PLSC

programmation dynamique

Plan

I. Problématique
II. Approche naïve
III. Formulation récursive
IV. Solution par programmation dynamique
V. Exemples de programmation dynamique

Plan

I. Problématique
II. Approche naïve
III. Formulation récursive
IV. Solution par programmation dynamique
V. Exemples de programmation dynamique

I – Problématique

<u>Problématique:</u>

- Étant données deux chaînes de caractères, trouver la plus longue sous-séquence commune.

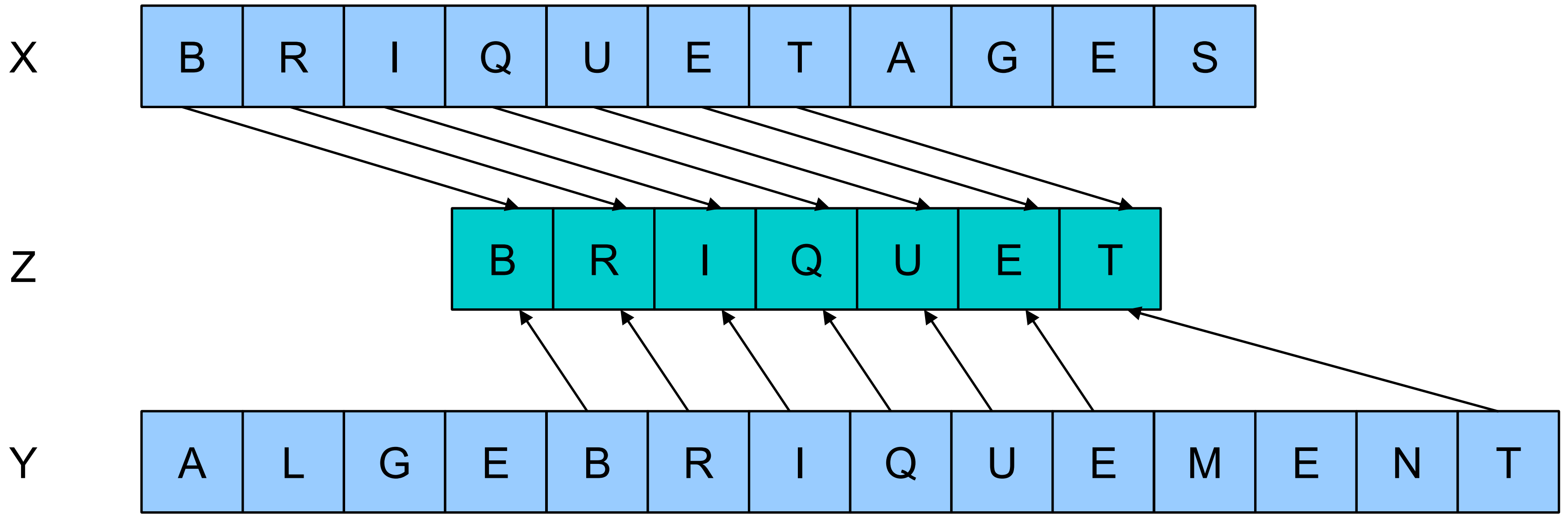
Exemples:

- Comparaison des séquences génétiques ACGT.
- Trouver des modifications dans deux fichiers dans une base de données.

I – Problématique

Approche:

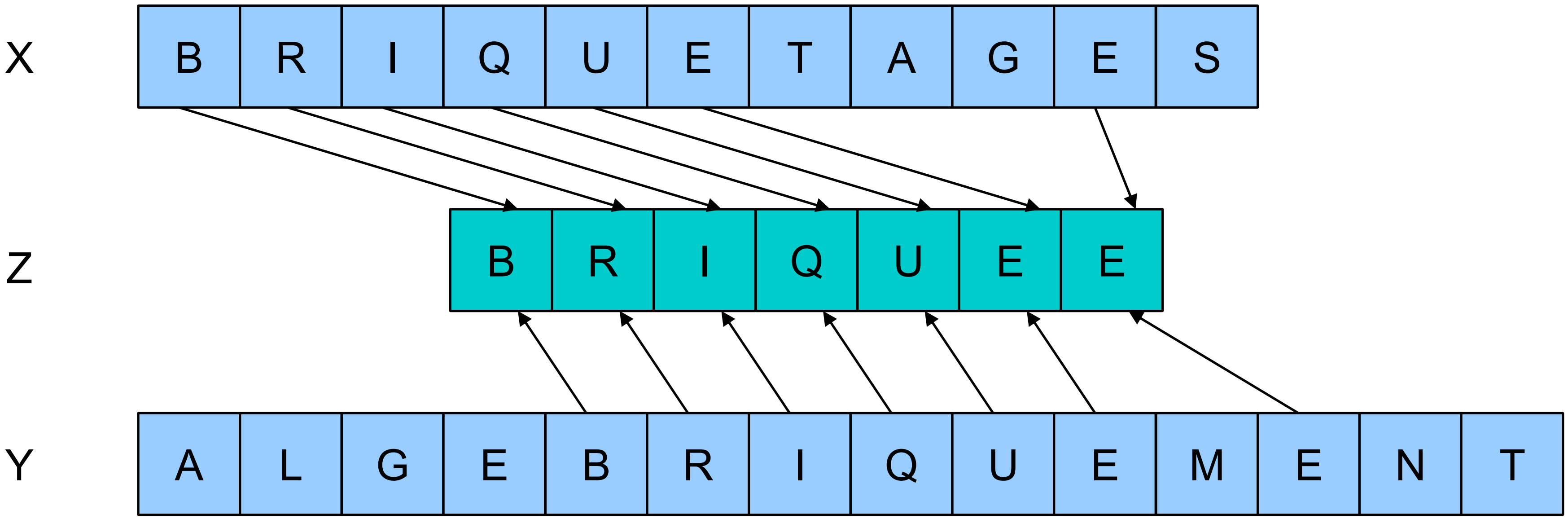
La sous-séquence n'est pas continue dans les chaînes d'entrée



I – Problématique

<u>Approche:</u>

La solution n'est pas unique

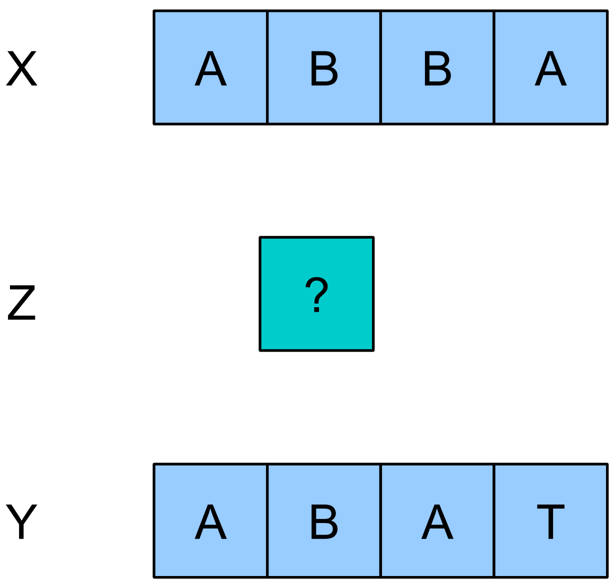


Plan

I. Problématique
II. Approche naïve
III. Formulation récursive
IV. Solution par programmation dynamique
V. Exemples de programmation dynamique

II – Approche naïve

- Il serait possible d'énumérer toutes les sous-séquences de X et Y et de trouver la plus longue sous-séquence commune:



II – Approche naïve

- Exemple:

X

A	B	B	A

Y

A	B	A	T

X:

1110	ABB	0110	BB
1101	ABA	0101	BA
1011	ABA	0011	BA
0111	BBA	1000	A
1100	AB	0100	B
1010	AB	0010	B
1001	AA	0001	A

Y:

