Allineamento di Sequenze
(e strutture dati)

Alberto Policriti

Dipartimento di Matematica e Informatica
Istituto di Genomica Applicata

7 Novembre, 2017

The Plan

Basics questions

1. what does it mean to *align*?
2. why should I care about *algorithms* to align?
3. data structures ... why?
4. time, space, correctness, completeness ... is this *all* bioinformaticians are interested on?

Our (operative) meaning to alignment

Aligning means ...

... converting

What do we mean by *aligning* two sequences?

Definition

An alignment of σ_1 and σ_2 is a *string* on the alphabet $\{\mathbf{m}, \mathbf{i}, \mathbf{d}, \mathbf{s}\}$

How do we define an *alignment*?

Forget about how to find an alignment or wether it is optimal (?)

Example

$$\begin{array}{cccccccccc} \sigma_1 \equiv & a & c & g & t & c & a & t & c & a \\ & \mathbf{i} & \mathbf{m} & \mathbf{s} & \mathbf{m} & \mathbf{m} & \mathbf{s} & \mathbf{d} & \mathbf{m} & \mathbf{m} & \mathbf{m} \\ \sigma_2 \equiv & t & a & a & g & t & g & & t & c & a \end{array}$$

What do we mean by *aligning* two sequences?

Definition

An alignment of σ_1 and σ_2 is a *string* on the alphabet $\{\mathbf{m}, \mathbf{i}, \mathbf{d}, \mathbf{s}\}$

How do we define an *alignment*?

Forget about how to find an alignment or wether it is optimal (?)

Example

$\sigma_1 \equiv$		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

What do we mean by *aligning* two sequences?

Definition

An alignment of σ_1 and σ_2 is a *string* on the alphabet $\{\mathbf{m}, \mathbf{i}, \mathbf{d}, \mathbf{s}\}$

How do we define an *alignment*?

Forget about how to find an alignment or wether it is optimal (?)

Example

$\sigma_1 \equiv$		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

What do we mean by *aligning* two sequences?

How do we define an *alignment*?

Forget about how to find an alignment or wether it is optimal (?)

Example

$$
\begin{array}{rcccccccccc}
\sigma_1 \equiv & & a & c & g & t & c & a & t & c & a \\
 & \mathbf{i} & \mathbf{m} & \mathbf{s} & \mathbf{m} & \mathbf{m} & \mathbf{s} & \mathbf{d} & \mathbf{m} & \mathbf{m} & \mathbf{m} \\
\sigma_2 \equiv & t & a & a & g & t & g & & t & c & a
\end{array}
$$

An alignment is a *string*

Alignment $\Leftrightarrow$ edit-string

In fact is a *"program"* converting σ_1 into σ_2 (or viceversa).

Terms and meaning

Ranking Alignments: Cost and Score

We can associate a cost to an edit-string (then we want $\downarrow$)

We can associate a score to an edit-string (then we want $\uparrow$)

Terms and meaning

Ranking Alignments: Cost and Score

We can associate a cost to an edit-string (then we want ↓)

We can associate a score to an edit-string (then we want ↑)

Assigning a score

matches (good): higher score (e.g. 1)

substitutions (bad): lower score (e.g. less than 0)

insertion/deletion (bad): lower score (e.g. less than 0)

Terms and meaning

Ranking Alignments: Cost and Score

We can associate a cost to an edit-string (then we want $\downarrow$)

We can associate a score to an edit-string (then we want $\uparrow$)

Example

Purines and Pyrimidines are similar, hence a (non-trivial) score matrix is the following:

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

Terms and meaning

Ranking Alignments: Cost and Score

We can associate a cost to an edit-string (then we want ↓)

We can associate a score to an edit-string (then we want ↑)

Example

Affine gap penalty:

penalize the opening of a gap
and
penalise its extension

(... how exactly?)

Terms and meaning

Global

The edit-string converts the *entire* σ_2 in *entire* σ_1.

Local

The edit-string converts the *entire* σ_2 in *a portion* of σ_1.

Terms and meaning

Distance

Think of the cost as a *distance* among strings.

Terms and meaning

Distance

Think of the cost as a *distance* among strings.

Example

Levensthein distance: cost 0 for **m**, 1 for the others.

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

Hamming distance: **i**'s and **d**'s not allowed.

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

Hamming distance can be used on equal length strings only.

Terms and meaning

Ranking Alignments: Cost and Score

We can associate a cost to an edit-string (then we want $\downarrow$)

We can associate a score to an edit-string (then we want $\uparrow$)

Global

The edit-string converts the *entire* σ_2 in *entire* σ_1.

