

## Algoritmi EURISTICI di allineamento

Sono nati insieme alle banche dati, con lo scopo di permettere una ricerca per similarità rapida anche se meno accurata contro le migliaia di sequenze depositate.

Attualmente i programmi più utilizzati sono:

**FASTA**: Lipman & Pearson (1985)

**BLAST**: Altshul (1990)

# FASTA

L'algoritmo implementato in FASTA si basa su una strategia di **INDICIZZAZIONE** delle parole: la proteina QUERY viene spezzettata in parole di lunghezza ktup (k-tuples) e la query viene così indicizzata (cioè si memorizzano solo gli indici, non la coppia).

**TFDERILGVQQTFWECIKGD**

1 . TF  
2 . FD  
3 . DE  
4 . ER  
5 . RI  
6 . IL  
7 . LG  
.....  
20 . GD

**ktup = 2**

Maggiore è il valore ktup più rapida e meno accurata sarà la ricerca. Per una proteina a ktup = 2 potrò avere al massimo  $20^2 = 400$  combinazioni, sempre.

Ogni SUBJECT della banca dati viene consultata allo stesso modo, ma anzichè indicizzarla, viene consultato l'indice della query per vedere se e dove si posiziona il match, e **viene memorizzata la diagonale corrispondente** (indice della query –indice della subject).

		----- QUERY ----->									
		0	1	2	3	4	5	6	7	8	9
----- SUBJECT ----->	-1										
	-2										
	-3										
	-4										
	-5										

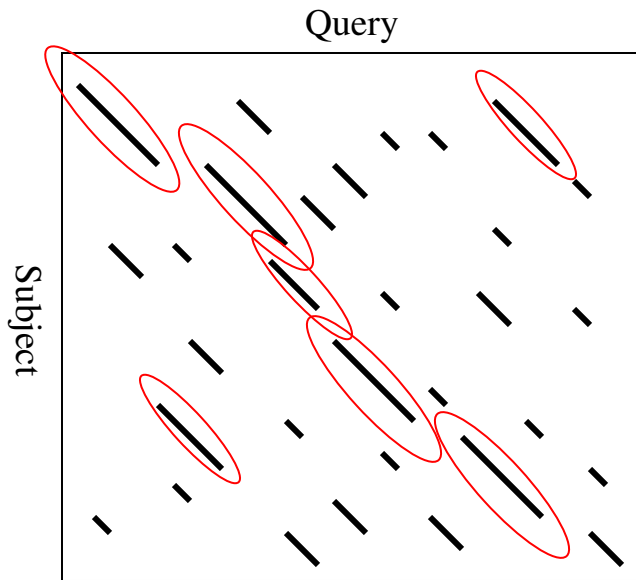
Se alla riga successiva della matrice, cioè alla parola successiva della subject, c'è corrispondenza sulla stessa diagonale, la parola viene memorizzata e quindi man mano si allunga la diagonale.

Considerando una matrice di sostituzione, vengono riconosciute le migliori diagonali come i segmenti più lunghi e che totalizzano quindi lo score maggiore.

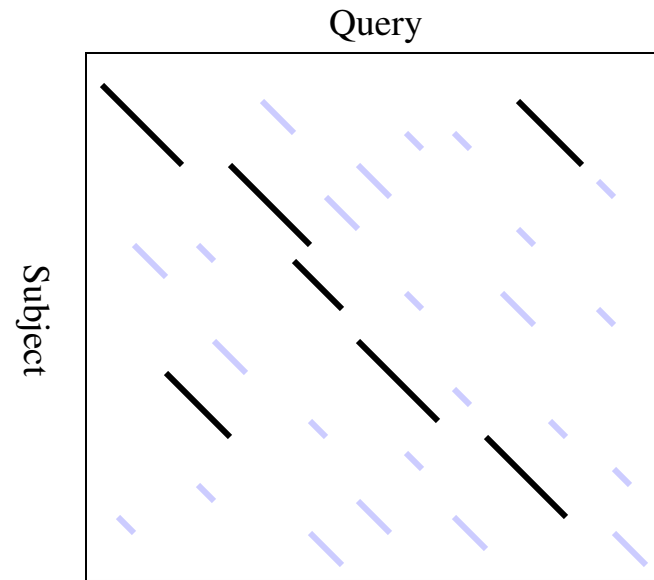
Il programma ripete queste operazioni per ogni subject della banca dati, identificando le **Best Initial Regions** i cui punteggi sono chiamati **Init1**. Con questi fa una graduatoria per decidere su quante e quali sequenze procedere. La scelta delle subject più adatte è stata fatta. Da ora procede con un numero molto minore di proteine subject.

