
INF2010 – ASD

Algorithmes sur les chaînes de caractères

Plan

Recherche de patron

- Problématique

- Rabin Karp

- Automate FSM

PLSC

- Problématique

- Solution par programmation dynamique

Plan

Recherche de patron

Problématique

Rabin Karp

Automate FSM

PLSC

Problématique

Solution par programmation dynamique

Problématique

Problématique:

Chercher la chaîne de caractères $P[1..m]$ dans un texte $T[1..n]$.
où $m \leq n$.

Ayant $m \leq n$, on traduit le problème par chercher tous les
 $s \leq n-m+1$ pour lesquels $T[s+1...s+m] = P[1..m]$

Exemples:

- Chercher un mot dans un fichier
- Chercher un fichier dans un volume (HDD, Clé Flash, CD-ROM)
- Chercher un mot dans des fichiers
- Chercher un mot dans le web (google, yahoo)

Algorithme naïf

```
Pour s=0 à n-m  
    Pour j= 1 à m  
        Si T[s+j] != P[j]  
            Reprendre au s suivant  
        Sinon Si j=m  
            Inclure s dans S  
Retourner S
```

Algorithme naïf

T

b

e

e

b

b

e

b

e

b

e

e

b

P

b

e

b

e

1

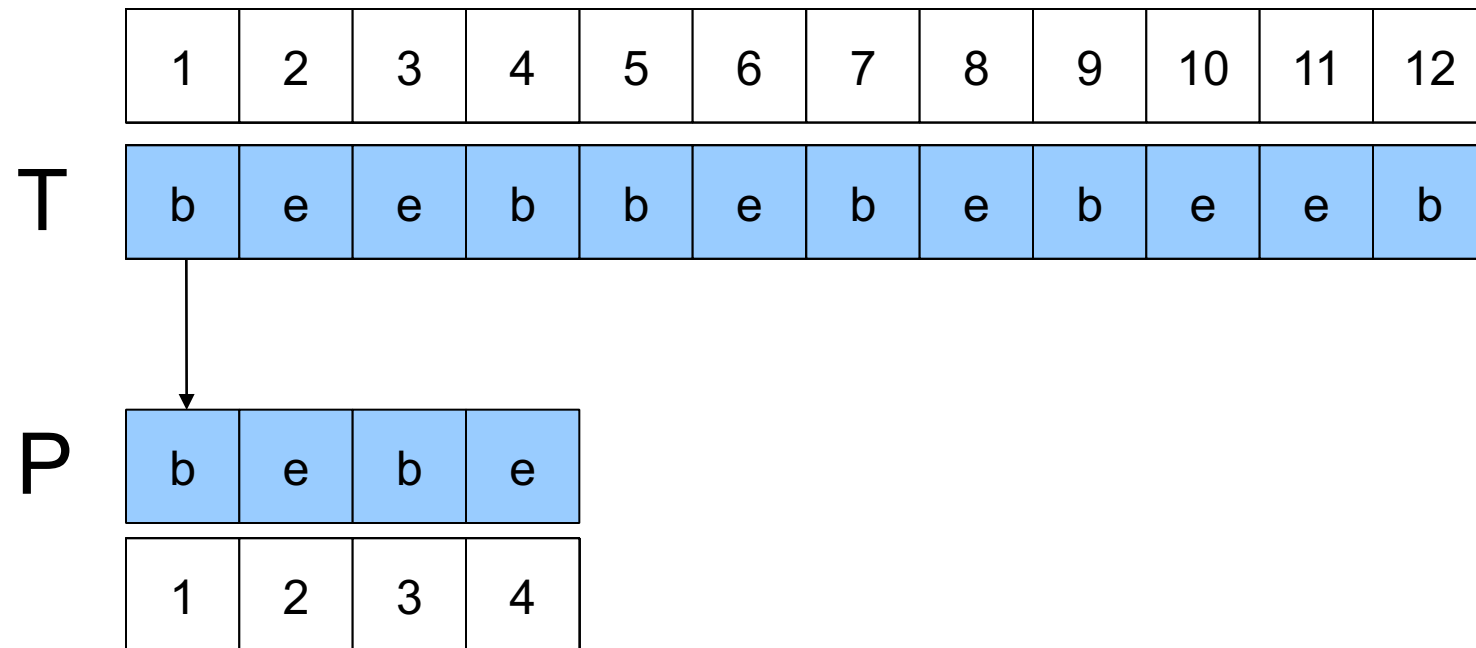
2

3

4

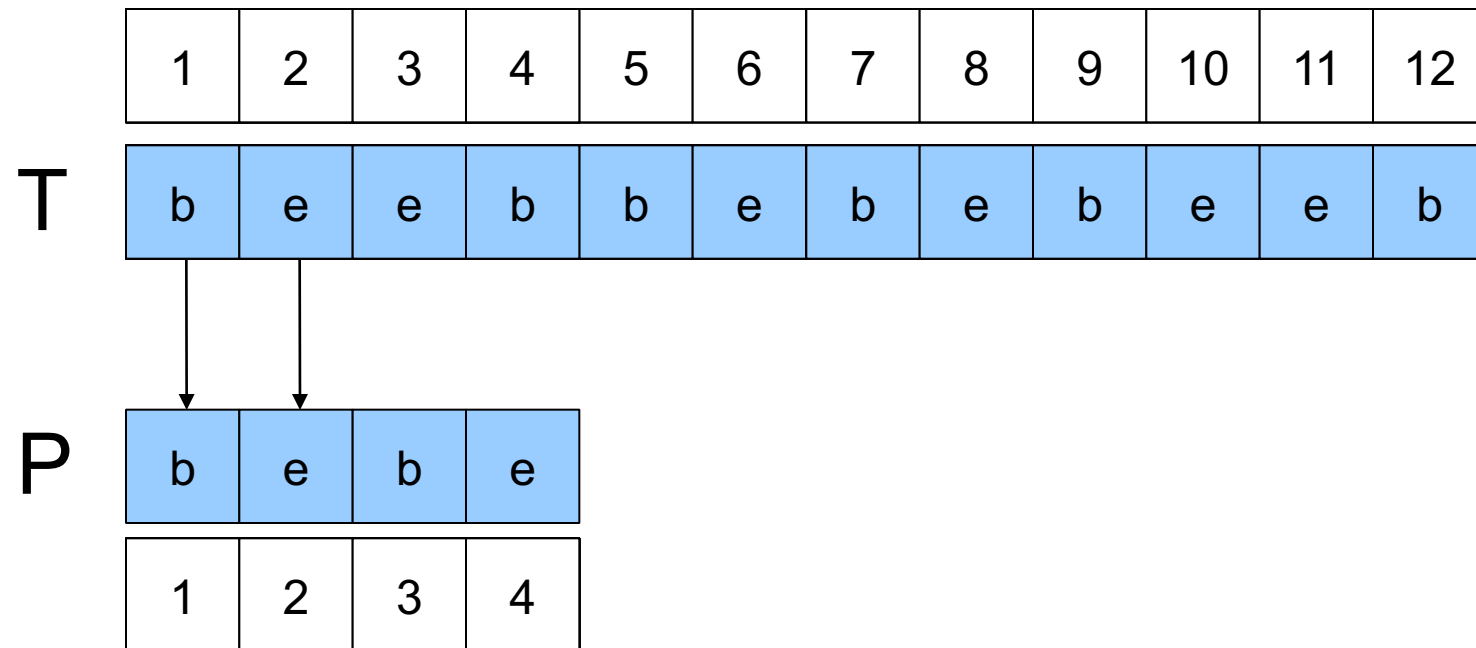
Algorithme naïf

$s=0, j=1$



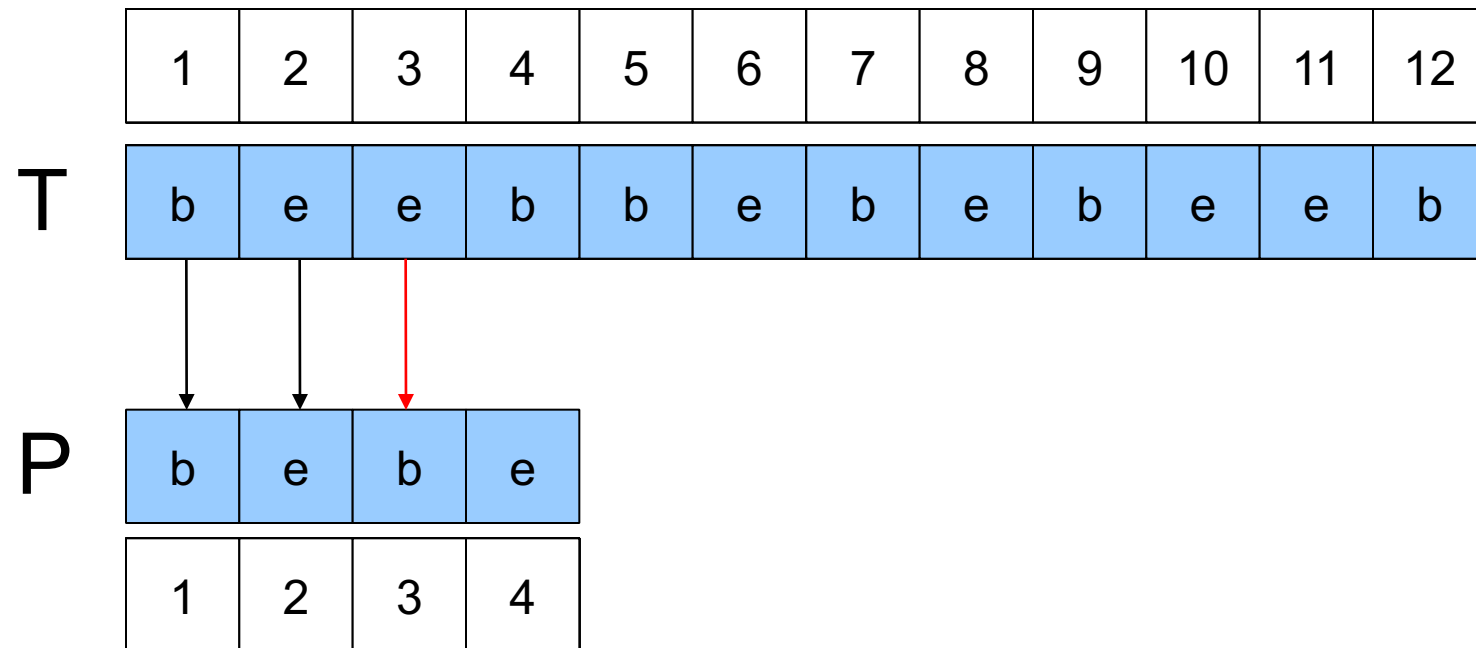
Algorithme naïf

$s=0, j=2$



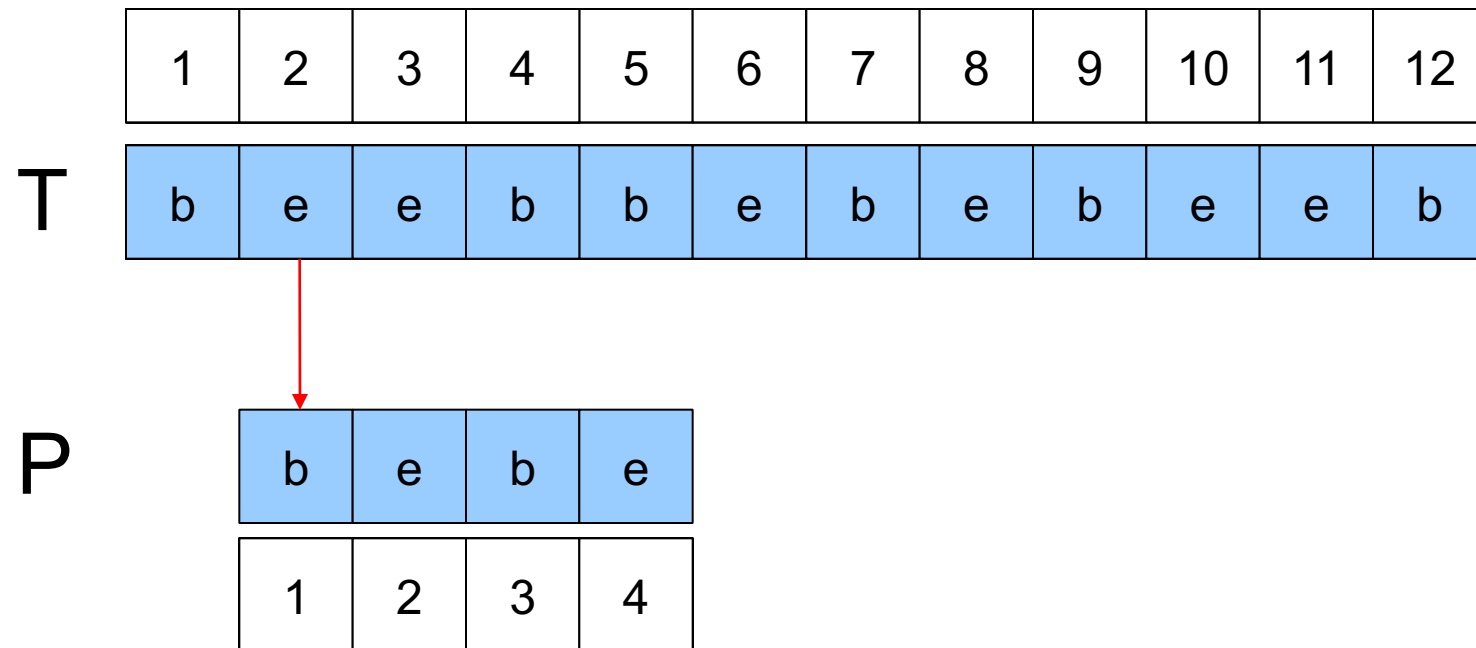
Algorithme naïf

$s=0, j=3$



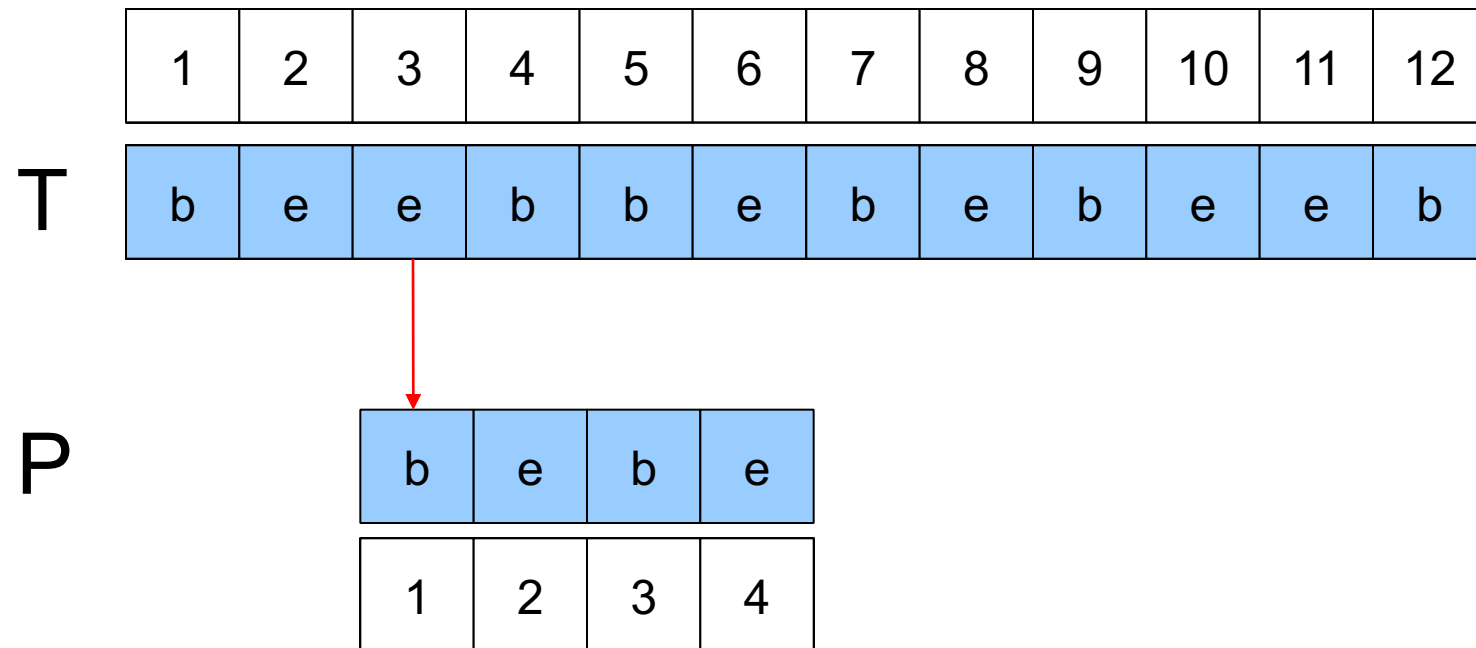
Algorithme naïf

$s=1, j=1$



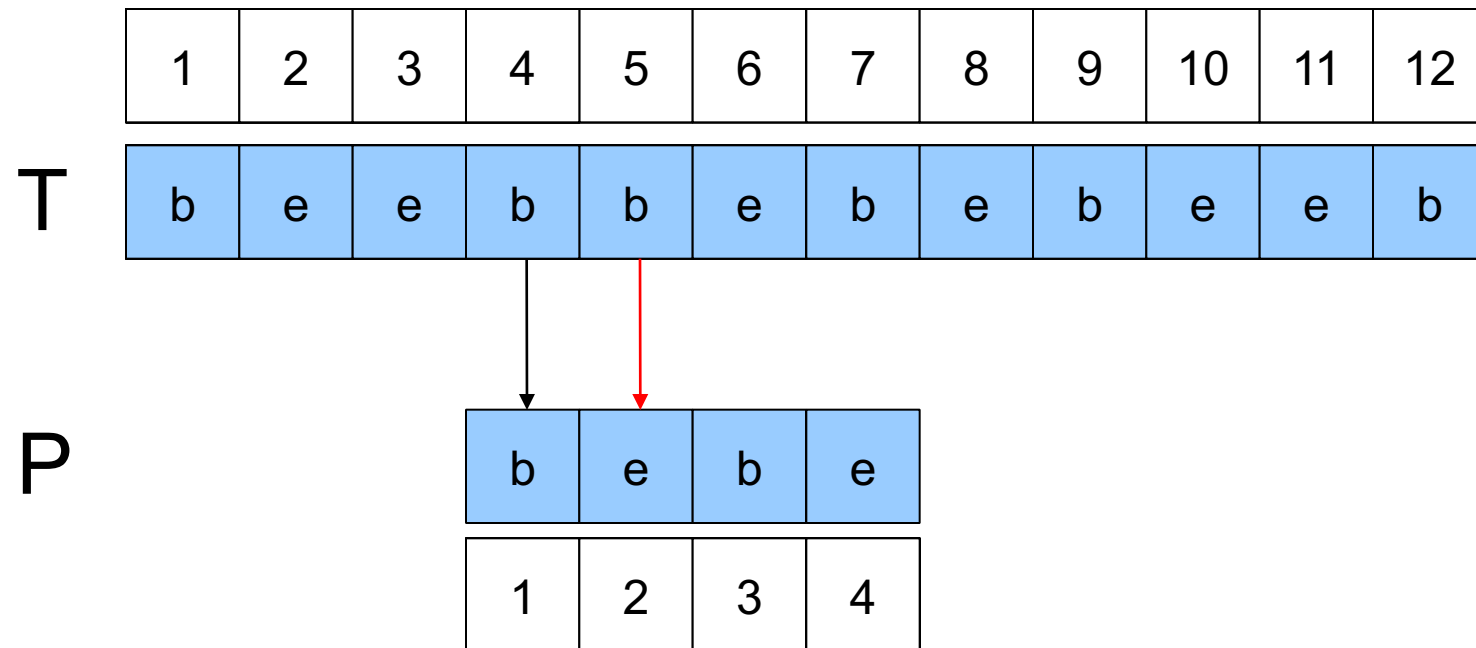
Algorithme naïf

$s=2, j=1$



Algorithme naïf

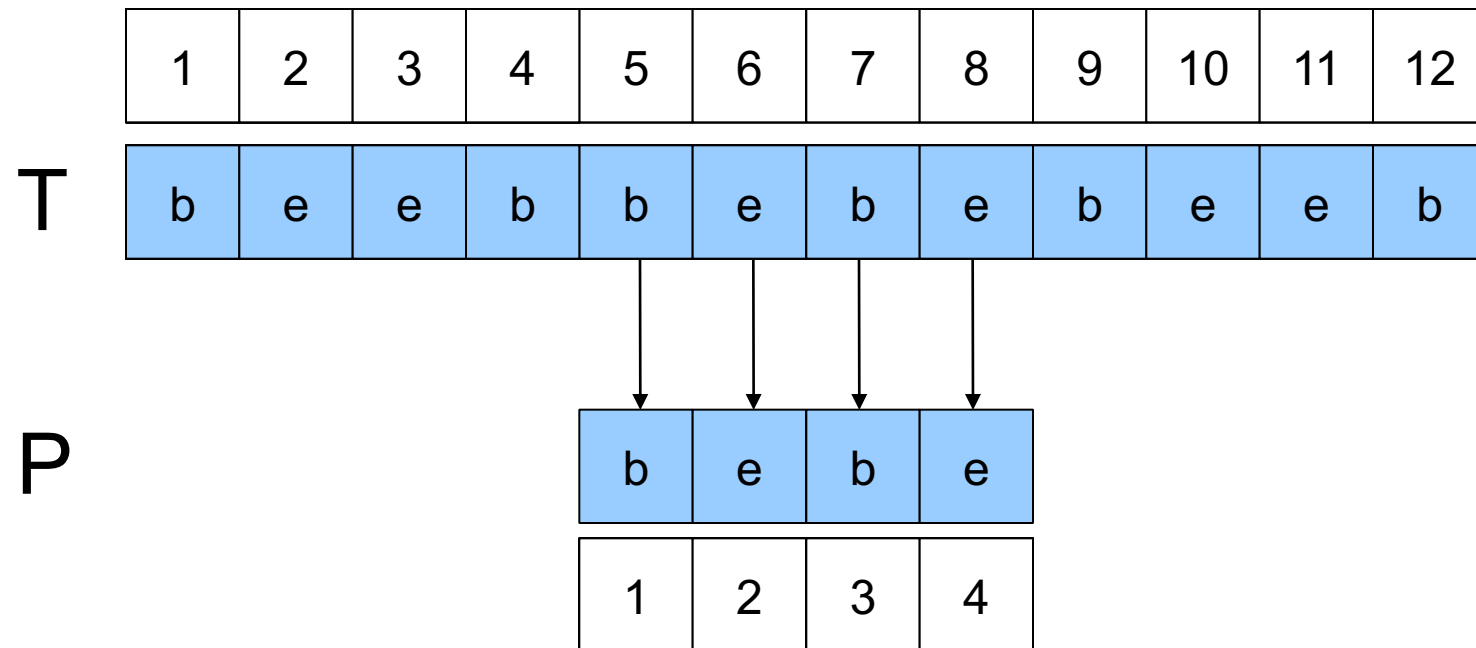
$s=3, j=2$



Algorithme naïf

$s=4, j=4=m$

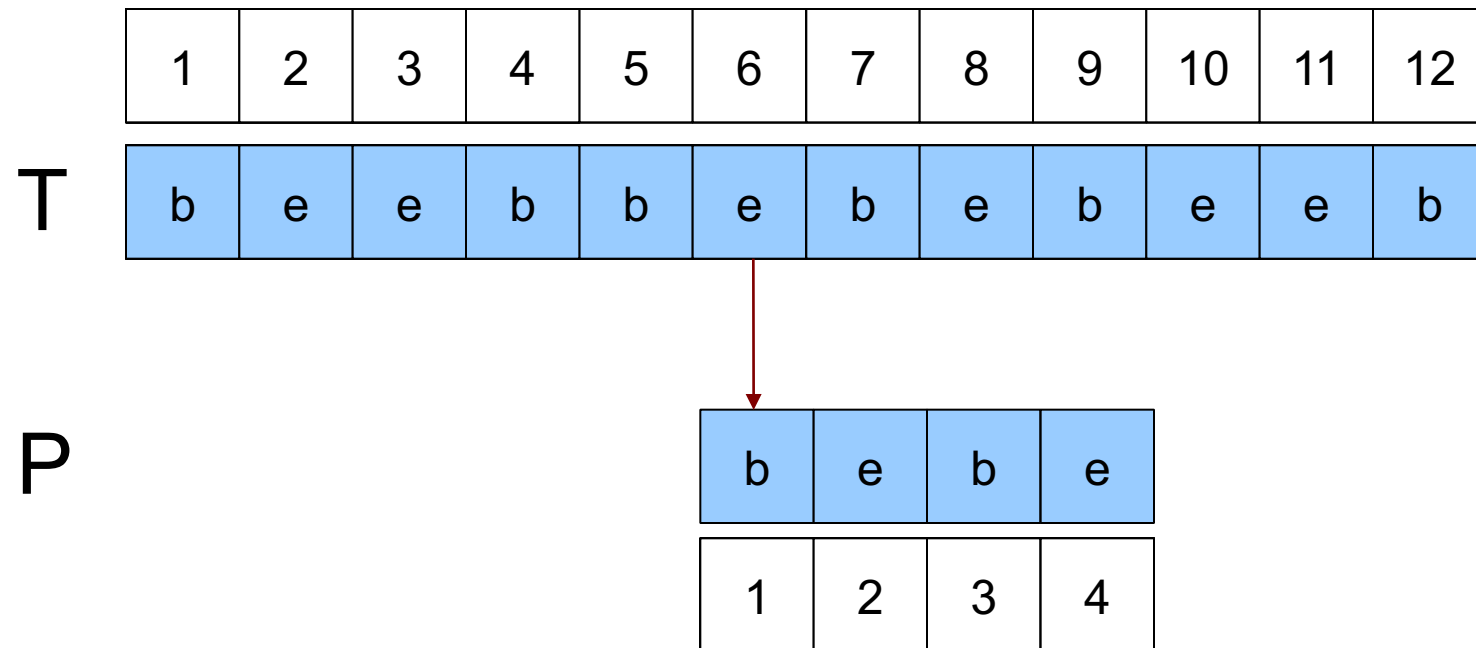
$S=\{4\}$



Algorithme naïf

$s=5, j=1$

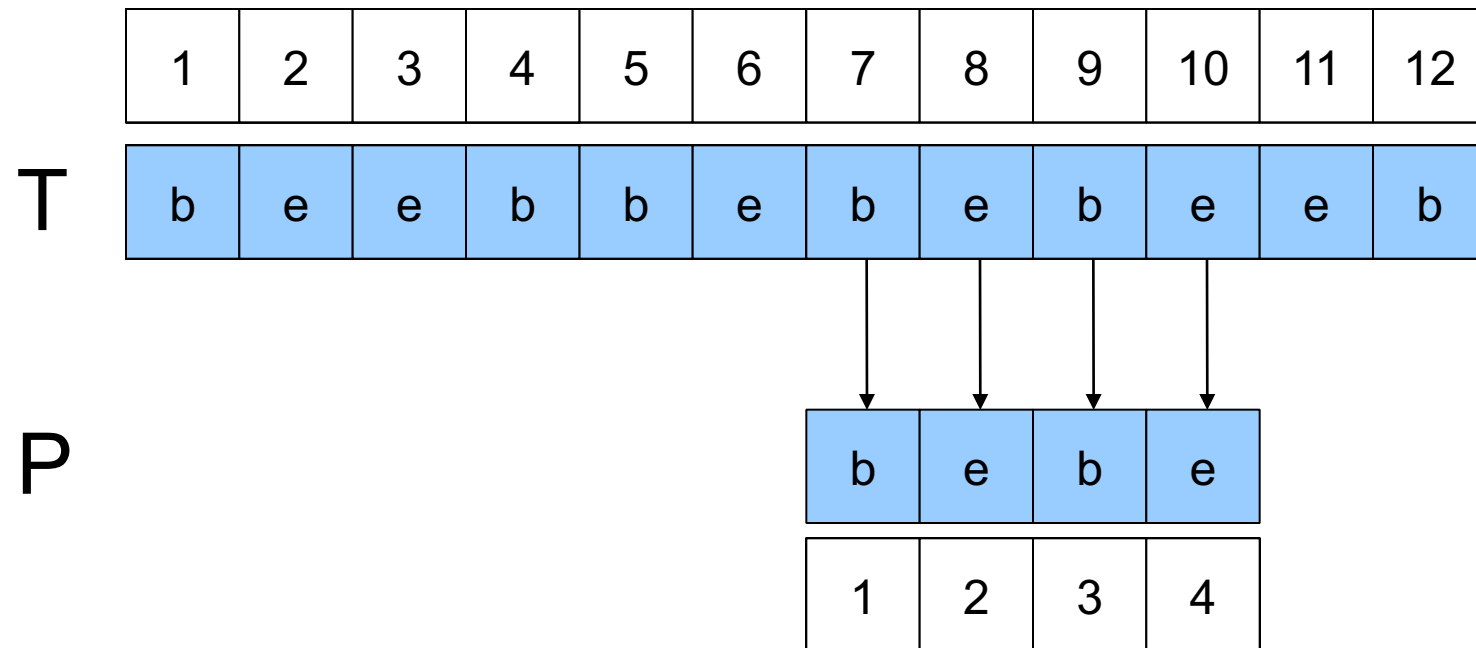
$S=\{4\}$



Algorithme naïf

$s=6, j=4$

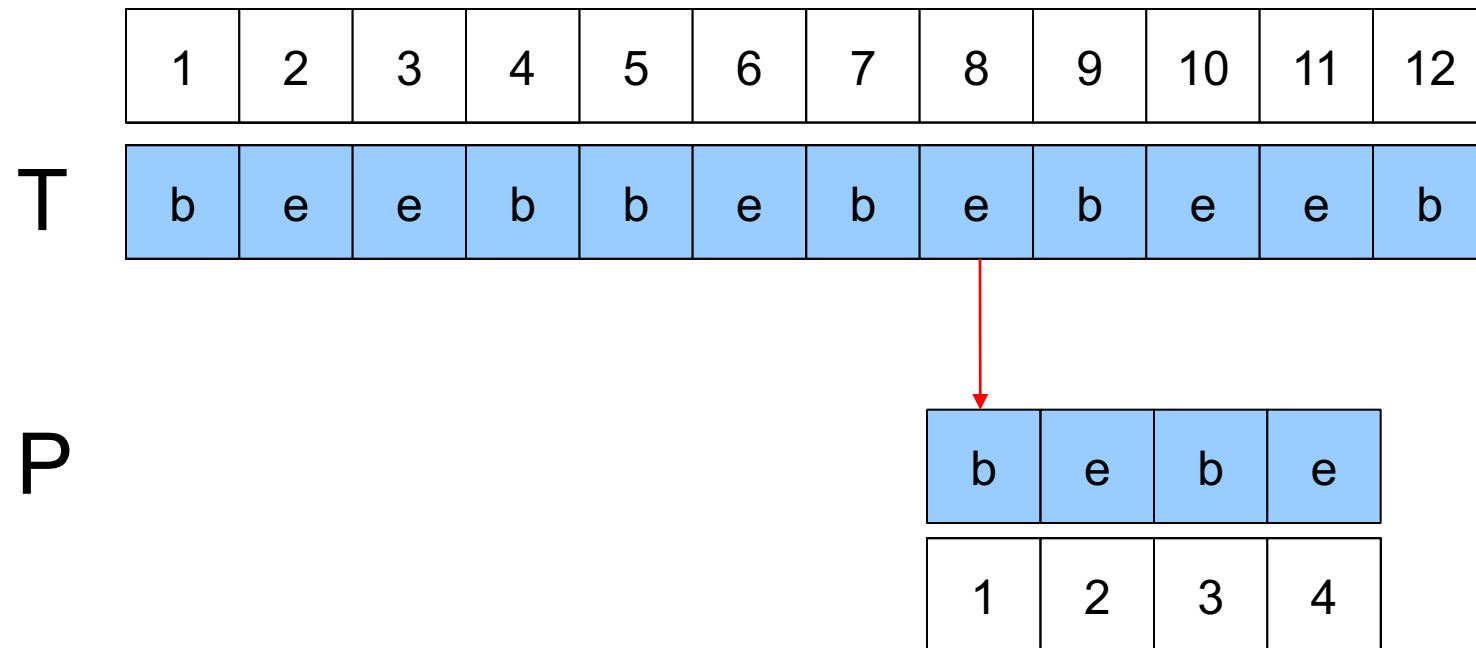
$S=\{4,6\}$



Algorithme naïf

$s=7, j=1$

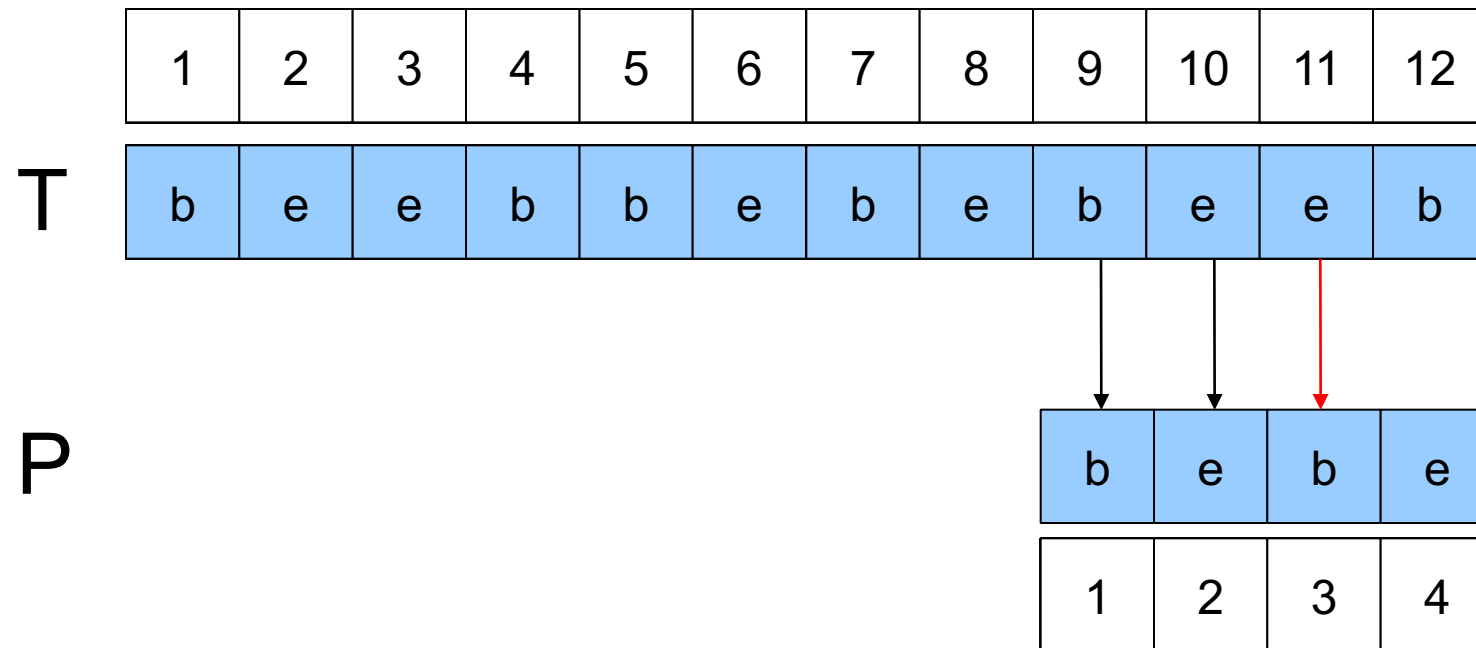
$S=\{4,6\}$



Algorithme naïf

$$s=8=12-4=n-m, j=3$$

$$S=\{4,6\}$$

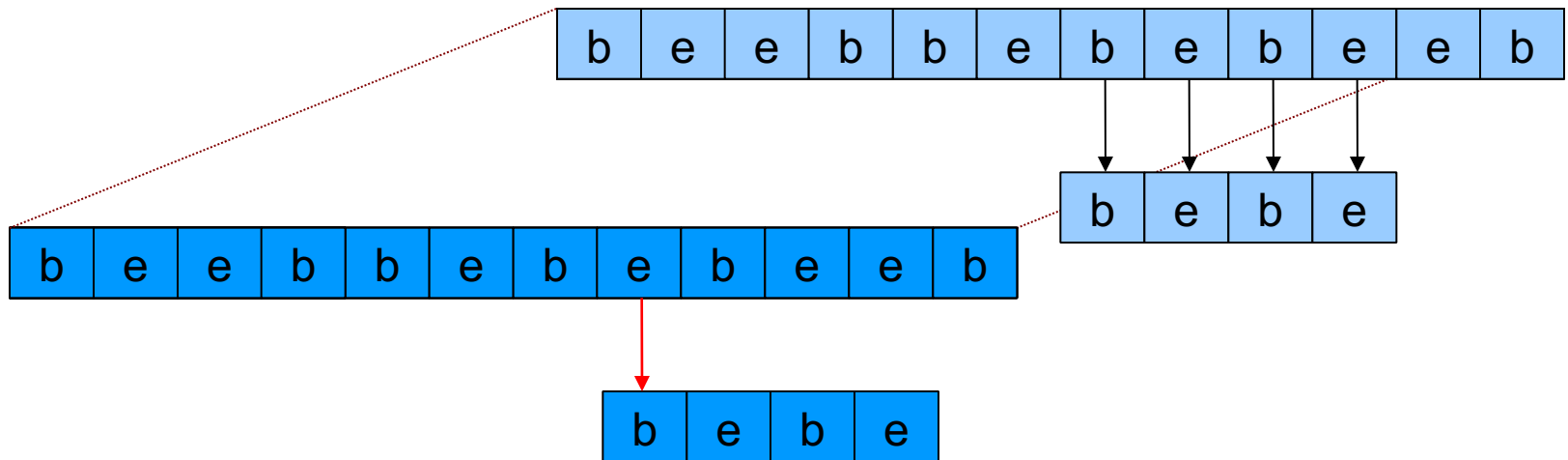


Algorithme naïf

Complexité de l'algorithme?

$$O(m(n-m+1))$$

Idée intéressante: réutiliser les résultats et analyser la chaîne de caractère P



Plan

Recherche de patron

Problématique

Rabin Karp

Automate FSM

PLSC

Problématique

Solution par programmation dynamique

Rabin-Karp

Objectif: battre l'algorithme naïf de complexité

$$O(m(n-m+1))$$

Analyser la chaîne de caractère:
pétraîtement (*preprocessing*)

Réutiliser les résultats précédent:
accélération

Rabin-Karp: formulation 1

Idée:

Pour un alphabet Σ , écrire P sous forme d'un nombre p
dans la base $d=|\Sigma|$

Analyser la chaîne de caractère:

Trouver une fois la valeur p associée à P

Rabin-Karp: formulation 1

Pour l'exemple, on utilise les chiffres 0-9:

Pour un alphabet $\Sigma=\{0, 1, 2, \dots, 8, 9\}$, écrire P sous forme d'un nombre dans la base $d=|\Sigma|=10$

Exemple:

P

1	2	3	5
---	---	---	---

$p = 1\ 235$

Rabin-Karp: formulation 1

P

1	2	3	5
---	---	---	---

$p = 1\ 235$

$$p = 1 \cdot 10^3 + 2 \cdot 10^2 + 3 \cdot 10^1 + 5 \cdot 10^0$$

Pour une longueur quelconque m , la phase de prétraîtement peut s'avérer ardue.

Rabin-Karp: formulation 1

P

1	2	3	5
---	---	---	---

p = 1 235

$$p = ((1 \cdot 10^1 + 2) \cdot 10^1 + 3) \cdot 10^1 + 5 \cdot 10^0$$

En utilisant l'algorithme de Horner, on réduit le nombre d'opérations pour le prétraitement.

Rabin-Karp: formulation 1

Calculer par Horner p de $P[1..m]$

Pour $s=0$ à $n-m$

Calculer par Horner t_s de $T[s+1..s+m]$

Si $t_s = p$

Inclure s dans S

Retourner S

Rabin-Karp: formulation 1

	1	2	3	4	5	6	7	8	9	10	11	12
T	1	3	2	3	4	1	2	3	5	1	2	3

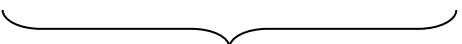
P	1	2	3	5
---	---	---	---	---

$p = 1\ 235$

Rabin-Karp: formulation 1

$s=0$

	1	2	3	4	5	6	7	8	9	10	11	12
T	1	3	2	3	4	1	2	3	5	1	2	3



$t_s = 1\ 323$

X

$p = 1\ 235$

1323

1 sort

4 entre

$$3234 = (1323 - 1 \times 10^{m-1}) \times 10 + 4$$

Rabin-Karp: formulation 1

$s=1$

	1	2	3	4	5	6	7	8	9	10	11	12
T	1	3	2	3	4	1	2	3	5	1	2	3

$t_s = 3\ 234$

X

$p = 1\ 235$

Rabin-Karp: formulation 1

$s=2$

	1	2	3	4	5	6	7	8	9	10	11	12
T	1	3	2	3	4	1	2	3	5	1	2	3

$t_s = 2\ 3\ 4\ 1$

X

$p = 1\ 2\ 3\ 5$

Rabin-Karp: formulation 1

$s=3$

	1	2	3	4	5	6	7	8	9	10	11	12
T	1	3	2	3	4	1	2	3	5	1	2	3

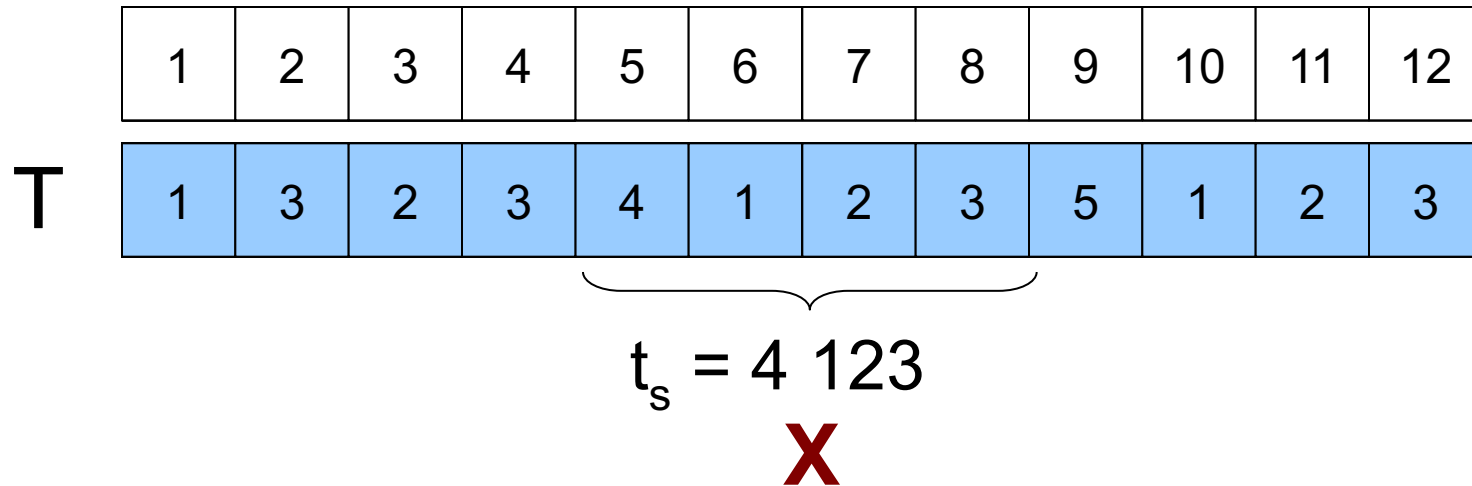
$t_s = 3\ 4\ 1\ 2$

X

$p = 1\ 2\ 3\ 5$

Rabin-Karp: formulation 1

$s=4$



$p = 1\ 235$

Rabin-Karp: formulation 1

$s=5$

$S=\{5\}$

	1	2	3	4	5	6	7	8	9	10	11	12
T	1	3	2	3	4	1	2	3	5	1	2	3

$t_s = 1\ 235$

✓

$p = 1\ 235$

Rabin-Karp: formulation 1

$s=6$

$S=\{5\}$

	1	2	3	4	5	6	7	8	9	10	11	12
T	1	3	2	3	4	1	2	3	5	1	2	3

$t_s = 2\ 3\ 5\ 1$

X

$p = 1\ 2\ 3\ 5$

Rabin-Karp: formulation 1

$s=7$

$S=\{5\}$

