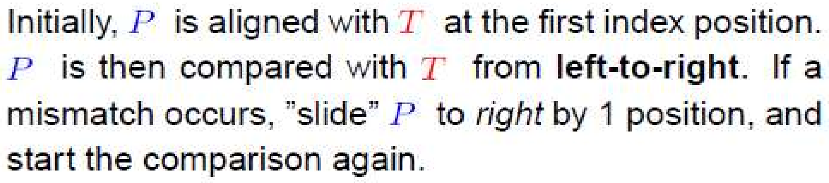
Advanced Algorithms

LAB -1

Language: C / C++ / Python

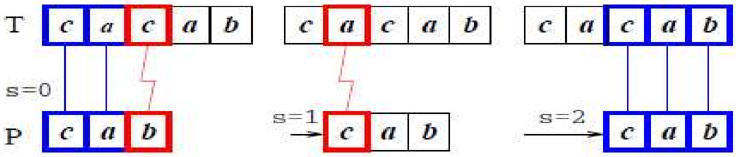
Brute-Force Algorithm



Initially, P is aligned with T at the first index position. P is then compared with T from **left-to-right**. If a mismatch occurs, "slide" P to *right* by 1 position, and start the comparison again.

Brute-Force Algorithm

Initially, P is aligned with T at the first index position. P is then compared with T from **left-to-right**. If a mismatch occurs, "slide" P to *right* by 1 position, and start the comparison again.

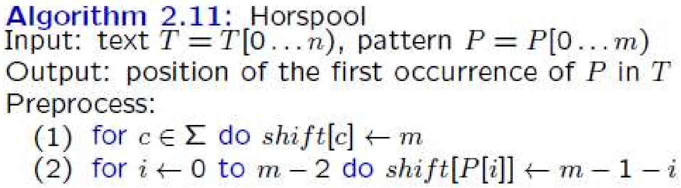


Complexity : Θ (m(n-m+1)) is equivalent to Θ(mn)

Brute-Force Algorithm

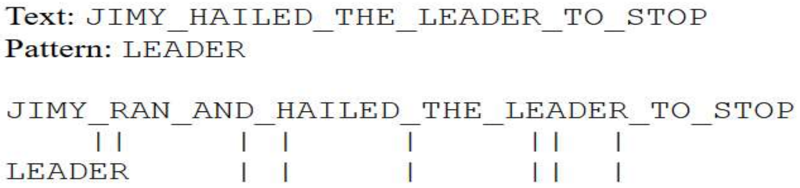
```
BF_StringMatcher(T, P) {
 n = length(T);
 m = length(P);

 // s increments by 1 in each iteration
 // => slide P to right by 1
 for (s=0; s<=n-m; s++) {
  // starts the comparison of P and T again
  i=1; j=1;
  while (j<=m && T[s+i]==P[j]) {
   // corresponds to compare P and T from
   // left-to-right
   i++; j++;
  }
  if (j==m+1)
   print "Pattern occurs with shift=", s
 }
}
```

Horspool's Algorithm



```
Algorithm 2.11: Horspool
Input: text $T = T[0 \ldots n)$, pattern $P = P[0 \ldots m)$
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 $shift[P[i]] \leftarrow m-1-i$
```

