

mot = suite sur l'alphabet A

sous-mot = sous-suite

m é d i t e r r a n é e
 m e r

$u = u_0 u_1 \dots u_{k-1}$ sous-mot de $x = x_0 x_1 \dots x_{m-1}$

si $x = v_0 u_0 v_1 u_1 \dots v_k u_{k-1} v_k$ avec $v_0, v_1, \dots, v_k \in A^*$

« sous-mot de » : relation d'ordre sur A^*

$SC(x, y) = \{u \text{ sous-mot de } x \text{ et de } y\}$

Problème 1 :

Calculer $PLSC(x, y) = \max \{ |u| \mid u \in SC(x, y) \}$

Problème 2

Déterminer un mot $u \in SC(x, y)$ tel que $|u| = PLSC(x, y)$

Comparaison de mots par distance

$$d(x, y) = |x| + |y| - 2 \cdot \text{PLCS}(x, y)$$

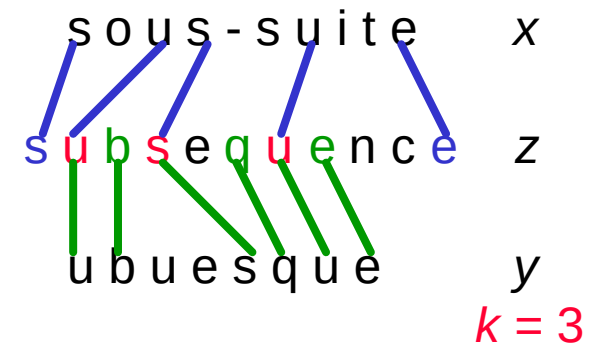
d est une distance

- $d(x, y) \geq 0$
- $d(x, y) = 0$ ssi $x = y$
- $d(x, y) = d(y, x)$
- inégalité triangulaire ?

u = un plcs de x et z

v = un plcs de z et y

k = nombre d'occurrences de lettres de z communes à u et v



$$\begin{aligned}
 d(x, z) + d(z, y) &= (|x| + |z| - 2 \cdot |u|) + (|z| + |y| - 2 \cdot |v|) \\
 &= |x| + |y| + (|z| - |u|) + (|z| - |v|) - |u| - |v| \\
 &\stackrel{3}{=} |x| + |y| + (|v| - k) + (|u| - k) - |u| - |v| \\
 &= |x| + |y| - 2k \\
 &\stackrel{3}{=} d(x, y)
 \end{aligned}$$

Comparaison de fichiers

lettre = ligne

> cat A

Belle Marquise,
vos beaux yeux
me font mourir d'amour

> cat B

D'amour mourir me font,
Belle Marquise,
vos beaux yeux

> diff A B**0 a 1****> D'amour mourir me font,****3 d 3****< me font mourir d'amour****Applications**

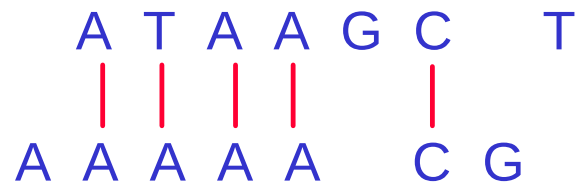
gestion de versions

compression par stockage de différences

restauration avec "ed" , "sed", ...

Comparaisons de séquences moléculaires

lettre = nucléotide



Transformation de x en y par :

insertions, suppressions, remplacements

Chaque opération possède un coût

Problème :

calculer le coût minimal de la transformation
(distance d'alignement)
un alignement correspondant

