Docente: **Matteo Re**

UNIVERSITÀ DEGLI STUDI DI MILANO

C.d.l. Informatica

Bioinformatica

A.A. 2013-2014 semestre I

3 Allineamento veloce (euristiche)

Banche dati primarie e secondarie

Esistono **due categorie principali** di banche dati biologiche: i collettori **primari** e le banche darti **secondarie**.

- **Collettori primari**: sono 3, GenBank, EMBL e DDBJ. Ogni sequenza sottomessa a ognuna di queste banche dati viene trasferita alle altre due nell'arco di 24 ore. Grande quantità di dati ma di bassa qualità.
- **Banche dati secondarie:** contengono meno informazioni. Le sequenze sono curate manualmente (qualità superiore). Spesso sono banche dati tematiche.

Banche dati secondarie (molti tipi)

Istituto	Dati contenuti	Database
NCBI	Reference Sequences	RefSeq
SIB (swiss institute of bioinformatics)	Sequenze proteiche annotate manualmente	SwissProt
Sanger institute	Annotazione di proteine sulla base dei domini proteici contenuti	PFam
NCBI	Annotazioni inerenti a geni (trascritti)	Gene
NCBI	Collezioni annotate di dati di espressione	GEO

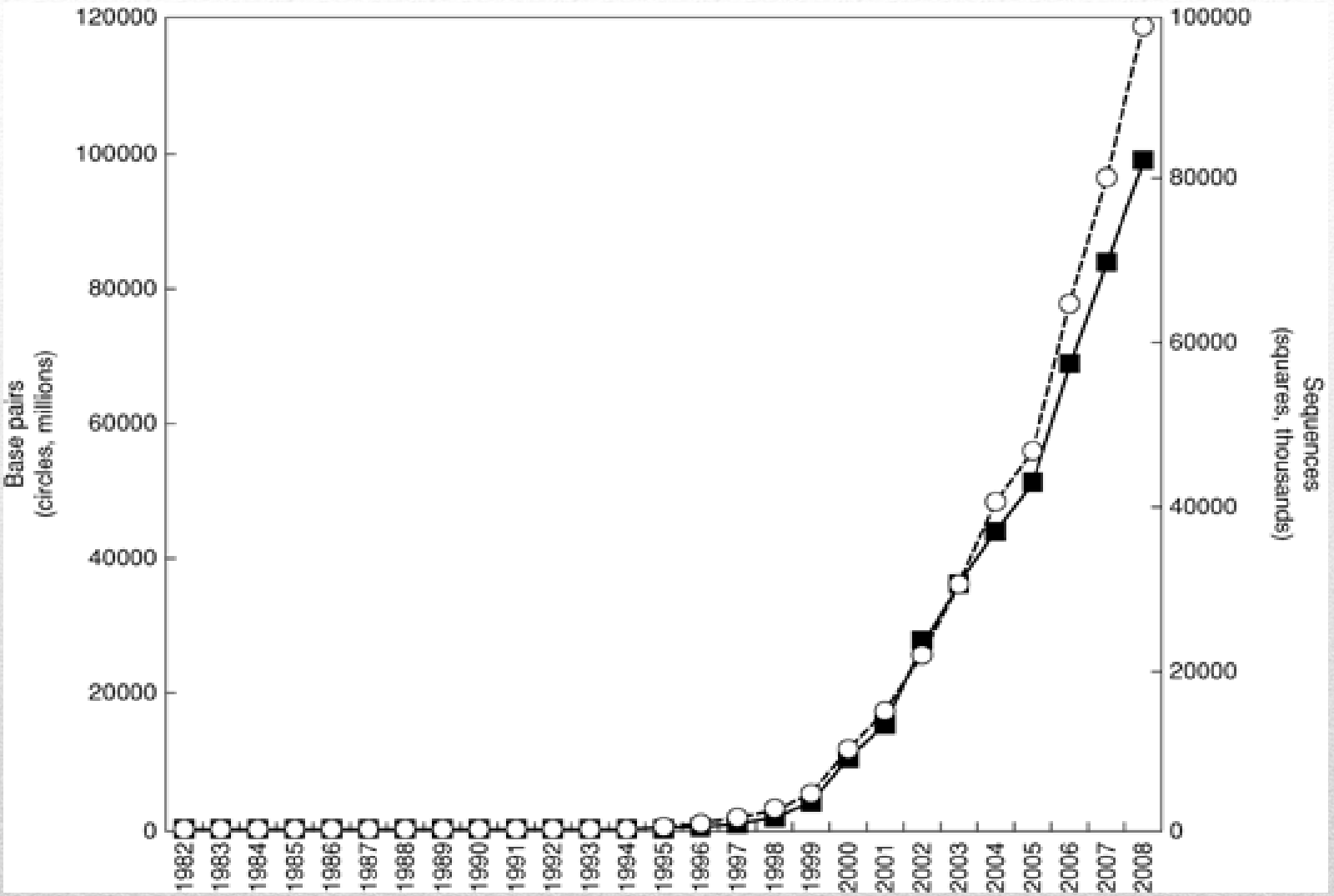
… e molte altre

Crescita del numero di sequenze nelle banche dati biologiche

Grazie ai progressi nelle tecnologie di sequenziamento il numero di sequenze contenute nelle banche dati biologiche è cresciuto ad un ritmo sempre maggiore negli ultimi anni.

Questo crea dei **problemi pratici**:

- Gli algoritmi di allinamento NW e SW garantiscono di trovare allineamenti **ottimali** (confronto pairwise).
- Purtroppo sono **troppo lenti** per permettere l'allineamento di una singola sequenza con i **milioni** di sequenze disponibili nelle banche dati pubbliche.



Genbank : num. seq. e paia di basi (1992-2008)

Possibile soluzione?
Scambiare accuratezza con velocità!

Quando eseguiamo una ricerca in banca dati mediante un algoritmo di allineamento abbiamo:

- una sequenza che vogliamo confrontare con il contenuto della banca dati (**sequenza sonda o query)**
- Una **collezione** di sequenze in banca dati

Vogliamo trovare tutti i **mach** biologicamente significativi tra la query e le sequenze in banca dati. Il confronto è effettuato a coppie. La query è costante, la sequenza del database considerata in un dato momento è detta **subject** (o target sequence).

Cercare **TUTTI** i match e restituire solo i «migliori» comporterebbe l'applicazione di NW o SW a **tutte** le possibili coppie (query, subject).

Bio CS

Scambiare accuratezza con velocità

E’ necessario fare alcune assunzioni che permettano di semplificare il problema (e questo ha dei costi). Dal punto di vista biologico possiamo assumere che:

- Nel corso di una ricerca per similarità contro il contenuto di una banca dati otterremo **molti match**. Ma non tutti sono «buoni» (alcuni sono dovuti al caso).
- Se un allineamento rappresenta un vero match (esistenza di omologia) allora **è attesa la presenza di corte regioni dell’allineamento ad alto punteggio di similarità**.

