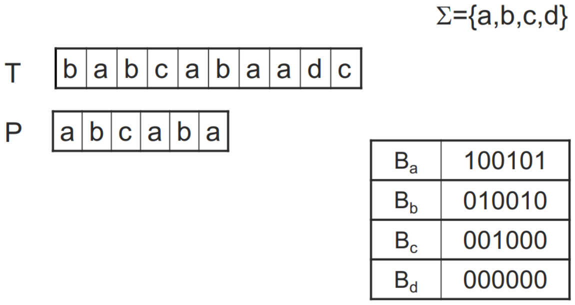
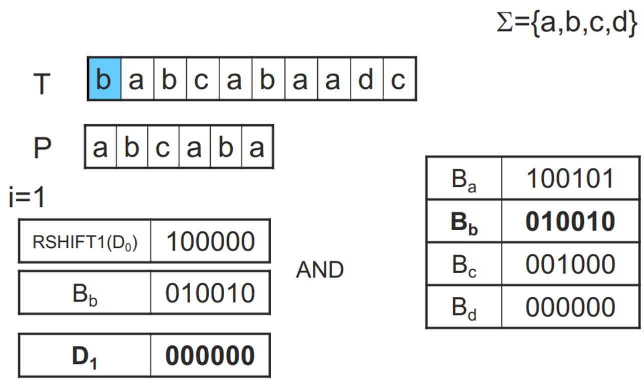
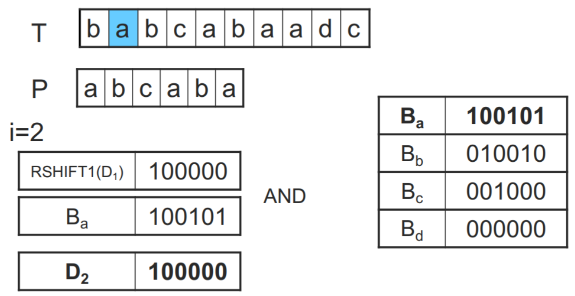
# Lezione 16 30/11/2023

## Esercizio BYG (che può capitare più corto in esame)

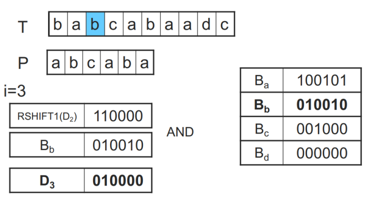


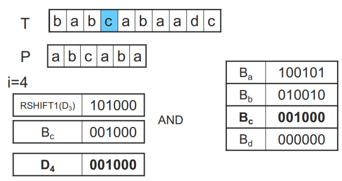



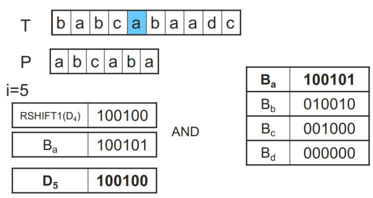

$D_0$ per definizione è tutti zeri.
Faccio partire la scansione, leggo il primo simbolo del testo. In questa posizione i=1 calcolo la parola: prendiamo la parola precedente, facciamo l'RSHIFT e la metto in AND logico alla riga di B relativa al simbolo letto.

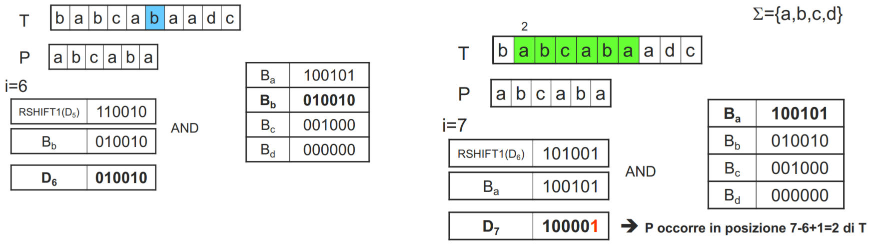



Leggo il secondo simbolo del testo, faccio un RSHIFT1 di $D_1$ e la metto in AND logico a $B_a$ perchè ho letto il simbolo a.

