shall see. KMP-MATCHER calls the auxiliary procedure COMPUTE-PREFIX-FUNCTION to compute π.

```
KMP-MATCHER(T, P)
n ← length[T]
m ← length[P]
π ← COMPUTE-PREFIX-FUNCTION(P)
q ← 0                             ▹Number of characters matched.
for i ← 1 to n                    ▹Scan the text from left to right.
     do while q > 0 and P[q + 1] ≠ T[i]
            do q ← π[q]     ▹Next character does not match.
        if P[q + 1] = T[i]
           then q ← q + 1         ▹Next character matches.
        if q = m                       ▹Is all of P matched?
           then print "Pattern occurs with shift" i - m
                q ← π[q]     ▹Look for the next match.
```

```
COMPUTE-PREFIX-FUNCTION(P)
 1 m ← length[P]
 2 π[1] ← 0
 3 k ← 0
 4 for q ← 2 to m
 5      do while k > 0 and P[k + 1] ≠ P[q]
 6             do k ← π[k]
 7         if P[k + 1] = P[q]
 8            then k ← k + 1
 9         π[q] ← k
10 return π
```

We begin with an analysis of the running times of these procedures. Proving these procedures correct will be more complicated.

Running-time analysis

The running time of COMPUTE-PREFIX-FUNCTION is Θ(m), using the potential method of amortized analysis (see Section 17.3). We associate a potential of k with the current state k of the algorithm. This potential has an initial value of 0, by line 3. Line 6 decreases k whenever it is executed, since $\pi[k] < k$. Since $\pi[k] \geq 0$ for all k, however, k can never become negative. The only other line that affects k is line 8, which increases k by at most one during each execution of the **for** loop body. Since $k < q$ upon entering the **for** loop, and since q is incremented in each iteration of the **for** loop body, $k < q$ always holds. (This justifies the claim that $\pi[q] < q$ as well, by line 9.) We can pay for each execution of the **while** loop body on line 6 with the corresponding decrease in the potential function, since $\pi[k] < k$. Line 8 increases the potential function by at most one, so that the amortized cost of the loop body on lines 5-9 is $O(1)$. Since the number of outer-loop iterations is Θ(m), and since the final potential function is at least as great as the initial potential function, the total actual worst-case running time of COMPUTE-PREFIX-FUNCTION is Θ(m).

A similar amortized analysis, using the value of q as the potential function, shows that the matching time of KMP-MATCHER is Θ(n).

Compared to FINITE-AUTOMATON-MATCHER, by using π rather than δ, we have reduced the time for preprocessing the pattern from $O(m\ |\Sigma|)$ to Θ(m), while keeping the actual matching time bounded by Θ(n).