Distance d'édition

Par énumération de sous-mots
temps exponentiel

Par "programmation dynamique"
temps $O(|x| \times |y|)$

Par automate
même temps

Propriété

UMLV ©

$$x = x_0 x_1 \dots x_{m-1} \quad y = y_0 y_1 \dots y_{n-1}$$

$$u = u_0 u_1 \dots u_{k-1} \quad \text{un plcs de } x \text{ et } y$$

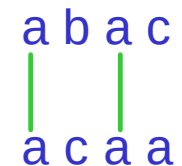
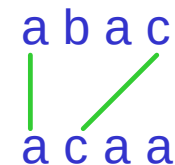
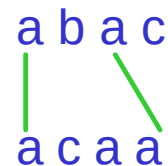
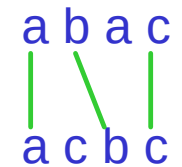
Proposition

$$x_{m-1} = y_{n-1} \quad \text{p} \quad u_{k-1} = x_{m-1} = y_{n-1}$$

$$\text{et } u_0 u_1 \dots u_{k-2} \text{ plcs de } x_0 x_1 \dots x_{m-2} \text{ et } y_0 y_1 \dots y_{n-2}$$

$$x_{m-1} \neq y_{n-1} \text{ et } u_{k-1} = x_{m-1} \quad \text{p} \quad u \text{ plcs de } x_0 x_1 \dots x_{m-2} \text{ et } y$$

$$x_{m-1} \neq y_{n-1} \text{ et } u_{k-1} = y_{n-1} \quad \text{p} \quad u \text{ plcs de } x \text{ et } y_0 y_1 \dots y_{n-2}$$



```
entier PLSC(mot x, longueur m, mot y, longueur n){  
    si (m = 0 ou n = 0) alors  
        retour 0 ;  
    sinon si (x[m-1] = y[n-1]) alors  
        retour PLSC(x, m-1, y, n-1)+1  
    sinon  
        retour max{PLSC(x, m, y, n-1), PLSC(x, m-1, y, n)}  
}
```

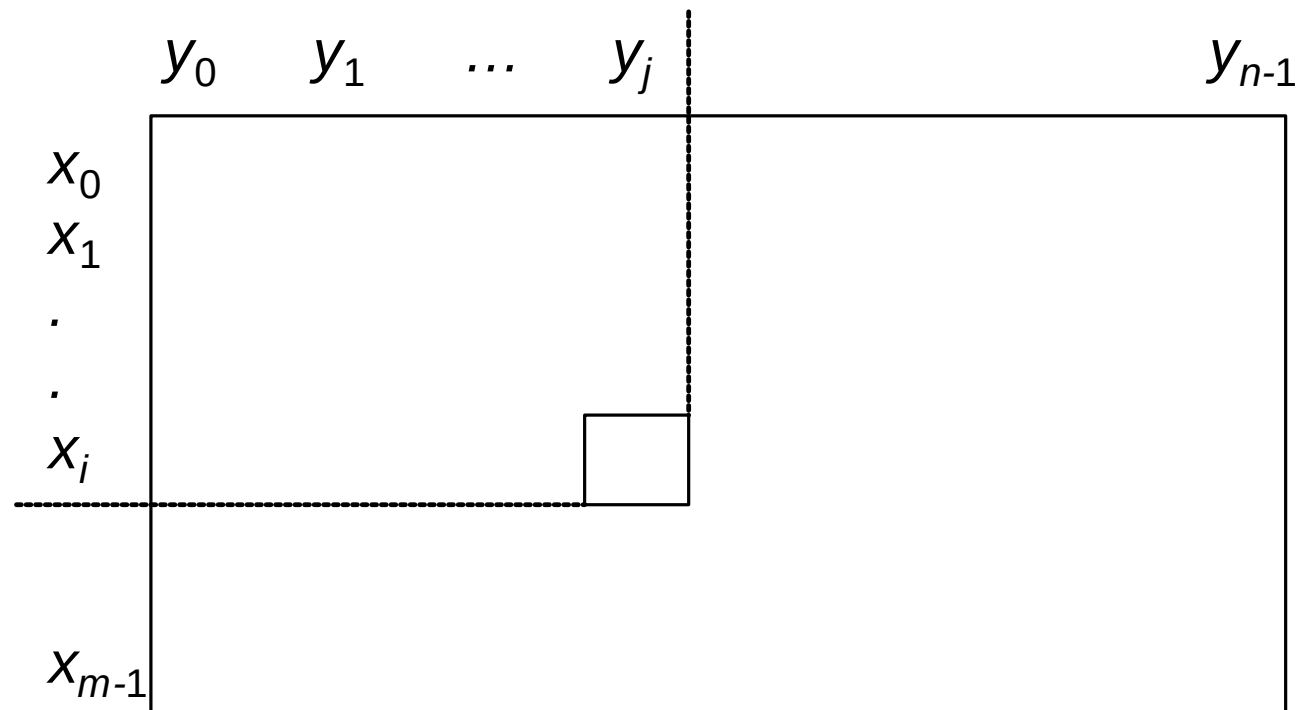
Stupidement exponentiel !

