

Algorithmes rapides pour la recherche de motif dans des fenêtres minimales

A. Mathelier et A. Carbone

Equipe Génomique Analytique
UPMC-Paris6 - INSERM-UMRS511

26 septembre 2007



Contexte

- ▶ Recherche de micro-ARN dans des génomes à partir de séquences de micro-ARN déjà découvertes

- ▶ Besoin d'un outil efficace de recherche **exhaustive** de ces séquences

Approximate String Matching

- ▶ Entrées : un texte T de taille n , un motif M de taille m et $k \geq 0$
- ▶ Sortie : les positions de T telles qu'il existe un suffixe incluant M avec au plus k erreurs (insertions, délétions, substitutions).
- ▶ Complexité en $O(\frac{k}{w} \cdot n)$ en utilisant des vecteurs de bits



E. W. Myers.

A fast bit-vector algorithm for approximate string matching based on dynamic programming.
Journal of the ACM, 46(3) :395–415, 1999.

Length of Longest Common Subsequence

- ▶ Entrées : un texte T de taille n , un motif M de taille m
- ▶ Sortie : la taille de la plus longue sous-séquence commune à T et M
- ▶ Complexité en $O(\frac{m}{w} \cdot n)$ en utilisant des vecteurs de bits



M. Crochemore, C. S. Iliopoulos, Y. J. Pinzon, and J. F. Reid.

A fast and practical bit-vector algorithm for the longest common subsequence problem.
In L. Brankovic and J. Ryan, editors, *Proceedings of the 11th Australasian Workshop On Combinatorial Algorithms*, pages 75–86, Hunter Valley, Australia, 2000.

Différences entre minimiser les erreurs et maximiser les matches

AT**ATGAT**
| | | |
ATTAT
minimisation des erreurs

ATATGAT
| | | |
AT T AT
maximisation des matches

ASM dans des fenêtres minimales

- ▶ Entrées : un texte T de taille n , un motif M de taille m , $k \geq 0$ et s la taille maximale d'une fenêtre
- ▶ Sortie : pour chaque position de T , le suffixe de taille minimale (bornée par s) qui minimise les erreurs avec M (au plus k)
- ▶ Complexité en $O(\lceil \frac{k \cdot \log_2(s)}{w} \rceil \cdot \lfloor \frac{w}{\log_2(s)} \rfloor \cdot n)$

LLCS dans des fenêtres minimales

- ▶ Entrées : un texte T de taille n , un motif M de taille m , a le nombre de matches minimum et s la taille maximale d'une fenêtre
- ▶ Sortie : pour chaque position de T , le suffixe de taille minimale (bornée par s) qui maximise les matches avec M (au moins a)
- ▶ Complexité en $O(\lceil \frac{m \cdot \log_2(s)}{w} \rceil \cdot \lfloor \frac{w}{\log_2(s)} \rfloor \cdot n \cdot s)$

Suffixe de taille minimale

- Pour la minimisation des erreurs

ATCATA
| | | |
ATTAT

ATCATA
| | |
ATTAT

- Pour la maximisation des matches

TGTACA
| | | |
GTGAA

TGTACA
| | | |
GTGAA

Utilisation de matrices

Les calculs d'erreurs, de matches et de longueurs entre le motif et le texte se feront en utilisant des matrices.

		text T					
		T_1	T_2	T_3	T_4	\dots	T_n
motif M	M_1						
	M_2						
	M_3						
	M_4			•			
	\vdots						
	M_m						

Recherche de motif par minimisation des erreurs

- └ Minimisation des erreurs
- └ Présentation du problème

Exemple



ATATGAT

ATTAT

- └ Minimisation des erreurs
- └ Présentation du problème

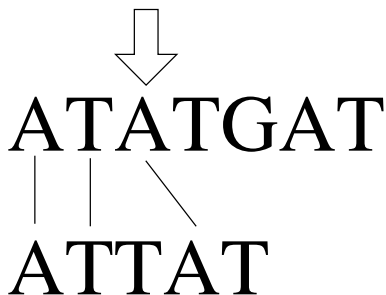
Exemple

ATATGAT

ATTAT

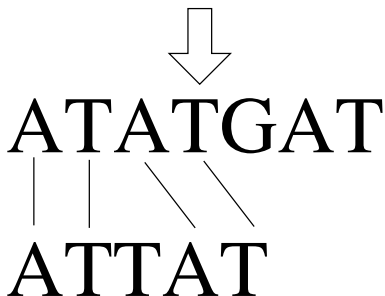
- └ Minimisation des erreurs
- └ Présentation du problème