Bio CS

FORMALIZZAZIONE DEL PROBLEMA

- Data una sequenza **Q** (query) ed una collezione **S** (subjects) composta da *n* sequenze (n molto grande) calcolare gli allineamenti in cui **esiste almeno una regione di alta similarità** tra Q e S_i (i-esima sequenza in banca dati)

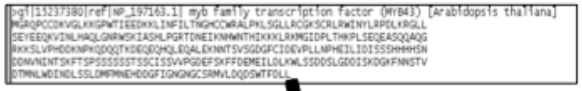
Ci sono 2 modi di risolvere il problema:

1) Calcolo tutti gli allineamenti e verifico se, all'interno di un dato allineamento esiste una regione con score di similarità > di una soglia T. In caso affermativo restituisco l'allineamento.

2) Spezzo Q e tutte le sequenze S in **parole** di lunghezza **w**. Verifico che Q ed Si **condividano almeno una parola** (regione di elevata similarità) e **solo in questo caso** allineo e restituisco il risultato.

Approccio euristico: **FASTA** e **BLAST**

SCHEMA di ricerca in banca dati di biosequenze (mediante BLAST)



Submit Query

BLAST Web Page

Request Results

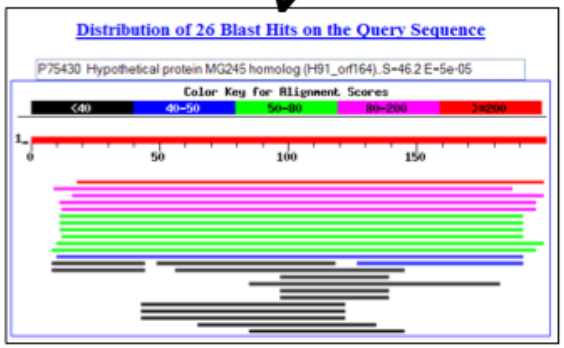
Return Formatted Results

BLAST server

Display Results

fetch ASN.1

fetch sequence



ASN.1

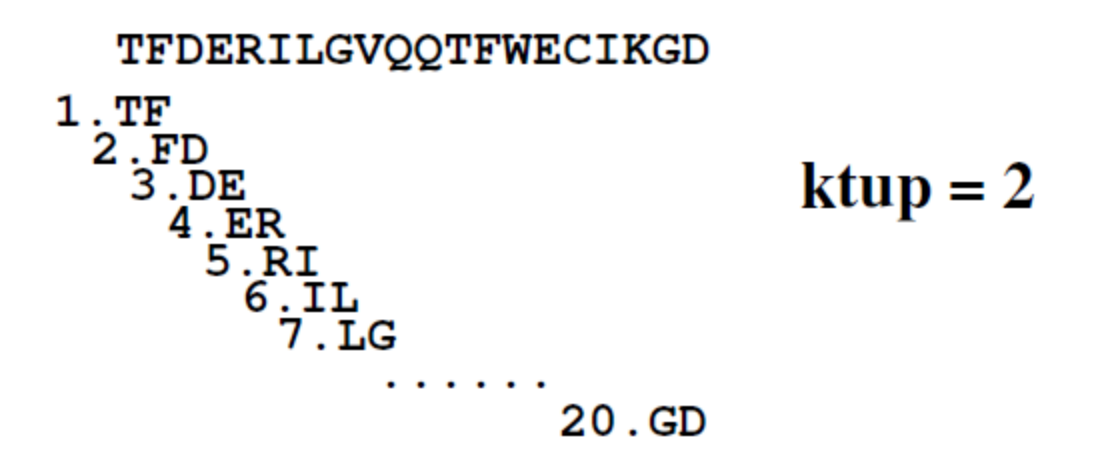
BLAST db

CS

FASTA (FAST-All, Lipman, Pearson 1985):

Prevede tre fasi:

1. indicizzazione: la query viene divisa in parole di lunghezza ktup e si memorizzano tutte le posizioni di inizio parola. Ad esempio:



Di solito per le proteine **ktup**=2; per il DNA **ktup** = 4,6.

Inizia poi l'analisi di tutte le sequenze subject. Viene costruita una lookup-table per ognuna.

CS

FASTA (FAST-All, Lipman, Pearson 1985):

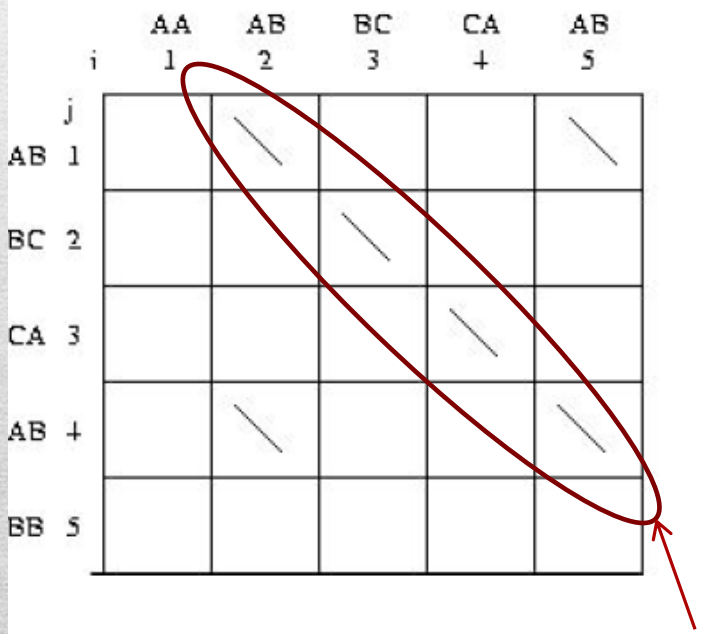
2. Ricerca (I) : Ogni **SUBJECT** della banca dati viene consultata allo stesso modo (anche queste sono indicizzate) e si confronta l'indice della ktup della query con l'indice della ktup della subject (nel caso in cui esista una corrispondenza). Trovata una corrispondenza ci si sposta di una posizione (1 simbolo) sulla **QUERY** e si verifica se esiste un match nella **SUBJECT**. Se c'è verifichiamo che la **differenza tra l'indice della ktup QUERY e l'indice della ktup SUBJECT sia uguale a quella osservata tra le ktup del passo precedente**. Se è così ci spostiamo avanti di un altro passo (sulla QUERY).

Cosa troviamo in questo modo?

