# Exact String Matching:
# KMP & Boyer-Moore Algorithms

R.C.T. Lee and Chin Lung Lu

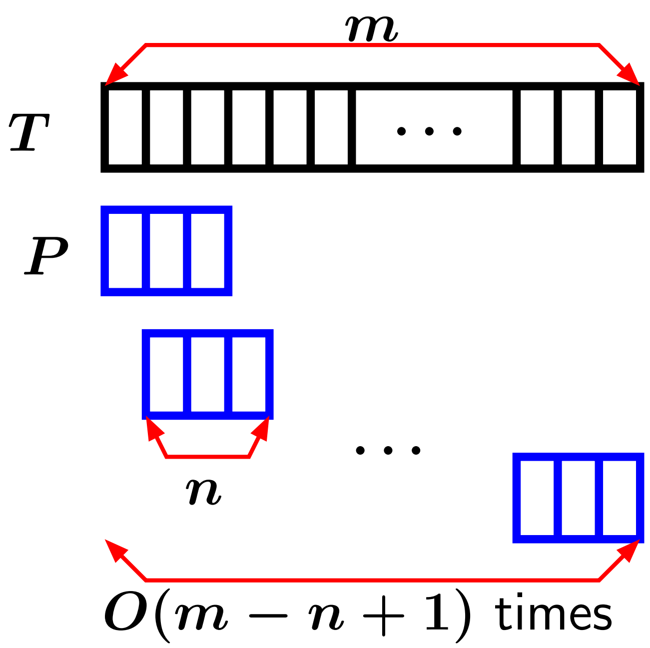
CS 5313 Algorithms for Molecular Biology



## Exact string matching

- Given two strings $T$ and $P$, where $|T| = m$ and $|P| = n$, find all the occurrences of $P$ in $T$?
- (**Simplified version**) Given two strings $T$ and $P$, if $P$ occurs in $T$?
- Brute-force method: $O(mn)$
- KMP algorithm (Knuth, Morris and Pratt, 1977): $\mathcal{O}(m+n)$
- Boyer-Moore algorithm (Boyer and Moore, 1977): $\mathcal{O}(m+n)$

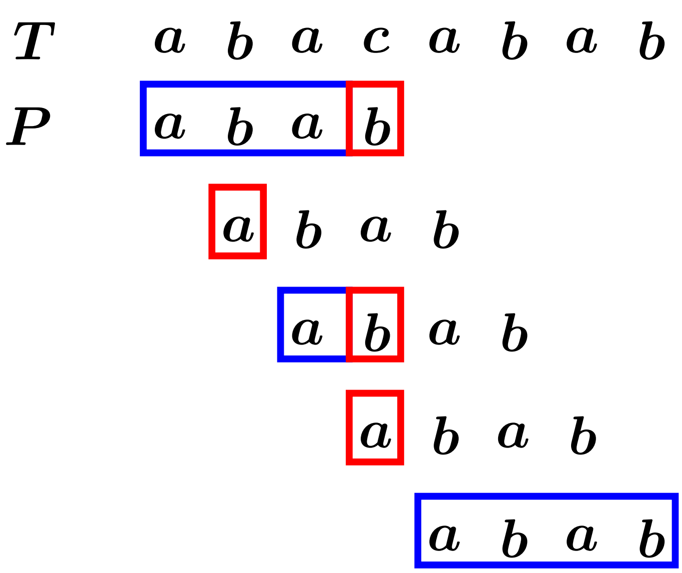
# Brute-force method




- Time complexity of brute-force method: $\mathcal{O}(mn)$



# Brute-force method




- Simple, but not efficient because it cannot avoid rescanning $T$

## How to speedup the brute-force method?

- Shift $P$ by more than one place when a mismatch occurs without missing any occurrence of $P$ in $T$