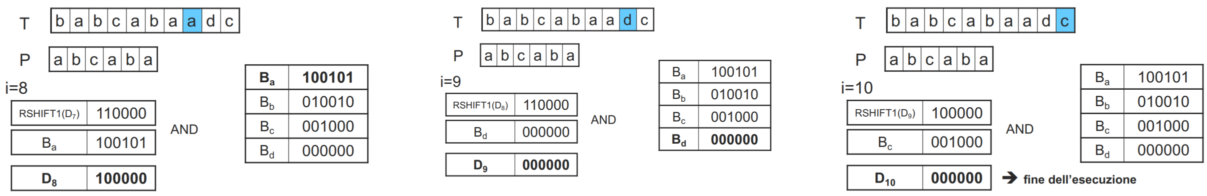




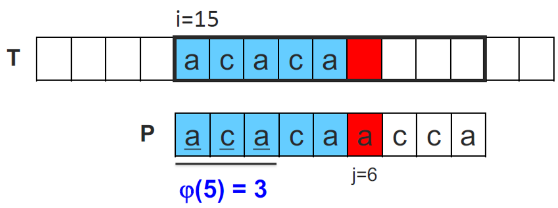

Per vedere dove inizia l'occorrenza esatta prendo la posizione 7, tolgo la lunghezza del pattern e aggiungo 1.




# Esercizi KMP

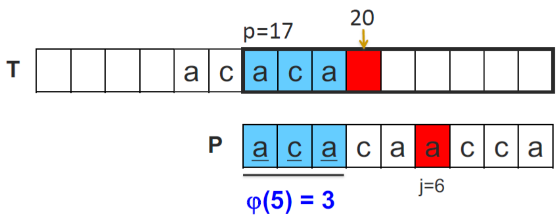
## Esercizio 1

A un certo punto dell'esecuzione dell'algoritmo KMP su P=acacaacca e un testo T, la finestra *W* si trova in posizione 15 e il primo simbolo di P che ha *mismatch* con T si trova in posizione 6. Calcolare la successiva posizione della finestra e la posizione dei simboli su T e su P da cui riparte il confronto.



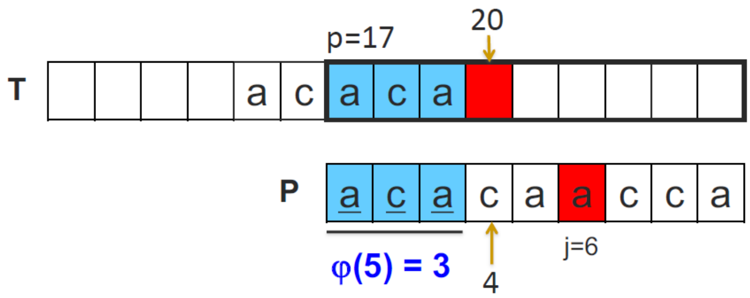

Nuova posizione di *W*

p = i + j - φ(j-1) - 1 = 15 + 6 - φ(5) -1 = 17




Posizione su T da cui riparte il confronto

i + j - 1 = 15 + 6 - 1 = 20




Posizione su P da cui riparte il confronto

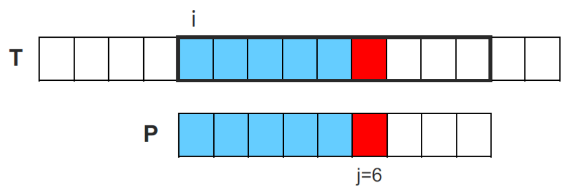
$$\varphi(j-1) + 1 = \varphi(5) + 1 = 4$$

## Esercizio 2

Il più lungo prefisso del pattern che occorre in posizione i del testo T è P[1,5]. Sapendo che la finestra *W*, durante l'esecuzione dell'algoritmo di KMP, compie un salto da i a i+3, determinare la lunghezza del bordo di P[1,5].

