Programmation dynamique pour l'analyse de séquences

Hélène Touzet

helene.touzet@univ-lille.fr

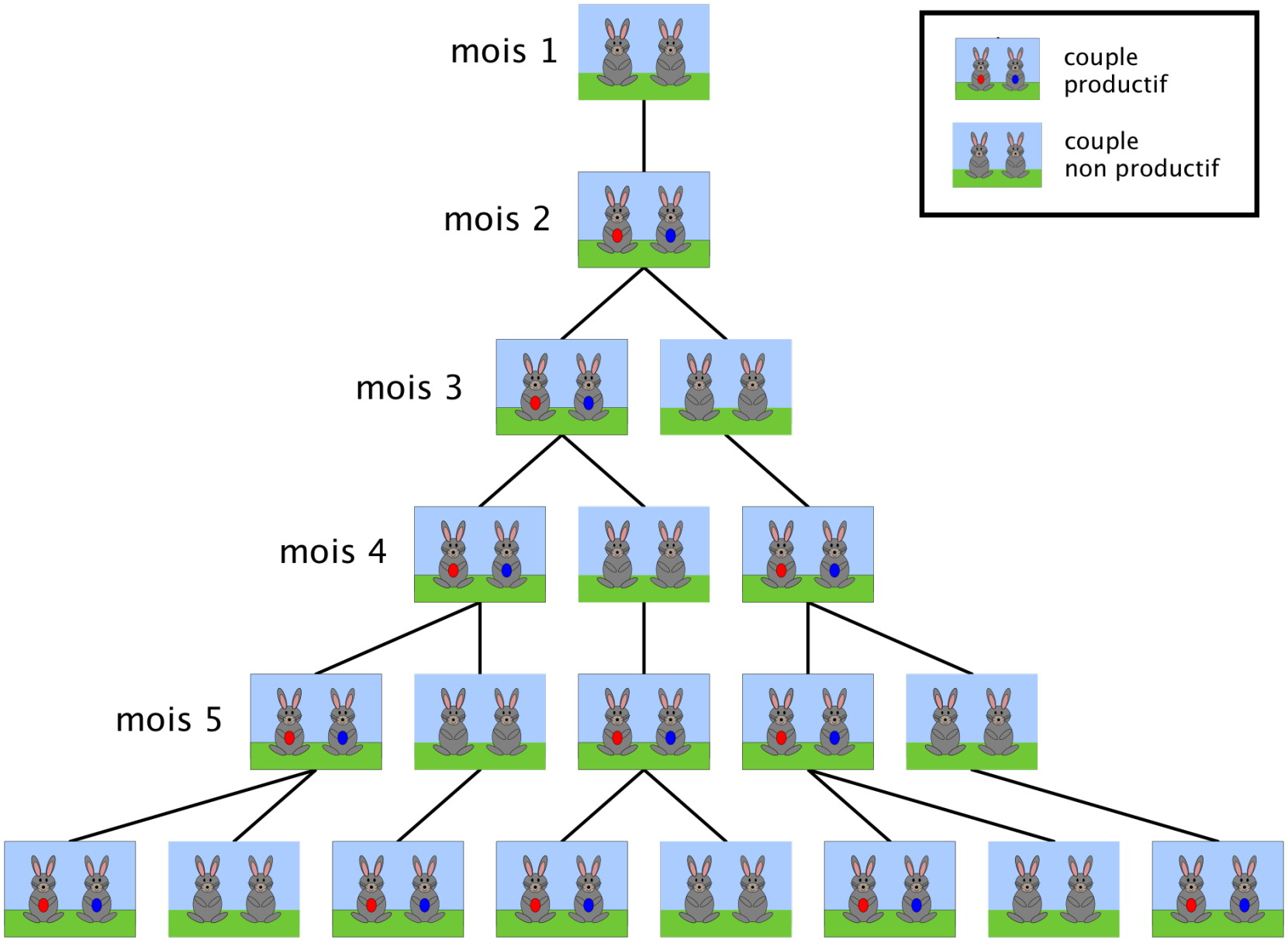
CNRS, Bonsai, CRIStAL

Les lapins de Fibonacci

Leonardo Fibonacci
(1175-1250)

- un couple de lapins productif donne naissance chaque mois à un nouveau couple
- un couple qui vient de naitre est productif au bout d'un mois, pour donner naissance à son premier couple le mois d'après
- les lapins ne meurent pas
- si on a un couple de lapins non encore productif au mois 1, combien a-t-on de couples au mois n ?

Les lapins de Fibonacci



Les lapins de Fibonacci

$$
\begin{aligned}
\text{Fibonacci}(0) &= 0 \\
\text{Fibonacci}(1) &= 1 \\
\text{Fibonacci}(n+2) &= \text{Fibonacci}(n)+\text{Fibonaci}(n+1)
\end{aligned}
$$

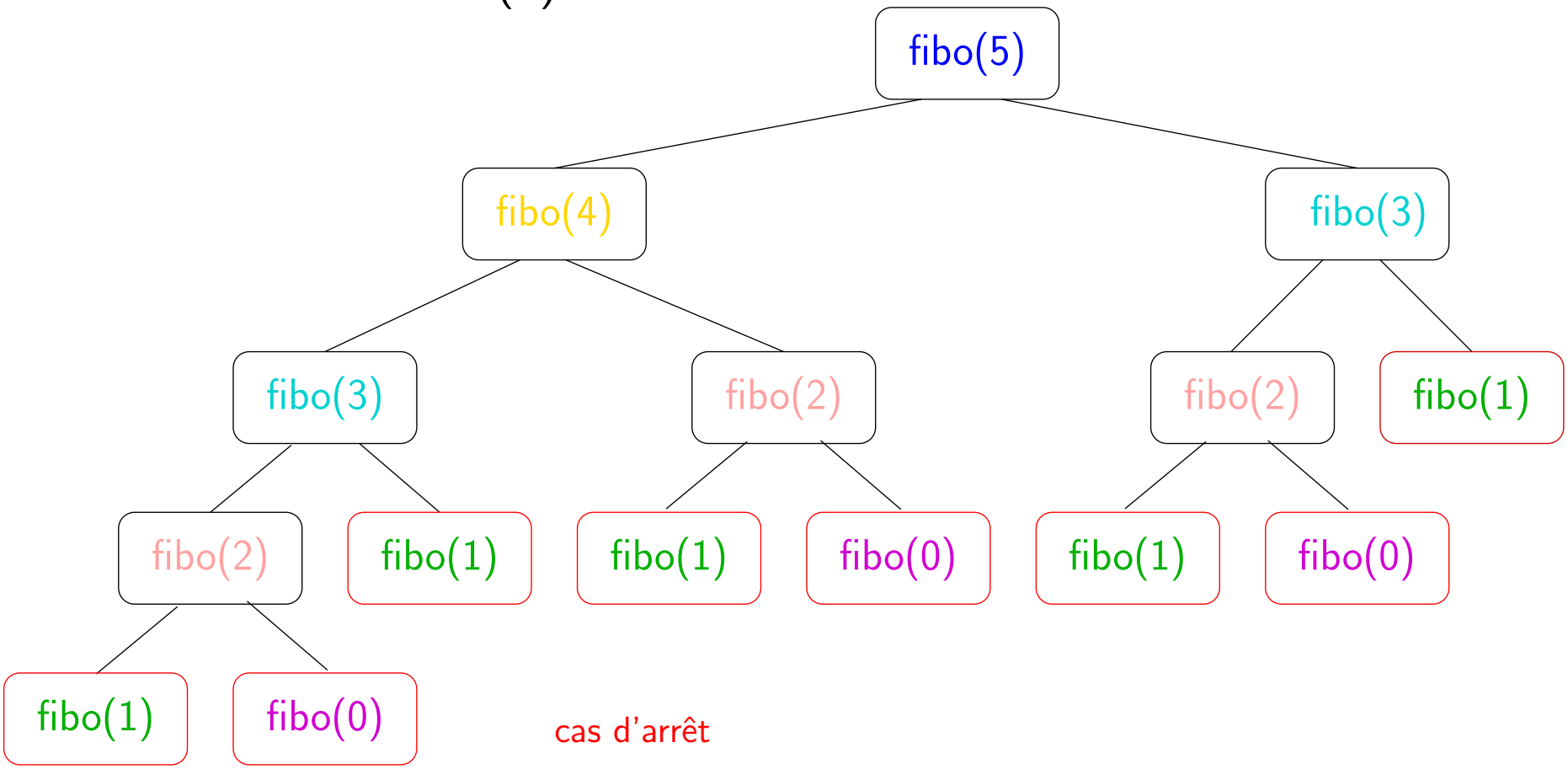
Formulation récursive

```
def fibonacci(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fibonacci(n - 1) + fibonacci(n - 2)
```

Où est le problème ?

