Allineamento di Sequenze

(e strutture dati)

Alberto Policriti

21 Novembre, 2019

Il Piano

Questioni di base

1. cosa significa *allineare* (due o più sequenze, e.g. di DNA)?
2. perchè mi dovrebbero interessare degli *algoritmi* per l'allineamento?
3. strutture dati ... perchè?
4. tempo, spazio, correttezza, completezza ... è *tutto* ciò cui sono interessati i bioinformatici?

Allineamento

Il significato (operativo) del termine allineare

Allineare due stringhe significa ...

... convertire la prima nella seconda

Il significato (operativo) del termine allineare

Allineare due stringhe significa ...

... convertire la prima nella seconda

Example

$$\sigma_1 \equiv \quad a \quad c \quad g \quad t \quad c \quad a \quad t \quad c \quad a$$
$$\sigma_2 \equiv \quad t \quad a \quad a \quad g \quad t \quad g \quad t \quad c \quad a$$

Che cosa intendiamo per convertire una sequenza in un'altra?

Definition
Un allineamento di σ_1 e σ_2 è una *stringa* nell'alfabeto

$$\{\mathbf{m}, \mathbf{i}, \mathbf{d}, \mathbf{s}\}$$

Che cosa intendiamo per convertire una sequenza in un'altra?

Definition
Un allineamento di σ_1 e σ_2 è una *stringa* nell'alfabeto

$$\{\mathbf{m}, \mathbf{i}, \mathbf{d}, \mathbf{s}\}$$

Example

$\sigma_1 \equiv$		a	c	g	t	c	a	t	c	a
	i	**m**	**s**	**m**	**m**	**s**	**d**	**m**	**m**	**m**
$\sigma_2 \equiv$	t	a	a	g	t	g		t	c	a

Due problemi

1. trovare un allineamento
2. stabilire se quello trovato è ottimo (in che senso?)

Che cosa intendiamo per convertire una sequenza in un'altra?

Definition
Un allineamento di σ_1 e σ_2 è una *stringa* nell'alfabeto

$$\{\mathbf{m}, \mathbf{i}, \mathbf{d}, \mathbf{s}\}$$

Example

$\sigma_1 \equiv$		a	c	g	t	c	a	t	c	a
	i	**m**	**s**	**m**	**m**	**s**	**d**	**m**	**m**	**m**
$\sigma_2 \equiv$	t	a	a	g	t	g		t	c	a

Per ora ignoriamo il problema di trovare un allineamento o quello di stabilire se è ottimo (in che senso?)

Che cosa intendiamo per convertire una sequenza in un'altra?

Definition
Un allineamento di σ_1 e σ_2 è una *stringa* nell'alfabeto

$$\{\mathbf{m}, \mathbf{i}, \mathbf{d}, \mathbf{s}\}$$

Un allineamento è una stringa

allineamento $\Leftrightarrow$ edit-string

In in effetti è un (semplice, lineare) *"programma"* che converte σ_1 in σ_2 (o viceversa).

Terminologia e significato

Ordiniamo Allineamenti: Costo e Punteggio
Possiamo associare un costo ad una edit-string (vogliamo $\downarrow$)

Possiamo associare un punteggio ad una edit-string (vogliamo $\uparrow$)

Ordiniamo Allineamenti: Costo e Punteggio
Possiamo associare un costo ad una edit-string (vogliamo $\downarrow$)

Possiamo associare un punteggio ad una edit-string (vogliamo $\uparrow$)

Example
Purine (a,g) e Pirimidine (c,t) sono simili, quindi una possibile (non-banale) matrice di punteggi è la seguente:

	a	c	g	t
a	2	-1	1	-1
c	-1	2	-1	1
g	1	-1	2	-1
t	-1	1	-1	2

Ordiniamo Allineamenti: Costo e Punteggio

Possiamo associare un costo ad una edit-string (vogliamo $\downarrow$)

Possiamo associare un punteggio ad una edit-string (vogliamo $\uparrow$)

Example

BLOSUM matrice di costi:

	Ala	Arg	Asn	Asp	Cys	Gln	Glu	Gly	His	Ile	Leu	Lys	Met	Phe	Pro	Ser	Thr	Trp	Tyr	Val
Ala	4																			
Arg	−1	5																		
Asn	−2	0	6																	
Asp	−2	−2	1	6																
Cys	0	−3	−3	−3	9															
Gln	−1	1	0	0	−3	5														
Glu	−1	0	0	2	−4	2	5													
Gly	0	−2	0	−1	−3	−2	−2	6												
His	−2	0	1	−1	−3	0	0	−2	8											
Ile	−1	−3	−3	−3	−1	−3	−3	−4	−3	4										
Leu	−1	−2	−3	−4	−1	−2	−3	−4	−3	2	4									
Lys	−1	2	0	−1	−3	1	1	−2	−1	−3	−2	5								
Met	−1	−1	−2	−3	−1	0	−2	−3	−2	1	2	−1	5							
Phe	−2	−3	−3	−3	−2	−3	−3	−3	−1	0	0	−3	0	6						
Pro	−1	−2	−2	−1	−3	−1	−1	−2	−2	−3	−3	−1	−2	−4	7					
Ser	1	−1	1	0	−1	0	0	0	−1	−2	−2	0	−1	−2	−1	4				
Thr	0	−1	0	−1	−1	−1	−1	−2	−2	−1	−1	−1	−1	−2	−1	1	5			
Trp	−3	−3	−4	−4	−2	−2	−3	−2	−2	−3	−2	−3	−1	1	−4	−3	−2	11		
Tyr	−2	−2	−2	−3	−2	−1	−2	−3	2	−1	−1	−2	−1	3	−3	−2	−2	2	7	
Val	0	−3	−3	−3	−1	−2	−2	−3	−3	3	1	−2	1	−1	−2	−2	0	−3	−1	4

Ordiniamo Allineamenti: Costo e Punteggio
Possiamo associare un costo ad una edit-string (vogliamo $\downarrow$)

Possiamo associare un punteggio ad una edit-string (vogliamo $\uparrow$)

Example
Penalizzazione dei gap (Affine). In effetti ci piacerebbe anche:

penalizzare l'apertura di un gap

e

penalizzare una sua estensione

... dobbiamo integrare la funzione di costo nel meccanismo ricorsivo.

Globale/Locale

La edit-string converte l'*intera* σ_2 nell'*intera* σ_1.

La edit-string converte l'*intera* σ_2 in *una porzione* di σ_1.

Distanza

Pensiamo al costo come una *distanza* tra stringhe.

Example

Distanza di Levensthein: costo 0 per **m**, 1 per gli altri.

$$\begin{array}{lcccccccccc} \sigma_1 \equiv & & a & c & g & t & c & a & t & c & a \\ & \mathbf{i} & \mathbf{m} & \mathbf{s} & \mathbf{m} & \mathbf{m} & \mathbf{s} & \mathbf{d} & \mathbf{m} & \mathbf{m} & \mathbf{m} \\ \sigma_2 \equiv & t & a & a & g & t & g & & t & c & a \end{array} \quad \Rightarrow \quad d_L(\sigma_1, \sigma_2) = 4.$$

Distanza di Hamming: **i** e **d** non sono ammesse.

$$\begin{array}{lcccccccccc} \sigma_1 \equiv & a & c & g & t & c & a & t & c & a \\ & \mathbf{s} & \mathbf{s} & \mathbf{s} & \mathbf{s} & \mathbf{s} & \mathbf{s} & \mathbf{m} & \mathbf{m} & \mathbf{m} \\ \sigma_2 \equiv & t & a & a & g & t & g & t & c & a \end{array} \quad \Rightarrow \quad d_H(\sigma_1, \sigma_2) = 6$$

La distanza di Hamming può essere usata esclusivamente su stringhe di uguale lunghezza.