CS

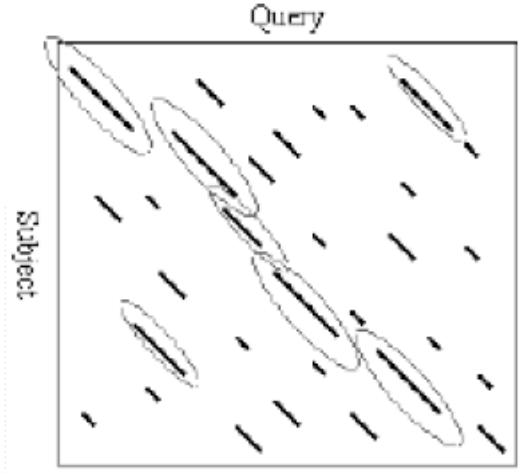
FASTA (FAST-All, Lipman, Pearson 1985):

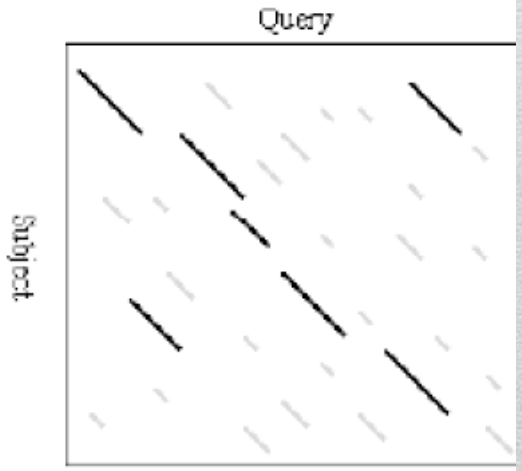
2. Ricerca (II) : Se la differenza tra gli indici delle ktup rimane **COSTANTE** abbiamo trovato una regione di **identità** estesa, che corrisponde ad una **diagonale** nella matrice di allineamento!



query: AABCAB

subject: ABCABB





Differenza i-j **costante** : c'è continuità!
(abbiamo trovato una diagonale)

CS

FASTA (FAST-All, Lipman, Pearson 1985):

2. Ricerca (III) : Ad ogni diagonale trovata si può assegnare uno score di similarità mediante matrici di sostituzione (qui ciò che varia è il premio per la conservazione dei vari aa presenti nelle diagonali.

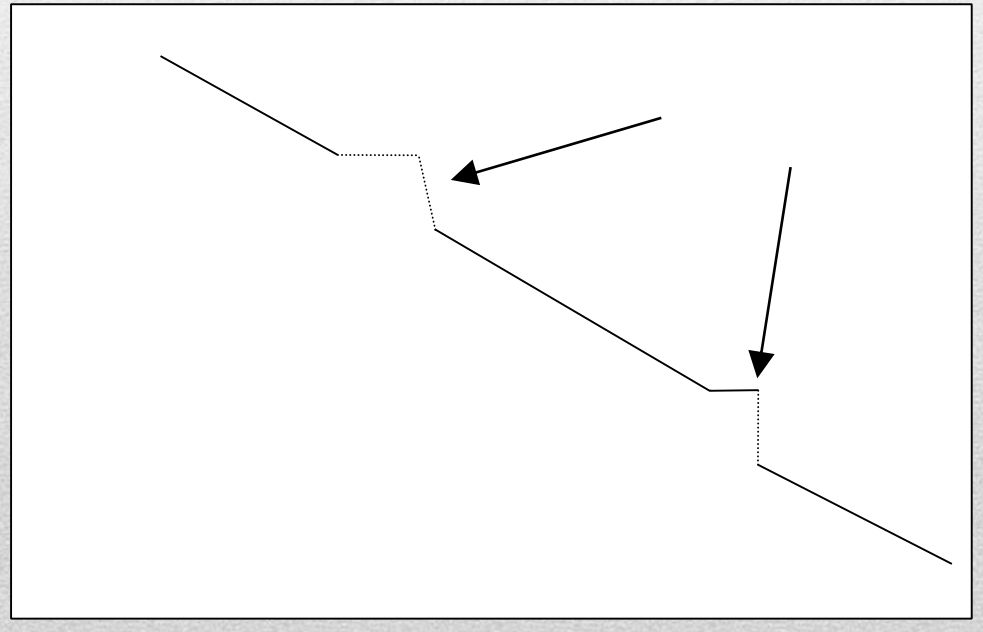
Il programma ripete queste operazioni per ogni subject della banca dati, identificando le **Best Initial Regions** i cui punteggi sono chiamati **Init1**. Con questi fa una graduatoria per decidere su quante e quali sequenze procedere. La scelta delle subject più adatte è stata fatta. Da ora procede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

CS

FASTA (FAST-All, Lipman, Pearson 1985):

3. Raffinamento (I):

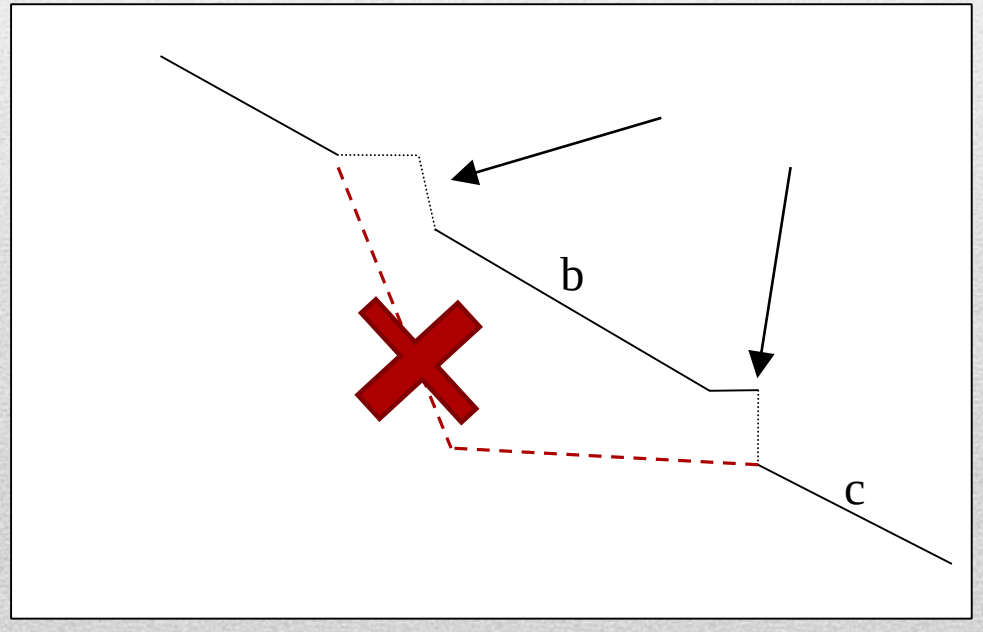
Ora il programma cerca di congiungere ogni best initial region **della subject** confermata utilizzando i parametri di penalty per i gaps. Le regioni vengono allungate e avranno un nuovo score, detto **InitN**. Ma come fa a scegliere **quali diagonali** congiungere (mediante gaps) tra quelle disponibili?



CS

FASTA (FAST-All, Lipman, Pearson 1985):

3. Raffinamento (II) : Il programma combina mediante gaps le diagonali **NON SOVRAPPOSTE** ed il cui ordine è compatibile con il processo di allineamento. Non proverà a congiungere la diagonale c con la diagonale a poiché esiste la diagonale b.