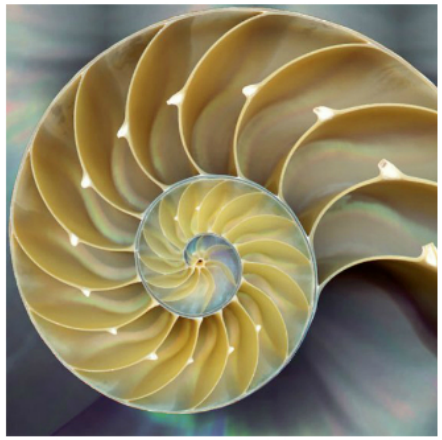
Arbre des appels récursifs pour

le calcul de `fibonacci(5)`

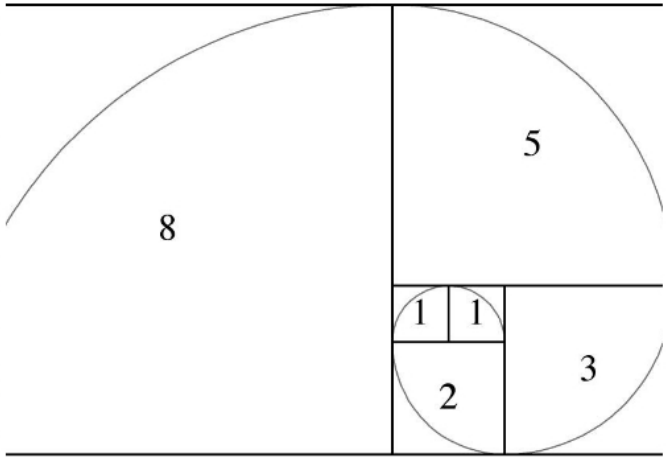


```
def fibonacci(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        num0 = 0
        num1 = 1
        series = 0
        for i in range(1,n):
            series = num0 + num1;
            num0 = num1;
            num1 = series;
        return series
```

Qu'est-ce qui change ?




5
8


8
5
1
1
2
3

Le problème du sac à doc

- un sac à dos de contenance maximale $C = 15kg$
- n objets de poids et de valeur différents : $p_1, \ldots, p_n$ et $v_1, \ldots, v_n$.

	poids (p_i)	valeur (v_i)
objet 1	2	1
objet 2	5	2
objet 3	7	3
objet 4	12	7
objet 5	9	10

Quelle est la sélection d'objets qui maximise la valeur totale tout en respectant la contenance de 15kg ?

Algorithme naïf

- on considère toutes les sélections possibles
- parmi celles de poids total $\leq C$, on garde celle de valeur maximale
- nombre de sélections à considérer ?

Algorithme pour le sac à dos

- $SD(i, p)$: valeur maximale atteignable avec les i premiers objets pour un poids inférieur ou égal à p
- $SD(n, C)$ donne la valeur maximale des objets emportés
- formules de récurrence pour le calcul de SD

  on regarde ce qu'on peut faire avec l'objet i

  - Si $p_i \leq p$, il y a deux possibilités : ajouter ou ne pas ajouter l'objet i

    $$SD(i, p) = \max \begin{cases} SD(i-1, p) & \text{pas d'ajoût de } i \\ SD(i-1, p-p_i) + v_i & \text{ajoût de } i \end{cases}$$

  - Si $p_i > p$, il n'y a pas le choix : l'objet i ne peut pas être ajouté

    $SD(i, p) = SD(i-1, p)$

- Implémentation : avec un tableau SD à deux entrées (i et p)
- Les 5 objets

poids p_i	2	5	7	12	9
valeur v_i	1	2	3	7	10

- Le tableau pour calculer la formule SD

SD	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	0	0	1	1	1	2	2	3	3	3	3	3	3	3	3	3
3	0	0	1	1	1	2	2	3	3	4	4	4	5	5	6	6
4	0	0	1	1	1	2	2	3	3	4	4	4	7	7	8	8
5	0	0	1	1	1	2	2	3	3	10	10	11	11	11	12	12

valeurs de p valeurs de i

Comment retrouver la sélection d'objets correspondant à la valeur optimale 12 ?

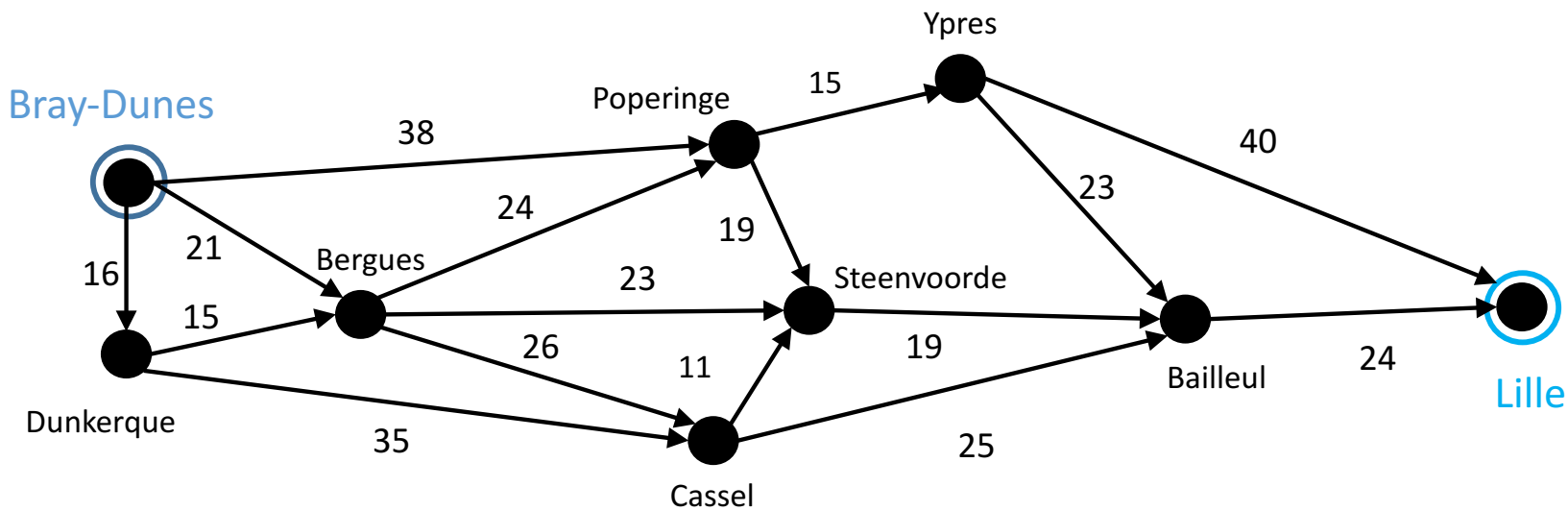
poids p_i	2	5	7	12	9
valeur v_i	1	2	3	7	10

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	0	0	1	1	1	2	2	3	3	3	3	3	3	3	3	3
3	0	0	1	1	1	2	2	3	3	4	4	4	5	5	6	6
4	0	0	1	1	1	2	2	3	3	4	4	4	7	7	8	8
5	0	0	1	1	1	2	2	3	3	10	10	11	11	11	12	12

objets sélectionnés objets non sélectionnés

Solution : $\{objet2, objet5\}$ pour un poids total de 14 kg et une valeur totale de 12.