Local

The edit-string converts the *entire* σ_2 in *a portion* of σ_1.

Distance

Think of the cost as a *distance* among strings.

Rule of thumb

Which distance should I use?

Long strings $\Leftrightarrow$ Levensthein

Short strings $\Leftrightarrow$ Hamming

Rule of thumb

Which distance & tool should I use?

Long strings $\Leftrightarrow$ Levensthein $\Leftrightarrow$ S.W., BLAST, BWA-MEM, ...

Short strings $\Leftrightarrow$ Hamming $\Leftrightarrow$ Erne, BWA, bowtie, ...

Rule of thumb

Which distance & tool & computer should I use?

Long strings $\Leftrightarrow$ Levensthein $\Leftrightarrow$ S.W., BLAST, BWA-MEM, ...

Short strings $\Leftrightarrow$ Hamming $\Leftrightarrow$ Erne, BWA, bowtie, ...

Always ... a powerful one (better if parallel)

$|\sigma|$ stands for the length of σ.
$O(|\sigma|)$ stands for *proportional* to the length of σ.

Global alignment: complexity

Best Hamming distance	Levensthein distance d	Levensthein
$O(\|\sigma_1\| + \|\sigma_2\|)$	$O(\|\sigma_1\| + \|\sigma_2\|)$	$O(\|\sigma_1\| * \|\sigma_2\|)$

$|\sigma|$ stands for the length of σ.
$O(|\sigma|)$ stands for *proportional* to the length of σ.

Global alignment: complexity

Best Hamming distance	Levensthein distance d	Levensthein
$O(\|\sigma_1\| + \|\sigma_2\|)$	$O(\|\sigma_1\| + \|\sigma_2\|)$	$O(\|\sigma_1\| * \|\sigma_2\|)$

Denote by ρ the reference (i.e. think of ρ as *large*).

Local alignment: complexity

Best Hamming distance	Levensthein distance d	Levensthein
$O(\|\sigma_1\| + \|\rho\|)$	$O(\|\sigma_1\| + \|\rho\|)$	$\underline{O(\|\sigma_1\| * \|\rho\|)}$

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

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

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

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

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

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

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

Algoritmo ricorsivo

... esponenziale
(vedi anche Cabada A., Nieto J.J., Torres A.: *An exact formula for the number of aligments between two DNA sequences.)*

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

Programmazione dinamica per l'allineamento globale

(Super)Ottimizzazione

Posso *tenere nota* dei risultati di tutti i possibili allineamenti parziali in una matrice.

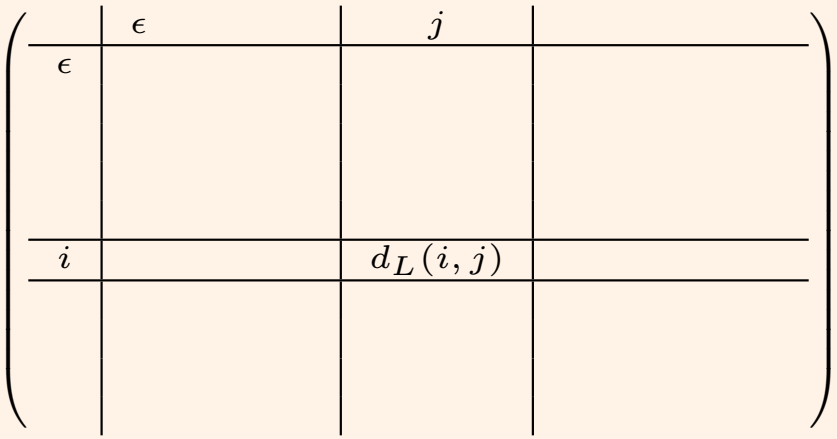
$$A(i,j) \quad = \quad \text{(costo dell')} \text{ allineamento di } \sigma_1[1,...,i] \text{ con } \sigma_2[1,...,j]$$

Programmazione dinamica per l'allineamento globale

(Super)Ottimizzazione

Posso *tenere nota* dei risultati di tutti i possibili allineamenti parziali in una matrice.

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



Programmazione dinamica per l'allineamento globale

Example

Matrice di programmazione dinamica (da non confondere con la matrice dei costi di allineamento).

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

Programmazione dinamica per l'allineamento globale

Example

Matrice di programmazione dinamica (da non confondere con la matrice dei costi di allineamento).

