Trasformata di Burrows-Wheeler
&
FM-index

Stefano Beretta

La *trasformata di Burrows-Wheeler* (BWT) [1] è un algoritmo che opera su stringa e viene sfruttato da molti algoritmi di compressione dati. Tale procedura consente di trasformare un testo T in una nuova stringa che risulta più semplice da comprimere.

Dato in input un testo T, la procedura $BWT(T)$ consiste nei seguenti passi:

1. Aggiungere in coda a T un carattere speciale \$ che è lessicograficamente minore di tutti i simboli dell'alfabeto considerato;

2. Creare un matrice $\mathcal{M}_T$ (solo concettuale) con tutte le possibili rotazioni del testo T\$;

3. Ordinare lessicograficamente le righe della matrice $\mathcal{M}_T$;

4. Costruire la trasformata del testo, $BWT(T)$ prendendo l'ultima colonna L della matrice $\mathcal{M}_T$ ordinata.

```
                F     L
                $acaacg
                aacg$ac
                acaacg$
acaacg$  -->    acg$aca  -->  gc$aaac
                caacg$a
                cg$acaa
                g$acaac
```

Si noti che la stringa così ottenuta $BWT(T)$, così come ogni colonna della matrice $\mathcal{M}_T$, costituisce una permutazione della stringa T\$ (stringa T con accodato il carattere \$). In particolare la prima colonna della matrice $\mathcal{M}_T$ denotata da F costituisce un ordinamento lessicografico dei simboli di

[1]Burrows M, Wheeler DJ: *A Block Sorting Lossless Data Compression Algorithm.* Technical Report 124 Palo Alto, CA: Digital Equipment Corporation; 1994.

$T\$$. Si consideri inoltre che ordinare le righe della matrice $\mathcal{M}_T$ corrisponde ad ordinare di suffissi di T data la presenza del carattere speciale \$.

Le rotazioni cicliche della stringa T e l'ordinamento delle righe di $\mathcal{M}_T$ sono necessarie per definire due proprietà fondamentali della trasformata BWT:

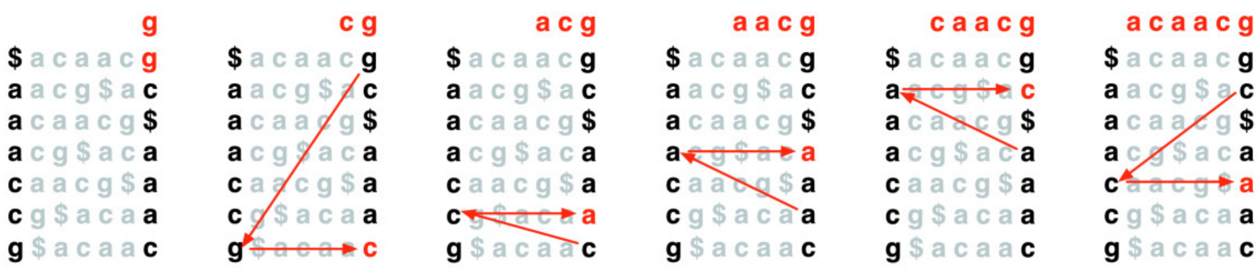
Proprietà 1. *Data la* i*-esima riga di $\mathcal{M}_T$, allora il suo ultimo carattere $L[i]$ precede il suo primo carattere $F[i]$ nella stringa originale T. La situazione è la seguente: $T = \ldots L[i]F[i]\ldots$.*

Proprietà 2 (LF-mapping)**.** *Sia $L[i] = c$ e sia r_i il "rank" della riga $\mathcal{M}_T[i]$ tra tutte le righe che terminano con il carattere c. Sia $\mathcal{M}_T[j]$ la r_i-esima riga di $\mathcal{M}_T$ che inizia con c. Allora il carattere corrispondente a $L[i]$ nella prima colonna F è posizionato in $F[j]$. Questa proprietà prende il nome di* LF-mapping *ed è espresso da $LF[i] = j$.*

In pratica nelle colonne L ed F di $\mathcal{M}_T$, viene preservato l'ordine dei caratteri uguali; così la i-esima occorrenza del carattere c nella colonna L corrisponderà alla i-esima occorrenza del carattere c nella colonna F.

Applicando ricorsivamente l'*LF-mapping* a partire dal primo carattere di $BWT(T)$ (ovvero quello che ha come successivo, nel testo $T\$$, il carattere speciale \$ e di conseguenza si trova in ultima posizione del testo originale T), è possibile ricostruire a ritroso il testo originale T.

