

# Introduction à la bioinformatique

3. l'analyse des séquences par paires et la programmation dynamique

1

## Objectifs

- Comprendre les notions d'identité, de similarité et d'homologie
- Être capable d'expliquer la notion d'un alignement optimal
- Expliquer les matrices de substitutions
- Comprendre la différence entre l'alignement global et local
- Expliquer les principes de la programmation dynamique
- Expliquer les solutions pour SSL et DE
- Comprendre la relation entre DE et l'alignement des séquences
- Expliquer les différences entre la pénalité linéaire et affine
- Comprendre comment on peut trouver des alignements sous-optimaux
- Être capable de développer tous les algorithmes

3

© Tom Lenaerts, 2012

## Bibliographie

- Zvelebil et Baum, Understanding bioinformatics
- Cormen et al, Introduction to Algorithms
- Gusfield, Algorithms on strings, trees and sequences
- MIT open course ware



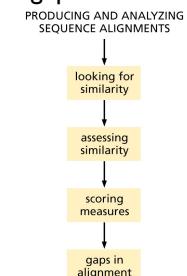
2

© Tom Lenaerts, 2012

## Pourquoi l'alignement ?

Comme expliqué au début, étant donné l'explosion de l'information génétique, comment peut-on remplacer la connaissance des séquences par la connaissance biologique?

WYFGKLGRKDAERQLLSFGNPRGTFLIREQTTKGAYSLISRWDMMGDKHVHYKIRKL  
DNGYYITTRAQFETLQQLVQHYSERAAGLCRLVPC  
  
WFFGKIPRAKAEEMLSKQRHDGAFLIRESESAPGDFSLSVKFGNDVQHFKVLRDGAGKYF  
LWVKFNLSNELVDYHRSTSVSRNQIFLDRIE  
  
WYYGVTRHQAEMLNERCHEGDFLIRDSESSPNDFSVSLKAQQKNNKHFKVQLKETVYCI  
GQRKFSTMEELVEHYKKAPIFTSEQGEKLYLVKHL



Pour détecter si ces séquences d'acides aminés sont des homologues, il faut les comparer entre elles.

4

# La difficulté

Le problème est que les gènes ou protéines (= séquences) ont changé pendant l'évolution

substitution

THISSEQUENCE  
THATSEQUENCE

Insertion et/ou suppression

THISISASEQUENCE  
THATSEQUENCE

*indels*

Ces modifications pourraient cacher une similarité fondamentale

Solution : chercher l'alignement qui produit la similarité maximale

5

# Quelques définitions

## l'identité

Le niveau d'identité est la fraction des résidus qui sont identiques dans les deux séquences

l'identité entre des séquences aléatoires n'est jamais 0 % !!!

## Similarité

Deux résidus sont similaires si leur substitution n'a aucun effet sur la fonctionnalité

Le niveau de similarité est la fraction des résidus qui sont similaires

THISISASEQUENCE  
|||||  
THAT---SEQUENCE

## Homologie

La similarité entre deux séquences est dérivée d'un ancêtre commun

7

# Pourquoi la similarité maximale?

Évolution divergente

Si on a deux gènes ou protéines qui viennent du même ancêtre commun, on veut que les nucléotides ou acides aminés qui font partie de l'ancêtre commun soient bien alignés

THISSEQUENCE  
|||||  
THATSEQUENCE

THISISASEQUENCE  
|||||  
THATSEQUENCE---

THISISA-SEQUENCE  
|||||  
TH----ATSEQUENCE

La meilleure façon pour obtenir ce résultat est de chercher la similarité maximale entre les séquences

6

# Attention !!!

Homologie implique un ancêtre commun,

en même temps, ça pourrait aussi suggérer une structure ou fonction similaire

Mais ça n'est pas toujours le cas:

Mutation pourrait produire des séquences similaires mais avec des fonctions différentes

On a besoin d'information additionnelle pour confirmer l'homologie

ou  
Il y a des protéines avec les mêmes fonctions mais qui ont des séquences différentes

ou  
deux séquences pourraient être similaires mais n'ont pas un ancêtre commun



Évolution convergente

8

# Attention !!!

Comme il est difficile de connaître l'ancêtre commun, un alignement n'est qu'une hypothèse

On a besoin d'une méthode quantitative qui prenne en compte l'aspect évolutif pour vérifier la qualité d'un alignement

Homologie est une relation binaire = deux séquences sont des homologues ou ne le sont pas

Il est plus facile de détecter l'homologie entre deux séquences d'acides aminés qu'entre deux séquences de nucléotides

1. il y a une plus grande probabilité de trouver un alignement entre deux nucléotides aléatoirement qu'entre deux acides aminés

2. La code génétique est冗余的

3. La fonction d'un gène est définie par la structure de la protéine

9

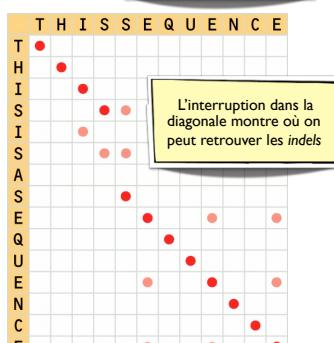
## Similarité basée sur l'identité

Le **dot plot** est une méthode qui montre la similarité entre deux séquences graphiquement

les points **rouges** sont des résidus qui s'assortissent

les points **roses** sur les deux cotés de la diagonale centrale montrent où on peut trouver les mêmes résidus dans d'autres positions de la séquence.

un dot est ajouté quand les deux positions s'assortissent



Chaque ligne diagonale montre une sous-séquence alignée

11

# Un score quantitatif

L'idée est que la méthode quantitative donne un score plus élevé aux alignements de séquences qui sont dérivées d'un ancêtre commun par rapport aux séquences qui sont choisies aléatoirement

L'alignement exact obtient le meilleur score  
=  
l'alignement optimal

Pour l'instant on n'a pas trouvé une méthode quantitative qui utilise un modèle exact de l'évolution

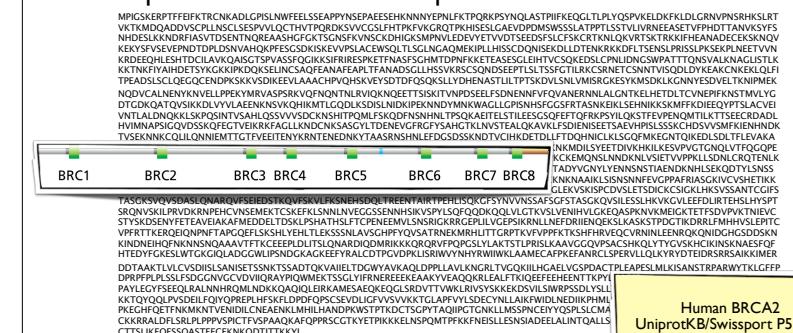
On n'a pas la garantie que le meilleur score donne l'alignement exact

On utilise les méthodes pour quantifier des alignements qui déterminent la similarité (MAX) et la différence (MIN) entre les séquences.

