Lezione 18 13/12/2023

Suffix Array e Burrows-Wheeler Transform

è una struttura di indicizzazione di un testo di lunghezza n. Non contiene l'informazione sui simboli del testo, ma è un vettore di interi. Non sta in piedi da sola, ha sempre bisogno del testo accanto.

memoria occupata → θ(n *log* n)

ricerca esatta di un pattern P lungo m in tempo O(m log n)

BWT sempre per lunghezza n, è un vettore di n elementi, ma ogni cella contiene un simbolo. Il vettore da la permutazione (reversibile) dei simboli del testo.

memoria occupata → θ(n *log* |Σ|)

usata in bzip2

FM-Index di un testo T (più recente) è una rappresentazione numerica della BWT tramite due funzioni numeriche.

ricerca esatta di un pattern lungo m in tempo θ(m)

Ordinamento lessicografico

- ✓ I simboli di Σ sono ordinati lessicograficamente
- ✓ Σ viene esteso il simbolo $ considerato lessicograficamente minore di tutti gli altri
- ✓ Il simbolo $ è usato come terminatore del testo
- ✓ n è la lunghezza del testo $-terminato

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

Σ = {$, a, c, g, t} $ < a < c < g < t

Viene dato un ordinamento all'alfabeto, e viene aggiunto il simbolo terminatore ($). Tutti i testi saranno terminati dal $. Questo è il simbolo più piccolo dell'alfabeto, inferiore agli altri. La lunghezza del testo quindi comprende anche il testo, che nell'esempio è lungo 9, non 8.

Suffisso di indice q

è il suffisso che inizia nella posizione q del testo e finisce in posizione n (ultima).

<u>DEF</u>: il suffisso di indice q (q-suffisso) è il suffisso che inizia nella posizione q del testo, cioé T[q,n]

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

Σ = {$, a, c, g, t} $ < a < c < g < t

ES1: suffisso di indice 3 → T[3,9] = **tcagtc$**

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

Σ = {$, a, c, g, t} $ < a < c < g < t

ES2: suffisso di indice 9 → T[9,9] = **$** (suffisso nullo)

l'n-suffisso è quindi il suffisso nullo, perché contiene solo il dollaro.

l'1-suffisso è invece quello che include tutto il testo.

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

Σ = {$, a, c, g, t} $ < a < c < g < t

ES3: suffisso di indice 1 → T[1,9] = **ggtcagtc$** (testo T)

Ordinamento dei suffissi

Dati due suffissi s e s', sia i la più piccola posizione tale che s[i] ≠ s'[i].

s[i] < s'[i] ⇒ s < s' (oppure s' > s)
s[i] > s'[i] ⇒ s > s' (oppure s' < s)

Quindi il suffisso in comune è quello i-1.

Per ordinare i suffissi andiamo quindi a vedere l'ordinamento del simbolo nella posizione i.

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

Σ = {$, a, c, g, t} $ < a < c < g < t

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

Σ = {$, a, c, g, t} $ < a < c < g < t

6-suffisso → s = gtc$
1-suffisso → s' = ggtcagtc$

Quindi la prima posizione diversa è la posizione 2

i = 2, s[i] > s'[i] ⇒ s > s' (oppure s' < s)

La prima posizione diversa è la 4. Dico quindi che s<s'.

La posizione i non può andare oltre la posizione del dollaro.

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

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

6-suffisso → s = gtc**$**
2-suffisso → s' = gtc**a**gtc$

$i = 4$, $s[i] < s'[i]$ $\Rightarrow$ $s < s'$ (oppure $s' > s$)

Suffix Array (SA)

<u>DEF</u>: il Suffix Array (SA) di un testo T lungo n è un array S lungo n, tale che S[i] = q se e solo se il q-suffisso è l'i-esimo suffisso nell'ordinamento lessicografico dei suffissi di T.

Posizione i_1 → $S[i_1]$-suffisso
Posizione i_2 → $S[i_2]$-suffisso

$i_1 < i_2 \Rightarrow S[i_1]\text{-suffisso} < S[i_2]\text{-suffisso}$

Queste sono le loro posizioni di inizio nel testo. Se prendiamo due posizione (i1 < i2) sul suffix array, sappiamo subito che S[i1]-suffisso è minore del S[i2]-suffisso.

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

$ < a < c < g < t

Suffix Array?