1110	ABA	0110	BA
1101	ABT	0101	BT
1011	AAT	0011	AT
0111	BAT	1000	A
1100	AB	0100	B
1010	AA	0010	A
1001	AT	0001	T

II – Approche naïve

- Exemple:

X	A	B	B	A

Y	A	B	A	T

COMPLEXITÉ EXPONENTIELLE

1110	ABB	0110	BB
1101	ABA	0101	BA
1011	ABA	0011	BA
0111	BBA	1000	A
1100	AB	0100	B
1010	AB	0010	B
1001	AA	0001	A

1110	ABA	0110	BA
1101	ABT	0101	BT
1011	AAT	0011	AT
0111	BAT	1000	A
1100	AB	0100	B
1010	AA	0010	A
1001	AT	0001	T

Plan

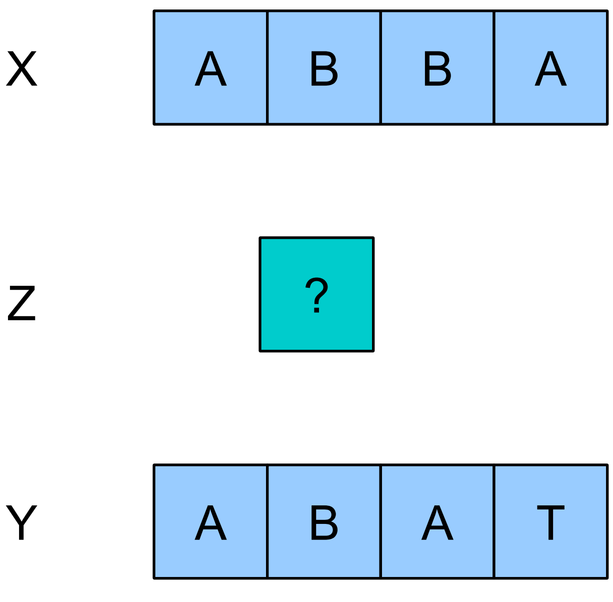
I. Problématique
II. Approche naïve
III. Formulation récursive
IV. Solution par programmation dynamique
V. Exemples de programmation dynamique

III – Formulation récursive

- Pour deux séquences d'entrée X[1..n], Y[1..m], et une PLSC Z[1..k] de X et Y, on note:
  - Si X[n] = Y[m],
    - alors Z[k]=X[n] et Z[1..k-1] est PLSC de X[1..n-1] Y[1..m-1]
  - Si X[n] ≠ Y[m],
    - alors si Z[k]≠X[n], Z[1..k] est PLSC de X[1..n-1] Y[1..m]
  - Si X[n] ≠ Y[m],
    - alors si Z[k]≠Y[m], Z[1..k] est PLSC de X[1..n] Y[1..m-1]

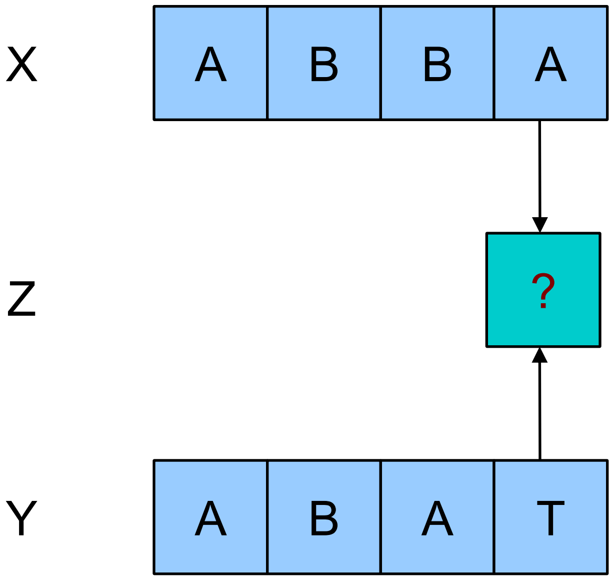
III – Formulation récursive

- Application de l'approche récursive:



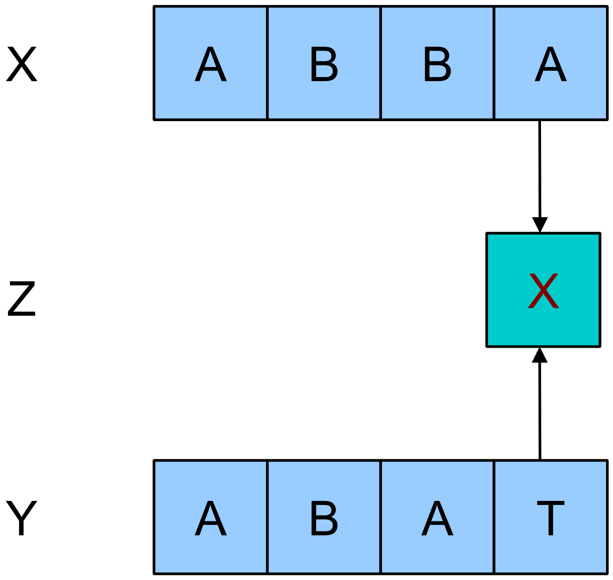
III – Formulation récursive

- Application de l'approche récursive:



III – Formulation récursive

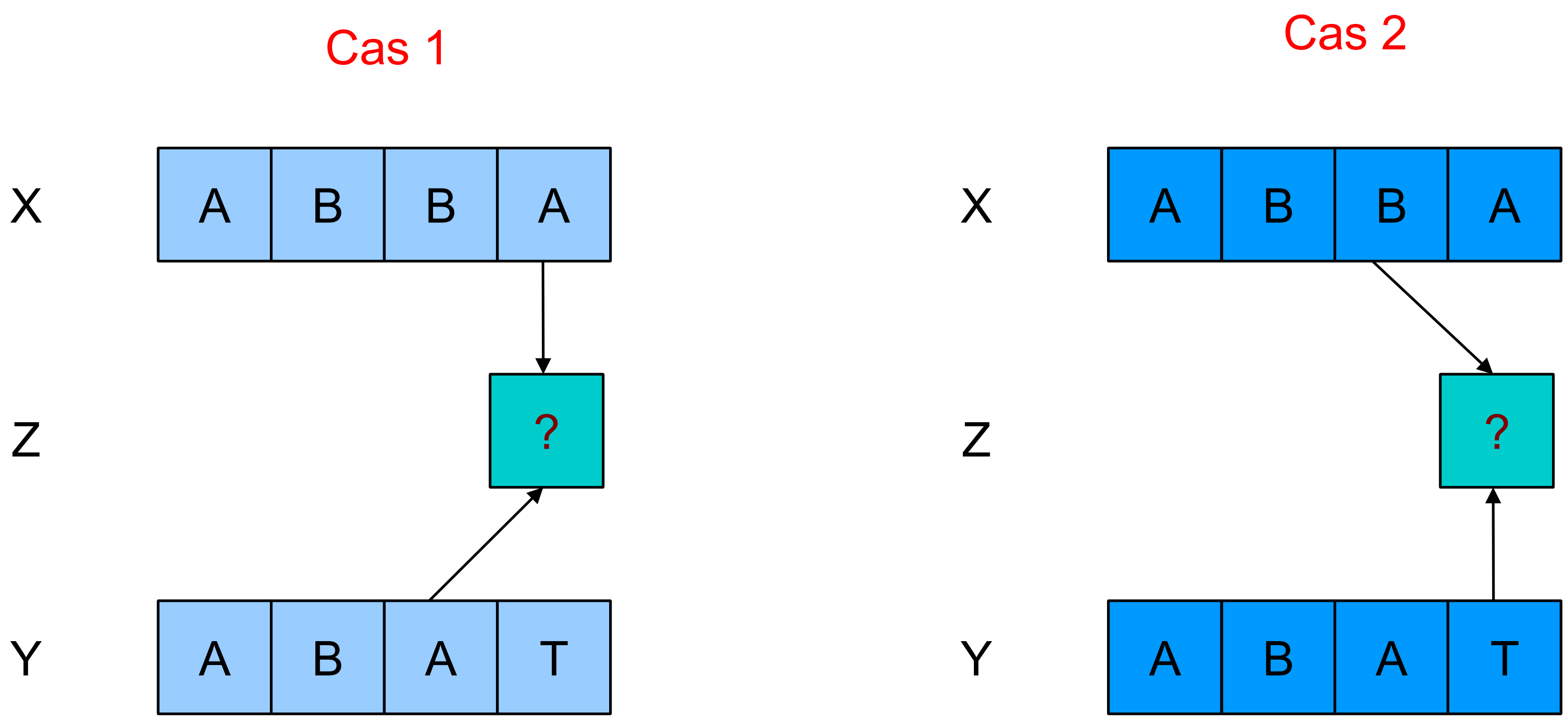
- Application de l'approche récursive:



Deux cas de figures!

