Lezione 20 15/12/2023

Q-intervallo

Q → stringa definita su Σ

DEFINIZIONE1 (rispetto alla BWT)

Intervallo [b,e) di posizioni della BWT che contengono i simboli che precedono i suffissi che condividono la stringa Q come prefisso

Sigma qui è l'alfabeto originale del testo, quindi senza il dollaro.

b è la posizione di inizio ed *e* è la posizione di fine, dove *e* è compresa, mentre *e* è esclusa.

DEFINIZIONE2 (rispetto al Suffix Array)

Intervallo [b,e) di posizioni sul SA che contengono gli indici dei suffissi che condividono la stringa Q come prefisso

Q-intervallo per Q=ε → [1,n+1)

Se la stirnga Q è la stringa vuota, l'epsilon-intervallo è quello che comprende tutta la BWT.

Esempio

$\Sigma = \{\$, a, c, g, t\}$

T = acaaacatat$

	S		B
1	11	$	t
2	3	aaacatat$	c
3	4	aacatat$	a
4	1	acaaacatat	$
5	5	acatat$	a
6	9	at$	t
7	7	atat$	c
8	2	caaacatat$	a
9	6	catat$	a
10	10	t$	a
11	8	tat$	a

[4,6) → aca-intervallo = ac-intervallo

	S		B
1	11	$	t
2	3	aaacatat$	c
3	4	aacatat$	a
4	1	acaaacatat	$
5	5	acatat$	a
6	9	at$	t
7	7	atat$	c
8	2	caaacatat$	a
9	6	catat$	a
10	10	t$	a
11	8	tat$	a

[2,8) → a-intervallo

	S		B
1	11	$	t
2	3	aaacatat$	c
3	4	aacatat$	a
4	1	acaaacatat	$
5	5	acatat$	a
6	9	at$	t
7	7	atat$	c
8	2	caaacatat$	a
9	6	catat$	a
10	10	t$	a
11	8	tat$	a

[9,10) → cat-intervallo

	S		B
1	11	$	t
2	3	aaacatat$	c
3	4	aacatat$	a
4	1	acaaacatat	$
5	5	acatat$	a
6	9	at$	t
7	7	atat$	c
8	2	caaacatat$	a
9	6	catat$	a
10	10	t$	a
11	8	tat$	a

[1,12) → ε-intervallo

Esempio

In un Q-intervallo [b,e):

- ✓ S[b], S[b+1], …, S[e-1]
  → indici dei suffissi che hanno Q come prefisso
- ✓ B[b], B[b+1], …, B[e-1]
  → simboli che precedono i suffissi che hanno Q come prefisso

In un Q-intervallo [b,e):

- ✓ S[b], S[b+1], …, S[e-1]
  → occorrenze esatte di Q nel testo T
- ✓ B[b], B[b+1], …, B[e-1]
  → simboli che precedono le occorrenze esatte di Q nel testo T

Quindi un q-intervallo rappresenta sempre le posizioni delle occorrenze esatte di Q nel testo.

Numero di occorrenze di Q in T → (e-b)

Nell'intervallo [b,e]

Esempio

$\Sigma = \{\$, a, c, g, t\}$

T = acaaacatat$

	S		B
1	11		t
2	3		c
3	4		a
4	**1**		**$**
5	**5**		**a**
6	9		t
7	7		c
8	2		a
9	6		a
10	10		a
11	8		a

[4,6) → aca-intervallo

Le occorrenze di aca in T:

- sono in numero pari a 6-4=2
- iniziano nelle posizioni: 1, 5
- sono precedute dai simboli: $, a

Backward extension

La *backward extension* di un Q-intervallo [b,e) con un simbolo σ è il σQ-intervallo, cioé l'intervallo relativo alla stringa ottenuta concatenando il simbolo σ con Q

Prende in input un Q-intervallo e un simbolo, il risultato dell'operazione è il sigmaQ-intervallo, estendendo a sinistra la stringa aggiungendo sigma.