Regole di massima

Che distanza devo usare?

Stringhe lunghe $\Leftrightarrow$ Levensthein

Stringhe corte $\Leftrightarrow$ Hamming

Algoritmi di Allineamento

Programmazione dinamica per l'allineamento globale

Idea!
Se dobbiamo trovare una stringa (i.e. η) determiniamone un carattere alla volta.

Programmazione dinamica per l'allineamento globale

Idea!
Se dobbiamo trovare una stringa (i.e. η) determiniamone un carattere alla volta.

- Immagino di aver determinato $\eta[1, ..., k]$ e voglio $\eta[k+1]$.

  Diciamo che $\eta[1, ..., k]$ allinei $\sigma_1[1, ..., i-1]$ e $\sigma_2[1, ..., j-1]$.
- 4 casi: $\eta[k+1] \in \{\mathbf{m}, \mathbf{s}, \mathbf{i}, \mathbf{d}\}$.

Programmazione dinamica per l'allineamento globale

Idea!
Se dobbiamo trovare una stringa (i.e. η) determiniamone un carattere alla volta.

1,2 Se $\eta[k+1] \in \{\mathbf{m}, \mathbf{s}\}$, allora $\eta[1, ..., k+1]$ allinea $\sigma_1[1, ..., i]$ e $\sigma_2[1, ..., j]$.

3 Se $\eta[k+1] \in \{\mathbf{i}\}$, allora $\eta[1, ..., k+1]$ allinea $\sigma_1[1, ..., i-1]$ e $\sigma_2[1, ..., j]$.

4 Se $\eta[k+1] \in \{\mathbf{d}\}$, allora $\eta[1, ..., k+1]$ allinea $\sigma_1[1, ..., i]$ e $\sigma_2[1, ..., j-1]$.

Programmazione dinamica per l'allineamento globale

Idea!
Se dobbiamo trovare una stringa (i.e. η) determiniamone un carattere alla volta.

1,2 Se $\eta[k+1] \in \{\mathbf{m}, \mathbf{s}\}$, allora $\eta[1, ..., k+1]$ allinea $\sigma_1[1, ..., i]$ e $\sigma_2[1, ..., j]$.

3 Se $\eta[k+1] \in \{\mathbf{i}\}$, allora $\eta[1, ..., k+1]$ allinea $\sigma_1[1, ..., i-1]$ e $\sigma_2[1, ..., j]$.

4 Se $\eta[k+1] \in \{\mathbf{d}\}$, allora $\eta[1, ..., k+1]$ allinea $\sigma_1[1, ..., i]$ e $\sigma_2[1, ..., j-1]$.

Determino $\eta[k+1]$ scegliendo quello che mi da il costo minimo.

Algoritmo ricorsivo
... esponenziale

Algorithm 1: *allineamento_exp*$(\sigma_1, \sigma_2, i, j)$

if $i == 0$ *or* $j == 0$ **then** `// casi base della ricorsione`
  return $i + j$

if $\sigma_1[i] == \sigma_2[j]$ **then**
  $c_{\mathbf{m}} \leftarrow$ *allineamento_exp*$(\sigma_1, \sigma_2, i-1, j-1)$;
else
  $c_{\mathbf{s}} \leftarrow$ *allineamento_exp*$(\sigma_1, \sigma_2, i-1, j-1) + 1$;

$c_{\mathbf{i}} \leftarrow$ *allineamento_exp*$(\sigma_1, \sigma_2, i-1, j) + 1$;

$c_{\mathbf{d}} \leftarrow$ *allineamento_exp*$(\sigma_1, \sigma_2, i, j-1) + 1$;

return $\min(c_{\mathbf{m}}, c_{\mathbf{s}}, c_{\mathbf{i}}, c_{\mathbf{d}})$;

(Super)Ottimizzazione

Ci sono al più $|\sigma_1| \cdot |\sigma_2|$ valori per le chiamate ricorsive ...
Posso *tenere nota* dei risultati di tutti i possibili allineamenti parziali in una matrice.

$A(i, j)$ $=$ (costo dell') allineamento di $\sigma_1[1, ..., i]$ con $\sigma_2[1, ..., j]$

Ci sono al più $|\sigma_1| \cdot |\sigma_2|$ valori per le chiamate ricorsive ...
Posso *tenere nota* dei risultati di tutti i possibili allineamenti parziali in una matrice.

$A(i,j)$ = (costo dell') allineamento di $\sigma_1[1,...,i]$ con $\sigma_2[1,...,j]$

	ϵ		j	
ϵ				
			$\vdots$	
i	$\cdots$		$d_L(i,j)$	$\cdots$
			$\vdots$	

Alogoritmo di programmazione dinamica

Algorithm 2: *allineamento_prog_din*$(\sigma_1, \sigma_2, i, j)$

for $i = 0; i \leq |\sigma_1|; i++$ **do** `// inizializzo (la prima colonna di` A`)`
  $A(i, 0) = i$

for $j = 0; j \leq |\sigma_2|; j++$ **do** `// inizializzo (la prima riga di` A`)`
  $A(0, j) = j$

`// completo` A `procedendo` $\downarrow$ `sulle colonne)`

for $j = 0; j \leq |\sigma_2|; j++$ **do**
  for $i = 0; i \leq |\sigma_1|; i++$ **do**
    if $\sigma_1[i] == \sigma_2[j]$ **then**
      $A(i, j) = \min\{A(i-1, j-1), A(i-1, j) + 1, A(i, j-1) + 1\};$
    else
      $A(i, j) = \min\{A(i-1, j-1) + 1, A(i-1, j) + 1, A(i, j-1) + 1\};$

Example

	ϵ	**a**	**c**	**g**	**t**	**c**	**a**	**t**	**c**	**a**
ϵ	0	1	2	3	4	5	6	7	8	9
t	1	1	2	3	3	4	5	6	7	8
a	2	1	2	3	4	4	4	5	6	7
a	3	2	2	3	4	5	4	5	6	6
g	4	3	3	2	4	5	5	5	6	7
t	5	4	4	3	2	3	4	5	6	7
g	6	5	5	4	3	3	4	5	6	7
t	7	6	6	5	4	4	4	4	5	6
c	8	7	6	6	5	4	5	5	4	5
a	9	8	7	7	6	5	4	5	5	4

Example

	ϵ	**a**	**c**	**g**	**t**	**c**	**a**	**t**	**c**	**a**
ϵ	0	1	2	3	4	5	6	7	8	9
t	1	1	2	3	3	4	5	6	7	8
a	2	1	2	3	4	4	4	5	6	7
a	3	2	2	3	4	5	4	5	6	6
g	4	3	3	2	4	5	5	5	6	7
t	5	4	4	3	2	3	4	5	6	7
g	6	5	5	4	3	3	4	5	6	7
t	7	6	6	5	4	4	4	4	5	6
c	8	7	6	6	5	4	5	5	4	5
a	9	8	7	7	6	5	4	5	5	4

$$\begin{array}{lcccccccccc} \sigma_1 \equiv & & a & c & g & t & c & a & t & c & a \\ & \mathbf{i} & \mathbf{m} & \mathbf{s} & \mathbf{m} & \mathbf{m} & \mathbf{s} & \mathbf{d} & \mathbf{m} & \mathbf{m} & \mathbf{m} \\ \sigma_2 \equiv & t & a & a & g & t & g & & t & c & a \end{array} \Rightarrow \quad d_L(\sigma_1, \sigma_2) = 4.$$

