KMP

Wednesday, December 1, 2021 2:05 PM

Algoritmo:

1. Processamento $O(m)$
   ↳ Calcolo di ϕ Prefix Function.
2. Scansione del Testo in $O(n)$

Passo 1 - Calcolo Prefix Function

$$\phi : \{0 \dots m\} \to \{-1 \dots m\}$$

$$\phi(j) = \begin{cases} |B(P[1,j])| & 1 \le j \le m \\ -1 & j = 0 \end{cases}$$

P	a	b	c	a	b	a	a	b	c	a	b	a	b	m=13

j	0	1	2	3	4	5	6	7	8	9	10	11	12	13
φ	-1	0	0	0	1	2	1	1	2	3	4	5	6	2

Bordo di una Stringa (Ricorsivo)

- $B(\varepsilon) = \varepsilon$
- $B(\sigma) = \varepsilon$
- $B(X\sigma) = \begin{cases} B(X)\sigma & X[|B(X)|+1] = \sigma \\ B(B(X)\sigma) & \text{Altrimenti} \end{cases}$

X	a	b	a	c	b	a	b	b	a	a	b	a	c	b

Caso 1.

B(X) = abac → B(X**b**) = B(X)**b** = abac**b**

X	a	b	a	c	b	a	b	b	a	a	b	a	c	a

Caso 2.

B(X) = abac → B(X**a**) = B(B(X)**a**) = B(abac**a**) = a

Calcolo di ϕ per Induzione

L'idea è quella di calcolare $\phi(j)$ partendo dai valori che conosciuti in precedenza ($\phi(j-1)$, $\phi(j-2) \dots \phi(0)$)

- Per definizione $\phi(j) = |B(P[1,j])|$

$$P[1,j] = P[1,j-1]\, P(j)$$

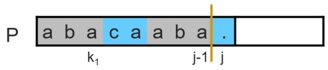
$$\Rightarrow \phi(j) = |B(\underbrace{P[1,j-1]}_{X}\ \underbrace{P(j)}_{\sigma})|$$

Notiamo che siamo ricaduti nella definizione ricorsiva di Bordo.

$$\Rightarrow \begin{cases} B(P[1,j-1])\, P(j) \\ B(B(P[1,j-1])\, P(j)) \end{cases}$$

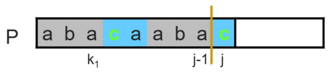
Poniamo $|B(P[1, j-1])| = K_1 = \phi(j-1)$

Per definizione della funzione ϕ

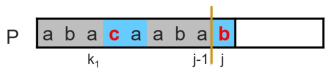


Dalla definizione ricorsiva di Bordo ricaviamo

$$\begin{cases} B(\underline{P[1,j-1]})\ \underline{P(j)}) & \cdot\ P[K_1+1] = P(j) \quad (1) \\ B(B(\underline{P[1,j-1]})\ \underline{P(j)})) & \text{Altrimenti} \quad (2) \end{cases}$$



Caso (1)



Caso (2)

Risoluzione del Caso (1)

$\phi(j) = |B(P[1,j-1]\ P[j])|$

Supponiamo allora che $P[K_1+1] = P[j]$

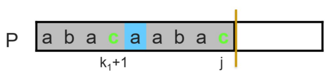
$\Rightarrow B(P[1,j-1]\ P[j]) = B(P[1,j-1])\ P[j])$

$\Rightarrow |B(P[1,j-1])\ P[j]| = \underbrace{|B(P[1,j-1])|}_{K_1} + 1$

$= K_1 + 1 = \phi(j-1) + 1$

$\Rightarrow \phi(j) = \phi(j-1) + 1$ ✓

Dimostrazione Mario Avolio Edition.



Risoluzione del Caso (2)

Vogliamo sempre trovare $\phi(j)$ in funzione dei valori di ϕ precedenti.