	1	2	3	4	5	6	7	8	9	10	11	12
T	1	3	2	3	4	1	2	3	5	1	2	3

$t_s = 3\ 512$

X

$p = 1\ 235$

Rabin-Karp: formulation 1

$$s=8=n-m$$

$$S=\{5\}$$

	1	2	3	4	5	6	7	8	9	10	11	12
T	1	3	2	3	4	1	2	3	5	1	2	3

$t_s = 5\ 123$
X

$$p = 1\ 235$$

Rabin-Karp: formulation 1

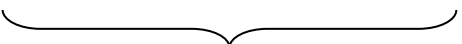
Réutiliser les résultats précédent:
accélération

$$t_{s+1} = d(t_s - d^{m-1}T[s]) + T[m+s+1]$$

Rabin-Karp: formulation 1

$s=0$

	1	2	3	4	5	6	7	8	9	10	11	12
T	1	3	2	3	4	1	2	3	5	1	2	3



$t_s = 1\ 3\ 2\ 3$

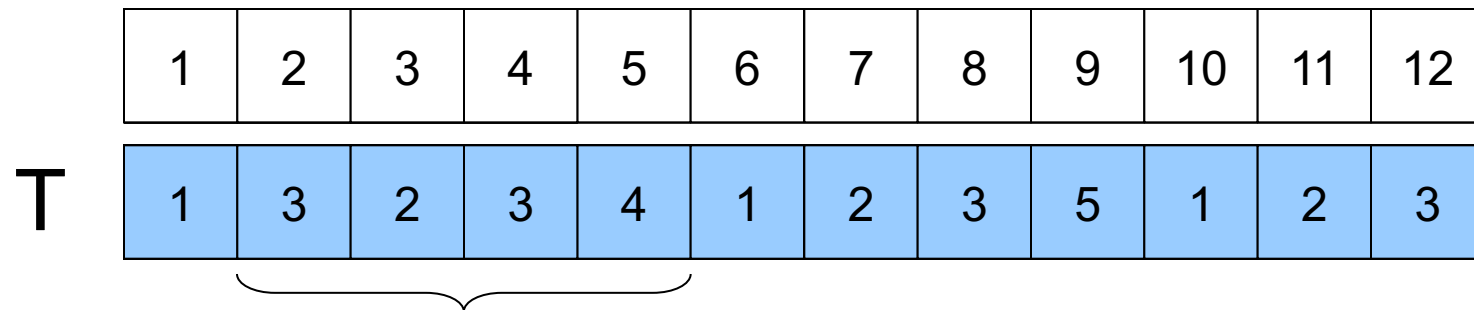
X

$p = 1\ 2\ 3\ 5$

Rabin-Karp: formulation 1

$s=1$

	1	2	3	4	5	6	7	8	9	10	11	12
T	1	3	2	3	4	1	2	3	5	1	2	3



$$t_s = 10(1\ 323 - 1000 \cdot 1) + 4 = 3\ 234$$

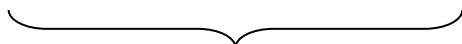
X

$p = 1\ 235$

Rabin-Karp: formulation 1

$s=2$

	1	2	3	4	5	6	7	8	9	10	11	12
T	1	3	2	3	4	1	2	3	5	1	2	3



$$t_s = 10(3\ 234 - 1000 \cdot 3) + 1 = 2\ 341$$

X

$p = 1\ 235$

Rabin-Karp: formulation 1

$s=3$

	1	2	3	4	5	6	7	8	9	10	11	12
T	1	3	2	3	4	1	2	3	5	1	2	3

$$t_s = 10(2\ 341 - 1000 \cdot 2) + 2 = 3\ 412$$

X

$p = 1\ 235$

Rabin-Karp: formulation 1

s=4

1	2	3	4	5	6	7	8	9	10	11	12
---	---	---	---	---	---	---	---	---	----	----	----

1	3	2	3	4	1	2	3	5	1	2	3
---	---	---	---	---	---	---	---	---	---	---	---

T

$$t_s = 10(3\ 412 - 1000 \cdot 3) + 3 = 4\ 123$$

X

p = 1 235

Rabin-Karp: formulation 1

$s=5$

$S=\{5\}$

1	2	3	4	5	6	7	8	9	10	11	12
1	3	2	3	4	1	2	3	5	1	2	3

$$t_s = 10(4 \ 123 - 1000 \cdot 4) + 5 = 1 \ 235$$

✓

$p = 1 \ 235$

Rabin-Karp: formulation 1

$s=6$

$S=\{5\}$

1	2	3	4	5	6	7	8	9	10	11	12
1	3	2	3	4	1	2	3	5	1	2	3

$$t_s = 10(1\ 235 - 1000 \cdot 1) + 1 = 2\ 351$$

X

$p = 2\ 351$

Rabin-Karp: formulation 1

$s=7$

$S=\{5\}$

	1	2	3	4	5	6	7	8	9	10	11	12
T	1	3	2	3	4	1	2	3	5	1	2	3

$$t_s = 10(2\ 351 - 1000 \cdot 2) + 2 = 3\ 512$$

X

$p = 1\ 235$

Rabin-Karp: formulation 1

$$s=8=n-m$$

$$S=\{5\}$$

	1	2	3	4	5	6	7	8	9	10	11	12
T	1	3	2	3	4	1	2	3	5	1	2	3

$$t_s = 10(3 \ 512 - 1000 \cdot 3) + 3 = 5 \ 123$$

X

$$p = 1 \ 235$$

Rabin-Karp: formulation 1

Problème:

L'alphabet Σ n'est pas celui des chiffres 0-9 et la base $d=|\Sigma|\neq 10$

ASCII étendu : 256 caractères $d = 256$

UNICODE: 65536 caractères $d = 65536$

Utiliser Horner et la technique itérative sur t_s risque de poser problème quand même. Les nombres vont être grands...