10

## Similarité basée sur l'identité 2

Les dot plots sont utilisés pour identifier des répétitions au sein de la séquence



12

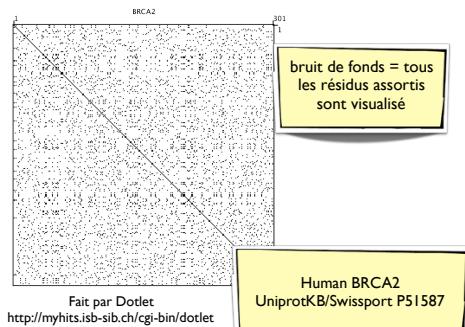
# Similarité basée sur l'identité 3

Les dot plots sont utilisés pour identifier des répétitions au sein de la séquence

Les dot plots ont des problèmes avec le **bruit de fond**

Seulement besoin de trouver les configurations significatives

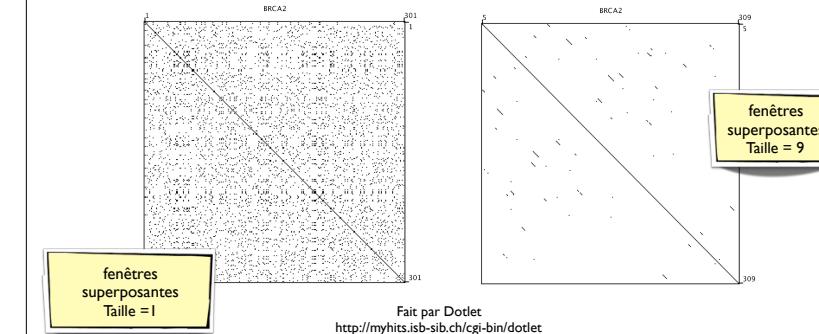
**Introduction d'un filtre:**  
Une fenêtre superposante (avec une certaine taille) dans laquelle il faut avoir un nombre minimal d'associations identiques



13

# Similarité basée sur l'identité 3

Seulement important de trouver les configurations significatives



14

# Identité et similarité

- Dans l'analyse précédente, on utilise **un score d'identité** :
  - On donne 1 quand les deux parties s'assortissent et un score de 0 autrement
- Quelques soucis:
  - Chaque acide aminé pourrait être remplacé par un autre avec les mêmes caractéristiques biochimiques
    - Dans ce cas, l'évolution pourrait avoir accepté le remplacement
  - Certains acides aminés sont importants pour la stabilité de la protéine
    - Évolution n'acceptera jamais que ces acides aminés soient remplacés
- Un score de similarité** tient compte de ces soucis
- Le score total pour une séquence est obtenu en additionnant les scores par paires

Au moins 30% de similarité total est nécessaire

15

# Matrices de substitution

Donnent des scores de similarité entre toutes les paires d'acides aminés

Chaque score est le taux entre la probabilité qu'une paire d'acides aminés s'assortissent par homologie et la probabilité qu'ils s'assortissent aléatoirement

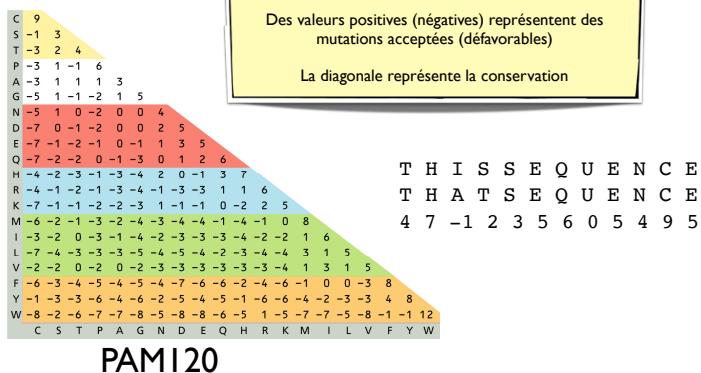
Les meilleures matrices étaient créées en utilisant des groupes de séquences homologues obtenues à partir d'espèces différentes

Deux exemples:  
**PAM** (M. Dayhoff et al (1978) A model of evolutionary change in proteins. In: *Atlas of protein sequence and structure Vol 5, No suppl 3, p.345-351*)  
**BLOSUM** (S. Henikoff et al (1992) Amino acid substitution matrices from protein blocks. *Proc Natl Acad Sci USA* 89:10915-10919)

16

## Matrices de substitution 2

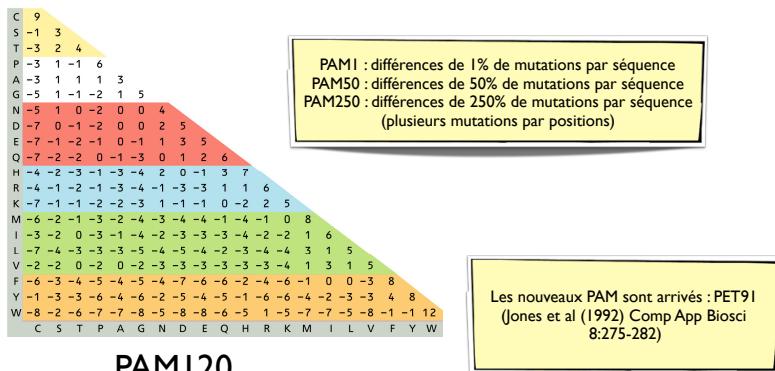
Les matrices PAM, créée par Margaret Dayhoff, donnent un score de substitution entre les acides aminés



17

## Matrices de substitution 4

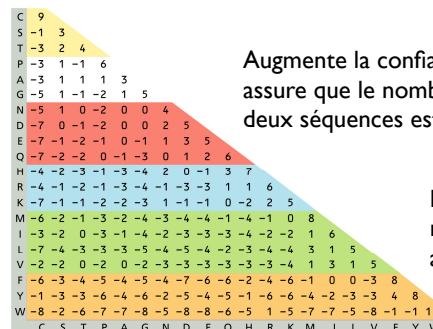
Il y a plusieurs de ces matrices, chacune reflétant une différence dans le nombre de substitutions acceptées



19

## Matrices de substitution 3

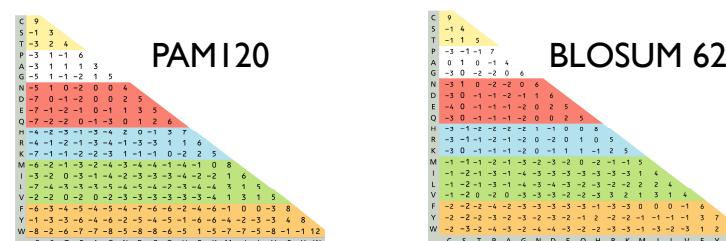
La matrice était créé en utilisant des vraies séquences similaires (~85%)



18

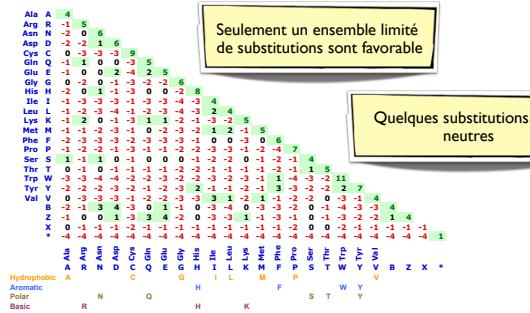
## Matrices de substitution 5

BLOSUM est déterminée en utilisant des blocs d'acides aminés bien conservés dans des protéines



Le 62 est une indication de l'identité entre les séquences qui étaient utilisées pour la construction de la matrice

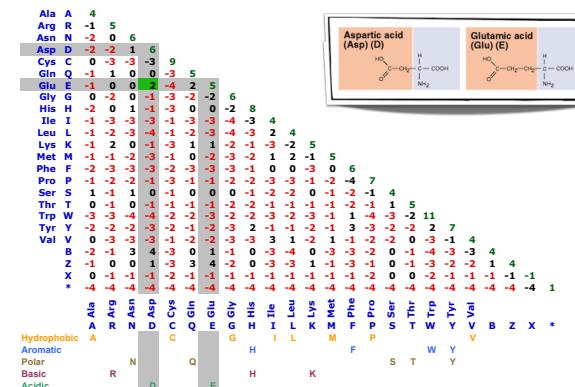
# Analyse de BLOSUM62



Les substitutions au sein des groupes d'acides aminés sont favorables

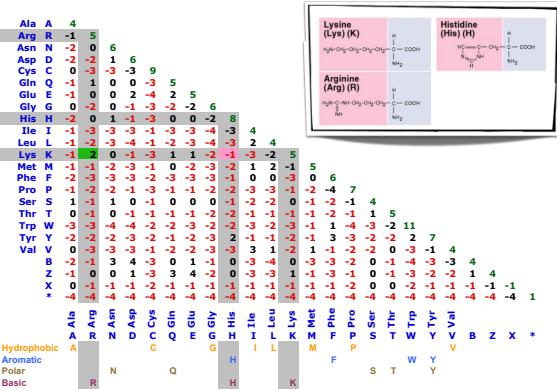
21

# Analyse de BLOSUM62



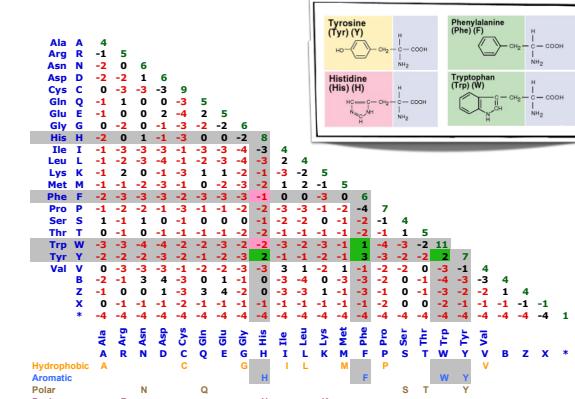
22

# Analyse de BLOSUM62



23

# Analyse de BLOSUM62



24

# Matrices de substitution 6

Quelles matrices est-ce qu'on doit utiliser ????

