Algorithmes rapides de recherche exacte de mot

Thierry Lecroq
Thierry.Lecroq@univ-rouen.fr

Laboratoire d'Informatique, du Traitement de l'Information et des Systèmes.

Une partie de ce travail a été effectuée en collaboration avec Maxime Crochemore (IGM, université de Marne-la-Vallée)

Séminaire Combinatoire & Algorithmique
Université de Rouen

12 avril 2007

Plan

Plan

Recherche exacte de mot

Problème

Trouver une ou plus généralement toutes les occurrences d'un mot x de longueur m dans un texte y de longueur n.

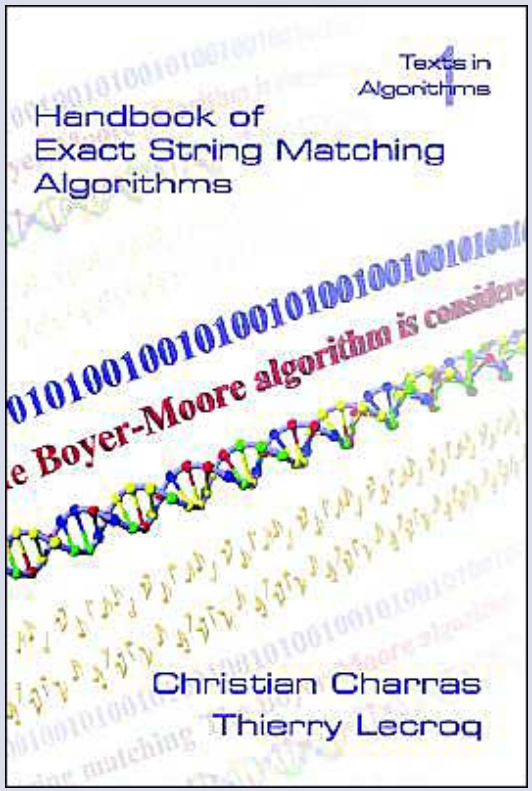
Les deux mots x and y sont construits sur un alphabet A de cardinalité s.

Recherche exacte de mot

Solutions

Beaucoup!!!

Voir `http://monge.univ-mlv.fr/~lecroq/string`



Recherche exacte de mot

1. Brute force algorithm
2. Search with an automaton
3. Karp-Rabin algorithm
4. Shift Or algorithm
5. Morris-Pratt algorithm
6. Knuth-Morris-Pratt algorithm
7. Simon algorithm
8. Colussi algorithm
9. Galil-Giancarlo algorithm
10. Apostolico-Crochemore algorithm
11. Not So Naive algorithm
12. Forward Dawg Matching algorithm
13. Boyer-Moore algorithm
14. Galil algorithm
15. Smyth algorithm
16. Turbo-BM algorithm
17. Apostolico-Giancarlo algorithm
18. Reverse Colussi algorithm
19. Horspool algorithm
20. Fast Search algorithm
21. Quick Search algorithm
22. Turbo Search algorithm
23. Tuned Boyer-Moore algorithm
24. Zhu-Takaoka algorithm
25. Berry-Ravindran algorithm
26. Smith algorithm
27. Raita algorithm
28. Reverse Factor algorithm
29. Turbo Reverse Factor algorithm
30. Backward Oracle Matching algorithm
31. Backward Nondeterministic Dawg Matching algorithm
32. Galil-Seiferas algorithm
33. Two Way algorithm
34. String Matching on Ordered Alphabets
35. Optimal Mismatch algorithm
36. Maximal Shift algorithm
37. Skip Search algorithm
38. KmpSkip Search algorithm
39. Alpha Skip Search algorithm