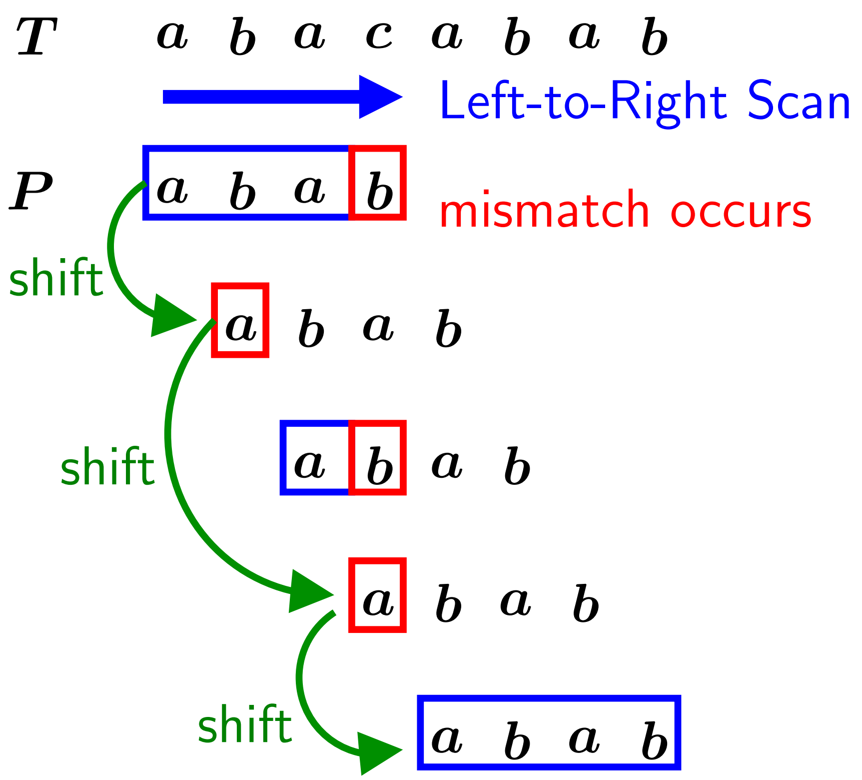
1. KMP algorithm
   - Proposed by Knuth, Morris and Pratt at 1977
   - Time complexity: $\mathcal{O}(m+n)$
2. Boyer-Moore algorithm
   - Proposed by Boyer and Moore at 1977
   - Time complexity: $\mathcal{O}(m+n)$



## KMP algorithm

- Left-to-right scan
- Failure function shift rule

# KMP algorithm

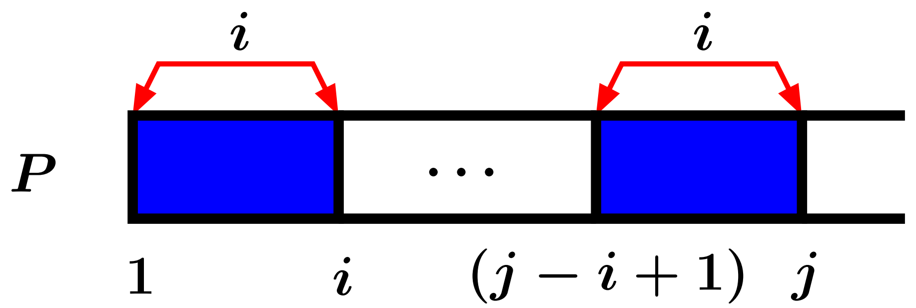





# Failure function

- If $P = p_1 p_2 \cdots p_n$, then its **failure function** $f$ is defined as follows, where $1 \leq j \leq n$.

$$f(j) = \begin{cases} \text{largest } i < j \text{ such that} & \text{if such an} \\ p_1 \cdots p_i = p_{j-i+1} \cdots p_j, & i \geq 1 \text{ exists,} \\ 0, & \text{otherwise.} \end{cases}$$



# Examples of failure function

- Example 1:

| $P$ | $a$ | $b$ | $a$ | $b$ | $a$ | $b$ | $c$ |
|---|---|---|---|---|---|---|---|
| $f(j)$ | 0 | 0 | 1 | 2 | 3 | 4 | 0 |

- Example 2:

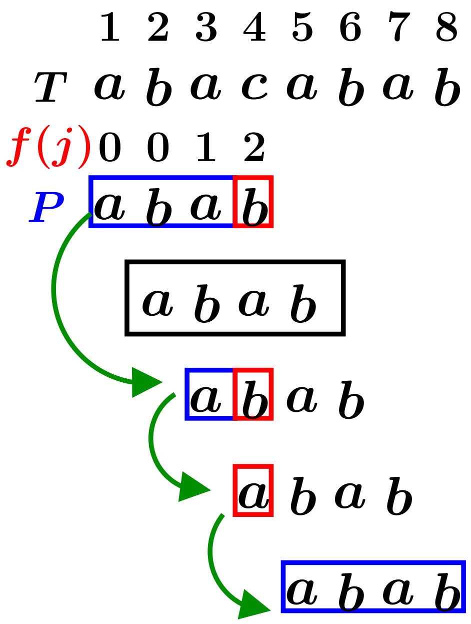
| $P$ | $a$ | $b$ | $a$ | $c$ | $a$ | $b$ | $a$ | $b$ |
|---|---|---|---|---|---|---|---|---|
| $f(j)$ | 0 | 0 | 1 | 0 | 1 | 2 | 3 | 2 |



# KMP algorithm

/* $T = T_1T_2\cdots T_m$ and $P = p_1p_2\cdots p_n$ */
$i = 1$; $j = 1$;
**while** $i \le m$ and $j \le n$ **do**
    **if** $T_i = p_j$ **then**
        $i = i + 1$; $j = j + 1$;
    **else if** $j = 1$ **then** $i = i + 1$; $j = 1$;
        **else** $j = f(j-1) + 1$;
    **end if**
**end while**
**if** $j = n + 1$ **then** "a match" **else** "no match".