Le numéro de PAM indique la divergence entre les séquences et le numéro de BLOSUM indique le pourcentage d'identité

Pour l'évaluation des séquences similaires on a besoin d'une matrice qui donne des valeurs hautes pour les assortiments identiques : BLOSUM 80 ou PAM120

La matrice est choisie en utilisant le taux de conservation prévue entre les séquences qu'on veut aligner

25

## Un exemple

WYFGKLRKDAERQLLSFGNPRGTFLIREQTTKGAYSLSIRDWDDMKGDHVVKHYKIRKL  
DNGGYITTRAQFETLQQLVQHYSERAAGLCRLVPC

WYYGVTRHQAEMLNERGHEGDFLIRDSESSPNDFSVSLKAQGKNHKFKVQLKETVYCI  
GQRKFSTMELVEHYKKAPIFTSEQGEKLYLVKHL

```
ALIGN calculates a global alignment of two sequences
version 2.2u
Please cite: Myers and Miller, CABIOS (1989) 4:11-17
unknow WYFGKLRKDAERQLLSFGNPRGTFLIREQTTKGAYSLSIRDWDDMKGDHVVKHYKIRKL
          98 bp                               98 aa vs.
unknow 95 bp                                95 aa
using matrix file: /usr/molbio/share/fasta3/blosum62.mat, gap open/
ext: -10/-1
30.1% identity in 103 aa overlap;           Global score: 103
```

10	20	30	40	50	60
unknow WYFGKLRKDAERQLLSFGNPRGTFLIREQTTKGAYSLSIRDWDDMKGDHVVKHYKIRKL	.....	.....	.....	.....	.....
unknow WYYGVTRHQAEMLNERGHEGDFLIRDSESSPNDFSVSLKAQGKNHKFKVQLKETVYCI	.....	.....	.....	.....	.....
10	20	30	40	50	

70	80	90			
unknow DNGGYITTRAQFETLQQLVQHYSERAAGLCRLVPC	.....	.....			
.....	.....	.....	.....	.....	
unknow ET--YCIGQRKFSTMELVEHYKKAPIFTSEQGEKLYLVKHL	.....	.....	.....	.....	
60	70	80	90		

LALIGN (BLOSUM 62, début de l'espace = -10, l'extension de l'espace = -1)  
[http://www.ch.embnet.org/software/  
LALIGN\\_form.html](http://www.ch.embnet.org/software/LALIGN_form.html)

# Les espaces (gaps)

Des séquences homologues ont souvent une longueur différente

THIS IS A SEQUENCE  
THAT SEQUENCE

THIS IS A-SEQUENCE  
|| | | | | | |  
TH---AT SEQUENCE

Si on veut aligner ces séquences on devrait introduire des espaces pour les insertions et les suppressions

On fait la différence entre le début d'un espace (pénalité de -10 au -15) ou l'extension d'un espace existant (penalité de -0.5 au -2)

26

## Un exemple 2

WYFGKLRKDAERQLLSFGNPRGTFLIREQTTKGAYSLSIRDWDDMKGDHVVKHYKIRKL  
DNGGYITTRAQFETLQQLVQHYSERAAGLCRLVPC

WYYGVTRHQAEMLNERGHEGDFLIRDSESSPNDFSVSLKAQGKNHKFKVQLKETVYCI  
GQRKFSTMELVEHYKKAPIFTSEQGEKLYLVKHL

```
ALIGN calculates a global alignment of two sequences
version 2.2u
Please cite: Myers and Miller, CABIOS (1989) 4:11-17
SH2A 9 WYFGKLRKDAERQLLSFGNPRGTFLIREQTTKGAYSLSIRDWDDMKGDHVVKHYKIRKL
          98 bp                               98 aa vs.
SH2B 95 bp                                95 aa
using matrix file: /usr/molbio/share/fasta3/blosum80.mat, gap open/ext: -10/-1
34.3% identity in 108 aa overlap;           Global score: 195
```

10	20	30	40	50
SH2A 9 WYFGKLRKDAERQLLSFGNPRGTFLIREQTTKGAYSLSIRDWDDMKGDHVVKHYKIRKL	.....	.....	.....	.....
SH2B 95 bp	.....	.....	.....	.....
10	20	30	40	50

60	70	80	90		
SH2A 9 RKLDRNGGGYITTRAQFETLQQLVQHYSERAAGLCRLVPC	.....	.....	.....	.....	
SH2B 95 bp	.....	.....	.....	.....	
10	20	30	40	50	

LALIGN (BLOSUM 80, début de l'espace = -10, l'extension de l'espace = -1)  
[http://www.ch.embnet.org/software/  
LALIGN\\_form.html](http://www.ch.embnet.org/software/LALIGN_form.html)

27

28

## Un exemple 3

```
WYFGKLGRKDAERQLLSFGNPRGTFLIRESQTTKGAYSLSIRDWDDMKGDHVHKHYKIRKL
DNGGYYTTRAQFETLQLQLVQHYSERAAGLCRLVVPC

WYYGKVTRHQAEMLNERGHEGDFLIRDSESSPNDFSVSLKAQGKNKHFKVQLKETVYCI
GQRKFSTMEELVEHYKKAPIFTSEQGEKLYLVKHL
```

ALIGN calculates a global alignment of two sequences  
version 2.2u  
Please cite: Myers and Miller, CABIOS (1989) 4:11-17  
SH2A 98 bp 98 aa vs.  
SH2B 95 bp 95 aa  
using matrix file: /usr/molbio/share/fasta3/pam120.mat, gap open/ext: -10/-1  
34.3% identity in 105 aa overlap; Global score: 114

10	20	30	40	50	60
SH2A 9	WYFGKLGRKDAERQLLSFGNPRGTFLIRESQTTKGAYSLSIRDWDDMKGDHVHKHYKIRKL				
:	:	:	:	:	:
SH2B	WYYGKVTRHQAEMLNERGHEGDFLIRDSESSPNDFSVSLKA---QGKN-KHEFKVQ-L				
10	20	30	40	50	

70	80	90			
SH2A 9	DNGGYYTTRAQFETLQLQLVQH-----SERAAGLCRLVVPC				
:	:	:	:	:	:
SH2B	KETVYCCIGQR-KFSTMEELVEHYKKAPIFTSEQGEKLY--LVKHL				
60	70	80	90		

LALIGN (**PAM120**, début de l'espace =-10,  
l'extension de l'espace = -1)  
[http://www.ch.embnet.org/software/  
LALIGN\\_form.html](http://www.ch.embnet.org/software/LALIGN_form.html)

29

## La programmation dynamique

- La méthode la plus efficace pour trouver un alignement optimal est la **programmation dynamique** (PD)
- Comme le *divide-and-conquer*, PD est une méthode pour la conception des algorithmes
- Les transparants suivantes donnent une explication de PD en utilisant quelques exemples
  - La sous-séquence la plus longue (*Longest common subsequence*) ou **SSL**
  - La distance d'édition (*Edit distance*) ou **DE**
- Après PD est utilisé pour la construction d'un algorithme d'alignement.

