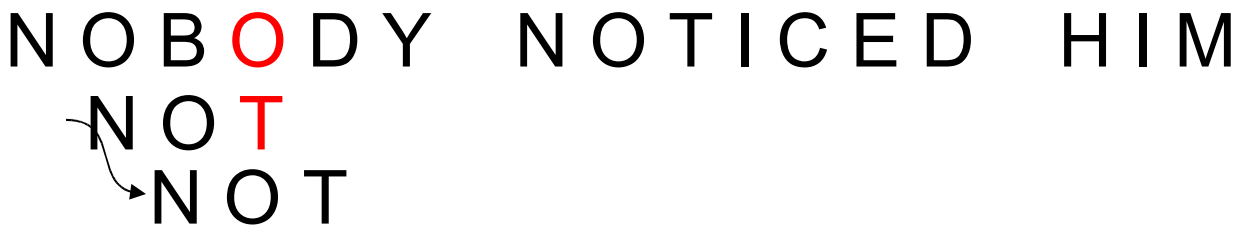
Case 1

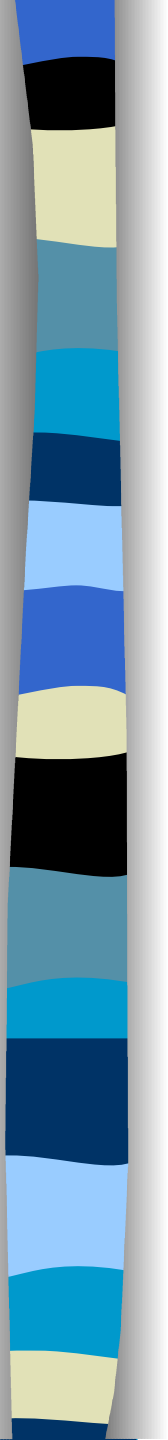
If there is no *c* in the pattern, then it can be safely shifted its entire length:



Case 2

If there are occurances of *c* in the pattern, but it is not the last one there, then the shift should align the rightmost occurance of *c* in the pattern with the *c* in the text:





Case 3

If *c* happens to be the last character of the sequence, but there are no other occurences of *c*, the pattern can be shifted by its entire length.

Case 4

If *c* happens to be the last character of the sequence and there are other occurences of *c*, the pattern must be shifted so that the rightmost occurance of *c* (excluding the last one) is aligned with the text's *c*.

The Shift Table

- For a pattern of length *m*, the amount the pattern can be shifted if *c* in the text mismatches is:
  - *m*, if *c* is not amongst the first *m – 1* characters of the pattern
  - the distance from the rightmost *c* amongst the first *m – 1* characters of the pattern to its last character

Computing the Shift Table

- Set all entries to the pattern's length, *m*
- Scan the pattern left to right overwriting the entry for the character *j* with the value *m – 1 – j*

The *not* Shift Table

A	B	C	D	N	O	T	Z	
3	3	3	3	2	1	3	3	3

The Horspool Search

N O B O D Y N O T I C E D H I M

N O T

Evaluation

- Although the worst and average times are still of the same efficiency class, the Horspool Algorithm is obviously faster on average.
- Good example of a space-time trade off.