Example

	ϵ	**a**	**c**	**g**	**t**	**c**	**a**	**t**	**c**	**a**
ϵ	0	1	2	3	4	5	6	7	8	9
t	1	1	2	3	3	4	5	6	7	8
a	2	1	2	3	4	4	4	5	6	7
a	3	2	2	3	4	5	4	5	6	6
g	4	3	3	2	4	5	5	5	6	7
t	5	4	4	3	2	3	4	5	6	7
g	6	5	5	4	3	3	4	5	6	7
t	7	6	6	5	4	4	4	4	5	6
c	8	7	6	6	5	4	5	5	4	5
a	9	8	7	7	6	5	4	5	5	4

$$\begin{array}{lcccccccccc} \sigma_1 \equiv & & a & c & g & t & c & a & t & c & a \\ & \mathbf{i} & \mathbf{m} & \mathbf{s} & \mathbf{m} & \mathbf{m} & \mathbf{s} & \mathbf{d} & \mathbf{m} & \mathbf{m} & \mathbf{m} \\ \sigma_2 \equiv & t & a & a & g & t & g & & t & c & a \end{array} \Rightarrow \quad d_L(\sigma_1, \sigma_2) = 4.$$

Example

	ϵ	**a**	**c**	**g**	**t**	**c**	**a**	**t**	**c**	**a**
ϵ	0	1	2	3	4	5	6	7	8	9
t	1	1	2	3	3	4	5	6	7	8
a	2	1	2	3	4	4	4	5	6	7
a	3	2	2	3	4	5	4	5	6	6
g	4	3	3	2	4	5	5	5	6	7
t	5	4	4	3	2	3	4	5	6	7
g	6	5	5	4	3	3	4	5	6	7
t	7	6	6	5	4	4	4	4	5	6
c	8	7	6	6	5	4	5	5	4	5
a	9	8	7	7	6	5	4	5	5	4

$$\begin{array}{lcccccccccc} \sigma_1 \equiv & & a & c & g & t & c & a & t & c & a \\ & \mathbf{i} & \mathbf{m} & \mathbf{s} & \mathbf{m} & \mathbf{m} & \mathbf{s} & \mathbf{d} & \mathbf{m} & \mathbf{m} & \mathbf{m} \\ \sigma_2 \equiv & t & a & a & g & t & g & & t & c & a \end{array} \Rightarrow \quad d_L(\sigma_1, \sigma_2) = 4.$$

Example

	ϵ	**a**	**c**	**g**	**t**	**c**	**a**	**t**	**c**	**a**
ϵ	0	1	2	3	4	5	6	7	8	9
t	1	1	2	3	3	4	5	6	7	8
a	2	1	2	3	4	4	4	5	6	7
a	3	2	2	3	4	5	4	5	6	6
g	4	3	3	2	4	5	5	5	6	7
t	5	4	4	3	2	3	4	5	6	7
g	6	5	5	4	3	3	4	5	6	7
t	7	6	6	5	4	4	4	4	5	6
c	8	7	6	6	5	4	5	5	4	5
a	9	8	7	7	6	5	4	5	5	4

$$\begin{array}{lcccccccccc} \sigma_1 \equiv & & a & c & g & t & c & a & t & c & a \\ & \mathbf{i} & \mathbf{m} & \mathbf{s} & \mathbf{m} & \mathbf{m} & \mathbf{s} & \mathbf{d} & \mathbf{m} & \mathbf{m} & \mathbf{m} \\ \sigma_2 \equiv & t & a & a & g & t & g & & t & c & a \end{array} \Rightarrow \quad d_L(\sigma_1, \sigma_2) = 4.$$

Example

	ϵ	**a**	**c**	**g**	**t**	**c**	**a**	**t**	**c**	**a**
ϵ	0	1	2	3	4	5	6	7	8	9
t	1	1	2	3	3	4	5	6	7	8
a	2	1	2	3	4	4	4	5	6	7
a	3	2	2	3	4	5	4	5	6	6
g	4	3	3	2	4	5	5	5	6	7
t	5	4	4	3	2	3	4	5	6	7
g	6	5	5	4	3	3	4	5	6	7
t	7	6	6	5	4	4	4	4	5	6
c	8	7	6	6	5	4	5	5	4	5
a	9	8	7	7	6	5	4	5	5	4

$$\begin{array}{lcccccccccc} \sigma_1 \equiv & & a & c & g & t & c & a & t & c & a \\ & \mathbf{i} & \mathbf{m} & \mathbf{s} & \mathbf{m} & \mathbf{m} & \mathbf{s} & \mathbf{d} & \mathbf{m} & \mathbf{m} & \mathbf{m} \\ \sigma_2 \equiv & t & a & a & g & t & g & & t & c & a \end{array} \Rightarrow \quad d_L(\sigma_1, \sigma_2) = 4.$$

Example

	ϵ	**a**	**c**	**g**	**t**	**c**	**a**	**t**	**c**	**a**
ϵ	0	1	2	3	4	5	6	7	8	9
t	1	1	2	3	3	4	5	6	7	8
a	2	1	2	3	4	4	4	5	6	7
a	3	2	2	3	4	5	4	5	6	6
g	4	3	3	2	4	5	5	5	6	7
t	5	4	4	3	2	3	4	5	6	7
g	6	5	5	4	3	3	4	5	6	7
t	7	6	6	5	4	4	4	4	5	6
c	8	7	6	6	5	4	5	5	4	5
a	9	8	7	7	6	5	4	5	5	4

$$\begin{array}{lcccccccccc} \sigma_1 \equiv & & a & c & g & t & c & a & t & c & a \\ & \mathbf{i} & \mathbf{m} & \mathbf{s} & \mathbf{m} & \mathbf{m} & \mathbf{s} & \mathbf{d} & \mathbf{m} & \mathbf{m} & \mathbf{m} \\ \sigma_2 \equiv & t & a & a & g & t & g & & t & c & a \end{array} \Rightarrow \quad d_L(\sigma_1, \sigma_2) = 4.$$

Example

	ϵ	**a**	**c**	**g**	**t**	**c**	**a**	**t**	**c**	**a**
ϵ	0	1	2	3	4	5	6	7	8	9
t	1	1	2	3	3	4	5	6	7	8
a	2	1	2	3	4	4	4	5	6	7
a	3	2	2	3	4	5	4	5	6	6
g	4	3	3	2	4	5	5	5	6	7
t	5	4	4	3	2	3	4	5	6	7
g	6	5	5	4	3	3	4	5	6	7
t	7	6	6	5	4	4	4	4	5	6
c	8	7	6	6	5	4	5	5	4	5
a	9	8	7	7	6	5	4	5	5	4

$$\begin{array}{lcccccccccc} \sigma_1 \equiv & & a & c & g & t & c & a & t & c & a \\ & \mathbf{i} & \mathbf{m} & \mathbf{s} & \mathbf{m} & \mathbf{m} & \mathbf{s} & \mathbf{d} & \mathbf{m} & \mathbf{m} & \mathbf{m} \\ \sigma_2 \equiv & t & a & a & g & t & g & & t & c & a \end{array} \Rightarrow \quad d_L(\sigma_1, \sigma_2) = 4.$$