Esempio

	S		B
1	11	$	t
2	3	aaacatat$	c
3	4	aacatat$	a
4	1	acaaacatat	$
5	5	acatat$	a
6	9	at$	t
7	7	atat$	c
8	2	caaacatat$	a
9	6	catat$	a
10	10	t$	a
11	8	tat$	a

[2,8) → a-intervallo

backward extension con c

[8,10) → ca-intervallo

	S		B
1	11	$	t
2	3	aaacatat$	c
3	4	aacatat$	a
4	1	acaaacatat	$
5	5	acatat$	a
6	9	at$	t
7	7	atat$	c
8	2	caaacatat$	a
9	6	catat$	a
10	10	t$	a
11	8	tat$	a

	S		B
1	11	$	t
2	3	aaacatat$	c
3	4	aacatat$	a
4	1	acaaacatat	$
5	5	acatat$	a
6	9	at$	t
7	7	atat$	c
8	2	caaacatat$	a
9	6	catat$	a
10	10	t$	a
11	8	tat$	a

[2,8) → a-intervallo

backward extension con g

intervallo vuoto (non esiste il ga-intervallo)

Ricerca esatta con BWT

Ricerca esatta di un pattern P lungo m

1. Si parte dal suffisso vuoto ε del pattern P
2. Si considera il Q-intervallo per Q=ε, cioè [1,n+1), e si inizializza un indice di posizione i=m
3. Si effettua una *backward extension* con il simbolo P[i] per ottenere il P[i]Q-intervallo [b_p, e_p) che fornisce le occorrenze del suffisso P[i,m]:
   a) se il risultato è l'intervallo vuoto, allora P non ha occorrenze in T e la ricerca si ferma.
   b) se il risultato non è l'intervallo vuoto, allora si decrementa di uno la posizione i e si ripete il punto 2 per l'intervallo ottenuto

2. m è l'ultimo simbolo del pattern

Alla prima iterazione fornirà le occorrenze dell'ultimo simbolo del pattern.

Devo distinguere il risultato che ottengo, se il risultato è vuoto allora mi fermo, perché non ci sono occorrenze.

Altrimenti si continua ad estendere all'indietro l'intervallo, fino a quando si raggiunge il primo simbolo del pattern.

b) se il risultato non è l'intervallo vuoto

i. se i è maggiore di 1, allora si decrementa di uno la posizione i e si ripete il punto 2 per l'intervallo ottenuto

ii. se i è uguale a 1, significa che è stato trovato il Q-intervallo $[b_P, e_P)$ per Q=P e vengono prodotte in output le occorrenze esatte $S[b_P]$, $S[b_P + 1]$, …, $S[e_P-1]$

se i>1, continuo ad estendere.

se i=1 vuol dire che sono arrivato al primo simbolo del pattern con un intervallo non vuoto, quindi ho trovato le occorrenze del pattern nel testo.

Se P occorre in T → esattamente m iterazioni

✓ **Prima iterazione**: *backward extension* dell'ε-intervallo con il simbolo P[m]

→ P[m]-intervallo

✓ **Iterazione intermedia**: *backward extension* del P[i+1,m]-intervallo con il simbolo P[i]

→ P[i,m]-intervallo

✓ **Ultima iterazione (m-esima)**: *backward extension* del P[2,m]-intervallo con il simbolo P[1]

→ P-intervallo (non vuoto)

Se P non occorre in T → numero di iterazioni ≤ m

✓ **Ultima iterazione**: *backward extension* del P[p+1,m]-intervallo con il simbolo P[p]

→ P[p,m]-intervallo vuoto

Massimo suffisso che occorre in T → P[p+1,m]

Numero di iterazioni = |P[p+1,m]| + 1

Esempio

T = acaaacatat$

P = acat

	S		B
1	11	$	t
2	3	aaacatat$	c
3	4	aacatat$	a
4	1	acaaacatat	$
5	5	acatat$	a
6	9	at$	t
7	7	atat$	c
8	2	caaacatat$	a
9	6	catat$	a
10	10	t$	a
11	8	tat$	a

	S		B
1	11	$	t
2	3	aaacatat$	c
3	4	aacatat$	a
4	1	acaaacatat	$
5	5	acatat$	a
6	9	at$	t
7	7	atat$	c
8	2	caaacatat$	a
9	6	catat$	a
10	10	t$	a
11	8	tat$	a