Recherche d'un plus court chemin dans un graphe



Quel est le trajet de Bray-Dunes à Lille
qui prend le moins de temps ?

- graphe orienté sans cycle
  - ensemble de nœuds $n_0, \ldots, n_k$
  - ensemble d'arcs $A(i, j)$
- nœud source : n_0, `Bray-Dunes`
- nœud cible : n_k, `Lille`
- $T(i)$: temps minimum pour se rendre au nœud n_i à partir du nœud source n_0
- solution du problème : celle qui correspond au temps $T(k)$

Représentation du graphe sous forme de matrice d'adjacence

		BD	Du	Be	Po	Yp	Ca	St	Ba	Li
Bray-Dunes	BD		16	21						
Dunkerque	Du			15			35			
Bergues	Be				24		26	23		
Poperinge	Po					15		19		
Ypres	Yp								23	40
Cassel	Ca							11	25	
Steenvoorde	St								19	
Bailleul	Ba									24
Lille	Li									

- nœuds ordonnés, numérotés de 0 (source) à n (cible)

0	1	2	3	4	5	6	7	8
BD	Du	Be	Po	Yp	Ca	St	Ba	Li

- $T(i)$: temps minimal pour aller du nœud source au nœud i

$$\begin{aligned} T(0) &= 0 \\ T(i) &= \min\{T(j) + A(j,i), 0 \leq j < i\} \end{aligned}$$

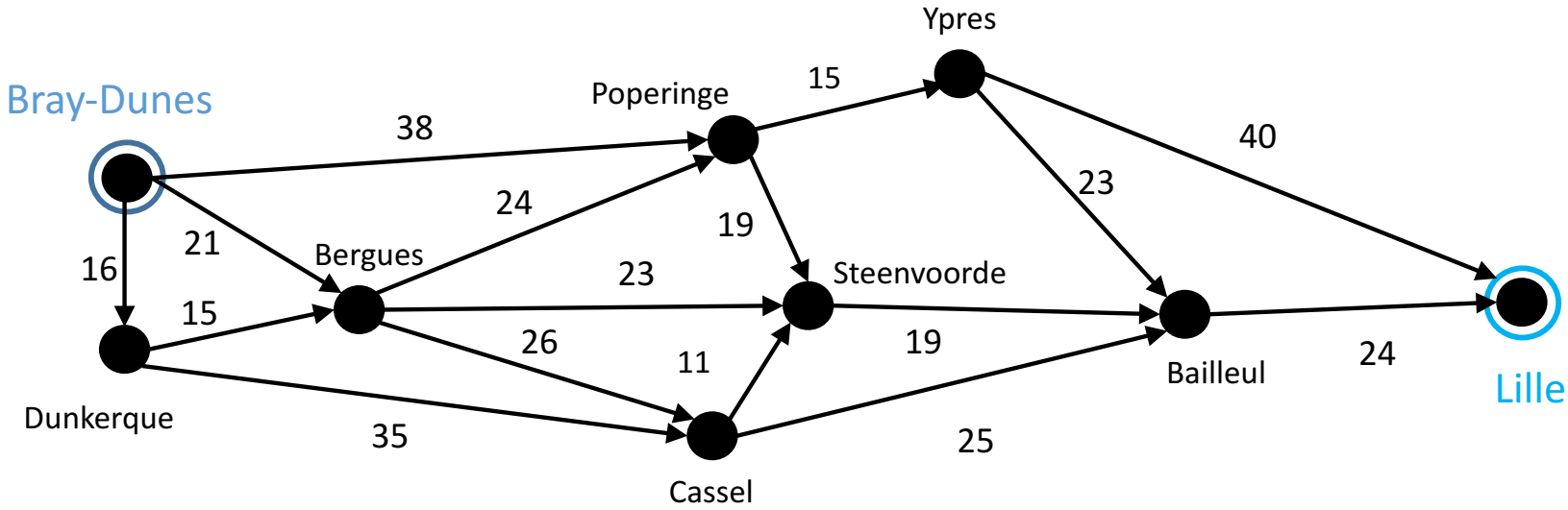


Tableau pour le calcul de T

	BD	Du	Be	Po	Yp	Ca	St	Ba	Li
T									

Trajet optimal ?