31

## Les différents types d'alignement

Il y a deux grandes classes d'alignement

**l'alignement global** : l'alignement entre des séquences complètes

Intéressant pour la comparaison entre des séquences apparentées

**l'alignement local** : l'alignement des régions conservées au sein des protéines

pour la détection des domaines ou des motifs communs

30

## Exemple I : SSL

Le problème : Etant données deux séquences  $x[1..m]$  et  $y[1..n]$ , trouvez **une** des sous-séquences les plus longues qui fait partie des deux séquences

il est possible que plusieurs solutions existent

32-1

# Exemple I : SSL

Le problème : Etant données deux séquences  $x[1..m]$  et  $y[1..n]$ , trouvez une des sous-séquences les plus longues qui fait partie des deux séquences

il est possible que plusieurs solutions existent

$x : A \ B \ C \ B \ D \ A \ B$

$y : B \ D \ C \ A \ B \ A$

Quelles sont les plus longues sous-séquences qu'on peut trouver au sein de  $x$  et  $y$  ?

32-2

# Exemple I : SSL

Le problème : Etant données deux séquences  $x[1..m]$  et  $y[1..n]$ , trouvez une des sous-séquences les plus longues qui fait partie des deux séquences

il est possible que plusieurs solutions existent

$x : A \ B \ C \ B \ D \ A \ B$

$y : B \ D \ C \ A \ B \ A$

BCAB,  
BCBA,  
BDAB

=  $L(x,y)$   
L( $x,y$ )= une relation entre  $x$  et  $y$

Comment résoudre ce problème?

32-4

# Exemple I : SSL

Le problème : Etant données deux séquences  $x[1..m]$  et  $y[1..n]$ , trouvez une des sous-séquences les plus longues qui fait partie des deux séquences

il est possible que plusieurs solutions existent

$x : A \ B \ C \ B \ D \ A \ B$

$y : B \ D \ C \ A \ B \ A$

BCAB,  
BCBA,  
BDAB

=  $L(x,y)$   
L( $x,y$ )= une relation entre  $x$  et  $y$

Quelles sont les plus longues sous-séquences qu'on peut trouver au sein de  $x$  et  $y$  ?

32-3

# Exemple I : SSL 2

Une approche est de vérifier toutes les sous-séquences de  $x$  pour voir s'ils font aussi partie de  $y$  = méthode approfondie

Temps d'exécution?

# Exemple I : SSL 2

Une approche est de vérifier toutes les sous-séquences de  $x$  pour voir s'ils font aussi partie de  $y$  = [méthode approfondie](#)

Temps d'exécution?

- I. Combien de temps est exigé pour vérifier qu'une sous-séquence de  $x$  fait aussi partie d'une séquence  $y$ ?

Obligé d'aller jusqu'au bout de la séquence  $y$  :  $O(n)$  par sous-séquence

33-2

# Exemple I : SSL 2

Une approche est de vérifier toutes les sous-séquences de  $x$  pour voir s'ils font aussi partie de  $y$  = [méthode approfondie](#)

le total de 1 et 2 fait  $O(n2^n) =$   
exponentielle = [lent](#)

Temps d'exécution?

- I. Combien de temps est exigé pour vérifier qu'une sous-séquence de  $x$  fait aussi partie d'une séquence  $y$ ?

Obligé d'aller jusqu'au bout de la séquence  $y$  :  $O(n)$  par sous-séquence

2. Combien de sous-séquences différentes pouvez-vous identifier au sein de la séquence  $x$  ?

Chaque sous-séquence est représentée par un vecteur de bits (0/1). Il y a  $2^m$  vecteurs

33-4

# Exemple I : SSL 2

Une approche est de vérifier toutes les sous-séquences de  $x$  pour voir s'ils font aussi partie de  $y$  = [méthode approfondie](#)

Temps d'exécution?

- I. Combien de temps est exigé pour vérifier qu'une sous-séquence de  $x$  fait aussi partie d'une séquence  $y$ ?

Obligé d'aller jusqu'au bout de la séquence  $y$  :  $O(n)$  par sous-séquence

2. Combien de sous-séquences différentes pouvez-vous identifier au sein de la séquence  $x$  ?

Chaque sous-séquence est représentée par un vecteur de bits (0/1). Il y a  $2^m$  vecteurs

x : A	B	C	B	D	A	B
v : 0	1	1	0	1	0	0
v : 1	0	1	0	0	0	0

33-3

# Exemple I : SSL 3

Pour trouver une solution moins lente, on introduit une simplification

34-1

# Exemple I : SSL 3

Pour trouver une solution moins lente, on introduit une simplification

1. Au lieu de chercher le SSL, essayez de trouver d'abord la **longueur**

$|s| = \text{la longueur de la séquence } s$

34-2

# Exemple I : SSL 3

Pour trouver une solution moins lente, on introduit une simplification

1. Au lieu de chercher le SSL, essayez de trouver d'abord la **longueur**

$|s| = \text{la longueur de la séquence } s$

2. Après, étendez l'algorithme pour trouver le SSL

Approche : Utilisez les **préfixes** des séquences  $x$  et  $y$

$$c[i,j] = |LCS(x[1..i], y[1..j])|$$

le préfixe  $i$  d'une séquence  $x$   
est  $x[1..i]$

$$c[m,n] = |LCS(x[1..m], y[1..n])|$$

$c[i,j] = \text{la longueur}$

34-4

# Exemple I : SSL 3

Pour trouver une solution moins lente, on introduit une simplification

1. Au lieu de chercher le SSL, essayez de trouver d'abord la **longueur**

$|s| = \text{la longueur de la séquence } s$

2. Après, étendez l'algorithme pour trouver le SSL

34-3

# Exemple I : SSL 4

L'idée est de résoudre le problème pour les séquences  $x[1..m]$  et  $y[1..n]$  (longueur  $m$  et  $n$ ) en utilisant les préfixes de longueur  $m-1$  en  $n-1$

35-1

## Exemple I : SSL 4

L'idée est de résoudre le problème pour les séquences  $x[1..m]$  et  $y[1..n]$  (longueur  $m$  et  $n$ ) en utilisant les préfixes de longueur  $m-1$  en  $n-1$

**Caractéristique 1 de PD  
La sous-structure optimale**

on dit qu'un problème a une sous-structure optimale si la solution optimale pour le problème contient des solutions optimales pour les sous-problèmes du problème global

35-2

## Exemple I : SSL 5

L'idée est de résoudre le problème pour les séquences  $x[1..m]$  et  $y[1..n]$  (longueur  $m$  et  $n$ ) en utilisant les préfixes de longueur  $m-1$  en  $n-1$

$$c[m,n]=k \text{ et } z[1..k]=\text{LCS}(x[1..m],y[1..n])$$

36-1

## Exemple I : SSL 4

L'idée est de résoudre le problème pour les séquences  $x[1..m]$  et  $y[1..n]$  (longueur  $m$  et  $n$ ) en utilisant les préfixes de longueur  $m-1$  en  $n-1$

**Caractéristique 1 de PD  
La sous-structure optimale**

on dit qu'un problème a une sous-structure optimale si la solution optimale pour le problème contient des solutions optimales pour les sous-problèmes du problème global

**Le problème SSL a des sous-structures optimales**

Si  $z[1..k]=\text{LCS}(x[1..m],y[1..n])$ , chaque préfixe de  $z$  est un SSL d'un préfixe de  $x$  et un préfixe de  $y$