P[1,5], per una finestra che sta nella posizione i, è il prefisso di match. Quindi il mismatch è sul successivo.

La nuova posizione della finestra la calcoliamo come i (finestra corrente) + j (posizione di mismatch) - fi ... (valore funzione di fallimento) -1.




Nuova posizione di *W*

$$p = i + j - \varphi(j-1) - 1$$

Abbiamo j, non abbiamo fi perchè non avendo il pattern non abbiamo la funzione di fallimento. Però abbiamo p, quindi metto i+3 al posto di p.

Nuova posizione di *W*

$$i + 3 = i + 6 - \varphi(5) - 1$$

$$\Rightarrow \varphi(5) = 6 - 3 - 1 = 2$$

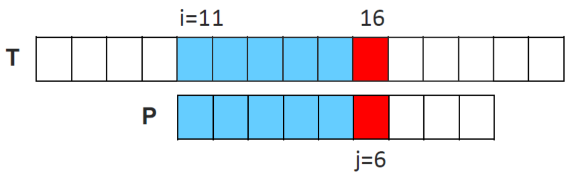
## Esercizio 3

Durante l'esecuzione dell'algoritmo di KMP la finestra *W* si trova in posizione 11 del testo. Dopo lo spostamento alla nuova posizione, il confronto riparte dal simbolo in posizione 16 del testo.
Individuare la lunghezza del prefisso di *match* del pattern per la posizione 11 di *W*.

La posizione 16 è la posizione del simbolo del testo che aveva dato mismatch con il pattern, in posizione 11. i=11.

$$16 = i + j - 1 \Rightarrow j = 16 - 11 + 1 = 6$$

j = posizione di *mismatch* sul pattern




j -1 = 5 → lunghezza del più lungo prefisso di P che occorre in i

j - 1 = 5 è la lunghezza del prefisso di match del pattern.

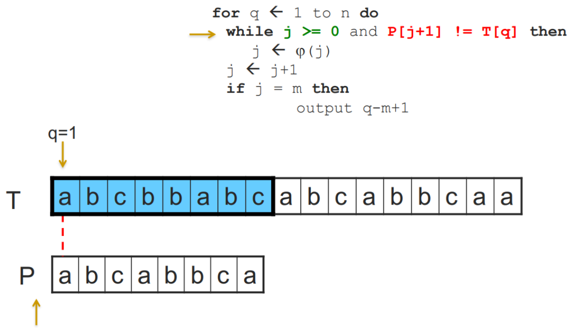
Gli esercizi in esame saranno simili, nei dati ci sarà tutto quello che bisogna inserire nelle formule.

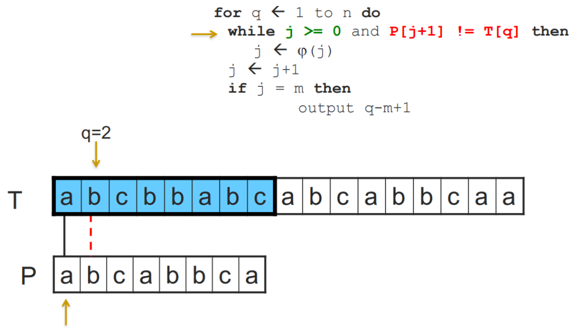
Non chiede esecuzioni di KMP o pseudocodice all'esame.