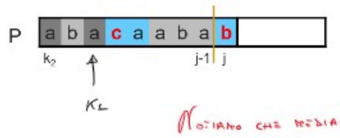
Supponiamo però che $P[K_1+1] \neq P[j]$

$\phi(j) = |\underbrace{B(P[1,j-1]\ P[j])}|$

Per definizione di Bordo ricorsivo.

$B(P[1,j-1]\ P[j]) = B(\underbrace{B(P[1,j-1])}_{P[1,K_1]}\ P[j])$

* Indichiamo con $K_2 = |B(P[1,K_1])|$



NOTIAMO CHE MEDIANTE GLI INDICI K_m STIAMO "SUDDIVIDENDO" IL PATTERN P.

* COME VISTO PRECEDENTEMENTE, POSSIAMO PORRE L'UGUAGLIANZA:

$$k_2 = |B(P[1,k_1])| = \phi(k_1) = \phi(\phi(j-1))$$

$$\Rightarrow B(B(P[1,j-1])\,P[j]) = B(\underbrace{P[1,k_1]}_{X}\ \underbrace{P[j]}_{\sigma})$$

SIAMO RICADUTI NEL CASO ANALOGO AL PRECEDENTE!

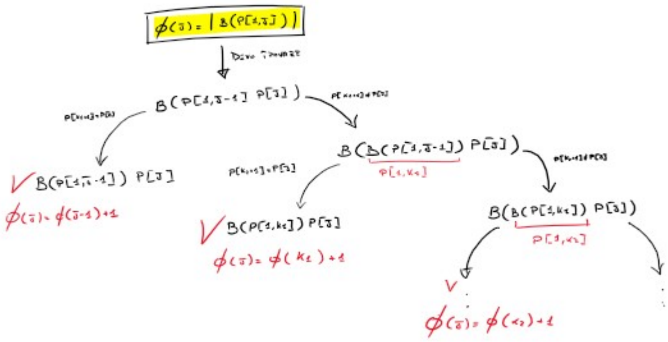
DOBBIAMO APPLICARE NUOVAMENTE LA DEFINIZIONE RICORSIVA DI BORDO.

$$B(P[1,k_1]\,P[j]) = \begin{cases} B(P[1,k_2])P[j] & P[k_2+1] = P[j] \ \text{AND} \ P[k_1+1] \neq P[j] \\ B(B(P[1,k_2])\,P[j]) & \text{ALTRIMENTI} \end{cases}$$

FAI ATTENZIONE! SIAMO IN UN SOTTOCASO DEL CASO 2

DIFATTI POTREMMO INDICARE I CASI SOPRASTANTI CON: 2.1 e 2.2

NOTIAMO UNA COSA: STIAMO ANDANDO A CREARE DEI SOTTOCASI OGNI QUAL VOLTA CHE $P[K_i+1] \neq P[J]$



QUANDO MI FERMO?

In generale:

- ✓ se si raggiunge un valore k_p tale per cui **$P[k_p+1] = P[j]$**
  $\varphi(j) = k_p + 1$
- ✓ se non si raggiunge mai un k_p tale per cui **$P[k_p+1] = P[j]$**, allora si raggiungerà a un certo punto un valore $k_p = -1$ che implica che il bordo di P[1,j] è nullo e quindi $\varphi(j)$ sarà uguale 0, cioè ancora calcolabile come k_p+1

ALGORITMO PSEUDOCODICE:

```
Procedura Compute-Prefix-Function(P)
 begin
    m ← |P|
    φ(0) ← -1
    φ(1) ← 0
    for j ← 2 to m do
      k ← φ(j-1)
      while k >= 0 and P[k+1] != P[j] then
         k ← φ(k)
      φ(j) ← k+1
    return φ
 end
```

ESEMPIO:

J=2

P	a	b	c	a	b	a

j	0	1	2	3	4	5	6
φ	-1	0	0	0	1	2	1

$k \leftarrow \varphi(2) = 0$

```
→ for j ← 2 to m do
      k ← φ(j-1)
      while k >= 0 and P[k+1] != P[j] then
         k ← φ(k)
      φ(j) ← k+1
```

PASSO 2 - SCANSIONE DEL TESTO IN O(n)