Recherche exacte de mot

Les plus connues

Knuth–Morris–Pratt et Boyer–Moore, 1977

Recherche exacte de mot

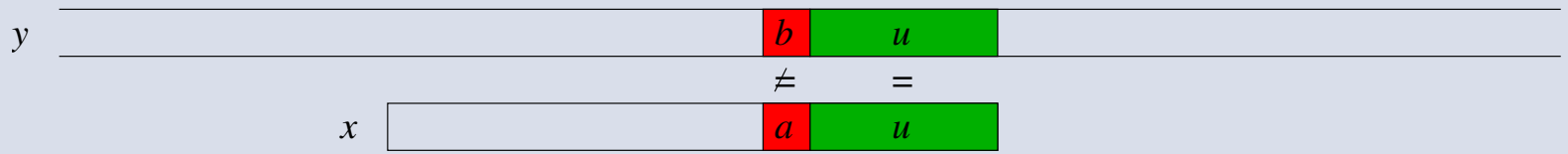
Mécanisme de « fenêtre glissante »

KMP : de gauche à droite ($\longrightarrow$)

BM : de droite à gauche ($\longleftarrow$)

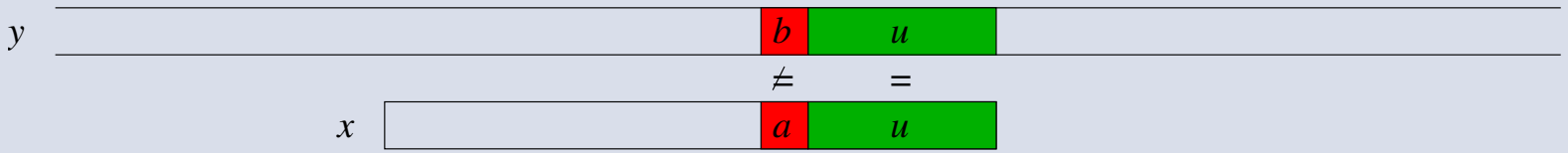
Boyer-Moore

Situation générale



Un suffixe u du mot x est détecté et une inégalité intervient entre une lettre a de x et une lettre b du texte y.

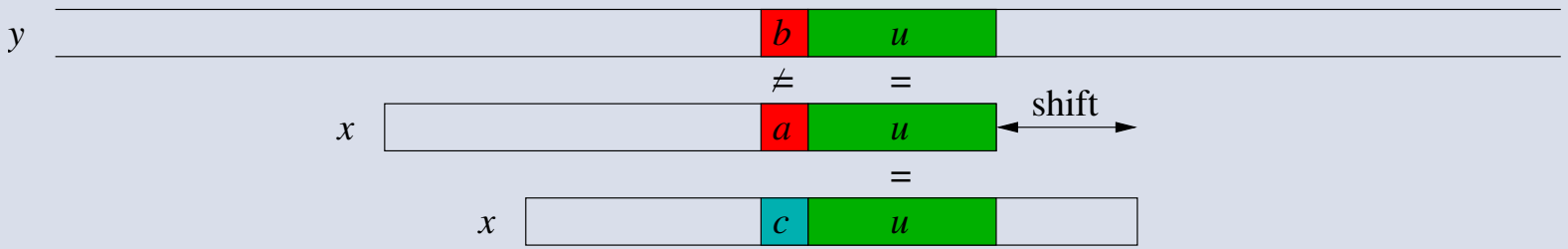
Décalage de « bon suffixe »



Le décalage de bon suffixe consiste à aligner le facteur $u = x[i+1..m-1] = y[i+j+1..j+m-1]$ avec une des ses réoccurrences dans x.

Trois décalages de bon suffixe (I)