Programmation dynamique

UMLV ©

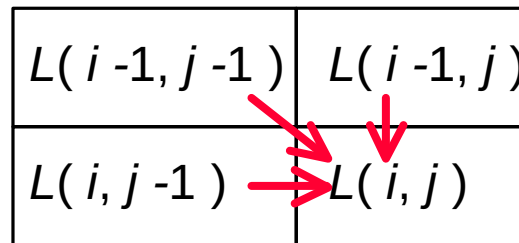
Problème décomposable en nombre fini de sous-problèmes
mais chevauchement des sous-problèmes

Méthode : mémoriser les résultats intermédiaires



$$L(i, j) = \text{PLSC} (x_0 x_1 \dots x_i, y_0 y_1 \dots y_j)$$

$$L(i, j) = \begin{cases} 0 & \text{si } i = -1 \text{ ou } j = -1 \\ L(i-1, j-1) + 1 & \text{sinon si } x_i = y_j \\ \max(L(i, j-1), L(i-1, j)) & \text{sinon} \end{cases}$$



Exemple

UMLV ©

L		-1	0	1	2	3	4	5	6	7	8
			c	b	a	c	b	a	a	b	a
-1		0	0	0	0	0	0	0	0	0	0
0	a	0	0	0	1	1	1	1	1	1	1
1	b	0	0	1	1	1	2	2	2	2	2
2	c	0	1	1	1	2	2	2	2	2	2
3	d	0	1	1	1	2	2	2	2	2	2
4	b	0	1	2	2	2	3	3	3	3	3
5	b	0	1	2	2	2	3	3	3	4	4

PLCS(a b c d b b , c b a c b a a b a) = 4

Algorithme

UMLV ©

```
entier PLSC(mot x, longueur m, mot y, longueur n){  
  pour i  $\leftarrow$  -1 à m-1 faire L[i,0]  $\leftarrow$  0 ;  
  pour j  $\leftarrow$  -1 à n-1 faire L[0,j]  $\leftarrow$  0 ;  
  pour i  $\leftarrow$  0 à m-1 faire  
    pour j  $\leftarrow$  0 à n-1 faire  
      si x[i] = y[j] alors{  
        L[i,j]  $\leftarrow$  L[i-1,j-1]+1 ;  
        P[i,j]  $\leftarrow$  '\' ;  
      }sinon si L[i-1,j]  $\geq$  L[i,j-1] alors{  
        L[i,j]  $\leftarrow$  L[i-1,j] ;  
        P[i,j]  $\leftarrow$  '-' ;  
      }sinon{  
        L[i,j]  $\leftarrow$  L[i,j-1] ;  
        P[i,j]  $\leftarrow$  ' ' ;  
      }  
    retour L[m-1,n-1] ;  
}
```

Exemple (suite)

L		-1	0	1	2	3	4	5	6	7	8
			c	b	a	c	b	a	a	b	a
-1		0	0	0	0	0	0	0	0	0	0
0	a	0	0	0	1	1	1	1	1	1	1
1	b	0	0	1	1	1	2	2	2	2	2
2	c	0	1	1	1	2	2	2	2	2	2
3	d	0	1	1	1	2	2	2	2	2	2
4	b	0	1	2	2	2	3	3	3	3	3
5	b	0	1	2	2	2	3	3	3	4	4

