Algorithmes vectoriels pour la recherche approchée

Nadia El-Mabrouk

Problème

- INPUT:
  - Un mot P de taille m
  - Un texte T de taille $n >> m$
  - Un nombre d'erreurs autorisées $k << m$
- OUTPUT: Toutes les occurrences de P dans T à k erreurs près.
- Programmation dynamique: $O(mn)$
- Différentes améliorations: $O(kn)$

Méthodes numériques – Algorithmes vectoriels

Basées sur une représentation en bits des états de la recherche. Dans ce cas, une approche naive est considérée, et rendue rapide grâce à la capacité des langages de programmation à manipuler les mots machine (opérations logiques et arithmétiques simples)

Algorithme vectoriel: Algorithme qui trouve un vecteur de sortie en appliquant un nb d'opérations sur les vecteurs d'entrée, indépendant de la longueur des vecteurs.

Mot machine: Vecteur binaire de 16, 32, 64 bits

0110001101001110

Opérations unitaires sur les mots machine: $x \vee y$, $x \wedge y$, $\neg x$, $x << a$ (déplacement ver la gauche), $x >> a$ (déplacement vers la droite).

Algorithme Shift-Or pour la recherche exacte

Recherche exacte d'un mot *P* de taille *m* dans un texte *T* de taille *n*.

Pour toute position *j* dans *T*, E_j est un vecteur de taille *m* tel que:

$$\text{Pour tout } i,\ 1 \leq i \leq m, \qquad E_j[i] = \begin{cases} 0 & \text{si } P[1:i] = T[j-i+1:j] \\ 1 & \text{sinon} \end{cases}$$

Occurrence de *P* dans *T* se terminant à la position *j* ssi $E_j[m] = 0$

Algorithme Shift-Or pour la recherche exacte

Phase de prétraitement: Pour tout $a \in \Sigma$, calculer le vecteur S_a:

$$\text{Pour tout } i,\ 1 \leq i \leq m, \quad S_a[i] = \begin{cases} 0 & \text{si } a = p_i \\ 1 & \text{sinon} \end{cases}$$

Alors: $E_{j+1} = (E_j << b) \vee S_{t_{j+1}}$ Ici b=1

Si $m \leq w$ (32 ou 63), complexité $O(n)$ en temps et $O(1)$ en espace.

a	b	d	a	b	b
1	**1**	**1**	**0**	**1**	

				a
			a	b
		a	b	b
	a	b	b	a
a	b	b	a	c

a	b	d	a	b	b	
	1	**1**	**0**	**1**	**0**	
					a	1
				a	b	0
			a	b	b	0
		a	b	b	a	1
	a	b	b	a	c	1

a	b	d	a	b	b	
	1	**1**	**0**	**1**	**0**	
					a	1
				a	b	0
			a	b	b	0
		a	b	b	a	1
	a	b	b	a	c	1
	1	**1**	**0**	**1**	**1**	

	1	**1**	**0**	**1**	**0**
V					
	1	1	0	0	1
=	**1**	**1**	**0**	**1**	**1**

- Exemple complet;

$\Sigma = \{a, b, c, d\}$, $P = abbac$, $T = abbabbac$

x	a	b	c	d
S_x	10110	11001	01111	11111

États successifs de la recherche dans T :

T		**a**	**b**	**b**	**a**	**b**	**b**	**a**	**c**
E_j	11111	11110	11101	11011	10110	11101	11011	10110	01111
									*

$$E_{j+1} = (E_j << b) \vee S_{t_{j+1}}$$

```
ALGORITHM SHIFT-OR (t, p)
SI (|p| > WORD)
    Error ("Use pattern size ≤ word size");

/* Preprocessing */
POUR (i = 0 to |Σ|) FAIRE
    T[i] = ¬0;
lim = 0; j = 1
POUR (chaque carac. p_c de p de dte. a gch.) FAIRE
    T[p_c] = T[p_c] ∧ ¬j
    lim = lim ∨ j;
    j = j << B;
FIN POUR
lim = ¬ (lim >> B); /* lim: 10000 Dernier Bit utilise*/

/* Recherche */
Match = 0;
Init = ¬0;
Etat = Init;
TANTQUE (pas fin de t)
    T[t_c]: Prochain caractere du texte;
    Etat = (Etat << B) ∨ T[t_c];
    SI Etat ≤ lim, Match = Match+1;
FIN TANT QUE
RETOURNE (Match)
```

Recherche avec mismatchs - Algorithme Shift-Add:

(*Baeza-Yates-Gonnet, 1992*)

Recherche de P dans T avec au plus k mismatch.

Pour toute pos. j de T, vecteur E_j de taille m tel que $E_j[i]$ contient le nombre d'erreurs entre $P[1..i]$ et $T[j-i+1..j]$.

Occurrence de P finissant à la position j ssi $E_j[m] \leq k$.

Phase de prétraitement: Pour tout $a \in \Sigma$, calculer le vecteur S_a:

$$\text{Pour tout } i,\ 1 \leq i \leq m, \quad S_a[i] = \begin{cases} 0 & \text{si } a = p_i \\ 1 & \text{sinon} \end{cases}$$

Alors, $E_{j+1}[i] = E_j[i-1] \ + \ S_{t_{j+1}}[i]$

					5	6			
T:	a	b	d	a	b	b	a	b	b
E[5]:	2	4	2	0	1				
					a				
				a	b				
			a	b	b				
		a	b	b	a				
P:	a	b	b	a	c				

					5	6			
T:	a	b	d	a	b	b	a	b	b
			4	2	0	1	0		
						a			
					a	b			
				a	b	b			
			a	b	b	a			

					5	6			
T:	a	b	d	a	b	b	a	b	b
		4	2	0	1	0			
						a	1		
					a	b	0		
				a	b	b	0		
			a	b	b	a	1		
		a	b	b	a	c	1		
							Sb		

	4	2	0	1	0
+	1	1	0	0	1
E[6]:	5	3	0	1	1

b : Nombre de bits nécessaires pour représenter chaque état individuel.