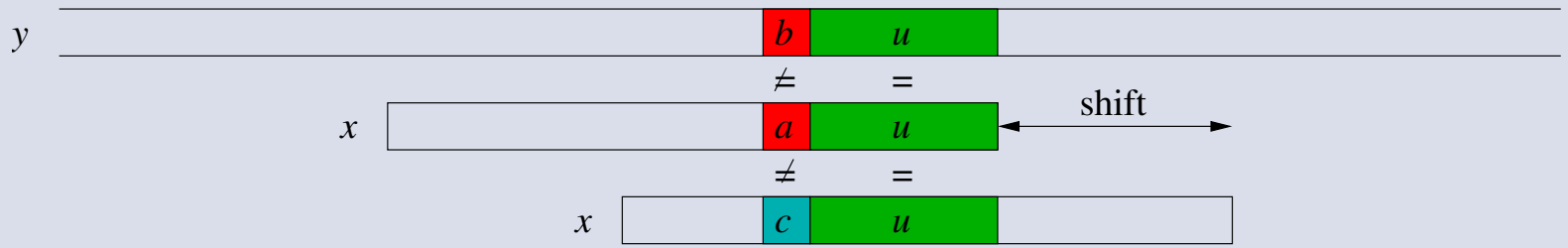
Décalage faible



Aucune condition sur la lettre c précédant u : il est alors possible que $c = a$.

Trois décalages de bon suffixe (II)

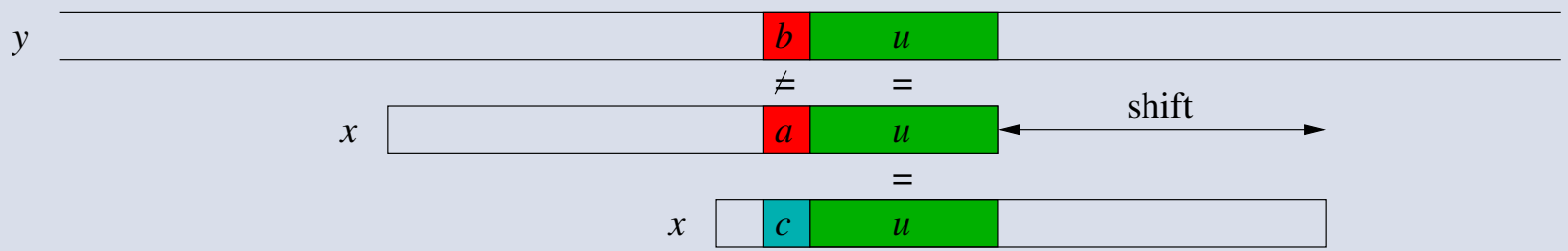
Décalage fort



La lettre c doit être différente de la lettre a.

Trois décalages de bon suffixe (III)

Meilleur décalage



c doit être égale à b.

Trois décalages de bon suffixe

- Les décalages faible et fort ne dépendent que de x.
- Le meilleur décalage dépend de x et de l'alphabet.

Complexité

Theorem (Cole, 1994)

Lors de la recherche de la première occurrence d'un mot x non-périodique dans un texte de longueur n, l'algorithme de Boyer-Moore effectue moins de $3n$ comparaisons entre des lettres de x et des lettres de y.

Principales améliorations

- Galil, 1980 : mémorisation de préfixe, $14n$
- Smyth, 2003 : mémorisation de préfixe, $4n$
- Turbo BM, 1994 : mémorisation de suffixe, $2n$
- Apostolico – Giancarlo, 1986 : mémorisation de suffixes, $\frac{3}{2}n$