Ora il programma cerca di congiungere ogni best initial region della subject confermata utilizzando i parametri di penalty per i gaps. Le regioni vengono allungate e avranno un nuovo score, detto **InitN**.

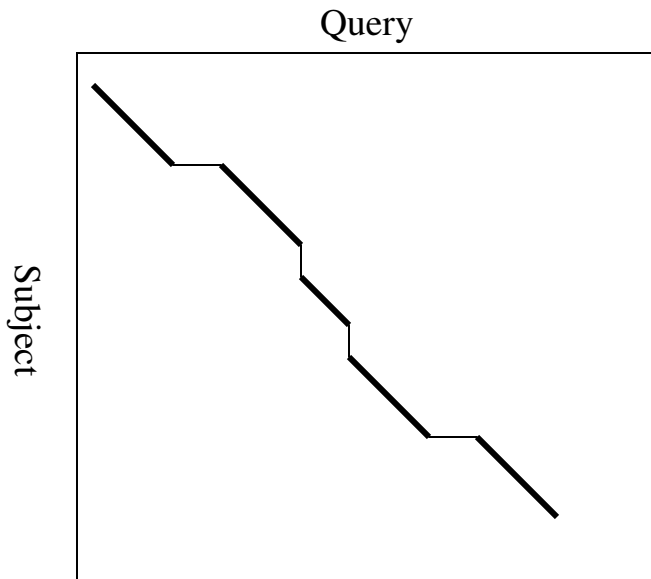
Alla fine tutta la regione della subject riconosciuta come simile viene allineata mediante l'algoritmo di Smith e Waterman, tenendo una finestra di analisi intorno alla diagonale principale abbastanza stretta (< 20 residui). Lo score definitivo è definito **Opt**.



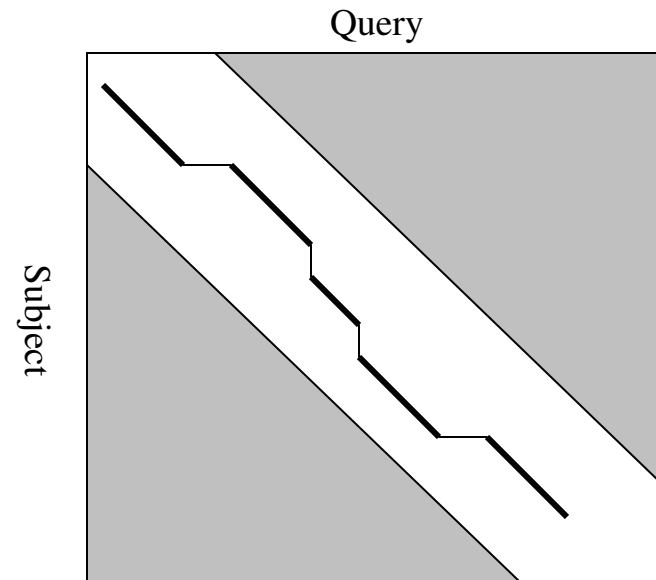
Fase 1 : Indicizzazione



Fase 2 : Calcolo degli Init1,  
generazione delle bits



Fase 3 : Definizione delle sequenze  
da approfondire, calcolo degli  
InitN



Fase 4 : Smith-Waterman (Opt)

# BLAST

E' basato anch'esso sull'indicizzazione delle parole, ma l'utilizzo degli indici è poi molto diverso. L'algoritmo è diviso in tre parti:

1- Creazione di un elenco di parole di lunghezza  $W$  dalla query e creazione di  $w$ -mers, cioè parole di lunghezza  $w$  che diano, secondo una matrice di sostituzione uno score  $> T$  se allineati sulla query stessa.

TFDER **LSH**GVQQTFWECIKGD

$w = 3$

VSH = 16

$t = 13$

ISH = 14

LAH = 13

LTH = 13

LSH = 13

Per ogni parola vengono generati anche tutti i possibili  $w$ -mers, quindi ci sono molte più parole che in FASTA.

2 - **Ricerca** di hits in banca dati **per ogni w-mer**, segnando ogni volta che un match è stato trovato e che la corrispondenza è alla parola, non al suo w-mer. Ottengo così una lista di proteine in cui è stata trovata una corrispondenza con i frammenti della query.

3 - Estensione di ogni hit verso entrambe le direzioni **senza inserimento di gap**, finchè il loro score non scende sotto S. Si ottengono regioni dette **HSP** (High-scoring Segment Pair). In realtà anche se lo score scende sotto S, ma solo per alcuni residui, e poi risale, l'HSP può ancora allungarsi. Il parametro X dice la quantità di perdita di score massima tollerabile se si prosegue con l'allungamento dell'HSP.

