STRUMENTI FORMALI PER LA BIOINFORMATICA

La Trasformata di Burrows-Wheeler (BWT)

Ringraziamenti

Gli esempi sono presi dagli articoli:

Sabrina Mantaci, Antonio Restivo, Giovanna Rosone, Marinella Sciortino: An extension of the Burrows-Wheeler Transform. Theor. Comput. Sci. 387(3): 298-312 (2007)

Sara Giuliani, Zsuzsanna Lipták, Francesco Masillo, Romeo Rizzi: When a dollar makes a BWT. Theor. Comput. Sci. 857: 123-146 (2021)

La Trasformata di Burrows-Wheeler (BWT)

La Trasformata di Burrows-Wheeler è un algoritmo di compressione dati "lossless" (cioè senza perdita) usata da programmi di compressione.

Fu inventata da David Wheeler negli anni 70. Successivamente fu sviluppata in un technical report del 1994, mai pubblicato, scritto con Michael Burrows.

La Trasformata di Burrows-Wheeler (BWT)

Il testo viene identificato con una stringa.

L'algoritmo prende in input una stringa w e fornisce in output una coppia (L, I), dove L è un'altra stringa più facile da comprimere e I è un intero positivo.

La coppia $\text{bwt}(w) = (L, I)$ è chiamata trasformata di Burrows-Wheeler di w.

È possibile calcolare $\text{bwt}(w)$ in tempo lineare.

La Trasformata di Burrows-Wheeler (BWT)

Da Wikipedia:

Se la stringa w contiene molte ripetizioni di altre sottostringhe, nella sua trasformata troveremo diversi punti in cui uno stesso carattere si ripete molte volte.

Questo è utile per la compressione perché diventa facile comprimere una stringa in cui compaiono lunghe sequenze di caratteri tutti uguali.

La Trasformata di Burrows-Wheeler (BWT)

Studiata da:

- teorici della combinatoria delle parole
- ricercatori di teoria degli algoritmi

Oggetto di interesse dei bioinformatici.

Definizione

Sia $\Sigma = \{a_0, \ldots, a_k\}$ *un alfabeto e sia* $a_0 < a_1 < \ldots < a_k$ *un ordinamento degli elementi di* Σ. *Siano* $x, y \in \Sigma^*$.
Diremo che $x < y$ *rispetto all'***ordine lessicografico** *se* x *e* y *verificano una delle condizioni seguenti:*

1. $y = xz$ *con* $z \in \Sigma^+$, *cioè* x *è un prefisso di* y *e* $x \neq y$.

Definizione

Sia $\Sigma = \{a_0, \ldots, a_k\}$ *un alfabeto e sia* $a_0 < a_1 < \ldots < a_k$ *un ordinamento degli elementi di* Σ. *Siano* $x, y \in \Sigma^*$.
Diremo che $x < y$ *rispetto all'***ordine lessicografico** *se* x *e* y *verificano una delle condizioni seguenti:*

1. $y = xz$ *con* $z \in \Sigma^+$, *cioè* x *è un prefisso di* y *e* $x \neq y$.
2. $x = zax'$, $y = zby'$, *con* $z, x', y' \in \Sigma^*$, $a, b \in \Sigma$ *e* $a < b$.

Definizione

Sia $\Sigma = \{a_0, \ldots, a_k\}$ *un alfabeto e sia* $a_0 < a_1 < \ldots < a_k$ *un ordinamento degli elementi di* Σ. *Siano* $x, y \in \Sigma^*$.
Diremo che $x < y$ *rispetto all'***ordine lessicografico** *se* x *e* y *verificano una delle condizioni seguenti:*

1. $y = xz$ *con* $z \in \Sigma^+$, *cioè* x *è un prefisso di* y *e* $x \neq y$.
2. $x = zax'$, $y = zby'$, *con* $z, x', y' \in \Sigma^*$, $a, b \in \Sigma$ *e* $a < b$.

- Negli esempi che seguono, supporremo le lettere da a a z ordinate come in un dizionario: $a < b < \ldots < z$.

Ordine lessicografico

Definizione

Sia $\Sigma = \{a_0, \ldots, a_k\}$ *un alfabeto e sia* $a_0 < a_1 < \ldots < a_k$ *un ordinamento degli elementi di* Σ. *Siano* $x, y \in \Sigma^*$.
Diremo che $x < y$ *rispetto all'***ordine lessicografico** *se* x *e* y *verificano una delle condizioni seguenti:*

1. $y = xz$ *con* $z \in \Sigma^+$, *cioè* x *è un prefisso di* y *e* $x \neq y$.
2. $x = zax'$, $y = zby'$, *con* $z, x', y' \in \Sigma^*$, $a, b \in \Sigma$ *e* $a < b$.

- Negli esempi che seguono, supporremo le lettere da a a z ordinate come in un dizionario: $a < b < \ldots < z$.
- Date due qualsiasi parole $x, y \in \Sigma^*$, con $x \neq y$, risulta $x < y$ oppure $y < x$.

Proposizione

Siano $x, y \in A^*$. Valgono le seguenti proprietà.

(1) *$x < y$ se e solo se $zx < zy$, per ogni $z \in A^*$.*

(2) *If $x < y$ e x non è prefisso di y, allora $xu < yv$ per ogni $u, v \in A^*$.*

Ordine lessicografico

(1) Siano $x, y \in A^*$. Allora $x < y$ se e solo se $zx < zy$, per ogni $z \in A^*$.

Ordine lessicografico

(1) Siano $x, y \in A^*$. Allora $x < y$ se e solo se $zx < zy$, per ogni $z \in A^*$.

- Esempio. Supponiamo $a < b$.
  $ab < aba$ e per ogni $z \in A^*$, $zab < zaba$.

Ordine lessicografico

(1) Siano $x, y \in A^*$. Allora $x < y$ se e solo se $zx < zy$, per ogni $z \in A^*$.

- Esempio. Supponiamo $a < b$.
  $ab < aba$ e per ogni $z \in A^*$, $zab < zaba$.

- Esempio. Supponiamo $a < b$.
  $aa < ab$ e per ogni $z \in A^*$, $zaa < zab$.

(2) Se $x < y$ e x non è prefisso di y, allora $xu < yv$ per ogni $u, v \in A^*$.

(2) Se $x < y$ e x non è prefisso di y, allora $xu < yv$ per ogni $u, v \in A^*$.

- Esempio. Supponiamo $a < b$.
  $aa < ab$ e per ogni $u, v \in A^*$, $aau < abv$

Una parola w è primitiva se $w = v^n$ implica $n = 1$.

Una parola w è primitiva se $w = v^n$ implica $n = 1$.

Nota che la parola vuota non è primitiva.

Una parola w è primitiva se $w = v^n$ implica $n = 1$.

Nota che la parola vuota non è primitiva.

$abab$ non è primitiva perché $abab = (ab)^2$. Invece aba, abb sono parole primitive.

Una parola w è primitiva se $w = v^n$ implica $n = 1$.

Nota che la parola vuota non è primitiva.