PLCS(**abcdbb** ,
cbacbaaba) = 4

P		-1	0	1	2	3	4	5	6	7	8
			c	b	a	c	b	a	a	b	a
-1		0	0	0	0	0	0	0	0	0	0
0	a	0	-	-	\	⌊	⌊	\	\	⌊	\
1	b	0	-	\	-	-	\	⌊	⌊	\	⌊
2	c	0	\	-	-	\	-	-	-	-	-
3	d	0	-	-	-	-	-	-	-	-	-
4	b	0	-	\	⌊	-	\	-	-	\	⌊
5	b	0	-	\	-	-	\	-	-	\	-

acbb est un plcs

Problème ouvert

calcul de PLSC en temps $< O(n^2 / \log n)$

Variantes

- calcul de PLSC en espace linéaire
[une ligne de L suffit]
- calcul d'un plsc en espace linéaire
[algorithme de Hirschberg]
- utilisation des « dominants »
- algorithme rapide en pratique
bien que temps maximal $= O(n^2 \log n)$
dans certaines implantations de **diff**
[algorithme de Hunt et Szymanski]

Algorithmme de Hunt & Szymanski

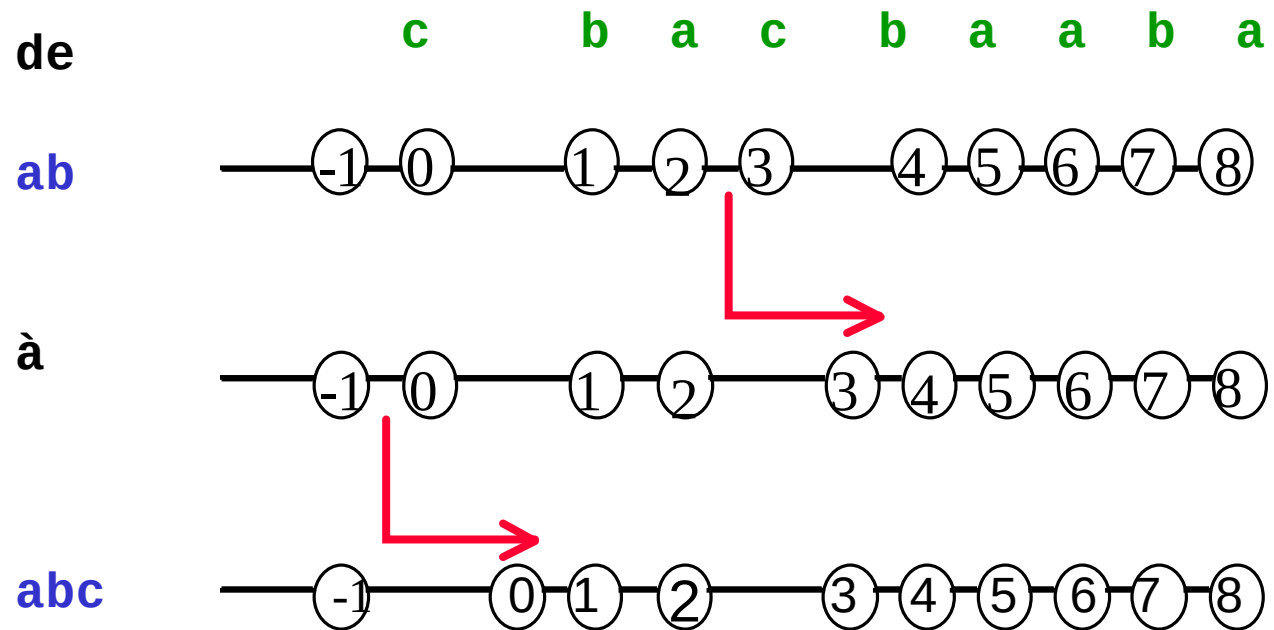
UMLV ©

Calcul de la matrice L par ligne

		c	b	a	c	b	a	a	b	a	
a		-1	0	1	2	3	4	5	6	7	8
		0			1						
b		-1	0	1	2	3	4	5	6	7	8
		0		1			2				
c		-1	0	1	2	3	4	5	6	7	8
		0	1			2					