Si E_j et S_a codés chacun sur un mot machine:

$$E_{j+1} = (E_j << b) + S_{t_{j+1}}$$

Valeurs possibles de $E_j[i]$: $0, \cdots, m \Longrightarrow b = \lceil \log_2(m+1) \rceil$

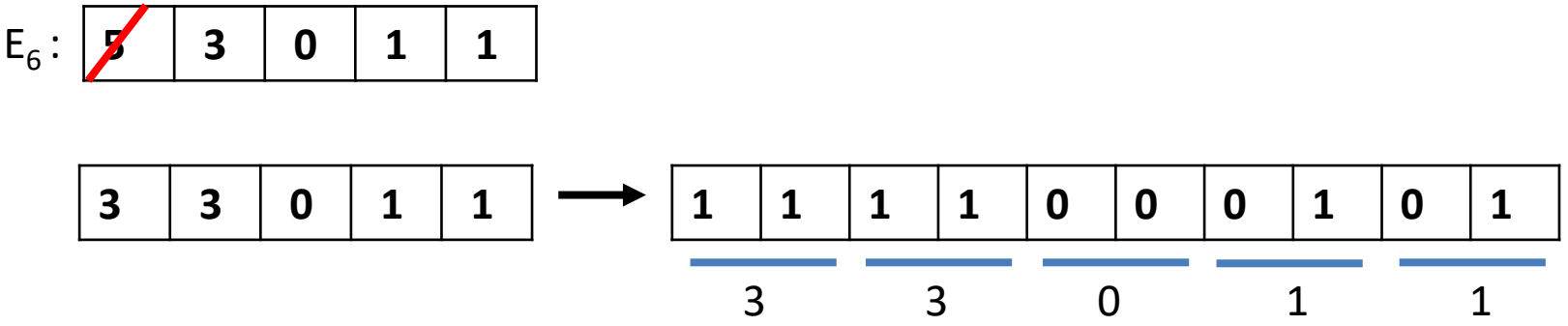
En fait, il suffit de représenter les valeurs de 0 à $k+1$

Par exemple, pour le vecteur E_6 = 5 3 0 1 1 de la page précédente, si *k=1*, on ne devrait pas avoir à garder la valeur 5. Et 3?

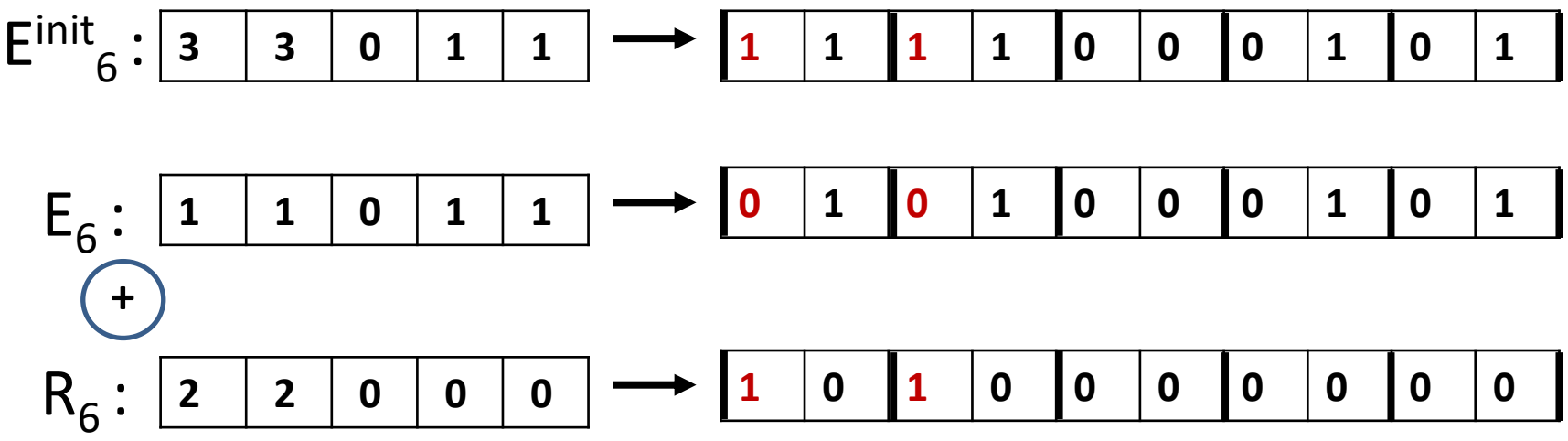
En fait on a besoin de garder $b = \lceil \log_2(k+1) \rceil + 1$