$abab$ non è primitiva perché $abab = (ab)^2$. Invece aba, abb sono parole primitive.

Ogni parola non vuota w è potenza di un'unica parola primitiva z. La stringa z prende il nome di radice di w.

Coniugazione

Due parole x, y sono coniugate se esistono parole u, v tali che $x = uv, y = vu$.

La relazione di coniugazione è una relazione di equivalenza.

Una classe di coniugazione è una classe di questa relazione di equivalenza.

Una classe di coniugazione è spesso chiamata necklace.

Inoltre, u è chiamata una rotazione ciclica di v se u, v sono coniugate.

Quante coniugate ha la stringa *abab*?

Quante coniugate ha la stringa *abab*?

La stringa *abab* ha due coniugate: *abab* e *baba*.

Quante coniugate ha la stringa *abab*?

La stringa *abab* ha due coniugate: *abab* e *baba*.

Quante coniugate ha la stringa *aaa*?

Quante coniugate ha la stringa *abab*?

La stringa *abab* ha due coniugate: *abab* e *baba*.

Quante coniugate ha la stringa *aaa*?

La stringa *aaa* ha una sola coniugata, è coniugata solo di sé stessa.

Se w è una parola primitiva, tutte le sue coniugate sono primitive.

Proposizione

Una parola primitiva di lunghezza n ha n distinte coniugate.

Definizione

Dati due insiemi non vuoti X e Y,
una funzione $f : X \to Y$ da X in Y è una relazione che associa a ogni elemento x in X uno e un solo $y = f(x)$ in Y.
X è il **dominio** *della funzione,*
Y è il **codominio** *della funzione.*

Definizione

Dati due insiemi non vuoti X e Y,
una funzione $f : X \to Y$ da X in Y è una relazione che associa a ogni elemento x in X uno e un solo $y = f(x)$ in Y.
X è il **dominio** *della funzione,*
Y è il **codominio** *della funzione.*

Quindi, per definire una specifica funzione occorre fornire:

Definizione

Dati due insiemi non vuoti X e Y,
una funzione $f : X \to Y$ da X in Y è una relazione che associa a ogni elemento x in X uno e un solo $y = f(x)$ in Y.
X è il **dominio** *della funzione,*
Y è il **codominio** *della funzione.*

Quindi, per definire una specifica funzione occorre fornire:

1. dominio

Definizione

Dati due insiemi non vuoti X e Y,
una funzione $f : X \to Y$ da X in Y è una relazione che associa a ogni elemento x in X uno e un solo $y = f(x)$ in Y.
X è il **dominio** *della funzione,*
Y è il **codominio** *della funzione.*

Quindi, per definire una specifica funzione occorre fornire:

1. dominio
2. codominio

Definizione

Dati due insiemi non vuoti X e Y,
una funzione $f : X \to Y$ da X in Y è una relazione che associa a ogni elemento x in X uno e un solo $y = f(x)$ in Y.
X è il **dominio** *della funzione,*
Y è il **codominio** *della funzione.*

Quindi, per definire una specifica funzione occorre fornire:

1. dominio
2. codominio
3. la relazione che a ogni elemento del dominio associa un elemento del codominio.

Definizione

Una funzione $f : X \to Y$ *è iniettiva se*

$\forall x, x' \in X \quad x \neq x' \Rightarrow f(x) \neq f(x')$

Definizione

Una funzione $f : X \to Y$ *è iniettiva se*

$\forall x, x' \in X \quad x \neq x' \Rightarrow f(x) \neq f(x')$

Definizione

Una funzione $f : X \to Y$ *è suriettiva* $\Leftrightarrow \forall y \in Y \ \exists x \in X : y = f(x)$

Definizione

Una funzione $f : X \to Y$ *è iniettiva se*
$\forall x, x' \in X \quad x \neq x' \Rightarrow f(x) \neq f(x')$

Definizione

Una funzione $f : X \to Y$ *è suriettiva* $\Leftrightarrow \forall y \in Y \ \exists x \in X : y = f(x)$

Definizione

Una funzione $f : X \to Y$ *è una funzione biettiva di* X *su* Y *(o una biezione tra* X *e* Y*) se* f *è iniettiva e suriettiva.*

Definizione

Una permutazione di X è una funzione biettiva di X su X.

Se $X = \{1, \ldots, n\}$ è un insieme finito, una permutazione di X è abitualmente rappresentata come segue

$$\begin{pmatrix} 1 & 2 & \cdots & n \\ f(1) & f(2) & \cdots & f(n) \end{pmatrix}$$

Esempio

Se $X = \{1, 2, 3, 4\}$, la permutazione $f : \{1, 2, 3, 4\} \to \{1, 2, 3, 4\}$ tale che

$$f(1) = 4, \quad f(2) = 3, \quad f(3) = 2, \quad f(4) = 1$$

è rappresentata da

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 3 & 2 & 1 \end{pmatrix}$$

Nel seguito una stringa $a_1 \cdots a_n$ verrà a volte identificata con la lista $(a_1, \ldots, a_n)$.

Data una stringa

$$w = a_1 a_2 \cdots a_n$$

per ottenere bwt(w) occorre innanzitutto costruire una speciale matrice che chiameremo bwt-matrix.

Data una stringa

$$w = a_1 a_2 \cdots a_n$$

per ottenere bwt(w) occorre innanzitutto costruire una speciale matrice che chiameremo bwt-matrix.

- Si forma un elenco di tutte le coniugate di w.

Data una stringa

$$w = a_1 a_2 \cdots a_n$$

per ottenere bwt(w) occorre innanzitutto costruire una speciale matrice che chiameremo bwt-matrix.

- Si forma un elenco di tutte le coniugate di w.
- Si ordinano le coniugate in ordine lessicografico.

Data una stringa

$$w = a_1 a_2 \cdots a_n$$

per ottenere bwt(w) occorre innanzitutto costruire una speciale matrice che chiameremo bwt-matrix.

- Si forma un elenco di tutte le coniugate di w.
- Si ordinano le coniugate in ordine lessicografico.
- La bwt-matrix avrà come righe le coniugate di w ordinate in ordine lessicografico.

Esempio 1

Sia $w = banana$.

Esempio 1

Sia $w = banana$.

tutte le coniugate

```
banana
ananab
nanaba
anaban
nabana
abanan
```

Esempio 1

Sia $w = banana$.

tutte le coniugate		tutte le coniugate ordinate	
banana		1	abanan
ananab		2	anaban
nanaba		3	ananab
anaban	$\rightarrow$	4	banana
nabana	ordine	5	nabana
abanan	lessicografico	6	nanaba

Esempio 2

- Sia $w = abraca$
- Ordiniamo lessicograficamente tutte le coniugate di w.

1	a	a	b	r	a	c
2	a	b	r	a	c	a
3	a	c	a	a	b	r
4	b	r	a	c	a	a
5	c	a	a	b	r	a
6	r	a	c	a	a	b

La Trasformata di Burrows-Wheeler (BWT)

La trasformata di Burrows-Wheeler bwt(w) di w è la coppia (L, I), dove L è l'ultima colonna della bwt-matrix e I è l'indice della riga che contiene la parola w.