III – Formulation récursive

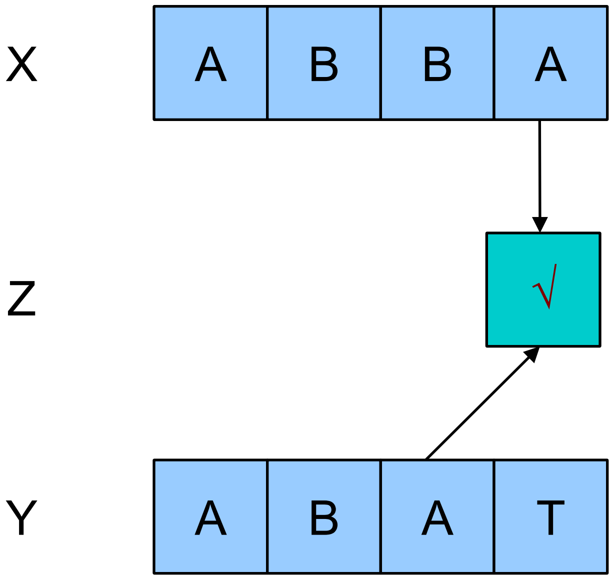
- Application de l'approche récursive:



III – Formulation récursive

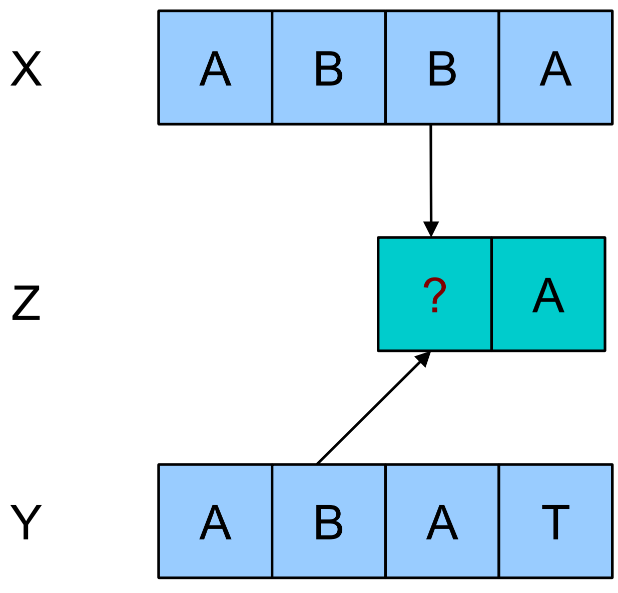
- Application de l'approche récursive:

Cas 1-1



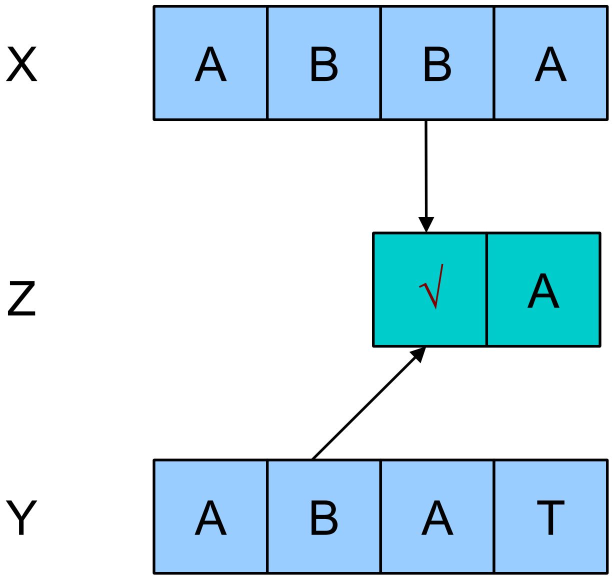
III – Formulation récursive

- Application de l'approche récursive:



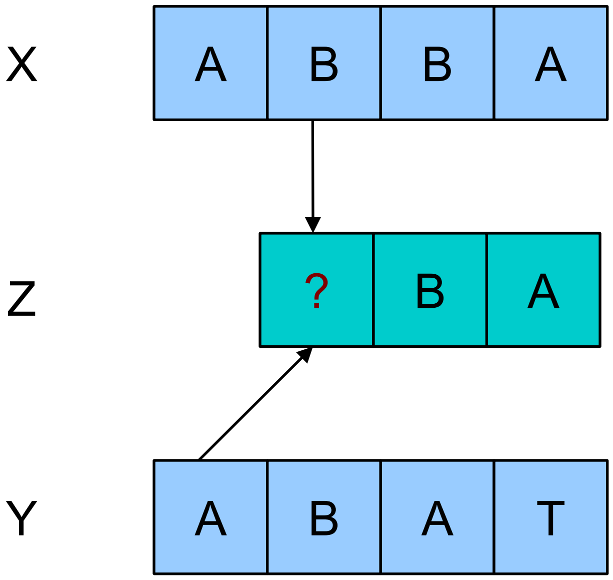
III – Formulation récursive

- Application de l'approche récursive:



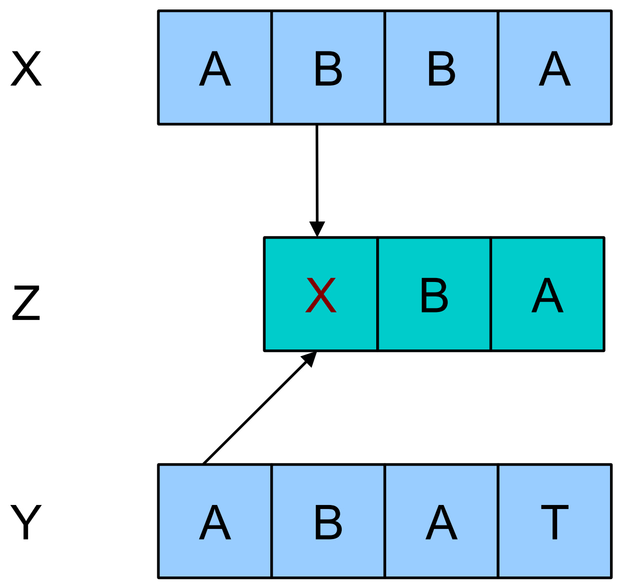
III – Formulation récursive

- Application de l'approche récursive:



III – Formulation récursive

- Application de l'approche récursive:

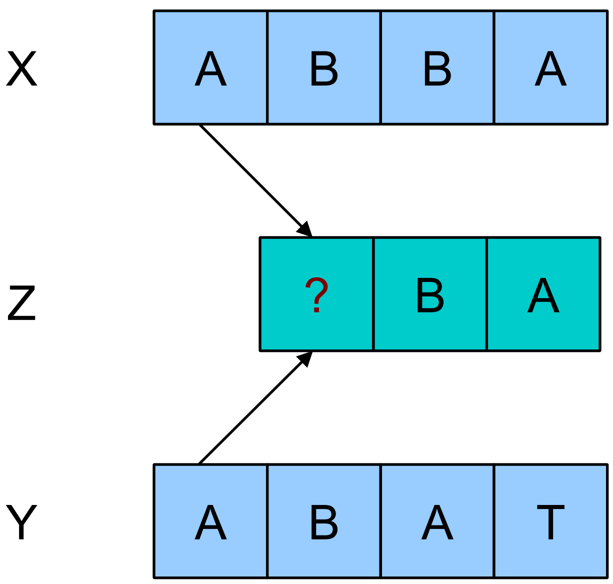


Deux cas de figures!