Amélioration récente

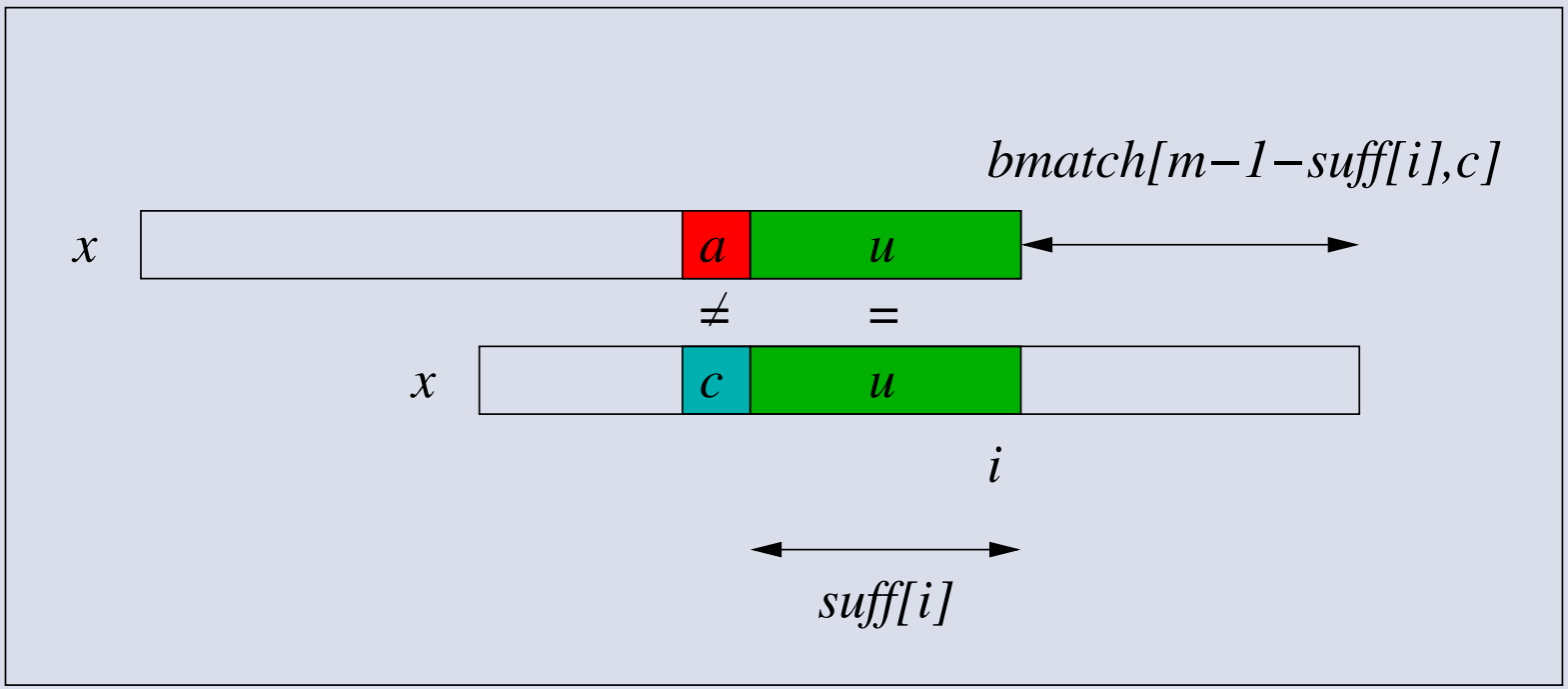
S. Lu, F. Cao et Y. Lu
Pama : a Fast String Matching Algorithm
International Journal on Foundations of Computer Science
17(2) (2006) 357–358

Meilleur décalage en espace quadratique et stratégie à la Turbo BM

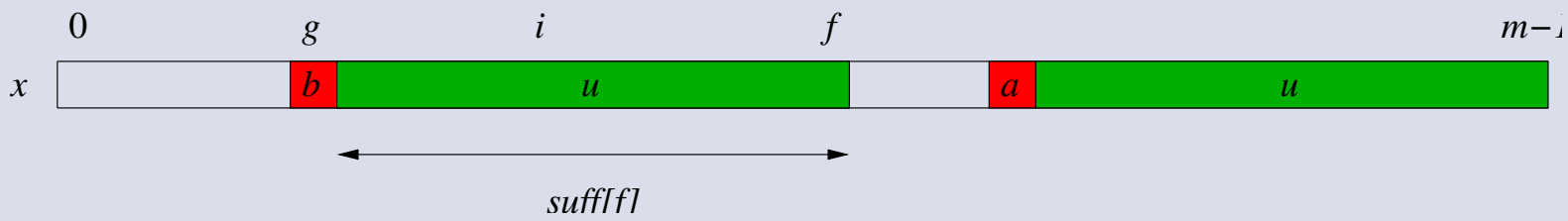
Plan

Calcul du meilleur décalage

Pour $0 \leq i \leq m-1$:
$suff[i]$ = longueur du plus long suffixe de x se terminant à la position i sur x.