Elenco dei suffissi per indice crescente	
1	g g t c a g t c $
2	g t c a g t c $
3	t c a g t c $
4	c a g t c $
5	a g t c $
6	g t c $
7	t c $
8	c $
9	$

Ordinamento dei suffissi	
9	$
5	a g t c $
8	c $
4	c a g t c $
1	g g t c a g t c $
6	g t c $
2	g t c a g t c $
7	t c $
3	t c a g t c $

L'ordinamento è dato dalle posizioni delle lettere dell'alfabeto. (questo non è l'algoritmo di costruzione del suffix array)

Questa colonna è quindi il suffix array. è un vettore lungo come il testo, indicizzato dall'alto, e contiene una permutazione delle posizioni del testo. Abbiamo quindi perso le informazioni sui singoli, abbiamo solo un vettore di posizioni.

Θ(n *log* n) in spazio

ESEMPIO: il **settimo** suffisso nell'ordinamento lessicografico è quello che inizia in posizione **2** del testo

Suffix Array S

1	**9**
2	**5**
3	**8**
4	**4**
5	**1**
6	**6**
7	**2**
8	**7**
9	**3**

Ricerca esatta con SA

1. Preprocessing del testo T per costruire il Suffix Array

2. Ricerca in tempo O(m *log* n) di un pattern P di lunghezza m

Qualche osservazione…

① se P occorre k volte in T, allora P è prefisso di k suffissi di T

T = x**ab**xab$
P = ab

P occorre 2 volte in T ⇒ P è prefisso di due suffissi di T:
✓ 2-suffisso abxab$

T = xabx**ab**$
P = ab

P occorre 2 volte in T ⇒ P è prefisso di due suffissi di T:
✓ 2-suffisso abxab$
✓ 5-suffisso ab$

② gli indici dei k suffissi sono le occorrenze di P e sono consecutivi nel Suffix Array

T = x**ab**x**ab**$
P = ab

S	
7	$
5	**ab**$
2	**ab**xab$
6	b$
3	bxab$
4	xab$
1	xabxab$

③ se P è prefisso del q-suffisso ed è minore (maggiore) di un q'-suffisso, allora anche il q-suffisso sarà minore (maggiore) del q'-suffisso

Questa osservazione serve per muoversi sul suffix array ed isolare il segmento corretto che può contenere i suffissi che hanno P come prefisso. Esempio qui sotto:

Algoritmo O(m log n)

1) Si inizializza un intervallo di posizioni [L,R] uguale a [1,n]

2) Si considera il suffisso di indice S[p] con p posizione di mezzo di [L,R], e si verifica in tempo O(m) se:

(a) P < S[p]-suffisso

si ripete **2)** ponendo [L,R] = [L,p-1]

(b) P > S[p]-suffisso

si ripete **2)** ponendo [L,R] = [p+1, R]

(c) P è prefisso di S[p]-suffisso

⇒ P occorre in posizione S[p]

Questo algoritmo trova un'occorrenza (la prima che trova), per trovare le altre dovrei muovermi sopra e sotto a quello trovato (non ci interessa). Esempio 1:

T = xabxab$
P = ab

[1,7]

	S	
1	7	$
2	5	ab$
3	2	abxab$
4	**6**	b$
5	3	bxab$
6	4	xab$
7	1	xabxab$

p = 4, S[p] = **6**

P < **6**-suffisso = b$

nuovo intervallo → [1,3]

T = xabxab$
P = ab

[1,3]

	S	
1	7	$
2	**5**	ab$
3	2	abxab$
4	6	b$
5	3	bxab$
6	4	xab$
7	1	xabxab$

p = 2, S[p] = **5**

P **=** prefisso di **5**-suffisso = ab$

→ **5** è occorrenza esatta di P in T

Esempio 2 (pattern che non occorre in T):

T = xabxab$
P = aba

[1,7] → 4

	S	
1	7	$
2	5	ab$
3	2	abxab$
4	**6**	b$
5	3	bxab$
6	4	xab$
7	1	xabxab$

p = 4, S[p] = **6**

P < **6**-suffisso = b$

nuovo intervallo → [1,3]

T = xabxab$
P = aba

[1,3] → 2

	S	
1	7	$
2	**5**	ab$
3	2	abxab$
4	6	b$
5	3	bxab$
6	4	xab$
7	1	xabxab$

p = 2, S[p] = **5**

P **>** **5**-suffisso = ab$

nuovo intervallo → [3,3]

T = xabxab$
P = aba

[3,2]

	S	
1	7	$
2	**5**	ab$
3	2	abxab$
4	6	b$
5	3	bxab$
6	4	xab$
7	1	xabxab$

p = 3, S[p] = **2**

P **<** **2**-suffisso = abxab$

nuovo intervallo → [3,2]

→ P non occorre in T

Noi prendiamo il suffix array come struttura di supporto al BWT.