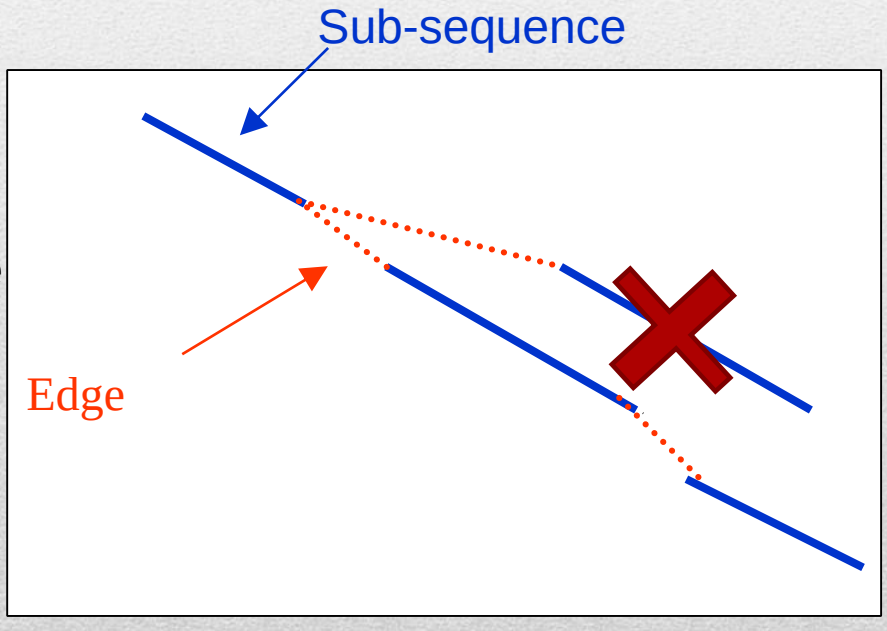


CS

FASTA (FAST-All, Lipman, Pearson 1985):

3. Raffinamento (III) : Il programma combina mediante gaps le diagonali **NON SOVRAPPOSTE** ed il cui ordine è compatibile con il processo di allineamento. Non proverà a congiungere la diagonale c con la diagonale a poiché esiste la diagonale b.

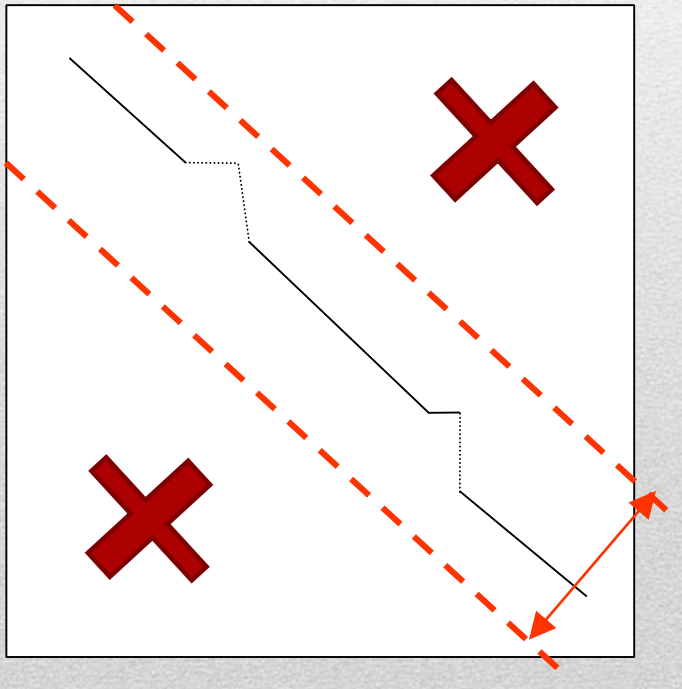
FASTA cerca il «miglior» **percorso** (che ora equivale all'**unione** di diagonali con score maggiore. E poi elimina le altre diagonali



CS

FASTA (FAST-All, Lipman, Pearson 1985):

3. Raffinamento (VI) : Identificata la miglior combinazione di diagonali (unite mediante gaps) FASTA esegue un allineamento mediante **programmazione dinamica** (alla SW) tra QUERY e SUBJECT e restituisce il risultato.



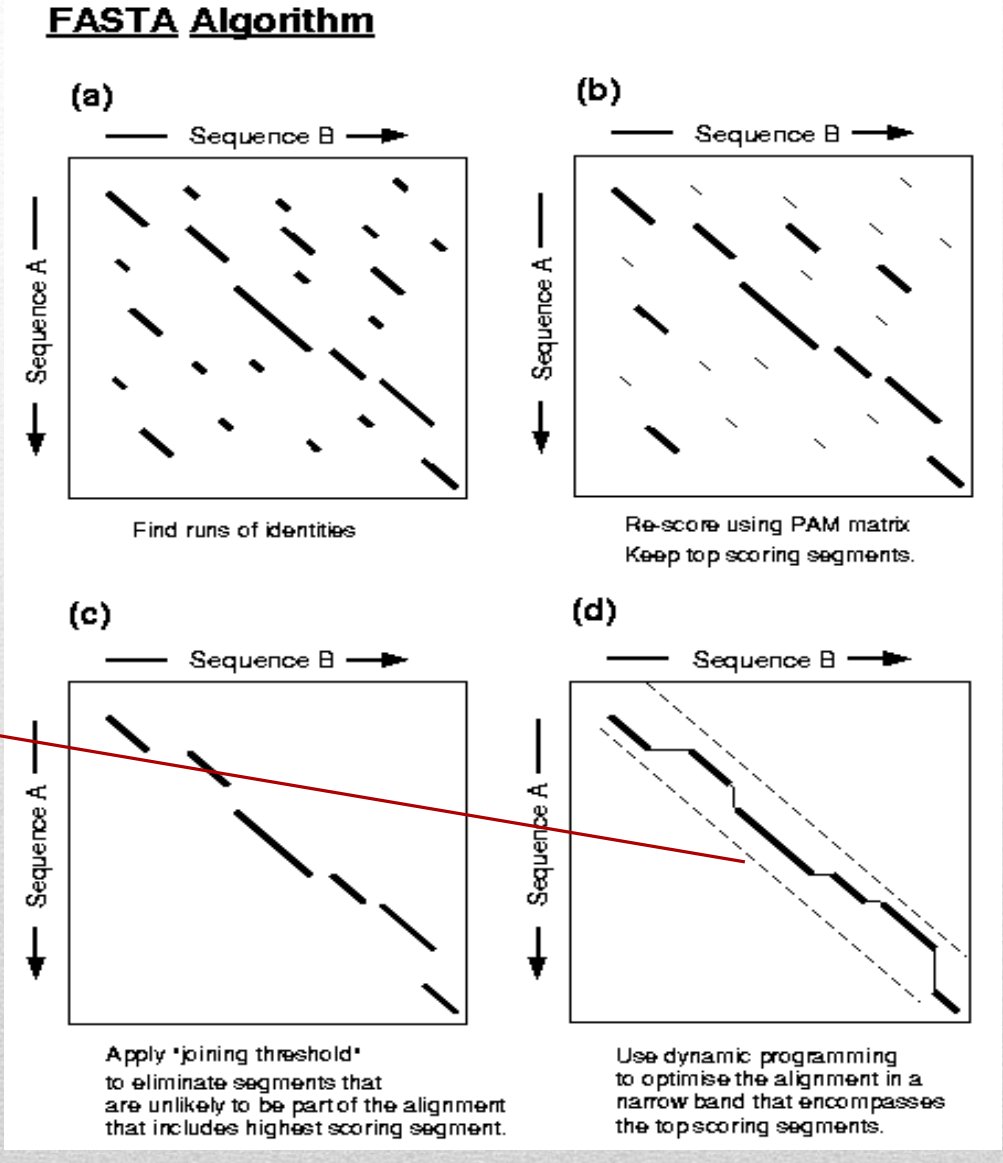
La larghezza di questa banda è un parametro

NB: L'algoritmo di allineamento può essere eseguito più **velocemente** poiche possiamo restringere l'attenzione alla parte della matrice di programmazione dinamica che **SAPPIAMO** contenere il miglior allineamento (migliore Unione di diagonali).