# KMP algorithm




$(1,1) \rightarrow (4,4), j = f(3)+1 = 2$

No this step

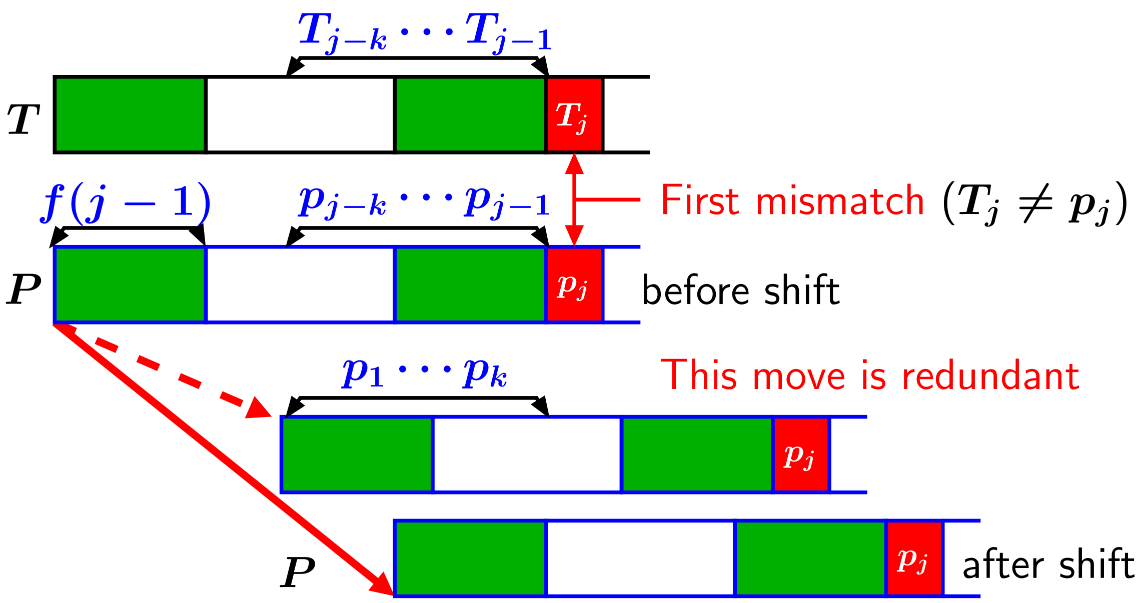
$(4,2) \rightarrow (4,2), j = f(1)+1 = 1$

$(4,1) \rightarrow (4,1), i = 4+1 = 5, j = 1$

$(5,1) \rightarrow (8,4), i = 8+1, j = 4+1$



# The correctness of KMP algorithm




Then $p_1 \cdots p_k = T_{j-k} \cdots T_{j-1} = p_{j-k} \cdots p_{j-1}$

Hence $f(j-1) = k > f(j-1)$, a contradiction.

# How to compute failure function?

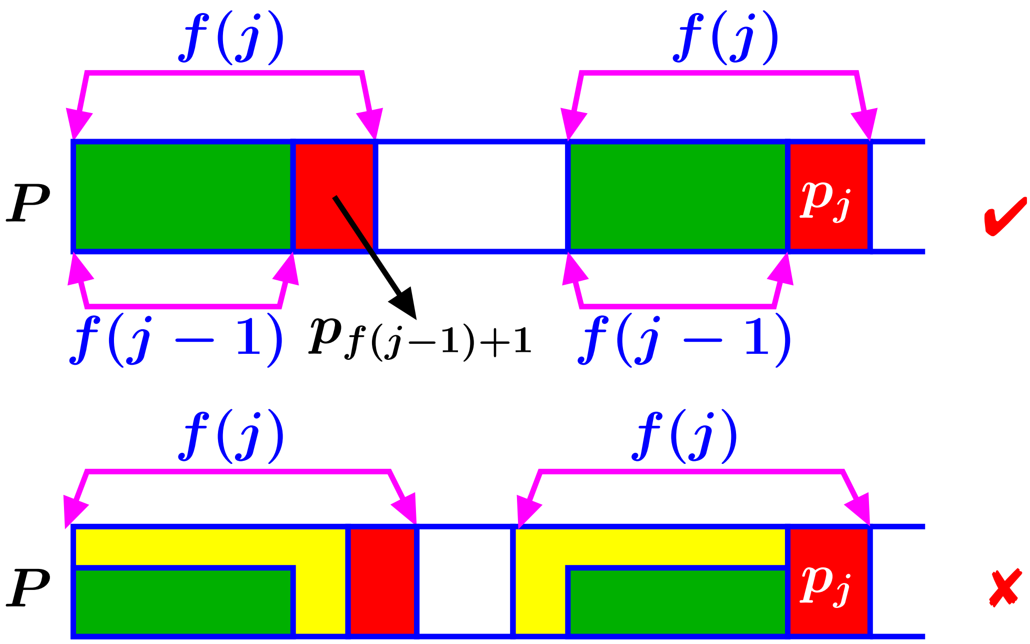
- Let $f^1(j) = f(j)$ and $f^k(j) = f(f^{k-1}(j))$.
- Let $P = p_1 p_2 \cdots p_n$. Then, $f(1) = 0$ and for $j \geq 2$,

