How to compute Boyer-Moore shifts

The good-suffix rule

For string s we define the set $R(s)$ of all boundaries as

$$R(s) =_{\text{def}} \left\{ s' \;\middle|\; s' \text{ is a boundary of } s \right\}.$$

Now, the conditions that an admissible shift σ has to satisfy can be expressed as follows:

$$\sigma \leq j \;\wedge\; s_{j+1} \dots s_{m-1} \in R(s_{j+1-\sigma} \dots s_{m-1}) \;\wedge s_j \neq s_{j-\sigma} \tag{1}$$

$$\sigma > j \;\wedge\; s_0 \dots s_{m-1-\sigma} \in R(s_0 \dots s_{m-1}) \tag{2}$$

Define the array S containing the shortest admissible shift for each $0 \leq j \leq m$ as

$$S[j] =_{\text{def}} \min \left\{ \sigma \;\middle|\; (\sigma, j) \text{ fulfills condition (1) or fulfills condition (2)} \right\}.$$

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

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

```
FOR i := 0 TO m
    S[i] := m

/* computing shifts according to condition (1) */

H[0] := -1
H[1] := 0
FOR j := 2 TO m
    WHILE k >= 0 AND s_{m-k-1} != s_{m-j}
        σ := j - k - 1
        S[m - k - 1] := min{S[m - k - 1], σ}
        k := H[k]
    H[j] := k + 1
    k := k + 1

/* computing shifts according to condition (2) */

B := ComputeBoundaries(s)            /* from Knuth-Morris-Pratt algorithm */
j := 0
i := B[m]
WHILE i >= 0
    WHILE j < m - i
        S[j] := min{S[j], m - i}
        j := j + 1
    i := B[i]
    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 an $0 \leq r \leq j-1$ 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.