

Sfoglia su PARADIGMA SHIFT-AND.

Si basa su 2 operazioni principali:

1. preprocessing del pattern P di lunghezza m in tempo  $O(|\Sigma|+m) \rightarrow$  calcolo di una tabella B di  $|\Sigma|$  parole di m bit
2. scansione del testo T di lunghezza n in tempo  $O(n)$  per la cercare le occorrenze esatte di P

$$\Sigma = \text{ALFABETO DEL TESTO E DEL PATTERN}$$

N<sub>sh</sub> funziona su SIMBOLI DEL PATTERN DEL TESTO

→ Non c'è confronto di simboli

↓  
PARADIGMA SHIFT-AND

COMPIE OPERAZIONI SU

PAROLE DI 3 BIT.

→ CHE SONO SOTTO  
STRIKENE DI  
BIT.

Operazioni che l'algoritmo userà:

### Operazioni bit a bit:

- 1 > congiunzione → AND
- 2 > disgiunzione inclusiva → OR
- 3 > shift dei bit di una posizione a destra con bit più significativo uguale a 0 → RSHIFT
- 4 > shift dei bit di una posizione a destra con bit più significativo uguale a 1 → RSHIFT1

① Congiunzione → AND

$w_1$  AND  $w_2$

restituisce una parola  $w$  tale che:

- $w[j] = 1 \Leftrightarrow w_1[j] = 1 \text{ e } w_2[j] = 1$
- $w[j] = 0$ , altrimenti

$w_1, w_2 = \text{Parole di Bit}$

1010 AND 0110 = 0010

NB: per comodità le posizioni dei bit vanno da sinistra a destra (cioè la posizione del bit più significativo è 1)

## 2) DISJUNZIONE INCLUSIVA $\rightarrow$ OR

$w_1$  OR  $w_2$

restituisce una parola  $w$  tale che:

- $w[j] = 1 \Leftrightarrow w_1[j] = 1 \text{ oppure } w_2[j] = 1$
- $w[j] = 0$ , altrimenti

1010 OR 0110 = 1110

## 3) RSHIFT

RSHIFT( $w_1$ )

restituisce una parola  $w$  tale che:

- $w[1] = 0$
- $w[j] = w_1[j-1]$ , se  $2 \leq j \leq |w|$

RSHIFT(1010) = 0101

## 4) RSHIFT1

RSHIFT1( $w_1$ ) = RSHIFT( $w_1$ ) OR 100...0

restituisce una parola  $w$  tale che:

- $w[1] = 1$
- $w[j] = w_1[j-1]$ , se  $2 \leq j \leq |w|$

RSHIFT  $\rightarrow$   $>>1$   
+  
OR  $\rightarrow$  |

E' UGUALE A RSHIFT  
MA SE PARLO BIT  
VIENE MODIFICATO SU  
1.

RSHIFT1( $w_1$ ) = RSHIFT( $w_1$ ) OR 100...0

restituisce una parola  $w$  tale che:

- $w[1] = 1$
- $w[j] = w_1[j-1]$ , se  $2 \leq j \leq |w|$

RSHIFT  $\rightarrow >>1$   
+  
OR  $\rightarrow |$

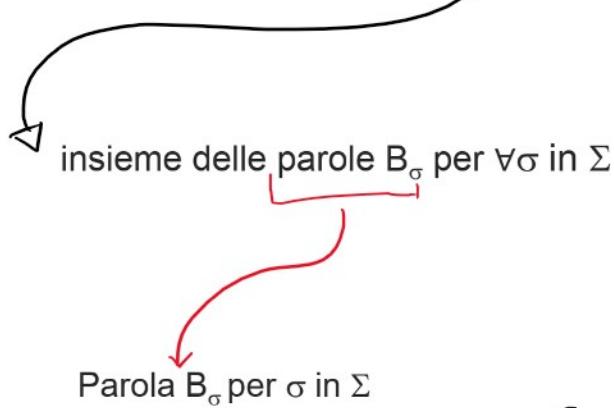
RSHIFT(1010) = 1101

E' UVALE A RSHIFT  
MA SE POKO SET  
VIETE MODIFICATO SU  
1.

