Informatica per le Biotecnologie
Algoritmica Lezione 7

1. Il problema dei pesi e dei gap

I PESI

I valori +1 e -1 impiegati finora possono non essere adeguati per il problema biologicoconsiderato

in particolare il peso di carattere-spazio rispetto al mismatch, e il peso di differenti situazioni di mismatch.

Per il peso di match e mismatch
si usa comunemente il criterio:

	A	T	C	G
A	+2	−2	−2	−1
T	−2	+2	−1	−2
C	−2	−1	+2	−2
G	−1	−2	−2	+2

infatti una mutazione tra purine (A e G)
o tra pirimidine (T e C) è in genere più
probabile che una mutazione tra una
purina e una pirimidina.

Il peso relativo di carattere-spazio rispetto al mismatch
deve essere indicato dal biologo.

Ricordiamo che cambiare i pesi richiede solo un'immediata variazione della formula per il calcolo di M[i,j] in tutti gli algoritmi.

I GAP (successioni di spazi)

```
Y:  A  T  A  A  T  C  G  A  C
X:  G  T  -  -  -  -  G  T  C
   -1 +2 -1 -1 -1 -1 +2 -2 +2 = -1
```

Esistono varie funzioni matematiche per trattare i gap. Detti ***k*** il numero di spazi nel gap, ***a*** il peso del primo spazio e ***b*** il peso di ogni spazio successivo, la funzione più comune detta ***costo affine*** è espressa come

$$c(k) = - a - (k-1)b, \quad \text{con } b < a$$

```
Y:  A  T  A  A  T  C  G  A  C
X:  G  T  -  -  -  -  G  T  C
   -1 +2 -1 -1 -1 -1 +2 -2 +2 = -1
```

Ponendo a = 1, b = 0.2

```
Y:  A  T  A   A    T    C   G  A  C
X:  G  T  -   -    -    -   G  T  C
   -1 +2 -1 -0.2 -0.2 -0.2 +2 -2 +2 = +1.4
```

Attenzione: Per calcolare il costo di uno spostamento orizzontale o verticale sulla matrice si deve sapere se nella cella di provenienza M[i,j-1] o M[i-1,j] è contenuto un valore determinato da un gap già iniziato: in questo caso si sottrae *b*, altrimenti si sottrae *a*

Si può impiegare una seconda matrice G

G[i,j] = 0, se M[i,j] è stato ottenuto da M[i-1,j-1];

G[i,j] = 1, se M[i,j] è stato ottenuto da M[i,j-1] ed eventualmente anche da M[i-1,j-1];

G[i,j] = 2, se M[i,j] è stato ottenuto da M[i-1,j] ed eventualmente anche da M[i-1,j-1];

G[i,j] = 3, se M[i,j] è stato ottenuto sia da M[i,j-1] che da M[i-1,j], ed eventualmente anche da M[i-1,j-1].

G è costruita assieme a M, calcolando a ogni passo il valore di entrambe in funzione dei valori nelle celle precedenti

Vediamo per esempio come varia l'algoritmo di *pattern comparison*

M	0	C	A	A	G	G	G	T	T	T	C	A
0	0	**0**	0	0	0	0	0	0	0	0	0	0
A	-1	-1	**+2**	+2	+1	+0.8	+0.6	+0.4	+0.2	0	-0.8	+2
A	-1.2	-1.2	+1	**+4**	+3	+2.8	+2.6	+2.4	+2.2	+2	+1.8	+1.6
T	-1.4	-1.4	+0.8	**+3**	+2	+1.8	+1.6	+4.6	+4.4	+4.2	+4	+3.8
G	-1.6	-1.6	+0.6	+2.8	**+5**	**+4**	**+3.8**	**+3.6**	**+3.4**	+3.2	+3	+2.8
T	-1.8	-1.8	+0.4	+2.6	+4	+3	+2.8	+5.8	+5.6	**+5.4**	+4.4	+4.2
C	-2	+0.2	+0.2	+2.4	+3.8	+2.8	+2.6	+4.8	+4.6	+4.4	**+7.4**	+6.4

Y: C A A - G G G T T T C A

X: A A T G - - - - T C

2. Individuazione di un gene all'interno di una sequenza di DNA

Individuare se, in quale posizione, e con quali differenze, è presente un gene di cui è nota la sequenza S all'interno del DNA di un dato organismo di cui è nota la sequenza T.