Example

	ϵ	**a**	**c**	**g**	**t**	**c**	**a**	**t**	**c**	**a**
ϵ	0	1	2	3	4	5	6	7	8	9
t	1	1	2	3	3	4	5	6	7	8
a	2	1	2	3	4	4	4	5	6	7
a	3	2	2	3	4	5	4	5	6	6
g	4	3	3	2	4	5	5	5	6	7
t	5	4	4	3	2	3	4	5	6	7
g	6	5	5	4	3	3	4	5	6	7
t	7	6	6	5	4	4	4	4	5	6
c	8	7	6	6	5	4	5	5	4	5
a	9	8	7	7	6	5	4	5	5	4

$$\begin{array}{lcccccccccc} \sigma_1 \equiv & & a & c & g & t & c & a & t & c & a \\ & \mathbf{i} & \mathbf{m} & \mathbf{s} & \mathbf{m} & \mathbf{m} & \mathbf{s} & \mathbf{d} & \mathbf{m} & \mathbf{m} & \mathbf{m} \\ \sigma_2 \equiv & t & a & a & g & t & g & & t & c & a \end{array} \Rightarrow \quad d_L(\sigma_1, \sigma_2) = 4.$$

Example

	ϵ	**a**	**c**	**g**	**t**	**c**	**a**	**t**	**c**	**a**
ϵ	0	1	2	3	4	5	6	7	8	9
t	1	1	2	3	3	4	5	6	7	8
a	2	1	2	3	4	4	4	5	6	7
a	3	2	2	3	4	5	4	5	6	6
g	4	3	3	2	4	5	5	5	6	7
t	5	4	4	3	2	3	4	5	6	7
g	6	5	5	4	3	3	4	5	6	7
t	7	6	6	5	4	4	4	4	5	6
c	8	7	6	6	5	4	5	5	4	5
a	9	8	7	7	6	5	4	5	5	4

$$\begin{array}{lcccccccccc} \sigma_1 \equiv & & a & c & g & t & c & a & t & c & a \\ & \mathbf{i} & \mathbf{m} & \mathbf{s} & \mathbf{m} & \mathbf{m} & \mathbf{s} & \mathbf{d} & \mathbf{m} & \mathbf{m} & \mathbf{m} \\ \sigma_2 \equiv & t & a & a & g & t & g & & t & c & a \end{array} \Rightarrow \quad d_L(\sigma_1, \sigma_2) = 4.$$

Example

	ϵ	**a**	**c**	**g**	**t**	**c**	**a**	**t**	**c**	**a**
ϵ	0	1	2	3	4	5	6	7	8	9
t	1	1	2	3	3	4	5	6	7	8
a	2	1	2	3	4	4	4	5	6	7
a	3	2	2	3	4	5	4	5	6	6
g	4	3	3	2	4	5	5	5	6	7
t	5	4	4	3	2	3	4	5	6	7
g	6	5	5	4	3	3	4	5	6	7
t	7	6	6	5	4	4	4	4	5	6
c	8	7	6	6	5	4	5	5	4	5
a	9	8	7	7	6	5	4	5	5	4

$$\sigma_1 \equiv \quad a \quad c \quad g \quad t \quad c \quad a \quad t \quad c \quad a$$

$$\quad\quad \mathbf{i}\ \ \mathbf{m}\ \ \mathbf{s}\ \ \mathbf{m}\ \ \mathbf{m}\ \ \mathbf{s}\ \ \mathbf{d}\ \ \mathbf{m}\ \ \mathbf{m}\ \ \mathbf{m} \Rightarrow \quad d_L(\sigma_1, \sigma_2) = 4.$$

$$\sigma_2 \equiv \quad t \quad a \quad a \quad g \quad t \quad g \quad t \quad c \quad a$$

Example

	ϵ	**a**	**c**	**g**	**t**	**c**	**a**	**t**	**c**	**a**
ϵ	0	1	2	3	4	5	6	7	8	9
t	1	1	2	3	3	4	5	6	7	8
a	2	1	2	3	4	4	4	5	6	7
a	3	2	2	3	4	5	4	5	6	6
g	4	3	3	2	4	5	5	5	6	7
t	5	4	4	3	2	3	4	5	6	7
g	6	5	5	4	3	3	4	5	6	7
t	7	6	6	5	4	4	4	4	5	6
c	8	7	6	6	5	4	5	5	4	5
a	9	8	7	7	6	5	4	5	5	4

$$\sigma_1 \equiv \quad a \quad c \quad g \quad t \quad c \quad a \quad t \quad c \quad a$$

$$\mathbf{i} \quad \mathbf{m} \quad \mathbf{s} \quad \mathbf{m} \quad \mathbf{m} \quad \mathbf{s} \quad \mathbf{d} \quad \mathbf{m} \quad \mathbf{m} \quad \mathbf{m} \Rightarrow \quad d_L(\sigma_1, \sigma_2) = 4.$$

$$\sigma_2 \equiv \quad t \quad a \quad a \quad g \quad t \quad g \quad t \quad c \quad a$$

Perché è complesso?

Domanda:

Cosa ho *veramente* computato?

Risposta:

L'allineamento di un prefisso di σ_1 con un prefisso di σ_2

σ_1 *pattern (query)* σ_2 *testo (reference, data-base, ...)* e ci interessano i **suffissi dei prefissi di** σ_2.

Allineamento Locale

Definition
L'allineamento locale (a distanza $\leq d$) di σ_1 in σ_2 è l'insieme di tutte le coppie $\langle i, \eta_i \rangle$ tali che η_i è l'allineamento di σ_1 e di un *prefisso* di $\sigma_2[i, ...]$ (a distanza $\leq d$).

N.B.
σ_1 *pattern (query)* σ_2 *testo (reference, data base, ...)* e ci interessano i **prefissi dei suffissi di** σ_2.

Basta ...

Allineamento Locale

Definition
L'allineamento locale (a distanza $\leq d$) di σ_1 in σ_2 è l'insieme di tutte le coppie $\langle i, \eta_i \rangle$ tali che η_i è l'allineamento di σ_1 e di un *prefisso* di $\sigma_2[i, ...]$ (a distanza $\leq d$).

N.B.
σ_1 *pattern (query)* σ_2 *testo (reference, data base, ...)* e ci interessano i **prefissi dei suffissi di** σ_2.

Basta ...

... compilare la matrice di allineamento ponendo tutti 0 nella prima riga.

Allineamento Locale

Example

	ϵ	**a**	**c**	**g**	**t**	**a**	**c**	**g**	**c**	**g**	**t**	**a**	**c**	**a**	**g**	**t**	**a**	$\cdots$
ϵ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	$\cdots$
c	1	1	0	1	1	1	0	1	0	1	1	1	0	1	1	1	1	$\cdots$
g	2	2	1	0	1	2	1	0	1	0	1	2	1	1	1	2	2	$\cdots$
t	3	3	2	2	0	1	2	1	1	1	0	1	2	2	2	1	2	$\cdots$
a	4	3	3	3	1	0	1	2	2	2	1	0	1	2	3	2	1	$\cdots$

Allineamento Locale

Example

	ϵ	**a**	**c**	**g**	**t**	**a**	**c**	**g**	**c**	**g**	**t**	**a**	**c**	**a**	**g**	**t**	**a**	$\cdots$
ϵ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	$\cdots$
c	1	1	0	1	1	1	0	1	0	1	1	1	0	1	1	1	1	$\cdots$
g	2	2	1	0	1	2	1	0	1	0	1	2	1	1	1	2	2	$\cdots$
t	3	3	2	2	0	1	2	1	1	1	0	1	2	2	2	1	2	$\cdots$
a	4	3	3	3	1	0	1	2	2	2	1	0	1	2	3	2	1	$\cdots$