Calcul linéaire de $suff$

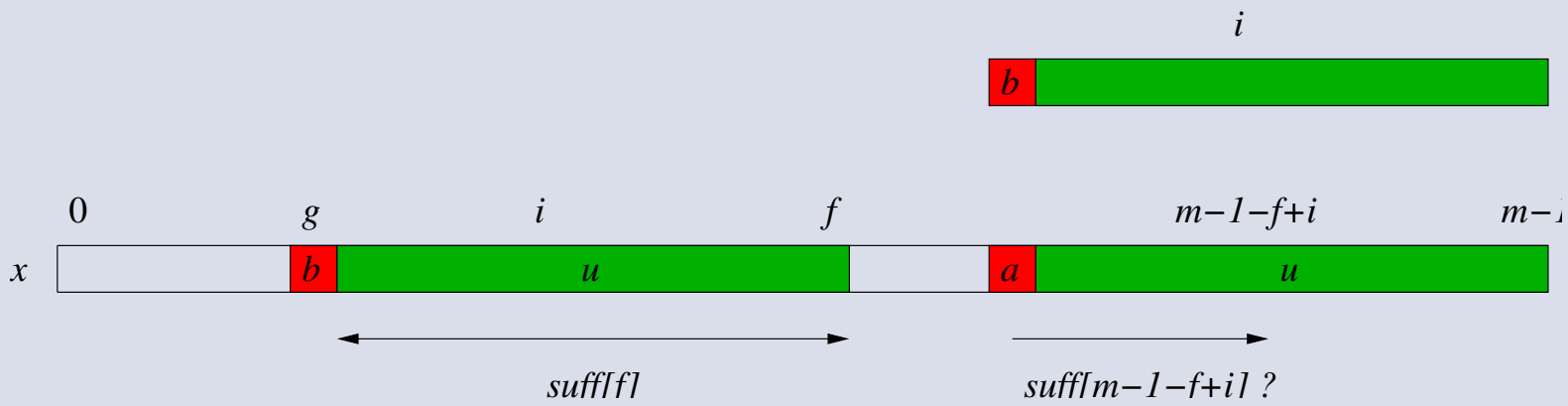


Invariant

$g = f - suff[f]$ est minimal

Calcul linéaire de $suff$

Trois cas



Calcul du meilleur décalage

Examiner les valeurs de la table $suff$ de gauche à droite.

Une autre preuve de linéarité.

Calcul du meilleur décalage

Exemple

	a	c	g	t
0				
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				

i	0	1	2	3	4	5	6	7	8	9	10	11
$x[i]$	c	a	t	a	c	a	t	a	a	a	t	a
$suff[i]$	0	1	0	3	0	1	0	3	1	1	0	12

Calcul du meilleur décalage

Exemple

	a	c	g	t
0				
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11		11		

i	0	1	2	3	4	5	6	7	8	9	10	11
$x[i]$	c	a	t	a	c	a	t	a	a	a	t	a
$suff[i]$	0	1	0	3	0	1	0	3	1	1	0	12

Calcul du meilleur décalage

Exemple

	a	c	g	t
0				
1				
2				
3				
4				
5				
6				
7				
8				
9				
10		10		
11		11		

i	0	1	2	3	4	5	6	7	8	9	10	11
$x[i]$	c	a	t	a	c	a	t	a	a	a	t	a
$suff[i]$	0	1	0	3	0	1	0	3	1	1	0	12

Calcul du meilleur décalage

Exemple

	a	c	g	t
0				
1				
2				
3				
4				
5				
6				
7				
8				
9				
10		10		
11		11		9

i	0	1	2	3	4	5	6	7	8	9	10	11
$x[i]$	c	a	t	a	c	a	t	a	a	a	t	a
$suff[i]$	0	1	0	3	0	1	0	3	1	1	0	12

Calcul du meilleur décalage

Exemple

	a	c	g	t
0				
1				
2				
3				
4				
5				
6				
7				
8		8		
9				
10		10		
11		11		9

i	0	1	2	3	4	5	6	7	8	9	10	11
$x[i]$	c	a	t	a	c	a	t	a	a	a	t	a
$suff[i]$	0	1	0	3	0	1	0	3	1	1	0	12