Exemple



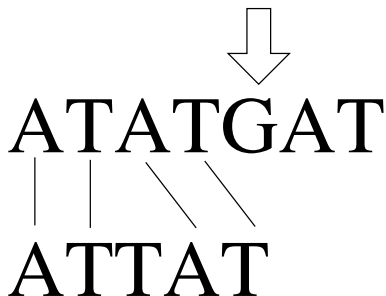
- └ Minimisation des erreurs
- └ Présentation du problème

Exemple



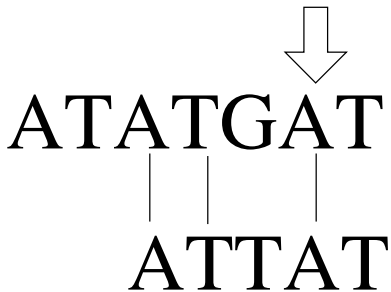
- └ Minimisation des erreurs
- └ Présentation du problème

Exemple



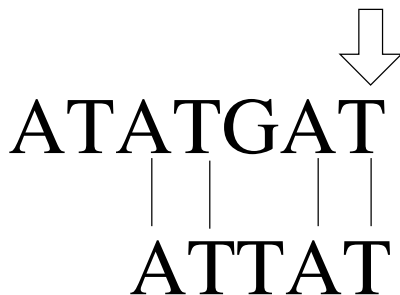
- └ Minimisation des erreurs
- └ Présentation du problème

Exemple



- └ Minimisation des erreurs
- └ Présentation du problème

Exemple



Matrice des erreurs

La matrice des erreurs E est calculée grâce à la récurrence :

$$E[0, j] = 0 \quad \text{pour tout } j \in [0..n]$$

$$E[i, 0] = i \quad \text{pour tout } i \in [0..m]$$

$$E[i, j] = \min \begin{cases} E[i-1, j-1] + \begin{cases} 0 & \text{si } M_i = T_j \\ 1 & \text{sinon} \end{cases} \\ E[i-1, j] + 1 \\ E[i, j-1] + 1 \end{cases}$$



Peter H. Sellers.

The theory and computation of evolutionary distances : Pattern recognition.

J. Algorithms, 1(4) :359–373, 1980.

Restriction de l'espace de calcul

Propriété

$$E[i, j] = E[i - 1, j - 1] \text{ ou } E[i, j] = E[i - 1, j - 1] + 1$$

Pour la colonne j , soit $p = \max\{i/E[i, j] \leq k\}$; alors, pour la colonne $j + 1$, seules les valeurs des lignes 0 à $p + 1$ seront $\leq k$.

⇒ Espace de calcul restreint en $O(k)$

		text					
			A	G	C	A	C
motif		X	X	X	X	X	X
	A	X	X	X	X	X	X
	C	X	X	X	X	X	X
	C			X	X	X	X
	G						
	T						



E. Ukkonen.

Finding approximate patterns in strings.

J. Algorithms, 6(1) :132–137, 1985.

Codification de E

motif \ text		A	G	C	A	C
	0	0	0	0	0	0
A	1	0	1	1	0	1
C	2	1	1	1	1	0
C	3	2	2	1	2	1

matrice E

différences verticales

motif \ text		A	G	C	A	C
	0	0	0	0	0	0
A	1	0	1	1	0	1
C	1	1	0	0	1	-1
C	1	1	1	0	1	1

$$V[i, j] = E[i, j] - E[i - 1, j] \in \{-1, 0, 1\}$$

différences horizontales

motif \ text		A	G	C	A	C
	0	0	0	0	0	0
A	0	-1	1	0	-1	1
C	0	-1	0	0	0	-1
C	0	-1	0	-1	1	-1

$$H[i, j] = E[i, j] - E[i, j - 1] \in \{-1, 0, 1\}$$



E. Ukkonen.

Finding approximate patterns in strings.

J. Algorithms, 6(1) :132–137, 1985.

M. S. Paterson and W. J. Masek.

A faster algorithm for computing string edit distances.