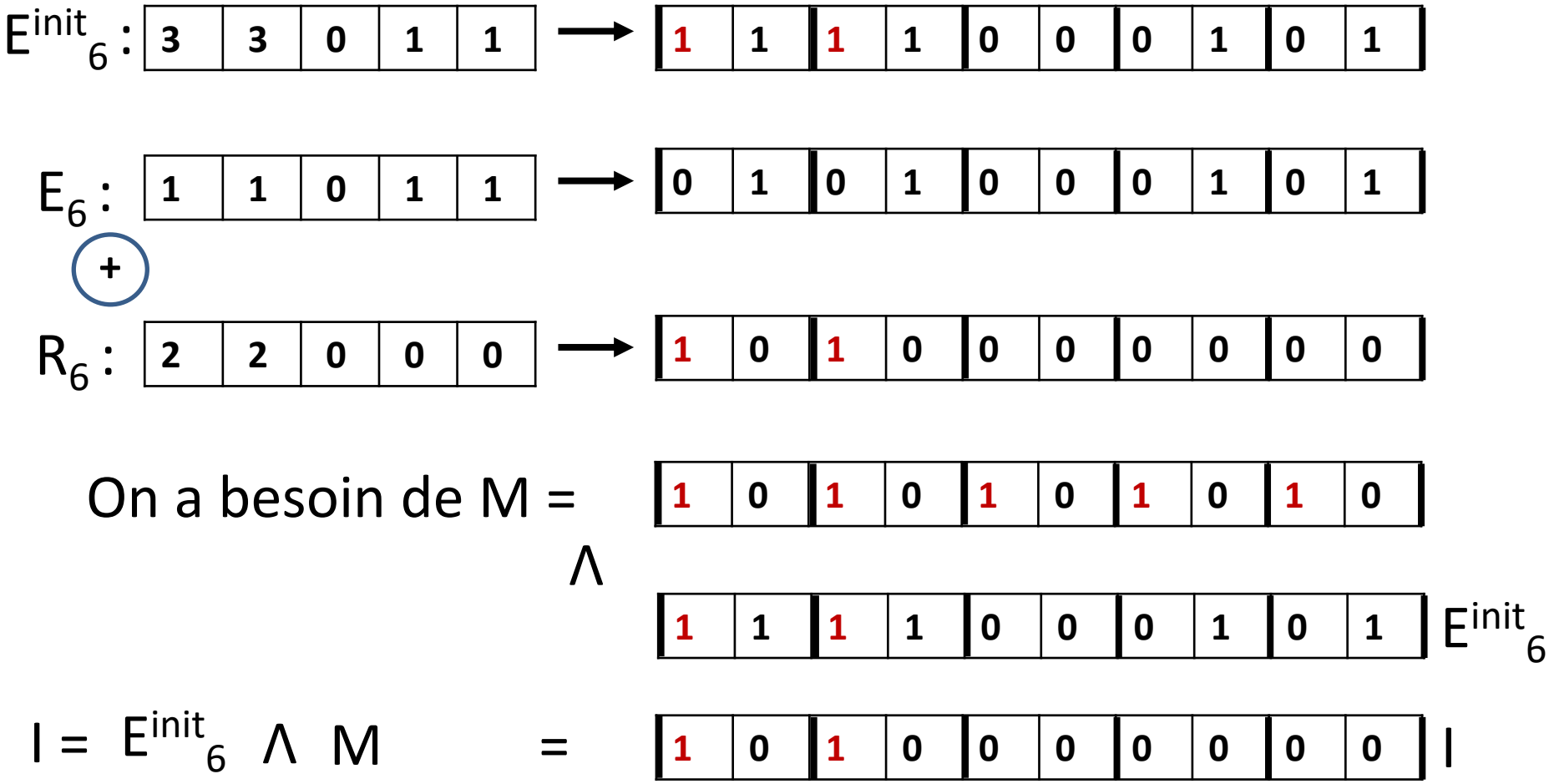
Pour k=1, $b = \lceil \log_2(k+1) \rceil + 1$ = 2 bits : permet de représenter les valeurs de 0 à 3. Si la taille m de P est 5, alors, E_j pour tout j est de taille 5.2=10

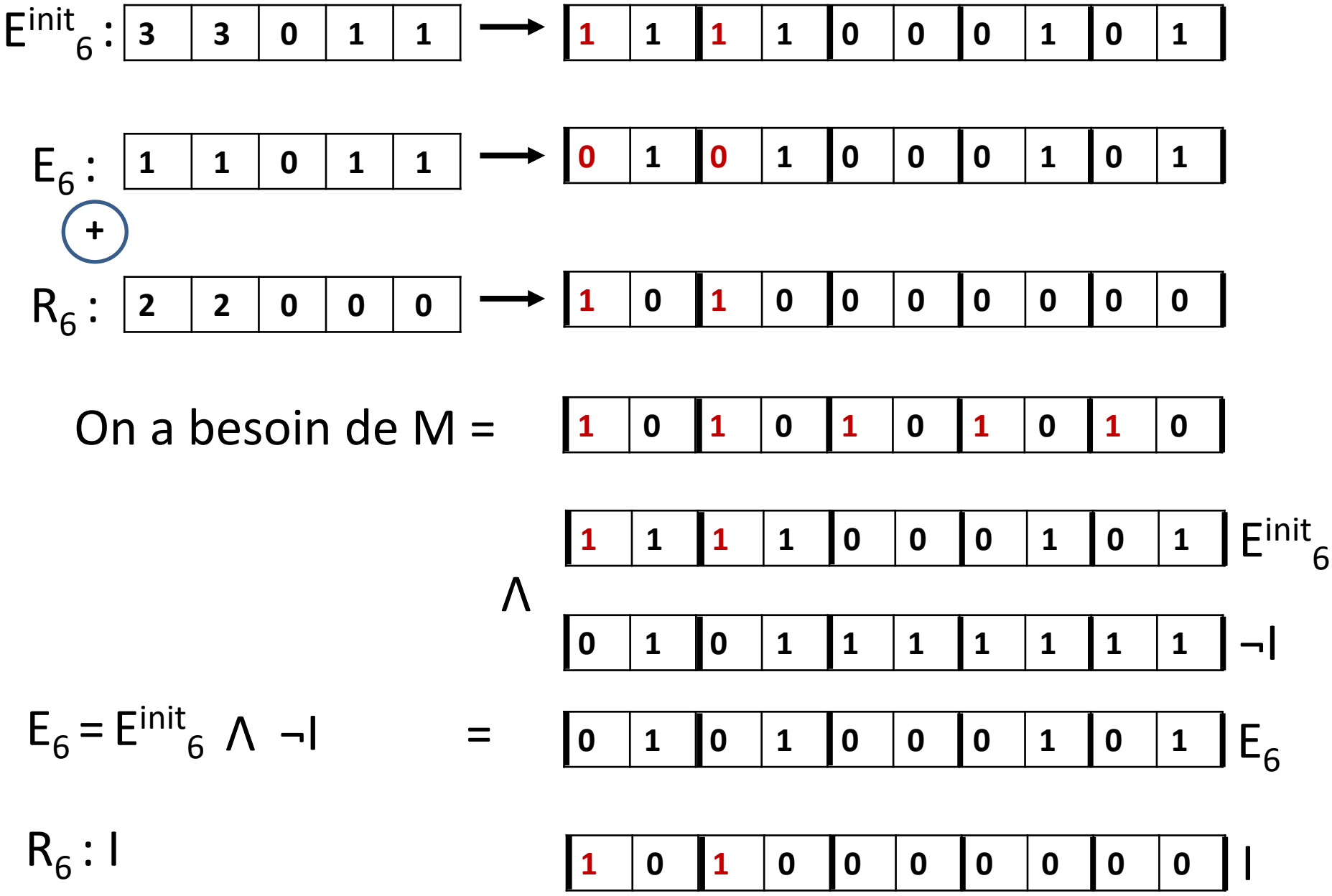
Donc pour E_6 = 5 3 0 1 1 , on a besoin de garder les valeurs de 0 à 3.