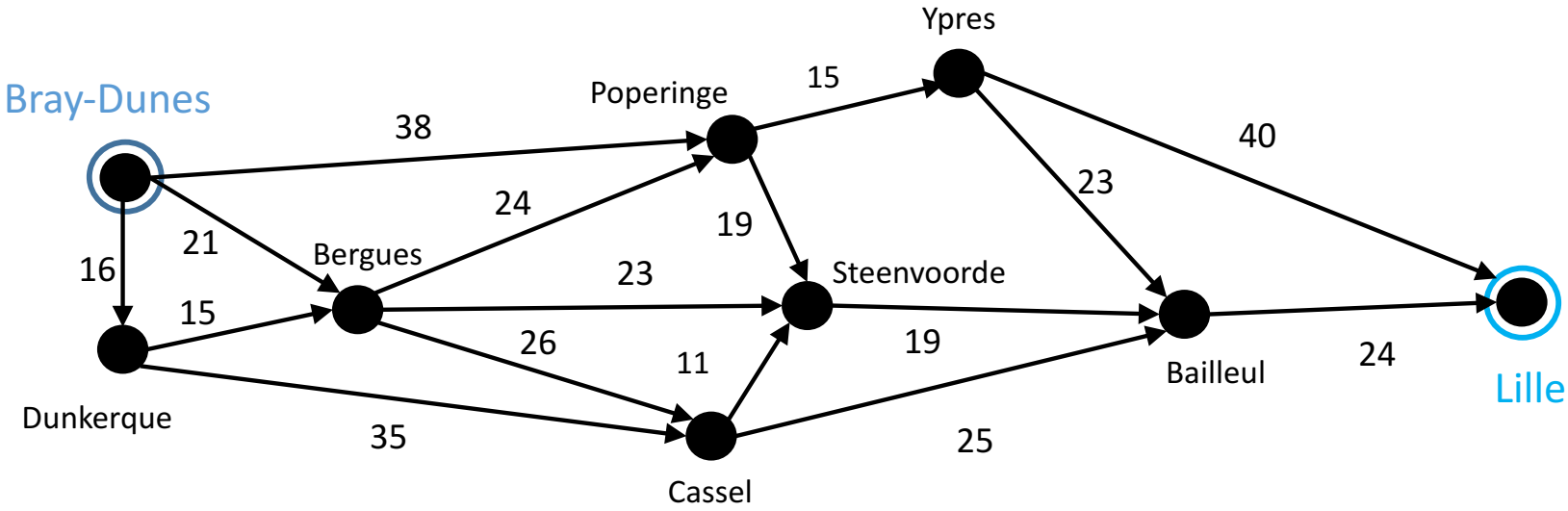


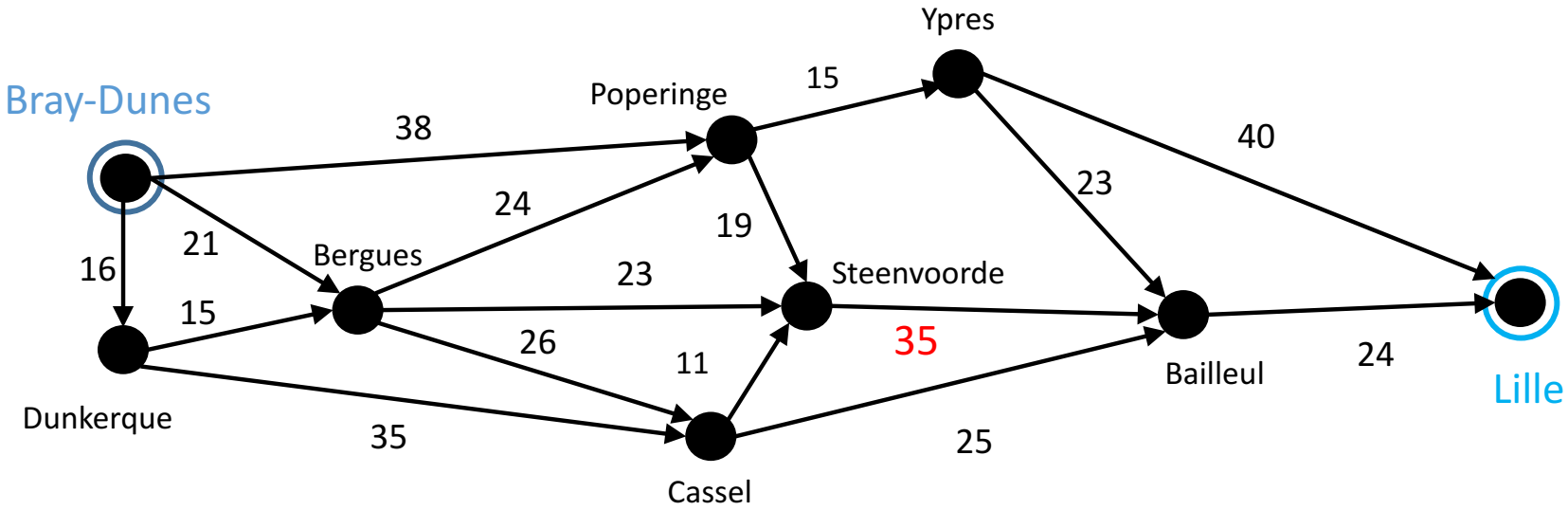
Tableau pour le calcul de T

	BD	Du	Be	Po	Yp	Ca	St	Ba	Li
T	0	16	21	38	53	47	44	63	87

Trajet optimal ?

Bray-Dunes $\rightarrow$ Bergues $\rightarrow$ Steenvorde $\rightarrow$ Bailleul $\rightarrow$ Lille

Accident sur le tronçon `Steenvoorde-Bailleul`



- Quelles valeurs doivent être modifiées dans le tableau T ?
- Nouveau tableau

	BD	Du	Be	Po	Yp	Ca	St	Ba	Li
T									

- Trajet optimal

Au sujet du sac à dos

https://interstices.info/le-probleme-du-sac-a-dos

Un sac de capacité 8 kg, trois objets

poids p_i	3	5	1
valeur v_i	40	50	20

Un sac de capacité 8 kg, trois objets

poids p_i	1	3	5
valeur v_i	20	40	50

Au sujet de recherche du plus court chemin dans un graphe sans cycle

Trouver le plus court chemin entre les s et t pour le graphe dont la matrice d'adjacence est donnée ci-dessous.

	s	a	b	c	d	t
s		19	8			
a			14		6	
b				4	22	
c					10	11
d						2
t						

L'algorithme fonctionne-t-il encore
si le graphe contient des cycles ?

Récursivité et mémoisation

- récursivité
  - fonction qui s'appelle elle-même avec une condition d'arrêt
  - simple à programmer
  - particulièrement adapté aux structures de données récursives (arbres, listes)
  - attention à la gestion de la mémoire
- mémoisation
  - mise en mémoire des valeurs intermédiaires

`Python` permet de faire de la mémoisation de manière automatique à partir d'une formulation récursive

- avec les decorators : `https://www.python-course.eu/python3_memoization.php`
- avec le module `functools` : `https://www.geeksforgeeks.org/python-functools-lru_cache/`

Programmation dynamique

- Un algorithme de *programmation dynamique* procède en réduisant le problème à plusieurs instances plus petites, elle-mêmes résolues par décomposition.
- Mémoisation : Les résultats des calculs intermédiaires sont stockés dans une table (matrice), ou une structure de donnée adéquate.
- La solution est ensuite construite à partir de la table, en remontant celle-ci.

  Trace back

Application à l'alignement de séquences

Exemple : séquence de l'insuline

```
éléphant     FVNQHLCGSHLVEALYLVCGERGFFYTPKTGIVEQCCTGVCSLYQLENYCN
             |||||||||||||||||||||||||| ||| ||||  ||||||||||
 hamster     FVNQHLCGSHLVEALYLVCGERGFFYTPKSGIVDQCCTSICSLYQLENYCN

éléphant     FVNQHLCGSHLVEALYLVCGERGFFYTPKTGIVEQCCTGVCSLYQLENYCN
             |||||||||||||||||||||||||| |||||||   ||||||||||
 baleine     FVNQHLCGSHLVEALYLVCGERGFFYTPKAGIVEQCCASTCSLYQLENYCN

éléphant     FVNQHLCGSHLVEALYLVCGERGFFYTPKTGIVEQCCTGVCSLYQLENYCN
               || ||||||| |||||||||||| || |||||||   ||||||||||
alligator    AANQRLCGSHLVDALYLVCGERGFFYSPKGGIVEQCCHNTCSLYQLENYCN
```

Alignement deux à deux

- modèle d'édition simple : `insertion` et `deletion`
  - alignement global : algorithme de Needleman et Wunsh
  - alignement semi-global
  - alignement local : algorithme de Smith et Waterman
- modèle avec gaps affines : `gap open` et `gap extend`

Alignement deux à deux, modèle simple

back to 1st semester

Alignement deux à deux, modèle simple

- mise en correspondance de deux séquences (ADN ou protéines)

```
R D I S L V - - - K N A G I
|   |   | |       | |   | |
R N I - L V S D A K N V G I
```

- 3 événements mutationnels élémentaires

substitution
insertion, délétion } indel

- score de chaque opération d'édition

identité, substitution : matrice de similarité $\rightarrow$ pénalité ou récompense
insertion, délétion (indel) : pénalité

- le score d'un alignement est la somme des scores de tous ces événements

Alignement global

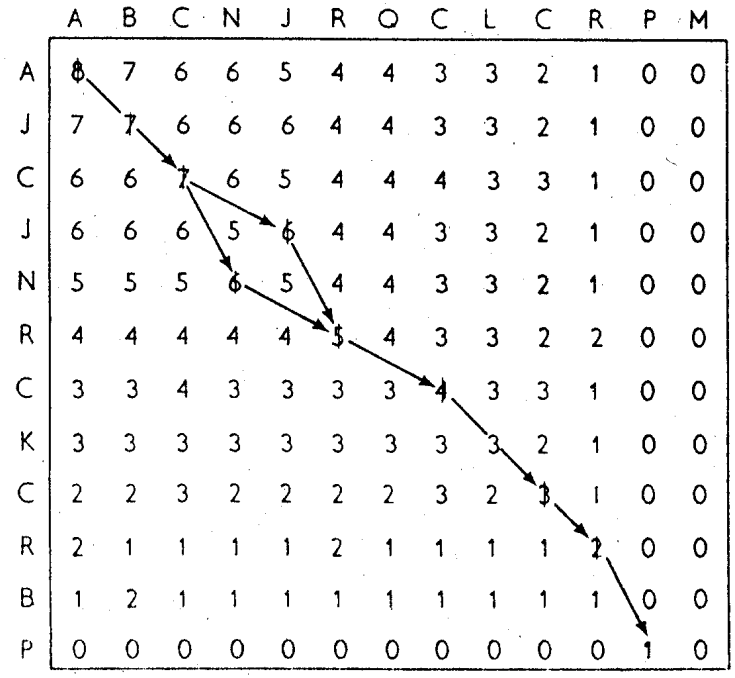
Needleman & Wunsch - 1970

- évaluation d'une ressemblance globale entre deux séquences
- alignement de deux gènes