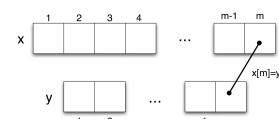
35-3

## Exemple I : SSL 5

L'idée est de résoudre le problème pour les séquences  $x[1..m]$  et  $y[1..n]$  (longueur  $m$  et  $n$ ) en utilisant les préfixes de longueur  $m-1$  en  $n-1$

$$c[m,n]=k \text{ et } z[1..k]=\text{LCS}(x[1..m],y[1..n])$$

I. Quand  $x[m]=y[n]$ ,  $z[k]=x[m]=y[n]$  et  $z[1..k-1]$  est un SSL de  $x[1..m-1]$  et  $y[1..n-1]$



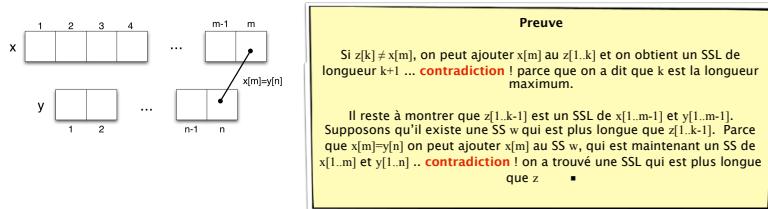
36-2

## Exemple I : SSL 5

L'idée est de résoudre le problème pour les séquences  $x[1..m]$  et  $y[1..n]$  (longueur  $m$  et  $n$ ) en utilisant les préfixes de longueur  $m-1$  en  $n-1$

$$c[m,n]=k \text{ et } z[1..k]=\text{LCS}(x[1..m],y[1..n])$$

1. Quand  $x[m]=y[n]$ ,  $z[k]=x[m]=y[n]$  et  $z[1..k-1]$  est un SSL de  $x[1..m-1]$  et  $y[1..n-1]$



36-3

## Exemple I : SSL 6

L'idée est de résoudre le problème pour les séquences  $x[1..m]$  et  $y[1..n]$  (longueur  $m$  et  $n$ ) en utilisant les préfixes de longueur  $m-1$  en  $n-1$

$$c[m,n]=k \text{ et } z[1..k]=\text{LCS}(x[1..m],y[1..n])$$

2. Quand  $x[m] \neq y[n]$ ,  $z[k] \neq x[m]$  implique que  $z[1..k]$  est un SSL de  $x[1..m-1]$  et  $y[1..n]$

si  $x[m]$  n'est pas le dernier élément de la séquence  $z[1..k]$ , on doit vérifier si  $x[m-1]=y[n]$

37-2

## Exemple I : SSL 6

L'idée est de résoudre le problème pour les séquences  $x[1..m]$  et  $y[1..n]$  (longueur  $m$  et  $n$ ) en utilisant les préfixes de longueur  $m-1$  en  $n-1$

$$c[m,n]=k \text{ et } z[1..k]=\text{LCS}(x[1..m],y[1..n])$$

37-1

## Exemple I : SSL 6

L'idée est de résoudre le problème pour les séquences  $x[1..m]$  et  $y[1..n]$  (longueur  $m$  et  $n$ ) en utilisant les préfixes de longueur  $m-1$  en  $n-1$

$$c[m,n]=k \text{ et } z[1..k]=\text{LCS}(x[1..m],y[1..n])$$

2. Quand  $x[m] \neq y[n]$ ,  $z[k] \neq x[m]$  implique que  $z[1..k]$  est un SSL de  $x[1..m-1]$  et  $y[1..n]$

si  $x[m]$  n'est pas le dernier élément de la séquence  $z[1..k]$ , on doit vérifier si  $x[m-1]=y[n]$

3. Quand  $x[m] \neq y[n]$ ,  $z[k] \neq y[n]$  implique que  $z[1..k]$  est un SSL de  $x[1..m]$  et  $y[1..n-1]$

si  $y[n]$  n'est pas le dernier élément de la séquence  $z[1..k]$ , on doit vérifier si  $x[m]=y[n-1]$

37-3

## Exemple I : SSL 7

Utilisant ces trois observations on peut définir un **algorithme récursif** pour résoudre le problème du SSL

$$c[i,j] = \begin{cases} 0 & \text{si } i=0 \text{ et } j=0 \\ c[i-1,j-1]+1 & \text{si } i,j>0 \text{ et } x[i]=y[j] \\ \max\{c[i-1,j], c[i,j-1]\} & \text{si } i,j>0 \text{ et } x[i]\neq y[j] \end{cases}$$

38-1

## Exemple I : SSL 7

Utilisant ces trois observations on peut définir un **algorithme récursif** pour résoudre le problème du SSL

$$c[i,j] = \begin{cases} 0 & \text{si } i=0 \text{ et } j=0 \\ c[i-1,j-1]+1 & \text{si } i,j>0 \text{ et } x[i]=y[j] \\ \max\{c[i-1,j], c[i,j-1]\} & \text{si } i,j>0 \text{ et } x[i]\neq y[j] \end{cases}$$

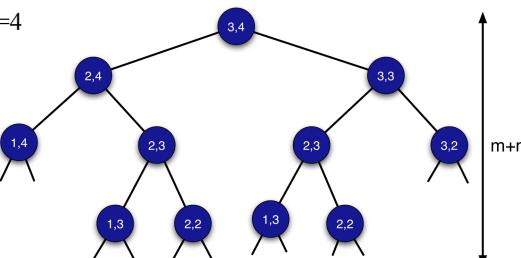
```
SSL(x,y,i,j)
  if (x[i] == y[j])
    then c[i,j]←SSL(x,y,i-1,j-1) + 1
    else c[i,j]←max {SSL(x,y,i-1,j), SSL(x,y,i,j-1)}
  return c[i,j]
```

38-2

## Exemple I : SSL 8

Temps d'exécution? Dans le pire des cas, l'algorithme doit toujours prendre le branchement “else” du code

m=3 et n=4



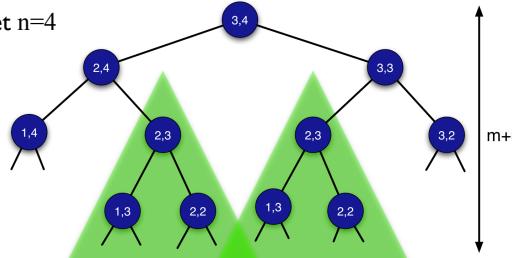
Nombre de noeuds est aussi exponentiel

39

## Exemple I : SSL 9

Temps d'exécution? Dans le pire des cas, l'algorithme doit toujours prendre le branchement “else” du code

m=3 et n=4



Mais le même sous-problème est résolu plusieurs fois !!!

40

## Exemple I : SSL 10

### Caractéristique 2 de PD Sous-problèmes superposants

Une solution récursive contient un nombre limité de sous-problèmes distincts répétés beaucoup de fois

41-1

## Exemple I : SSL 10

### Caractéristique 2 de PD Sous-problèmes superposants

Une solution récursive contient un nombre limité de sous-problèmes distincts répétés beaucoup de fois

Le nombre de sous-problèmes différents de SSL pour deux séquences de longueur  $m$  et  $n$  est  $mn$

Si un problème a les deux caractéristiques (sous-structure optimale et sous-problèmes superposants), le PD pourrait être utilisé pour la création d'un algorithme

41-2

## Exemple I : SSL 11

*Memoization* ( $\neq$ PD): Quand une solution pour un sous-problème est calculé, enregistrez-le dans un tableau

Les appels ultérieurs au SSL vont vérifier le tableau pour éviter qu'on refasse le même travail

42-1

## Exemple I : SSL 11

*Memoization* ( $\neq$ PD): Quand une solution pour un sous-problème est calculé, enregistrez-le dans un tableau

Les appels ultérieurs au SSL vont vérifier le tableau pour éviter qu'on refasse le même travail

```
SSL(x,y,i,j)
    if (c[i,j] == NIL) //Si le travail n'est pas fait, faites le
        if (x[i] == y[j])
            then c[i,j]←SSL(x,y,i-1,j-1) + 1
            else c[i,j]←max {SSL(x,y,i-1,j), SSL(x,y,i,j-1)}
    return c[i,j]
```

42-2

## Exemple I : SSL 11

*Memoization* ( $\neq PD$ ): Quand une solution pour un sous-problème est calculé, enregistrez-le dans un tableau