Abbiamo introdotto un *gap inizialie a costo zero*

Funzioni di costo per i gap

- Lineare: un gap di lunghezza n ha costo pari a $\gamma(n) \equiv n \cdot g$
- Convesso: un gap di lunghezza n ha costo pari a $\gamma(n)$ tale che

$$\gamma(i+1) - \gamma(i) \leq \gamma(i) - \gamma(i-1)$$

- Affine: un gap di lunghezza n ha costo pari a $\gamma(n) \equiv g_o + (n-1) \cdot g_e$

Funzioni di costo per i gap

- Lineare: un gap di lunghezza n ha costo pari a $\gamma(n) \equiv n \cdot g$
- Convesso: un gap di lunghezza n ha costo pari a $\gamma(n)$ tale che

$$\gamma(i+1) - \gamma(i) \leq \gamma(i) - \gamma(i-1)$$

- Affine: un gap di lunghezza n ha costo pari a $\gamma(n) \equiv g_o + (n-1) \cdot g_e$

L'equazione di ricorrenza

$$A(i,j) \quad = \quad max \begin{cases} A(i-1, j-1) + score(\sigma_1[i], \sigma_1[j]) \\ max\{A(k,j) - \gamma(i-k) \mid k = 0, \ldots, i-1\} \\ max\{A(i,k) - \gamma(j-k) \mid k = 0, \ldots, j-1\} \end{cases}$$

Storia

- Needleman e Wunsh (1970) allineamento globale
- Smith e Waterman (1981) allineamento locale

Faint similarity strong (local) similarity

$\mathbf{S_1}$ *NGPEVRELW*

$\mathbf{S_2}$ *ARDIWA*

$\mathbf{S_3}$ *QARESIYA*

$\mathbf{S_4}$ *VRESLWS*

$\mathbf{S_5}$ *WYVRDASLWS*

Multiple alignment

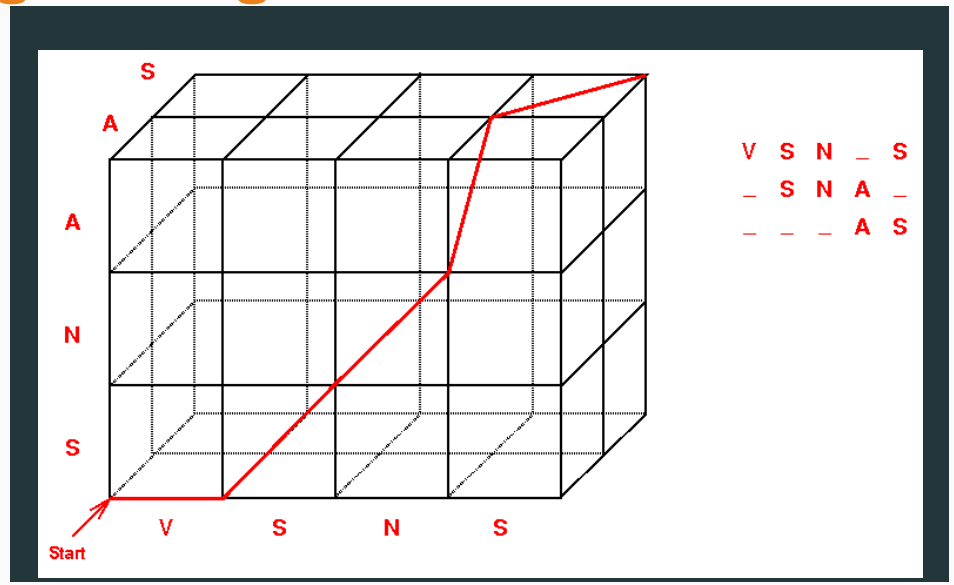
Faint similarity strong (local) similarity

$\mathbf{S_1}$	N	G	P	E	V	R	E	$-$	$-$	L	W	$-$
$\mathbf{S_2}$	$-$	$-$	$-$	$-$	A	R	D	$-$	$-$	I	W	A
$\mathbf{S_3}$	$-$	$-$	$-$	Q	A	R	E	$-$	S	I	Y	A
$\mathbf{S_4}$	$-$	$-$	$-$	$-$	V	R	E	$-$	S	L	W	S
$\mathbf{S_5}$	$-$	$-$	W	Y	V	R	D	A	S	L	W	S

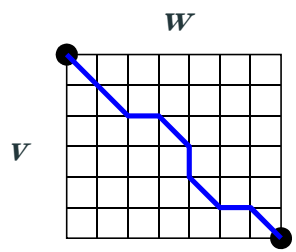
Definition
Sum-of-pairs score of a Multiple Sequence Alignment A:

$$Sc(A) = \sum_{i<j} Sc(P_{i,j}(A))$$

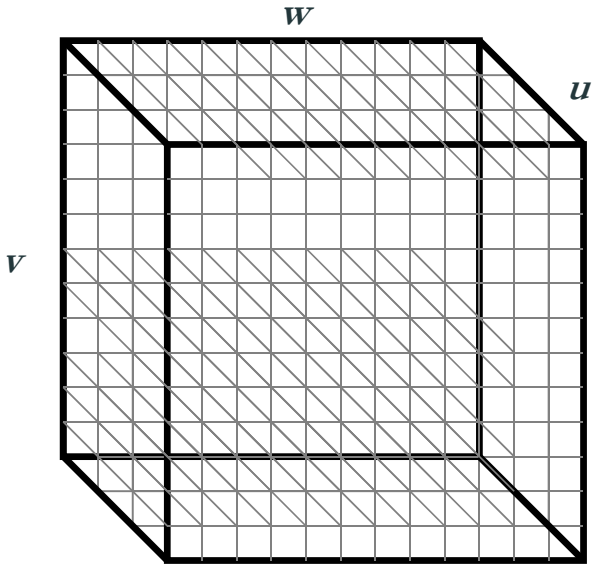
Dynamic Programming?



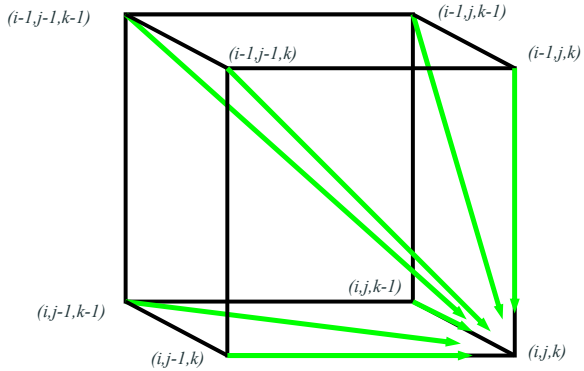
Multiple alignment



2-D edit graph



3-D edit graph



Multiple alignment

Heuristics
Fast algorithms, that do not necessarily find the mathematically optimal alignment.

Two groups $\mathcal{G}_1, \mathcal{G}_2$ of sequences aligned by A_1, A_2 can be aligned like two single sequences if A_1, A_2remain unchanged.

Approach: progressive alignment
Align sucessively sequences and groups of previously aligned sequences, until all sequences are aligned in one MSA.