Esempio 1 - continua

Sia $w = banana$.

tutte le coniugate		tutte le coniugate ordinate	
banana		1	abanan
ananab		2	anaban
nanaba		3	ananab
anaban	$\rightarrow$	4	banana
nabana	ordine	5	nabana
abanan	lessicografico	6	nanaba

$L = nnbaaa$, $I = 4$, quindi $\text{bwt}(w) = (nnbaaa, 4)$.

Esempio 2 - continua

- Sia $w = abraca$
- Ordiniamo lessicograficamente tutte le coniugate di w.

1	a	a	b	r	a	c
2	a	b	r	a	c	a
3	a	c	a	a	b	r
4	b	r	a	c	a	a
5	c	a	a	b	r	a
6	r	a	c	a	a	b

- $L = caraab$, $I = 2$, quindi $\text{bwt}(w) = (caraab, 2)$.

Nota. Negli esempi, le parole *banana* e *abraca* sono primitive. Per questa ragione le loro bwt-matrix hanno un numero di righe distinte uguale alla loro lunghezza, per entrambe uguale a 6.

Invece se la parola non fosse primitiva, come *abab*, come definire la bwt-matrix?

Le coniugate distinte di una parola non primitiva sono in numero inferiore alla lunghezza della stringa.

Differenti definizioni

Sia w una parola non primitiva, sia $w = z^n$, $n > 1$, con z unica parola primitiva. La stringa z è la radice di w.

Vedremo che per ricostruire la stringa dalla sua BWT dovremmo comunque riportare le righe uguali nella bwt-matrix di w oppure dovremmo costruire la bwt(z) della radice z di w, memorizzando l'esponente n.

Per tale ragione alcuni ricercatori di combinatoria delle parole definiscono la trasformata di Burrows-Wheeler bwt(w) di w solo se w è una parola primitiva.

Nota. La definizione data è quella usata dai ricercatori di combinatoria delle parole. Ci sono piccole differenze con la definizione dei ricercatori di teoria degli algoritmi.

Nota. La definizione data è quella usata dai ricercatori di combinatoria delle parole. Ci sono piccole differenze con la definizione dei ricercatori di teoria degli algoritmi.

Essi assumono che ogni stringa w termini con un carattere di fine stringa, denotato \$, e che tale carattere sia più piccolo di ogni carattere in Σ

$$\forall \sigma \in \Sigma \quad \$ < \sigma$$

Nota. La definizione data è quella usata dai ricercatori di combinatoria delle parole. Ci sono piccole differenze con la definizione dei ricercatori di teoria degli algoritmi.

Essi assumono che ogni stringa w termini con un carattere di fine stringa, denotato \$, e che tale carattere sia più piccolo di ogni carattere in Σ

$$\forall \sigma \in \Sigma \quad \$ < \sigma$$

Esempio. Se $w = banana$, allora si considera $y = banana\$$.

Nota. La definizione data è quella usata dai ricercatori di combinatoria delle parole. Ci sono piccole differenze con la definizione dei ricercatori di teoria degli algoritmi.

Essi assumono che ogni stringa w termini con un carattere di fine stringa, denotato \$, e che tale carattere sia più piccolo di ogni carattere in Σ

$$\forall \sigma \in \Sigma \quad \$ < \sigma$$

Esempio. Se $w = banana$, allora si considera $y = banana\$$.

La trasformata di Burrows-Wheeler bwt(w) di w è solo l'ultima colonna della bwt-matrix.

Nota. La definizione data è quella usata dai ricercatori di combinatoria delle parole. Ci sono piccole differenze con la definizione dei ricercatori di teoria degli algoritmi.

Essi assumono che ogni stringa w termini con un carattere di fine stringa, denotato \$, e che tale carattere sia più piccolo di ogni carattere in Σ

$$\forall \sigma \in \Sigma \quad \$ < \sigma$$

Esempio. Se $w = banana$, allora si considera $y = banana\$$.

La trasformata di Burrows-Wheeler bwt(w) di w è solo l'ultima colonna della bwt-matrix.

Perdiamo informazione? No perché è facile provare che sulla prima riga della bwt-matrix di $w\$$ ci sarà la stringa $\$w$.

Reversibilità della BWT

L'aspetto interessante della BWT è la sua reversibilità: la stringa iniziale si ricostruisce a partire da L e I.

Vediamo come.

Nota: Ogni colonna della bwt-matrix di w è una stringa che si ottiene permutando i caratteri della stringa w.

Cioè data $w = a_1 a_2 \cdots a_n$, per ogni colonna $(b_1, \ldots, b_n)$ nella bwt-matrix di w, esiste una permutazione f di $\{1, \ldots, n\}$ tale che $(b_1, \ldots, b_n) = (a_{f(1)}, \ldots, a_{f(n)})$.

Nota: Ogni colonna della bwt-matrix di w è una stringa che si ottiene permutando i caratteri della stringa w.

Cioè data $w = a_1 a_2 \cdots a_n$, per ogni colonna $(b_1, \ldots, b_n)$ nella bwt-matrix di w, esiste una permutazione f di $\{1, \ldots, n\}$ tale che $(b_1, \ldots, b_n) = (a_{f(1)}, \ldots, a_{f(n)})$.

Perché?

La matrice $\mathcal{B}(w)$ delle coniugate di $w = a_1 a_2 \cdots a_n$, non ordinate lessicograficamente, ha questa forma

$$\mathcal{B}(w) = \begin{matrix} a_1 & a_2 & \dots & a_n \\ a_2 & a_3 & \dots & a_1 \\ a_3 & a_4 & \dots & a_2 \\ \vdots & \vdots & \dots & \vdots \\ a_n & a_1 & \dots & a_{n-1} \end{matrix}$$

Le coniugate le posso "leggere" per riga e per colonna. Più precisamente: ogni riga è una coniugata di w e ogni colonna è una coniugata di w.

Esempio 1

Sia $w = banana$.

$$\mathcal{B}(w) = \begin{matrix} b & a & n & a & n & a \\ a & n & a & n & a & b \\ n & a & n & a & b & a \\ a & n & a & b & a & n \\ n & a & b & a & n & a \\ a & b & a & n & a & n \end{matrix}$$

Esempio 2

Sia $w = abraca$

$$\mathcal{B}(w) = \begin{matrix} a & b & r & a & c & a \\ b & r & a & c & a & a \\ r & a & c & a & a & b \\ a & c & a & a & b & r \\ c & a & a & b & r & a \\ a & a & b & r & a & c \end{matrix}$$

Reversibilità della BWT

La bwt-matrix di w è una permutazione delle righe di $\mathcal{B}(w)$ e quindi i caratteri in colonna restano gli stessi anche se in un diverso ordine.

Allora ogni colonna nella bwt-matrix di w è una stringa che si ottiene permutando i caratteri di una coniugata di w e quindi permutando i caratteri della stringa w.