En fait on a besoin d'un vecteur « retenue » des additions :




E^{init}_6 : 3 3 0 1 1
1 1 1 1 0 0 0 1 0 1
E_6 : 1 1 0 1 1
0 1 0 1 0 0 0 1 0 1
+
R_6 : 2 2 0 0 0
1 0 1 0 0 0 0 0 0 0
On a besoin de M =
1 0 1 0 1 0 1 0 1 0
Λ
1 1 1 1 0 0 0 1 0 1
E^{init}_6
I = E^{init}_6 Λ M =
1 0 1 0 0 0 0 0 0 0
I


E^{init}_6 : 3 3 0 1 1
1 1 1 1 0 0 0 1 0 1
E_6 : 1 1 0 1 1
0 1 0 1 0 0 0 1 0 1
+
R_6 : 2 2 0 0 0
1 0 1 0 0 0 0 0 0 0
On a besoin de M =
1 0 1 0 1 0 1 0 1 0
1 1 1 1 0 0 0 1 0 1
E^{init}_6
Λ
0 1 0 1 1 1 1 1 1 1
¬I
$E_6 = E^{init}_6$ Λ ¬I =
0 1 0 1 0 0 0 1 0 1
E_6
R_6 : I
1 0 1 0 0 0 0 0 0 0
I

Soit M le vecteur de $m.b$ bits: $2^{b-1}2^{b-1}\cdots 2^{b-1}$.

À chaque position j dans T, les vecteurs E_{j+1} et R_{j+1} sont calculés à partir de E_j et R_j de la façon suivante:

- $E_{j+1} = (E_j << b) + S_{t_{j+1}}$;
- $I = E_{j+1} \wedge M$;
- $E_{j+1} = E_{j+1} \wedge \neg I$;
- $R_{j+1} = (R_j << b) \vee I$

Exemple:

$\Sigma = \{a, b, c, d\}$, $P = abbac$ et $k = 1$.

Alors $b = \lceil \log_2(2) \rceil + 1 = 2$. Table S:

x	a	b	c	d
S_x	10110	11001	01111	11111

États successifs lors de la recherche de P dans le texte $T = abdabbabbac$:

T		**a**	**b**	**d**	**a**	**b**	**b**	**a**	**b**	**b**	**a**	**c**
E_j	00000	10110	10101	10101	11100	00001	11011	00000	11001	01011	00000	01111
R_j	22222	22220	22200	22020	20220	22200	22000	20220	02200	22000	20220	02200
									*			*

a	b	d	a	b
2	**4**	**2**	**0**	**1**

				a
			a	b
		a	b	b
	a	b	b	a
a	b	b	a	c
E_5 : **0**	**0**	**0**	**0**	**1**
+				
R_5 : **2**	**2**	**2**	**0**	**0**
=				
2	2	2	0	1

a	b	d	a	b	a
	5	**2**	**1**	**2**	**0**
					a
				a	b
			a	b	b
		a	b	b	a
	a	b	b	a	c
E_6 :	1	0	1	0	0
R_6 :	2	2	0	2	0

Comment les calculer à partir de E_5 et R_5?

	a	b	d	a	b	a
					a
				a	b
			a	b	b
		a	b	b	a
	a	b	b	a	c
E_5 :	0	0	0	0	1
R_5 :	2	2	2	0	0

$$E_{j+1} = (E_j << b) + S_{t_{j+1}};$$
$$I = E_{j+1} \wedge M;$$
$$E_{j+1} = E_{j+1} \wedge \neg I;$$
$$R_{j+1} = (R_j << b) \vee I$$

a	b	d	a	b	a	
					a	0
				a	b	1
			a	b	b	1
		a	b	b	a	0
	a	b	b	a	c	1

E_5 : 0 0 0 1 0 ←

$$E_{j+1} = (E_j << b) + S_{t_{j+1}}; \leftarrow$$
$$I = E_{j+1} \wedge M;$$
$$E_{j+1} = E_{j+1} \wedge \neg I;$$
$$R_{j+1} = (R_j << b) \vee I$$

a	b	d	a	b	a	
					a	0
				a	b	1
			a	b	b	1
		a	b	b	a	0
	a	b	b	a	c	1

E_5 : 0 0 0 1 0 ←

\+

S_6 : 1 0 1 1 0

$$E_{j+1} = (E_j << b) + S_{t_{j+1}}; \leftarrow$$

$$I = E_{j+1} \wedge M;$$

$$E_{j+1} = E_{j+1} \wedge \neg I;$$

$$R_{j+1} = (R_j << b) \vee I$$

a	b	d	a	b	a	
					a	0
				a	b	1
			a	b	b	1
		a	b	b	a	0
	a	b	b	a	c	1

E_5 :		0	0	0	1	0 ←
+						
S_6 :		1	0	1	1	0
=						
E_6 :		1	0	1	2	0

$$E_{j+1} = (E_j << b) + S_{t_{j+1}}; \leftarrow$$
$$I = E_{j+1} \wedge M;$$
$$E_{j+1} = E_{j+1} \wedge \neg I;$$
$$R_{j+1} = (R_j << b) \vee I$$