P = aca**t**

[b,e) ← ε-intervallo [1,12)

Iterazione 1

backward extension di [1,12) con **P[4]=t**

→ **t**-intervallo

	S		B
1	11	$	t
2	3	aaacatat$	c
3	4	aacatat$	a
4	1	acaaacatat	$
5	5	acatat$	a
6	9	at$	t
7	7	atat$	c
8	2	caaacatat$	a
9	6	catat$	a
10	10	t$	a
11	8	tat$	a

P = acat

[b,e) ← ε-intervallo [1,12)

Iterazione 1
backward extension di [1,12) con **P[4]=t**
→ **t**-intervallo → [10,12)

Iterazione 2
backward extension di [10,12) con **P[3]=a**
→ **at**-intervallo → [6,8)

	S		B
1	11	$	t
2	3	aaacatat$	c
3	4	aacatat$	a
4	1	acaaacatat	$
5	5	acatat$	a
6	9	at$	t
7	7	atat$	c
8	2	caaacatat$	a
9	6	catat$	a
10	10	t$	a
11	8	tat$	a

P = acat

[b,e) ← ε-intervallo [1,12)

Iterazione 1
backward extension di [1,12) con **P[4]=t**
→ **t**-intervallo → [10,12)

Iterazione 2
backward extension di [10,12) con **P[3]=a**
→ **at**-intervallo → [6,8)

Iterazione 3
backward extension di [6,8) con **P[2]=c**
→ **cat**-intervallo → [9,10)

	S		B
1	11	$	t
2	3	aaacatat$	c
3	4	aacatat$	a
4	1	acaaacatat	$
5	5	acatat$	a
6	9	at$	t
7	7	atat$	c
8	2	caaacatat$	a
9	6	catat$	a
10	10	t$	a
11	8	tat$	a

P = acat

[b,e) ← ε-intervallo [1,12)

Iterazione 1
backward extension di [1,12) con **P[4]=t**
→ **t**-intervallo → [10,12)

Iterazione 2
backward extension di [10,12) con **P[3]=a**
→ **at**-intervallo → [6,8)

Iterazione 3
backward extension di [6,8) con **P[2]=c**
→ **cat**-intervallo → [9,10)

Iterazione 4
backward extension di [9,10) con **P[1]=a**
→ **acat**-intervallo → [5,6)

	S	F	B
1	11	$	t
2	3	aaacatat$	c
3	4	aacatat$	a
4	1	acaaacatat	$
5	5	acatat$	a
6	9	at$	t
7	7	atat$	c
8	2	caaacatat$	a
9	6	catat$	a
10	10	t$	a
11	8	tat$	a

Il P-intervallo è [5,6)
→ 5 è l'unica occorrenza di P in T

Esempio

	S	F	B
1	11	$	t
2	3	aaacatat$	c
3	4	aacatat$	a
4	1	acaaacatat	$
5	5	acatat$	a
6	9	at$	t
7	7	atat$	c
8	2	caaacatat$	a
9	6	catat$	a
10	10	t$	a
11	8	tat$	a

P = agat

[b,e) ← ε-intervallo [1,12)

Iterazione 1

backward extension di [1,12) con **P[4]=t**

→ **t**-intervallo → [10,12)

Iterazione 2

backward extension di [10,12) con **P[3]=a**

→ **at**-intervallo → [6,8)

Iterazione 3

backward extension di [6,8) con **P[2]=~~c~~** g

→ **~~c~~at**-intervallo → ~~[9,10)~~ g

~~Iterazione 4~~

~~backward extension di [6,8) con **P[1]=a**~~

~~→ **acat**-intervallo → [5,6)~~

Calcolo della backward extension

QUESTIONE: come trovare, dato un Q-intervallo [b,e) e un simbolo σ, il σQ-intervallo?

