The Knoth-Pratt-Morris Algorithm

Input: Text $T[1..n]$ and pattern $P[1..m]$. Array $\Pi[1..m]$ where $\Pi[j]$ is the length of the longest string that is both a proper suffix and a proper prefix of $P[1..j]$.

Output: All positions $q$ ($0 \leq q \leq n - m$) where the pattern “occurs”, that is, such that for all $i = 1, 2, \ldots, m$, we have that $P[i] = T[i + q]$.

1 $j \leftarrow 1$
2 $q \leftarrow 0$
3 while $(q \leq n - m)$
4     if $(j == m + 1)$
5         report $q$ as an occurrence of $P$ in $T$
6         $q \leftarrow q + (j - 1) - \Pi[j - 1]$
7         $j \leftarrow \Pi[j - 1] + 1$
8     else if ($T[q + j] == P[j]$)
9         $j \leftarrow j + 1$
10    else
11       $q \leftarrow q + (j - 1) - \Pi[j - 1]$
12       $j \leftarrow \Pi[j - 1] + 1$
13 endif
14 endwhile

For the analysis: every execution of the while increases the quantity $2q + j$.

Computing the array $\Pi$

Input: Pattern $P[1..m]$.

Output: Array $\Pi[1..m]$ where $\Pi[j]$ is the length of the longest string that is both a proper suffix and a proper prefix of $P[1..j]$.

1 $\Pi[1] \leftarrow 0$
2 $\Pi[0] \leftarrow -1$
3 for $q \leftarrow 2$ to $m$
4     $j \leftarrow \Pi[q - 1]$
5     while $(j \geq 0$ and $P[q] \neq P[j + 1])$
6         $j \leftarrow \Pi[j]$
7     endwhile
8     $\Pi[q] \leftarrow j + 1$
9 endwhile

Idea/invariant used in the running time analysis: In Line 3 (except when executed the first time), $j$ is incremented. $q$ is also incremented, so every time we execute lines 4 and 8, or 5 and 6, the quantity $2q - j$ increases.