Needleman & Wunsh (Journal of Molecular Biology, 1970)

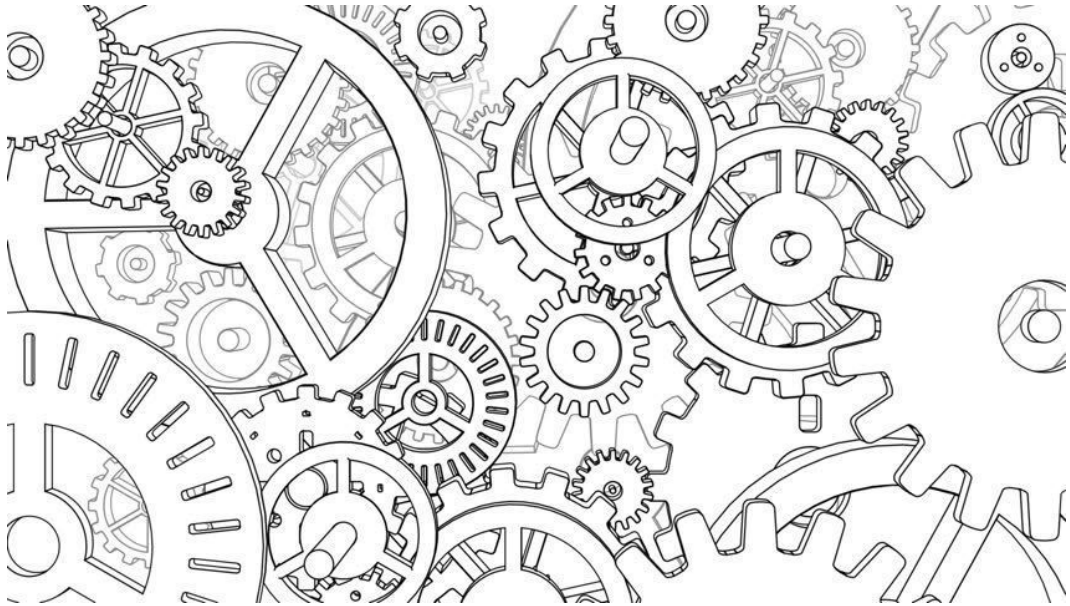
A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins

SAUL B. NEEDLEMAN AND CHRISTIAN D. WUNSCH
Department of Biochemistry, Northwestern University, and Nuclear Medicine Service, V. A. Research Hospital Chicago, Ill. 60611, U.S.A.



	A	B	C	N	J	R	O	C	L	C	R	P	M
A	8	7	6	6	5	4	4	3	3	2	1	0	0
J	7	7	6	6	6	4	4	3	3	2	1	0	0
C	6	6	7	6	5	4	4	4	3	3	1	0	0
J	6	6	6	5	6	4	4	3	3	2	1	0	0
N	5	5	5	6	5	4	4	3	3	2	1	0	0
R	4	4	4	4	4	5	4	3	3	2	2	0	0
C	3	3	4	3	3	3	3	4	3	3	1	0	0
K	3	3	3	3	3	3	3	3	3	2	1	0	0
C	2	2	3	2	2	2	2	3	2	3	1	0	0
R	2	1	1	1	1	2	1	1	1	1	2	0	0
B	1	2	1	1	1	1	1	1	1	1	1	0	0
P	0	0	0	0	0	0	0	0	0	0	0	1	0

Algorithme



- Séquences à aligner : ACGGCTAT et ACTGTAT
- Scores : match = 2, mismatch = -1 et indel= -2.
- Que peut-il se passer pour la dernière opération ?

Substitution de T en T

```
ACGGCTA T
  ? ? ? |
 ACTGTA T
```

score de ACGGCTA / ACTGTA $+2$

Délétion de T

```
ACGGCTA T
  ? ? ?
ACTGTAT -
```

score de ACGGCTA / ACTGTAT -2

Insertion de T

```
ACGGCTAT -
  ? ? ?
 ACTGTA T
```

score de ACGGCTAT / ACTGTA -2

- $\mathtt{S}(i,j)$: score optimal entre les préfixes $U(1..i)$ et $V(1..j)$.
- formule de récurrence

$$
\begin{aligned}
\mathtt{S}(0,0) &= 0 \\
\mathtt{S}(0,j) &= \mathtt{S}(0,j-1) + \mathit{Ins}(V(j)) \\
\mathtt{S}(i,0) &= \mathtt{S}(i-1,0) + \mathit{Del}(U(i)) \\
\mathtt{S}(i,j) &= \max \begin{cases} \mathtt{S}(i-1,j-1) + \mathit{Sub}(U(i),V(j)) \\ \mathtt{S}(i-1,j) + \mathit{Del}(U(i)) \\ \mathtt{S}(i,j-1) + \mathit{Ins}(V(j)) \end{cases}
\end{aligned}
$$

- méthode : programmation dynamique

calculs intermédiaires

=

scores d'alignements entre préfixes

Étape 1 : création d'une table indexée par les deux séquences

		A	C	G	G	C	T	A	T
A									
C									
T									
G									
T									
A									
T									

Case (i, j) : score entre les i premières bases de `ACGGCTAT` et les j premières bases de `ACTGTAT`.

Étape 1 : création d'une table indexée par les deux séquences

		A	C	G	G	C	T	A	T
	0	-2	-4	-6	-8	-10	-12	-14	-16
A	-2								
C	-4								
T	-6								
G	-8								
T	-10								
A	-12								
T	-14								

Case (i, j) : score entre les i premières bases de `ACGGCTAT` et les j premières bases de `ACTGTAT`.

Cas de base - initialisation

Étape 1 : création d'une table indexée par les deux séquences.

		A	C	G	G	C	T	A	T
	0	-2	-4	-6	-8	-10	-12	-14	-16
A	-2	2	0	-2	-4	-6	-8	-10	-12
C	-4								
T	-6								
G	-8								
T	-10								
A	-12								
T	-14								

Case (i, j) : score entre les i premières bases de `ACGGCTAT` et les j premières bases de `ACTGTAT`.

Remplissage ligne par ligne

Étape 1 : création d'une table indexée par les deux séquences

		A	C	G	G	C	T	A	T
	0	-2	-4	-6	-8	-10	-12	-14	-16
A	-2	2	0	-2	-4	-6	-8	-10	-12
C	-4	0	4	2	0	-2	-4	-6	-8
T	-6	-2	2	3	1	-1	0	-2	-4
G	-8	-4	0	4	5	3	1	-1	-3
T	-10	-6	-2	2	3	4	5	3	1
A	-12	-8	-4	0	1	2	3	7	5
T	-14	-10	-6	-2	-1	0	4	5	9

Case (i, j) : score entre les i premières bases de `ACGGCTAT` et les j premières bases de `ACTGTAT`.