J. Comput. Syst. Sci., 20 :18–31, 1980.

Codification en vecteurs de bits

$$P_v[i] = 1 \text{ ssi } V[i, j] = 1; M_v[i] = 1 \text{ ssi } V[i, j] = -1$$

$$P_h[i] = 1 \text{ ssi } H[i, j] = 1; M_h[i] = 1 \text{ ssi } H[i, j] = -1$$

	A	C
	0	0
A	0	1
C	1	0
C	2	1
G	2	2
T	3	3

E

	C
	0
A	+1
C	-1
C	+1
G	+1
T	+1

V



E. W. Myers.

A fast bit-vector algorithm for approximate string matching based on dynamic programming.

Journal of the ACM, 46(3) :395–415, 1999.

Codification en vecteurs de bits

$$P_v[i] = 1 \text{ ssi } V[i, j] = 1; M_v[i] = 1 \text{ ssi } V[i, j] = -1$$

$$P_h[i] = 1 \text{ ssi } H[i, j] = 1; M_h[i] = 1 \text{ ssi } H[i, j] = -1$$

	A	C
	0	0
A	0	1
C	1	0
C	2	1
G	2	2
T	3	3

E

	C
	0
A	+1
C	-1
C	+1
G	+1
T	+1

V

	C
	0
A	1
C	0
C	1
G	1
T	1

 P_v 

E. W. Myers.

A fast bit-vector algorithm for approximate string matching based on dynamic programming.

Journal of the ACM, 46(3) :395–415, 1999.

Codification en vecteurs de bits

$$P_v[i] = 1 \text{ ssi } V[i, j] = 1; M_v[i] = 1 \text{ ssi } V[i, j] = -1$$

$$P_h[i] = 1 \text{ ssi } H[i, j] = 1; M_h[i] = 1 \text{ ssi } H[i, j] = -1$$

	A	C
	0	0
A	0	1
C	1	0
C	2	1
G	2	2
T	3	3

E

	C
	0
A	+1
C	-1
C	+1
G	+1
T	+1

V

	C
	0
A	1
C	0
C	1
G	1
T	1

 P_v

	C
	0
A	0
C	1
C	0
G	0
T	0

 M_v 

E. W. Myers.

A fast bit-vector algorithm for approximate string matching based on dynamic programming.

Journal of the ACM, 46(3) :395–415, 1999.

Codification en vecteurs de bits

$$P_v[i] = 1 \text{ ssi } V[i, j] = 1; M_v[i] = 1 \text{ ssi } V[i, j] = -1$$

$$P_h[i] = 1 \text{ ssi } H[i, j] = 1; M_h[i] = 1 \text{ ssi } H[i, j] = -1$$

	A	C
	0	0
A	0	1
C	1	0
C	2	1
G	2	2
T	3	3

E

	C
	0
A	+1
C	-1
C	+1
G	+1
T	+1

V

	C
	0
A	1
C	0
C	1
G	1
T	1

 P_v

	C
	0
A	0
C	1
C	0
G	0
T	0

 M_v

	C
	0
A	+1
C	-1
C	-1
G	0
T	0

H

	C
	0
A	1
C	0
C	0
G	0
T	0

 P_h

	C
	0
A	0
C	1
C	1
G	0
T	0

 M_h 

E. W. Myers.

A fast bit-vector algorithm for approximate string matching based on dynamic programming.

Journal of the ACM, 46(3) :395–415, 1999.

Algorithme de Myers

1. $P_{v_0}(i) = 1, \forall i, M_{v_0}(i) = 0, \forall i$, et $score_0 = m$
2. Pour j allant de 1 à n :

$$P_{h_j} = M_{v_{j-1}} \vee \neg(X_{h_j} \vee P_{v_{j-1}})$$

$$M_{h_j} = P_{v_{j-1}} \wedge X_{h_j}$$

$$Score_j = Score_{j-1} + (1 \text{ si } P_{h_j}(m)) - (1 \text{ si } M_{h_j}(m))$$

