How to compute Boyer-Moore shifts

The good-suffix rule

For string \( s \) we define the set \( R(s) \) of all boundaries as
\[
R(s) = \{ s' \mid s' \text{ is a boundary of } s \}.
\]

Now, the conditions that an admissible shift \( \sigma \) has to satisfy can be expressed as follows:
\[
\begin{align*}
\sigma &\leq j \land s_{j+1} \ldots s_{m-1} \in R(s_{j+1-\sigma} \ldots s_{m-1}) \land s_j \neq s_{j-\sigma} & (1) \\
\sigma &> j \land s_0 \ldots s_{m-1-\sigma} \in R(s_0 \ldots s_{m-1}) & (2)
\end{align*}
\]

Define the array \( S \) containing the shortest admissible shift for each \( 0 \leq j \leq m \) as
\[
S[j] = \min \{ \sigma \mid (\sigma, j) \text{ fulfills condition (1) or fulfills condition (2)} \}.
\]

This rule for computing \( S \) is called good-suffix rule. The computation of \( S \) according to the good-suffix rule can be done as described in the following algorithm:

Algorithm: \text{ComputeShifts}
Input: string \( s \) with \( |s| = m \)
Output: array \( S \) containing shortest admissible shifts (according to the good-suffix rule)

1. FOR \( i := 0 \) TO \( m \)
2. \( S[i] := m \)
   /* computing shifts according to condition (1) */
3. \( H[0] := -1 \)
4. \( H[1] := 0 \)
5. FOR \( j := 2 \) TO \( m \)
6. WHILE \( k \geq 0 \) AND \( s_{m-k-1} \neq s_{m-j} \)
7. \( \sigma := j - k - 1 \)
8. \( S[m-k-1] := \min\{S[m-k-1], \sigma\} \)
9. \( k := H[k] \)
10. \( H[j] := k + 1 \)
11. \( k := k + 1 \)
   /* computing shifts according to condition (2) */
12. \( B := \text{ComputeBoundaries}(s) \) /* from Knuth-Morris-Pratt algorithm */
13. \( j := 0 \)
14. \( i := B[m] \)
15. WHILE \( i \geq 0 \)
16. WHILE \( j < m - i \)
17. \( S[j] := \min\{S[j], m - i\} \)
18. \( j := j + 1 \)
19. \( i := B[i] \)
20. \( j := 0 \)
The bad-character rule

How can we beneficially integrate the motivating idea that has led to the right-to-left approach? We define another rule called bad-character rule. Consider a mismatch at \((i, j)\) caused by symbol \(t_{i+j} = x\). There are two possible cases:

1. There is a \(0 \leq r \leq j - 1\) such that \(s_r = x\). Then, define \(\sigma = \text{def } j - r\).

2. For all \(0 \leq r \leq j - 1\) it holds \(s_r \neq x\). Then, define \(\sigma = \text{def } j + 1\).

It is easily seen that we can combine the good suffix rule and the bad character rule by taking the maximum of the shifts for each \(j\).