Query:83

LMVAISNVGDTLSHLEAQNKIKSASHNLSLTLQKSK

+++AIS GT+++SH +AQ++IK+AS+ L L + ++

Subject:48

VILAISGFGTESMSHADAQDRIKAASYQLCLKIDRAE

←-----→  
**HSP**

## **BLAST ha dunque 4 parametri fondamentali:**

**W:** word size, maggiore è il numero, minore è il numero di parole generate, minore è il tempo di esecuzione. Ma la sensibilità decresce sensibilmente.

**T:** threshold, minore è il numero, maggiore è il numero di w-mers inclusi nella lista, maggiore è il tempo di esecuzione. Si ha però un incremento di sensibilità.

**S:** score, minore è il numero, maggiore sarà la lunghezza degli HSP

**X:** maggiore è il numero, più estesamente sarà osservato l'intorno di una HSP, aumentando il tempo di esecuzione.

BLAST è stato recentemente implementato con un **two-hit method**, che prevede che l'estensione delle HSP possa avvenire solo se due hits indipendenti si verificano entro un numero di residui  $A$ , senza gaps in mezzo.

La potenza di BLAST sta nella sua base statistica che permette di dire **quanto accurati** sono i suoi risultati:

**dato un  $S$**  in fatti è possibile **prevedere quanti HSP** si verificheranno in una banca dati della stessa grandezza di quella vera ma composta da proteine casuali. Questo numero è definito  $E$  (expected)

=> è molto più semplice pensare in termini di  $E$  che non in termini di  $S$ , quindi in realtà non si imposta  $S$  ma  $E$ , ed  $S$  viene calcolato automaticamente, rispattando una complessa relazione statistica tra i due valori. Tenendo  $E$  molto basso, si è quasi certi di avere solo allineamenti significativi.

## Varie versioni di BLAST

**blastp**: cerca similarità in banche dati proteiche a partire da un a query di amino acidi.

**blastn**: cerca similarità in banche dati di nucleotidi a partire da una query di nucleotidi.

**blastx**: cerca similarità in banche dati proteiche a partire da una query di nucleotidi che viene tradotta in tutti i frame.

**tblastn**: cerca similarità in banche dati di nucleotidi a partire da una query di amino acidi, traducendo in amino acidi tutti i subject della banca dati, in tutti frame.

**tblastx**: cerca similarità in banche dati di nucleotidi a partire da una query di nucleotidi, traducendo in amino acidi tutti i subject della banca dati

**gapped-blast:** porta avanti la fase di estensione delle HSP considerando la possibilità di inserzione dei gap.

**PSI-BLAST:** effettua una ricerca iterativa utilizzando le HSP per generare dei profili caratteristici della query.

**PHI-BLAST:** estensione di PSI-BLAST per la ricerca in banca dati di pattern proteici più che di query esatte.

**BL2SEQ:** adattamento di blast per l'allineamento a coppie

**MegaBLAST:** può concatenare molte queries tra loro per minimizzare il tempo di esecuzione dovuto a sequenze query troppo lunghe (è adatto a sequenze nucleotidiche molto simili tra loro)

## Meglio FASTA o BLAST ?

Gli algoritmi di Blast e Fasta sono simili come strategia, ma molto diversi nei contenuti.

Il fatto che Fasta indicizzi le query in modo esatto lo porta a ridurre di molto il numero di subject su cui lavorerà in seguito, e questo è una tappa limitante che Blast non ha grazie ai w-mers.

Però Blast crea i w-mers basati su una matrice quindi può accadere che match esatti diano meno score S di match non esatti:

es. secondo la Blosum62

- il match perfetto AIS-AIS dà score 12
- lo score inesatto LSH-MSH dà score 14

=> il secondo è premiato più del primo

Per ricerche in banche dati nucleotidiche, l'indicizzazione in w-mers ha poca rilevanza.

Inoltre il valore w di default di Blast per i nucleotidi è 11, il che lo porta a non riconoscere sequenze che condividano in modo esatto meno di 11 basi, è questo è un limite grosso.

Fasta è molto più tollerante per sequenze che presentano gaps, visto che già nelle prime fasi prevede il loro inserimento, mentre Blast li inserisce solo in fase di allungamento.

⇒ FASTA è più adatto a ricerche in banche dati nucleotidiche

⇒ BLAST è più adatto a ricerche in banche dati proteiche

Anche se questa regola è un po' troppo arbitraria...

# FASTA

<http://www.ebi.ac.uk/fasta33/>

# BLAST

<http://www.ncbi.nlm.nih.gov/BLAST/>