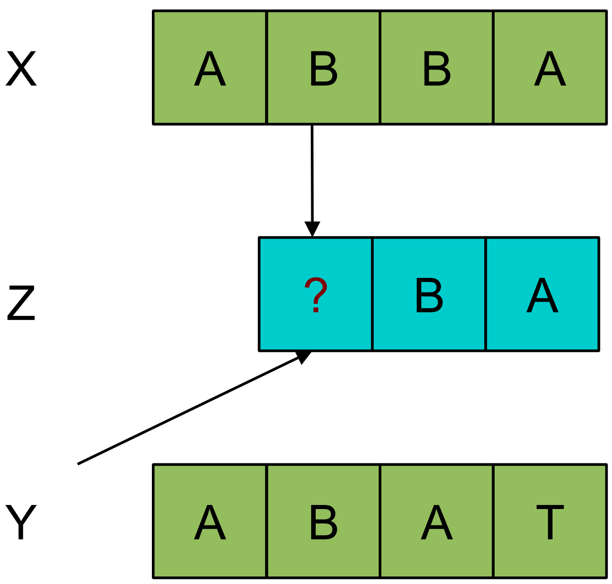
III – Formulation récursive

- Application de l'approche récursive:

Cas 1-1

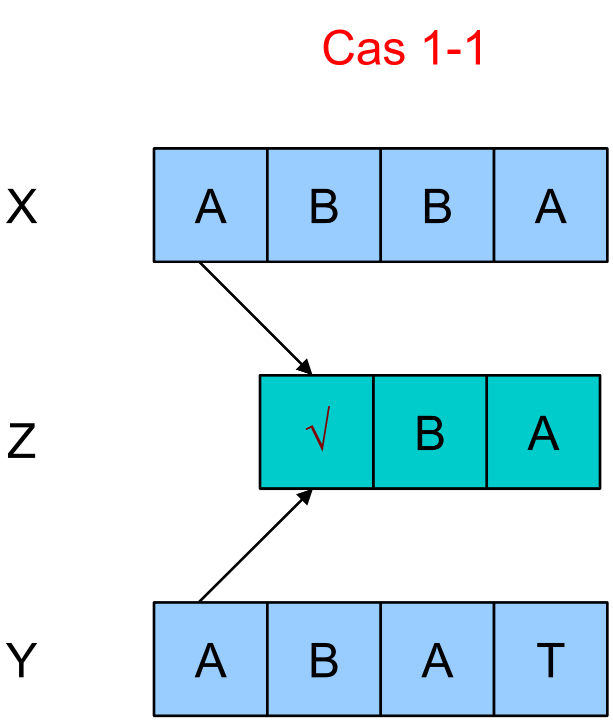


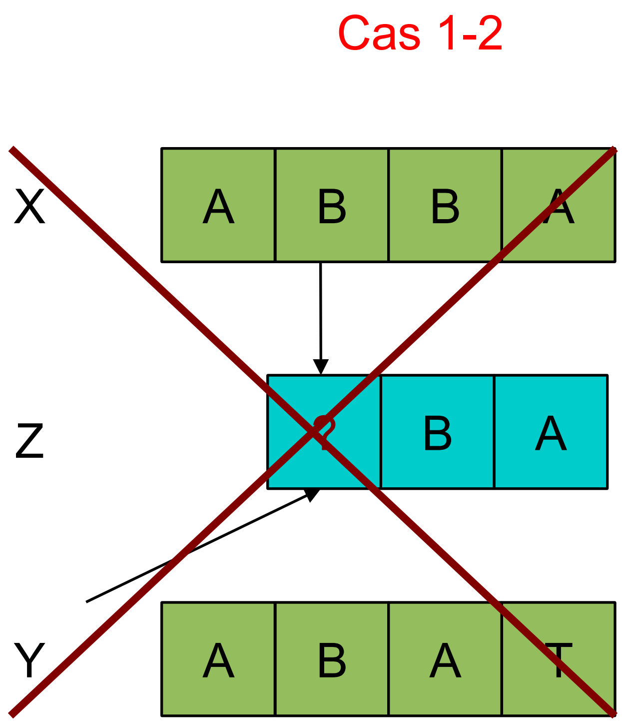
Cas 1-2



III – Formulation récursive

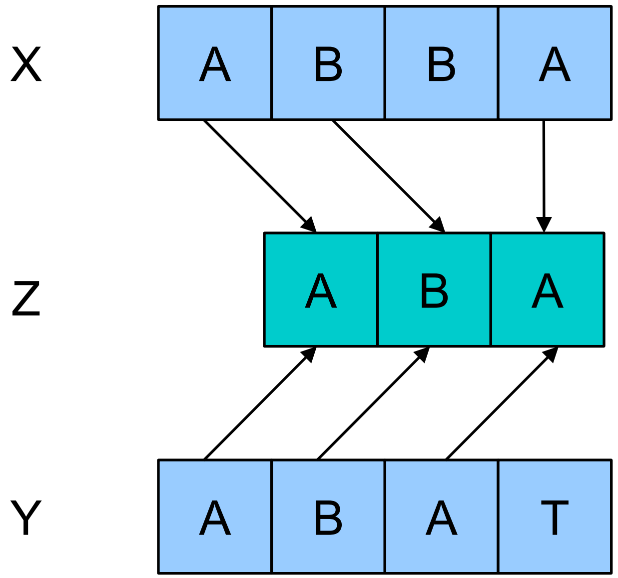
- Application de l'approche récursive:





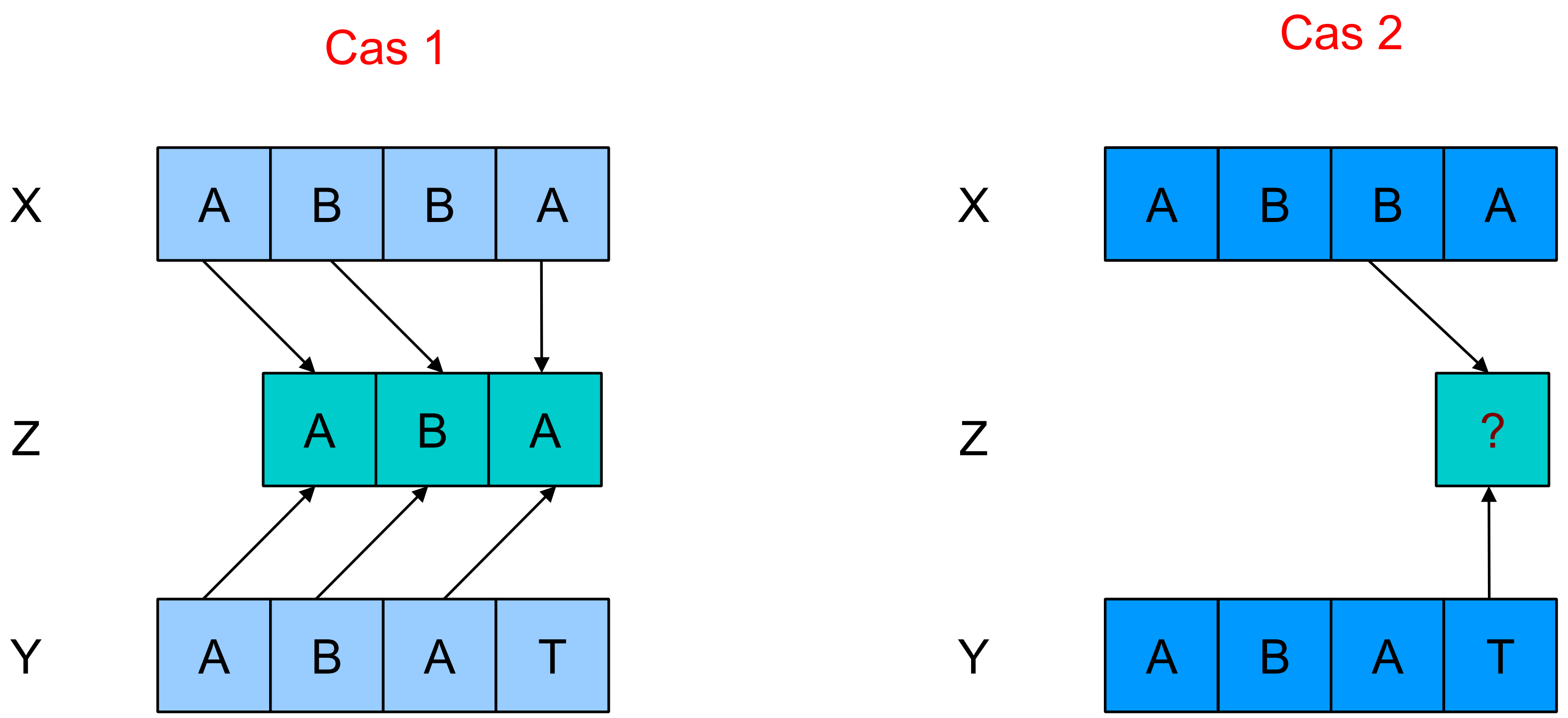
III – Formulation récursive

- Nous avons une première solution de longueur 3:



III – Formulation récursive

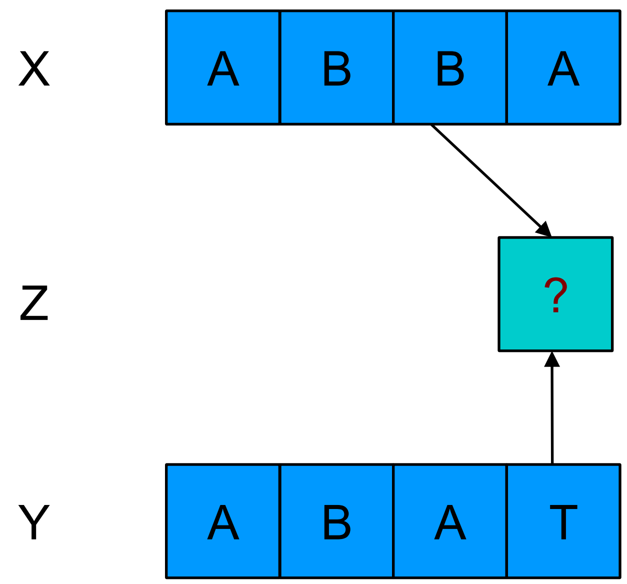
- Longueur minimale : 3



III – Formulation récursive

- Application de l'approche récursive:

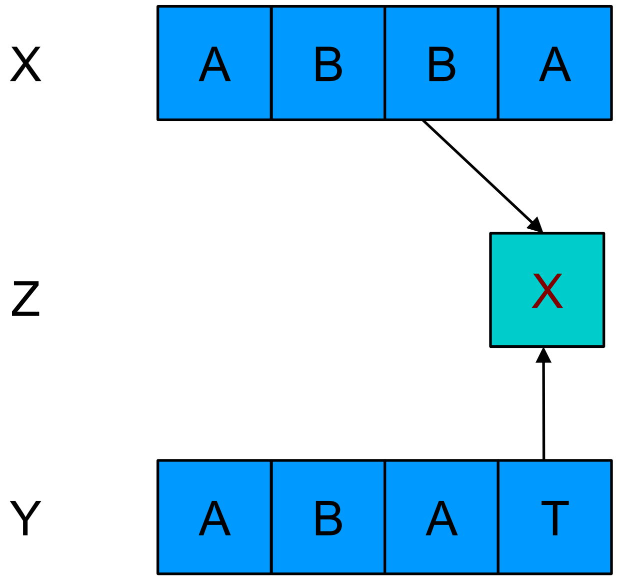
Cas 2



III – Formulation récursive

- Application de l'approche récursive:

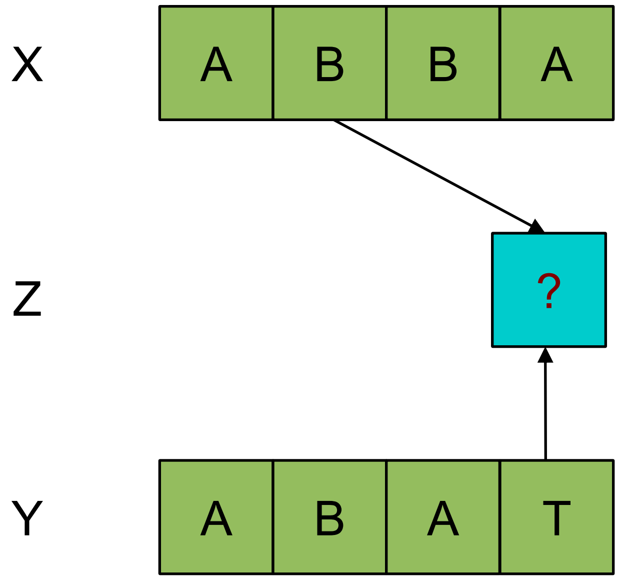
Deux cas de figures!



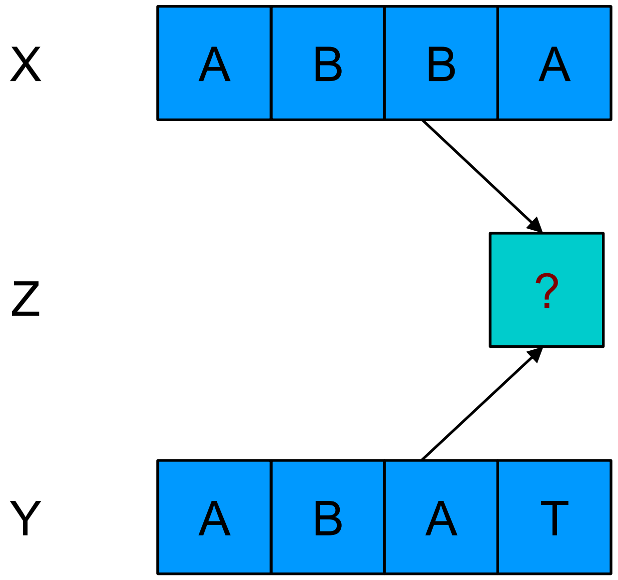
III – Formulation récursive

- Application de l'approche récursive:

Cas 2-1



Cas 2-2

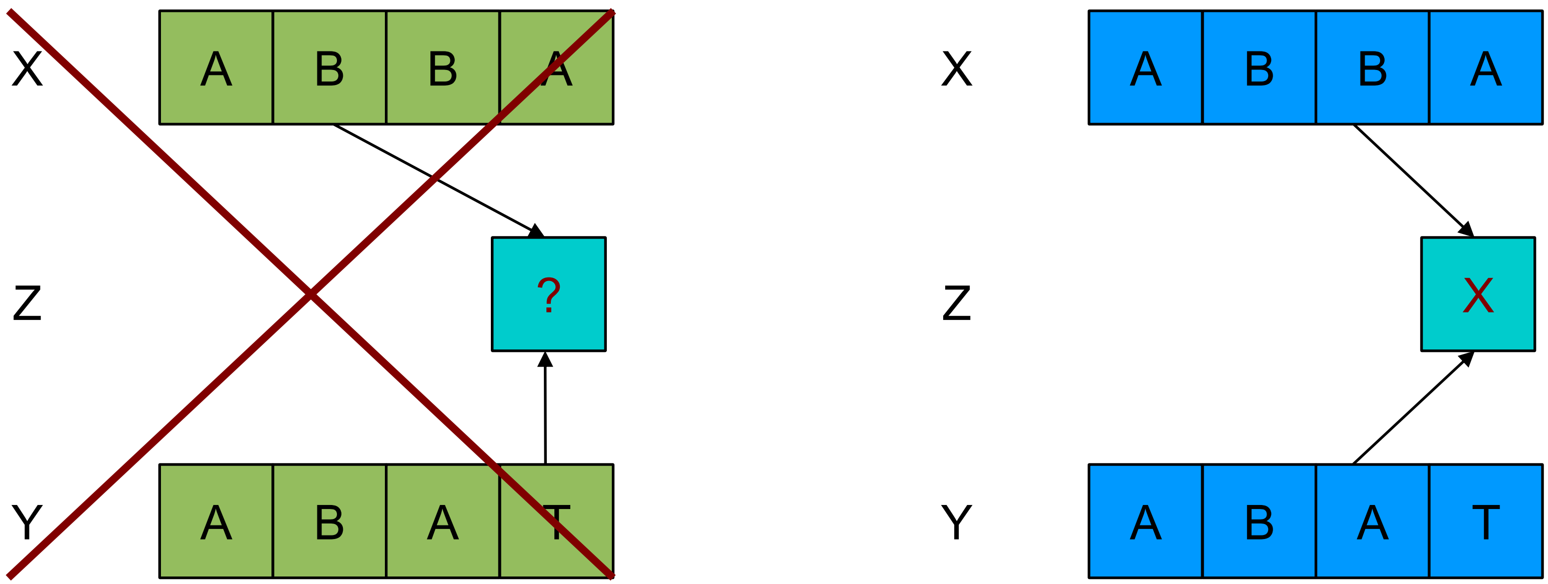


III – Formulation récursive

- Application de l'approche récursive:

LONGUEUR : 2

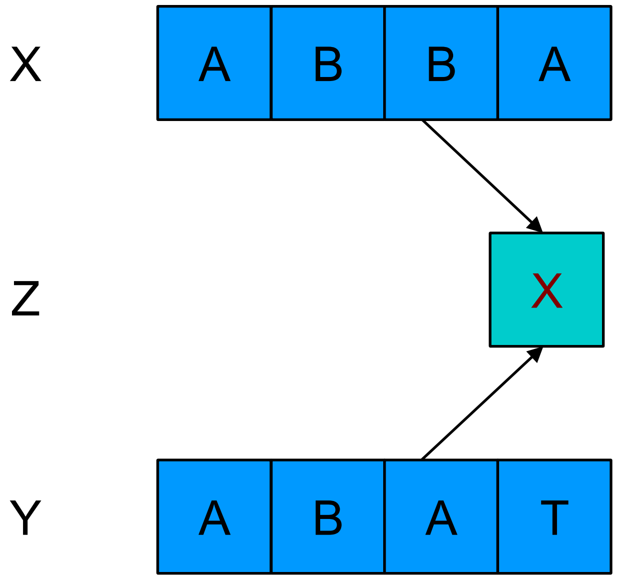
Cas 2-2



III – Formulation récursive

- Application de l'approche récursive:

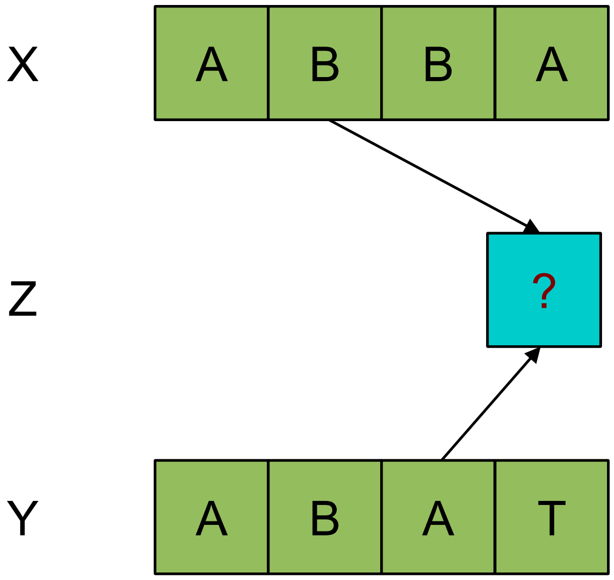
Deux cas de figures!



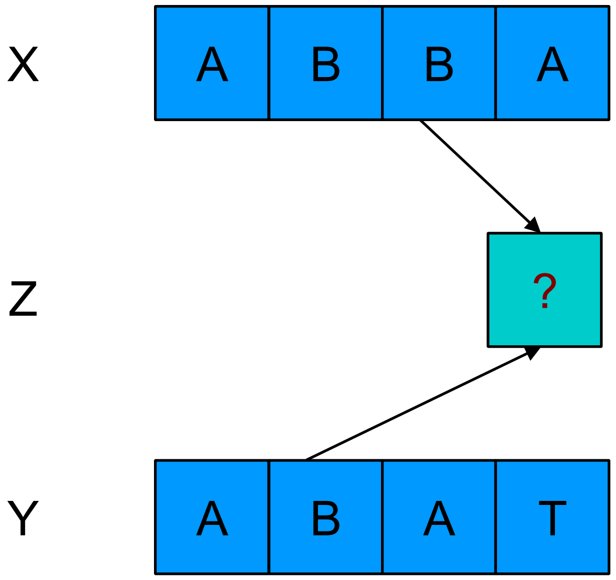
III – Formulation récursive

- Application de l'approche récursive:

Cas 2-2-1

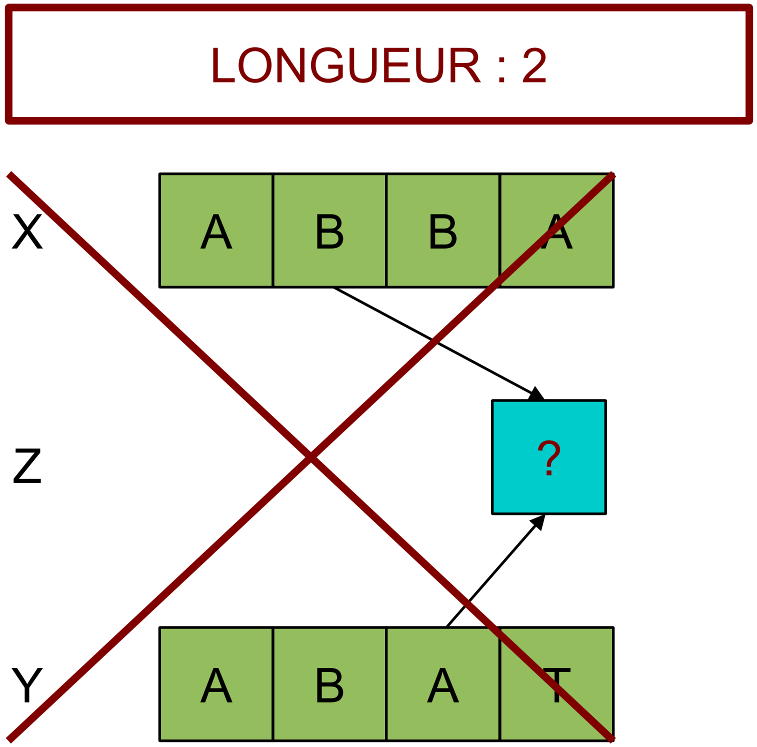


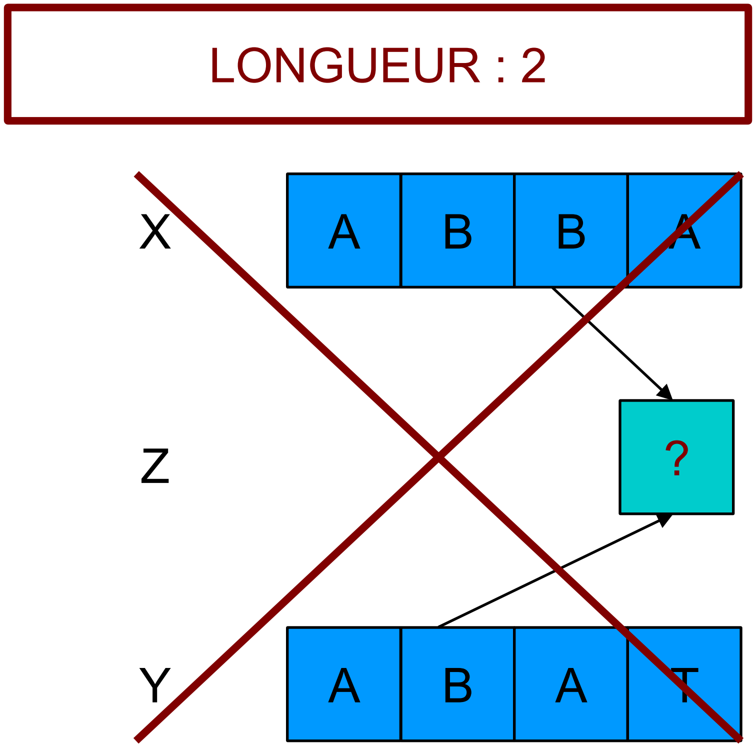
Cas 2-2-2



III – Formulation récursive

- Application de l'approche récursive:





Plan

I. Problématique
II. Approche naïve
III. Formulation récursive
IV. Solution par programmation dynamique
V. Exemples de programmation dynamique

IV – Solution par programmation dynamique

- En reprenant l'expression récursive, il est possible de formuler un algorithme de **programmation dynamique** donnant toutes les solutions possibles:
  - On définit deux tables 2-D auxiliaires *d* (direction) et *t* (taille)
  - *t* a une taille (n+1) x (m+1) (initialisée à 0)
  - *d* a une taille n x m
  - On enregistre dans t[i+1, j+1] la taille de la PLSC de X[1..i], Y[1..j]
  - On enregistre dans d[i, j] la direction (HAUT, GAUCHE, DIAG) vers l'origine pour trouver la PLSC de X[1..i], Y[1..j]

IV – Solution par programmation dynamique

Pour i = 1 : n
    Pour j = 1 : m

Si X[i] == Y[j]

t [i+1, j+1] = *t* [i, j] + 1
d [i, j] = DIAG (↖)

Sinon Si t[i, j+1] ≥ t[i+1, j]

t [i+1, j+1] = *t* [i, j+1]
d [i, j] = HAUT (↑)

Sinon

t [i+1, j+1] = *t* [i+1, j]
d [i, j] = GAUCHE (←)

IV – Solution par programmation dynamique

d	A	B	A	T
A				
B				
B				
A				

t		A	B	A	T
	0	0	0	0	0
A	0				
B	0				
B	0				
A	0				

IV – Solution par programmation dynamique

d	A	B	A	T
A				
B				
B				
A				

t		A	B	A	T
	0	0	0	0	0
A	0				
B	0				
B	0				
A	0				

IV – Solution par programmation dynamique

d	A	B	A	T
A				
B				
B				
A				

t		A	B	A	T
	0	0	0	0	0
A	0	1			
B	0				
B	0				
A	0				

IV – Solution par programmation dynamique

d	A	B	A	T
A				
B				
B				
A				

t		A	B	A	T
	0	0	0	0	0
A	0	1			
B	0				
B	0				
A	0				

IV – Solution par programmation dynamique

d	A	B	A	T
A				
B				
B				
A				

t		A	B	A	T
	0	0	0	0	0
A	0	1			
B	0				
B	0				
A	0				

IV – Solution par programmation dynamique

d	A	B	A	T
A				
B				
B				
A				

t		A	B	A	T
	0	0	0	0	0
A	0	1	1		
B	0				
B	0				
A	0				

IV – Solution par programmation dynamique

d	A	B	A	T
A				
B				
B				
A				

t		A	B	A	T
	0	0	0	0	0
A	0	1	1		
B	0				
B	0				
A	0				

IV – Solution par programmation dynamique

d	A	B	A	T
A				
B				
B				
A				

t		A	B	A	T
	0	0	0	0	0
A	0	1	1	1	
B	0				
B	0				
A	0				

IV – Solution par programmation dynamique

d	A	B	A	T
A	↖	←	↖	
B				
B				
A				

t		A	B	A	T
	0	0	0	0	0
A	0	1	1	1	
B	0				
B	0				
A	0				

IV – Solution par programmation dynamique

d	A	B	A	T
A				
B				
B				
A				

t		A	B	A	T
	0	0	0	0	0
A	0	1	1	1	
B	0				
B	0				
A	0				

IV – Solution par programmation dynamique

d	A	B	A	T
A	↖	←	↖	←
B				
B				
A				

t		A	B	A	T
	0	0	0	0	0
A	0	1	1	1	1
B	0				
B	0				
A	0				

IV – Solution par programmation dynamique

d	A	B	A	T
A	↖	←	↖	←
B	↑	↖	←	←
B				
A				

t		A	B	A	T
	0	0	0	0	0
A	0	1	1	1	1
B	0	1	2	2	2
B	0				
A	0				

IV – Solution par programmation dynamique

d	A	B	A	T
A	↖	←	↖	←
B	↑	↖	←	←
B	↑	↖	↑	↑
A				

t		A	B	A	T
	0	0	0	0	0
A	0	1	1	1	1
B	0	1	2	2	2
B	0	1	2	2	2
A	0				

IV – Solution par programmation dynamique

d	A	B	A	T
A	↖	←	↖	←
B	↑	↖	←	←
B	↑	↖	↑	↑
A	↖	↑	↖	←

t		A	B	A	T
	0	0	0	0	0
A	0	1	1	1	1
B	0	1	2	2	2
B	0	1	2	2	2
A	0	1	2	3	3

IV – Solution par programmation dynamique

d	A	B	A	T
A	↖	←	↖	←
B	↑	↖	←	←
B	↑	↖	↑	↑
A	↖	↑	↖	←

t		A	B	A	T
	0	0	0	0	0
A	0	1	1	1	1
B	0	1	2	2	2
B	0	1	2	2	2
A	0	1	2	3	3

En suivant la direction des flèches, on peut retrouver la PLSC

IV – Solution par programmation dynamique

d	A	B	A	T
A	↖	←	↖	←
B	↑	↖	←	←
B	↑	↖	↑	↑
A	↖	↑	↖	←

t		A	B	A	T
	0	0	0	0	0
A	0	1	1	1	1
B	0	1	2	2	2
B	0	1	2	2	2
A	0	1	2	3	3

Les caractères de la PLSC sont aux flèches diagonales

IV – Solution par programmation dynamique

d	A	B	A	T
A	↖	←	↖	←
B	↑	↖	←	←
B	↑	↖	↑	↑
A	↖	↑	↖	←

t		A	B	A	T
	0	0	0	0	0
A	0	1	1	1	1
B	0	1	2	2	2
B	0	1	2	2	2
A	0	1	2	3	3

On pourrait minimiser la mémoire de t à deux lignes

IV – Solution par programmation dynamique

- Appliquer l'algorithme pour l'exemple suivant:

X	O	V	A	L	E	S	
Z		?					
Y	V	A	L	I	S	E	S

Plan

I. Problématique
II. Approche naïve
III. Formulation récursive
IV. Solution par programmation dynamique
V. Exemples de programmation dynamique

V – Exemple de programmation dynamique

- La programmation dynamique est une méthode de résolution de problème dite bottom-up (par opposition aux méthodes dites top-down comme diviser pour régner).