Rabin-Karp: formulation 2

Idée:

Utiliser une écriture modulaire. Pour un alphabet Σ , écrire P sous forme d'un nombre p dans la base $d=|\Sigma|$ modulo q

Choix de q :

On prend q de telle sorte que $d \cdot q$ tiennent dans un mot machine (32 bits) :

Raison: essayer d'exprimer p en fonction de Horner...

Rabin-Karp: formulation 2

Pour illustrer l'algorithme, reprenons l'exemple, on utilise les chiffres 0-9: on traduit P sous la forme du nombre p dans la base $d=|\Sigma|=10$ modulo $q = 11$

P

1	2	3	5
---	---	---	---

Exemple:

$$p = \{ \{ \{ \{ 1 \cdot 10 + 2 \} \bmod 11 \} \cdot 10 + 3 \} \bmod 11 \} \cdot 10 + 5 \} \bmod 11$$

$$p = \{ \{ \{ \{ 12 \} \bmod 11 \} \cdot 10 + 3 \} \bmod 11 \} \cdot 10 + 5 \} \bmod 11$$

$$p = \{ \{ \{ \{ 1 \} \cdot 10 + 3 \} \bmod 11 \} \cdot 10 + 5 \} \bmod 11$$

$$p = \{ \{ 2 \} \cdot 10 + 5 \} \bmod 11$$

$$p = \{ 25 \} \bmod 11 = 3$$

Rabin-Karp: formulation 2

Réutiliser les résultats précédent:
accélération

$$t_{s+1} = \{d(t_s - hT[s]) + T[m+s+1]\} \bmod q$$

$$h = d^{m-1} \bmod q$$

Rabin-Karp: formulation 2

	1	2	3	4	5	6	7	8	9	10	11	12
T	1	3	2	3	4	1	2	3	5	1	2	3

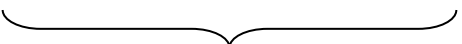
P	1	2	3	5
---	---	---	---	---

$$p = 1\ 235 \bmod 11 = 3$$

Rabin-Karp: formulation 2

$s=0$

	1	2	3	4	5	6	7	8	9	10	11	12
T	1	3	2	3	4	1	2	3	5	1	2	3



$$t_s = 1\ 323 \bmod 11 = 3$$

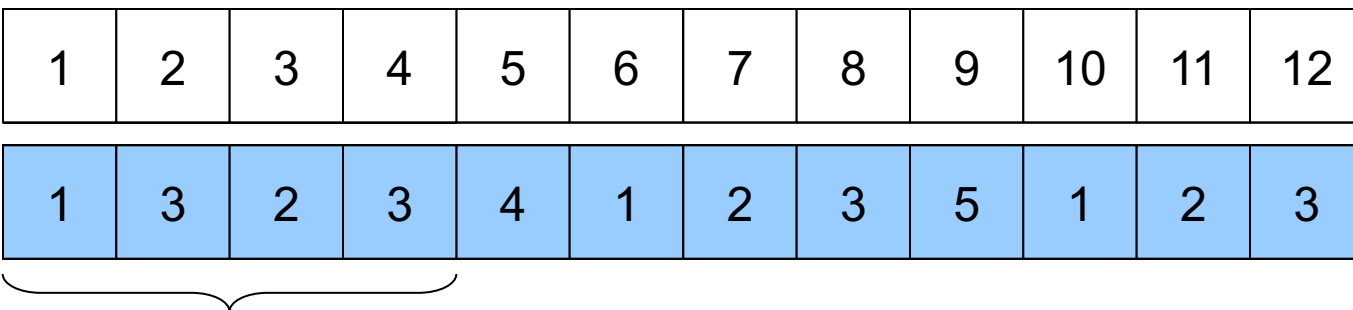
?

$p = 3$

Rabin-Karp: formulation 2

$s=0$

	1	2	3	4	5	6	7	8	9	10	11	12
T	1	3	2	3	4	1	2	3	5	1	2	3



$$t_s = 1\ 323 \bmod 11 = 3$$

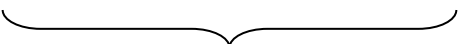
faux-positif

$p = 3$

Rabin-Karp: formulation 2

$s=0$

	1	2	3	4	5	6	7	8	9	10	11	12
T	1	3	2	3	4	1	2	3	5	1	2	3



$$t_s = 1\ 323 \bmod 11 = 3$$

X

$p = 3$

P

1	2	3	5
---	---	---	---

Rabin-Karp: formulation 2

$s=1$

	1	2	3	4	5	6	7	8	9	10	11	12
T	1	3	2	3	4	1	2	3	5	1	2	3

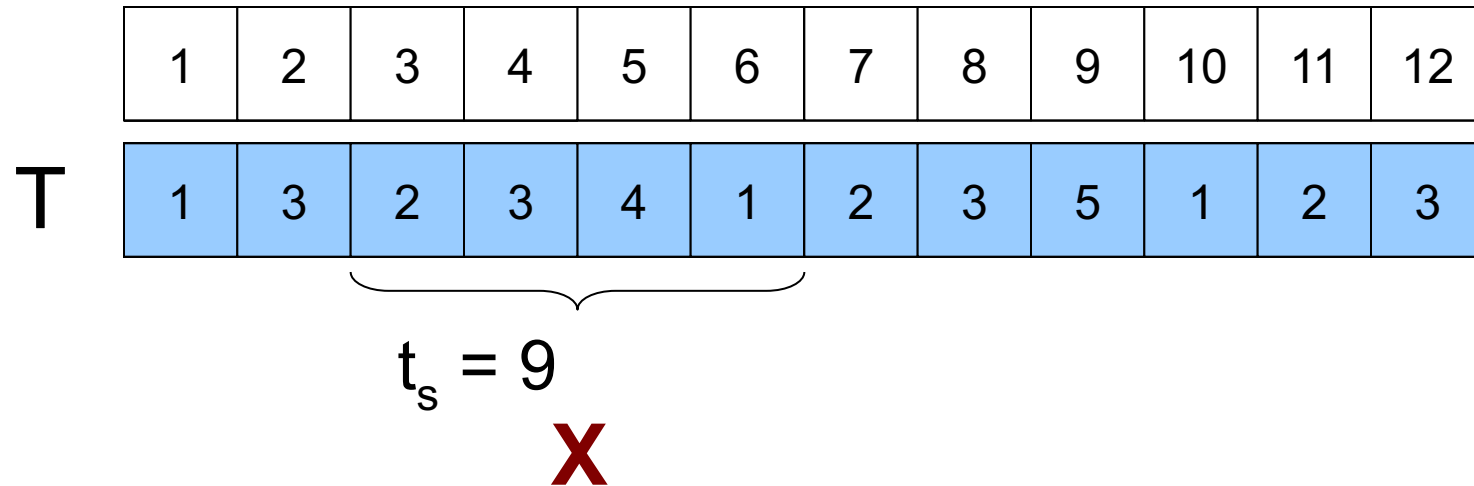
$t_s = 0$

X

$p = 3$

Rabin-Karp: formulation 2

$s=2$



$p = 3$

Rabin-Karp: formulation 2

$s=3$

	1	2	3	4	5	6	7	8	9	10	11	12
T	1	3	2	3	4	1	2	3	5	1	2	3

$t_s = 2$
X

$p = 3$

Rabin-Karp: formulation 2

$s=4$

	1	2	3	4	5	6	7	8	9	10	11	12
T	1	3	2	3	4	1	2	3	5	1	2	3

$t_s = 9$
X

$p = 3$

Rabin-Karp: formulation 2

$s=5$

$S=\{5\}$

	1	2	3	4	5	6	7	8	9	10	11	12
T	1	3	2	3	4	1	2	3	5	1	2	3

$t_s = 3$
?

$p = 3$

Rabin-Karp: formulation 2

$s=5$

$S=\{5\}$

	1	2	3	4	5	6	7	8	9	10	11	12
T	1	3	2	3	4	1	2	3	5	1	2	3

$t_s = 3$

$p = 3$

P

1	2	3	5
---	---	---	---

Rabin-Karp: formulation 2

$s=6$

$S=\{5\}$

	1	2	3	4	5	6	7	8	9	10	11	12
T	1	3	2	3	4	1	2	3	5	1	2	3

$t_s = 8$
X

$p = 3$

Rabin-Karp: formulation 2

$s=7$

$S=\{5\}$

	1	2	3	4	5	6	7	8	9	10	11	12
T	1	3	2	3	4	1	2	3	5	1	2	3

$t_s = 3$
?

$p = 3$

Rabin-Karp: formulation 2

$s=7$

$S=\{5\}$

	1	2	3	4	5	6	7	8	9	10	11	12
T	1	3	2	3	4	1	2	3	5	1	2	3

$t_s = 3$

X

$p = 3$

P

1	2	3	5
---	---	---	---

Rabin-Karp: formulation 2

$$s=8=n-m$$

$$S=\{5\}$$

	1	2	3	4	5	6	7	8	9	10	11	12
T	1	3	2	3	4	1	2	3	5	1	2	3

$t_s = 7$
X

$$p = 3$$

Rabin-Karp

Complexité de l'algorithme à cause des faux-positifs

$$O(m(n-m+1))$$

1	3	2	3	4	1	2	3	5	1	2	3
---	---	---	---	---	---	---	---	---	---	---	---

$t_s = 3$ **X**

P

1	2	3	5
---	---	---	---

$O(n)$ en cas moyen et en meilleur cas

Néanmoins, l'exécution de Rabin Karp est **plus rapide** que l'algorithme naïf.

Plan

Recherche de patron

Problématique

Rabin Karp

Automate FSM

PLSC

Problématique

Solution par programmation dynamique

Automate FSM

Objectif: meilleure complexité

Battre $O(m(n-m+1))$

Meilleure performances que Rabin-Karp

Analyser la chaîne de caractère:

Construire une machine à états (prétraîtement)

Réutiliser les résultats précédents:

L'automate possède une mémoire interne de son état (accélération)

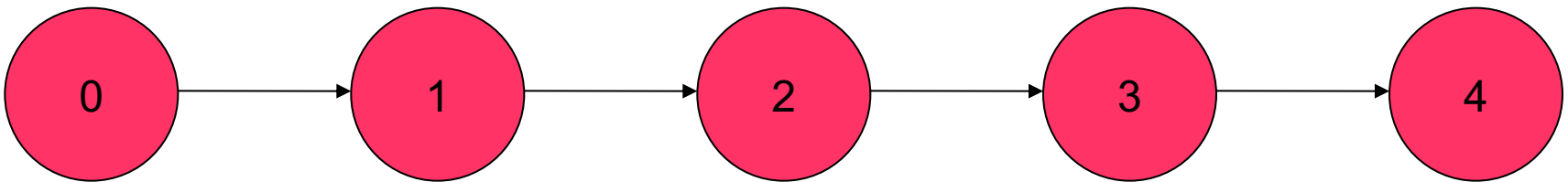
Automate FSM

Construire - Le nombre d'états est égal à $m+1$

Exemple:

P

b	e	b	e
---	---	---	---



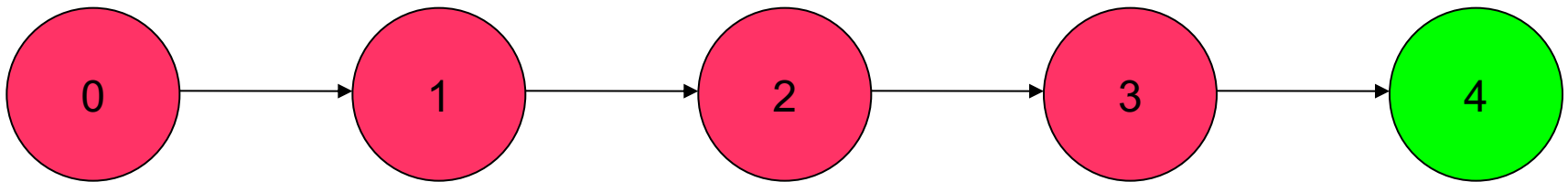
Automate FSM

Construire - Le dernier état valide l'entrée

Exemple:

P

b	e	b	e
---	---	---	---



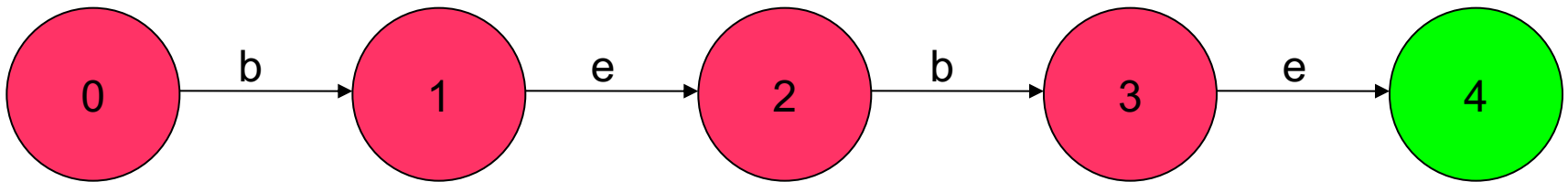
Automate FSM

Construire – Les arcs répondent aux unités

Exemple:

P

b	e	b	e
---	---	---	---



Automate FSM

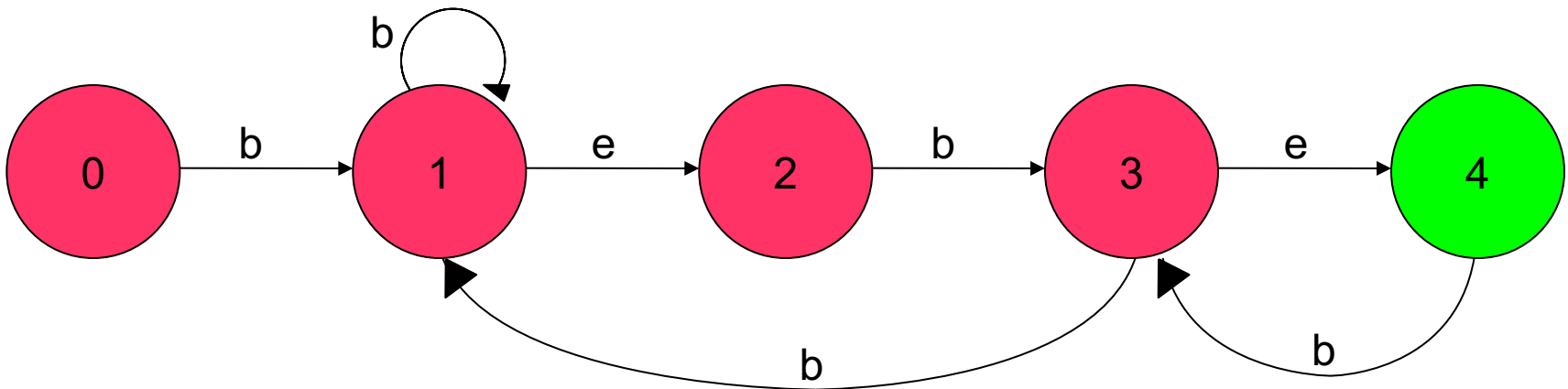
ConstruRe – Ajouter les arcs restants

Exemple:

P

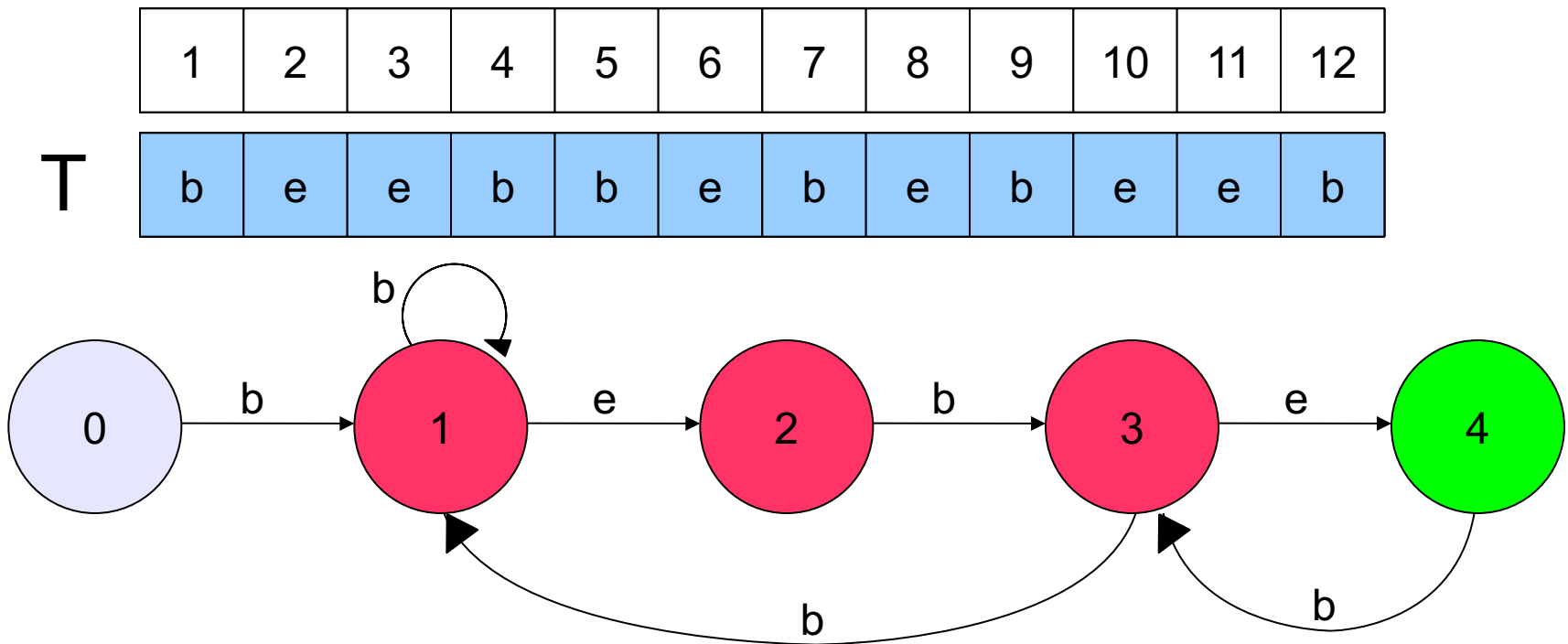
b	e	b	e
---	---	---	---

Note: les arcs absents mènent à l'état de départ (0)