Les appels ultérieurs au SSL vont vérifier le tableau pour éviter qu'on refasse le même travail

```
SSL(x,y,i,j)
  if (c[i,j] == NIL) //si le travail n'est pas fait, faites le
    if (x[i] == y[j])
      then c[i,j]←SSL(x,y,i-1,j-1) + 1
      else c[i,j]←max {SSL(x,y,i-1,j), SSL(x,y,i,j-1)}
  return c[i,j]
```

Temps d'exécution? Nombre de calculs est  $mn$  :  $O(mn)$   
Mémoire?  $O(mn)$

42-3

## Exemple I : SSL 12

La solution précédente est construite utilisant une approche *top-down*. On pourrait aussi construire la solution en utilisant une approche *bottom-up*  $\Rightarrow PD$

	A	B	C	B	D	A	B
B	0						
D	0						
C	0						
A	0						
B	0						
A	0						

$$c[i,j] = \begin{cases} 0 & \text{si } i=0 \text{ et } j=0 \\ c[i-1,j-1]+1 & \text{si } i,j>0 \text{ et } x[i]=y[j] \\ \max\{c[i-1,j], c[i,j-1]\} & \text{si } i,j>0 \text{ et } x[i]\neq y[j] \end{cases}$$

43

## Exemple I : SSL 12

La solution précédente est construite utilisant une approche *top-down*. On pourrait aussi construire la solution en utilisant une approche *bottom-up*  $\Rightarrow PD$

	A	B	C	B	D	A	B
B	0	0	0	0	0	0	0
D	0	0	1	1	1	1	1
C	0						
A	0						
B	0						
A	0						

$$c[i,j] = \begin{cases} 0 & \text{si } i=0 \text{ et } j=0 \\ c[i-1,j-1]+1 & \text{si } i,j>0 \text{ et } x[i]=y[j] \\ \max\{c[i-1,j], c[i,j-1]\} & \text{si } i,j>0 \text{ et } x[i]\neq y[j] \end{cases}$$

44

## Exemple I : SSL 12

La solution précédente est construite utilisant une approche *top-down*. On pourrait aussi construire la solution en utilisant une approche *bottom-up*  $\Rightarrow PD$

	A	B	C	B	D	A	B
B	0						
D	0						
C	0						
A	0						
B	0						
A	0						

$$c[i,j] = \begin{cases} 0 & \text{si } i=0 \text{ et } j=0 \\ c[i-1,j-1]+1 & \text{si } i,j>0 \text{ et } x[i]=y[j] \\ \max\{c[i-1,j], c[i,j-1]\} & \text{si } i,j>0 \text{ et } x[i]\neq y[j] \end{cases}$$

Temps d'exécution?  $O(mn)$

## Exemple 1 : SSL 12

La solution précédente est construite utilisant une approche *top-down*. On pourrait aussi construire la solution en utilisant une approche *bottom-up*  $\Rightarrow$  PD

	A	B	C	B	D	A	B
B	0	0	0	0	0	0	0
D	0	0	1	1	1	1	1
C	0	0	1	2	2	2	2
A	0	1	1	2	2	2	3
B	0	1	2	2	3	3	4
A	0	1	2	2	3	3	4

Temps d'exécution? O( $mn$ )  
Mémoire? O( $mn$ )

Trouver le SSL en retournant sur trace vers le début

46

## Exemple 1 : SSL 12

La solution précédente est construite utilisant une approche *top-down*. On pourrait aussi construire la solution en utilisant une approche *bottom-up*  $\Rightarrow$  PD

	A	B	C	B	D	A	B
B	0	0	0	0	0	0	0
D	0	0	1	1	1	1	1
C	0	0	1	2	2	2	2
A	0	1	1	2	2	2	3
B	0	1	2	2	3	3	4
A	0	1	2	2	3	3	4

Utilise une matrice additionnelle pour enregistrer l'origine

	A	B	C	B	D	A	B
B	0	L	L	L	L	L	L
D	T	T	D	L	D	L	L
C	T	L	T	D	L	L	L
A	T	D	L	T	L	L	D
B	T	T	D	L	D	L	D
A	T	D	T	L	T	L	D

47

## Exemple 2 : DE

Le problème : Calculer la distance d'édition (DE) entre deux chaînes de caractères, avec une transcription d'édition optimale (*edit transcript*) qui décrit la transformation entre les deux *strings*

4 opérations de transformation : I (insertion), D (suppression), R (remplacement) et M (les deux caractères sont les mêmes)

RIMDMDMI

IRMDMDMMI

x[1..7]:vintner

v intner

vintner

y[1..7]:writers

wri t ers

wri t ers

Les opérations sont faites sur la première chaîne de caractères

## Exemple 1 : SSL 13

```
SSL(x,y,m,n)
  for i ← 1 to m
    do c[i,0] ← 0
      bt[i,0] ← "T"
    for j ← 1 to n
      do c[0,j] ← 0
        bt[i,0] ← "L"
  for i ← 1 to m
    do for j ← 1 to n
      do if(x[i]==y[j])
        then
          c[i,j] ← c[i-1,j-1]+1
          bt[i,j] ← "D"
        else
          c[i,j] ← max{c[i-1,j],c[i,j-1]}
          bt[i,0] ← (argmax{c[i-1,j],c[i,j-1]}==0?"T":"L")
  return c
```

48

49

## Exemple 2 : DE 2

Définition 1: une chaîne utilisant des caractères de l'alphabet {D,I,M,R} qui décrit la transformation d'une chaîne de caractères vers une autre est **une transcription d'édition** des deux chaînes de caractères

Levenshtein distance

Définition 2: La **distance d'édition** entre deux chaînes de caractères est définie comme le nombre minimum d'opérations d'ensemble {I,D,R} nécessaires pour transformer la première chaîne de caractères en la deuxième

La transcription liée à la distance d'édition est la transcription optimale

50

## Exemple 2 : DE 4

La relation récursive :

$$D(i,j) = \begin{cases} i & \text{si } i>0 \text{ et } j=0 \\ j & \text{si } i=0 \text{ et } j>0 \\ \min\{D(i-1,j)+1, D(i,j-1)+1, D(i-1,j-1)+t(i,j)\} & \text{si } i,j>0 \text{ et si } x[i]=y[j], \\ & t(i,j)=0 \text{ autrement } t(i,j)=1 \end{cases}$$

$D(i,0)=i$  parce que on a besoin de  $i$  suppressions pour changer  $i$  caractères de  $x$  en 0 caractères de  $y$

	w	r	i	t	e	r	s	
v	0	1	2	3	4	5	6	7
i	1							
n	2							
t	3							
n	4							
e	5							
r	6							
r	7							

Comme le problème de SSI, on peut utiliser une approche bottom-up pour calculer la distance d'édition

## Exemple 2 : DE 3

Est-ce que le problème de DE contient les caractéristiques typiques d'un problème qui pourrait être résolu par PD?

les deux chaînes sont  $x[1..m]$  et  $y[1..n]$

La sous-structure optimale?

$D(i,j)$  est la fonction de la distance d'édition entre les chaînes  $x[1..i]$  et  $y[1..j]$

Les sous-problèmes superposants?