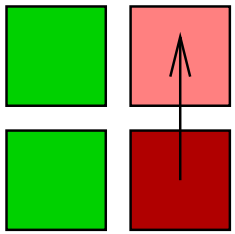
Remplissage ligne par ligne

Étape 2 : recherche du chemin des scores maximaux dans la matrice

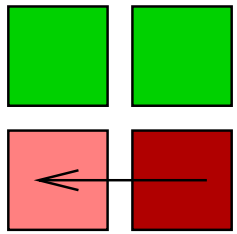
		A	C	G	G	C	T	A	T
	0	-2	-4	-6	-8	-10	-12	-14	-16
A	-2	2	0	-2	-4	-6	-8	-10	-12
C	-4	0	4	2	0	-2	-4	-6	-8
T	-6	-2	2	3	1	-1	0	-2	-4
G	-8	-4	0	4	5	3	1	-1	-3
T	-10	-6	-2	2	3	4	5	3	1
A	-12	-8	-4	0	1	2	3	7	5
T	-14	-10	-6	-2	-1	0	4	5	9

Étape 3 : construction de l'alignement

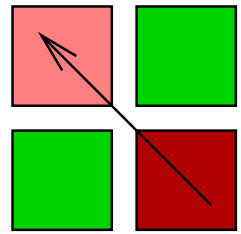
- sur le chemin des scores maximaux, on regarde quelle est l'opération correspondante.



insertion



délétion



substitution
ou
identité

- résultat

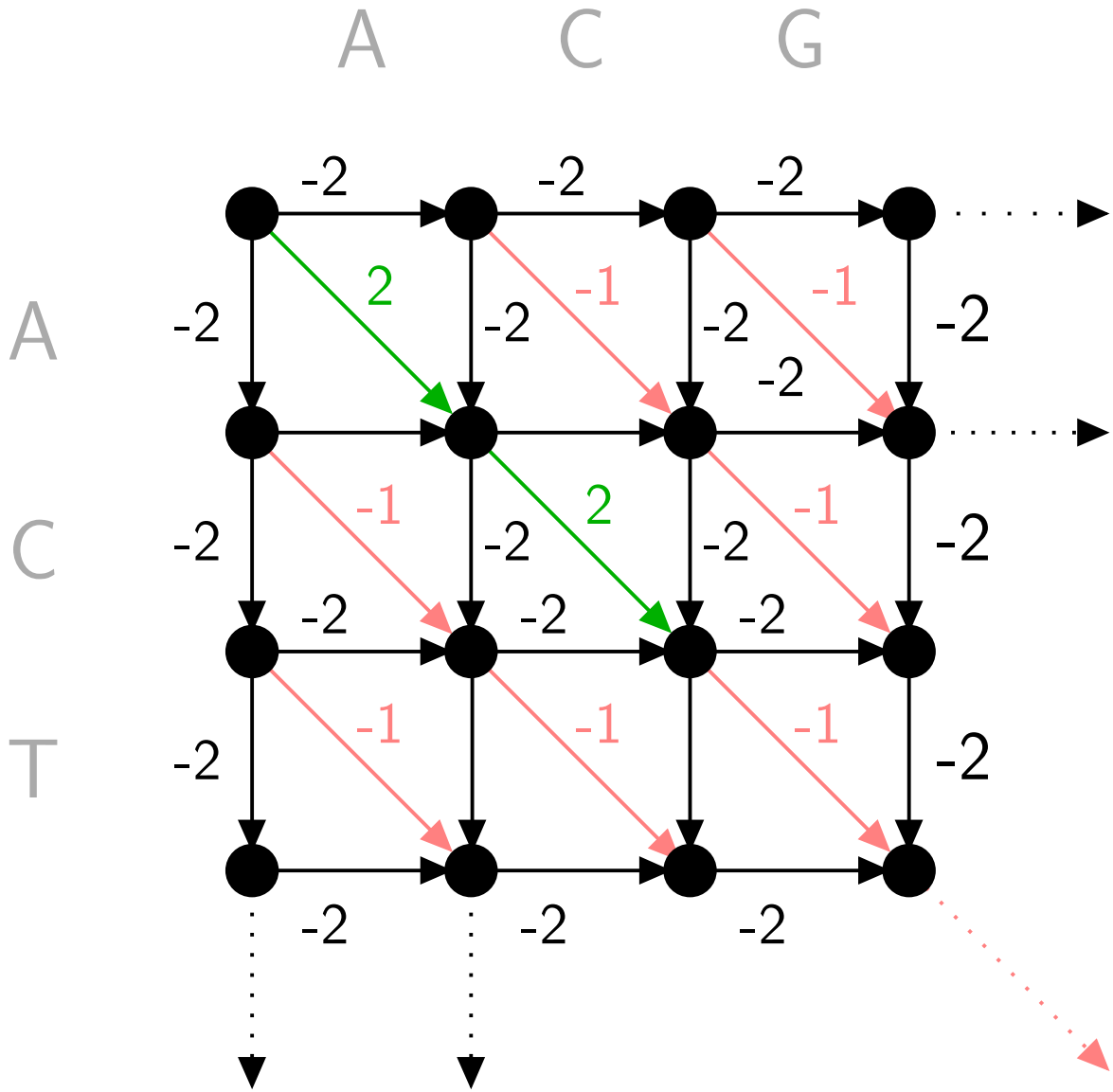
```
A C G G C T A T
| |   |   | | |
A C T G - T A T
```

Complexité de l'algorithme

- m longueur de la première séquence, n longueur de la seconde séquence
- Pour le calcul du score d'alignement uniquement (étape 1)
  - $O(n \times m)$ en temps
  - $O(\min\{n, m\})$ en espace
- Pour la construction de l'alignement (étapes 1, 2 et 3)
  - $O(n \times m)$ en temps et en espace

Vous avez déjà vu cet algorithme dans ce cours.

Où ?


A C G
A
C
T
-2
2
-1

Exercice

score : match +2, mismatch -1, indel -3

		T	T	G	T	C	A	A	G	T
	0	-3	-6	-9	-12	-15	-18	-21	-24	-27
A	-3	-1	-4	-7	-10	-13	-13	-16	-19	-22
T	-6	-1	1	-2	-5	-8	-11	-14	-17	-17
T	-9	-4	1	0	0	-3	-6	-9	-12	-15
G	-12	-7	-2	3	0	-1	-4	-7	-7	-10
C	-15	-10	-5	0	2	2	-1	-4	-7	-8
A	-18	-13	-8	-3	-1	1	4	1	-2	-5
G	-21	-16	-11	-6	-4	-2	1	3	3	0
T	-24	-19	-14	-9	-4	-5	-2	0	2	5
A	-27	-22	-17	-12	-7	-5	-3	0	-1	2
G	-30	-25	-20	-15	-10	-8	-6	-3	2	-1
C	-33	-28	-23	-18	-13	-8	-9	-6	-1	1

Quel est l'alignement global optimal ?

Exercice

		T	T	G	T	C	A	A	G	T
	0	-3	-6	-9	-12	-15	-18	-21	-24	-27
A	-3	-1	-4	-7	-10	-13	-13	-16	-19	-22
T	-6	-1	1	-2	-5	-8	-11	-14	-17	-17
T	-9	-4	1	0	0	-3	-6	-9	-12	-15
G	-12	-7	-2	3	0	-1	-4	-7	-7	-10
C	-15	-10	-5	0	2	2	-1	-4	-7	-8
A	-18	-13	-8	-3	-1	1	4	1	-2	-5
G	-21	-16	-11	-6	-4	-2	1	3	3	0
T	-24	-19	-14	-9	-4	-5	-2	0	2	5
A	-27	-22	-17	-12	-7	-5	-3	0	-1	2
G	-30	-25	-20	-15	-10	-8	-6	-3	2	-1
C	-33	-28	-23	-18	-13	-8	-9	-6	-1	1

```
- T T G T C A - - A G T
  | | |   | |     | |
A T T G - C A G T A G C
```

Alignement semi-global

Variation de l'alignement global

- les gaps en début et en fin de séquence ne sont pas pénalisés

  meilleur alignement global entre un préfixe d'une séquence et un suffixe de l'autre séquence, ou entre une séquence et un facteur de l'autre séquence
- permet de s'affranchir des artefacts d'alignements dus aux séquences incomplètes
- souvent le mode par défaut des aligneurs globaux


```
Alignement global
CAGCACTTGGATTCTCGG
||||     | |    ||
CAGC-----G-T----GG
```

```
Alignement semi-global
 ACAGCA-CTTGGATTCTCGG
     || | |||
     CAGCGTGG
```

Que faut-il changer dans l'algorithme de Needleman et Wunsh ?