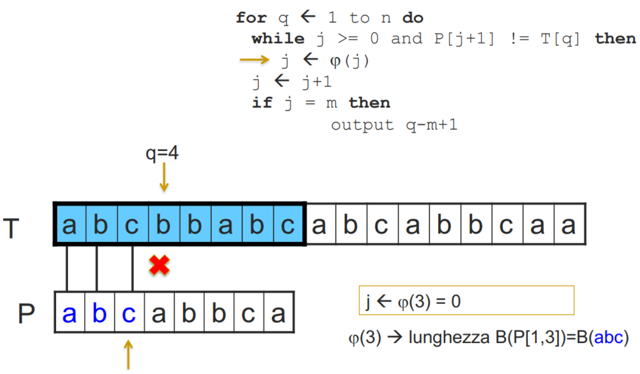
# Scansione del testo T con KMP (algoritmo)

```
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
```

Condizioni while false



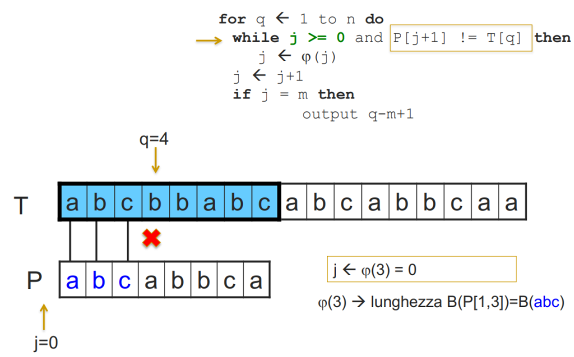



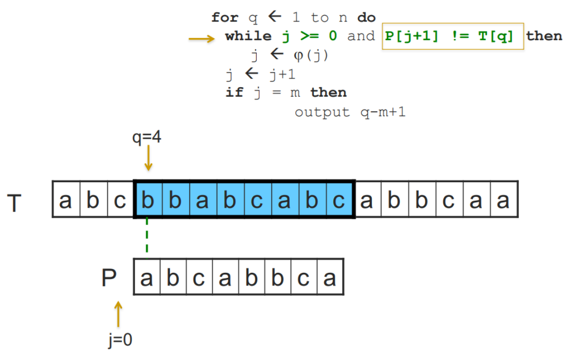




Aggiorna il valore di j tramite la funzione di fallimento phi(3).

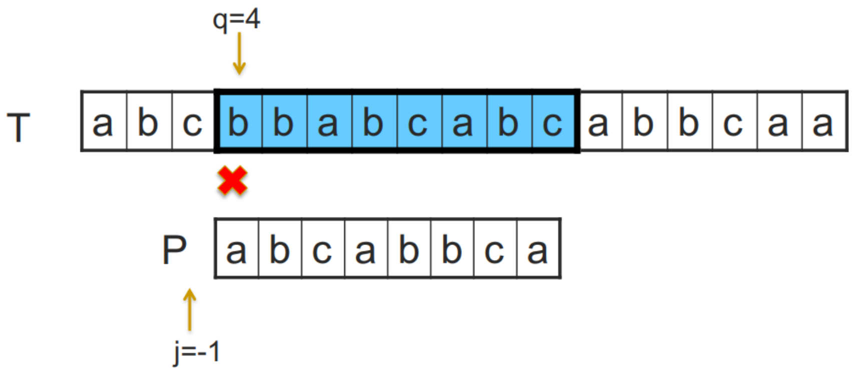
phi è la lunghezza del bordo del pattern da 1 a 3.

Torniamo a testare la condizione del while.




La condizione è vera quindi c'è un nuovo mismatch. Aggiorna quindi j al default -1




Mi sono perso, slides sul drive (KMP-2).

# Esercizi su BYG

## Esercizio 1

Una word $D_i$ dell'algoritmo BYG è uguale a 11111 e si riferisce al simbolo T[i]=**c** sul testo. Si chiede di specificare il pattern P.

$D_i$ = **<u>1</u>**1111

$D_i[1] = 1 \Rightarrow P[1,1] = suff(T[1,i]) \Rightarrow P[1] = T[i] = \mathbf{c}$

Il primo 1 ci dice che il prefisso lungo 1 del pattern occorre esattamente come suffisso del prefisso i del testo.