Riepilogo algoritmo FASTA

(FAST-All, Lipman, Pearson 1985)

Lo score finale ottenuto dopo l'allineamento (d) è definito **Opt** score.



CS

BLAST (basic **local** aligment search tool) Altshul, 1990

E’ ottimizzato per trovare allineamenti **locali** privi di gap. L’algoritmo prevede tre fasi:

1. leggendo la sequenza QUERY viene formato un elenco di parole di lunghezza W. Per ognuna viene creata una lista di parole affini (W-meri): vengono considerati tutti i W-meri che superano una soglia di similarità fissata **T** quando viene allineato con la parola della query;

2. vengono esaminate tutte le sequenze SUBJECT, per cercare la presenza di tutti i W-meri dell’elenco. Ogni corrispondenza trovata (hit) viene considerata come parte di un allineamento più esteso;

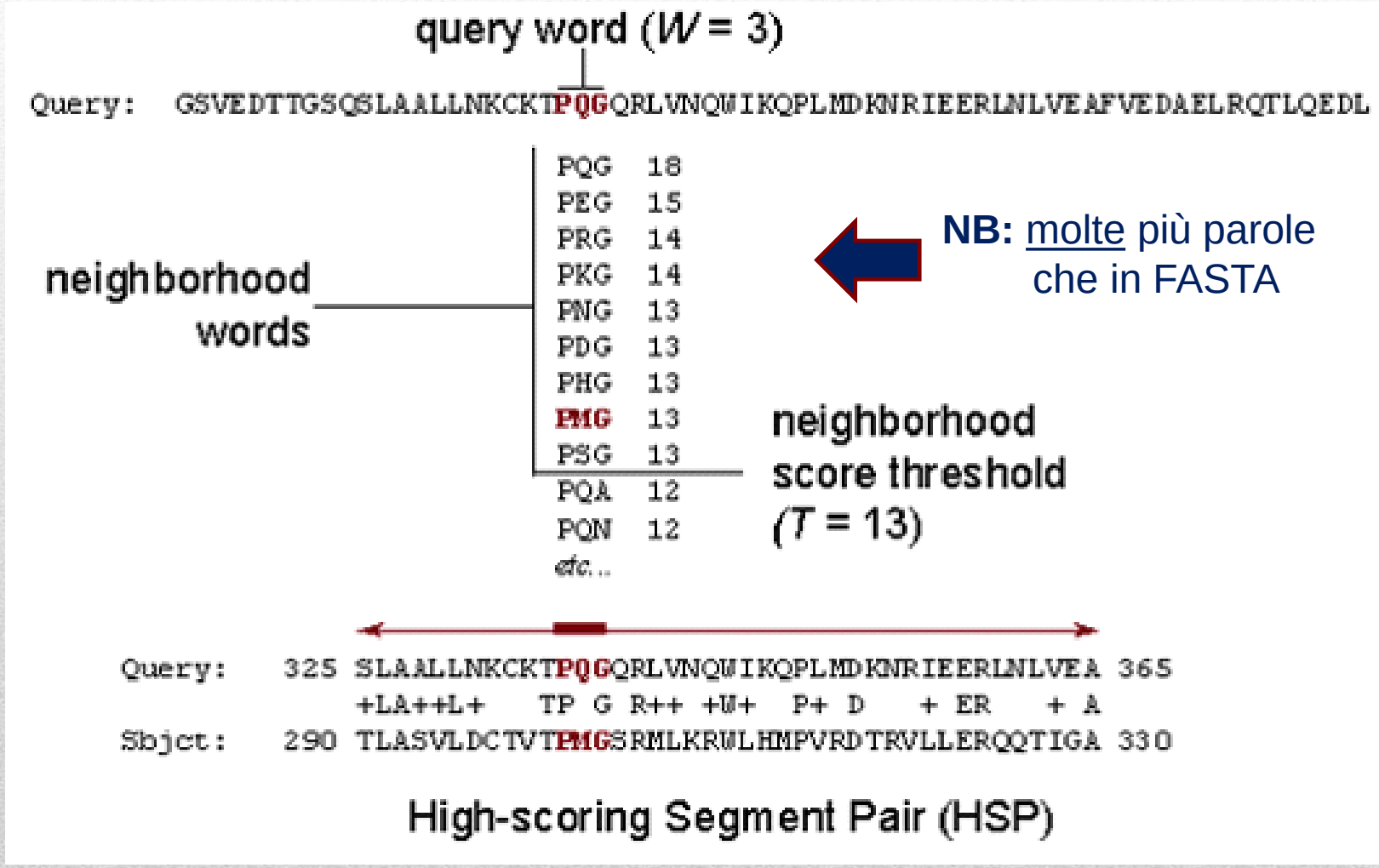
3. viene considerata la possibilità di **estendere** ogni hit in **entrambe le direzioni**, senza aggiunte di gap. Si ottiene un **segmento di allineamento locale detto HSP** (high-scoring segment pair) e il suo relativo score.