a	b	d	a	b	a	
					a	0
				a	b	1
			a	b	b	1
		a	b	b	a	0
	a	b	b	a	c	1

E_6 : 1 0 1 2 0

$$E_{j+1} = (E_j << b) + S_{t_{j+1}};$$
$$I = E_{j+1} \wedge M;$$
$$E_{j+1} = E_{j+1} \wedge \neg I;$$
$$R_{j+1} = (R_j << b) \vee I$$

a	b	d	a	b	a	
					a	0
				a	b	1
			a	b	b	1
		a	b	b	a	0
	a	b	b	a	c	1

E_6 : 1 0 1 2 0

E_6 :

0	1	0	0	0	1	**1**	0	0	0

∧

M :

1	**0**	**1**	**0**	**1**	**0**	1	**0**	**1**	**0**

I =

0	0	0	0	0	0	**1**	0	0	0

$$E_{j+1} = (E_j << b) + S_{t_{j+1}};$$
$$I = E_{j+1} \wedge M;$$
$$E_{j+1} = E_{j+1} \wedge \neg I;$$
$$R_{j+1} = (R_j << b) \vee I$$

a b d a b a

				a	0
			a	b	1
		a	b	b	1
	a	b	b	a	0
a	b	b	a	c	1

I =

0	0	0	0	0	0	**1**	0	0	0

E_6 : 1 0 1 2 0

$$E_{j+1} = (E_j << b) + S_{t_{j+1}};$$
$$I = E_{j+1} \wedge M;$$
$$E_{j+1} = E_{j+1} \wedge \neg I;$$
$$R_{j+1} = (R_j << b) \vee I$$

a	b	d	a	b	a	
					a	0
				a	b	1
			a	b	b	1
		a	b	b	a	0
	a	b	b	a	c	1

E_6 : 1 0 1 2 0

I = | 0 | 0 | 0 | 0 | 0 | 0 | **1** | 0 | 0 | 0 |

¬I = | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |

$$E_{j+1} = (E_j << b) + S_{t_{j+1}};$$
$$I = E_{j+1} \wedge M;$$
$$E_{j+1} = E_{j+1} \wedge \neg I;$$
$$R_{j+1} = (R_j << b) \vee I$$

a	b	d	a	b	a	
					a	0
				a	b	1
			a	b	b	1
		a	b	b	a	0
	a	b	b	a	c	1

E_6 : 1 0 1 **0** 0

E_6 : | 0 | 1 | 0 | 0 | 0 | 1 | **1** | 0 | 0 | 0 |

∧

¬I = | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |

E_6 : | 0 | 1 | 0 | 0 | 0 | 1 | **0** | 0 | 0 | 0 |

$$E_{j+1} = (E_j << b) + S_{t_{j+1}};$$
$$I = E_{j+1} \wedge M;$$
$$E_{j+1} = E_{j+1} \wedge \neg I;$$
$$R_{j+1} = (R_j << b) \vee I$$

a	b	d	a	b	a	
					a	0
				a	b	1
			a	b	b	1
		a	b	b	a	0
	a	b	b	a	c	1

E_6 : 1 0 1 0 0

R_5 : 2 2 2 0 0

$$E_{j+1} = (E_j << b) + S_{t_{j+1}};$$
$$I = E_{j+1} \wedge M;$$
$$E_{j+1} = E_{j+1} \wedge \neg I;$$
$$R_{j+1} = (R_j << b) \vee I$$

a b d a b a

				a	0
			a	b	1
		a	b	b	1
	a	b	b	a	0
a	b	b	a	c	1

R_6 :

1	0	1	0	0	0	0	0	0	0

V

I =

0	0	0	0	0	0	1	0	0	0

R_6 :

1	0	1	0	0	0	**1**	0	0	0

E_6 : 1 0 1 0 0

R_6 : 2 2 0 0 0

$$E_{j+1} = (E_j << b) + S_{t_{j+1}};$$

$$I = E_{j+1} \wedge M;$$

$$E_{j+1} = E_{j+1} \wedge \neg I;$$

$$R_{j+1} = (R_j << b) \vee I$$

a	b	d	a	b	a	
					a	0
				a	b	1
			a	b	b	1
		a	b	b	a	0
	a	b	b	a	c	1

E_6 : 1 0 1 0 0

R_6 : 2 2 0 **2** 0

R_6 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

V

I = | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

R_6 : | 1 | 0 | 1 | 0 | 0 | 0 | **1** | 0 | 0 | 0 |

$$E_{j+1} = (E_j << b) + S_{t_{j+1}};$$
$$I = E_{j+1} \wedge M;$$
$$E_{j+1} = E_{j+1} \wedge \neg I;$$
$$R_{j+1} = (R_j << b) \vee I$$

a	b	d	a	b	a	
	5	**2**	**1**	**2**	**0**	
					a	0
				a	b	1
			a	b	b	1
		a	b	b	a	0
	a	b	b	a	c	1

E_6 : 1 0 1 0 0

R_6 : 2 2 0 **2** 0

R_6 V I =

R_6	1	0	1	0	0	0	0	0	0	0
I =	0	0	0	0	0	0	1	0	0	0
R_6 :	1	0	1	0	0	0	**1**	0	0	0

$$E_{j+1} = (E_j << b) + S_{t_{j+1}};$$
$$I = E_{j+1} \wedge M;$$
$$E_{j+1} = E_{j+1} \wedge \neg I;$$
$$R_{j+1} = (R_j << b) \vee I$$

Complexité:

$|\Sigma| = c$.