Burrows-Wheeler Transform (BTW)

Esempio di BWT (Wikipedia)

```
TRENTATRE.TRENTINI.ANDARONO.A.TRENTO.TUTTI.E.TRENTATRE.TROTTERELLANDO
```

↓

```
OIIEEAEO..LDTTNN.RRRRRRRTNTTLEAAIOEEEENTRDRTTETTTTATNNTTNNAAO....OU.T
```

BWT → permutazione reversibile dei simboli del testo T

Rotazione di indice q

DEF: la rotazione di indice q (q-rotazione) è la concatenazione del q-suffisso T[q,n] e del prefisso T[1,q-1]

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

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

ES1: rotazione di indice 3 → T[3,9] T[1,2] = tcagtc$gg

La q-rotazione inizia con il q-suffisso.

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

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

ES2: rotazione di indice 9 → T[9,9] T[1,8] = $ggtcagtc

La n-rotazione sposta il dollaro ad inizio testo.

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

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

ES3: rotazione di indice 1 → T[1,9] T[1,0] = ggtcagtc$

La 1-rotazione coincide con il testo completo e con l'1-suffisso.

La q-rotazione r è una stringa lunga n che ha:

1. il q-suffisso come prefisso
2. il prefisso lungo q-1 come suffisso

⇒ r[1] è uguale a T[q] (primo simbolo del q-suffisso)
⇒ r[n] è uguale a T[q-1] se q > 1
⇒ r[n] è uguale a T[n] se q = 1

⇒ l'ultimo simbolo della q-rotazione è il simbolo che in T precede il q-suffisso (se q > 1) oppure è il simbolo T[n] (se q = 1)

Il primo simbolo della rotazione è sempre uguale al simbolo q nel testo.

L'ultimo simbolo della rotazione è uguale al simbolo in posizione q-1 se q>1, altrimenti se q=1 è l'ultimo simbolo del testo.

Esempi:

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

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

3-rotazione r = tcagtc$gg

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

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

1-rotazione r = ggtcagtc$

⇒ r[1] è uguale a T[q] (primo simbolo del q-suffisso)
⇒ r[n] è uguale a T[q-1] se q > 1
⇒ r[n] è uguale a T[n] se q = 1

⇒ l'ultimo simbolo della q-rotazione è il simbolo iniziale della (q-1)-rotazione (e quindi del (q-1)-suffisso) se q > 1
⇒ il (q-1)-suffisso contiene il q-suffisso come suffisso se q > 1

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

Σ = {$, a, c, g, t} $ < a < c < g < t

3-rotazione r = tcagtc$gg

3-rotazione r = tcagtc$gg 2-rotazione r' = gtcagtc$g

Quindi se prendo l'ultimo simbolo della 3-rotazione e lo sposto alla prima posizione, ottengo la 2-rotazione.

⇒ r[n] è uguale a T[n] se q = 1

⇒ l'ultimo simbolo della q-rotazione è il simbolo iniziale della n-rotazione (e quindi dell'n-suffisso) se q = 1

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

Σ = {$, a, c, g, t} $ < a < c < g < t

1-rotazione r = ggtcagtc$

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

Σ = {$, a, c, g, t} $ < a < c < g < t

1-rotazione r = ggtcagtc$ 9-rotazione r' = $ggtcagtc

Ordinamento delle rotazioni

Date due rotazioni r e r', sia i la più piccola posizione tale che r[i] ≠ r'[i].

r[i] < r'[i] ⇒ r < r' (oppure r' > r)
r[i] > r'[i] ⇒ r > r' (oppure r' < r)

è la stessa regola di prima

Esempi:

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

Σ = {$, a, c, g, t} $ < a < c < g < t

6-rotazione → r = gtc$ggtca
1-rotazione → r' = ggtcagtc$

i = 2, r[i] > r'[i] ⇒ r > r' (oppure r' < r)

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

Σ = {$, a, c, g, t} $ < a < c < g < t

6-rotazione → r = gtc$ggtca
2-rotazione → r' = gtcagtc$g

i = 4, r[i] < r'[i] ⇒ r < r' (oppure r' > r)

In generale, la q-rotazione è minore della p-rotazione se e solo se il q-suffisso è minore del p-suffisso

Grazie all’aggiunta del simbolo finale $, le q-rotazioni e i q-suffissi sono sincronizzati, se una q-rotazione è minore di un’altra p-rotazione allora il q-suffisso è minore del p-suffisso.