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

Programmazione dinamica per l'allineamento globale

Example

Matrice di programmazione dinamica (da non confondere con la matrice dei costi di allineamento).

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

Programmazione dinamica per l'allineamento globale

Example

Matrice di programmazione dinamica (da non confondere con la matrice dei costi di allineamento).

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

Perché è complesso?

Domanda:

Cosa ho *veramente* computato?

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

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

Allineamento Locale

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

Abbiamo introdotto un *gap inizialie a costo zero*

Gap

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 $\gamma(n) \equiv g_o + (n-1) \cdot g_e$

Gap

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

Multiple alignment

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

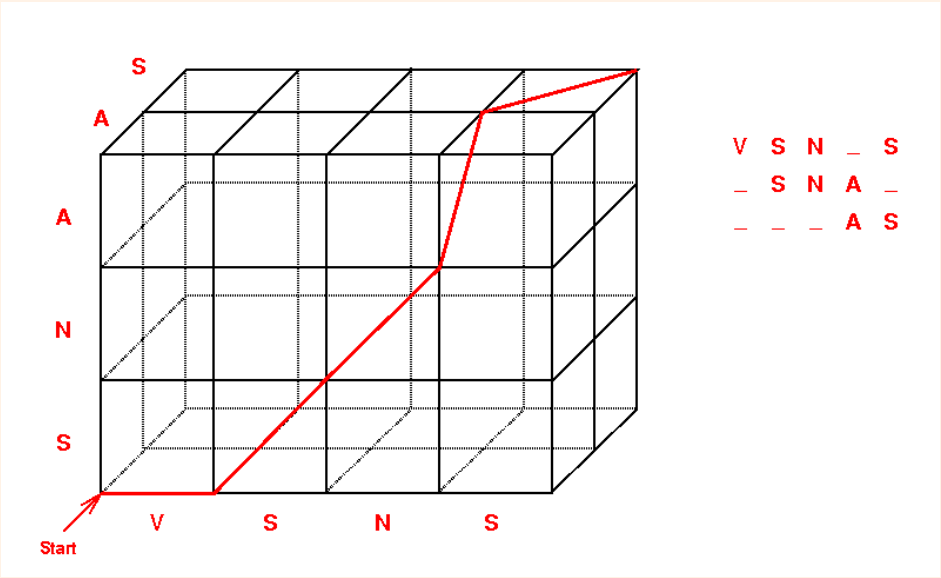
Multiple alignment

Definition

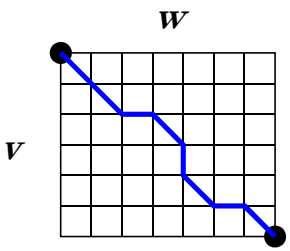
Sum-of-pairs score of a Multiple Sequence Alignment A:

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

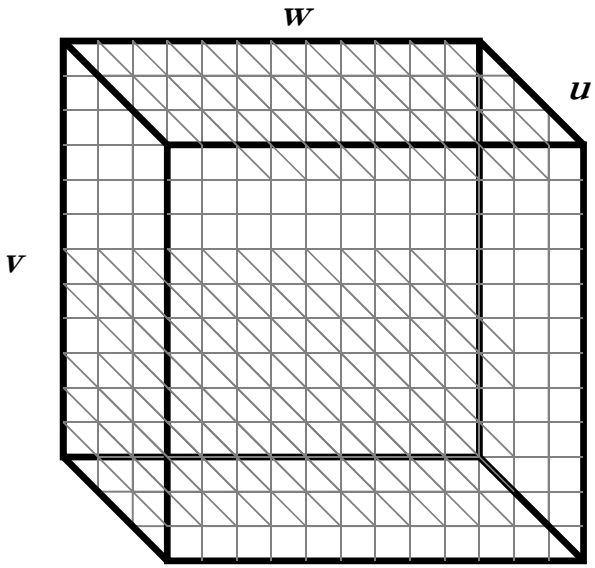
Dynamic Programming?



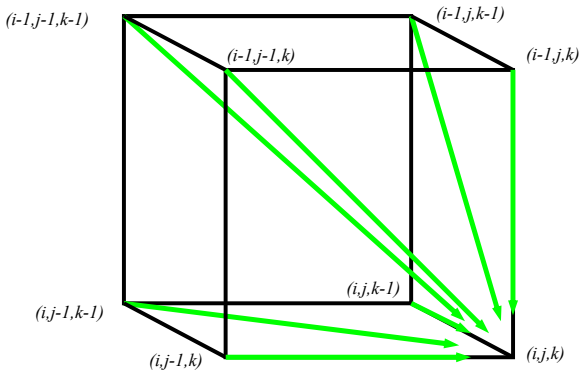
Multiple alignment



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.