$O(\lceil \frac{mb}{\omega} \rceil)$: temps nécessaire pour effectuer un nombre constant d'opérations sur un état de longueur mb représenté sur un mot machine de longueur ω.

- Calcul de S: temps $O\left(\lceil \frac{mb}{\omega} \rceil (m + c)\right)$ et espace $O(\lceil \frac{mb}{\omega} \rceil c)$.
- Phase de recherche: temps dans le pire des cas et en moyenne en $O(\lceil \frac{mb}{\omega} \rceil n)$

Lorsque $mb \leq \omega$, complexité totale en temps en $O(n + m + c)$ et complexité en espace en $O(c)$

Lorsque le nombre de mots machine utilisé n'est pas trop grand, Shift-Add reste efficace en pratique.

Le cours s’arrête ici.

La suite est juste pour vous, mais ne comptera pas à l’examen

Algorithme Shift-And pour la recherche avec erreurs:

Utilisé dans le programme *agrep* de UNIX.

Phase de prétraitement: Pour tout $a \in \Sigma$, calculer C_a: inverse (voir Hirschberg) du vecteur caractéristique de a dans P.

Exemple: Si $P = abbac$, alors $C_a = 01001$, $C_b = 00110$, $C_c = 10000$ et $C_d = 00000$.

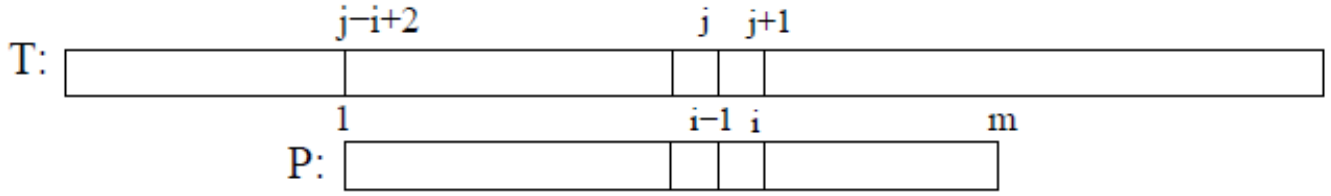
Phase de recherche: Utilise $k+1$ vecteurs $E_j^0, E_j^1, \cdots E_j^k$:

$$\text{Pour tout } i, \quad E_j^d[i] = \begin{cases} 1 & \text{si } P[1:i] = T[j-i+1:j] \text{ à } d \text{ erreurs près} \\ 0 & \text{sinon} \end{cases}$$

Exemple: Si $T = abcbcabba$ et $P = abbac$, alors $E_5^1 = 00001$ et $E_5^2 = 10111$

Initialisation: Pour tout d, $E_0^d = 0^{m-d}1^d$

Pour que $P[1..i]$ s'aligne avec $T[j-i+2..j+1]$ à d erreurs près, 4 cas possibles:



- $P[1..i-1]$ s'aligne avec $T[j-i+2..j]$ avec au plus d erreurs, et $P_i = T_{j+1} \longrightarrow E^d_{j+1} = E^d_j << 1 \ \wedge \ C_{T_{j+1}}$
- $P[1..i-1]$ s'aligne avec $T[j-i+2..j]$ avec au plus $d-1$ erreurs (cas d'un match ou d'un mismatch) $\longrightarrow E^d_{j+1} = E^{d-1}_j << 1$
- Suppression de P_i: $P[1..i-1]$ s'aligne avec $T[j-i+2..j+1]$ avec au plus $d-1$ erreurs $\longrightarrow E^d_{j+1} = E^{d-1}_{j+1} << 1$
- Insertion de T_{j+1}: $P[1..i]$ s'aligne avec $T[j-i+2..j]$ avec au plus $d-1$ erreurs $\longrightarrow E^d_{j+1} = E^{d-1}_j$

Et donc, pour tout $d > 0$:

$$\begin{aligned} E^d_{j+1} &= ((E^d_j << 1) \wedge C_{t_{j+1}}) \vee (E^{d-1}_j << 1) \vee \\ &\quad (E^{d-1}_{j+1} << 1) \vee E^{d-1}_j \\ &= ((E^d_j << 1) \wedge C_{T_{j+1}}) \vee \\ &\quad ((E^{d-1}_j \vee E^{d-1}_{j+1}) << 1) \vee E^{d-1}_j \end{aligned}$$

Pour $d = 0$:

$$E^0_{j+1} = (E^0_j << 1) \wedge C_{T_{j+1}}$$

Complexité: La phase de recherche prend un temps en $O(\lceil \frac{m}{\omega} \rceil kn)$

Lorsque $m \leq \omega$, temps total de Shift-And en $O(kn + m + c)$ et complexité en espace en $O(c)$.

Exemple:

On recherche `annual` dans le texte `annealing` à $k = 2$ erreurs près.

Au départ: $E^0 = 000000$, $E^1 = 000001$, $E^2 = 000011$.