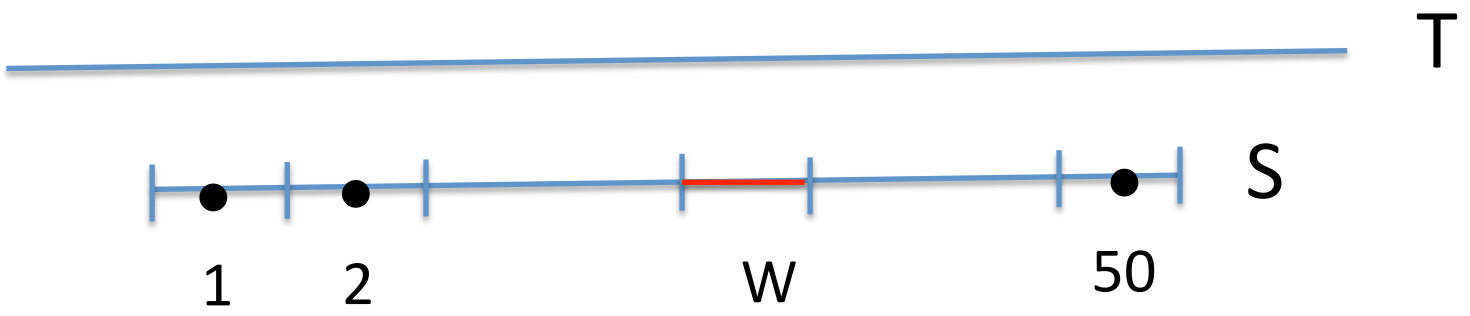
È il problema di pattern comparison già studiato, discusso in un contesto biologico con dati significativi in questo campo. In particolare la differenza di lunghezza tra S e T può essere grandissima.

Per esempio se la S ha una lunghezza |S| di migliaia di basi e la T ha lunghezza |T| di centinaia di milioni di basi, l'algoritmo quadratico di programmazione dinamica richiede un numero di operazioni di ordine |S|•|T| che è elevatissimo anche se polinomiale nella dimensione dell'input.

Utilizzeremo un nuovo metodo che illustriamo con un esempio.

Poniamo che il risultato che S è contenuta in T sia accettabile con al più $e = 49$ errori in totale:

dividendo S in $e + 1 = 50$ tratti consecutivi di $|S|/50$ basi ciascuno, almeno un tratto W deve essere privo di errori