La scansione del testo T avviene tramite una finestra *W* di confronto lunga m che scorre lungo T da sinistra a destra, e inizialmente è posizionata in corrispondenza della posizione i=1 di T

Per ogni posizione i di *W*, ogni simbolo di P viene confrontato con il corrispondente simbolo del testo T in *W*, procedendo da sinistra a destra

Se tutti i simboli sono confrontati con successo, allora nella posizione i di T esiste un'occorrenza esatta di P

La finestra viene poi spostata verso destra in una nuova posizione e si ripete il confronto

T	a	b	c	b	b	a	a	b	c	a	b	c	a	b	c	a	a
P	a	b	c	a	b	b	c	a									

LA DIFFERENZA CON UNA NORMALISSIMA RICERCA NEL TESTO RISIEDE IN 2 DIFFERENTI OPERAZIONI CHE VENGONO ESEGUITE:

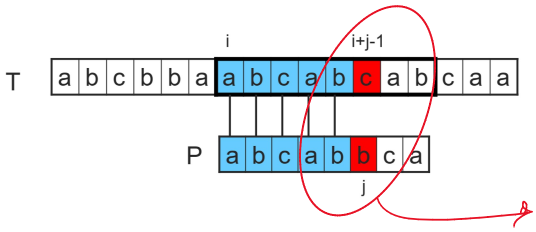
1. LA FINESTRA W VIENE SPOSTATA ANCHE SU PIÙ POSIZIONI UTILIZZANDO LA FUNZIONE φ

2. Il confronto riparte dal simbolo del testo che aveva dato mismatch con il pattern quando W era nella precededente posizione

ESEMPIO:

Si assume che:

- la finestra *W* si trova in posizione i su T
- il primo carattere di P che determina un *mismatch* con T è in posizione j > 1 → *mismatch* su T in posizione i+j-1



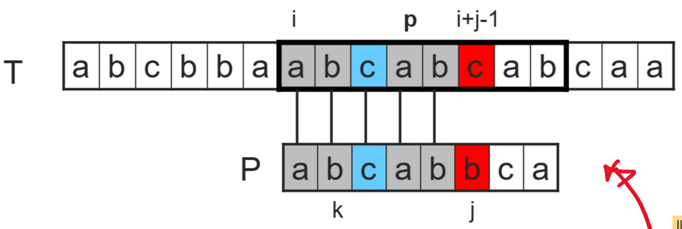
Dobbiamo spostare la finestra mediante l'utilizzo di ϕ

Per definizione, il valore k = φ(j-1) è la lunghezza del bordo del prefisso P[1,j-1]

→ P[1,k] ha un'occorrenza su T che inizia in posizione i

→ P[1,k] ha un'occorrenza su T che inizia in posizione

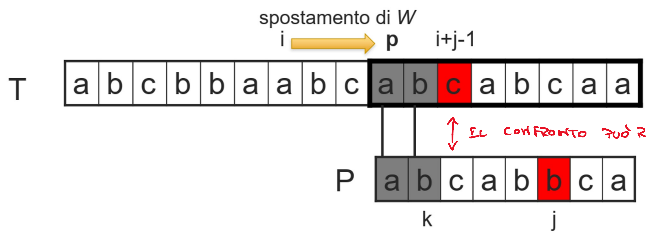
p = i + j - k - 1 = i + j - φ(j-1) - 1



Il salto da i a p garantisce:

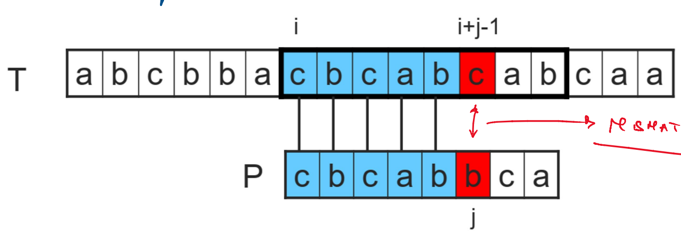
- di non perdere occorrenze di P
- che i primi k=φ(j-1) simboli di P hanno un *matching* con T a partire dalla nuova posizione p

Eseguo lo spostamento di *W*.