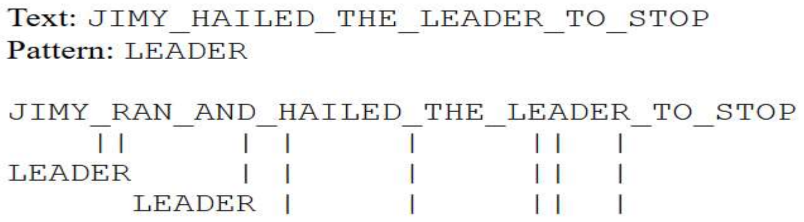
Horspool's Algorithm



Text: JIMY_HAILED_THE_LEADER_TO_STOP
Pattern: LEADER

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

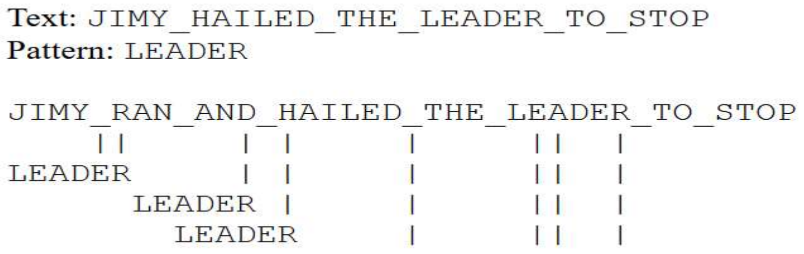
Horspool’s Algorithm



Text: JIMY_HAILED_THE_LEADER_TO_STOP
Pattern: LEADER

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

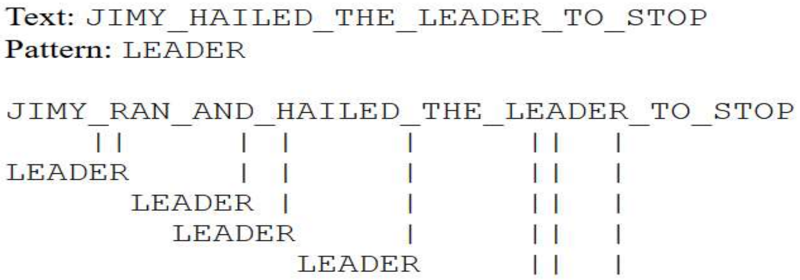
Horspool’s Algorithm



Text: JIMY_HAILED_THE_LEADER_TO_STOP
Pattern: LEADER

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

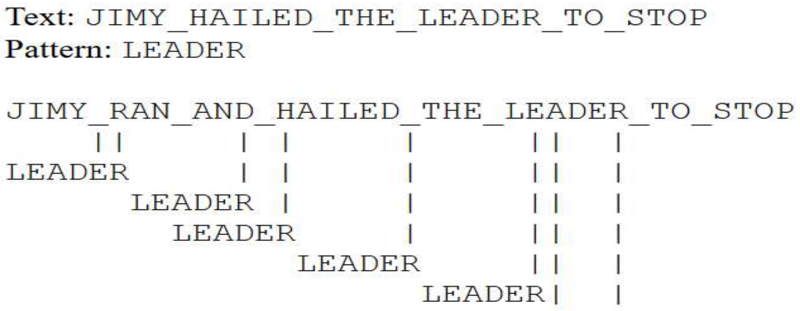
Horspool's Algorithm



Text: JIMY_HAILED_THE_LEADER_TO_STOP
Pattern: LEADER

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

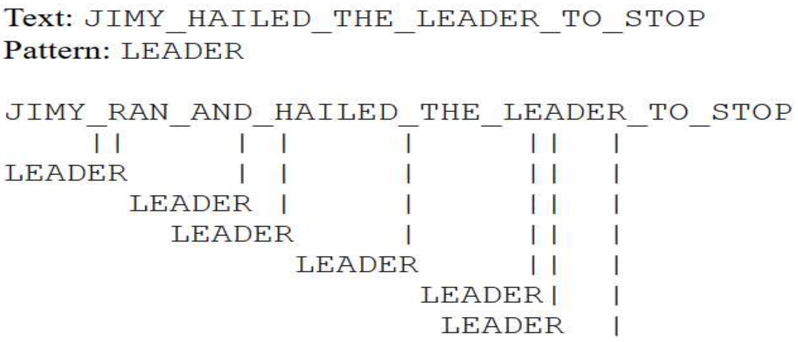
Horspool's Algorithm



Text: JIMY_HAILED_THE_LEADER_TO_STOP
Pattern: LEADER

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

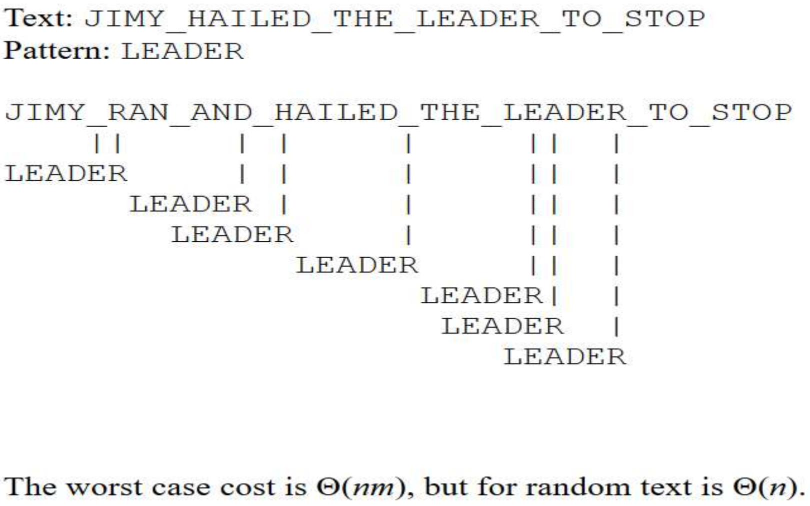
Horspool’s Algorithm



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

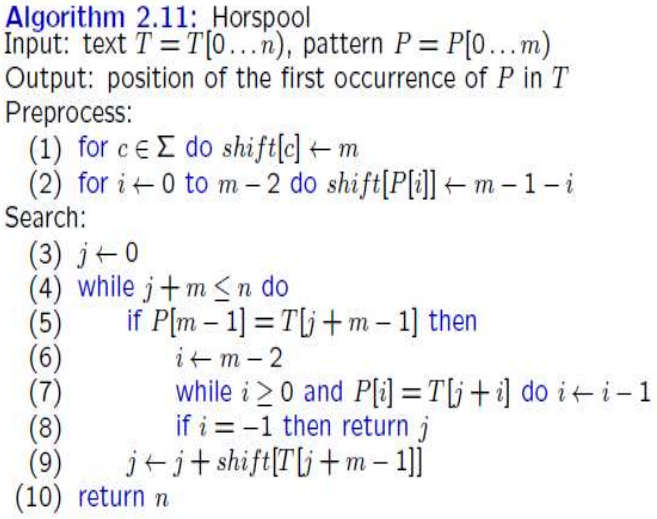