http://blast.ncbi.nlm.nih.gov/Blast.cgi

CS

BLAST (basic **local** aligment search tool) Altshul, 1990

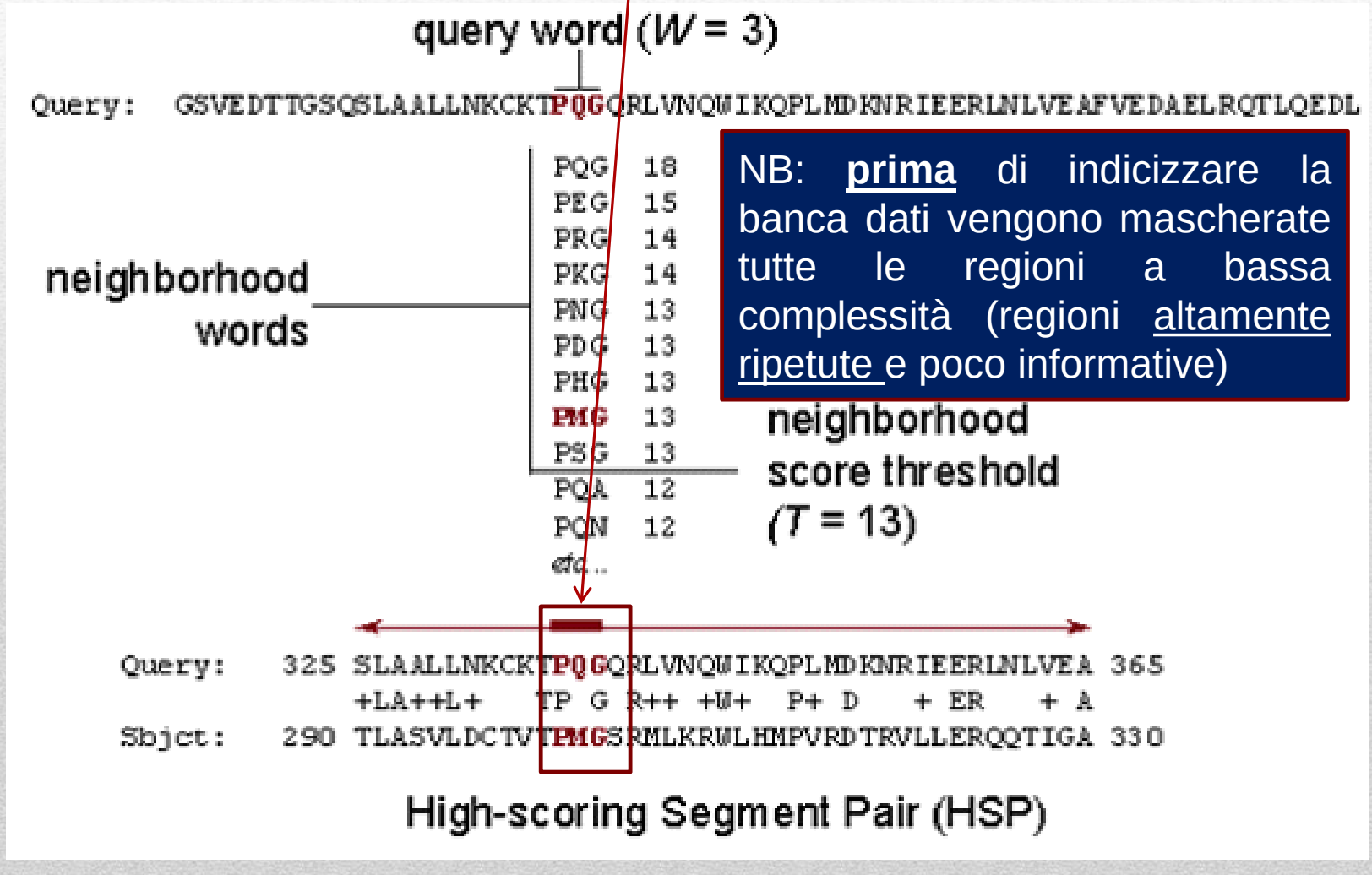
1. Inizializzazione: Creazione di un set di parole dalla query sequence:



CS

BLAST (basic **local** aligment search tool) Altshul, 1990

2. Ricerca (I): Cercare parole «**simili**» nelle sequenze presenti in banca dati



NB: **prima** di indicizzare la banca dati vengono mascherate tutte le regioni a bassa complessità (regioni altamente ripetute e poco informative)

CS

BLAST (basic **local** aligment search tool) Altshul, 1990

2. Ricerca (I): Cercare parole «**simili**» nelle sequenze presenti in banca dati:

- Ogni sequenza SUBJECT presente nella banca dati viene esaminata alla ricerca di corrispondenze (**hit**) tra la **collezione di W-meri** corrispondente alla parola corrente della sequenza QUERY e le parole della sequenza SUBJECT.
- In caso sia stata trovata una corrispondenza (hit) vengono memorizzate le posizioni delle parole sia nella QUERY che nella sequenza SUBJECT. (es. RQC si trova in posizione 125 nella QUERY ed in posizione 100 nella sequenza SUBJECT).
- Vengono considerate hit accettabili solo quelle in cui le parole si trovano ad una distanza massima **A**. Questo equivale ad accettare solo hit che si trovano, presumibilmente, su una buona diagonale (**cfr FASTA**).

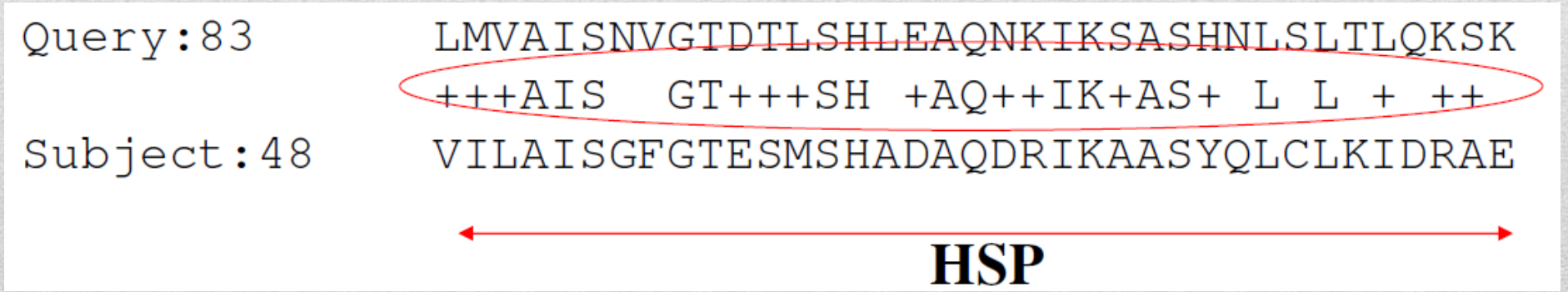
CS

BLAST (basic **local** aligment search tool) Altshul, 1990

3. Estensione :

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** solo per alcuni residui, e poi risale, l'HSP può ancora estendersi. Il parametro **X** dice la **quantità di perdita di score massima tollerabile** se si prosegue con l'allungamento dell'HSP.



CS

BLAST (basic **local** aligment search tool) Altshul, 1990

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**.