Pour les chaînes  $x$  et  $y$  de longueur  $m$  et  $n$  respectivement, le nombre de sous problèmes est  $mn$

51

## Exemple 2 : DE 4

La relation récursive :

$$D(i,j) = \begin{cases} i & \text{si } i>0 \text{ et } j=0 \\ j & \text{si } i=0 \text{ et } j>0 \\ \min\{D(i-1,j)+1, D(i,j-1)+1, D(i-1,j-1)+t(i,j)\} & \text{si } i,j>0 \text{ et si } x[i]=y[j], \\ & t(i,j)=0 \text{ autrement } t(i,j)=1 \end{cases}$$

$D(i,0)=i$  parce que on a besoin de  $i$  suppressions pour changer  $i$  caractères de  $x$  en 0 caractères de  $y$

	w	r	i	t	e	r	s	
v	0	1	2	3	4	5	6	7
i	1	1	2	3	4	5	6	7
n	2							
t	3							
n	4							
e	5							
r	6							
r	7							

Comme le problème de SSI, on peut utiliser une approche bottom-up pour calculer la distance d'édition

52

53

## Exemple 2 : DE 4

## La relation récursive :

$$D(i,j) = \begin{cases} i & \text{si } i>0 \text{ et } j=0 \\ j & \text{si } i=0 \text{ et } j>0 \\ \min(D(i-1,j)+1, D(i,j-1)+1, D(i-1,j-1)+t(i,j)) & \text{autre cas} \end{cases}$$

w	r	i	t	e	r	s
0	1	2	3	4	5	6
1	1	2	3	4	5	6
2	2	2	2	3	4	5
3	3	3	3	3	4	5
4	4	4	4	3	4	5
5	5	5	5	4	4	5
6	6	6	6	5	4	5
7	7	6	7	6	5	4

si  $i > 0$  et  $i = 0$       D(i,0)=i parce que on a besoin de i suppressions pour changer i caractères de x en 0 caractères de y

$\left. \begin{array}{l} \text{si } i>0 \text{ et } j=0 \\ \text{si } i=0 \text{ et } j>0 \end{array} \right\} +1, D(i,j-1)+1, D(i-1,j-1)+t(i,j) \right\}$   
 $\quad \quad \quad \text{si } i,j>0 \text{ et si } x[i]=y[j],$   
 $\quad \quad \quad t(i,j)=0 \text{ autrement } t(i,j)=1$

La distance d'édition est 5 :

On a besoin d'un minimum de 5 caractères de l'ensemble {R,D,I} pour la transformation

## Exemple 2 : DE 4

## Le retour arrière

3 solutions

	w	r	i	t	e	r	s	
v	0	1	2	3	4	5	6	7
i	1	1	2	3	4	5	6	7
n	2	2	2	2	3	4	5	6
t	3	3	3	3	3	4	5	6
n	4	4	4	4	3	4	5	6
e	5	5	5	5	4	4	5	6
r	6	6	6	6	5	4	5	6
r	7	7	6	7	6	5	4	5

RIMDMMDMMI v intner wri t ers	RRRMDMMMI vintner writ ers
<b>IRMDMDMMI</b> vintner wri t ers	

# Objectifs

- Comprendre les notions d'identité, de similarité et d'homologie
  - Être capable d'expliquer la notion d'un alignement optimal
  - Expliquer les matrices de substitutions
  - Comprendre la différence entre l'alignement global et local
  - Expliquer les principes de la programmation dynamique
  - Expliquer les solutions pour SSL et DE
  - Comprendre la relation entre DE et l'alignement des séquences
  - Expliquer les différences entre la pénalité linéaire et affine
  - Comprendre comment on peut trouver des alignements sous-optimales
  - Être capable de développer tous les algorithmes

## L'alignement global

Le problème de la DE est équivalent au problème d'alignement

Deux caractères différents dans un alignement sont des *replacements* dans une transcription d'édition

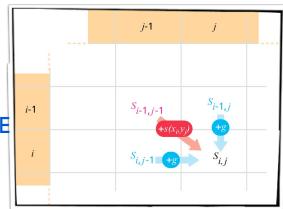
Un espace dans la première chaîne d'un alignement est une insertion dans la deuxième chaîne

Un espace dans la deuxième chaîne d'un alignement est une suppression du caractère qui apparaît dans la première chaîne

# L'alignement global 2

Le DP algorithme pour l'alignement était proposé par Saul Needleman et Christian Wunsch en 1970

L'algorithme de Needleman et Wunsch cherche l'alignement qui donne la plus grande similarité, ce qui est équivalent à chercher la transcription minimale en DE



$$S(i,j) = \begin{cases} ig & \text{si } i>0 \text{ et } j=0 \\ jg & \text{si } i=0 \text{ et } j>0 \\ \max\{S(i-1,j)+g, S(i,j-1)+g, S(i-1,j-1)+t(i,j)\} & \text{si } i,j>0, t(i,j)=MATSUB[x[i],y[j]] \end{cases}$$

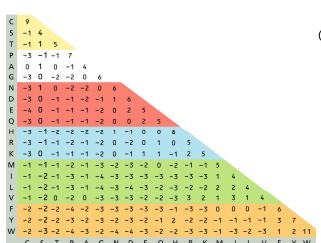
et  $g (<0)$  est la pénalité pour des espaces (gap penalty)

58

# L'alignement global 4

## L'alignement semiglobal

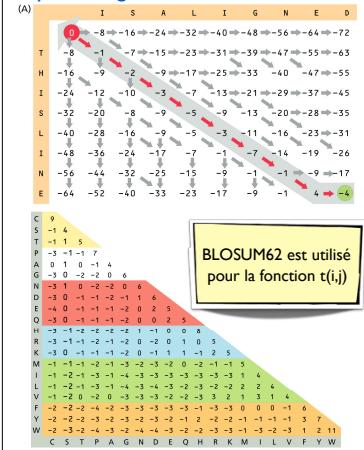
Maintenant les pénalités pour les positions  $S(i,0)$  en  $S(0,j)$  sont 0. On enlève l'obligation que les séquences doivent s'aligner au début et fin de la matrice.



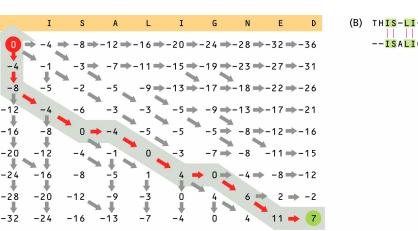
Le retour arrière commence à la position dans la dernière ligne avec la plus haute valeur

60

# L'alignement global 3

pénalité  $g$  est -8

(B) THIS LINE IS ALIGNED  
Quand la pénalité est trop haute, on n'obtient pas des espaces dans l'alignement

pénalité  $g$  est -4

59

# L'alignement global 5

Toutes les méthodes précédentes utilisent une pénalité linéaire pour les espaces dans l'alignement

$$g(n_{gap}) = -n_{gap} E$$

linear gap penalty

On pourrait aussi différencier entre la pénalité pour le début d'un espace (*gap open*) et l'augmentation d'un espace (*gap extend*)

$$g(n_{gap}) = -I - (n_{gap}-1) E$$

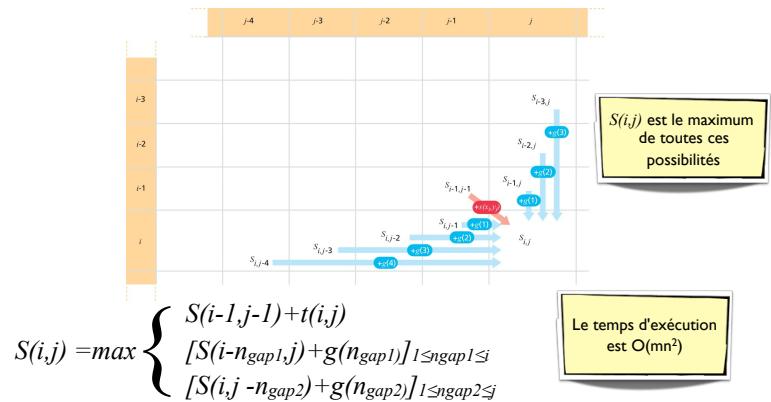
Affine gap penalty

Comment est-ce qu'on doit adapter l'algorithme?

61

# L'alignement global 6

Est-ce qu'on commence ou on étend l'espace?