Cioè, data $w = a_1 a_2 \cdots a_n$, per ogni colonna $(b_1, \ldots, b_n)$ nella bwt-matrix di w, esiste una permutazione f di $\{1, \ldots, n\}$ tale che $(b_1, \ldots, b_n) = (a_{f(1)}, \ldots, a_{f(n)})$.

Reversibilità della BWT

Questo vuol dire che, data $bwt(w) = (L, I)$, possiamo ricostruire da L la prima colonna, F.

Reversibilità della BWT

Questo vuol dire che, data $bwt(w) = (L, I)$, possiamo ricostruire da L la prima colonna, F.

Come?

Questo vuol dire che, data $bwt(w) = (L, I)$, possiamo ricostruire da L la prima colonna, F.

Come?

F è costituita dagli stessi caratteri che si trovano in L e con lo stesso numero di occorrenze con cui si trovano in L, cioè F è una permutazione di L.

Reversibilità della BWT

Questo vuol dire che, data $bwt(w) = (L, I)$, possiamo ricostruire da L la prima colonna, F.

Come?

F è costituita dagli stessi caratteri che si trovano in L e con lo stesso numero di occorrenze con cui si trovano in L, cioè F è una permutazione di L.

Ma in che ordine dobbiamo scrivere i caratteri di L per avere F?

Reversibilità della BWT

Questo vuol dire che, data $bwt(w) = (L, I)$, possiamo ricostruire da L la prima colonna, F.

Come?

F è costituita dagli stessi caratteri che si trovano in L e con lo stesso numero di occorrenze con cui si trovano in L, cioè F è una permutazione di L.

Ma in che ordine dobbiamo scrivere i caratteri di L per avere F?

Siccome la stringa in ogni riga della bwt-matrix è minore o uguale a quella nella successiva riga, la colonna F è costituita dai caratteri della stringa w in ordine non decrescente.

Esempio 1 - continua

Data $w = banana$ e $\text{bwt}(w) = (nnbaaa, 4)$, costruiamo F ordinando lessicograficamente gli elementi di L

	F		L
1	a	n	1
2	a	n	2
3	a	b	3
4	b	a	4
5	n	a	5
6	n	a	6

Esempio 2 - continua

Data $w = abraca$ e $\text{bwt}(w) = (caraab, 2)$, costruiamo F ordinando lessicograficamente gli elementi di L

	F		L
1	a	c	1
2	a	a	2
3	a	r	3
4	b	a	4
5	c	a	5
6	r	b	6

Reversibilità della BWT

Consideriamo la lista $F = (b_1, \ldots, b_n)$ e la lista $L = (c_1, \ldots, c_n)$.

Siccome le due liste differiscono solo per l'ordine in cui sono rappresentati gli elementi, esiste almeno una permutazione f di $\{1, \ldots, n\}$ tale che

$$F = (b_1, \ldots, b_n) = (c_{f(1)}, \ldots, c_{f(n)})$$

Useremo la notazione

$$f = \begin{pmatrix} 1 & 2 & \cdots & n \\ f(1) & f(2) & \cdots & f(n) \end{pmatrix}$$

Non sempre la permutazione è unica.

Esempio 1

$w = banana$, $\mathrm{bwt}(w) = (nnbaaa, 4)$.

	F
1	a
2	a
3	a
4	b
5	n
6	n

	L
n	1
n	2
b	3
a	4
a	5
a	6

$$f = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 4 & 5 & 6 & 3 & 1 & 2 \end{pmatrix}$$

$$(a, a, a, b, n, n) = (b_1, b_2, b_3, b_4, b_5, b_6)$$

$$= (c_{f(1)}, c_{f(2)}, c_{f(3)}, c_{f(4)}, c_{f(5)}, c_{f(6)})$$

Esempio 2

$w = abraca$, $\mathrm{bwt}(w) = (caraab, 2)$.

	F
1	a
2	a
3	a
4	b
5	c
6	r

	L
c	1
a	2
r	3
a	4
a	5
b	6

$$f = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 4 & 5 & 6 & 1 & 3 \end{pmatrix}$$

$$(a, a, a, b, c, r) = (b_1, b_2, b_3, b_4, b_5, b_6)$$

$$= (c_{f(1)}, c_{f(2)}, c_{f(3)}, c_{f(4)}, c_{f(5)}, c_{f(6)})$$

Esempio 2

$w = abraca,\ bwt(w) = (caraab, 2).$

	F		L
1	a	c	1
2	a	a	2
3	a	r	3
4	b	a	4
5	c	a	5
6	r	b	6

$$g = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 5 & 2 & 4 & 6 & 1 & 3 \end{pmatrix}$$

$$(a, a, a, b, c, r) = (b_1, b_2, b_3, b_4, b_5, b_6)$$

$$= (c_{g(1)}, c_{g(2)}, c_{g(3)}, c_{g(4)}, c_{g(5)}, c_{g(6)})$$

Proprietà LF

La permutazione che ci interessa per la ricostruzione di w da bwt(w) è basata su una proprietà particolare della bwt-matrix, nota come proprietà LF.

La proprietà LF si trova spesso enunciata al modo seguente:

> *Per ogni carattere σ, l'i-esima occorrenza di σ in F corrisponde alla i-esima occorrenza di σ in L.*

Che significa?

Se in w ho s occorrenze della lettera σ, allora in L ho s occorrenze del carattere σ. Queste s occorrenze del carattere σ in L corrispondono a s coniugate di w che terminano con σ,

$$x_1\sigma, \ldots, x_s\sigma$$

dove $x_i \neq x_j$ per $1 \leq i, j \leq s$, $i \neq j$.

Conseguentemente in F ho s occorrenze della lettera σ che corrispondono a s coniugate di w che iniziano con σ. Queste coniugate sono

$$\sigma x_1, \ldots, \sigma x_s$$

poiché queste parole sono s coniugate distinte di w ed iniziano con σ.

Per ogni carattere σ, l'i-esima occorrenza di σ in F corrisponde alla i-esima occorrenza di σ in L.

Significa:

Proprietà LF

Per ogni carattere σ, l'i-esima occorrenza di σ in F corrisponde alla i-esima occorrenza di σ in L.

Significa:

Se $x_i\sigma$ precede $x_j\sigma$ nella bwt-matrix allora σx_i precede σx_j nella bwt-matrix.

Per ogni carattere σ, l'i-esima occorrenza di σ in F corrisponde alla i-esima occorrenza di σ in L.

Significa:

Se $x_i\sigma$ precede $x_j\sigma$ nella bwt-matrix allora σx_i precede σx_j nella bwt-matrix.

Dimostriamolo.

Proprietà LF

Se $x_i\sigma$ precede $x_j\sigma$ nella bwt-matrix allora $x_i\sigma < x_j\sigma$.

Proprietà LF

Se $x_i\sigma$ precede $x_j\sigma$ nella bwt-matrix allora $x_i\sigma < x_j\sigma$.

La parola $x_i\sigma$ non può essere prefisso proprio di $x_j\sigma$ (le due parole hanno la stessa lunghezza).

Se $x_i\sigma$ precede $x_j\sigma$ nella bwt-matrix allora $x_i\sigma < x_j\sigma$.