## 1 Preprocessing

preprocessing del pattern  $P$  di lunghezza  $m$  in tempo  $O(|\Sigma| + m)$   $\rightarrow$  calcolo di una tabella  $B$  di  $|\Sigma|$  parole di  $m$  bit

SERVE A CORRERE  
SU PATTERN



Esempio:

$$B_\sigma = b_1 b_2 \dots b_m$$

$$P \quad \boxed{a \ b \ c \ a \ b \ a} \quad \Sigma = \{a, b, c, d\}, \ m=6$$

tale che:

$$B_b = 010010$$

$$b_j = 1 \Leftrightarrow P[j] = \sigma$$

ESEMPIO: PROVATE A CALCOLARE LA TABELLA B

$$P \quad \boxed{a \ b \ c \ a \ b \ a} \quad \Sigma = \{a, b, c, d\}, \ m=6$$

$B_a$	100101
$B_b$	010010
$B_c$	001000
$B_d$	000000

← Non compare nel pattern.

A COSA SERVE QUINDI QUESTA TABECCA?

→ SERVE NELLA SECONDA FASE DELL'ALGORITMO

AL FINE DI RISPONDERE ALLA DOMANDA

$$P_{[j]} = C ?$$

→ "In POSIZIONE  $\frac{j}{\sigma}$   
DEC PATTERN C'E'  
SE SIMBOLO C?"

↓  
RISPOSTA

DEVO CONTROLLARE  
SE BJT È-ESISTE PAROLA  
PAROLA Ba

$$P_{[j]} = C \Leftrightarrow B_G[j] = 1$$

Come viene creata la TABELLA  $B$ ?

STEP1: tutte le parole  $B_\sigma$  vengono inizializzate a m zeri

STEP2: viene inizializzata una parola M (maschera) di m bit tutti uguali a 0 tranne il più significativo che è uguale a 1

STEP3: si esegue una scansione di P da sinistra verso destra, e per ogni posizione j:

$$1) B_{P[j]} = (M \text{ OR } B_{P[j]})$$

$$2) M = \text{RSHIFT}(M)$$

ESEMPIO:

① STEP1 → Inizializzazione delle parole  $B_\sigma$

D [a b c a b c] n=6 abcdef m=6

1

STEP1 → Inizializzazione delle parole  $B_\sigma$

P 

a	b	c	a	b	a
---	---	---	---	---	---

 $\Sigma = \{a, b, c, d\}$ ,  $m=6$

$B_a$	000000
$B_b$	000000
$B_c$	000000
$B_d$	000000

2

STEP2 → Inizializzazione della maschera M

P 

a	b	c	a	b	a
---	---	---	---	---	---

 $\Sigma = \{a, b, c, d\}$ ,  $m=6$

$B_a$	000000	M=100000
$B_b$	000000	
$B_c$	000000	
$B_d$	000000	

3

STEP3 → scansione di P per j da 1 a 6

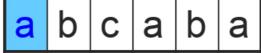
P 

a	b	c	a	b	a
---	---	---	---	---	---

 $\Sigma = \{a, b, c, d\}$ ,  $m=6$

j=1

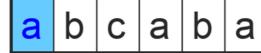
$B_a$	000000	M=100000
$B_b$	000000	$B_a = 000000$
$B_c$	000000	OR
$B_d$	000000	