$$P_{v_j} = (M_{h_j} \lll 1) \vee \neg(X_{v_j} \vee (P_{h_j} \lll 1))$$

$$M_{v_j} = (P_{h_j} \lll 1) \wedge X_{v_j}$$

avec $X_{v_j} = (M_i = T_j) \vee M_{v_{j-1}}$

Matrice des longueurs L

$$L[1, j] = 1 \quad \text{pour tout } j \in [1..n]$$

$$L[i, 1] = 1 \quad \text{pour tout } i \in [1..m]$$

$$L[i, j] = \begin{cases} L[i-1, j] & \text{si (a) ou ((b) et (c) et (d))} & (1) \\ L[i, j-1] + 1 & \text{si non (a) et non (b) et non (e)} & (2) \\ L[i-1, j-1] + 1 & \text{sinon} & (3) \end{cases}$$

où

$$(a) E[i, j] = E[i-1, j] + 1 \equiv P_v$$

$$(b) T_j = M_i$$

(c) M_i est la première occurrence de la lettre dans M

$$(d) E[i-1, j] = E[i-1, j-1] \equiv \sim P_h | M_h$$

$$(e) E[i-1, j-1] \leq E[i, j-1] \equiv P_v | \sim M_v$$

Calcul de la matrice des longueurs

1. Initialisation des masques $M_a \dots M_e$ correspondants aux conditions (a) ... (e)

Calcul de la matrice des longueurs

1. Initialisation des masques $M_a \dots M_e$ correspondants aux conditions (a) ... (e)
2. Initialisation des masques M_1, M_2 et M_3 correspondants aux affectations (1) (2) et (3)

Calcul de la matrice des longueurs

1. Initialisation des masques $M_a \dots M_e$ correspondants aux conditions (a) ... (e)
2. Initialisation des masques M_1, M_2 et M_3 correspondants aux affectations (1) (2) et (3)
3. Calcul des valeurs correspondantes aux affectations (2) et (3)

Calcul de la matrice des longueurs

1. Initialisation des masques $M_a \dots M_e$ correspondants aux conditions (a) ... (e)
2. Initialisation des masques M_1, M_2 et M_3 correspondants aux affectations (1) (2) et (3)
3. Calcul des valeurs correspondantes aux affectations (2) et (3)
4. Calcul des valeurs correspondantes à l'affectation (1)
⇒ Shifts

Exemple de construction de L

$L =$

	A	G	C
A	1	1	1
C	1	2	
C	1	2	
G	1	2	
T	1	2	

Exemple de construction de L

- Initialisation des masques M_1 , M_2 et M_3

	A	G	C
A	1	1	1
C	1	2	
C	1	2	
G	1	2	
T	1	2	

0
0
1
1

0
0
0
0

1
1
0
0

Exemple de construction de L

- Calcul des valeurs correspondantes aux affectations (2) et (3)

$$L = \begin{array}{c|ccc} & A & G & C \\ \hline A & 1 & 1 & 1 \\ C & 1 & 2 & 2 \\ C & 1 & 2 & 3 \\ G & 1 & 2 & \\ T & 1 & 2 & \end{array} \quad M_1 = \begin{array}{c} \\ 0 \\ 0 \\ 1 \\ 1 \end{array} \quad M_2 = \begin{array}{c} \\ 0 \\ 0 \\ 0 \\ 0 \end{array} \quad M_3 = \begin{array}{c} \\ 1 \\ 1 \\ 0 \\ 0 \end{array}$$

Exemple de construction de L

- Calcul des valeurs correspondantes à l'affectation (1)

$$L = \begin{array}{c|ccc} & A & G & C \\ \hline A & 1 & 1 & 1 \\ C & 1 & 2 & 2 \\ C & 1 & 2 & 3 \\ G & 1 & 2 & 3 \\ T & 1 & 2 & \end{array} \quad M_1 = \begin{array}{c} \\ 0 \\ 0 \\ 1 \\ 1 \end{array} \quad M_2 = \begin{array}{c} \\ 0 \\ 0 \\ 0 \\ 0 \end{array} \quad M_3 = \begin{array}{c} \\ 1 \\ 1 \\ 0 \\ 0 \end{array}$$

Exemple de construction de L

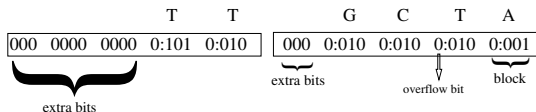
- Calcul des valeurs correspondantes à l'affectation (1)

$$L = \begin{array}{c|ccc} & A & G & C \\ \hline A & 1 & 1 & 1 \\ C & 1 & 2 & 2 \\ C & 1 & 2 & 3 \\ G & 1 & 2 & 3 \\ T & 1 & 2 & 3 \end{array} \quad M_1 = \begin{array}{c} \square \\ 0 \\ 0 \\ 1 \\ 1 \end{array} \quad M_2 = \begin{array}{c} \square \\ 0 \\ 0 \\ 0 \\ 0 \end{array} \quad M_3 = \begin{array}{c} \square \\ 1 \\ 1 \\ 0 \\ 0 \end{array}$$