Progressive alignment: example

$\mathbf{S_1}$	E	V	R	E	$-$	V	W	$-$
$\mathbf{S_2}$	$-$	A	R	D	$-$	T	W	A
$\mathbf{S_3}$	Q	A	R	E	S	I	Y	A
$\mathbf{S_4}$	$-$	$-$	R	E	S	L	W	S

$\mathbf{S_5}$	R	E	W	S	L	W	S
$\mathbf{S_6}$	R	E	Y	S	$-$	$-$	S

Progressive alignment: example

$\mathbf{S_1}$	E	V	R	E	$-$	$-$	V	W	$-$
$\mathbf{S_2}$	$-$	A	R	D	$-$	$-$	T	W	A
$\mathbf{S_3}$	Q	A	R	E	$-$	S	I	Y	A
$\mathbf{S_4}$	$-$	$-$	R	E	$-$	S	L	W	S

$\mathbf{S_5}$	$-$	$-$	R	E	W	S	L	W	S
$\mathbf{S_6}$	$-$	$-$	R	E	Y	S	$-$	$-$	S

General strategy

- Construct phylogenetic tree of input sequences $\mathbf{S}_1, ..., \mathbf{S}_n$ ('guide tree').
- Traverse T from leaves to root
- At every inner node, construct profile alignments of sequences corresponding to daughter nodes

Implementation

`ClustalΩ`

Dot-plot

	A	A	T	C	T	T	C	A	G	C	G	T	A	T	T	G	C	T
A	1	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0
T	0	0	1	0	1	1	0	0	0	0	0	1	0	1	1	0	0	1
C	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	1	0
T	0	0	1	0	1	1	0	0	0	0	0	1	0	1	1	0	0	1
T	0	0	1	0	1	1	0	0	0	0	0	1	0	1	1	0	0	1
A	1	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0
G	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1	0	0
C	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	1	0
C	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	1	0
G	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1	0	0
G	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1	0	0
A	1	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0
G	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1	0	0
G	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1	0	0
T	0	0	1	0	1	1	0	0	0	0	0	1	0	1	1	0	0	1
A	1	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0
T	0	0	1	0	1	1	0	0	0	0	0	1	0	1	1	0	0	1
T	0	0	1	0	1	1	0	0	0	0	0	1	0	1	1	0	0	1

Dot-plot

	A	A	T	C	T	T	C	A	G	C	G	T	A	T	T	G	C	T
A	1	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0
T	0	0	1	0	1	1	0	0	0	0	0	1	0	1	1	0	0	1
C	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	1	0
T	0	0	1	0	1	1	0	0	0	0	0	1	0	1	1	0	0	1
T	0	0	1	0	1	1	0	0	0	0	0	1	0	1	1	0	0	1
A	1	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0
G	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1	0	0
C	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	1	0
C	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	1	0
G	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1	0	0
G	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1	0	0
A	1	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0
G	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1	0	0
G	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1	0	0
T	0	0	1	0	1	1	0	0	0	0	0	1	0	1	1	0	0	1
A	1	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0
T	0	0	1	0	1	1	0	0	0	0	0	1	0	1	1	0	0	1
T	0	0	1	0	1	1	0	0	0	0	0	1	0	1	1	0	0	1

Le diagonali di 1 ...

... possono essere molto utili!

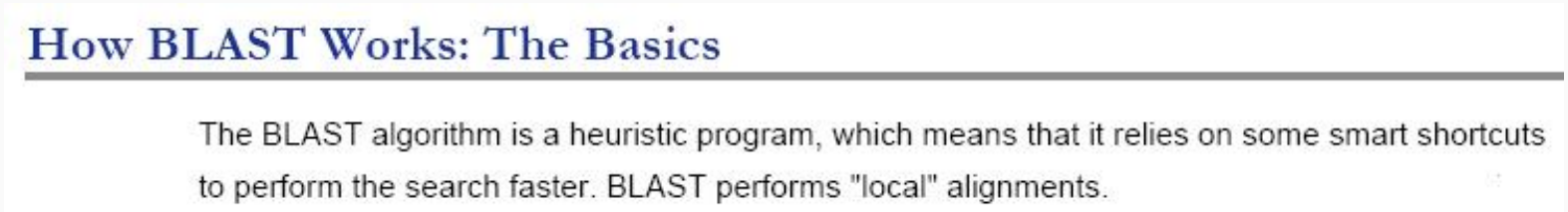
BLAST

Idea
Perché non usare “pezzi” di sequenza uguali *pre-computati*?

BLAST

Idea
Perché non usare “pezzi” di sequenza uguali *pre-computati*?

Dal capitolo 16 dell’NCBI handbook



How BLAST Works: The Basics

The BLAST algorithm is a heuristic program, which means that it relies on some smart shortcuts to perform the search faster. BLAST performs "local" alignments.

...

When a query is submitted via one of the BLAST Web pages, the sequence, plus any other input information such as the database to be searched, word size, expect value, and so on, are fed to the algorithm [`http://www.ncbi.nlm.nih.gov/Education/BLASTinfo/BLAST_algorithm.html`] on the BLAST server. BLAST works by first making a look-up table of all the "*words*" (short subsequences, which for proteins the default is three letters) and "neighboring words", i.e., similar words in the query sequence. The sequence database is then scanned for these "hot spots". When a match is identified, it is used to initiate gap-free and gapped extensions of the "word".

...

BLAST stages:

First stage: Identify exact matches of length W (default W=3) between the query and the sequences in the database

Second stage: Extend the match in both directions in an attempt to boost the alignment score (insertions and deletions are not considered)

Third stage: If a high-scoring ungapped alignment is found: Perform a gapped local alignment using dynamic programming

Building an Index for a reference

What is an Index?

Indexing

Definition (Wikipedia)

- A Lookup table, a data structure, usually an array or associative array, often used to replace a runtime computation with a simpler array indexing operation
- Array index, an integer pointer (into an array data structure) that identifies an element of the array
- A key in an associative array
- Database index, a data structure that improves the speed of data retrieval operations on a database table
- Index mapping, mapping of raw data, used directly as in array index, for an array
- Index register, a processor register used for modifying operand addresses during the run of a program
- The dataset maintained by search engine indexing
- ...

Indexing

But ...

... the same idea can be used to build a lookup table of the reference when searching for short strings!

Reads as Numbers

Definition

- $\Sigma_{dna} = \{a, c, g, t\}$ alphabet;
- $\mathbf{P}$ pattern and $|\mathbf{P}| = m$;
- $\mathbf{T}$ text and $|\mathbf{T}| = n$;

T_s the m-character string $T[s, \ldots, s + m - 1]$.