P   $\Sigma = \{a, b, c, d\}$ ,  $m=6$

j=1

$B_a$	100000
$B_b$	000000
$B_c$	000000
$B_d$	000000

M=100000  
 $B_a = 000000$  OR  
 $B_a = 100000$

P   $\Sigma = \{a, b, c, d\}$ ,  $m=6$

j=1

$B_a$	100000
$B_b$	000000
$B_c$	000000
$B_d$	000000

RSHIFT  
M=100000

P   $\Sigma = \{a, b, c, d\}$ ,  $m=6$

j=1

$B_a$	100000
$B_b$	000000
$B_c$	000000
$B_d$	000000

RSHIFT  
M=010000

P   $\Sigma = \{a, b, c, d\}$ ,  $m=6$

j=2

$B_a$	100000
$B_b$	000000
$B_c$	000000
$B_d$	000000

M=010000

P [a | b | c | a | b | a]  $\Sigma = \{a, b, c, d\}$ , m=6

j=2

$B_a$	100000	M=010000
$B_b$	000000	$B_b = 000000$ OR
$B_c$	000000	
$B_d$	000000	

P [a | b | c | a | b | a]  $\Sigma = \{a, b, c, d\}$ , m=6

j=2

$B_a$	100000	M=010000
$B_b$	010000	$B_b = 000000$ OR
$B_c$	000000	
$B_d$	000000	

$B_b = 010000$

j=2

$B_a$	100000	M=010000
$B_b$	010000	RSHIFT
$B_c$	000000	
$B_d$	000000	

$M=010000$

j=2

$B_a$	100000	M=001000
$B_b$	010000	RSHIFT
$B_c$	000000	
$B_d$	000000	

$M=001000$

P [a | b | c | a | b | a]  $\Sigma = \{a, b, c, d\}$ , m=6

j=3

$B_a$	100000	M=001000
$B_b$	010000	
$B_c$	000000	
$B_d$	000000	

P [a | b | c | a | b | a]  $\Sigma = \{a, b, c, d\}$ , m=6

j=3

$B_a$	100000	M=001000
$B_b$	010000	
$B_c$	000000	$B_c = 000000$ OR
$B_d$	000000	

P [a | b | c | a | b | a]  $\Sigma = \{a, b, c, d\}$ , m=6

j=3

$B_a$	100000	M=001000
$B_b$	010000	
$B_c$	001000	$B_c = 000000$ OR
$B_d$	000000	$B_c = 001000$

P [a | b | c | a | b | a]  $\Sigma = \{a, b, c, d\}$ , m=6

j=3

$B_a$	100000	 RSHIFT M=001000
$B_b$	010000	
$B_c$	001000	
$B_d$	000000	

P 

a	b	c	a	b	a
---	---	---	---	---	---

 $\Sigma = \{a, b, c, d\}$ ,  $m = 6$

j=3

$B_a$	100000
$B_b$	010000
$B_c$	001000
$B_d$	000000

RSHIFT 

M=000100

P 

a	b	c	a	b	a
---	---	---	---	---	---

 $\Sigma = \{a, b, c, d\}$ ,  $m = 6$

j=4

$B_a$	100000
$B_b$	010000
$B_c$	001000
$B_d$	000000

M=000100

P 

a	b	c	a	b	a
---	---	---	---	---	---

 $\Sigma = \{a, b, c, d\}$ ,  $m = 6$

j=4

$B_a$	100000
$B_b$	010000
$B_c$	001000
$B_d$	000000

M=000100  
 $B_a = 100000$  OR

P 

a	b	c	a	b	a
---	---	---	---	---	---

 $\Sigma = \{a, b, c, d\}$ ,  $m = 6$

j=4

$B_a$	100100
$B_b$	010000
$B_c$	001000
$B_d$	000000

M=000100  
 $B_a = 100000$  OR  
 $B_a = 100100$

P [a | b | c | **a** | b | a]       $\Sigma = \{a, b, c, d\}$ , m=6

j=4

$B_a$	100100
$B_b$	010000
$B_c$	001000
$B_d$	000000

RSHIFT

M=000100

⋮

Accia finge si arriva a:

P [a | b | c | a | b | a]       $\Sigma = \{a, b, c, d\}$ , m=6

j=6

$B_a$	100101
$B_b$	010010
$B_c$	001000
$B_d$	000000

A corretto!

```

Procedura Compute-table-B(P)
begin
    m  $\leftarrow$  |P|
    foreach  $\sigma$  in  $\Sigma$  do
         $B_\sigma \leftarrow 00\dots0$ 
        M  $\leftarrow 10\dots0$ 
        for j  $\leftarrow 1$  to m do
             $\sigma \leftarrow P[j]$ 
             $B_\sigma \leftarrow M \text{ OR } B_\sigma$ 
            M = RSHIFT(M)
    return the table B of the words  $B_\sigma$ 
end

```

Complessità  $\rightarrow O(|\Sigma|+m)$

2

## Scansione del Testo

scansione del testo  $T$  di lunghezza  $n$  in tempo  $O(n)$  per la cercare le occorrenze esatte di  $P$

### Passi della scansione:

1. Il testo  $T$  viene scandito dalla prima all'ultima posizione
2. Per ogni posizione  $i$  del testo  $T$  viene calcolata una parola  $D_i$  di  $m$  bit
3. Ogni volta che  $D_i$  ha il bit meno significativo uguale a 1, viene prodotta in output l'occorrenza esatta  $i-m+1$

→ Ci formisce le occorrenze del pattern del testo.

### Parola $D_i$

Si denoti con:

$$P[1,j] = \text{suff}(T[1,i])$$

il fatto che:

$P[1,j]$  occorre esattamente come suffisso di  $T[1,i]$

T	a	b	c	b	a	b	c	a	b	a	d	c
												$i=7$

P	a	b	c	a	b	a	
							$j=3$

$$P[1,3] = \text{suff}(T[1,7])$$

### DEFINIZIONE DI PAROLA $D_i$ ( $0 \leq i \leq n$ )

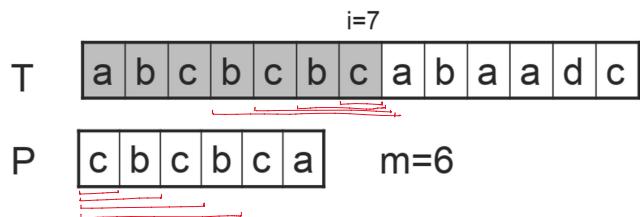
$$D_i = d_1 d_2 \dots d_m$$

tale che:

$$d_j = D_i[j] = 1 \Leftrightarrow P[1,j] = \text{suff}(T[1,i])$$

ESEMPIO:

$$D_7 = 101010$$



In particolare, la parola  $D_0$  è (per definizione):

$$D_0 = 00\dots 0 \quad (\forall j, P[1,j] \neq \text{suff}(T[1,0]))$$

Inoltre:

$$d_m = D_i[m] = 1 \Leftrightarrow P[1,m] = \text{suff}(T[1,i])$$

↓ ovvero

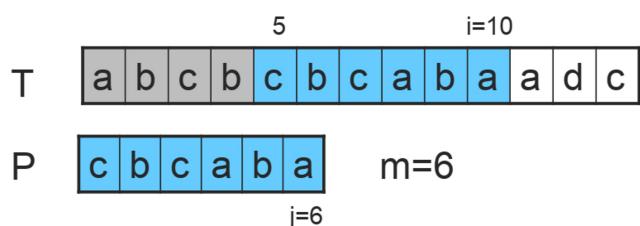
P ha un'occorrenza che finisce in i

↓ ovvero

P ha un'occorrenza che inizia in  $i-m+1$

ESEMPIO:

$$D_{10} = 000001 \rightarrow P \text{ occorre in } i-m+1 = 5$$



Che cosa sono le scattate del Testo?