La parola $x_i\sigma$ non può essere prefisso proprio di $x_j\sigma$ (le due parole hanno la stessa lunghezza).

Quindi $x_i\sigma = zby$, $x_j\sigma = zb'y'$, con $b < b'$.

Se $x_i\sigma$ precede $x_j\sigma$ nella bwt-matrix allora $x_i\sigma < x_j\sigma$.

La parola $x_i\sigma$ non può essere prefisso proprio di $x_j\sigma$ (le due parole hanno la stessa lunghezza).

Quindi $x_i\sigma = zby$, $x_j\sigma = zb'y'$, con $b < b'$.

Ne consegue $|y| \geq 1$. Infatti se y fosse la parola vuota, allora $b = \sigma$ da cui $x_i = z = x_j$ (x_i, x_j hanno la stessa lunghezza). Questo implicherebbe che anche y' è la parola vuota e $b' = \sigma = b$, assurdo. Analogamente si può provare che $|y'| \geq 1$.

Se $x_i\sigma$ precede $x_j\sigma$ nella bwt-matrix allora $x_i\sigma < x_j\sigma$.

La parola $x_i\sigma$ non può essere prefisso proprio di $x_j\sigma$ (le due parole hanno la stessa lunghezza).

Quindi $x_i\sigma = zby$, $x_j\sigma = zb'y'$, con $b < b'$.

Ne consegue $|y| \geq 1$. Infatti se y fosse la parola vuota, allora $b = \sigma$ da cui $x_i = z = x_j$ (x_i, x_j hanno la stessa lunghezza). Questo implicherebbe che anche y' è la parola vuota e $b' = \sigma = b$, assurdo. Analogamente si può provare che $|y'| \geq 1$.

Quindi, per il Lemma di Levi, $y = y_1\sigma$, $y' = y_1'\sigma$ e $x_i = zby_1$, $x_j = zb'y_1'$.

Se $x_i\sigma$ precede $x_j\sigma$ nella bwt-matrix allora $x_i\sigma < x_j\sigma$.

La parola $x_i\sigma$ non può essere prefisso proprio di $x_j\sigma$ (le due parole hanno la stessa lunghezza).

Quindi $x_i\sigma = zby$, $x_j\sigma = zb'y'$, con $b < b'$.

Ne consegue $|y| \geq 1$. Infatti se y fosse la parola vuota, allora $b = \sigma$ da cui $x_i = z = x_j$ (x_i, x_j hanno la stessa lunghezza). Questo implicherebbe che anche y' è la parola vuota e $b' = \sigma = b$, assurdo. Analogamente si può provare che $|y'| \geq 1$.

Quindi, per il Lemma di Levi, $y = y_1\sigma$, $y' = y_1'\sigma$ e $x_i = zby_1$, $x_j = zb'y_1'$.

Questo implica $x_i < x_j$ e quindi, per una delle proprietà dell'ordine lessicografico, $\sigma x_i < \sigma x_j$, cioè σx_i precede σx_j nella bwt-matrix.

La permutazione f la costruiamo tenendo conto di questa proprietà.

Cioè l'immagine della prima occorrenza di σ in F sarà la prima occorrenza di σ in L, l'immagine della seconda occorrenza di σ in F sarà la seconda occorrenza di σ in L e così via.

In generale, per ogni carattere σ in F, l'immagine della i-esima occorrenza di σ in F sarà l'i-esima occorrenza di σ in L.

Esempio 1

$w = banana$, $\mathrm{bwt}(w) = (nnbaaa, 4)$.

	F
1	a
2	a
3	a
4	b
5	n
6	n

	L
n	1
n	2
b	3
a	4
a	5
a	6

$$f = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 4 & 5 & 6 & 3 & 1 & 2 \end{pmatrix}$$

$$(a, a, a, b, n, n) = (b_1, b_2, b_3, b_4, b_5, b_6)$$

$$= (c_{f(1)}, c_{f(2)}, c_{f(3)}, c_{f(4)}, c_{f(5)}, c_{f(6)})$$

Esempio 2

$w = abraca$, $\mathrm{bwt}(w) = (caraab, 2)$.

	F
1	a
2	a
3	a
4	b
5	c
6	r

	L
c	1
a	2
r	3
a	4
a	5
b	6

$$f = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 4 & 5 & 6 & 1 & 3 \end{pmatrix}$$

$$(a, a, a, b, c, r) = (b_1, b_2, b_3, b_4, b_5, b_6)$$

$$= (c_{f(1)}, c_{f(2)}, c_{f(3)}, c_{f(4)}, c_{f(5)}, c_{f(6)})$$

Un'altra proprietà

Per ricostruire w da $\text{bwt}(w) = (L, I)$, dopo aver definito F e f, dobbiamo utilizzare un'altra proprietà.

Un'altra proprietà

Per ogni $i \in \{1, \ldots, n\}, i \neq I$, $L(i)$ *precede* $F(i)$ *in* w.

Un'altra proprietà

Per ogni $i \in \{1, \ldots, n\}, i \neq I$, $L(i)$ *precede* $F(i)$ *in* w.

Perché?

Un'altra proprietà

Per ogni $i \in \{1, \ldots, n\}, i \neq I$, $L(i)$ *precede* $F(i)$ *in* w.

Perché?

Per ogni $i \in \{1, \ldots, n\}, i \neq I$, la riga i-esima è una coniugata di $w = a_1 \cdots a_n$, cioè è una stringa della forma

$$a_j \cdots a_n a_1 \cdots a_{j-1}$$

Un'altra proprietà

Per ogni $i \in \{1, \ldots, n\}, i \neq I$, $L(i)$ *precede* $F(i)$ *in* w.

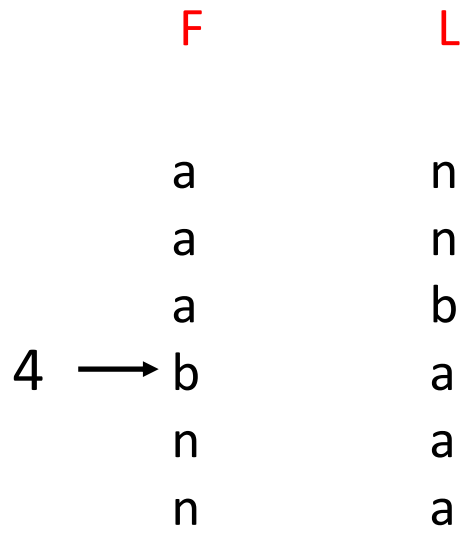
Perché?