Multiple alignment

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	

Multiple alignment

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

Multiple alignment

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

Indexing

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

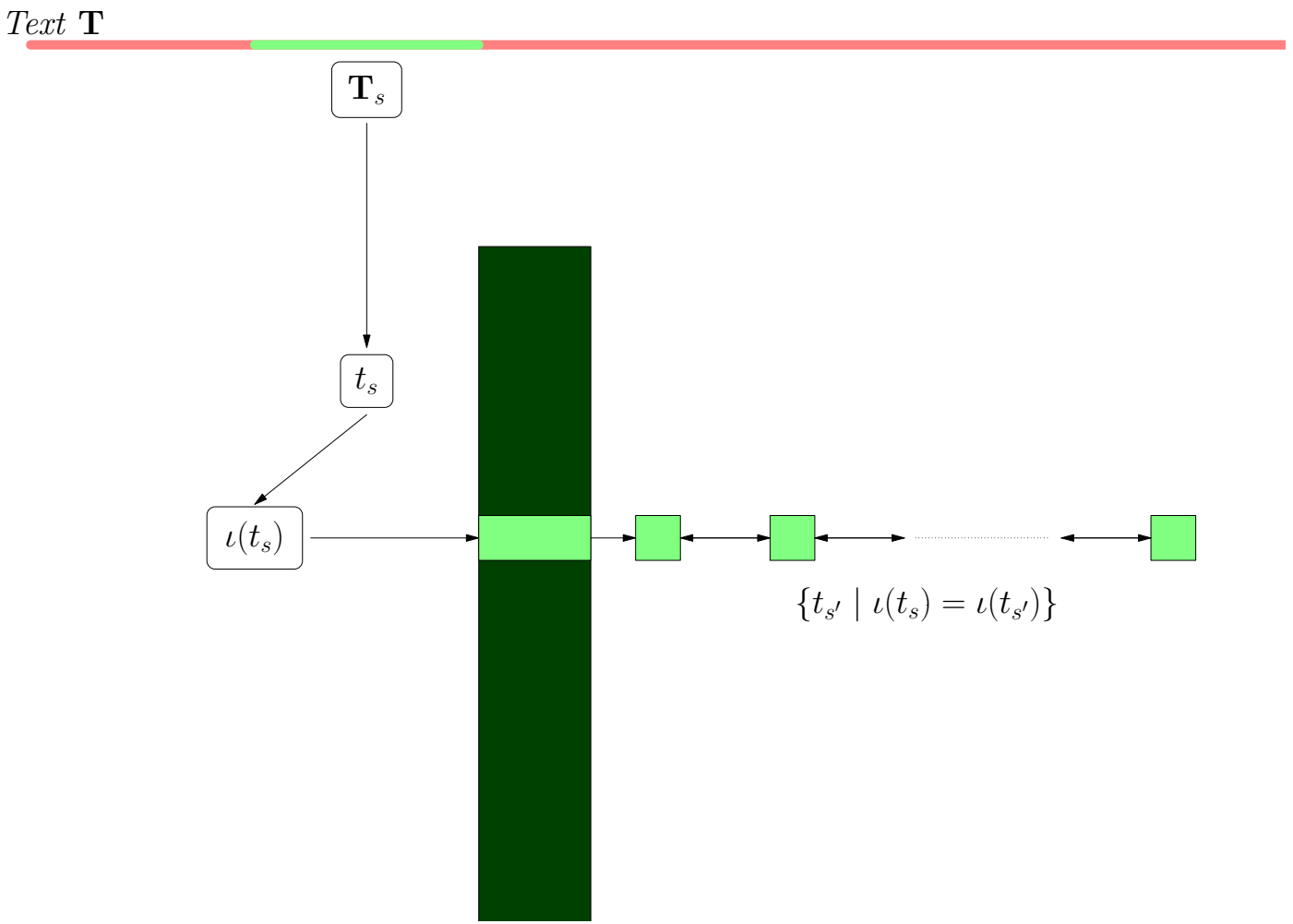
Careful!