Que faut-il changer dans l'algorithme de Needleman et Wunsh ?

- initialisation à 0 sur la première ligne et la première colonne
- recherche du résultat sur la dernière ligne et la dernière colonne

Alignement local

Smith & Waterman - 1981

- identification des régions de forte similarité entre les deux séquences
- regions conservées


```
Alignement global
1  G G C T G A C C A C C _ T T G T A - - -   16
   |     |   | |     | |   |   |   |
1  G A _ T C A C T T C C A T G G C A G T A   19

Alignement local
5   G A C C A C C T T  13       14   G T A   16
    | |   | | | |   | |              | | |
1   G A T C A C _ T T  8        17   G T A   19
```

Smith & Waterman (Journal of Molecular Biology, 1981)

Identification of Common Molecular Subsequences

The identification of maximally homologous subsequences among sets of long sequences is an important problem in molecular sequence analysis. The problem is straightforward only if one restricts consideration to contiguous subsequences (segments) containing no internal deletions or insertions. The more general problem has its solution in an extension of sequence metrics (Sellers 1974; Waterman *et al.*, 1976) developed to measure the minimum number of "events" required to convert one sequence into another.

These developments in the modern sequence analysis began with the heuristic homology algorithm of Needleman & Wunsch (1970) which first introduced an iterative matrix method of calculation. Numerous other heuristic algorithms have been suggested including those of Fitch (1966) and Dayhoff (1969). More mathematically rigorous algorithms were suggested by Sankoff (1972), Reichert *et al.* (1973) and Beyer *et al.* (1979), but these were generally not biologically satisfying or interpretable. Success came with Sellers (1974) development of a true metric measure of the distance between sequences. This metric was later generalized by Waterman *et al.* (1976) to include deletions/insertions of arbitrary length. This metric represents the minimum number of "mutational events" required to convert one sequence into another. It is of interest to note that Smith *et al.* (1980) have recently shown that under some conditions the generalized Sellers metric is equivalent to the original homology algorithm of Needleman & Wunsch (1970).

In this letter we extend the above ideas to find a pair of segments, one from each of two long sequences, such that there is no other pair of segments with greater similarity (homology). The similarity measure used here allows for arbitrary length deletions and insertions.

Algorithm

The two molecular sequences will be $\underset{\sim}{A}=a_1a_2\ldots a_n$ and $\underset{\sim}{B}=b_1b_2\ldots b_m$. A similarity $s(a,b)$ is given between sequence elements a and b. Deletions of length k are given weight W_k. To find pairs of segments with high degrees of similarity, we set up a matrix H. First set

$$H_{k0}=H_{0l}=0 \text{ for } 0\le k\le n \text{ and } 0\le l\le m.$$

Preliminary values of H have the interpretation that H_{ij} is the maximum similarity of two segments *ending* in a_i and b_j, respectively. These values are obtained from the relationship

$$H_{ij}=\max\{H_{i-1,j-1}+s(a_i,b_j),\ \max_{k\ge1}\{H_{i-k,j}-W_k\},\ \max_{l\ge1}\{H_{i,j-l}-W_l\},\ 0\},\quad (1)$$

$1\le i\le n$ and $1\le j\le m$.

The formula for H_{ij} follows by considering the possibilities for ending the segments at any a_i and b_j.

(1) If a_i and b_j are associated, the similarity is

$$H_{i-1,j-1}+s(a_i,b_j).$$

(2) If a_i is at the end of a deletion of length k, the similarity is

$$H_{i-k,j}-W_k.$$

(3) If b_j is at the end of a deletion of length l, the similarity is

$$H_{i-k,j}-W_l.$$

(4) Finally, a zero is included to prevent calculated negative similarity, indicating no similarity up to a_i and b_j.†

The pair of segments with maximum similarity is found by first locating the maximum element of H. The other matrix elements leading to this maximum value are than sequentially determined with a traceback procedure ending with an element of H equal to zero. This procedure identifies the segments as well as produces the corresponding alignment. The pair of segments with the next best similarity is found by applying the traceback procedure to the second largest element of H not associated with the first traceback.

A simple example is given in Figure 1. In this example the parameters $s(a_ib_j)$ and W_k required were chosen on an *a priori* statistical basis. A match, $a_i=b_j$, produced an $s(a_ib_j)$ value of unity while a mismatch produced a minus one-third. These values have an average for long, random sequences over an equally probable four letter set of zero. The deletion weight must be chosen to be at least equal to the difference between a match and a mismatch. The value used here was $W_k=1{\cdot}0+1/3{*}k$.

	Δ	C	A	G	C	C	U	C	G	C	U	U	A	G
Δ	0·0	0·0	0·0	0·0	0·0	0·0	0·0	0·0	0·0	0·0	0·0	0·0	0·0	0·0
A	0·0	0·0	1·0	0·0	0·0	0·0	0·0	0·0	0·0	0·0	0·0	0·0	1·0	0·0
A	0·0	0·0	1·0	0·7	0·0	0·0	0·0	0·0	0·0	0·0	0·0	0·0	1·0	0·7
U	0·0	0·0	0·0	0·7	0·3	0·0	1·0	0·0	0·0	0·0	1·0	1·0	0·0	0·7
G	0·0	0·0	0·0	1·0	0·3	0·0	0·0	0·7	1·0	0·0	0·0	0·7	0·7	1·0
C	0·0	1·0	0·0	0·0	2·0	1·3	0·3	1·0	0·3	2·0	0·7	0·3	0·3	0·3
C	0·0	1·0	0·7	0·0	1·0	3·0	1·7	1·3	1·0	1·3	1·7	0·3	0·0	0·0
A	0·0	0·0	2·0	0·7	0·3	1·7	2·7	1·3	1·0	0·7	1·0	1·3	1·3	0·0
U	0·0	0·0	0·7	1·7	0·3	1·3	2·7	2·3	1·0	0·7	1·7	2·0	1·0	1·0
U	0·0	0·0	0·3	0·3	1·3	1·0	2·3	2·3	2·0	0·7	1·7	2·7	1·7	1·0
G	0·0	0·0	0·0	1·3	0·0	1·0	1·0	2·0	3·3	2·0	1·7	1·3	2·3	2·7
A	0·0	0·0	1·0	0·0	1·0	0·3	0·7	0·7	2·0	3·0	1·7	1·3	2·3	2·0
C	0·0	1·0	0·0	0·7	1·0	2·0	0·7	1·7	1·7	3·0	2·7	1·3	1·0	2·0
G	0·0	0·0	0·7	1·0	0·3	0·7	1·7	0·3	2·7	1·7	2·7	2·3	1·0	2·0
G	0·0	0·0	0·0	1·7	0·7	0·3	0·3	1·3	1·3	2·3	1·3	2·3	2·0	2·0

FIG. 1. H_{ij} matrix generated from the application of eqn (1) to the sequences A-A-U-G-C-C-A-U-U-G-A-C-G-G and C-A-G-C-C-U-C-G-C-U-U-A-G. The underlined elements indicate the trackback path from the maximal element 3·30.

† Zero need not be included unless there are negative values of $s(a,b)$.