- 1 > inizia dalla parola  $D_0 = 00\dots0$
- 2 > per ogni  $i$  da 1 a  $n$  calcola la parola  $D_i$  a partire dalla parola  $D_{i-1}$  (calcolata per  $i-1$ )
- 3 > ogni volta che  $D_i$  ha il bit più a destra uguale a 1, viene prodotta in output la posizione  $i-m+1$  dell'inizio di un'occorrenza di  $P$  su  $T$

Passo 2 - Come calcolare  $D_i$  partendo da  $D_{i-1}$

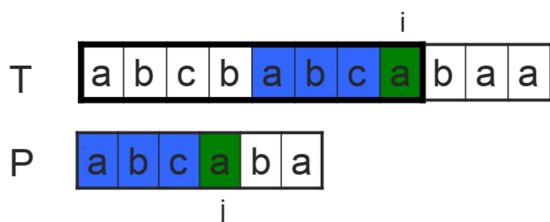
Dividiamo se calcolo in 2 parti:

2.1.  $\overline{d} > \underline{d}$

2.2.  $\overline{d} = \underline{d}$

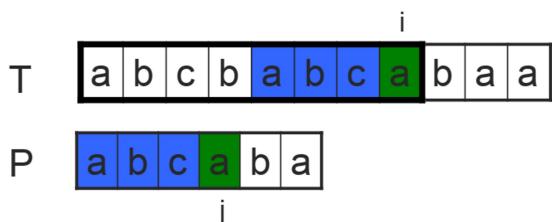
Passo 2.1 -  $\overline{d} > \underline{d}$

$$\begin{aligned} D_i[j] = 1 &\Leftrightarrow P[1,j] = \text{suff}(T[1,i]) \\ &\Leftrightarrow P[1,j-1] = \text{suff}(T[1,i-1]) \text{ AND } P[j] = T[i] \\ &\quad \underbrace{\qquad\qquad\qquad}_{D_{i-1}[j-1] = 1} \end{aligned}$$



$\triangleright_A$  Osservazione:  $P[j] = T[i] \Leftrightarrow B_{T[i]}[j] = 1$

$$D_i[j] = 1 \Leftrightarrow D_{i-1}[j-1] = 1 \text{ AND } P[j] = T[i]$$

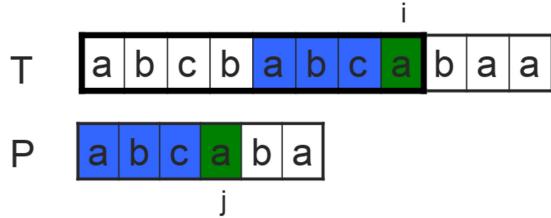


∴ arrivo alla conclusione:

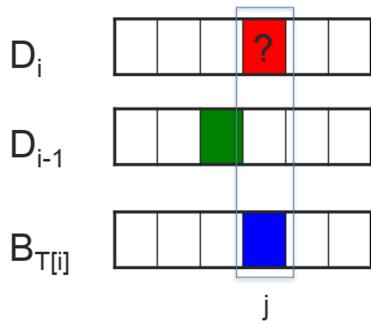
## S. Esegue alla conclusione:

$$D_i[j] = D_{i-1}[j-1] \text{ AND } B_{T[i]}[j]$$

Posso a questo punto sostituire il simbolo  $\Leftrightarrow$  di doppia implicazione con il simbolo di uguale =



$$D_i[j] = D_{i-1}[j-1] \text{ AND } B_{T[i]}[j]$$



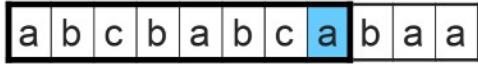
Per  $j > 1$ , il bit in posizione  $i$  della parola  $D_i$  (bit **rosso**) è dato dall'AND logico tra il bit in posizione  $i-1$  della parola  $D_{i-1}$  (bit **verde**) con il bit in posizione  $i$  della parola  $B_{T[i]}$  (bit **blu**)

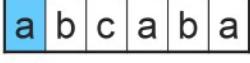
**CONCLUSIONE:** per  $j > 1$ , il  $j$ -esimo bit della parola  $D_i$  è uguale all'AND logico tra il  $(j-1)$ -esimo bit della parola  $D_{i-1}$  e il  $j$ -esimo bit della parola  $B_{\sigma}$  in corrispondenza del simbolo  $\sigma = T[i]$