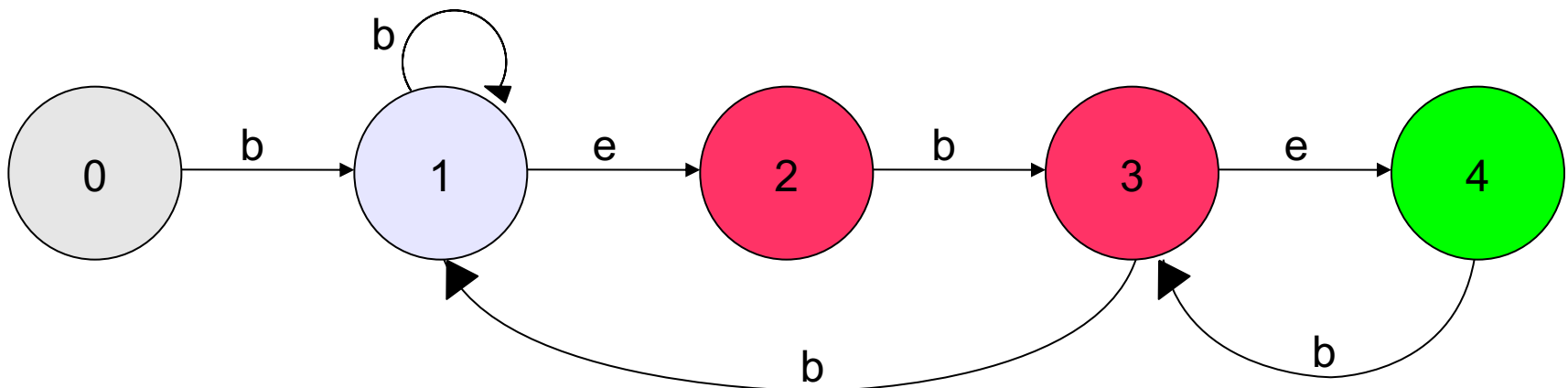
Automate FSM

Traîter – La chaîne T est traversée une fois



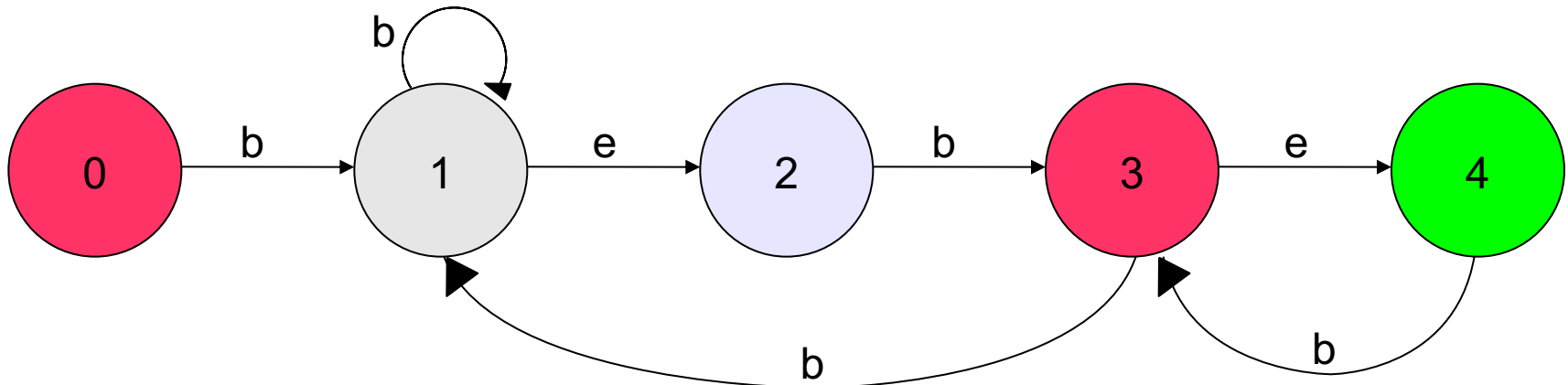
Automate FSM

	1	2	3	4	5	6	7	8	9	10	11	12
T	b	e	e	b	b	e	b	e	b	e	e	b



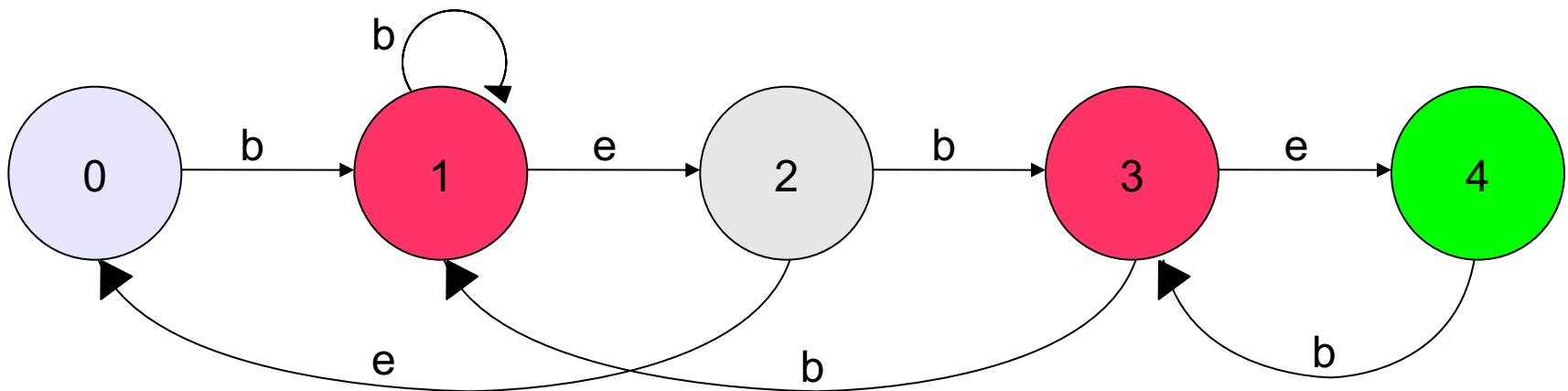
Automate FSM

1	2	3	4	5	6	7	8	9	10	11	12
b	e	e	b	b	e	b	e	b	e	e	b



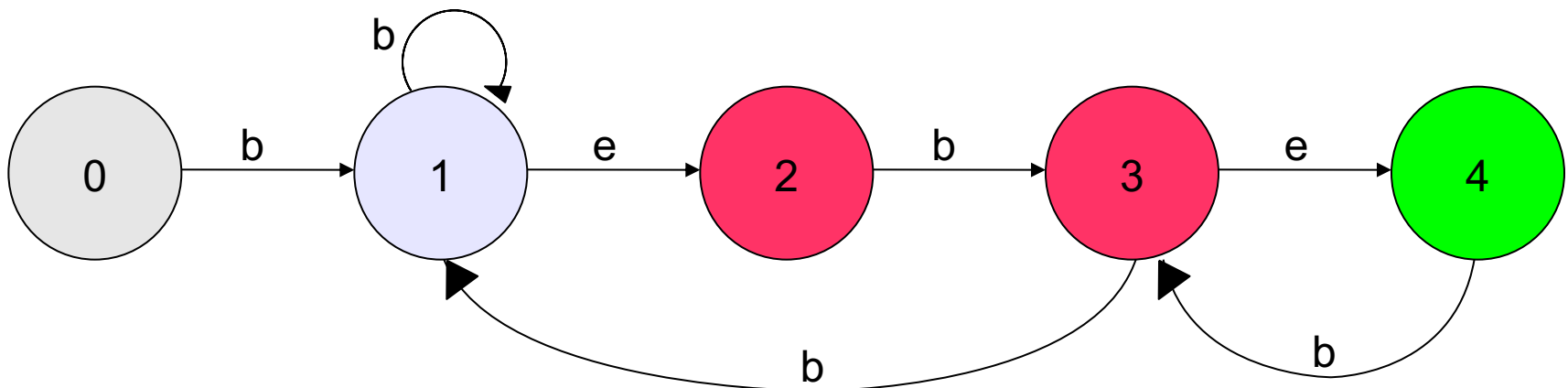
Automate FSM

1	2	3	4	5	6	7	8	9	10	11	12
b	e	e	b	b	e	b	e	b	e	e	b



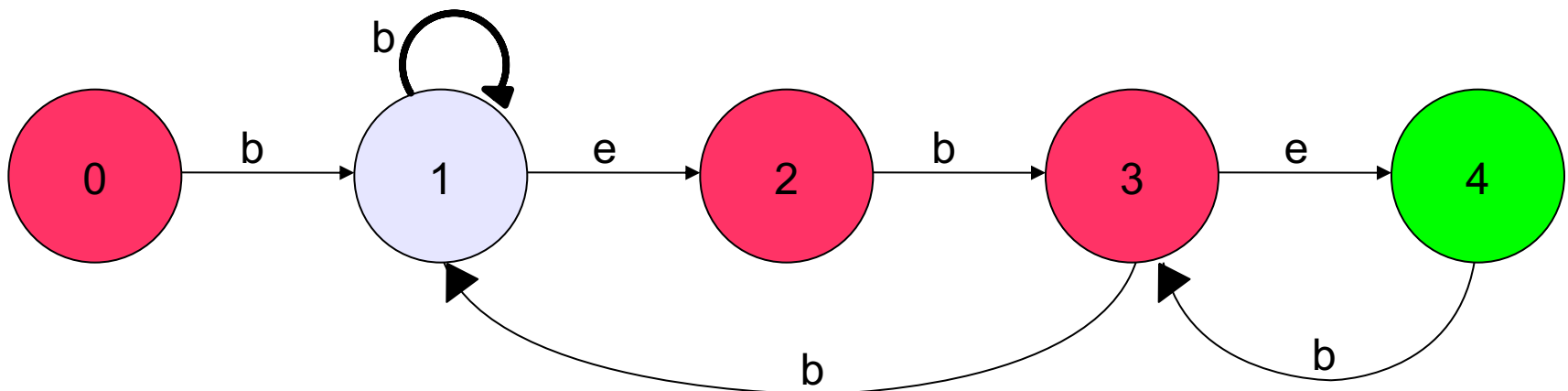
Automate FSM

1	2	3	4	5	6	7	8	9	10	11	12
b	e	e	b	b	e	b	e	b	e	e	b



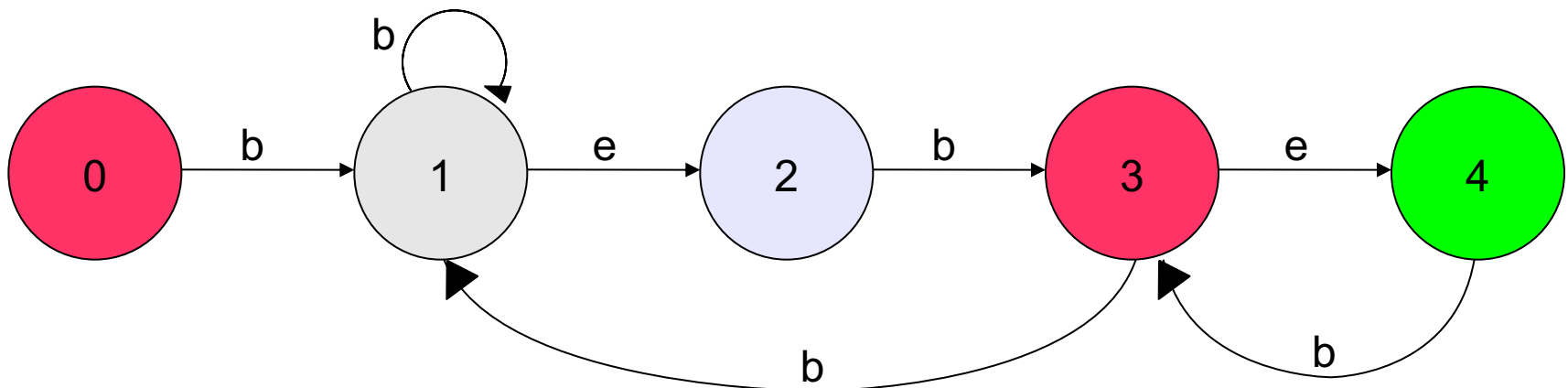
Automate FSM

1	2	3	4	5	6	7	8	9	10	11	12
b	e	e	b	b	e	b	e	b	e	e	b



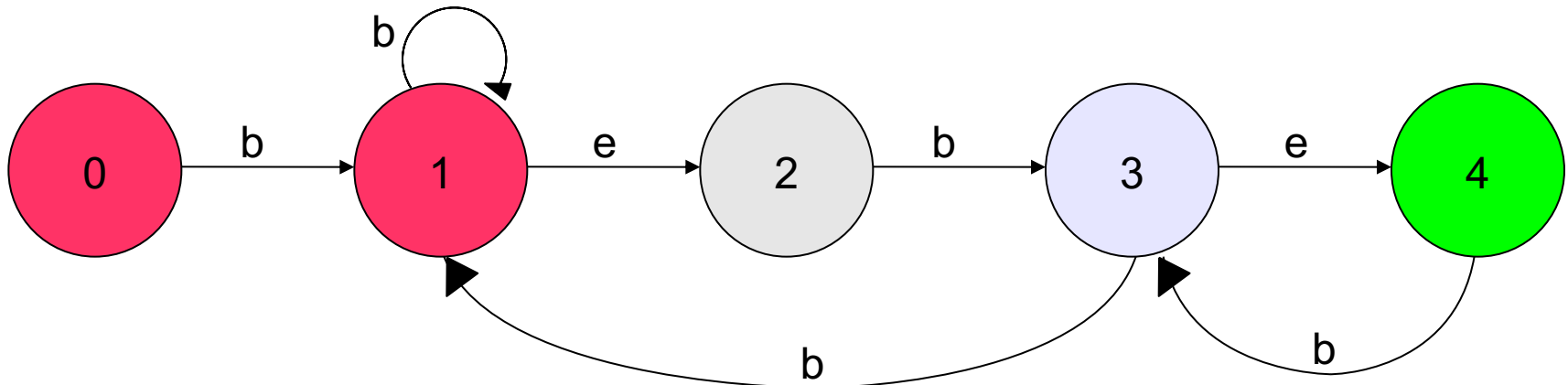
Automate FSM

1	2	3	4	5	6	7	8	9	10	11	12
b	e	e	b	b	e	b	e	b	e	e	b



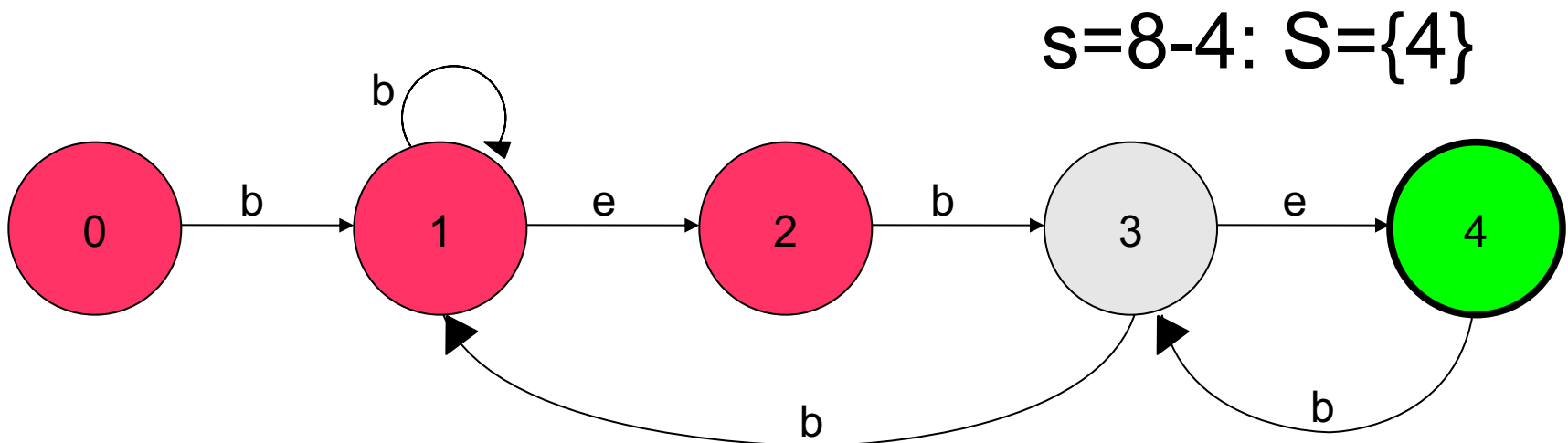
Automate FSM

1	2	3	4	5	6	7	8	9	10	11	12
b	e	e	b	b	e	b	e	b	e	e	b



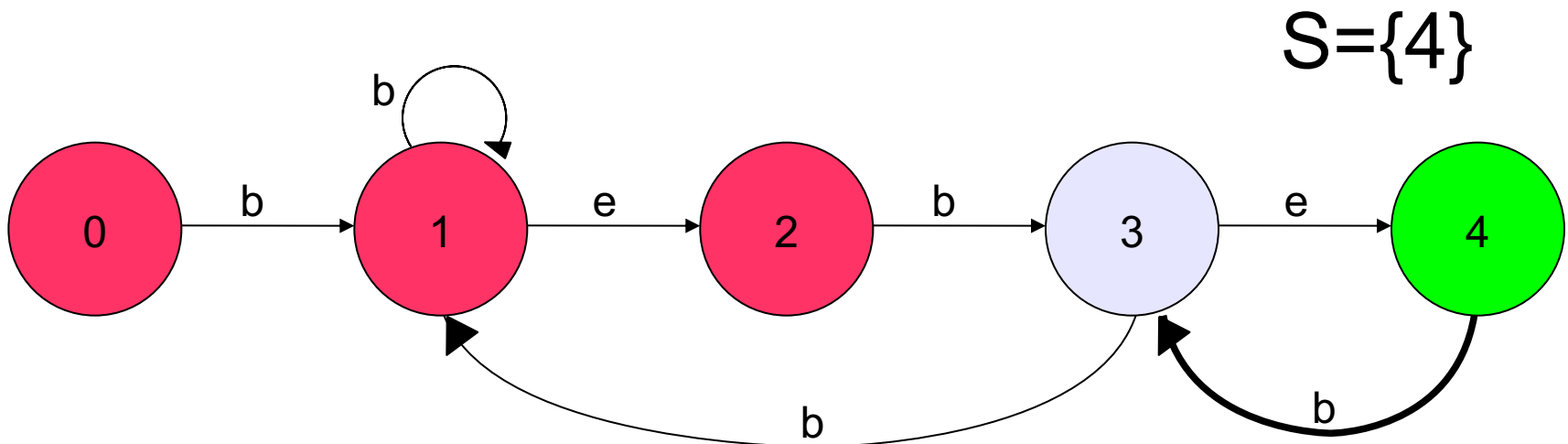
Automate FSM

	1	2	3	4	5	6	7	8	9	10	11	12
T	b	e	e	b	b	e	b	e	b	e	e	b



Automate FSM

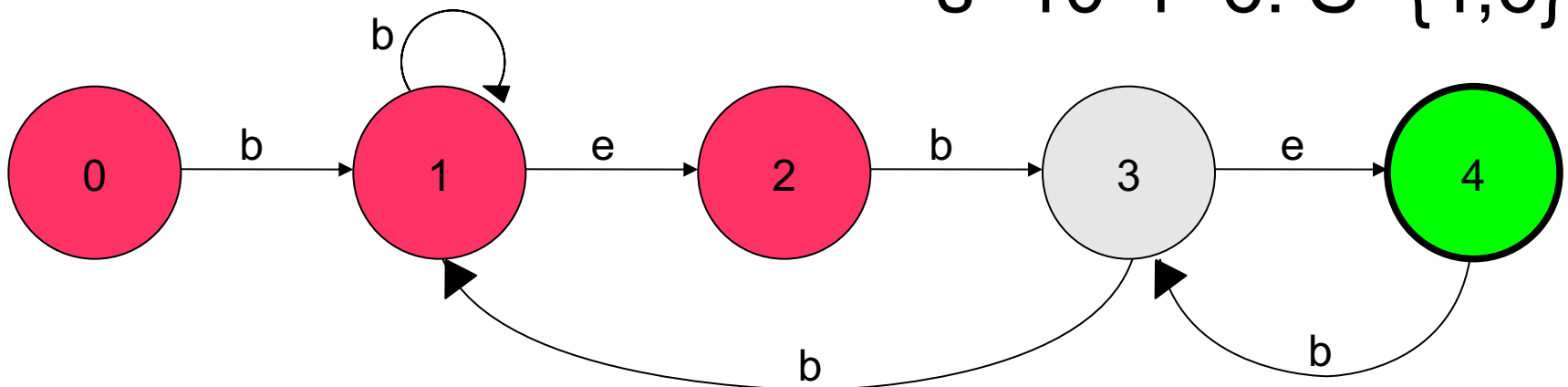
1	2	3	4	5	6	7	8	9	10	11	12
b	e	e	b	b	e	b	e	b	e	e	b



Automate FSM

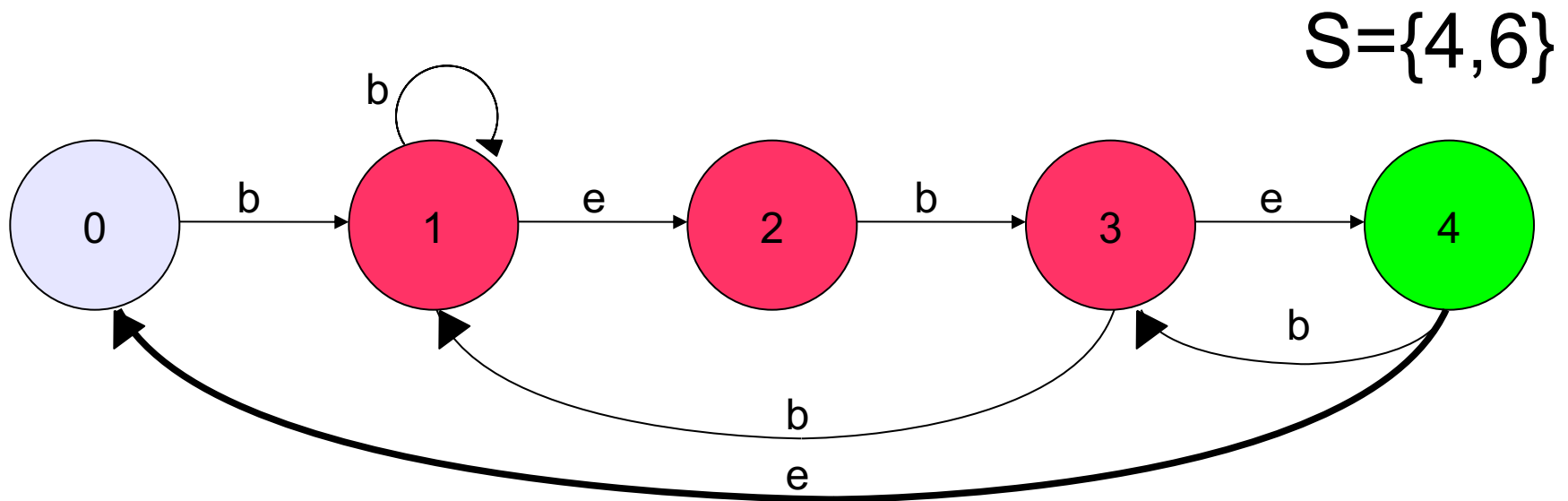
1	2	3	4	5	6	7	8	9	10	11	12
b	e	e	b	b	e	b	e	b	e	e	b

$$s=10-4=6: S=\{4,6\}$$



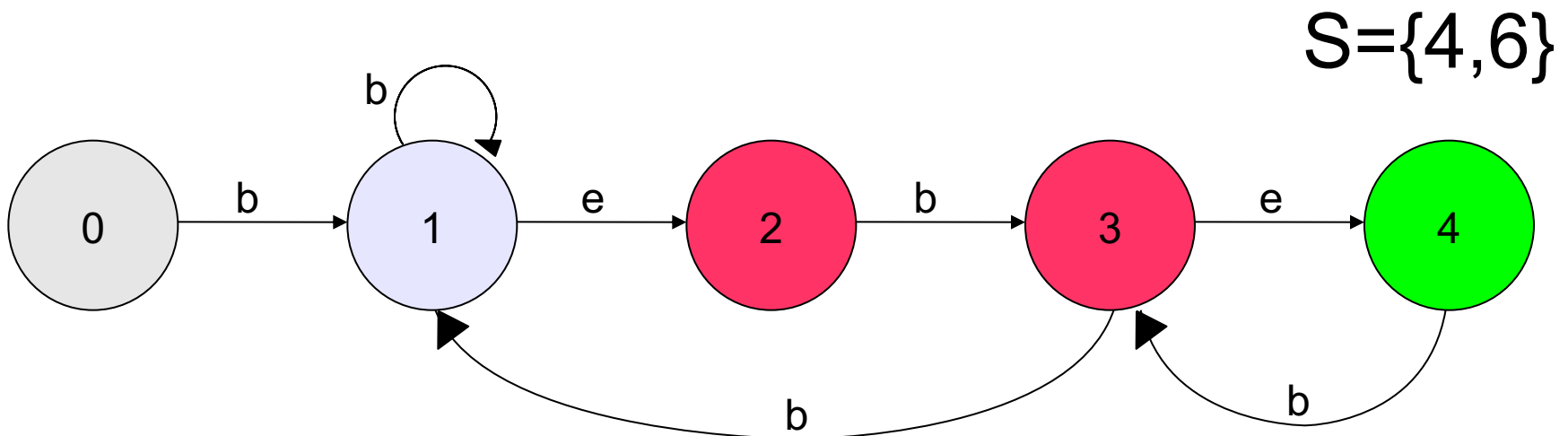
Automate FSM

1	2	3	4	5	6	7	8	9	10	11	12
b	e	e	b	b	e	b	e	b	e	e	b



Automate FSM

1	2	3	4	5	6	7	8	9	10	11	12
b	e	e	b	b	e	b	e	b	e	e	b



Automate FSM

Quelques définitions

Ensemble des états $Q = \{q_0, q_1, q_2, \dots\}$

Les états peuvent être représentés par un entier (leur indice)

État initial: 0

État final: m

Fonction post-fixe est notée: \supset

Fonction préfixe est notée: \subset

Le préfixe de P de longueur i est noté P_i ($P_i \subset P$)

Alphabet: Σ

Alphabet réduit aux lettre du patron: Σ_p

Fonction de transition $\delta : Q \times \Sigma_p \rightarrow Q$

Construction

Algorithme de construction de la fonction de transition

```
Initialiser  $\delta$  à 0
Pour  $q = 0 : m$ 
    Pour chaque caractère  $a$  dans  $\Sigma_P$ 
         $k = \min(m+1, q+2)$ 
        Répéter
             $k = k - 1$ 
            Tant que  $k > 0$  et  $(P_k \supset P_q a)$  est faux
                 $\delta(q, a) = k$ 
        Fin Pour
Fin Pour
```

Exemple

Rechercher: aabab

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$

$Q_0 = 0$

État final : 5

$\Sigma_p = \{a, b\}$

$q_0 \leftrightarrow \varepsilon$

$q_1 \leftrightarrow a$

$q_2 \leftrightarrow aa$

$q_3 \leftrightarrow aab$

$q_4 \leftrightarrow aaba$

$q_5 \leftrightarrow aabab$

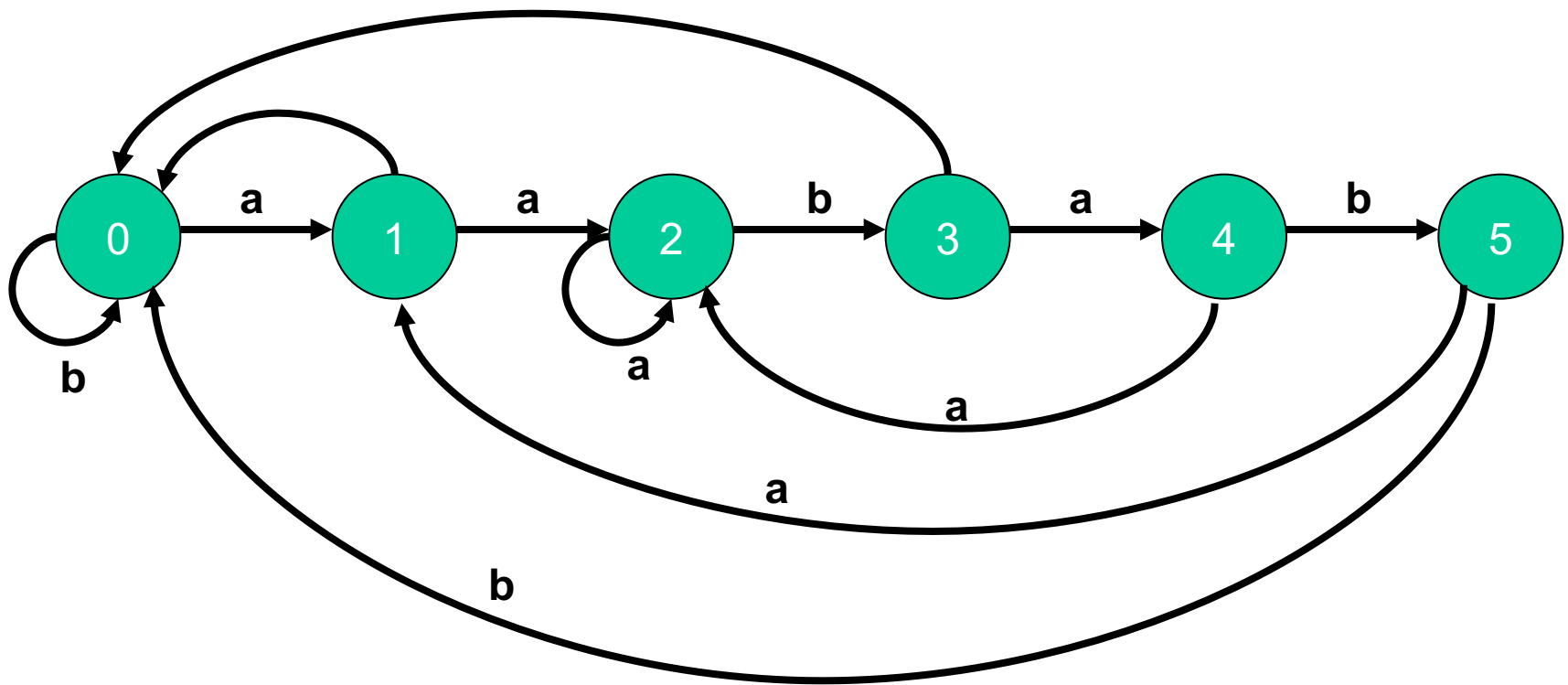
Exemple

- Fonction de transition δ :

	a	b
q0	1	0
q1	2	0
q2	2	3
q3	4	0
q4	2	5
q5	1	0

Patron: aabab

Exemple



Construction

Idée: chaque état a reconnu le préfixe de la chaîne à rechercher qui lui est associée