Strings in Σ_{dna} are numbers in base 4

$$\mathbf{P} \rightsquigarrow p \qquad \mathbf{T}_s \rightsquigarrow t_s$$

$$a \equiv 0 \quad c \equiv 1 \quad g \equiv 2 \quad t \equiv 3$$

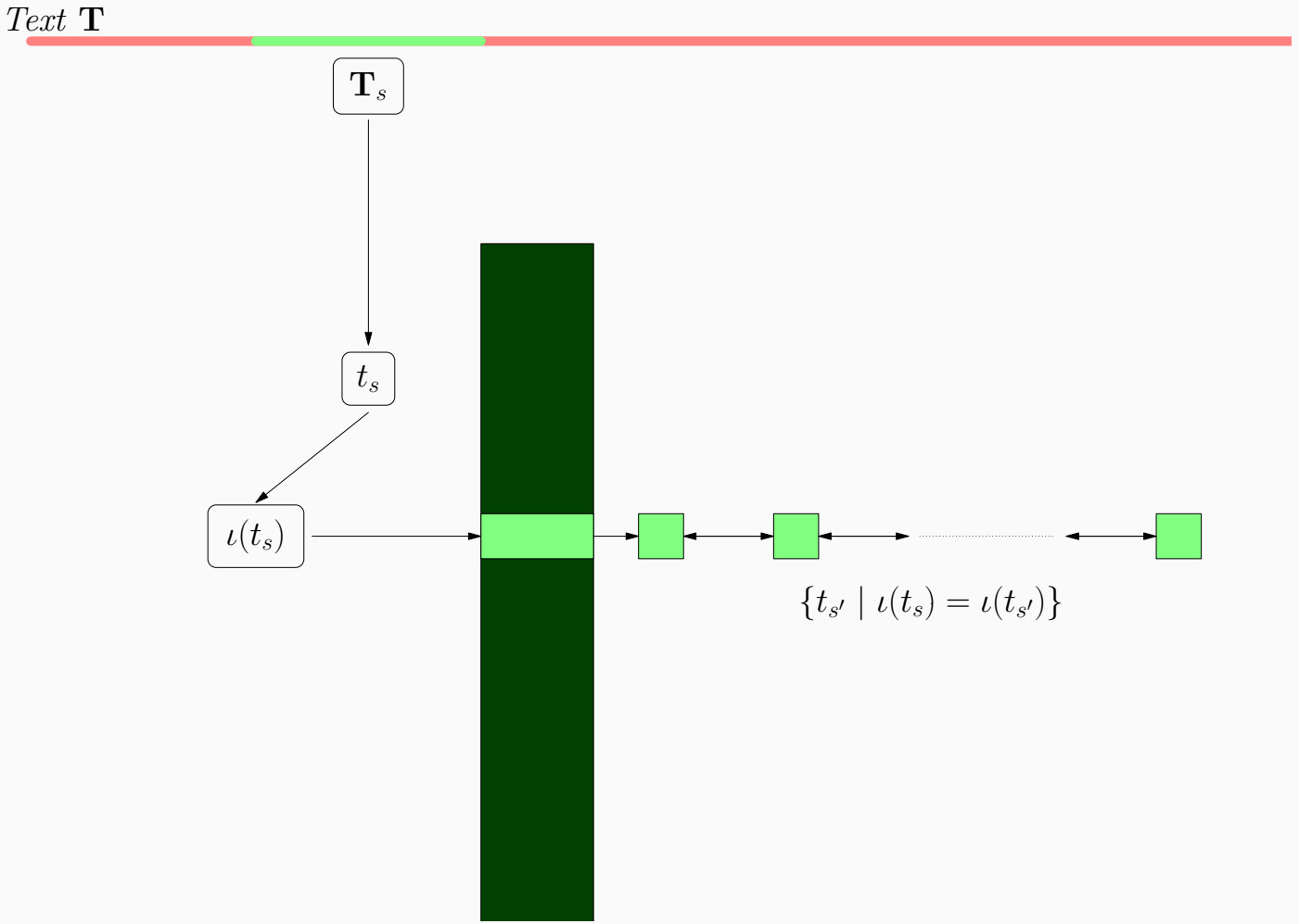
Indexing

Careful!

4^m is BIG

Use an index
A function ι mapping a—*large*—domain in a—*small*—array

Indexing



The problems

Mismatches (Insertions/Deletions)

1. We must search for a string and its variants
2. The index should allow to speed-up search. It must be structured

Strutture dati

Un passo indietro

Definition (Ricerca esatta)
Dato un *pattern* P ed un testo T, trova tutte le occorrenze esatte di P in T.

Un passo indietro

Definition (Ricerca esatta)
Dato un *pattern* P ed un testo T, trova tutte le occorrenze esatte di P in T.

Applicazioni?
Pattern corti.

Pattern lunghi se osservo che sotto-pattern corti *devono* occorrere in modo esatto.

Un passo indietro

Definition (Ricerca esatta)
Dato un *pattern* P ed un testo T, trova tutte le occorrenze esatte di P in T.

Applicazioni?
Pattern corti.

Pattern lunghi se osservo che sotto-pattern corti *devono* occorrere in modo esatto.

Complessità?

Un passo indietro

Definition (Ricerca esatta)
Dato un *pattern* P ed un testo T, trova tutte le occorrenze esatte di P in T.

Applicazioni?
Pattern corti.

Pattern lunghi se osservo che sotto-pattern corti *devono* occorrere in modo esatto.

Complessità?

$$|P| + |T|$$

Un passo indietro

Definition (Ricerca esatta)
Dato un *pattern* P ed un testo T, trova tutte le occorrenze esatte di P in T.

Applicazioni?
Pattern corti.

Pattern lunghi se osservo che sotto-pattern corti *devono* occorrere in modo esatto.

Complessità?

$$|P| + |T|$$

Si può fare di meglio?

Due Osservazioni

Lemma
Se P occorre in T a dstanza (Hamming/Levensthein) d, allora una sotto-stringa di P di lunghezza almeno $|P|/(d+1)$ occorre in modo esatto.

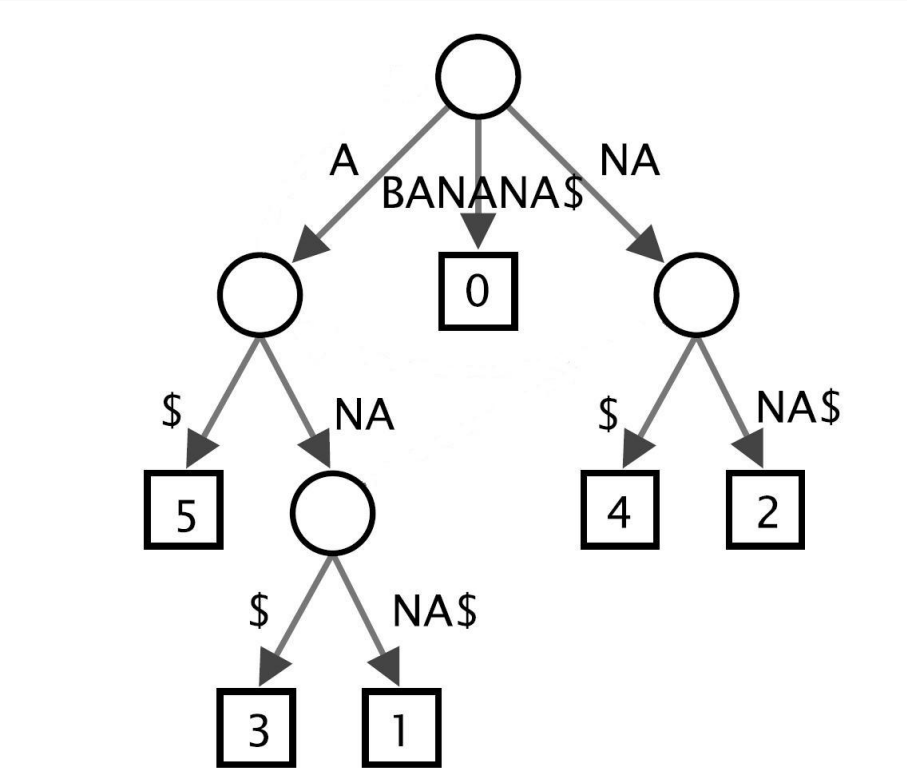
Due Osservazioni

Lemma
Se P occorre in T a dstanza (Hamming/Levensthein) d, allora una sotto-stringa di P di lunghezza almeno $|P|/(d+1)$ occorre in modo esatto.