- On trouve donc des solutions atomiques (optimales) aux sous-problèmes qui l'on recompose ensemble pour former la solution globale au problème considéré.

V – Exemple de programmation dynamique

Exemple 1:

- Considérons le cas des nombres de Fibonacci:

$$\begin{cases} F_n = F_{n-1} + F_{n-2} \\ F_0 = 1 \\ F_1 = 1 \end{cases}$$

V – Exemple de programmation dynamique

- Formulation naïve

```
Fibonacci( n : positive integer )

    If   n = 0
        return  0
    Elseif   n = 1
        return   1
    Else
        return   Fibonacci( n−1 ) + Fibonacci( n−2 )
```

V – Exemple de programmation dynamique

- Formulation naïve

Fibonacci(n : positive integer)

If n = 0
  return 0
Elseif n = 1
  return 1
Else
  return **Fibonacci**(n−1) + **Fibonacci**(n−2)

COMPLEXITÉ EXPONENTIELLE:

Il suffit pour s'en convaincre de dessiner l'arbre équivalent aux appels récursifs et de relever la redondance

V – Exemple de programmation dynamique

- Solution en programmation dynamique:

```
Fibonacci( n )

    T[0..n], T[0] = 1, T[1] = 1

    For i in 2 to n
        T[ i ] = T[ i-1 ] + T[ i-2 ]

    return  T[ n ]
```

V – Exemple de programmation dynamique

Programmation dynamique ne nécessite pas de tableau (le nom est antérieur à la programmation):

```
Fibonacci( n )

    p = 1, c = 1

    if n = 0
        return   0
    elseif n = 1
        return   1
    else
        for i = 2 à n
            t = p + c, p = c, c = t

    return c
```

V – Exemple de programmation dynamique

Notons finalement qu'il existe une solution top-down:

Soit
M: une mappe accessible qui prend en clé i et associe une valeur f (i,f)

Initialisation
Insérer les paires (0,1) et (1,1)

Fibonacci(n)

```
if ~M.contains( n )
    M.insert(  (n, Fibonacci( n-1 ) + Fibonacci( n-2 ) ) )

return  M.get( n )
```

V – Exemple de programmation dynamique

Exemple 2:

- Considérons le problème de multiplication de matrices:

On désire multiplier *n* matrices en minimisant le coût des opérations

$A_1\ A_2\ A_3\ A_4\ \ldots\ A_n$

Voir document en ligne sur le site Moodle

V – Exemple de programmation dynamique

Rappel:

Multiplier les matrices A (p x q) et B (q x r) :

for i **in** 1 **to** p

  for j **in** 1 **to** r

    C[i, j] = 0

    for k **in** 1 **to** q

      C[i, j] += A[i,k] x B[k, j]

est dominé par p x q x r multiplications

V – Exemple de programmation dynamique

<u>Problème d'optimisation:</u>

Quelle séquence de multiplication utiliser pour réduire le coût?

<u>Exemple:</u>

A (10000 x 1) B (1x 10000) C (10000 x 10)

D = A x B x C , où D (10000 x 10)

D = (A x B) x C donne $10^8 + 10^9$ multiplications

D = A x (B x C) donne $10^5 + 10^5$ multiplications

V – Exemple de programmation dynamique

Exemple 2: Énoncé

- On désire multiplier *n* matrices en réduisant au minimum le coût des opérations

$A_1\, A_2\, A_3\, A_4\, \ldots\, A_n$

- La matrice A_i a une dimension p_{i-1} x p_i

V – Exemple de programmation dynamique

- Formulation naïve: énumérer toutes les parenthèses possibles

P(n) le nombre de parenthésages possibles d'une séquence de n matrices

$$P(1) = 1$$

$$P(n) = \sum_{k=1}^{n-1} P(k)P(n-k), \quad n \geq 2$$

V – Exemple de programmation dynamique

- On peut <u>facilement</u> montrer que

  $P(\, n \,) \geq 2^n$

  COMPLEXITÉ EXPONENTIELLE

  pour n suffisamment grand

V – Exemple de programmation dynamique

- Sous-structure optimale:

$$A_i \ldots A_j = (A_i \ldots A_k)(A_{k+1} \ldots A_j)$$

- Si le coût total du produit de i à j est optimal, alors les sous-produits i à k et k+1 à j sont optimaux

V – Exemple de programmation dynamique

- Coût minimal:

  m[i, j] le coût minimal du produit des matrices A_i à A_j
  m[1, *n*] est le plus bas coût de la multiplication totale

- Le coût ici réfère au nombre de multiplications effectuées

V – Exemple de programmation dynamique

- Coût minimal, définition récursive:

  m[i, i] est nul

  $m[i, j] = m[i, k] + m[k+1, j] + p_{i-1}p_kp_j$

- Or k n'est pas connu...

V – Exemple de programmation dynamique

- Coût minimal, algorithme récursif:

  m[i, i] est nul

  Pour k=i à j-1

  Trouver m[i, j] = min m[i, k] + m[k+1, j] + $p_{i-1}p_kp_j$

V – Exemple de programmation dynamique

<u>Exemple</u>

A_1 (2 x 4) A_2 (4 x 3) A_3 (3 x 6) A_4 (6 x 2)

Temps d'exécution exponentiel

Recouvrement des sous problèmes

V – Exemple de programmation dynamique

- Solution en programmation dynamique:
  - Une table m[1..n, 1..n] pour les coûts
  - Une table s[1..n, 1..n] pour retenir les k de m[i, j]

V – Exemple de programmation dynamique

Solution en programmation dynamique:

Voir document en ligne sur le site Moodle

```
OrdreMin( p: n+1 tailles )

| for i in 1 to n
|     m[ i, i ] = 0
|
| for v in 2 to n
|   | for i = 1 to n - v + 1
|   |
|   |     j = i + v - 1
|   |     m[i, j] = MAX_INT
|   |
|   |   | for k in i to j-1
|   |   |
|   |   |   | q = m[i,k] + m[k+1,j] + p(i-1) * p( k ) * p( j)
|   |   |   |
|   |   |   | if q < m[i, j]
|   |   |   |       m[i, j] = q
|   |   |   |       s[i, j] = k
```

V – Exemple de programmation dynamique

A_1 (2 x 4) A_2 (4 x 3) A_3 (3 x 6) A_4 (6 x 2)

m	1	2	3	4
1	0	24	60	72
2	-	0	72	60
3	-	-	0	36
4	-	-	-	0

s	1	2	3	4
1	-	1	2	2
2	-	-	2	2
3	-	-	-	3
4	-	-	-	-

V – Exemple de programmation dynamique

A_1 (2 x 4) A_2 (4 x 3) A_3 (3 x 6) A_4 (6 x 2)

m	1	2	3	4
1	0	24	60	72
2	-	0	72	60
3	-	-	0	36
4	-	-	-	0

s	1	2	3	4
1	-	1	2	2
2	-	-	2	2
3	-	-	-	3
4	-	-	-	-

$(A_1 A_2)(A_3 A_4)$