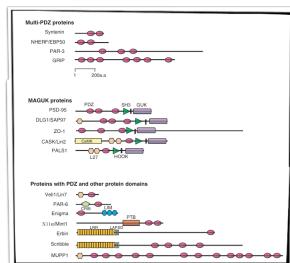


62

# L'alignement local

Parfois on veut seulement trouver des parties qui sont similaires entre des séquences

Smith et Waterman ont proposé une méthode qui ressemble fortement aux méthodes précédentes



Smith and Waterman (1981) Identification of common molecular subsequences J Mol Biol 147:195-197

64

# L'alignement global 7

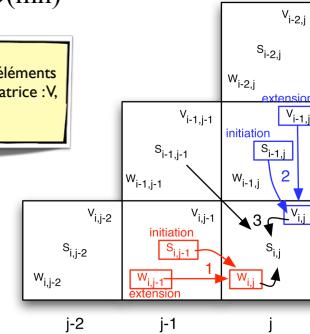
On peut adapter l'algorithme précédent de sorte à obtenir un temps d'exécution  $O(mn)$

$$V(i,j) = \max \begin{cases} S(i-1,j) - I \\ V(i-1,j) - E \end{cases}$$

$$W(i,j) = \max \begin{cases} S(i,j-1) - I \\ W(i,j-1) - E \end{cases}$$

$$S(i,j) = \max \begin{cases} S(i-1,j-1) + t(i,j) \\ V(i,j) \\ W(i,j) \end{cases}$$

On enregistre trois éléments par position de la matrice : V, W et S



63

# L'alignement local 2

La relation de récurrence de Smith et Waterman adaptée au pénalité affine

$$S(i,j) = \max \begin{cases} S(i-1,j-1) + t(i,j) \\ [S(i-n_{gap1},j) + g(n_{gap1})]_{1 \leq n_{gap1} \leq i} \\ [S(i,j-n_{gap2}) + g(n_{gap2})]_{1 \leq n_{gap2} \leq j} \\ 0 \end{cases}$$

Quand un alignement des caractères donne un score négatif, il est remplacé par 0  
et les valeurs de  $S(i,0)$  et  $S(0,j)$  sont aussi 0 comme dans l'alignement semiglobal

Le retour arrière commence à la position avec la plus haute valeur dans la matrice

65

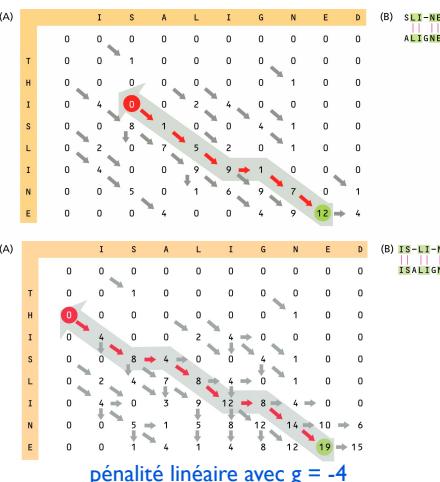
## L'alignement local 3

pénalité linéaire avec  $g = -8$

	C	S	T	P	A	G	N	R	K	M	I	L	V	F	Y	W				
C	9	-1	-4	-3	-1	-1	7	-1	1	-2	-2	0	6	-3	-1	-2	-2	0	6	
S	-1	4	-1	-1	0	-1	4	-1	-1	-1	-1	2	5	-3	-1	-1	-1	-2	0	6
T	-1	-1	0	-1	0	0	-1	0	-1	0	0	-1	0	-1	0	0	0	0	0	
P	-3	-1	1	-1	-1	7	-1	-1	-1	-1	-1	-1	2	-3	-1	-1	-1	-1	-1	0
A	0	1	0	-1	4	-1	1	-1	-1	-1	-1	-1	0	-3	-1	-1	-1	-1	-1	0
G	-3	0	-2	0	6	-1	-1	-1	-1	-1	-1	-1	5	-3	-1	-1	-1	-1	-1	0
N	-1	-1	-2	-2	0	-1	-1	-1	-1	-1	-1	-1	6	-3	-1	-1	-1	-1	-1	0
R	0	0	1	-1	-1	2	-1	0	-1	-1	-1	-1	5	-3	-1	-1	-1	-1	-1	0
M	-4	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	6	-3	-1	-1	-1	-1	-1	0
I	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	5	-3	-1	-1	-1	-1	-1	0
L	0	0	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	6	-3	-1	-1	-1	-1	-1	0
V	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	6	-3	-1	-1	-1	-1	-1	0
Y	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	6	-3	-1	-1	-1	-1	-1	0
W	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	6	-3	-1	-1	-1	-1	-1	0
C	9	-1	-4	-3	-1	-1	7	-1	1	-2	-2	0	6	-3	-1	-1	-1	-1	-1	0
S	-1	4	-1	-1	0	-1	4	-1	-1	-1	-1	-1	5	-3	-1	-1	-1	-1	-1	0
T	-1	-1	0	-1	0	0	-1	0	-1	-1	-1	-1	6	-3	-1	-1	-1	-1	-1	0
P	-3	-1	1	-1	-1	7	-1	-1	-1	-1	-1	-1	5	-3	-1	-1	-1	-1	-1	0
A	0	1	0	-1	4	-1	1	-1	-1	-1	-1	-1	6	-3	-1	-1	-1	-1	-1	0
G	-3	0	-2	0	6	-1	-1	-1	-1	-1	-1	-1	5	-3	-1	-1	-1	-1	-1	0
N	-1	-1	-2	-2	0	-1	-1	-1	-1	-1	-1	-1	6	-3	-1	-1	-1	-1	-1	0
E	0	0	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	6	-3	-1	-1	-1	-1	-1	0
D	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	6	-3	-1	-1	-1	-1	-1	0

BLOSUM62 est utilisé pour la fonction  $t(i,j)$

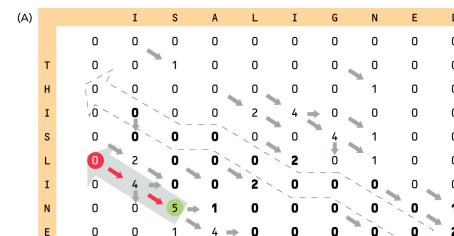
66



pénalité linéaire avec  $g = -4$

## L'alignement local 4

Des alignements sous-optimaux donnent de l'information sur des autres répétitions dans les deux séquences (par exemple des protéines avec plusieurs domaines)



pénalité linéaire avec  $E = 4$

Après avoir trouvé la séquence avec le plus haut score, on met toutes les valeurs qui appartiennent à ce chemin à zéro  
On recalcule les éléments de la matrice dans le voisinage du chemin optimal  
Seulement les éléments en dessous sont affectés par cette modification  
On ne recalcule plus pour les éléments qui ne changent plus

67

## L'alignement local 5

Exemple fait par LALIGN

Comparison of:  
(A) ./wwtmp/.10910.1.seq ABL 1130 bp  
                                SH2 Pkinase\_Tyr F\_actin\_bind  
  
(B) ./wwtmp/.10910.2.seq GRB2 217 bp  
                                SH2 SH2

```
using matrix file: /usr/molbio/share/fasta3/blosum62.mat (11/-4),
gap-open/ext: -10/-1 E(limit) 0.05
```

36.2% identity in 141 aa overlap (67-206:4-139); score: 235 E(10000): -1

70	80	90	100	110	120
ABL	VALIDPVASQNTLTSITKGKLLRVLGVGNHNGCEAQTKNGQGWPSNYITPVN				
GRB2	IAKYDFKATADDELSFKRGDLILKLNLNECQDNWYKAELNGKDGIFPNVYI				---EMKKPH
	10	20	30	40	50

130	140	150	160	170	180	
ABL	WTGCPVSRMAMTILSSSIIQ-GLVVVSESSGCGQSSRLAIVSVVYVYVYVPAQKKA					
GRB2	WFFGKIKPRAKAEMLSKQRIDGAFLIRESESAPDFSLSVKFGNDVQHFKVLRDGAK				Y	
	60	70	80	90	100	110

190	200					
ABL	VSSSERFVTLAEDVHHHSTVA					
GRB2	FLWVVFNFSLNELVDYHRSTS					
	120	130				

31.7% identity in 60 aa overlap (61-120:156-214); score: 87 E(10000): -1

70	80	90	100	110	120	
ABL	NDPNLFVALYDFVASQNTLTSITKGKLLRVLGVGNHNGCEAQTKNGQGWPSNYITPV					
GRB2	QQPTTYQALFDFFDPCEDGELGRGRDFINVWD-NSDPNWPKWGAHCQGHGFFPNVYI					
	160	170	180	190	200	210

68