4^m is BIG

Use an *index*

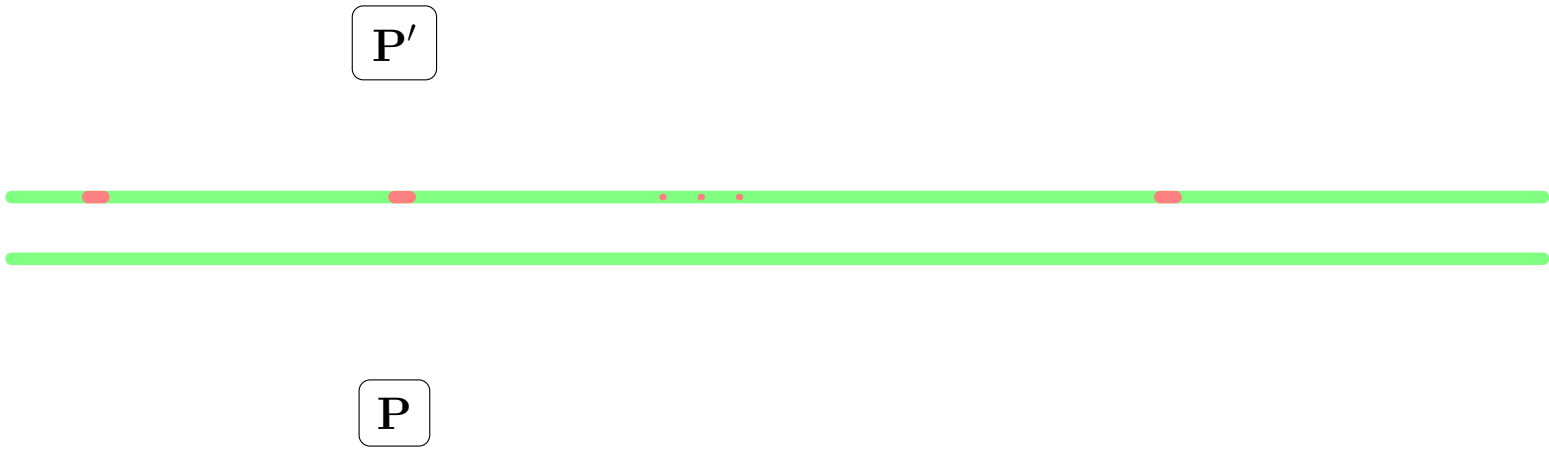
A function ι mapping a—*large*—domain in a—*small*—array

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

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.

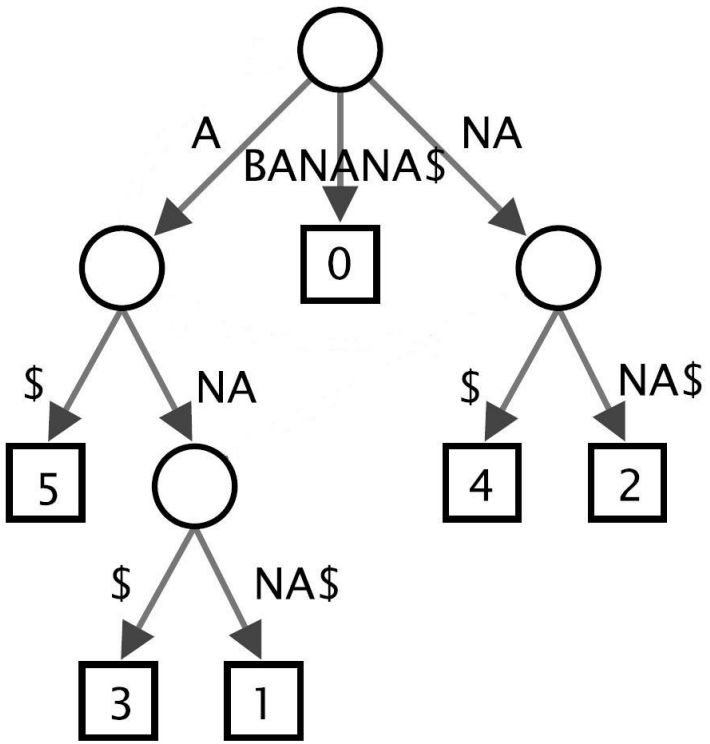
Il testo può essere memorizzato in una *struttura dati*

$\Rightarrow$ non sono obbligato a leggere tutto il testo per ogni pattern
$\Rightarrow$ posso cercare P ad un costo $O(|P|)$!