$$f(j) = \begin{cases} f^k(j-1)+1 & \text{if there exists } k \text{ which is \textbf{the least integer} such that } p_{f^k(j-1)+1} = p_j, \\ 0 & \text{otherwise.} \end{cases}$$



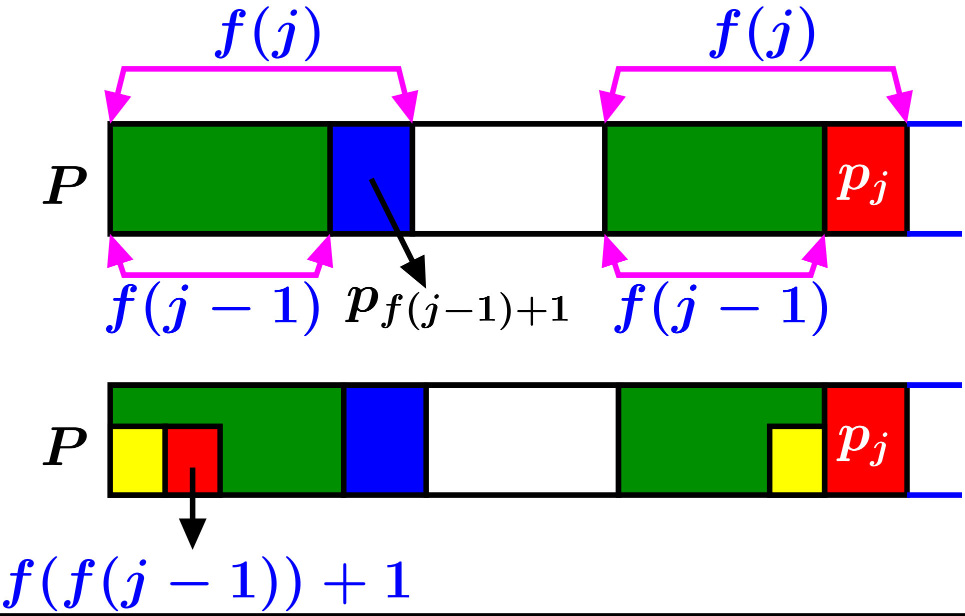
# How to compute failure function?

- If $p_{f(j-1)+1} = p_j$, then $f(j) = f(j-1)+1$.