CS

BLAST (basic **local** aligment search tool) Altshul, 1990

QUANTI TIPI DI BLAST ?

blastN	$q_{DNA} \leftrightarrow s_{DNA}$
blastP	$q_{prot} \leftrightarrow s_{prot}$
TblastN	$q_{prot} \leftrightarrow s_{t_1(DNA)}\ s_{t_1^c(DNA)}\ s_{t_2(DNA)}\ s_{t_2^c(DNA)}\ s_{t_3(DNA)}\ s_{t_3^c(DNA)}$
blastX	$q_{t_1(DNA)}\ q_{t_1^c(DNA)}\ q_{t_2(DNA)}\ q_{t_2^c(DNA)}\ q_{t_3(DNA)}\ q_{t_3^c(DNA)} \leftrightarrow s_{prot}$
TblastX	$q_{t_1(DNA)}\ q_{t_1^c(DNA)}\ q_{t_2(DNA)}\ q_{t_2^c(DNA)}\ q_{t_3(DNA)}\ q_{t_3^c(DNA)} \leftrightarrow s_{t_1(DNA)}\ s_{t_1^c(DNA)}\ s_{t_2(DNA)}\ s_{t_2^c(DNA)}\ s_{t_3(DNA)}\ s_{t_3^c(DNA)}$

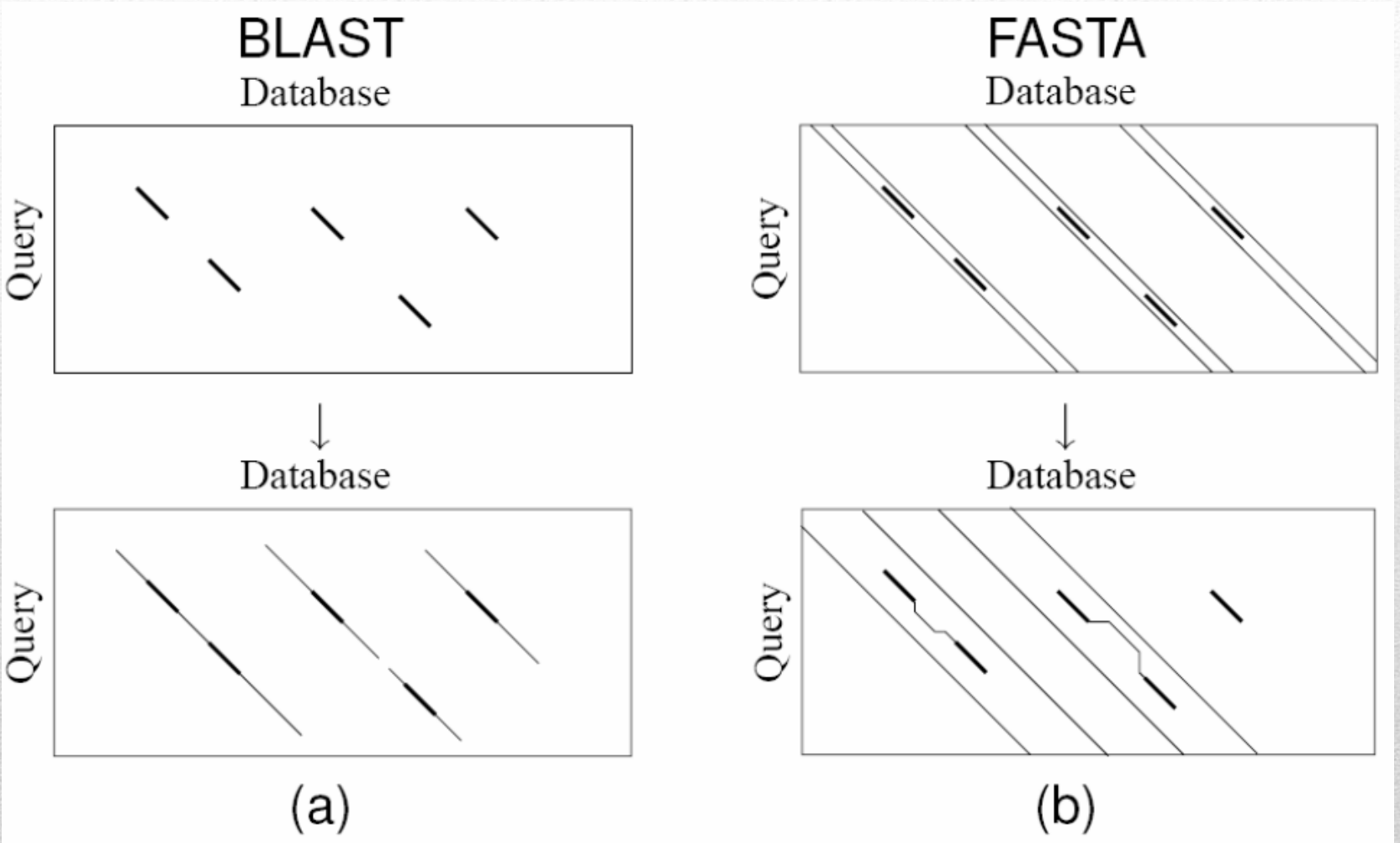
Sensibilità rispetto ad omologie lontane crescente

Bio

FASTA vs **BLAST**

differenze tra gli allineamenti prodotti

CS



Bio

Riepilogo euristiche di allineamento *(confronto tra algoritmi)*

CS

Programmazione dinamica (DP) :

1) Massima sensibilità:

- DP utilizza, di fatto, **TUTTA** l'informazione contenuta nelle sequenze confrontate.

2) Tempo di esecuzione elevato:

- DP calcola **tutti** i valori nella matrice di programmazione dinamica (anche quelli situati nelle **aree inutili)** in modo da garantire la produzione di un allineamento **ottimale**.

Bio

Riepilogo euristiche di allineamento *(confronto tra algoritmi)*

CS

FASTA:

1) <u>Meno sensibile</u> di DP e BLAST:

- FASTA utilizza **PARTE** dell'informazione contenuta nelle sequenze confrontate in modo da accelerare il calcolo.
- FASTA cerca match **ESATTI** tra le k-tuple nella fase iniziale di scansione della banca dati
- FASTA **non valuta** statisticamente gli allineamenti prodotti

2) Tempo di esecuzione MINORE rispetto a DP:

- (stessi motivi riportati al punto 1)

Bio

Riepilogo euristiche di allineamento *(confronto tra algoritmi)*

CS

BLAST:

1) <u>Più sensibile</u> di FASTA:

- BLAST (come vedremo) valuta statisticamente gli allineamenti prodotti.
- BLAST utilizza, nella fase iniziale, match tra parole «simili» (collezioni di W-meri).