Date una q-rotazione r_1 e una p-rotazione r_2 tali che:

1. $r_1 < r_2$
2. $r_1[n] = r_2[n]$

Allora la (q-1)-rotazione r'_1 è minore della (p-1)-rotazione r'_2

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

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

7-rotazione r_1 = tc$ggtcag < 3-rotazione r_2 = tcagtc$gg

⇒ 6-rotazione < 2-rotazione

Questo se i due simboli finali sono uguali.

Date una q-rotazione r_1 e una p-rotazione r_2 tali che:

~~1. $r_1 < r_2$~~

2. $r_1[n] < r_2[n]$

Allora la (q-1)-rotazione r'_1 è minore della (p-1)-rotazione r'_2

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

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

3-rotazione r_2 = tcagtc$gg 8-rotazione r_2 = c$ggtcagt

⇒ 2-rotazione < 7-rotazione

Definizione di BWT

DEF: la BWT B di un testo T lungo n è un array di lunghezza n tale che $B[i] = r_i[n]$ se e solo se r_i è l'i-esima rotazione nell'ordinamento lessicografico delle rotazioni di T.

Nella posizione i-esima di BWT c'è un simbolo.
Immaginiamo di avere tutte le rotazioni del testo, di ordinarle lessicograficamente, prendiamo l'ultimo simbolo della rotazione più piccola e la metto in BWT.

Esempio:

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

① Riempiamo una matrice n x n con tutte le rotazioni di T

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

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

② Ordiniamo lessicograficamente le rotazioni

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

In ogni riga ci mettiamo una rotazione.

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

La BWT B è l'ultima colonna della matrice delle rotazioni ordinate lessicograficamente

$\theta(n \, log \, |\Sigma|)$ in spazio

BWT B
c
c
t
t
$
a
g
g
g

L'ultima colonna è la BWT. Salva quindi l'ultimo simbolo delle rotazioni, con le rotazioni in ordine lessicografico.

	F								BWT B
9	$								c
5	a								c
8	c								t
4	c								t
1	g								$
6	g								a
2	g								g
7	t								g
3	t								g

La prima colonna (F) contiene i simboli iniziali delle rotazioni ordinate. (Mentre la BWT sono i simboli finali delle rotazioni ordinate)

F fornisce sempre l'ordinamento lessicografico dei simboli del testo.

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

ESEMPIO:
per i=3
- B[i] = **t** coincide con T[7]
- F[i] = **c** coincide con T[8]

Sia B[i] = r_i[n] con r_i i-esima più piccola rotazione nell'ordinamento lessicografico.
Se r_i è la q-rotazione di T, allora B[i] è l'ultimo simbolo della q-rotazione, cioè T[q-1].
Il primo simbolo della q-rotazione è il simbolo T[q] e coincide con il simbolo in posizione F[i] del vettore F.

	F	BWT B
9	$	c
5	a	c
8	c	t
4	c	t
1	g	$
6	g	a
2	g	g
7	t	g
3	t	g

Prima proprietà

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

Per ogni posizione i, il simbolo B[i] precede nel testo il simbolo F[i]

→ Proprietà P1

ESEMPIO:
per i=6
- B[i] = **a** coincide con T[5]
- F[i] = **g** coincide con T[6]

	F	BWT B
9	$	c
5	a	c
8	c	t
4	c	t
1	g	$
6	g	a
2	g	g
7	t	g
3	t	g

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

Per ogni posizione i, il simbolo B[i] precede nel testo il simbolo F[i]

→ Proprietà P1

NB: B[1] è sempre T[n-1]

ESEMPIO:
per i=1
- B[i] = **c** coincide con T[8]
- F[i] = **$** coincide con T[9]

	F	BWT B
9	$	c
5	a	c
8	c	t
4	c	t
1	g	$
6	g	a
2	g	g
7	t	g
3	t	g

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

Per ogni posizione i tale che B[i] è diverso da $, il simbolo B[i] precede nel testo il simbolo F[i].
Se B[i] = $, allora F[i] = T[1]

→ Proprietà P1

ESEMPIO:
per i=5
- B[i] = **$** coincide con T[9]
- F[i] = **g** coincide con T[1]

	F	BWT B
9	$	c
5	a	c
8	c	t
4	c	t
1	g	$
6	g	a
2	g	g
7	t	g
3	t	g

Modifichiamo leggermente la (prima) proprietà per farla funzionare anche con il dollaro.