- $\mathrm{Loc}(i,j)$: score optimal entre un suffixe de $U(1..i)$ et un suffixe $V(1..j)$
- Formule de récurrence

$$
\begin{aligned}
\mathrm{Loc}(0,0) &= 0 \\
\mathrm{Loc}(0,j) &= 0 \\
\mathrm{Loc}(i,0) &= 0 \\
\mathrm{Loc}(i,j) &= \max \begin{cases} \mathrm{Loc}(i-1,j-1) + \mathit{Sub}(U(i),V(j)) \\ \mathrm{Loc}(i-1,j) + \mathit{Del}(U(i)) \\ \mathrm{Loc}(i,j-1) + \mathit{Ins}(V(j)) \\ 0 \end{cases}
\end{aligned}
$$

- alignement de meilleur score : valeur max dans la matrice
- Complexité : $n \times m$
- Implémentation : voir TP

	G	G	C	T	G	A	C	C	A	C	C	T	T
G	2	1	0	0	2	1	0	0	0	0	0	0	0
A	1	1	0	0	0	4	3	2	2	1	0	0	0
T	0	0	0	2	1	0	3	2	1	0	0	2	2
C	0	0	2	1	1	0	2	5	4	3	2	1	1
A	0	0	1	1	0	3	2	4	7	6	5	4	3
C	0	0	2	1	0	2	5	4	6	9	8	7	6
T	0	0	1	4	3	2	4	4	5	8	8	10	9
T	0	0	0	3	3	2	3	3	4	7	7	10	12
C	0	0	2	2	2	2	4	5	4	6	6	9	11
C	0	0	2	1	1	1	4	6	4	6	8	8	10
A	0	0	1	1	0	3	3	5	8	7	7	7	9
T	0	0	0	3	2	2	2	4	7	7	6	9	11
G	2	2	1	2	5	4	3	3	6	6	6	8	10

Quel est le score du meilleur alignement local ? Quel est l'alignement correspondant ?

	G	G	C	T	G	A	C	C	A	C	C	T	T
G	2	1	0	0	2	1	0	0	0	0	0	0	0
A	1	1	0	0	0	4	3	2	2	1	0	0	0
T	0	0	0	2	1	0	3	2	1	0	0	2	2
C	0	0	2	1	1	0	2	5	4	3	2	1	1
A	0	0	1	1	0	3	2	4	7	6	5	4	3
C	0	0	2	1	0	2	5	4	6	9	8	7	6
T	0	0	1	4	3	2	4	4	5	8	8	10	9
T	0	0	0	3	3	2	3	3	4	7	7	10	12
C	0	0	2	2	2	2	4	5	4	6	6	9	11
C	0	0	2	1	1	1	4	6	4	6	8	8	10
A	0	0	1	1	0	3	3	5	8	7	7	7	9
T	0	0	0	3	2	2	2	4	7	7	6	9	11
G	2	2	1	2	5	4	3	3	6	6	6	8	10

Score du meilleur alignement local : 12

Poids des opérations d'édition : 2 identité, -1 substitution, -1 indel

```
G A C C A C C T T
| |   | |   | | |
G A T C A - C T T
```

Alignement 2 à 2 : Amélioration du modèle avec le traitement des gaps

- Gap : succession de délétions ou d'insertions
- Un gap correspond à un seul événement mutationnel
- Système de score affine
  - `Ouv` : pénalité d'ouverture de gap
  - `Ext` : pénalité d'extension de gap

```
AAGTCATTCGCACATCG
||  |||||   ||| |
AACACATTC---CATGG
```

identité	: 5
substitution	: -4
ouverture de gap	: -10
extension de gap	: -0,5

Quel est le score de cet alignement ?

On peut distinguer deux manières de compter : en appliquant l'extension gu gap dès la première position ou à partir de la deuxième position seulement.

Algorithme

Trois tables

$a(i,j)$ score maximal d'un alignement entre $U(1..i)$ et $V(1..j)$ qui termine par un match ou un mismatch entre $U(i)$ et $V(j)$

$b(i,j)$ score maximal d'un alignement entre $U(1..i)$ et $V(1..j)$ qui termine par l'insertion de $V(j)$

$c(i,j)$ score maximal d'un alignement entre $U(1..i)$ et $V(1..j)$ qui termine par la délétion de $U(i)$

Formules de récurrence

`Ouv` : pénalité d'ouverture de gap
`Ext` : pénalité d'extension de gap

$$a(i,j) = Sub(i,j) + \max \begin{cases} a(i-1,j-1) \\ b(i-1,j-1) \\ c(i-1,j-1) \end{cases}$$

$$b(i,j) = \max \begin{cases} \texttt{Ouv} + \texttt{Ext} + a(i,j-1) \\ \texttt{Ext} + b(i,j-1) \\ \texttt{Ouv} + \texttt{Ext} + c(i,j-1) \end{cases}$$

$$c(i,j) = \max \begin{cases} \texttt{Ouv} + \texttt{Ext} + a(i-1,j) \\ \texttt{Ouv} + \texttt{Ext} + b(i-1,j) \\ \texttt{Ext} + c(i-1,j) \end{cases}$$

: L'extension est appliquée dès la première position du gap

Initialisation

$$
\begin{aligned}
a(0,0) &= 0 \\
a(i,0) &= -\infty \\
a(0,j) &= -\infty \\
\\
b(i,0) &= -\infty \\
b(0,j) &= \texttt{Ouv} + \texttt{Ext} \times j \\
\\
c(i,0) &= \texttt{Ouv} + \texttt{Ext} \times i \\
c(0,j) &= -\infty
\end{aligned}
$$

Exemple : EAGAWGHE et PAWHEAE

- score des substitutions

	A	E	G	H	P	W
A	5	-1	0	-2	-1	-3
E		6	-3	0	-1	-3
G			8	-2	-2	-3
H				10	-2	-3
P					10	-4
W						15

- pénalités d'ouverture de gaps : -10
- pénalités d'extension : -2
- alignement global avec gaps affines optimal

```
E A G A W G H - E
P - - A W H E A E
```

a (substitution)

		E	A	G	A	W	G	H	E
	0	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
P	$-\infty$	-1	-13	-16	-17	-22	-22	-24	-25
A	$-\infty$	-13	4	-13	-10	-20	-19	-23	-24
W	$-\infty$	-17	-16	1	-11	5	-15	-17	-19
H	$-\infty$	-16	-17	-10	-1	-14	3	3	-9
E	$-\infty$	-12	-17	-13	-11	-4	-10	3	9
A	$-\infty$	-21	-7	-12	-8	-14	-4	-11	2
E	$-\infty$	-16	-22	-10	-13	-11	-14	-4	-3

b (insertion)

		E	A	G	A	W	G	H	E
	0	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
P	-12	-24	-26	-28	-30	-32	-34	-36	-38
A	-14	-13	-25	-27	-29	-31	-33	-35	-37
W	-16	-15	-8	-20	-22	-24	-26	-28	-30
H	-18	-17	-10	-11	-23	-7	-19	-21	-23
E	-20	-19	-12	-13	-13	-9	-9	-9	-21
A	-22	-21	-14	-15	-15	-11	-11	-9	-3
E	-24	-23	-16	-17	-17	-13	-13	-11	-5

c (délétion)

		E	A	G	A	W	G	H	E
	0	-12	-14	-16	-18	-20	-22	-24	-26
P	$-\infty$	-24	-13	-15	-17	-19	-21	-23	-25
A	$-\infty$	-26	-25	-8	-10	-12	-14	-16	-18
W	$-\infty$	-28	-27	-20	-11	-13	-7	-9	-11
H	$-\infty$	-30	-28	-22	-22	-13	-15	-9	-9
E	$-\infty$	-32	-24	-24	-25	-23	-16	-18	-9
A	$-\infty$	-34	-33	-19	-21	-20	-22	-16	-18
E	$-\infty$	-36	-28	-28	-22	-24	-23	-25	-16