						6	7		
T:	a	n	n	e	a	l	i	n	g
E_6^0:	0	0	0	0	0	0			
E_6^1:	1	0	0	0	1	1			
E_6^2:	1	0	0	1	1	1			
						a			
					a	n			
				a	n	n			
			a	n	n	u			
P:		a	n	n	u	a			
	a	n	n	u	a	l			

						6	7			
T:	a	n	n	e	a	l	i		n	g
		0	0	0	0	0	1			
		0	0	0	1	1	1			
		0	0	1	1	1	1			
							a	0		
						a	n	0		
					a	n	n	0		
				a	n	n	u	0		
P:			a	n	n	u	a	0		
		a	n	n	u	a	l	0		
								S_i		

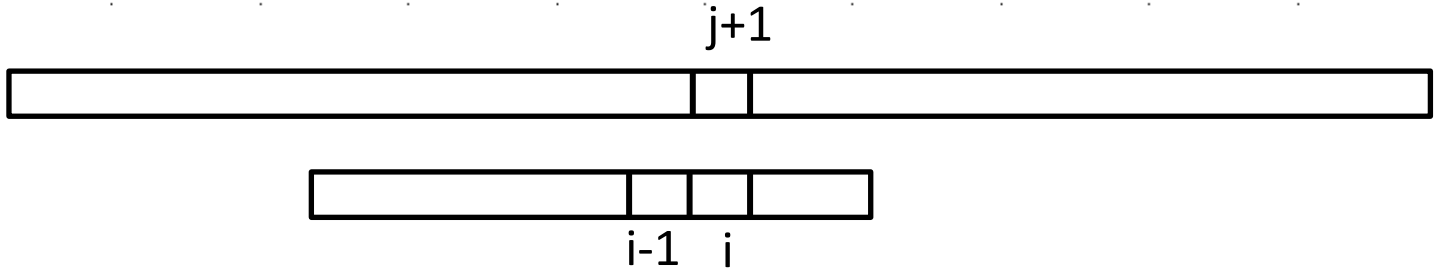
E_7^0 =(000000 << 1) & 000000 = 000000

E_7^1 =((100011 << 1) & 000000) | (000000 << 1) | (000000 << 1)
| 000000 = 000001

E_7^2 =((100111 << 1) & 000000) | (100011 << 1) | (000001 << 1)
| 100011 = 100111

Pour que *P[1..i]* s'aligne avec *T[..j+1]* à *d* erreurs près :

- $P[1..i-1]$ s'aligne avec $T[j-i+2..j]$ avec au plus d erreurs, et $P_i = T_{j+1} \longrightarrow E^d_{j+1} = E^d_j << 1 \ \wedge \ C_{T_{j+1}}$



P =

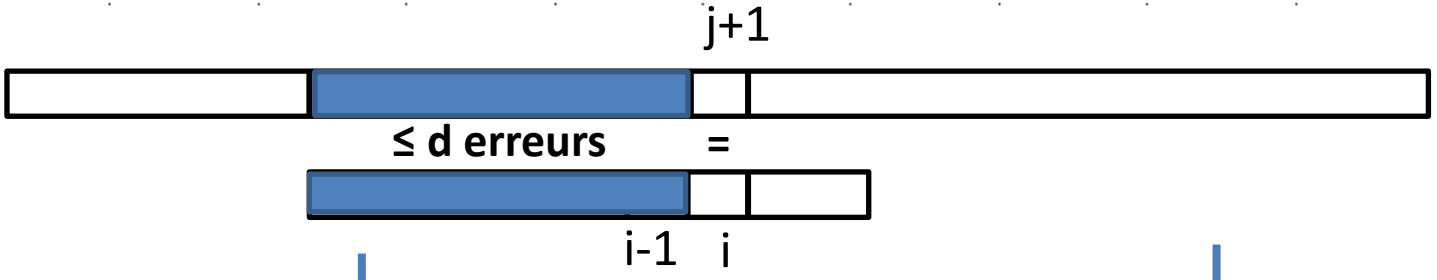
1	2	3	4	5	6
a	n	n	u	a	l

$T_7 = i$ $C_i =$

1	2	3	4	5	6
0	0	0	0	0	0

Pour que *P[1..i]* s'aligne avec *T[..j+1]* à *d* erreurs près :

- $P[1..i-1]$ s'aligne avec $T[j-i+2..j]$ avec au plus d erreurs, et $P_i = T_{j+1} \longrightarrow E^d_{j+1} = E^d_j << 1 \wedge C_{T_{j+1}}$



E_6^1 :

6	5	4	3	2	1
1	0	0	0	1	1

<< 1

6	5	4	3	2	1
0	0	0	1	1	1

∧

C_i =

6	5	4	3	2	1
0	**0**	**0**	**0**	**0**	**0**

E_7^1 :

0	0	0	0	0	0

C_i =

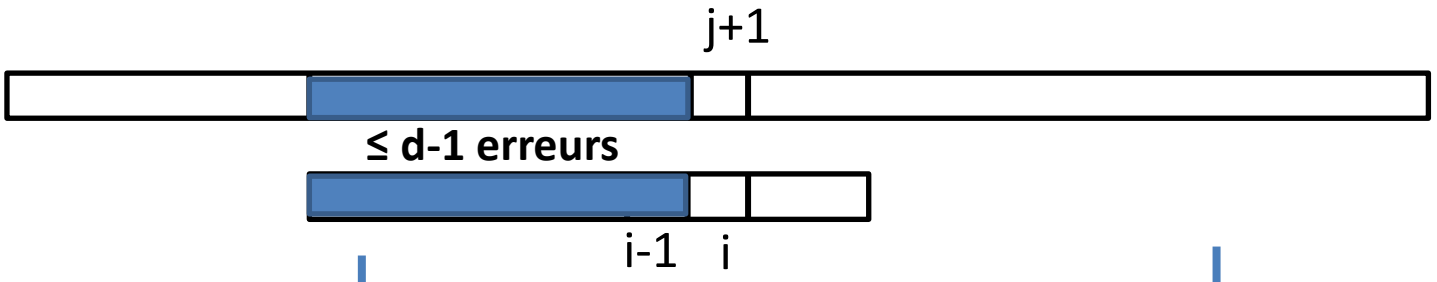
6	5	4	3	2	1
0	**0**	**0**	**0**	**0**	**0**

P =

1	2	3	4	5	6
a	**n**	**n**	**u**	**a**	**l**

Pour que *P[1..i]* s'aligne avec *T[..j+1]* à *d* erreurs près :