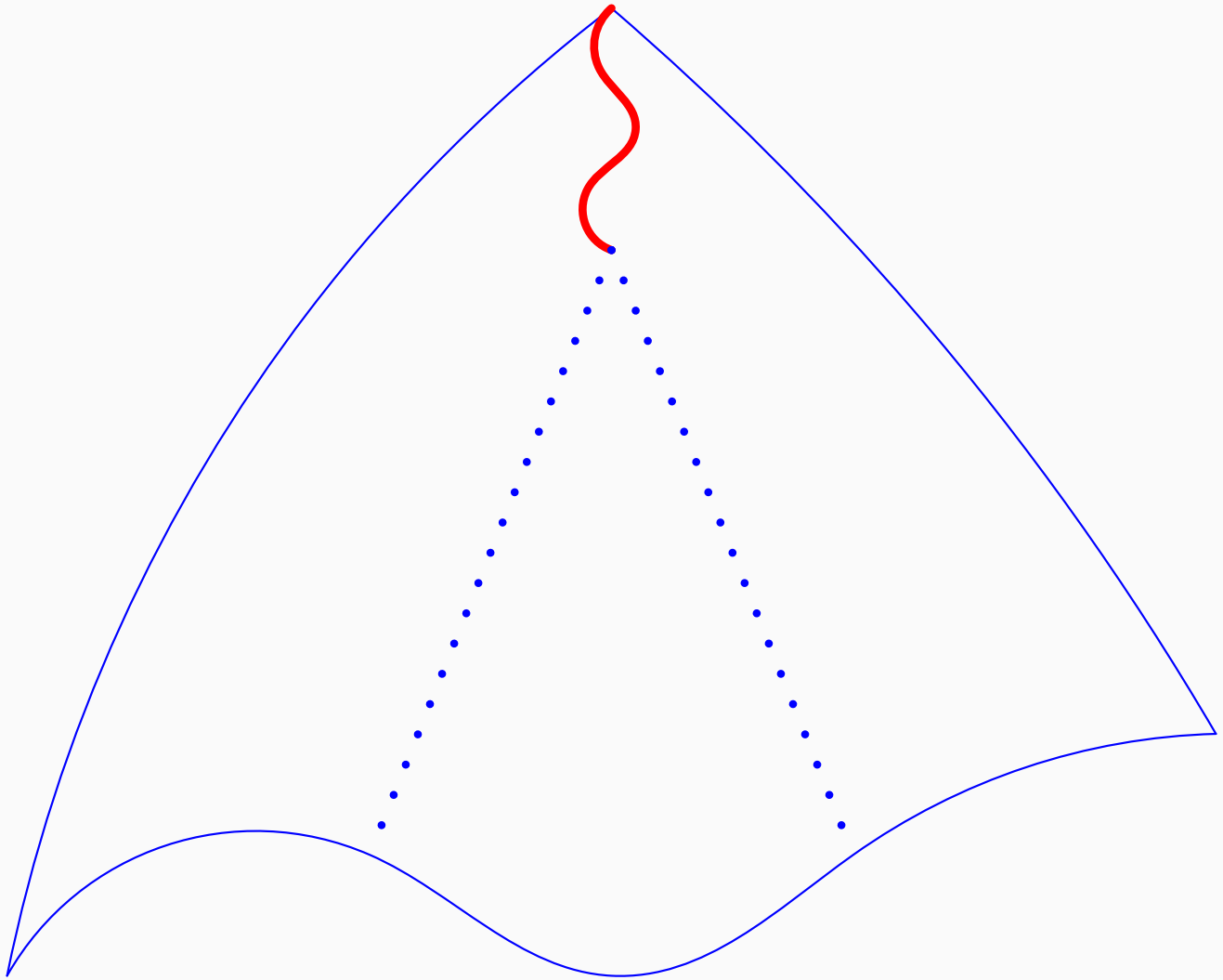
Il testo può essere memorizzato in una struttura dati
$\Rightarrow$ non sono obbligato a leggere tutto il testo per ogni pattern $\Rightarrow$ posso cercare P ad un costo $O(|P|)$!

Idea

1. memorizziamo tutti i suffissi di T
2. evitiamo di memorizzare due volte le stesse stringhe (prefissi)



B	A	N	A	N	A	$
0	1	2	3	4	5	



Suffix Arrays

i	$A_{SA[i]}$
0	acaaacatat$
1	caaacatat$
2	aaacatat$
3	aacatat$
4	acatat$
5	catat$
6	atat$
7	tat$
8	at$
9	t$
10	$

$T = acaaacatat \quad P = ata$

Suffix Arrays

i	SA	$A_{SA[i]}$
0	2	aaacatat$
1	3	aacatat$
2	0	acaaacatat$
3	4	acatat$
4	6	atat$
5	8	at$
6	1	caaacatat$
7	5	catat$
8	7	tat$
9	9	t$
10	10	$

$T = acaaacatat \quad P = ata$

Suffix Arrays

i	SA	$A_{SA[i]}$
0	2	aaacatat$
1	3	aacatat$
2	0	acaaacatat$
3	4	acatat$
4	6	atat$
5	8	at$
6	1	caaacatat$
7	5	catat$
8	7	tat$
9	9	t$
10	10	$

$T = acaaacatat\$\ P = ata$

Suffix Arrays

i	SA	$A_{SA[i]}$
0	2	aaacatat$
1	3	aacatat$
2	0	acaaacatat$
3	4	acatat$
4	6	atat$
5	8	at$
6	1	caaacatat$
7	5	catat$
8	7	tat$
9	9	t$
10	10	$

$T = acaaacatat\$\ P = ata$

Suffix Arrays

i	SA	$A_{SA[i]}$
0	2	aaacatat$
1	3	aacatat$
2	0	acaaacatat$
3	4	acatat$
4	6	atat$
5	8	at$
6	1	caaacatat$
7	5	catat$
8	7	tat$
9	9	t$
10	10	$

$T = acaaacatat\$\ P = ata$

Suffix Arrays

i	SA	$A_{SA[i]}$
0	2	aaacatat$
1	3	aacatat$
2	0	acaaacatat$
3	4	acatat$
4	6	atat$
5	8	at$
6	1	caaacatat$
7	5	catat$
8	7	tat$
9	9	t$
10	10	$

$T = acaaacatat\$\ P = ata$

Enhanced Suffix Arrays

I Suffix Array sono migliorabili?

- complessità della costruzione
- complessità della ricerca
- informazioni di supporto (*enhancement*)
- ... compressione: *si possono comprimere contemporaneamente indice e testo?*

La trasformata di Burrows-Wheeler

Passi per il calcolo

- Considero i primi caratteri

s	**wiss·miss·missing**
w	iss·miss·missing
i	ss·miss·missing
s	s·miss·missing
s	·miss·missing
·	miss·missing
m	iss·missing
i	ss·missing
s	s·missing
s	·missing
·	missing
m	issing
i	ssing
s	sing
s	ing
i	ng
n	g
g	

La trasformata di Burrows-Wheeler

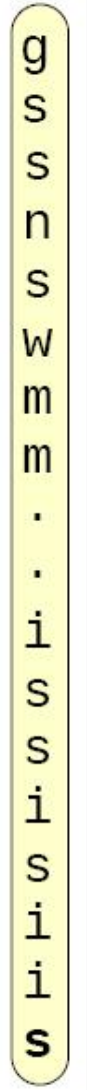
Passi per il calcolo

- Ordino lessicograficamente la parte azzurra

g	
s	·miss·missing
s	·missing
n	g
s	ing
w	iss·miss·missing
m	iss·missing
m	issing
·	miss·missing
·	missing
i	ng
s	s·miss·missing
s	s·missing
i	ss·miss·missing
s	sing
i	ss·missing
i	ssing
s	**wiss·miss·missing**

Passi per il calcolo

- Ottengo una permutazione della stringa di partenza



La trasformata di Burrows-Wheeler

- La BWT è reversibile: dalla stringa trasformata è possibile ricostruire il testo originale ...
- ... ma la stringa trasformata è più "facile" da comprimere in quanto "localmente omogenea".
- Si dimostra che la BWT permette di ottenere una compressione elevata usando compressori "semplici" cioè che non guardano il contesto dei simboli.
- Equivalentemente, comprimendo BWT(S) fino alla sua entropia H_0 e equivalente a comprimere S fino a H_k per ogni $k \geq 0$.