問題: 給定一 $T[1,\dots,n]$ 之文章

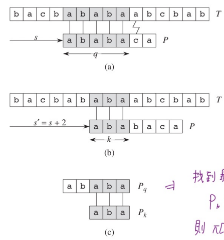
而 $P[1,\dots,m]$ 為一 pattern, $m \le n$

若 $T[s+1,\dots,s+m] = P[1,\dots,m]$, 則稱 s 為一 valid shift

其中: $0 \le s \le n-m$, 欲求出 T 中的所有 valid shift

Example: T: "ababaa" ∴ $s=0,2$ 為2 valid shift

P: "aba"



$\Rightarrow$ 找到最大 k 且 $k<q$ 㪚 $P_k \sqsupset P_q$, 即 P_k 為 P_q 的 suffix

則 $\pi[q]=k$

$s' = s+(q-\pi[q]) = s+(q-k)$ 為下一个可能的 valid shift

Figure 32.10 The prefix function π. **(a)** The pattern P = ababaca aligns with a text T so that the first $q = 5$ characters match. Matching characters, shown shaded, are connected by vertical lines. **(b)** Using only our knowledge of the 5 matched characters, we can deduce that a shift of $s+1$ is invalid, but that a shift of $s' = s+2$ is consistent with everything we know about the text and therefore is potentially valid. **(c)** We can precompute useful information for such deductions by comparing the pattern with itself. Here, we see that the longest prefix of P that is also a proper suffix of P_5 is P_3. We represent this precomputed information in the array π, so that $\pi[5] = 3$. Given that q characters have matched successfully at shift s, the next potentially valid shift is at $s' = s + (q - \pi[q])$ as shown in part (b).

Bruteforce: 共做 n 輪, 每一輪:

(1). 將 $T[1+s,\dots,m+s]$ 和 $P[1,\dots,m]$ 比对

(2). shift s 1 格

Example: ababaa $s=0,2$ 為2 valid shift

```
s=0 aba
1    aba
2     aba
3      aba
```

Time Complexity: $O(n-m+1)\cdot O(m)$

$= O(nm)$

KMP Algorithm: 利用已計算資訊減少 shift 次數

定義 P_k 為 $P[1,\dots,k]$

∵ 設下一次的 valid shift 為 P', P' 的 prefix 一定要和 P_k 之 suffix 配对

∴ 可查表得到 P_k 的 suffix 和 P 之最大配对長度

Example:

```
         1 2 3 | 4 5 6
T        a b a | b a a
P3 →     a b a |
             a | b a
          s'   ↓
             π[3]
```

∵ $\pi[3]$ 為 1 ∴ shift s' 後 prefix 長度為1, 可推得 $s' = k - \pi[k]$

故需先解得 prefix function problem

Prefix function: 令 $\pi : \{1, \dots, m\} \longrightarrow \{0, \dots, m-1\}$ by

$\pi[k] = \max\{ r \mid r < k$ 且 P_r 為 P_k 之 prefix $\}$

且 $\pi[1] = 0$

Example: $P =$ "ababaa"

	1	2	3	4	5	6
π	0	0	1	2	3	1

k=2:
ab|abaa
|ababaa

k=3:
aba|baa
a|babaa

k=4:
abab|aa
ab|abaa
|ababaa

k=5
ababa|a
aba|baa

k=6
ababaa|
a|babaa

Time Complexity: Prefix function 若用 amortized cost 為 $O(m)$

∵ 每次只是把 pattern 向後移，最多移 $O(m)$

同理 KMP 為 $O(n)$

故為 $O(n+m)$

COMPUTE-PREFIX-FUNCTION(P)

```
1  m = P.length
2  let π[1..m] be a new array
3  π[1] = 0
4  k = 0
5  for q = 2 to m
6      while k > 0 and P[k+1] ≠ P[q]
7          k = π[k]
8      if P[k+1] == P[q]
9          k = k + 1
10     π[q] = k
11 return π
```

∵ $k \geq 0$，又 while loop 會使 k 值下降

而 line 9 會使 k 值 increment

又 for loop 執行 m-1 次，可知：k 會 bound 在 m-1 間

從 π[q] 定義可知。 ↑ codomain

∴ while loop 最多做 m-1 次

⇒ 可得 $O(m)$.

These two procedures have much in common, because both match a string against the pattern P: KMP-MATCHER matches the text T against P, and COMPUTE-PREFIX-FUNCTION matches P against itself.

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

操作題:

Note: 若為將 prefix function 定義為 failure function，其中 array index 由 0 開始

BC 為 $f[0] = -1$

①. 求 failure function

i	0	1	2	3	4	5	6	7	8
P_i	a	b	a	a	b	a	a	a	a
f_i	-1	-1	0	0	1	2	3	0	0

3 2

E D C B A A

0 0

0 2

0 3 4

0 3 6 6

6 9 8