- $P[1..i-1]$ s'aligne avec $T[j-i+2..j]$ avec au plus $d-1$ erreurs (cas d'un match ou d'un mismatch) $\longrightarrow E_{j+1}^{d} = E_{j}^{d-1} << 1$



E_6^0 :

6	5	4	3	2	1
0	0	0	0	0	0

E_7^1 :

6	5	4	3	2	1
0	0	0	0	0	1

C_i =

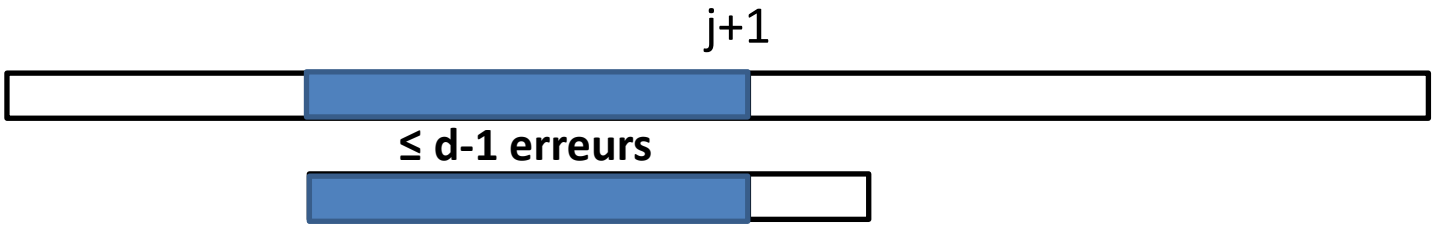
6	5	4	3	2	1
0	**0**	**0**	**0**	**0**	**0**

P =

1	2	3	4	5	6
a	**n**	**n**	**u**	**a**	**l**

Pour que *P[1..i]* s'aligne avec *T[..j+1]* à *d* erreurs près :

- Suppression de P_i: $P[1..i-1]$ s'aligne avec $T[j-i+2..j+1]$ avec au plus $d-1$ erreurs $\longrightarrow E^d_{j+1} = E^{d-1}_{j+1} << 1$



E_7^0 :

6	5	4	3	2	1
0	0	0	0	0	0

E_7^1 :

6	5	4	3	2	1
0	0	0	0	0	1

C_i =

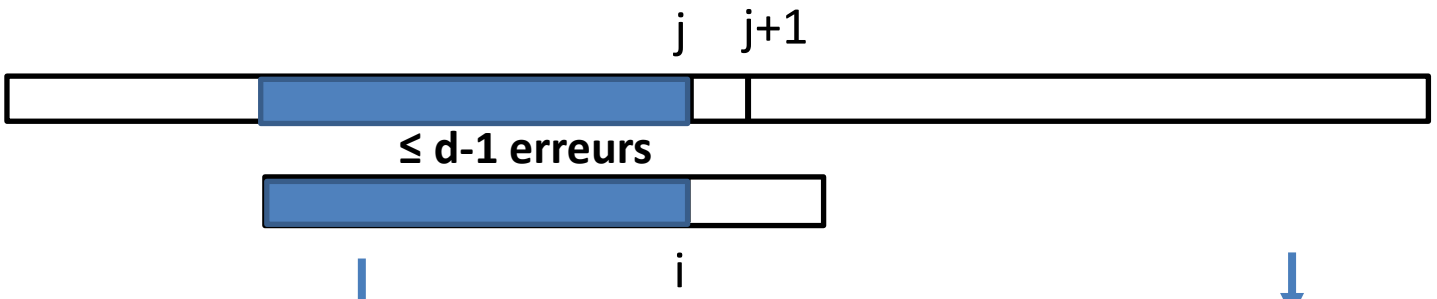
6	5	4	3	2	1
0	**0**	**0**	**0**	**0**	**0**

P =

1	2	3	4	5	6
a	**n**	**n**	**u**	**a**	**l**

Pour que *P[1..i]* s'aligne avec *T[..j+1]* à *d* erreurs près :

- Insertion de T_{j+1}: $P[1..i]$ s'aligne avec $T[j-i+2..j]$ avec au plus $d-1$ erreurs $\longrightarrow E_{j+1}^{d} = E_{j}^{d-1}$



E_6^0 :

6	5	4	3	2	1
0	0	0	0	0	0

E_7^1 :

6	5	4	3	2	1
0	0	0	0	0	0

C_i =

6	5	4	3	2	1
0	**0**	**0**	**0**	**0**	**0**

P =

1	2	3	4	5	6
a	**n**	**n**	**u**	**a**	**l**

Exemple:

On recherche `annual` dans le texte `annealing` à $k = 2$ erreurs près.

Au départ: $E^0 = 000000$, $E^1 = 000001$, $E^2 = 000011$.

						6	7		
T:	a	n	n	e	a	l	i	n	g
E_6^0:	0	0	0	0	0	0			
E_6^1:	1	0	0	0	1	1			
E_6^2:	1	0	0	1	1	1			
						a			
					a	n			
				a	n	n			
			a	n	n	u			
P:		a	n	n	u	a			
	a	n	n	u	a	l			

						6	7			
T:	a	n	n	e	a	l	i		n	g
		0	0	0	0	0	1			
		0	0	0	1	1	1			
		0	0	1	1	1	1			
							a	0		
						a	n	0		
					a	n	n	0		
				a	n	n	u	0		
P:			a	n	n	u	a	0		
		a	n	n	u	a	l	0		
								S_i		

E_7^0 =(000000 << 1) & 000000 = 000000

E_7^1 =((100011 << 1) & 000000) | (000000 << 1) | (000000 << 1)
| 000000 = 000001

E_7^2 =((100111 << 1) & 000000) | (100011 << 1) | (000001 << 1)
| 100011 = 100111