Il costo computazionale per calcolare la trasformata di *Burrows-Wheeler* è di $O(|T|)$ dove $|T|$ rappresenta la lunghezza del testo T. Questo perchè vi è una relazione tra la trasformata BWT e i *suffix array*, in quanto effettuare l'ordinamento delle righe della matrice $\mathcal{M}_T$, corrisponde ad effettuare on ordinamento di tutti i suffissi del testo T (che è proprio quello che succede nella costruzione di un suffix array).



Oltre che per la ricostruzione del testo originale T, la proprietà di *LF-mapping* viene sfruttata per la ricerca di pattern all'interno di T. L'algoritmo utilizzato a tale scopo prende il nome di *FM-index*[2].

[2]Ferragina, P. and Manzini, G.: *Opportunistic data structures with applications.* Proceedings of the 41st Annual Symposium on Foundations of Computer Science, 2000

La procedura di *Exact-match* per la ricerca del pattern $P[1,p]$ in T con *FM-index* mantiene un range di righe della matrice $\mathcal{M}_T$ associati a suffissi sempre maggiori (ad ogni step dell'algoritmo) del pattern P. Ad ogni iterazione questo range (delimitato da due indici) o si restringe o rimane invariato (tale range fa riferimento al numero di righe, non alla posizione nella matrice che potrebbe cambiare). Quando l'algoritmo termina, le righe di $\mathcal{M}_T$ che iniziano con il pattern P nel range trovato, corrispondono ad occorrenze esatte di tale pattern nel testo. Se tale range è vuoto significa che il pattern non è contenuto nel testo.

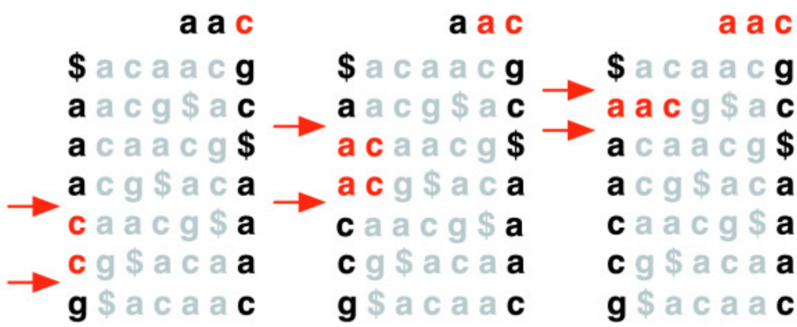
L'algoritmo riportato di seguito illustra la procedura di *Exact-match* del pattern $P[1,p]$ nel testo T (a cui è stata applicata la BWT). Lo pseudocodice è stato espresso facendo riferimento al vettore $C[c]$ che mappa il carattere c al numero totale di occorrenze in $BWT(T)$ di caratteri lassicograficamente minori di c; è presenta inoltre una funzione $Occ(c,r)$ che calcola il numero di occorrenze di c in $BWT(T)$ fino alla riga r della matrice (escluso quest'ultima).

Algorithm 1: Exact-match

Data: Pattern $P[1,p]$

$c \leftarrow P[p]$;
$sp \leftarrow C[c]+1$;
$ep \leftarrow C[c+1]+1$;
$i \leftarrow p-1$;
while $sp < ep$ *and* $i \geq 1$ **do**
  $C \leftarrow P[i]$;
  $sp \leftarrow C[c] + Occ(c, sp) + 1$;
  $ep \leftarrow C[c] + Occ(c, ep) + 1$;
  $i \leftarrow i-1$;
return *sp*,*ep*;

Nella successiva figura è mostrato un esempio di esecuzione di tale algoritmo in cui la due frecce indicano i due estremi del range (*sp* e *ep*).



Una volta trovato il pattern P nella matrice, bisogna localizzarlo all'interno di T, ovvero trovare l'offset rispetto l'inizio del testo. Per fare ciò bisognerebbe continuare a ricostruire a ritroso il testo tramite l'*LF-mapping* partendo dalla riga in cui è stata trovata un'occorrenza del pattern in modo da calcolare il numero di caratteri necessari (ovvero l'offset). In alternativa è possibile pre-calcolare degli offset in alcune posizioni della matrice consentendo quindi di non tornare a ritroso fino all'inizio del testo, ma solo fino a trovare uno di questi indici pre-calcolati.