Codification de L

- ▶ Les différences verticales et horizontales ne sont pas bornées
 \implies on code directement les valeurs en binaire
- ▶ Les longueurs acceptantes sont $\leq s$
 \implies on code les valeurs $\leq s$ et $s + 1$ sinon
- ▶ Une entrée de L est codée sur $\log_2(s + 1) + 1 = bsize$ bits
 \implies une colonne est codée sur $\lceil \frac{m \cdot bsize}{w} \rceil$ mots mémoire

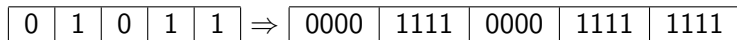
Exemple :



Codification appliquée à E

- Codification des entrées de E avec le même nombre de bits que pour celles de L

Exemple :



Complexité

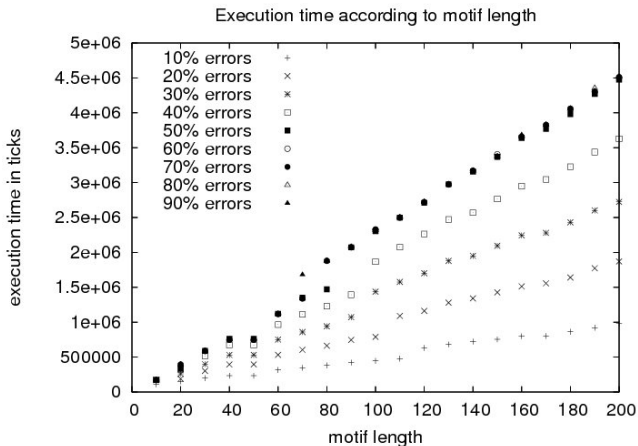
- ▶ Le calcul de E avec la nouvelle codification se fait en $O(\lceil \frac{\log_2(s) \cdot k}{w} \rceil \cdot n)$ au lieu de $O(\lceil \frac{k}{w} \rceil \cdot n)$ avec la codification de Myers
- ▶ Le calcul de L s'effectue en $O(\lceil \frac{\log_2(s) \cdot k}{w} \rceil \cdot \lfloor \frac{w}{\log_2(s)} \rfloor \cdot n)$
- ▶ L'algorithme total est de complexité temporelle $O(\lceil \frac{\log_2(s) \cdot k}{w} \rceil \cdot \lfloor \frac{w}{\log_2(s)} \rfloor \cdot n)$

Temps d'exécution empiriques

X	X	X	X	X	X
X	X	X	X	X	X
X	X	X			X

X	X	X	X	X	X
X	X	X	X	X	X
X	X	X	X	X	X
		X	X	X	X

X	X	X	X	X	X
X	X	X	X	X	X
X	X	X	X	X	X
		X	X	X	X
			X	X	X
				X	X



Recherche de motif par maximisation des matches

Maximisation des matches

- ▶ Entrées : un texte T de taille n , un motif M de taille m , a le nombre de matches minimum et s la taille maximale d'une fenêtre

- ▶ Sortie : pour chaque position de T , le suffixe de taille minimale (bornée par s) qui maximise les matches avec M (au moins a)

Calcul de la matrice des matches /

La matrice I correspond à la matrice des longueurs des plus longues sous-séquences communes.

$$I[i, 0] = 0 \quad \text{pour tout } i \in [0..m]$$

$$I[0, j] = 0 \quad \text{pour tout } j \in [0..n]$$

$$I[i, j] = \begin{cases} I[i - 1, j - 1] + 1 & \text{if } M_i = T_j \\ \max(I[i - 1, j], I[i, j - 1]) & \text{sinon} \end{cases}$$



R. A. Wagner and M. J. Fischer.

The string-to-string correction problem.

J. ACM, 21(1) :168–173, 1974.



D. Sankoff and J. B. Kruskal, editors.

Time Warps, String Edits and Macromolecules : the Theory and Practice of Sequence Comparison.

Addison-Wesley, 1983.