$q_0 \leftrightarrow \varepsilon$

$q_1 \leftrightarrow a$

$q_2 \leftrightarrow aa$

$q_3 \leftrightarrow aab$

$q_4 \leftrightarrow aaba$

$q_5 \leftrightarrow aabab$

Exemple de construction

Position: 012345

Chaine: aabab

$P_0 = \varepsilon$

$P_1 = a$

$P_2 = aa$

$P_3 = aab$

$P_4 = aaba$

$P_5 = aabab$

P_i est le préfixe de P de i lettres

Exemple de construction

Algorithme

Patron: aabab

```
Initialiser  $\delta$  à 0
Pour  $q = 0 : m$ 
    Pour chaque caractère  $a$  dans  $\Sigma_P$ 
         $k = \min(m+1, q+2)$ 
        Répéter
             $k = k - 1$ 
            Tant que  $k > 0$  et  $(P_k \supset P_q a)$  est faux
                 $\delta(q, a) = k$ 
        Fin Pour
Fin Pour
```

	a	b
q0	0	0
q1	0	0
q2	0	0
q3	0	0
q4	0	0
q5	0	0

Transition 0, a

```
1: m ← 5
2: q ← 0
3: a ← 'a'
4: k ← min(6, 2)
   k ← 2
5: k ← 1
6: a ⊃ εa ?
   a ⊃ a  ✓
7: δ(0, a) ← 1
```

Patron: aabab

	a	b
q0	0	0
q1	0	0
q2	0	0
q3	0	0
q4	0	0
q5	0	0

Transition 0, a

1: $m \leftarrow 5$
2: $q \leftarrow 0$
3: $a \leftarrow 'a'$
4: $k \leftarrow \min(6, 2)$
 $k \leftarrow 2$
5: $k \leftarrow 1$
6: $a \supset \varepsilon a$?
 $a \supset a$ \checkmark
7: $\delta(0, a) \leftarrow 1$

Patron: aabab

	a	b
q0	1	0
q1	0	0
q2	0	0
q3	0	0
q4	0	0
q5	0	0

Transition 0, b

Patron: aabab

3: $a \leftarrow 'b'$
4: $k \leftarrow \min(6, 2)$
 $\leftarrow 2$
5: $k \leftarrow 1$
6: $a \supset \varepsilon b$
 $a \supset b$
 $\leftarrow F$
5: $k \leftarrow 0$
6: $\varepsilon \supset \varepsilon b$
 $\varepsilon \supset b$
 $\leftarrow T$
7: $\delta(0, b) \leftarrow 0$

	a	b
q0	1	0
q1	0	0
q2	0	0
q3	0	0
q4	0	0
q5	0	0

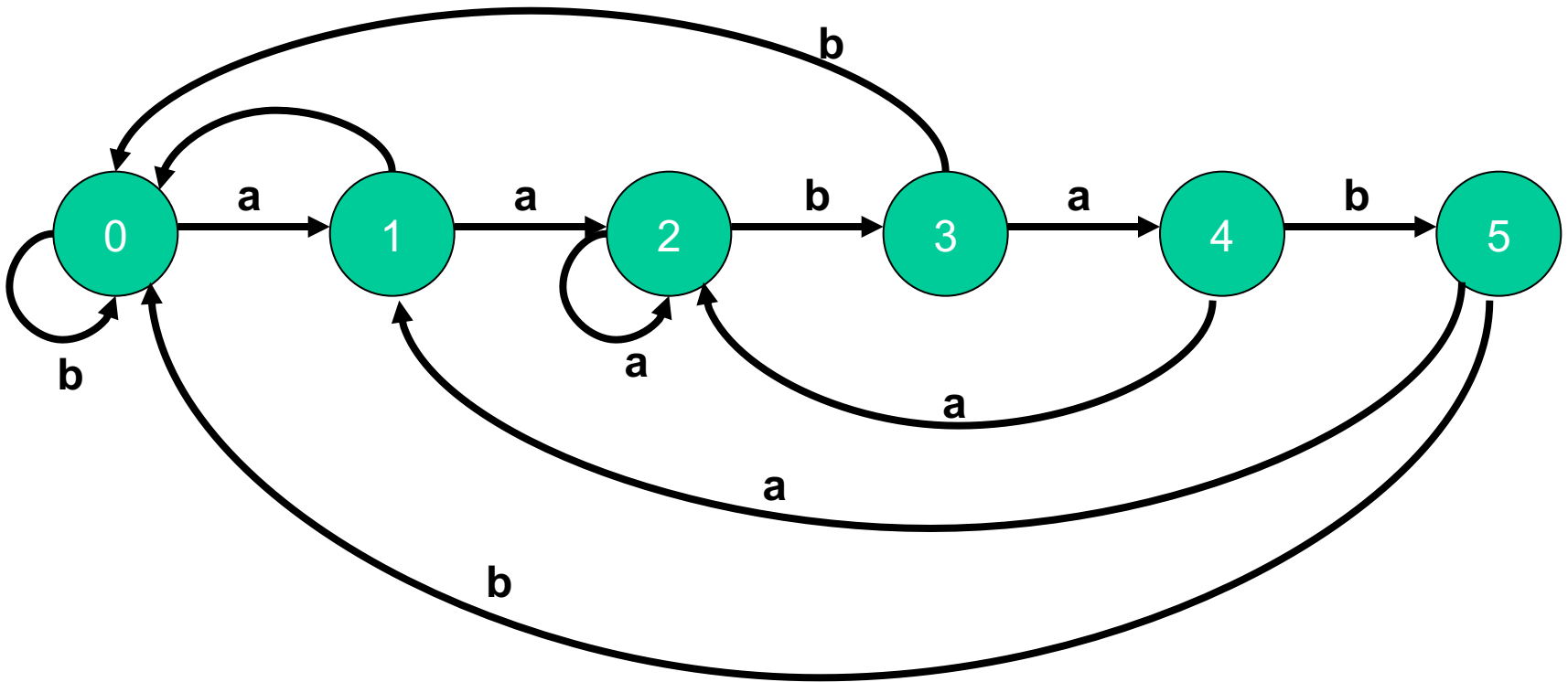
Transition 5, b

Patron: aabab

	a	b
q0	1	0
q1	2	0
q2	2	3
q3	4	0
q4	2	5
q5	1	0

Transition 5, b

$$\delta(5, b) \leftarrow 0$$



Automate FSM

Dans le meilleur des cas, l'automate FSM donne une complexité

$$O(n) = \Theta(n)$$

La construction de la machine à états peut être coûteuse: $O(m^3d)$. Ce qui peut devenir handicapant pour certains problèmes de recherche:

$$O(m^3d_p) + \Theta(n) \text{ vs. } O(m(n-m+1) + m)$$
$$d_p = |\Sigma_p|$$

Plan

Recherche de patron

Problématique

Rabin Karp

Automate FSM

PLSC

Problématique

Solution par programmation dynamique

Problématique

Problématique:

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

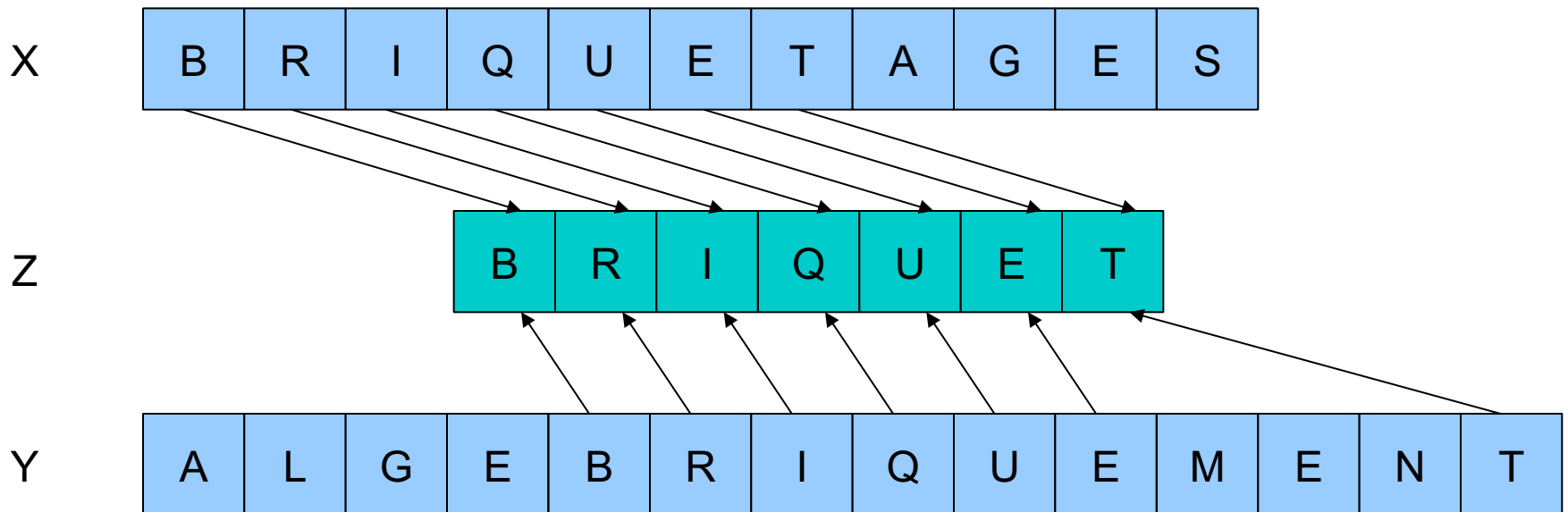
Exemples:

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

Problématique

Approche:

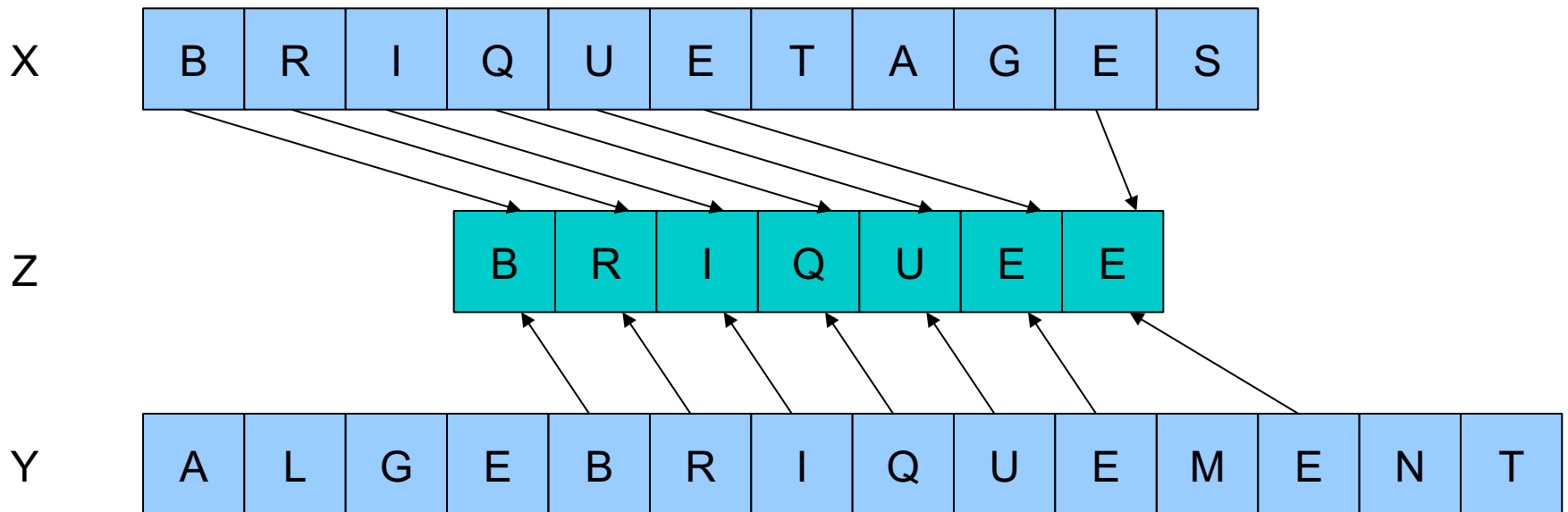
La sous-séquence n'est pas continue dans les chaînes d'entrée



Problématique

Approche:

La solution n'est pas unique



Approche naïve

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

X

A	B	B	A
---	---	---	---

Z

?

Y

A	B	A	T
---	---	---	---

Approche naïve

- Exemple:

X	A	B	B	A
---	---	---	---	---

Y	A	B	A	T
---	---	---	---	---

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

Approche naïve

- Exemple:

X	A	B	B	A	Y	A	B	A	T
---	---	---	---	---	---	---	---	---	---

COMPLEXITÉ EXPONENTIELLE

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

Plan

Recherche de patron

- Problématique

- Rabin Karp

- Automate FSM

PLSC

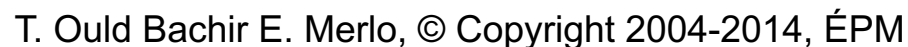
- Problématique

- Solution par programmation dynamique

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 $X[n] \neq Y[m]$,
 - alors si $Z[k] \neq X[n]$, $Z[1..k]$ est PLSC de $X[1..n-1]$ $Y[1..m]$
 - Si $X[n] \neq Y[m]$,
 - alors si $Z[k] \neq Y[m]$, $Z[1..k]$ est PLSC de $X[1..n]$ $Y[1..m-1]$

La formulation récursive donne l'arbre d'exécution suivant:

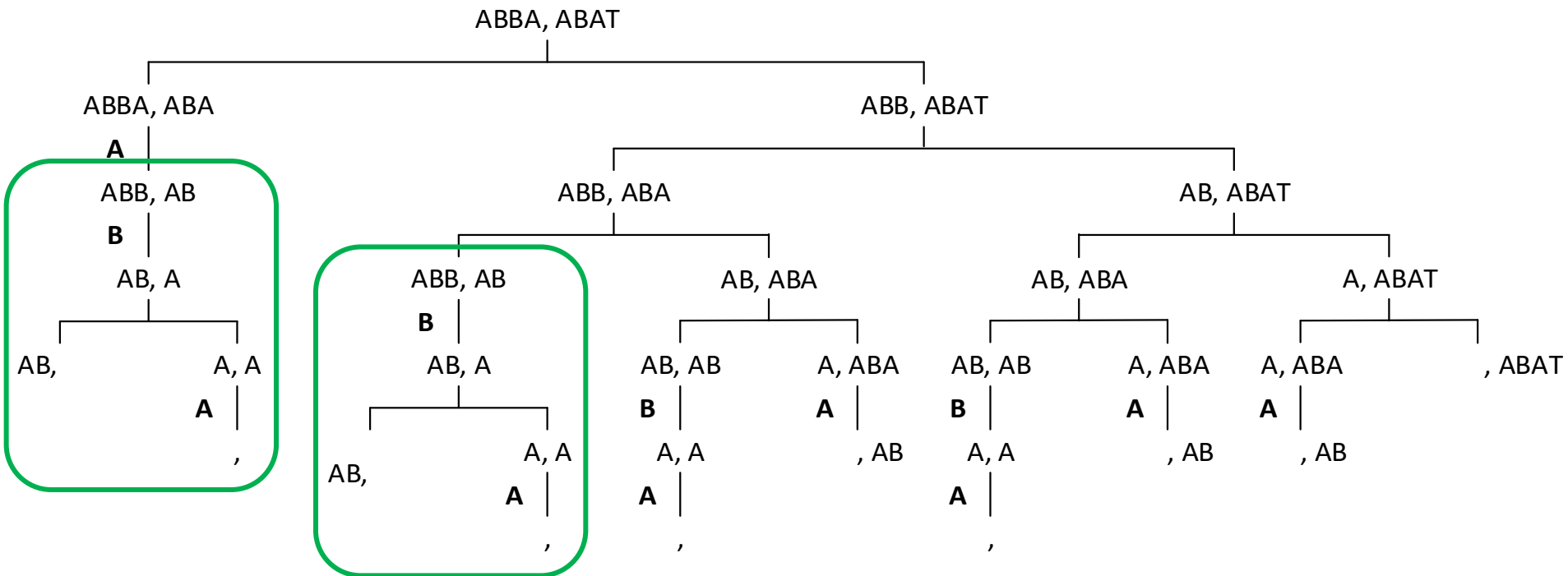


La formulation récursive donne l'arbre d'exécution suivant:



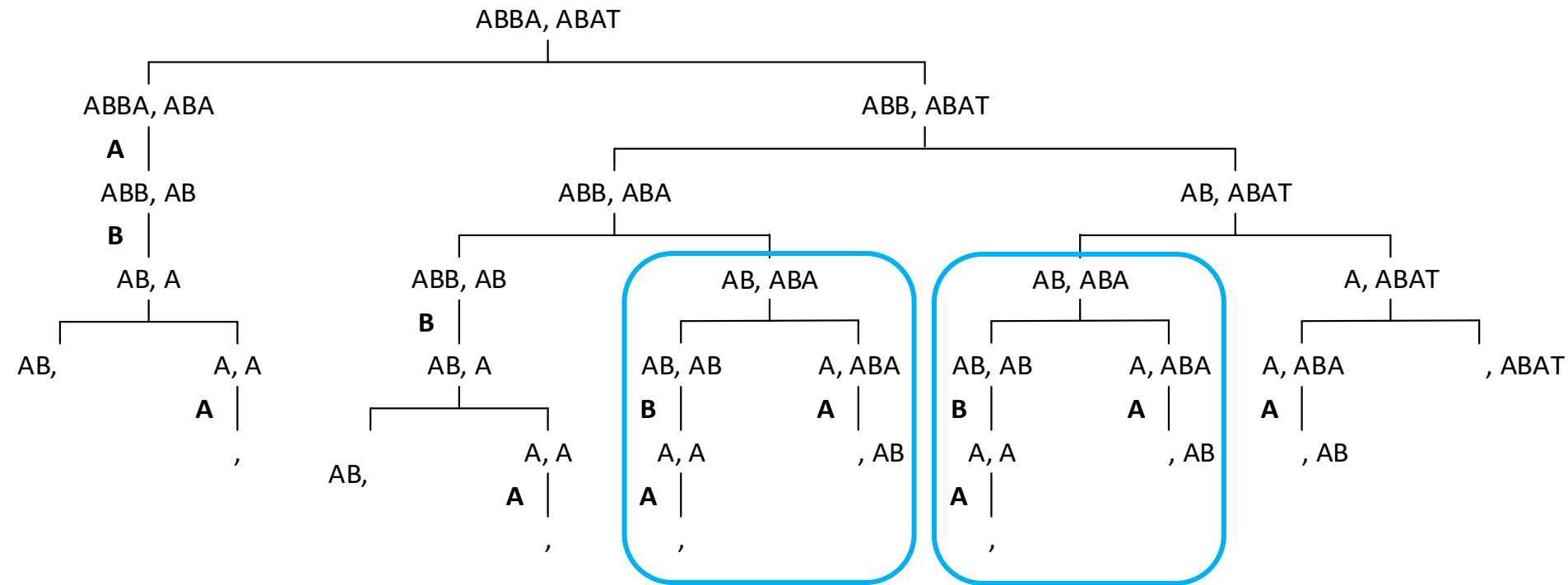
Solution par programmation dynamique

La formulation récursive donne l'arbre d'exécution suivant:



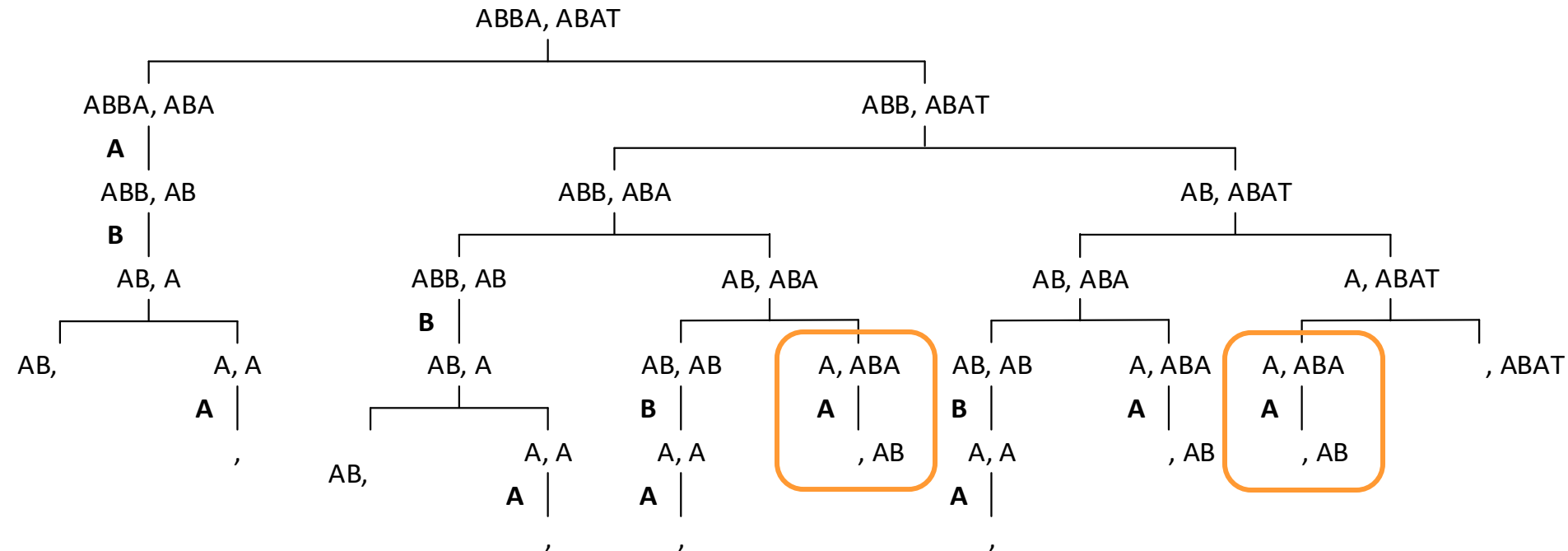
Solution par programmation dynamique

La formulation récursive donne l'arbre d'exécution suivant:



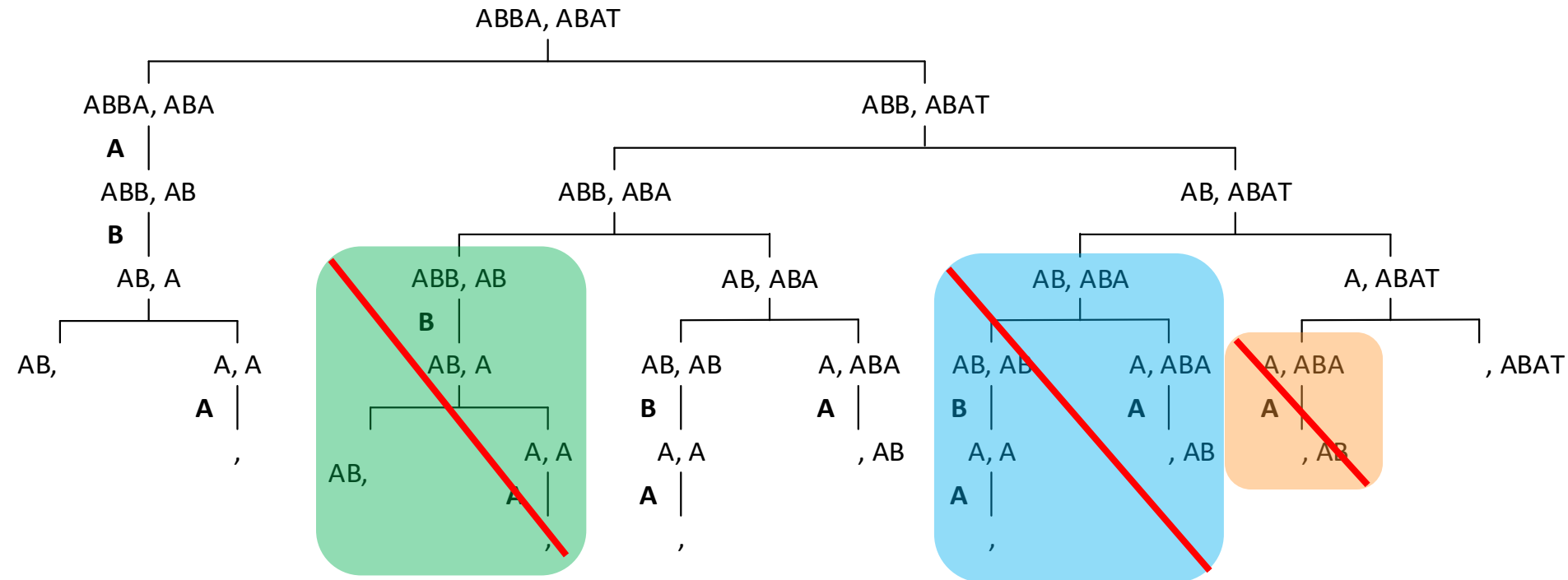
Solution par programmation dynamique

La formulation récursive donne l'arbre d'exécution suivant:



Solution par programmation dynamique

La solution par programmation dynamique consiste à construire l'arbre de bas vers le haut, et de réutiliser les résultats redondants:



Solution par programmation dynamique

- Il est alors possible de formuler un algorithme de **programmation dynamique**:
 - On définit deux tables 2-D auxiliaires d (direction) et t (taille)
 - t a une taille $(n+1) \times (m+1)$ (initialisée à 0)
 - d a une taille $n \times m$
 - 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 $X[1..i]$, $Y[1..j]$

Solution par programmation dynamique

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

Si $X[i] == Y[j]$

$t[i+1, j+1] = t[i, j] + 1$
 $d[i, j] = \text{DIAG} (\swarrow)$

Sinon Si $t[i, j+1] \geq t[i+1, j]$

$t[i+1, j+1] = t[i, j+1]$
 $d[i, j] = \text{HAUT} (\uparrow)$

Sinon

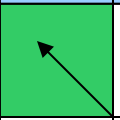
$t[i+1, j+1] = t[i+1, j]$
 $d[i, j] = \text{GAUCHE} (\leftarrow)$

Solution par programmation dynamique

<i>d</i>	A	B	A	T
A				
B				
B				
A				

<i>t</i>		A	B	A	T
	0	0	0	0	0
A	0				
B	0				
B	0				
A	0				

Solution par programmation dynamique

<i>d</i>	A	B	A	T
A				
B				
B				
A				

<i>t</i>		A	B	A	T
	0	0	0	0	0
A	0	1			
B	0				
B	0				
A	0				

Solution par programmation dynamique

<i>d</i>	A	B	A	T
A				
B				
B				
A				

<i>t</i>		A	B	A	T
	0	0	0	0	0
A	0	1			
B	0				
B	0				
A	0				

Solution par programmation dynamique

<i>d</i>	A	B	A	T
A				
B				
B				
A				

<i>t</i>		A	B	A	T
	0	0	0	0	0
A	0	1			
B	0				
B	0				
A	0				

Solution par programmation dynamique


<i>d</i>	A	B	A	T
A				
B				
B				
A				

Diagram illustrating a sequence alignment problem. The sequence *d* (A, B, A, T) is aligned with the sequence A, B, A, T. The alignment is shown by arrows: one arrow points from the first 'A' in *d* to the first 'A' in the sequence, and another arrow points from the first 'B' in *d* to the first 'B' in the sequence. The cell containing 'B' in the sequence is highlighted in green.

<i>t</i>		A	B	A	T
	0	0	0	0	0
A	0	1	1		
B	0				
B	0				
A	0				



Diagram illustrating a sequence alignment problem. The sequence *t* (A, B, A, T) is aligned with the sequence A, B, A, T. The alignment is shown by arrows: one arrow points from the first 'A' in *t* to the first 'A' in the sequence, and another arrow points from the first 'B' in *t* to the first 'B' in the sequence. The cell containing 'B' in the sequence is highlighted in green.

Solution par programmation dynamique

<i>d</i>	A	B	A	T
A				
B				
B				
A				




<i>t</i>		A	B	A	T
	0	0	0	0	0
A	0	1	1		
B	0				
B	0				
A	0				

Solution par programmation dynamique

<i>d</i>	A	B	A	T
A				
B				
B				
A				

<i>t</i>		A	B	A	T
	0	0	0	0	0
A	0	1	1	1	
B	0				
B	0				
A	0				

Solution par programmation dynamique

<i>d</i>	A	B	A	T
A				
B				
B				
A				



<i>t</i>		A	B	A	T
	0	0	0	0	0
A	0	1	1	1	
B	0				
B	0				
A	0				

Solution par programmation dynamique

<i>d</i>	A	B	A	T
A	↖	←	↖	
B				
B				
A				

<i>t</i>		A	B	A	T
	0	0	0	0	0
A	0	1	1	1	
B	0				
B	0				
A	0				

Solution par programmation dynamique

<i>d</i>	A	B	A	T
A				
B				
B				
A				

<i>t</i>		A	B	A	T
	0	0	0	0	0
A	0	1	1	1	1
B	0				
B	0				
A	0				

Solution par programmation dynamique

<i>d</i>	A	B	A	T
A				
B				
B				
A				

<i>t</i>		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				

Solution par programmation dynamique

<i>d</i>	A	B	A	T
A				
B				
B				
A				

<i>t</i>		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				

Solution par programmation dynamique

<i>d</i>	A	B	A	T
A				
B				
B				
A				

<i>t</i>		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

Solution par programmation dynamique

<i>d</i>	A	B	A	T
A	↖	←	↖	←
B	↑	↖	←	←
B	↑	↖	↑	↑
A	↖	↑	↖	←

<i>t</i>		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

Solution par programmation dynamique

<i>d</i>	A	B	A	T
A	↖	←	↖	←
B	↑	↖	←	←
B	↑	↖	↑	↑
A	↖	↑	↖	←

<i>t</i>		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

Solution par programmation dynamique

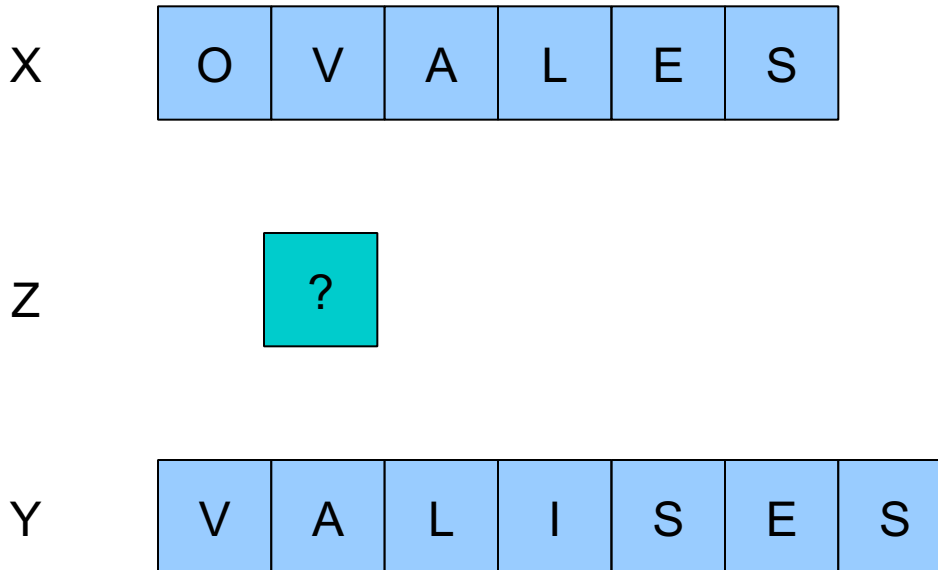
<i>d</i>	A	B	A	T
A	↖	←	↖	←
B	↑	↖	←	←
B	↑	↖	↑	↑
A	↖	↑	↖	←

<i>t</i>		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

Solution par programmation dynamique

- Appliquer l'algorithme pour l'exemple suivant:



Solution par programmation dynamique

<i>d</i>	O	V	A	L	E	S
V	↑	↖	←	←	←	←
A	↑	↑	↖	←	←	←
L	↑	↑	↑	↖	←	←
I	↑	↑	↑	↑	↑	↑
S	↑	↑	↑	↑	↑	↖
E	↑	↑	↑	↑	↖	↑
S	↑	↑	↑	↑	↑	↖

<i>t</i>		O	V	A	L	E	S
	0	0	0	0	0	0	0
V	0	0	1	1	1	1	1
A	0	0	1	2	2	2	2
L	0	0	1	2	3	3	3
I	0	0	1	2	3	3	3
S	0	0	1	2	3	3	4
E	0	0	1	2	3	4	4
S	0	0	1	2	3	4	5