Per ogni $i \in \{1, \ldots, n\}, i \neq I$, la riga i-esima è una coniugata di $w = a_1 \cdots a_n$, cioè è una stringa della forma

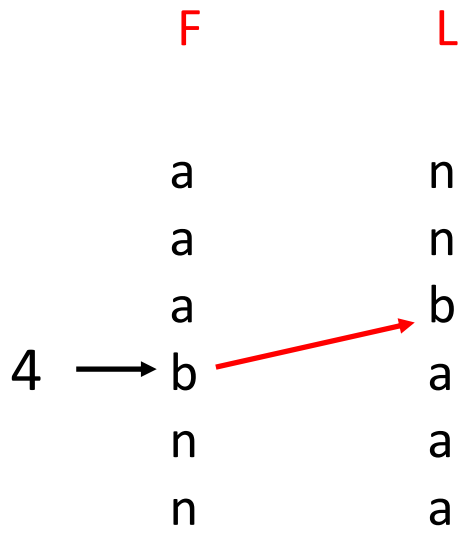
$$a_j \cdots a_n a_1 \cdots a_{j-1}$$

Ovviamente $L(i) = a_{j-1}$ precede $F(i) = a_j$ in w.

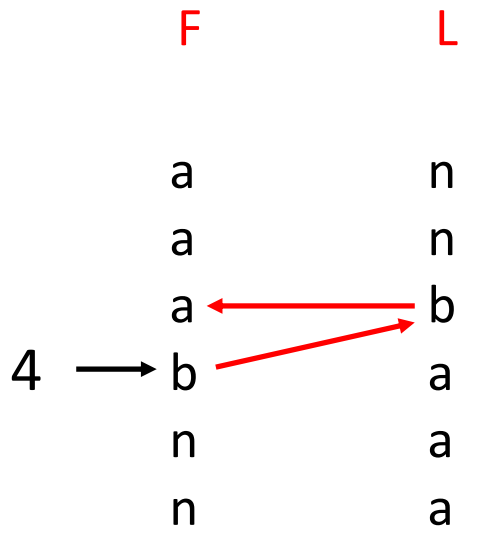
Reversibilità della BWT



Reversibilità della BWT

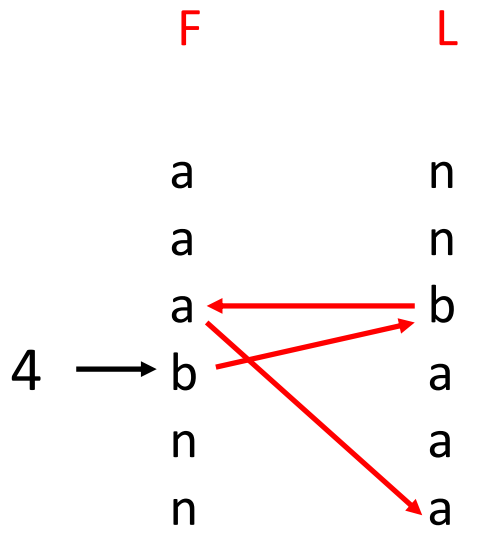


Reversibilità della BWT



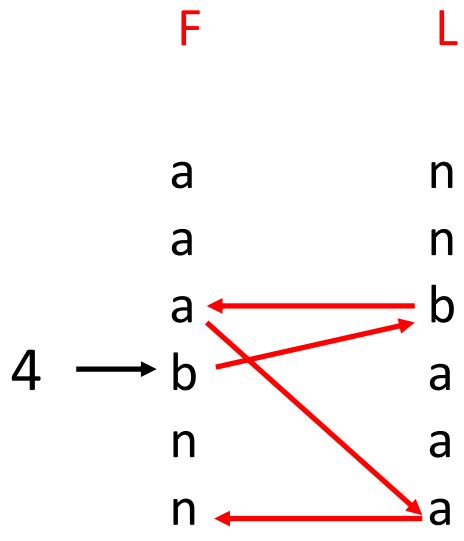
ba

Reversibilità della BWT



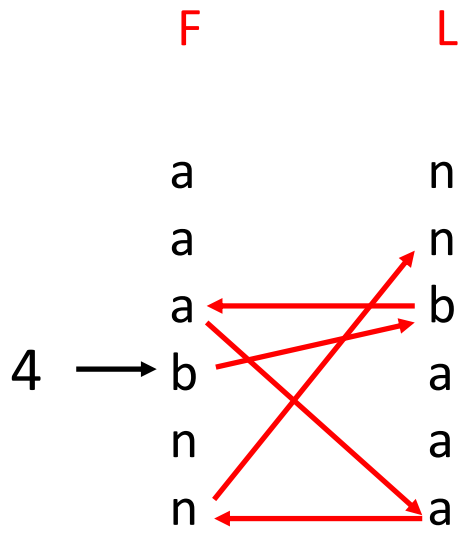
ba

Reversibilità della BWT



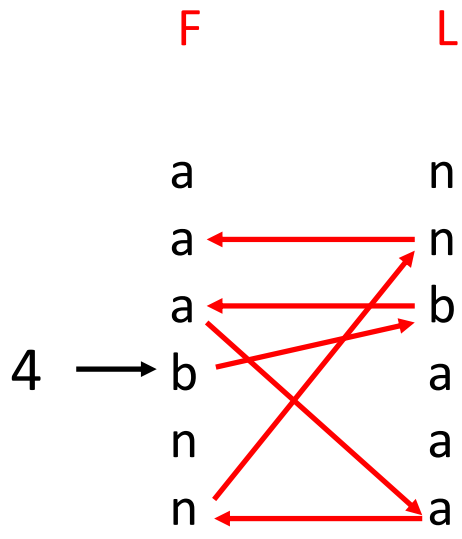
ban

Reversibilità della BWT



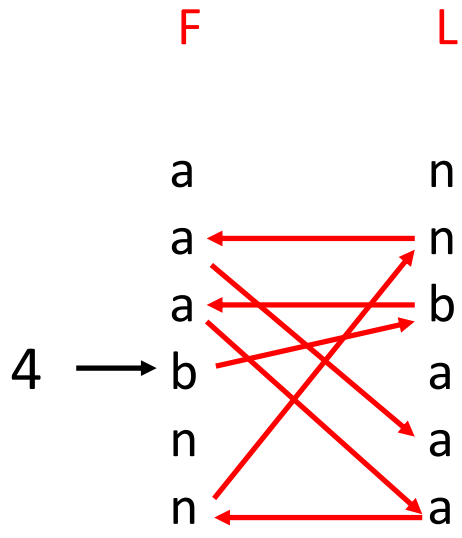
ban

Reversibilità della BWT



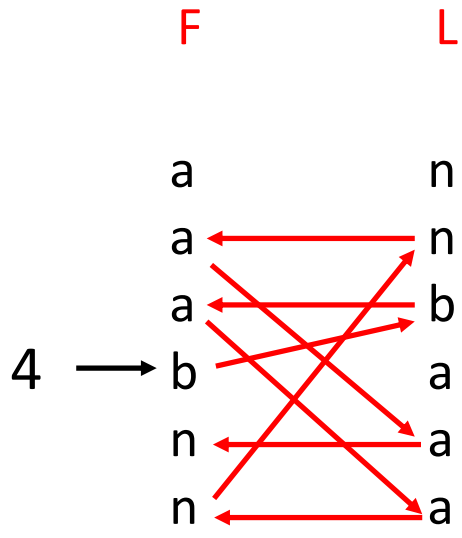
bana

Reversibilità della BWT



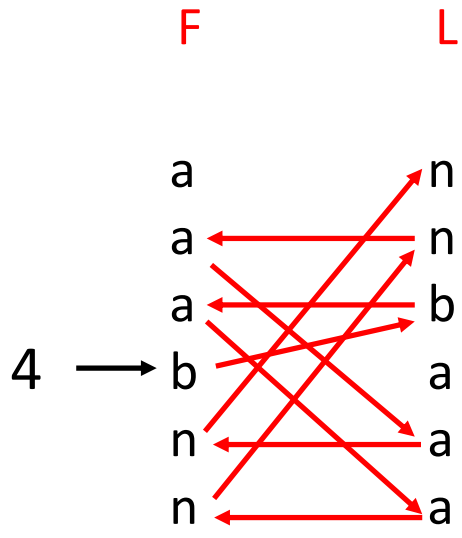
bana

Reversibilità della BWT



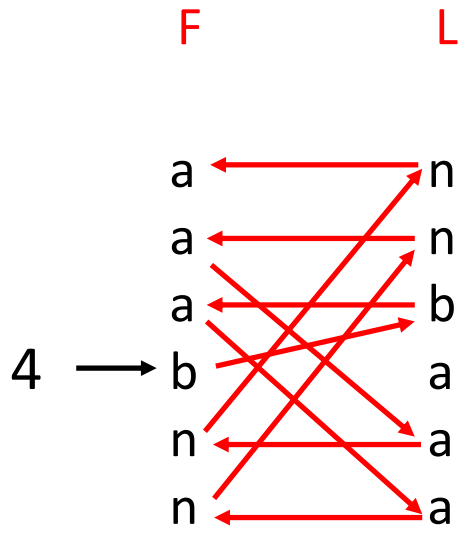
banan

Reversibilità della BWT



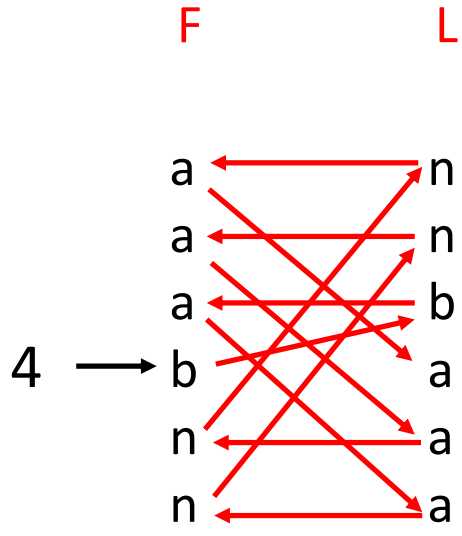
banan

Reversibilità della BWT



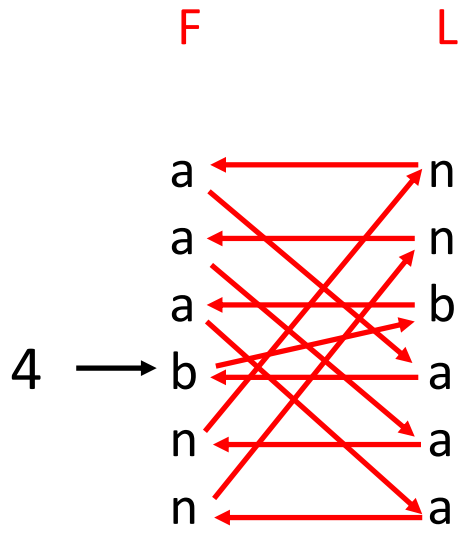
banana

Reversibilità della BWT



banana

Reversibilità della BWT



banana

Reversibilità della BWT

Per ricostruire w da $\text{bwt}(w) = (L, I)$, dopo aver definito F e f, utilizziamo il seguente algoritmo ricorsivo, in cui $w = a_1 \cdots a_n$, $F = (b_1, \ldots, b_n)$ e $L = (c_1, \ldots, c_n)$.

Per ricostruire w da $\text{bwt}(w) = (L, I)$, dopo aver definito F e f, utilizziamo il seguente algoritmo ricorsivo, in cui $w = a_1 \cdots a_n$, $F = (b_1, \ldots, b_n)$ e $L = (c_1, \ldots, c_n)$.

PASSO BASE: La prima lettera a_1 di w è quella di indice I in F.

Reversibilità della BWT

Per ricostruire w da $\text{bwt}(w) = (L, I)$, dopo aver definito F e f, utilizziamo il seguente algoritmo ricorsivo, in cui $w = a_1 \cdots a_n$, $F = (b_1, \ldots, b_n)$ e $L = (c_1, \ldots, c_n)$.

