

## Horspool's Algorithm

**Text:** JIMY\_HAILED\_THE\_LEADER\_TO\_STOP

**Pattern:** LEADER

```

JIMY_RAN_AND_HAILED_THE_LEADER_TO_STOP
      ||      | |      |      || |
LEADER      | |      |      || |

```

## Horspool's Algorithm

**Text:** JIMY\_HAILED\_THE\_LEADER\_TO\_STOP

**Pattern:** LEADER

```

JIMY_RAN_AND_HAILED_THE_LEADER_TO_STOP
      ||      | |      |      || |
LEADER      | |      |      || |
      LEADER |      |      || |

```

## Horspool's Algorithm

**Text:** JIMY\_HAILED\_THE\_LEADER\_TO\_STOP  
**Pattern:** LEADER

```

JIMY_RAN_AND_HAILED_THE_LEADER_TO_STOP
      ||      | |      |      || |
LEADER      | |      |      || |
      LEADER |      |      || |
      LEADER      |      || |

```

## Horspool's Algorithm

**Text:** JIMY\_HAILED\_THE\_LEADER\_TO\_STOP  
**Pattern:** LEADER

```

JIMY_RAN_AND_HAILED_THE_LEADER_TO_STOP
      ||      | |      |      || |
LEADER      | |      |      || |
      LEADER |      |      || |
      LEADER      |      || |
      LEADER      || |

```

## Horspool's Algorithm

**Text:** JIMY\_HAILED\_THE\_LEADER\_TO\_STOP  
**Pattern:** LEADER

```

JIMY_RAN_AND_HAILED_THE_LEADER_TO_STOP
  ||      | |      |      || |
LEADER      | |      |      || |
      LEADER |      |      || |
        LEADER      |      || |
          LEADER      || |
            LEADER      || |
              LEADER | |

```

## Horspool's Algorithm

**Text:** JIMY\_HAILED\_THE\_LEADER\_TO\_STOP  
**Pattern:** LEADER

```

JIMY_RAN_AND_HAILED_THE_LEADER_TO_STOP
  ||      | |      |      || |
LEADER      | |      |      || |
      LEADER |      |      || |
        LEADER      |      || |
          LEADER      || |
            LEADER      || |
              LEADER | |
                LEADER |

```

## Horspool's Algorithm

**Text:** JIMY\_HAILED\_THE\_LEADER\_TO\_STOP  
**Pattern:** LEADER

```

JIMY_RAN_AND_HAILED_THE_LEADER_TO_STOP
  ||      | |      |      || |
LEADER      | |      |      || |
      LEADER |      |      || |
        LEADER      |      || |
          LEADER      || |
            LEADER |      |
              LEADER |
                LEADER
  
```

The worst case cost is  $\Theta(nm)$ , but for random text is  $\Theta(n)$ .

## Try Yourself

**Text:** JIM\_SAW\_ME\_IN\_A\_BARBER\_SHOP

**Pattern:** BARBER

# Advanced Algorithm

## String Matching

### Horspool's Algorithm

Consider, as an example, searching for the pattern BARBER in some text:

$$s_0 \quad \dots \quad \quad \quad c \quad \dots \quad s_{n-1}$$

B A R B E R

Starting with the last R of the pattern and moving right to left, we compare the corresponding pairs of characters in the pattern and the text. If all the pattern's characters match successfully, a **matching** substring is found. (Then the search can be either stopped altogether or continued if another occurrence of the same pattern is desired.) If, however, we encounter a mismatch, we need to shift the pattern to the right. Clearly, we would like to make as large a shift as possible without risking the possibility of missing a **matching** substring in the text. Horspool's **algorithm** determines the size of such a shift by looking at the character  $c$  of the text that was aligned against the last character of the pattern.

## Try Yourself (Solution)

Text: JIM \_SAW \_ME \_IN \_A \_BARBER \_SHOP

Pattern: BARBER

## Horspool's Algorithm

**Algorithm 2.11:** Horspool

Input: text  $T = T[0 \dots n]$ , pattern  $P = P[0 \dots m]$

Output: position of the first occurrence of  $P$  in  $T$

Preprocess:

- (1) for  $c \in \Sigma$  do  $shift[c] \leftarrow m$
- (2) for  $i \leftarrow 0$  to  $m - 2$  do  $shift[P[i]] \leftarrow m - 1 - i$

Search:

- (3)  $j \leftarrow 0$
- (4) while  $j + m \leq n$  do
- (5)     if  $P[m - 1] = T[j + m - 1]$  then
- (6)          $i \leftarrow m - 2$
- (7)         while  $i \geq 0$  and  $P[i] = T[j + i]$  do  $i \leftarrow i - 1$
- (8)         if  $i = -1$  then return  $j$
- (9)      $j \leftarrow j + shift[T[j + m - 1]]$
- (10) return  $n$