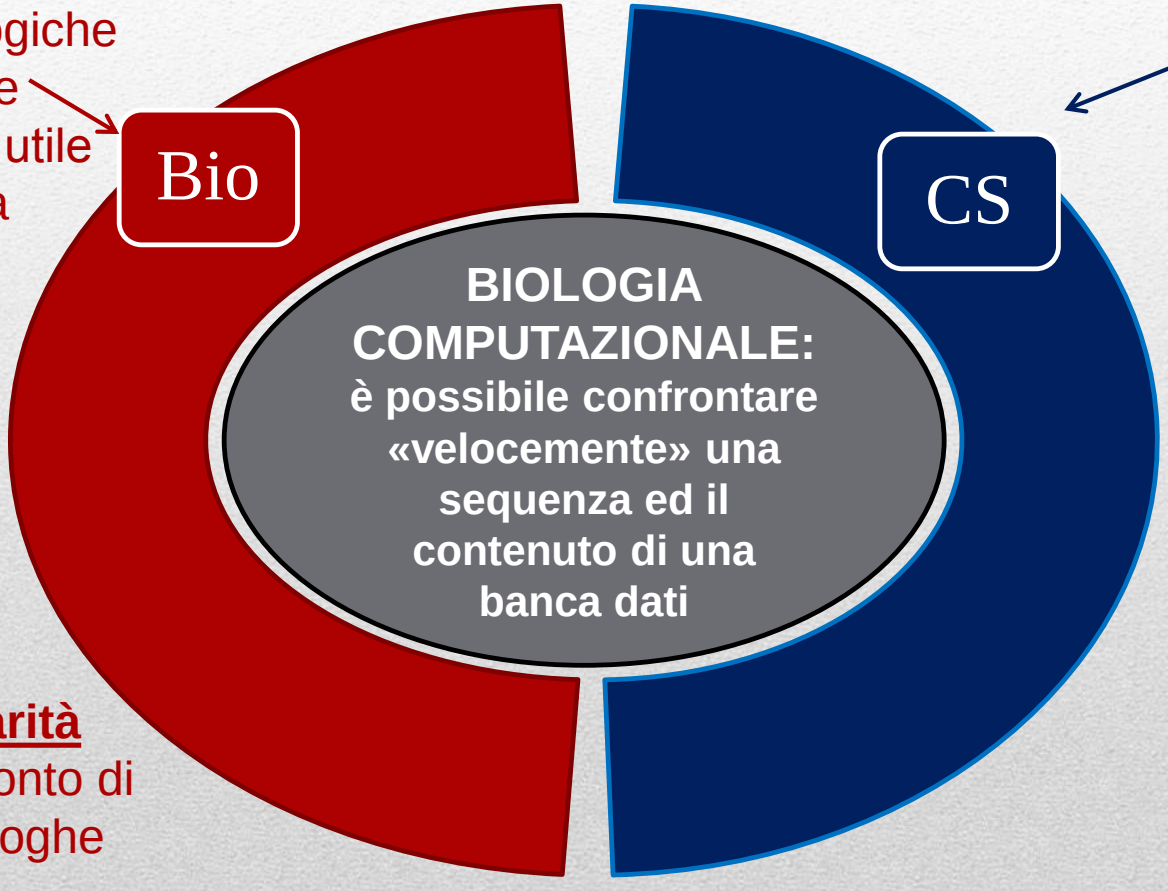
2) Tempo di esecuzione MINORE rispetto a FASTA:

- BLAST elimina il rumore (filtro regioni a bassa complessità) prima di iniziare la scansione del database.

EURISTICHE ALLINEAMENTO

Sequenze biologiche **variano** durante l'evoluzione. E' utile confrontare una sequenza con **collezioni** di sequenze.

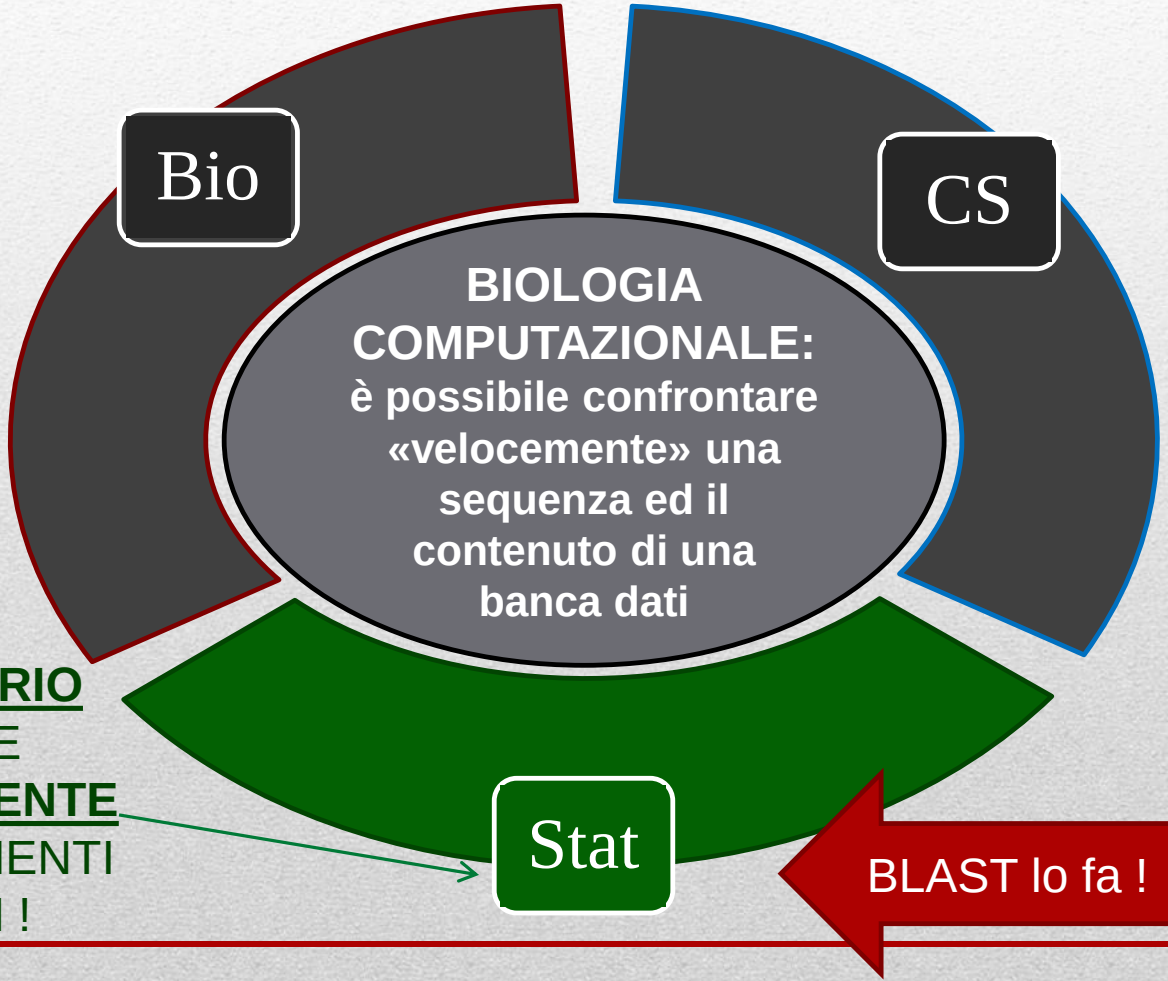
NB:
E' **attesa** la presenza di regioni ad **alto score di similarità** durante il confronto di sequenze omologhe



E' **possibile** effettuare una ricerca per similarità in banca dati in maniera veloce **ponendo un filtro a monte dell' allineamento (riduzione numero di confronti)**

PROBLEMA: se confrontiamo una sequenza contro una banca dati ... lo **score** ottenuto è **SEMPRE statisticamente significativo**?

EURISTICHE ALLINEAMENTO (e ricerca in banche dati)



E' **NECESSARIO** VALUTARE **STATISTICAMENTE** GLI ALLINEAMENTI OTTENUTI !