Text: JIMY_HAILED_THE_LEADER_TO_STOP
Pattern: LEADER

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

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

Horspool’s Algorithm



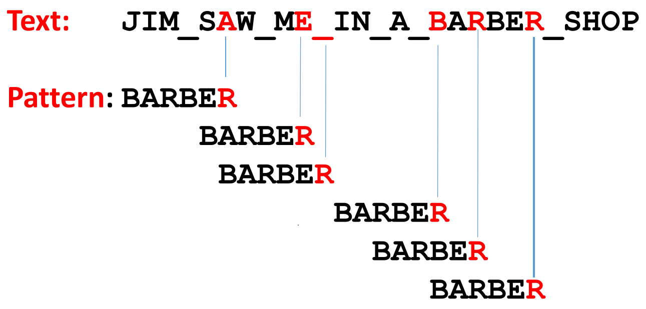
```
Algorithm 2.11: Horspool
Input: text $T = T[0 \dots n)$, pattern $P = P[0 \dots m)$
Output: position of the first occurrence of $P$ in $T$
Preprocess:
  for $c \in \Sigma$ do $shift[c] \leftarrow m$
  for $i \leftarrow 0$ to $m-2$ do $shift[P[i]] \leftarrow m-1-i$
Search:
  $j \leftarrow 0$
  while $j + m \leq n$ do
      if $P[m-1] = T[j+m-1]$ then
          $i \leftarrow m-2$
          while $i \geq 0$ and $P[i] = T[j+i]$ do $i \leftarrow i-1$
          if $i = -1$ then return $j$
      $j \leftarrow j + shift[T[j+m-1]]$
  return $n$
```

Try Yourself

Text: `JIM_SAW_ME_IN_A_BARBER_SHOP`

Pattern: `BARBER`

Try Yourself (Solution)



```
Text:    JIM_SAW_ME_IN_A_BARBER_SHOP

Pattern: BARBER
             BARBER
              BARBER
                    BARBER
                      BARBER
                         BARBER
```