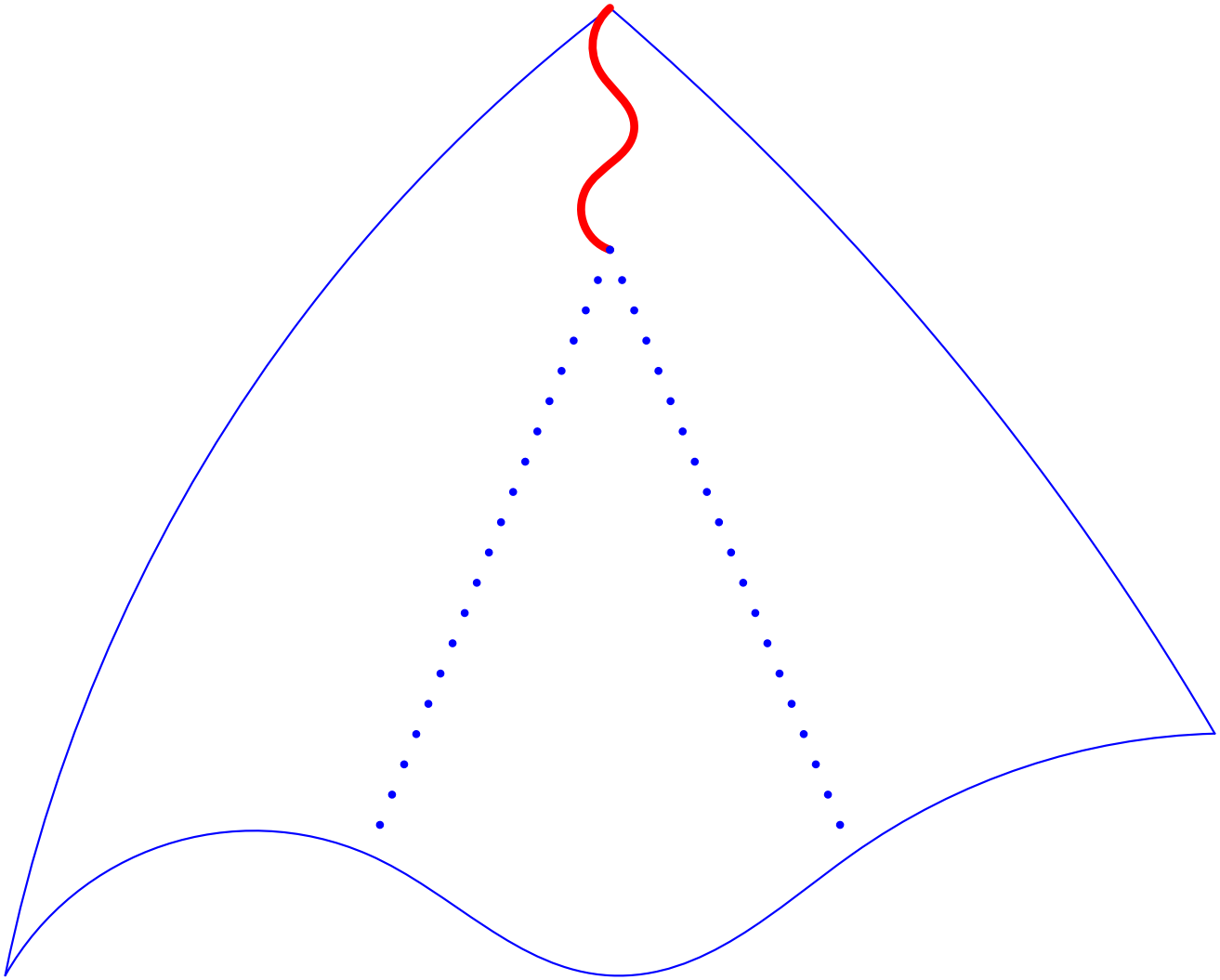
Suffix Trees

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	$

$$T = acaaacatat \qquad P = ata$$

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
- Ordino lessicograficamente la parte azzurra
- Ottengo una permutazione della stringa di partenza

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

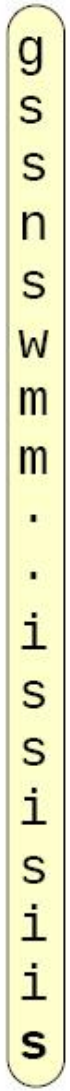
- Considero i primi caratteri
- Ordino lessicograficamente la parte azzurra
- Ottengo una permutazione della stringa di partenza

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**

La trasformata di Burrows-Wheeler

Passi per il calcolo

- Considero i primi caratteri
- Ordino lessicograficamente la parte azzurra
- 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 $k \geq 0$.