PASSO BASE: La prima lettera a_1 di w è quella di indice I in F.

PASSO RICORSIVO: Se la lettera a_i in posizione i di w occupa la posizione di indice j in F, cioè $a_i = b_j$, allora la lettera a_{i+1} in posizione $i+1$ di w è $a_{i+1} = b_h$ con $h = f(j)$, $1 \leq i \leq n-1$. Cioè cerco l'immagine di $a_i = b_j$ in L tramite la permutazione f e se $c_h = c_{f(j)}$, allora $a_{i+1} = b_h$.

Esempio 1 - continua

$$
\begin{array}{cccc}
 & a & & n \\
 & a & & n \\
 & a & & b \\
4 \to & b & & a \\
 & n & & a \\
 & n & & a
\end{array}
\qquad
f = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 4 & 5 & 6 & 3 & 1 & 2 \end{pmatrix}
$$

Esempio 1 - continua

$$
\begin{array}{rcc}
 & a & n \\
 & a & n \\
 & a & b \\
4 \rightarrow & b & a \\
 & n & a \\
 & n & a
\end{array}
\qquad
f = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 4 & 5 & 6 & 3 & 1 & 2 \end{pmatrix}
$$

La prima lettera è quella di indice 4 in F, cioè b.

Esempio 1 - continua

$$
\begin{array}{rll}
 & a & n \\
 & a & n \\
 & a & b \\
4 \rightarrow & b & a \\
 & n & a \\
 & n & a
\end{array}
\qquad
f = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 4 & 5 & 6 & 3 & 1 & 2 \end{pmatrix}
$$

La prima lettera è quella di indice 4 in F, cioè b.

Cerco b in L e scopro che b precede la terza occorrenza di a, ottengo ba.

Esempio 1 - continua

$$
\begin{array}{rll}
 & a & n \\
 & a & n \\
 & a & b \\
4 \rightarrow & b & a \\
 & n & a \\
 & n & a
\end{array}
\qquad
f = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 4 & 5 & 6 & 3 & 1 & 2 \end{pmatrix}
$$

La prima lettera è quella di indice 4 in F, cioè b.

Cerco b in L e scopro che b precede la terza occorrenza di a, ottengo ba.