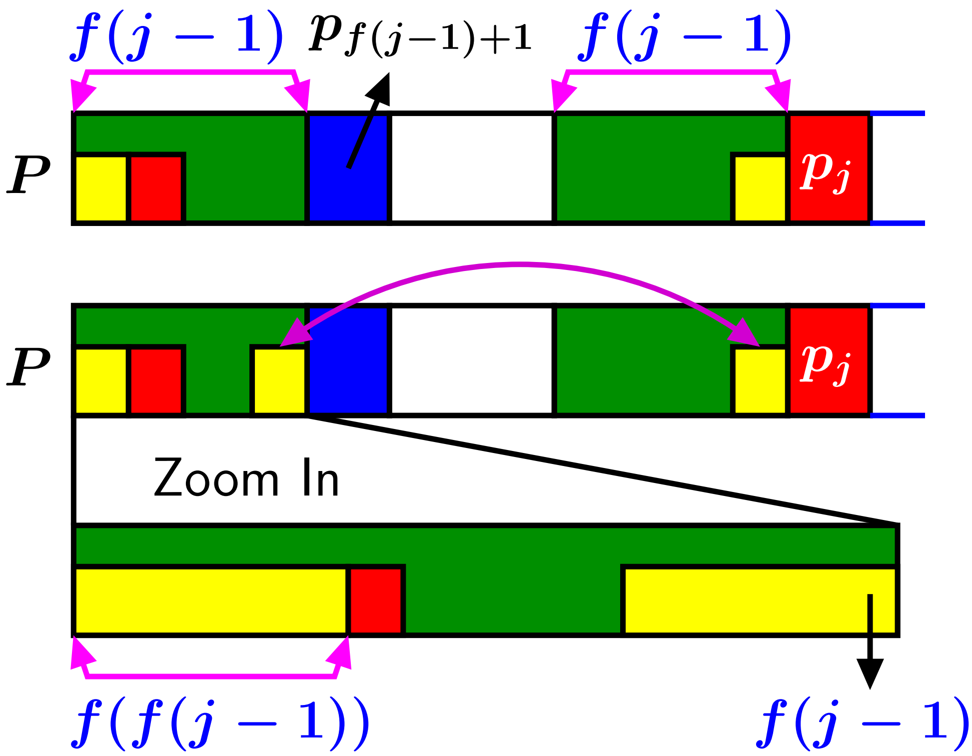
# How to compute failure function?

- If $p_{f(j-1)+1} \neq p_j$ and $p_{f(f(j-1))+1} = p_j$, then $f(j) = f(f(j-1)) + 1$.






# How to compute failure function?



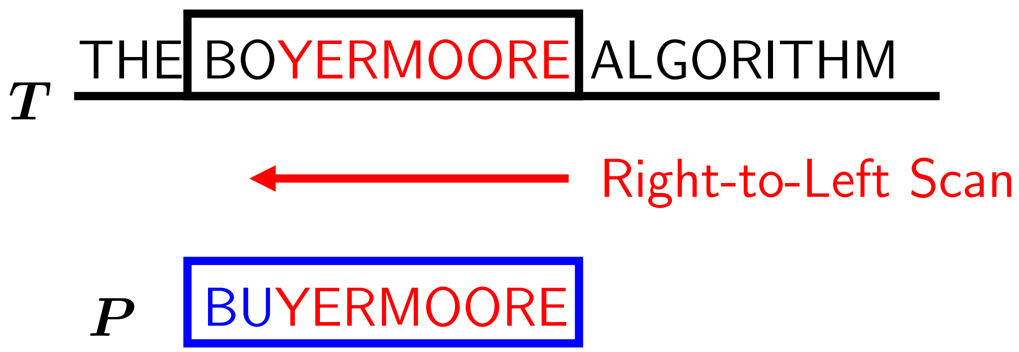
# Boyer-Moore algorithm

- Right-to-left scan
- Bad character shift rule
- Good suffix shift rule



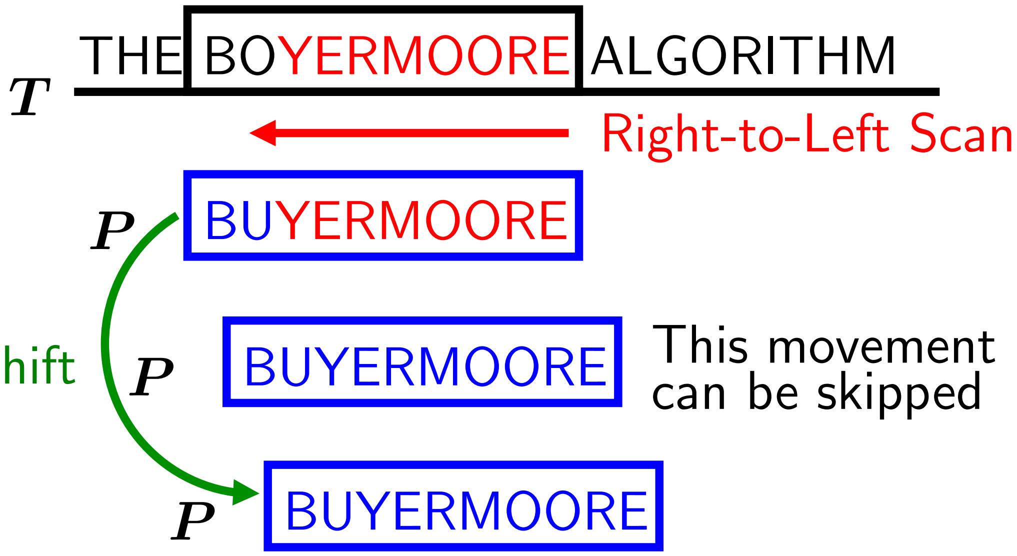
# Right-to-left scan

- Check whether $P$ occurs in $T$ at some position in the right-to-left scaning manner