Calcul du meilleur décalage

Exemple

	a	c	g	t
0				
1				
2				
3				
4				
5				
6				
7				
8		8		
9				
10		10		
11		7		9

i	0	1	2	3	4	5	6	7	8	9	10	11
$x[i]$	c	a	t	a	c	a	t	a	a	a	t	a
$suff[i]$	0	1	0	3	0	1	0	3	1	1	0	12

Calcul du meilleur décalage

Exemple

	a	c	g	t
0				
1				
2				
3				
4				
5				
6				
7				
8		8		
9				
10		6		
11		7		9

i	0	1	2	3	4	5	6	7	8	9	10	11
$x[i]$	c	a	t	a	c	a	t	a	a	a	t	a
$suff[i]$	0	1	0	3	0	1	0	3	1	1	0	12

Calcul du meilleur décalage

Exemple

	a	c	g	t
0				
1				
2				
3				
4				
5				
6				
7				
8		8		
9				
10		6		
11		7		5

i	0	1	2	3	4	5	6	7	8	9	10	11
$x[i]$	c	a	t	a	c	a	t	a	a	a	t	a
$suff[i]$	0	1	0	3	0	1	0	3	1	1	0	12

Calcul du meilleur décalage

Exemple

	a	c	g	t
0				
1				
2				
3				
4				
5				
6				
7				
8		4		
9				
10		6		
11		7		5

i	0	1	2	3	4	5	6	7	8	9	10	11
$x[i]$	c	a	t	a	c	a	t	a	a	a	t	a
$suff[i]$	0	1	0	3	0	1	0	3	1	1	0	12

Calcul du meilleur décalage

Exemple

	a	c	g	t
0				
1				
2				
3				
4				
5				
6				
7				
8		4		
9				
10	3	6		
11		7		5

i	0	1	2	3	4	5	6	7	8	9	10	11
$x[i]$	c	a	t	a	c	a	t	a	a	a	t	a
$suff[i]$	0	1	0	3	0	1	0	3	1	1	0	12

Calcul du meilleur décalage

Exemple

	a	c	g	t
0				
1				
2				
3				
4				
5				
6				
7				
8		4		
9				
10	2	6		
11		7		5

i	0	1	2	3	4	5	6	7	8	9	10	11
$x[i]$	c	a	t	a	c	a	t	a	a	a	t	a
$suff[i]$	0	1	0	3	0	1	0	3	1	1	0	12

Calcul du meilleur décalage

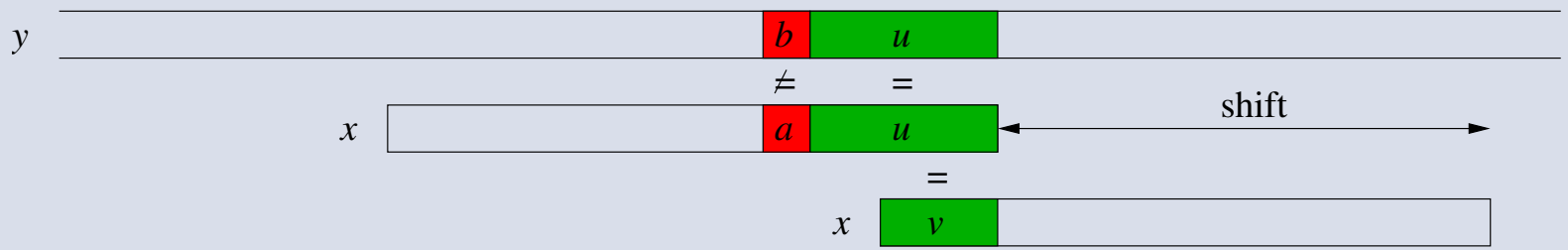
Exemple

	a	c	g	t
0				
1				
2				
3				
4				
5				
6				
7				
8		4		
9				
10	2	6		
11		7		1

i	0	1	2	3	4	5	6	7	8	9	10	11
$x[i]$	c	a	t	a	c	a	t	a	a	a	t	a
$suff[i]$	0	1	0	3	0	1	0	3	1	1	0	12