Calcul de la matrice des matches /

0
1
2
2
2
3
3
3

column $j-1$

0
0
0
1
0
0
1
0

Peq

Calcul de la matrice des matches /

- Calcul des positions pour lesquelles $M_i = T_j$ grâce à un masque

0
1
2
2
2
3
3
3

column $j-1$

0
0
0
1
0
0
1
0

Peq

0
0
0
3
0
0
4
0

 $I[i,j] = I[i-1,j-1] + 1$

Calcul de la matrice des matches /

► Décalage de ces valeurs (shifts)

0
1
2
2
2
3
3
3

column $j-1$

0
0
0
1
0
0
1
0

Peq

0
0
0
3
0
0
4
0

 $I[i,j] = I[i-1,j-1] + 1$

0
0
0
3
3
3
4
4

 $I[i,j] = I[i-1,j]$

Calcul de la matrice des matches /

- Calcul des valeurs maximales entre $I[i-1, j]$ et $I[i, j-1]$

0
1
2
2
2
3
3
3

column $j-1$

0
0
0
1
0
0
1
0

Peq

0
0
0
3
0
0
4
0

 $I[i, j] = I[i-1, j-1] + 1$

0
0
0
3
3
3
4
4

 $I[i, j] = I[i-1, j]$

0
1
2
3
3
3
4
4

 $I[i, j] = I[i, j-1]$

Calcul de la matrice des longueurs L

$$L[i, 0] = 0 \quad \text{for all } i \in [0..m]$$

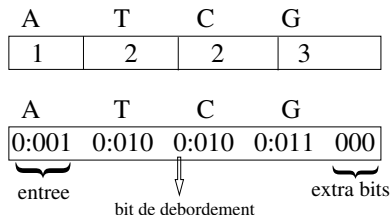
$$L[0, j] = 0 \quad \text{for all } j \in [0..n]$$

$$L[i, j] = \begin{cases} L[i-1, j] & \text{if } I[i, j] = I[i-1, j] \\ L[i-1, j-1] + 1 & \text{if } I[i, j] = I[i-1, j-1] + 1 \\ L[i, j-1] + 1 & \text{if } I[i, j] = I[i, j-1] \end{cases}$$

Codification

- ▶ $I[i, j] \leq s$ et $L[i, j] \leq s$ donc on code chaque entrée par leurs valeurs en binaire sur $\log_2(s) + 1$ bits
- ▶ Une colonne est donc codée sur $\lceil \frac{(\log_2(s)+1) \cdot m}{w} \rceil$ mots mémoire

Exemple :

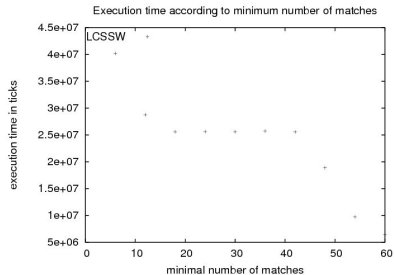
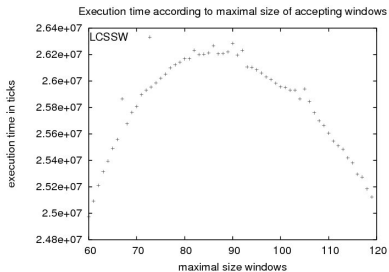
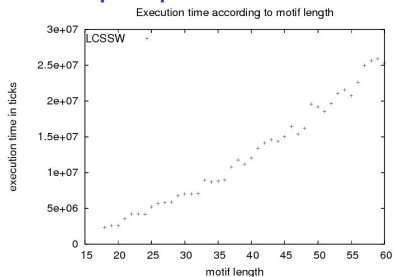


Complexité

- ▶ Pour la recherche sur un texte T , on construit I et L jusqu'à $L[m, j] = s$ pour $I[m, j] = v$
On initialise les colonnes à 0 et on lit le texte à partir de sa p^e lettre ($p = j - s + 1 + a - v$)
- ▶ L'algorithme est dans le pire cas en $O(\lceil \frac{\log_2(s) \cdot m}{w} \rceil \cdot \lfloor \frac{w}{\log_2(s)} \rfloor \cdot s \cdot n)$

- └ Maximisation des matches
- └ Complexité et temps d'exécution empiriques

Temps d'exécution empiriques



Merci de votre attention