Notation

For the following analysis, we have some predefined notations.

- For any array A, any indices l and r, $A[l, r)$ represents the subarray of A in range $[l, r)$:

  $$A[l, r) := (A[l], \ldots, A[r-1]).$$

  Note that when $l \geq r$, $A[l, r)$ is empty.

  Similarly,

  $$\begin{aligned} A[l, r] &:= (A[l], \ldots, A[r]), \\ A(l, r] &:= (A[l+1], \ldots, A[r]), \\ A(l, r) &:= (A[l+1], \ldots, A[r-1]). \end{aligned}$$

Algorithm

```
1 compute_prefix_function(P[0, m))
2     pi[0, m) = empty array
3     pi[0] = 0
4     for q = 1 to m - 1
5         k = pi[q - 1]
6         while k > 0 and P[k] != P[q]
7             k = pi[k - 1]
8         if P[k] == P[q]
9             k++
10        pi[q] = k
11    return pi
```

Correctness

Compute Prefix Function

Input: A string P of size m, where $m \geq 1$.
Output: An array $\pi[0, m)$ such that for all valid indices q, $\pi[q]$ represents the size k of the longest proper prefix of $P[0, q]$ that is also a suffix of $P[0, q]$. Let's refer to it as the longest *prefix-suffix* of $P[0, q]$.

$$\pi[q] = \max\{k : P[0, k) = P(q - k, q] \text{ and } k \leq q\} \tag{1}$$

Proof

Define the set of loop invariant $\mathcal{I}$ that at the start each iteration of the for loop in line 4-10,

- $\mathcal{I}_1$: In bounds
$$0 < q \leq m$$
- $\mathcal{I}_2$: $\pi[0, q)$ is correct as the desired output.

1. **Initialization:** At the start we have $q = 1$, and $\pi[0]$ is correctly initialized in line 3, thus $\mathcal{I}$ is correct trivially.
2. **Maintenance:** Assume that $\mathcal{I}$ is true at the start of an iteration, we aim to show that it remains true at the start of the next iteration.

   The boundary check $\mathcal{I}_1$ remains true trivially since q is only modified in the for loop statement. Let's focus on $\mathcal{I}_2$.

Based on the assumption of $\mathcal{I}$, in line 5 we have k as the size of the longest prefix-suffix of $P[0, q-1]$.

$$P[0, k) = P[q-k, q), \quad 0 \leq k < q \tag{2}$$

And now we want to see whether we expand or shrink this prefix by first checking if $P[k] = P[q]$.

(a) Let's first consider $k = 0$ and $P[k] \neq P[q]$. This means the first element fails to match, so a prefix-suffix does not exist, and we stored 0 into $\pi[q]$. $\mathcal{I}_2$ is correct.

(b) Now we consider $P[k] = P[q]$. This means the previous prefix-suffix remains matched and can be expanded in this iteration.

In this case, the while loop at line 6-7 is skipped, and we enter line 9: k is increased by 1. That is, the prefix expands, now including $P[k]$ corresponding to $P[q]$:

$$P[0, k) = P[q-(k-1), q] = P(q-k, q], \quad k \leq q.$$

We can also claim that this prefix-suffix is the longest possible since it is inherited from $\pi[q-1]$. Suppose to the contrary that there were a longest valid prefix-suffix of length $k' > k$, its first $k'-1$ characters would contradict the maximality of $\pi[q-1]$. Thus, k is the size of the longest proper prefix and the suffix of $P[0, q]$, and we stored it correctly in $\pi[q]$. $\mathcal{I}_2$ remains true.

(c) Finally, let's tackle the case where $k \neq 0$ and $P[k] \neq P[q]$. This means that the previous prefix-suffix fails to match in this iteration. But we do not have to discard it completely since the π we recorded so far provides some useful information.

Let's examine how the *fallback* mechanic works in the while loop at line 6-7 with another set of loop invariant $\mathcal{J}$.

- $\mathcal{J}_1$: In bounds.

$$0 \leq k < q$$

- $\mathcal{J}_2$: k is the size of a proper prefix and suffix of $P[0, q)$. (not necessarily longest)

$$P[0, k) = P[q-k, q)$$

- $\mathcal{J}_3$: There does not exist $k' > k+1$ such that k' is the size of a proper prefix and suffix of $P[0, q]$.

$$\nexists\, k' > k+1 : \quad P[0, k') = P(q-k', q]$$

With $\mathcal{J}$, we have $(k+1)$ always being a candidate for $\pi[q]$ if $P[k]$ and $P[q]$ match, and we do not miss any valid prefix-suffix along the way.

$\boxed{\text{Initialization}}$ By equation (2) in line 5, $\mathcal{J}_1$ and $\mathcal{J}_2$ are true. And the maximality of $k = \pi[q-1]$ ensures $\mathcal{J}_3$.

$\boxed{\text{Maintenance}}$ Assume that $\mathcal{J}$ is true at the start of an iteration, we want to show that it remains true at the end of this iteration.

Since $\pi[k-1]$ represents the size of the longest prefix-suffix of $P[0, k-1]$, let $\hat{k} = \pi[k-1]$ be the update of k at line 7, then we have

$$P[0, \hat{k}) = P[k - \hat{k}, k), \quad 0 \leq \hat{k} < k \tag{3}$$

Combine this with $\mathcal{J}_2$, then

$$P[0, \hat{k}) = P[k - \hat{k}, k) = P[q - \hat{k}, q), \quad 0 \leq \hat{k} < k < q \tag{4}$$

Thus, $\hat{k}$ is the size of a valid prefix-suffix. $\mathcal{J}_1$ and $\mathcal{J}_2$ remain true after updating k.

By the assumption of $\mathcal{J}_3$, we know that there does not exist $k' > k+1$ being the size of a valid prefix-suffix, and we want to expand the invalid range to $k' > \hat{k} + 1$.

This can be established by the maximality of $\pi[k-1]$ as the following. Suppose to the contrary that there were a prefix-suffix of $P[0, q]$ of size k',

$$P[0, k') = P(q - k', q], \quad \hat{k} + 1 < k' \leq k + 1$$

that is,

$$P[0, k' - 1) = P[q - (k' - 1), q), \quad \hat{k} < k' - 1 \leq k.$$

But if we combine this with equation (4), we would get

$$P[0, k' - 1) = P[q - (k' - 1), q) = P[k - (k' - 1), k).$$

This means we would have a prefix-suffix of $P[k-1]$ of size $k' - 1$, which would be longer than $\pi[k-1] = \hat{k}$. This contradicts the maximality of $\pi[k-1]$.

Thus, $\mathcal{J}_3$ remains true after we update k as $\hat{k}$.

$\boxed{\text{Termination}}$ The while loop always terminates since with equation (4), k is always decreasing, so either k reaches 0 first, or we find $P[k] = P[q]$ first.

Based on the initialization and maintenance of $\mathcal{J}$, we now have either

(i) $k = 0$ and $P[k] \neq P[q]$, this has been solved in case (a), or

(ii) $P[k] = P[q]$. Based on $\mathcal{J}$, we have $k+1$ as the size of the longest prefix-suffix of $P[0, q]$, so we can increase k by 1, and correctly store it in $\pi[q]$. $\mathcal{I}_2$ remains true.

3. **Termination:** The loop always terminates since q is increasing and always reaches m.

When the loop terminates as $q = m$, based on the initialization and maintenance of $\mathcal{I}$, we have $\pi[0, m)$ all being the correct output. Therefore, the algorithm is correct.