Relazione Progetto
Elementi di Bioinformatica
Long Bitap

Davide Cozzi
829827
d.cozzi@campus.unimib.it

Bitap

Nell'analisi un bit rappresentato in colonna presenta il bit più significativo, MSB, in basso
Innazitutto si descrive l'algoritmo di base, funzionante per pattern lunghi al massimo quanto una *word*, `w` della cpu.
Si hanno in input un *pattern* `P` di lunghezza `p` e un *testo* `T` di lunghezza `t`. Si ha quindi `p <= w`.
A livello teorico si costruisce una **matrice booleana** `D`, di dimensioni $\texttt{p} \times \texttt{t}$, e si stabiliscono due indici:

1. `i` che itera sul pattern
2. `j` che itera sul testo

Si ha che la generica posizione di indici `i`, `j` nella matrice è 1 sse i primi `i` caratteri del pattern matchano un numero `i` di caratteri del testo terminanti all'indice `j`.
Si ha quindi la seguente **equazione di ricorrenza**:

$$D[i,j] = \begin{cases} 1 & \text{sse } \texttt{P}[1..\texttt{i}] = \texttt{T}[\texttt{j} - \texttt{i} + 1..\texttt{j}] \\ 0 & \text{altrimenti} \end{cases}$$

Nell'`i`-sima riga si ha che le occorrenze di 1 indicano i punti nel testo dove termina una copia di $\texttt{P}[1..\texttt{i}]$.
Invece la `j`-sima colonna mostra tutti i prefissi del testo che finiscono nella posizione `j` del testo.

Nell'ultima riga della matrice si ha la soluzione, ovvero si ha 1 dove termina un match del pattern nel testo.
In termini pratici questo algoritmo può essere costruito mediante operazioni **bit a bit** in quanto le singole colonne della matrice teorica possono essere viste come numeri in rappresentazione binaria.
Per procedere si ha una fase di preprocessamento in cui si costruice un array `U`, di lunghezza pari a quella dell'alfabeto in uso, che contiene in posizione `k` il binario rappresentante le occorrenze del carattere `k` nel pattern. Nel pattern il vettore `U` viene definito di lunghezza `CHAR_MAX`, ovvero 127, per usare la tabella ASCII a 7 bit. Per costruire tale array si procede inizializzando tutte le celle a 0. Si itera poi lungo il pattern aggiornando `U` nella posizione del carattere preso in considerazione del pattern facendo l'`or` tra l'attuale contenuto di `U` in quella posizione e il numero la cui rappresentazione binaria presenta 1 solo nella posizione di indice `i` (questo comportamento è ottenibile con il *left-shift* di 1 di *posizioni*). Dopo aver iterato su tutto il pattern ottengo il vettore `U` correttamente caricato.

Esempio 1. *Immaginiamo un pattern semplice: caac.*
Il primo carattere è `c` *e* `i` *è in posizione 0.* `U` *è ancora caricato con soli zeri. Quindi si ha:*

$$U[\texttt{c}] = U[99] = \begin{matrix}0\\0\\0\end{matrix} \vee \begin{matrix}0&1&1\\0&0&0\\0&0&0\end{matrix} = \begin{matrix}0&1&1\\0&0&0\\0&0&0\end{matrix}$$

proseguo con `a` *e shifto* `i` *di 1:*

$$U[\texttt{a}] = U[97] = \begin{matrix}0\\0\\0\end{matrix} \vee \begin{matrix}0&0&0\\1&0&0\\0&0&0\end{matrix} = \begin{matrix}0&0&0\\1&0&0\\0&0&0\end{matrix}$$

il carattere successivo è ancora `a`*:*

$$U[\texttt{a}] = U[97] = \begin{matrix}1\\0\\0\end{matrix} \vee \begin{matrix}0&0&0\\0&1&0\\0&0&0\end{matrix} = \begin{matrix}1&1&1\\0&1&0\\0&0&0\end{matrix}$$

e infine trovo ancora `c`*:*

$$U[\texttt{c}] = U[99] = \begin{matrix}0\\0\\0\end{matrix} \vee \begin{matrix}1&0&1\\0&0&0\\0&0&0\end{matrix} = \begin{matrix}1&0&1\\0&0&0\\0&0&0\end{matrix}$$

Alla fine quindi $U[\texttt{c}] = 1001|_2 = 9$ *e* $U[\texttt{a}] = 0110|_2 = 6$

L'algortimo prosegue inizializzando la prima colonna a 1 se testo e pattern condividono il primo cararattere, 0 altrimenti.
Si procede poi col calcolo delle colonne successive alla prima sfruttando la colonna precedente. Si procede con un *left-shift* del valore rappresentante la colonna precedente con l'aggiunta di un 1 in testa. Si procede poi con l'`and` tra il risultato appena ottenuto e il valore di `U` nella posizione del carattere che sto considerando. Si controlla infine ogni valore rappreentante una colonna vedendo se presenta 1 nell'ultimo bit. Per ottenere questo risultato si procede con il *left-shift* di uno di un'unità pari alla lunghezza del pattern meno uno e all'`and` con il valore rappresentante la colonna. Mi verra infatti restituito un binario avente valore o 0 o 2^{p-1} (nel caso di word grandi 3 avrei $100|_2 = 4$ avendo quindi 1 nell'ipotetica ultima riga della matrice, avendo quindi un match).
Il limite di queto algoritmo è hardware e consiste nella rappresentazione (e quindi anche nelle operazioni) su binari oltre il numero di bit della word.

Long Bitap