Il confronto può ripartire da qui.

Esempio – $\phi(j-1) = 0$:

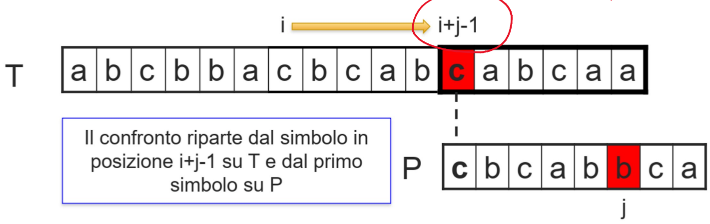


Mismatch!

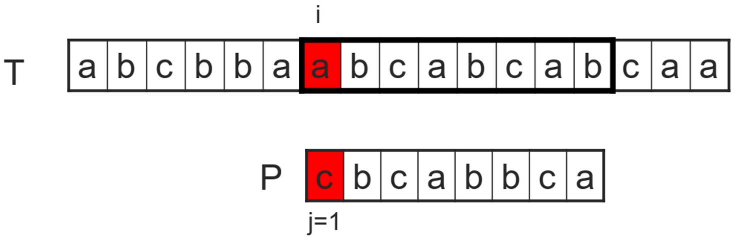
Come procediamo?

- *W* viene spostata alla posizione p = i + j - φ(j-1) - 1
- i primi φ(j-1) simboli di P non vengono più confrontati con i corrispondenti simboli su T

$\phi(j-1)=0$



Esempio 3 - j = 1 :

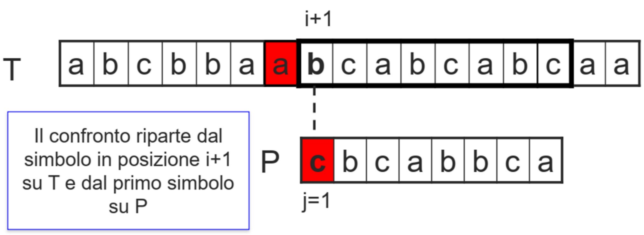


$P = i + j - \phi(j-1) - 1$

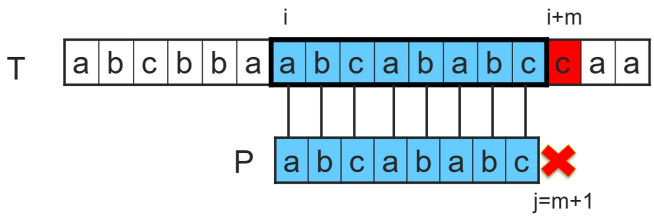
$= i + 1 - \phi(0) - 1$

$= i$ Non possiamo

Spostiamo in "i+1"

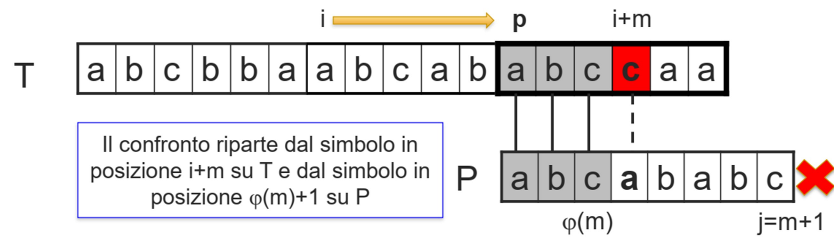


Esempio 4 - j = m+1



$P = i + (m+1) - \phi(m) - 1$

$= i + m - \phi(m)$



Algoritmo Pseudocodice :

```
KMP(P,T,φ)
 begin
     m ← |P|
     n ← |T|
     j ← 0
     for q ← 1 to n do
       while j >= 0 and P[j+1] != T[q] then
          j ← φ(j)
       j ← j +1
       if j = m then
          output q-m+1
end
```