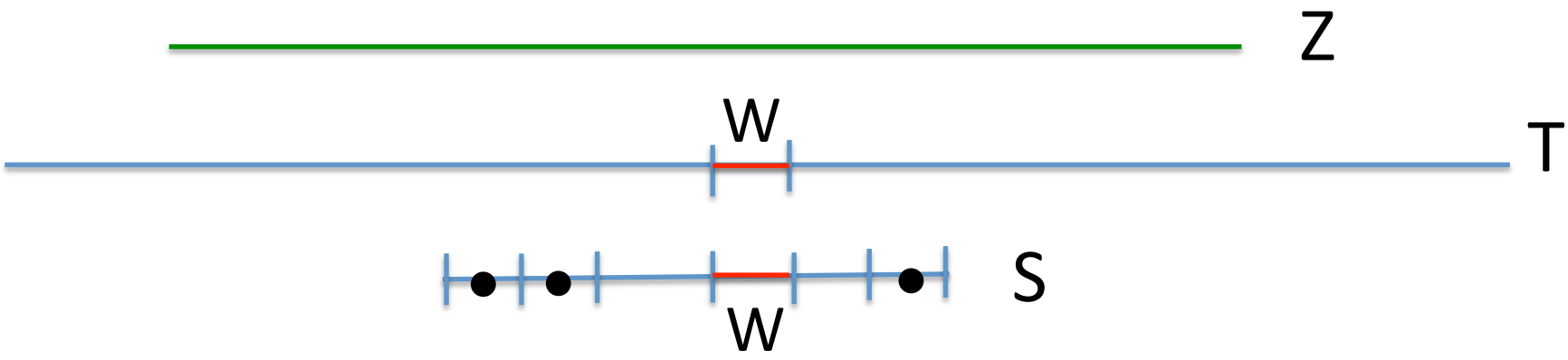
Esistono algoritmi più efficienti della programmazione dinamica per determinare la presenza **esatta** (cioè senza errori) di un pattern S in un testo T. Uno di questi detto **KMP** funziona in tempo di ordine $|T|+|S|$, lineare nella dimensione dell'input

nel nostro esempio questo ordine è $|T|$, perché $|T|$ è molto maggiore di $|S|$.

Iniziamo la ricerca di S in T applicando 50 volte l'algoritmo KMP per la ricerca di ognuno dei 50 tratti di S in T.
Due risultati sono possibili:

1) nessun tratto appare esattamente in T, dunque S non appare in T;

2) un tratto W appare esattamente in T: se S è contenuto (con errori) in T, S deve trovarsi in una zona Z di T attorno a W con lunghezza $|Z| = 2|S|$. Si confrontano quindi S e Z con l'algoritmo di programmazione dinamica.



si confrontano S e Z con l'algoritmo di programmazione dinamica

1) nessun tratto di S appare esattamente in T:

tempo richiesto di ordine $e|T|$ per la ricerca degli $e+1$ tratti in T.

2) Un tratto W appare esattamente in T:

si confrontano S e la zona Z di T :

tempo di ordine $e|T|$, più $|S|^2$ per il confronto tra S e Z; totale di ordine $e|T|$ se $e|T|>|S|^2$ come nel nostro esempio.

L’applicazione diretta dell’algoritmo di programmazione dinamica a S e T avrebbe richiesto un tempo $|S| \cdot |T| >> e|T|$ (circa 100 volte superiore nel nostro esempio).

3. Determinazione della sequenza di DNA di un organismo.

Per sequenziare **l'intero genoma G** di un organismo di cui già si conosce il **genoma di riferimento R**, si parte dai frammenti di **G** e si impiega un metodo simile a quello appena descritto.

Nel caso del genoma umano la differenza tra due genomi **G** e **R** è dell'ordine di $\mathbf{10^6}$ basi su un totale di $\mathbf{3.2 \cdot 10^9}$, cioè circa 0.3 millesimi.

Anziché assemblare i frammenti di **G**, in linea di principio si cercano questi frammenti in **R** per stabilire in che zona di G si dovranno collocare e quindi con quali altri frammenti collegarli.

Il procedimento è molto più complicato, ma sequenziare un genoma con questo metodo richiede un numero di operazioni molto inferiore al completo fragment assemby.

Due considerazioni che si estendono a problemi e algoritmi in genere

1. L'apparizione esatta di un pattern in un testo si determina con l'algoritmo KMP, ma anche con la programmazione dinamica in cui le apparizioni esatte del pattern corrispondono alle celle nell'ultima riga della matrice relative a errore zero:

l'algoritmo KMP è molto più complicato ma molto più efficiente del secondo. Il problema è facile perché risolubile in tempo lineare, e KMP è un algoritmo ottimo.

2. Molti problemi su sequenze biologiche, che non trovano spazio in queste lezioni, appartengono alla classe dei problemi **NP-completi**: per risolverli si ricorre ad algoritmi euristici e/o approssimati.