Vado alla terza occorrenza di a in L e scopro che precede la seconda occorrenza di n, ottengo ban

Esempio 1 - continua

$$\begin{array}{rcc} & a & n \\ & a & n \\ & a & b \\ 4 \rightarrow & b & a \\ & n & a \\ & n & a \end{array} \qquad f = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 4 & 5 & 6 & 3 & 1 & 2 \end{pmatrix}$$

La prima lettera è quella di indice 4 in F, cioè b.

Cerco b in L e scopro che b precede la terza occorrenza di a, ottengo ba.

Vado alla terza occorrenza di a in L e scopro che precede la seconda occorrenza di n, ottengo ban

poi $bana$, $banan$ e infine $banana$.

Esempio 2 - continua

$$
\begin{array}{cll}
 & a & c \\
2 \to & a & a \\
 & a & r \\
 & b & a \\
 & c & a \\
 & r & b
\end{array}
\qquad
f = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 4 & 5 & 6 & 1 & 3 \end{pmatrix}
$$

Esempio 2 - continua

$$
\begin{array}{rll}
 & a & c \\
2 \rightarrow & a & a \\
 & a & r \\
 & b & a \\
 & c & a \\
 & r & b
\end{array}
\qquad
f = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 4 & 5 & 6 & 1 & 3 \end{pmatrix}
$$

$a_1 = F(2) = a$, $f(2) = 4$, $a = L(4)$ è seguita da $F(4) = b$.

Esempio 2 - continua

$$
\begin{array}{cll}
 & a & c \\
2 \rightarrow & a & a \\
 & a & r \\
 & b & a \\
 & c & a \\
 & r & b
\end{array}
\qquad
f = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 4 & 5 & 6 & 1 & 3 \end{pmatrix}
$$

$a_1 = F(2) = a$, $f(2) = 4$, $a = L(4)$ è seguita da $F(4) = b$.

$a_2 = F(4) = b$, $f(4) = 6$, $b = L(6)$ è seguita da $r = F(6)$.

Esempio 2 - continua

$$
\begin{array}{cll}
 & a & c \\
2 \rightarrow & a & a \\
 & a & r \\
 & b & a \\
 & c & a \\
 & r & b
\end{array}
\qquad
f = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 4 & 5 & 6 & 1 & 3 \end{pmatrix}
$$

$a_1 = F(2) = a$, $f(2) = 4$, $a = L(4)$ è seguita da $F(4) = b$.

$a_2 = F(4) = b$, $f(4) = 6$, $b = L(6)$ è seguita da $r = F(6)$.

$a_3 = F(6) = r$, $f(6) = 3$, $r = L(3)$ è seguita da $a = F(3)$.

Esempio 2 - continua

$$
\begin{array}{cll}
 & a & c \\
2 \to & a & a \\
 & a & r \\
 & b & a \\
 & c & a \\
 & r & b
\end{array}
\qquad
f = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 4 & 5 & 6 & 1 & 3 \end{pmatrix}
$$

$a_1 = F(2) = a$, $f(2) = 4$, $a = L(4)$ è seguita da $F(4) = b$.

$a_2 = F(4) = b$, $f(4) = 6$, $b = L(6)$ è seguita da $r = F(6)$.

$a_3 = F(6) = r$, $f(6) = 3$, $r = L(3)$ è seguita da $a = F(3)$.

$a_4 = F(3) = a$, $f(3) = 5$, $a = L(5)$ è seguita da $c = F(5)$.

Esempio 2 - continua

$$
\begin{array}{cll}
 & a & c \\
2 \to & a & a \\
 & a & r \\
 & b & a \\
 & c & a \\
 & r & b
\end{array}
\qquad
f = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 4 & 5 & 6 & 1 & 3 \end{pmatrix}
$$

$a_1 = F(2) = a$, $f(2) = 4$, $a = L(4)$ è seguita da $F(4) = b$.

$a_2 = F(4) = b$, $f(4) = 6$, $b = L(6)$ è seguita da $r = F(6)$.

$a_3 = F(6) = r$, $f(6) = 3$, $r = L(3)$ è seguita da $a = F(3)$.

$a_4 = F(3) = a$, $f(3) = 5$, $a = L(5)$ è seguita da $c = F(5)$.

$a_5 = F(5) = c$, $f(5) = 1$, $c = L(1)$ è seguita da $a = F(1)$.

Esempio 2 - continua

$$
\begin{array}{cll}
 & a & c \\
2 \to & a & a \\
 & a & r \\
 & b & a \\
 & c & a \\
 & r & b
\end{array}
\qquad
f = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 4 & 5 & 6 & 1 & 3 \end{pmatrix}
$$

$a_1 = F(2) = a$, $f(2) = 4$, $a = L(4)$ è seguita da $F(4) = b$.

$a_2 = F(4) = b$, $f(4) = 6$, $b = L(6)$ è seguita da $r = F(6)$.

$a_3 = F(6) = r$, $f(6) = 3$, $r = L(3)$ è seguita da $a = F(3)$.

$a_4 = F(3) = a$, $f(3) = 5$, $a = L(5)$ è seguita da $c = F(5)$.

$a_5 = F(5) = c$, $f(5) = 1$, $c = L(1)$ è seguita da $a = F(1)$.

$a_6 = F(1) = a$.

Esempio 2 - continua

$$
\begin{array}{cll}
 & a & c \\
2 \to & a & a \\
 & a & r \\
 & b & a \\
 & c & a \\
 & r & b
\end{array}
\qquad
f = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 4 & 5 & 6 & 1 & 3 \end{pmatrix}
$$

$a_1 = F(2) = a$, $f(2) = 4$, $a = L(4)$ è seguita da $F(4) = b$.

$a_2 = F(4) = b$, $f(4) = 6$, $b = L(6)$ è seguita da $r = F(6)$.

$a_3 = F(6) = r$, $f(6) = 3$, $r = L(3)$ è seguita da $a = F(3)$.

$a_4 = F(3) = a$, $f(3) = 5$, $a = L(5)$ è seguita da $c = F(5)$.

$a_5 = F(5) = c$, $f(5) = 1$, $c = L(1)$ è seguita da $a = F(1)$.

$a_6 = F(1) = a$.

Ottengo *ab*, *abr*, *abra*, *abrac* e infine *abraca*.

Versione iterativa dell'algoritmo ricorsivo

Sia $w = a_1 \cdots a_n$. Allora

$$a_i = F[f^{i-1}(I)],\ 1 \leq i \leq n$$

dove $f^0(x) = x$ e $f^j(x) = f(f^{j-1}(x))$, $1 \leq j \leq n-1$.

Estensioni della BWT

Data una stringa w, la BWT di w è univocamente definita.

Inoltre, abbiamo dimostrato che, se w e w' sono due stringhe diverse allora $\text{bwt}(w) \neq \text{bwt}(w')$.

Equivalentemente, la funzione che associa a una stringa w la coppia $\text{bwt}(w)$ è iniettiva.

Sfortunatamente vi sono esempi che mostrano che tale funzione non è suriettiva.

Sono state definite estensioni della BWT. Una di esse è basata su un risultato di combinatoria delle parole del 1993 dovuto a Gessel e Reutenauer. Essa conduce a una funzione biettiva ed ha applicazioni in bioinformatica.