## Bad character shift rule

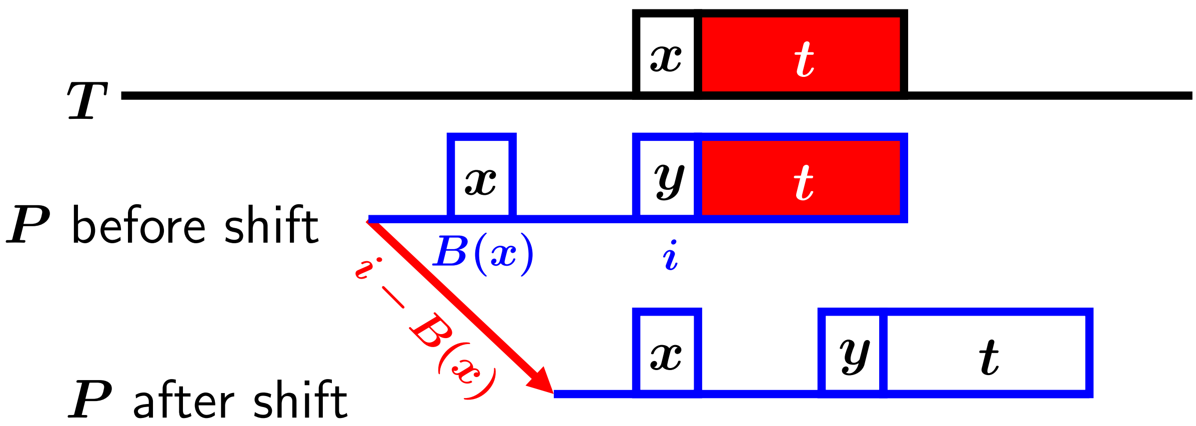
- What happened if the initial mismatch occurs?



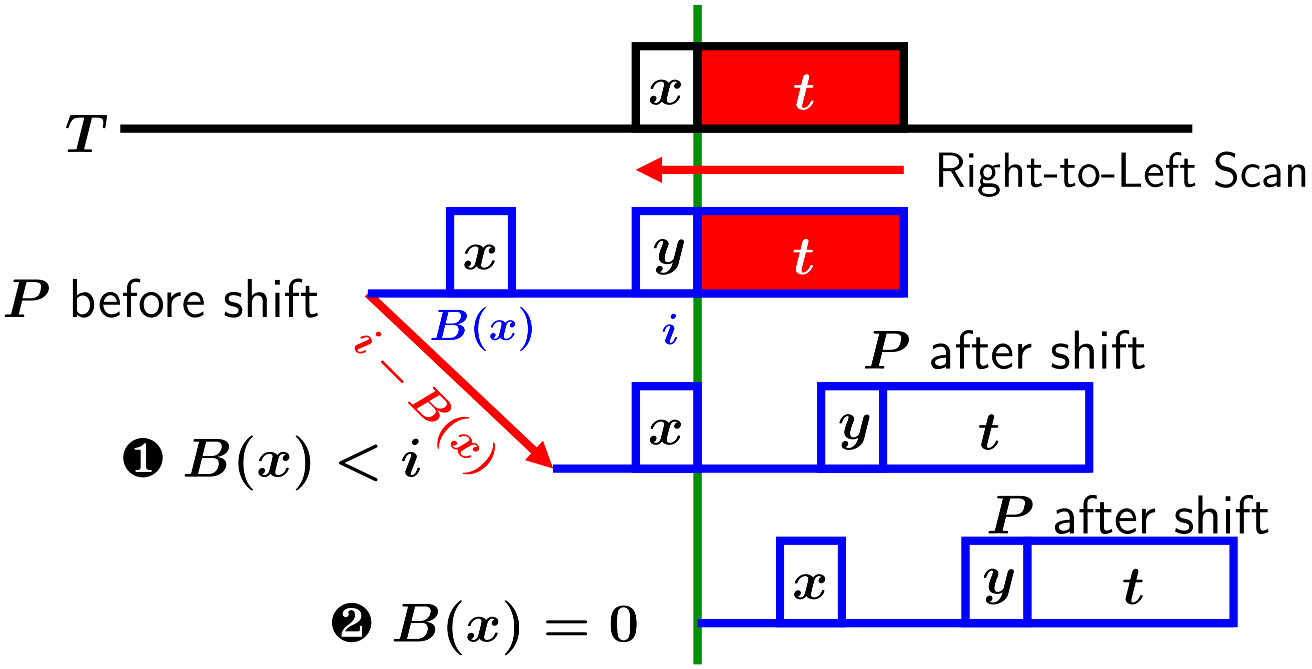



## Bad character shift rule

- Bad character rule: If $x \neq y$, then $P$ is shifted right by $\max\{1, i - B(x)\}$ places.
- $B(x)$: the position of right-most occurrence of $x$ in $P$ ($B(x) = 0$ if $x$ does not occur in $P$)