Comme un boulier

UMLV ©



$$X = X_0X_1 \dots X_{m-1} \quad Y = Y_0Y_1 \dots Y_{n-1}$$

$$\text{Pos}[a] = \{j \mid y_j = a\} \text{ pour } a \in A$$

$$\text{Pour } i \text{ fixé : } J[k] = \{j \mid \text{PLSC}(X_0X_1 \dots X_i, Y_0Y_1 \dots Y_j) = k\}$$

```
entier PLSC(mot x, longueur m, mot y, longueur n){
    J[-1] ← {-1, 0, 1, ..., n-1};
    pour k ← 0 à n-1 faire J[k] ← ∅ ;
    pour i ← 0 à m-1 faire
        pour chaque p ∈ Pos[x[i]] en décroissant faire{
            k ← CLASSE(p);
            si k = CLASSE(p-1) alors{
                (J[k], X) ← PARTAGE(J[k], p);
                J[k+1] ← UNION(X, J[k+1]);
            }
        }
    }
    retour CLASSE(n-1);
}
```


Type abstrait "PARTAGE/FUSION"

Ensemble de base

suites $(E_0, E_1, \dots, E_n)$ d'ensembles disjoints
telles que $E_0 \cup E_1 \cup \dots \cup E_n = \{0, 1, \dots, n\}$

Opérations

Initialisation

CLASSE : élément $\mathbb{R}$ ensemble

$\text{CLASSE}(p) = k$ tel que $p \hat{=} E_k$

PARTAGE : élément $\times$ ensemble $\mathbb{R}$ ensemble $\times$ ensemble

$\text{PARTAGE}(E, p) = (S, T)$

$S = \{q \hat{=} E \mid q < p\}, T = \{q \hat{=} E \mid q \geq p\}$

UNION : ensemble $\times$ ensemble $\mathbb{R}$ ensemble

Implémentations possibles

listes, arbres, ...

Si les E_k sont des intervalles : **B-arbres, 2-3-arbres, ...**

2-3-arbres pour les J_k

initialisation	$O(n)$
CLASSE(p)	$O(\log n)$
PARTAGE(J, p)	$O(\log n)$
UNION(X, J)	$O(\log n)$

Temps : si Pos listes pré-calculées sur y

$$\begin{aligned} & O(\log n \cdot \sum (|\text{Pos}(x_i)| \mid i = 0, 1, \dots, n-1)) \\ &= O(\log n \cdot \text{card} \{ (i, j) \mid x_i = y_j \}) \\ &= O(n \cdot m \cdot \log n) \end{aligned}$$

En pratique, sur fichiers : $O(n \cdot \log n)$

Pré-calcul de Pos

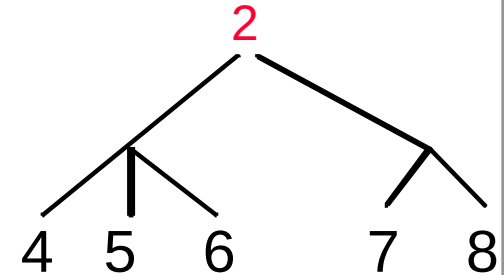
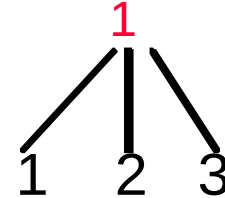
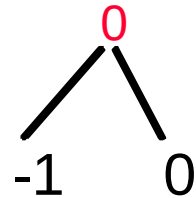
$$|y| = n \text{ } \textcircled{R} \text{ } O(n)$$

si fichiers de n lignes, par hachage $O(n)$ en moyenne

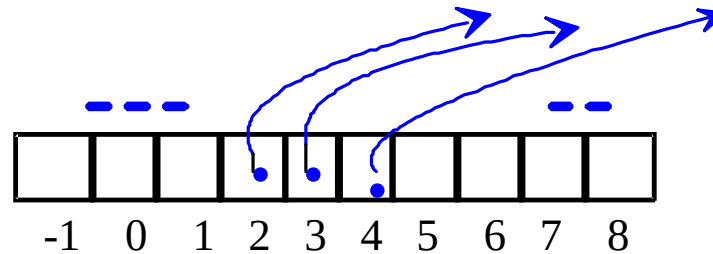
Utilisation des 2-3 arbres

UMLV ©

Numéro de classe
à la racine



Pointeurs
sur feuilles



CLASSE : par liens « père »

PARTAGE : en descendant, duplication, et restructuration

UNION : rattachement au bon niveau, et restructuration