Passo 2.2 -  $j=1$

$$D_i[1] = 1 \Leftrightarrow P[1,1] = \text{suff}(T[1,i]) \Leftrightarrow P[1] = T[i]$$

$$P[1] = T[i] \Leftrightarrow B_{T[i]}[1] = 1$$

T 

P   
j=1

$\Delta$  Così deduciamo che:

$$\underbrace{D_i[1] = 1 \Leftrightarrow B_{T[i]}[1] = 1}_{\text{Ovvvero: } D_i[1] = 1 \text{ AND } B_{T[i]}[1]}$$

$$D_i[1] = 1 \text{ AND } B_{T[i]}[1]$$

$D_i$	?				
$D_{i-1}$	1				
$B_{T[i]}$					

j=1

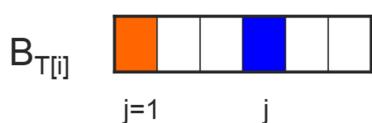
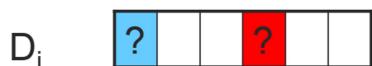
Per  $j = 1$ , il bit in posizione 1 della parola  $D_i$  (bit azzurro) è dato dall'AND logico tra il bit 1 con il bit in posizione 1 della parola  $B_{T[i]}$  (bit arancio)

**CONCLUSIONE:** il primo bit della parola  $D_i$  è uguale all'AND logico tra 1 e il primo bit della parola  $B_\sigma$  in corrispondenza del simbolo  $\sigma = T[i]$

CASE WHERE  $j = 2$  CASE -  $j \geq 1$

$$D_i[j] = D_{i-1}[j-1] \text{ AND } B_{T[i]}[j]$$

$$D_i[1] = 1 \text{ AND } B_{T[i]}[1]$$

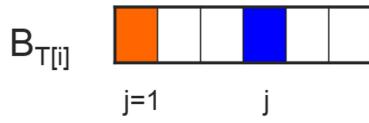


CASE POSSIBILE  
AVERE 3 bit?

SHIFTING RSHIFT1



RSHIFT1( $D_{i-1}$ )



$j=1 \quad j$

$\rightarrow$   $D_i$  COSE DERIVANTE:

$$D_i[j] = \text{RSHIFT1}(D_{i-1})[j] \text{ AND } B_{T[i]}[j]$$

$j > 1$

$$D_i[1] = \text{RSHIFT1}(D_{i-1})[1] \text{ AND } B_{T[i]}[1]$$

$j = 1$

$$D_i[j] = \text{RSHIFT1}(D_{i-1})[j] \text{ AND } B_{T[i]}[j] \quad j \geq 1$$

GENERALIZZATA

$$D_i = \text{RSHIFT1}(D_{i-1}) \text{ AND } B_{T[i]}$$

## Accorciato Scansione del Testo

- inizializza una maschera  $M = 00\dots 1$  di  $m$  bit tutti uguali a 0 tranne quello più a destra che è uguale a 1
- inizializza la parola  $D_0 = 00\dots 0$
- per  $i$  compreso tra 1 e  $n$ , calcola la parola  $D_i$   
$$D_i = \text{RSHIFT1}(D_{i-1}) \text{ AND } B_{T[i]}$$
- ogni volta che la parola  $(D_i \text{ AND } M)$  è uguale a  $00\dots 01$ , viene prodotta in output l'occorrenza esatta  $i-m+1$

**Procedura find-occurrences ( $B, T$ )**

```
begin
    n ← |T|
    D ← 00...00
    M ← 00...01
    for i ← 1 to n do
        σ ← T[i]
        D ← RSHIFT1(D) AND Bσ
        if (D AND M) = 00...01 then
            output i-m+1 //Occorrenza di P in T
end
```

Complexità  
 $O(m)$