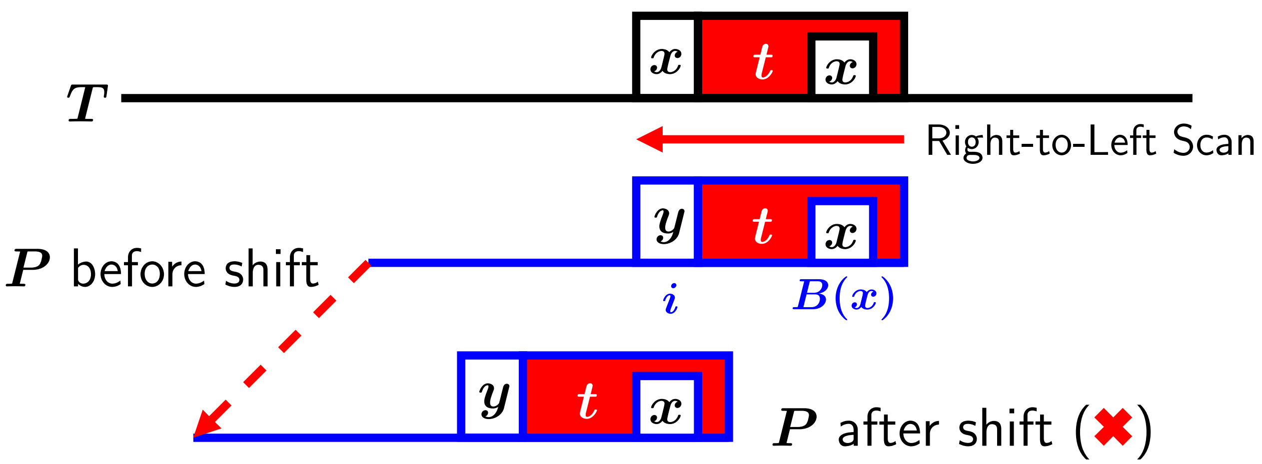
# Bad character shift rule






# Bad character shift rule

- If $B(x) > i$, then bad character rule has no effect.



# Bad character shift rule

- How to compute $B(x)$ for all $x$ in $P$? $(n = |P|)$

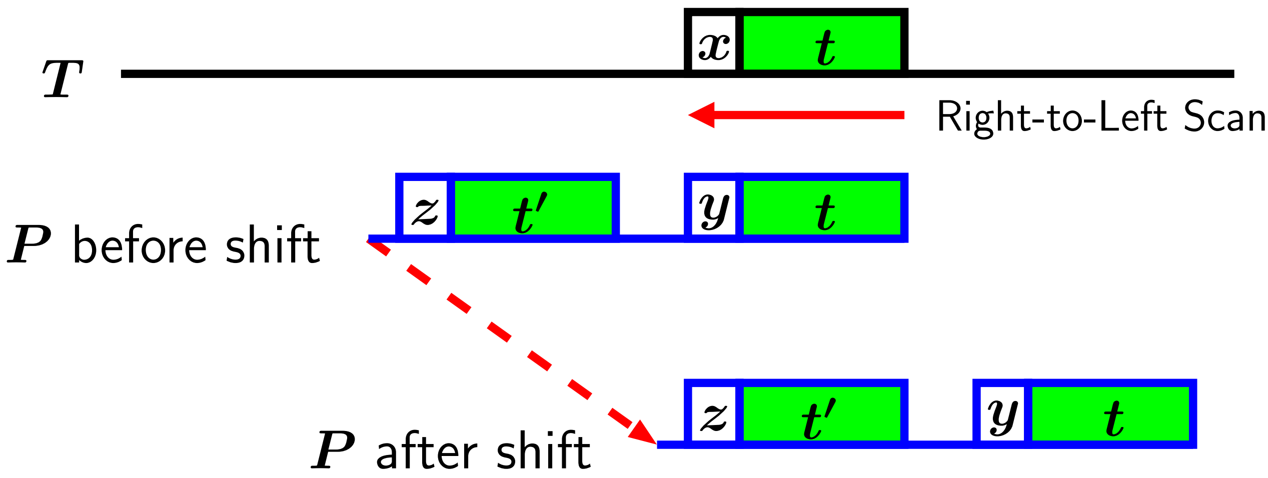
  **for** $i = 1$ to $n$ **do**
  
  $\quad B(x) = 0$;
  
  **end for**
  
  **for** $i = n$ to $1$ **do**
  
  $\quad$ **if** $B(P[i]) = 0$ **then** $B(P[i]) = i$;
  
  **end for**

- **Extended** bad character shift rule:
  For each position $i$ in $P$ and for each $x$ in $\Sigma$, find the position of the closest occurrence of $x$ in $P$ to the left of $i$.