	S	F	B
1	11	$	t
2	3	a	c
3	4	a	a
4	1	a	$
5	5	a	a
6	9	a	t
7	7	a	c
8	2	c	a
9	6	c	a
10	10	t	a
11	8	t	a

QUESTIONE: come trovare, dato il Q-intervallo [2,8) per Q=**a** e il simbolo **c**, il **c**Q-intervallo?

i_1=**2** → B[i_1]=**c** precede il **3**-suffisso

i_2=**7** → B[i_2]=**c** precede il **7**-suffisso

il **3**-suffisso ha Q come prefisso

il **7**-suffisso ha Q come prefisso

B[i_1] + **3**-suffisso → suffisso che ha **c**Q come prefisso

B[i_2] + **7**-suffisso → suffisso che ha **c**Q come prefisso

j_1 = LF(i_1=**2**) = **8**

j_2 = LF(i_2=**7**) = **9**

7	7	a	c
8	2	c	a
9	6	c	a
10	10	t	a
11	8	t	a

B[i_1] + **3**-suffisso → suffisso che ha **c**Q come prefisso

B[i_2] + **7**-suffisso → suffisso che ha **c**Q come prefisso

j_1 = LF(i_1=**2**) = **8**

j_2 = LF(i_2=**7**) = **9**

[8,10) è il **c**Q-intervallo

Q-intervallo [b,e) e simbolo σ:

- ✓ siano $i_1 < i_2 < \dots < i_k$ le k posizioni in [b,e) tali che:

  $B[i_p] = \sigma \qquad 1 \le p \le k$
- ✓ $B[i_p]$ precede $S[i_p]$-suffisso per $1 \le p \le k$
- ✓ $S[i_p]$-suffisso ha Q come prefisso per $1 \le p \le k$
- ✓ $B[i_p]$ + $S[i_p]$-suffisso per $1 \le p \le k$ è il suffisso che inizia con $B[i_p]$ e che ha σQ come prefisso
- ✓ $B[i_p]$ + $S[i_p]$-suffisso per LF-mapping avrà una posizione nel Suffix Array data da $j_p = LF(i_p)$
- ✓ $j_1, j_2, \dots, j_p, \dots, j_k \rightarrow [j_1, j_k+1)$ è il σQ-intervallo

```
Procedura Backward_extend(b,e,σ)
    i₁ ← più piccola posizione ≥ b tale che B[i₁] = σ
    iₖ ← più grande posizione < e tale che B[iₖ] = σ

    return [LF(i₁), LF(iₖ)+1)
```

```
Procedura Search_pattern(P, S)
    n ← |S|
    [b,e) ← [1, n+1)
    i ← |P|
    while [b,e) is not null and i ≥ 1 do
      σ ← P[i]
      [b,e) ← Backward_extend(b,e,σ)
      i ← i-1
    if [b,e) is not null then
      output S[b], S[b+1], …, S[e-1]
```

Per il momento…

Complessità ≠ O(m)

FM-index

è una rappresentazione della BWT in forma numerica.

FM-index = funzione **C** + funzione **Occ**

C: $\Sigma \rightarrow \{0, 1, 2, 3, \dots, n\}$

C(σ) = numero di simboli della BWT che sono < σ

La funzione C mappa i simboli dell'alfabeto (dollaro compreso) in un numero da 0 a n.

Occ: $\{1, 2, 3, \dots, n+1\} \times \Sigma \rightarrow \{0, 1, 2, 3, \dots, n\}$

Occ(i, σ) = numero di simboli σ in B[1,i-1]

Occ mappa ogni coppia che si può formare con un numero da 1 a n+1 e un simbolo, ad un intero da 0 a n.

Queste due funzioni rappresentano la BWT. Questo è un self index, quindi potremmo tenere queste due funzioni senza la BWT, perché si può ricostruire partendo da queste sia la BWT che il testo.

Costruiamo la funzione C:

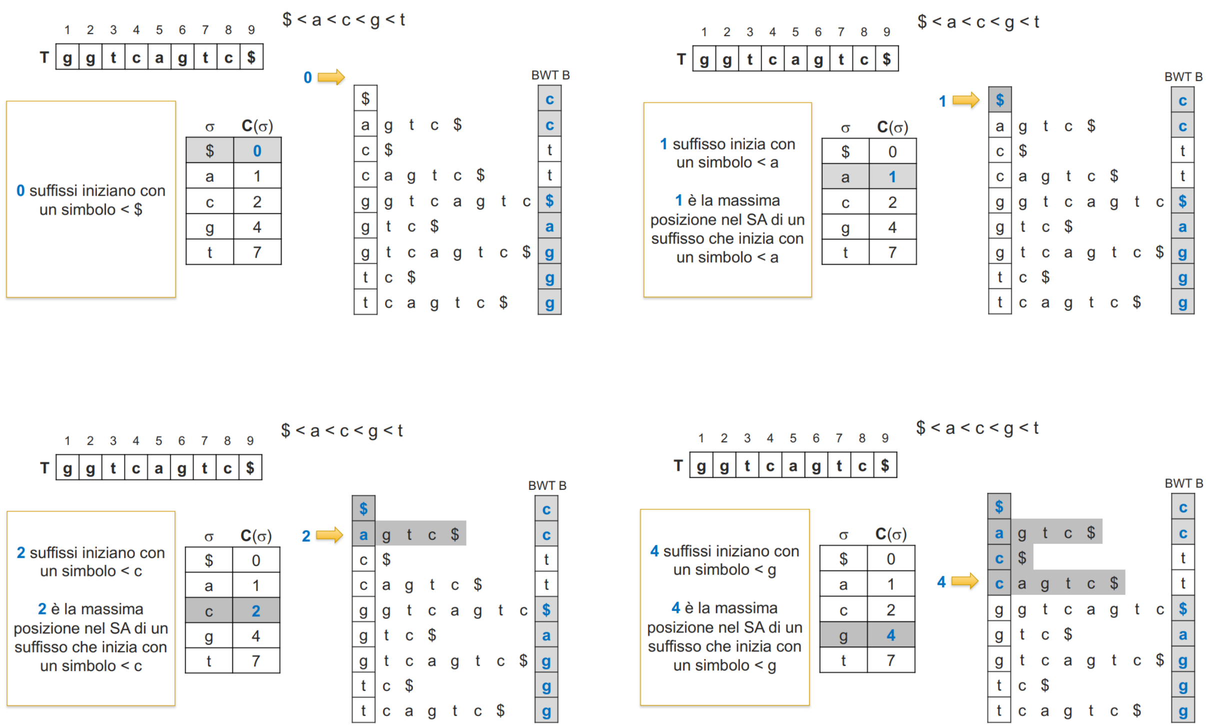
$\$ < a < c < g < t$

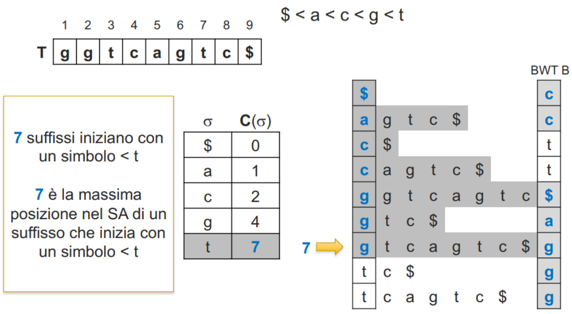
	1	2	3	4	5	6	7	8	9
T	g	g	t	c	a	g	t	c	$

σ	C(σ)
$	0
a	1
c	2
g	4
t	7

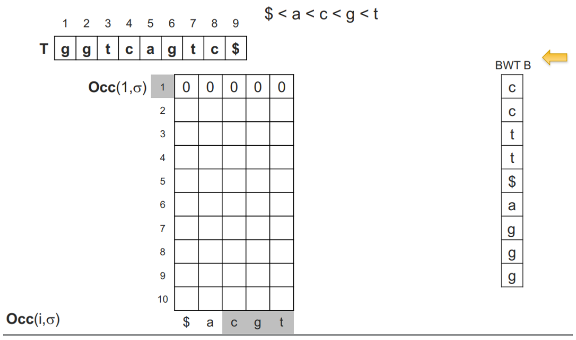
BWT B
c
c
t
t
$
a
g
g
g

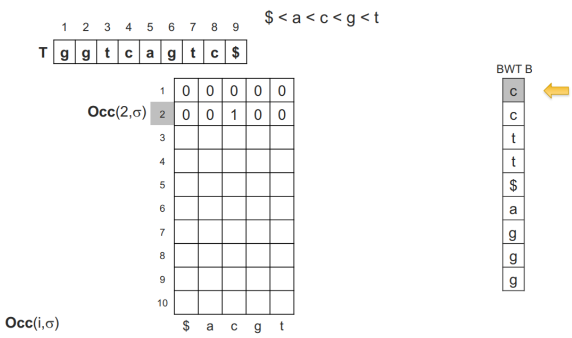
C(sigma) fornisce il numero di suffissi che iniziano con un simbolo lessicograficamente inferiore a sigma.
Questo valore C(sigma) fornisce la massima posizione nel suffix array di un simbolo che inizia con un simbolo inferiore a sigma.



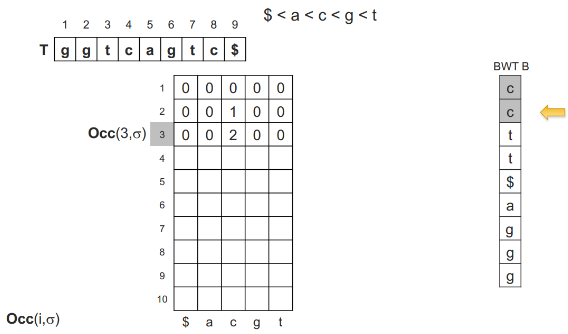


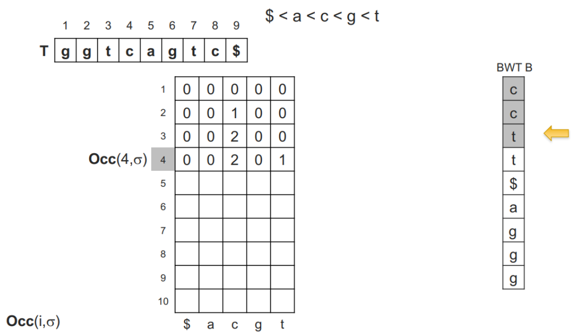
Costruiamo Occ

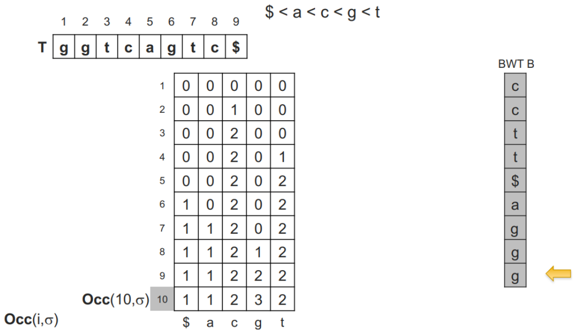


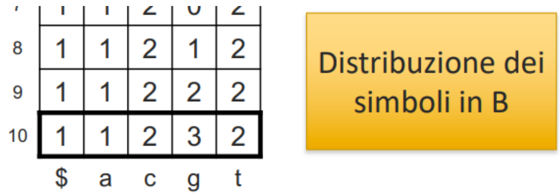


sostanzialmente abbiamo copiato la riga 1, e incrementato di 1 la casella corrispondente a quello che leggo dalla BWT nella posizione 1.









L'ultima riga della tabella ci dice che abbiamo 1 dollaro, 1a, 2c, 3g e 2t. Di conseguenza potremmo ricavare la funzione C dal quest'ultima riga.

σ	C(σ)
$	0
a	1
c	2
g	4
t	7

```
Procedura Build-FMindex(B)
   n ← |B|
   C ← empty vector of size |Σ|
   Occ ← empty table (n+1) X |Σ|
   foreach σ do
      Occ[1,σ] ← 0
   for i from 1 to n do
      foreach σ in Σ do
         Occ[i+1,σ] ← Occ[i,σ]
      Occ[i+1,B[i]] ← Occ[i+1,B[i]]+1
   C[$] ← 0
   //Scorro i simboli in ordine lessicografico
   foreach σ in Σ \ {$} do
      σ' ← simbolo immediatamente prima di σ in Σ
      C[σ] ← C[σ'] + Occ[n+1, σ']
   return (C, Occ)
```

Calcolo della LF-function

ha fatto qualche slides e esempio ma dovrebbe rifarlo la prossima volta