$D_i[2] = 1 \Rightarrow P[1,2] = suff(T[1,i]) \Rightarrow P[2] = T[i] = \mathbf{c}$

Il prefisso lungo 2 del pattern occorre esattamente come suffisso del prefisso i del testo. Quindi il secondo simbolo del pattern è T[i].

etc etc

## Esercizio 2

Una word $D_i$ dell'algoritmo di BYG è uguale a 01001. Si chiede di specificare la lunghezza del bordo del pattern.

Esaminiamo i bit a 1.

$D_i = 0100\underline{\mathbf{1}}$

$D_i[5] = 1 \Rightarrow P[1,5] = P = suff(T[1,i])$

Questo ci dice che il prefisso lungo 5 del pattern (tutto il pattern) occorre come suffisso del prefisso i del testo.

$D_i[2] = 1 \Rightarrow P[1,2] = suff(T[1,i]) \Rightarrow P[1,2] = suff(P)$

Mettendo insieme le due cose posso concludere che il prefisso da 1 a 2 è suffisso di P. (Il bordo è il più lungo prefisso che occorre come suffisso)

Per capire che è il più lungo vedo gli zeri:

$$D_i[3] = 0 \Rightarrow P[1,3] \neq suff(T[1,i]) \Rightarrow P[1,3] \neq suff(P)$$
$$D_i[4] = 0 \Rightarrow P[1,4] \neq suff(T[1,i]) \Rightarrow P[1,4] \neq suff(P)$$

$$\Rightarrow P[1,2] = B(P) \Rightarrow |B(P)| = 2$$

In generale, data una *word* $D_i$, la lunghezza del bordo di P è j se e solo se $D_i[m]=1$ e j è la più grande posizione < m tale che $D_i[j]=1$.

$D_i = 0110001100\underline{1}0000\boxed{1}$

j = 11 **|B(P)| = 11**

## Esercizio 3

Sia $D_7$ = 0000 una word dell'algoritmo di BYG per P=catg e un dato testo T. Sapendo che T[5,7]=atg, si può dire con certezza che in posizione 4 di T <u>non</u> c'è il simbolo c?

$D_7 = 0000$,     $T[5,7]$ = atg,     P = catg

La sottostringa di T di lunghezza 4, che finisce in posizione i=7, <u>non</u> ha *match* esatto con P

$\Rightarrow T[4] \neq c$

In caso contrario, si avrebbe:

$$T[4,7] = T[4]\,T[5,7] = catg = P$$

$$\Rightarrow D_7[4] = 1 \text{ contro l'ipotesi}$$

---

Sia $D_7 = 0000$ una *word* dell'algoritmo di BYG per P=catg e un dato testo T. ~~Sapendo che T[5,7] = atg,~~ si può dire con certezza che in posizione 4 di T <u>non</u> c'è il simbolo c?

Non si può dire con certezza che T[4] è diverso da c.

i=7

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| T | a | b | c | <u>c</u> | c | b | a | a | b | a | a | d | c |

| | | | | | | |
|---|---|---|---|---|---|---|
| P | c | a | t | g | m=4 | **$D_7$ = 0000** |

## Esercizio 4

Alla i-esima iterazione dell'algoritmo di BYG per cercare P = aabaa (di cui si conosce la tabella B), viene calcolata la *word* $D_i$ = 11000. Sapendo che la *word* $D_{i-1}$ è uguale a $D_i$, specificare il simbolo di T in posizione i.

$D_i = D_{i-1} = 11000$

Si deve trovare la *word* $B_\sigma$ tale per cui:

$$D_i = \textbf{RSHIFT1}(D_{i-1}) \textbf{ AND } B_\sigma$$

Si avrà $T[i] = \sigma$

$B_a = 11011$
$B_b = 00100$

Prova le due B della tabella una alla volta e vedi quali verifica l'equazione.

$D_i = D_{i-1} = 11000$

Si deve trovare la *word* $B_\sigma$ tale per cui:

$$11000 = 11100 \textbf{ AND } \mathbf{11011}$$

Si avrà $T[i] = \mathbf{a}$

$B_a = \mathbf{11011}$
$B_b = 00100$