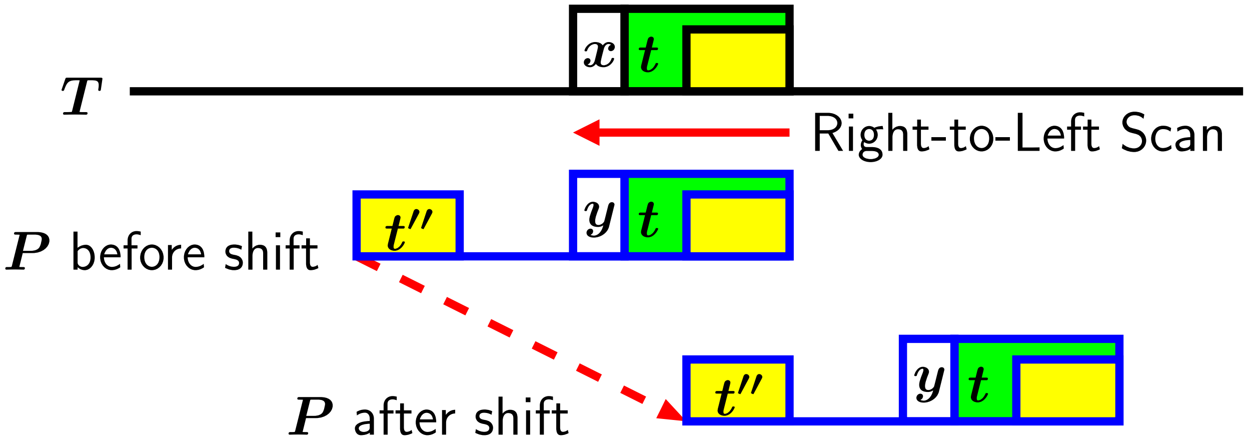
# Good suffix rule: Case ❶

- If $x \neq y$, then find the right-most copy $t'$ of $t$ in $P$ such that $t'$ is not a suffix of $P$ and $z \neq y$



# Good suffix rule: Case ❷

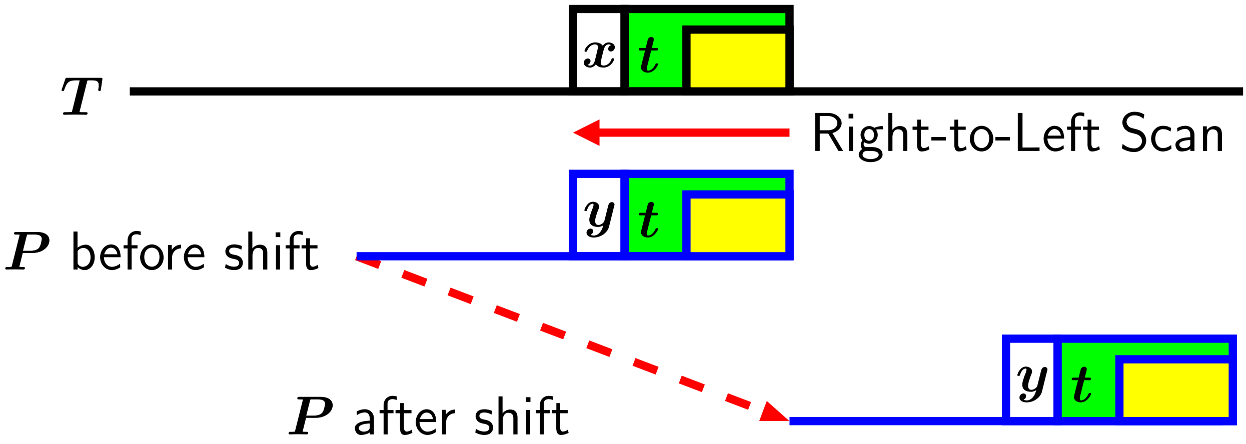
- If $t'$ does not exist, then find the largest prefix $t''$ of $P$ such that it is equal to a suffix of $t$






# Good suffix rule: Case ❸

- If $t'$ and $t''$ do not exist, then



# Good suffix rule