Meilleur décalage

Cas dégénéré



u ne réapparait pas dans x

v est le plus long préfixe de x suffixe de u (qui est un suffixe de x)

v est un bord qui peut être détecté lorsque $suff[i] = i + 1$

Calcul du meilleur décalage

Exemple

	a	c	g	t
0	12	12	12	12
1	12	12	12	12
2	12	12	12	12
3	12	12	12	12
4	12	12	12	12
5	12	12	12	12
6	12	12	12	12
7	12	12	12	12
8	12	12	12	12
9	12	12	12	12
10	12	12	12	12
11	12	12	12	12

i	0	1	2	3	4	5	6	7	8	9	10	11
$x[i]$	c	a	t	a	c	a	t	a	a	a	t	a
$suff[i]$	0	1	0	3	0	1	0	3	1	1	0	12

Calcul du meilleur décalage

Exemple

	a	c	g	t
0	12	12	12	12
1	12	12	12	12
2	12	12	12	12
3	12	12	12	12
4	12	12	12	12
5	12	12	12	12
6	12	12	12	12
7	12	12	12	12
8	4	12	12	12
9	12	12	12	12
10	2	6	12	12
11	12	7	12	1

i	0	1	2	3	4	5	6	7	8	9	10	11
$x[i]$	c	a	t	a	c	a	t	a	a	a	t	a
$suff[i]$	0	1	0	3	0	1	0	3	1	1	0	12

Plan

Hachage de q-grams

Definition

q-gram : facteur de longueur q

Méthode

- Calcul d'une valeur de hachage dans $[0, 255]$ pour chaque q-grams du mot x
- Calcul d'une longueur de décalage pour chaque valeur de hachage
- Déboucler tant que possible

Hachage de q-grams

Exemple

i	0	1	2	3	4	5	6	7	8	9	10	11
$x[i]$	c	a	t	a	c	a	t	a	a	a	t	a

$shift[i] \leftarrow 10$

$\forall i \in [0; 255]$

Hachage de q-grams

Exemple

i	0	1	2	3	4	5	6	7	8	9	10	11
$x[i]$	c	a	t	a	c	a	t	a	a	a	t	a

$$h(\texttt{cat}) = ((rank(\texttt{c}) \times 2 + rank(\texttt{a})) \times 2 + rank(\texttt{t}) = 194$$

$$shift[194] = 10$$

$$shift[194] \leftarrow 9$$

Hachage de q-grams

Exemple

i	0	1	2	3	4	5	6	7	8	9	10	11
$x[i]$	c	a	t	a	c	a	t	a	a	a	t	a

$$h(\texttt{ata}) = ((rank(\texttt{a}) \times 2 + rank(\texttt{a})) \times 2 + rank(\texttt{a}) = 205$$

$$shift[205] = 10$$

$$shift[205] \leftarrow 8$$

Hachage de q-grams

Exemple

i	0	1	2	3	4	5	6	7	8	9	10	11
$x[i]$	c	a	t	a	c	a	t	a	a	a	t	a

$$h(\texttt{tac}) = ((rank(\texttt{t}) \times 2 + rank(\texttt{a})) \times 2 + rank(\texttt{c}) = 245$$

$$shift[245] = 10$$

$$shift[245] \leftarrow 7$$

Hachage de q-grams

Exemple

i	0	1	2	3	4	5	6	7	8	9	10	11
$x[i]$	c	a	t	a	c	a	t	a	a	a	t	a

$$h(\texttt{aca}) = ((rank(\texttt{a}) \times 2 + rank(\texttt{c})) \times 2 + rank(\texttt{a}) = 171$$

$$shift[171] = 10$$

$$shift[171] \leftarrow 6$$

Hachage de q-grams

Exemple

i	0	1	2	3	4	5	6	7	8	9	10	11
$x[i]$	c	a	t	a	c	a	t	a	a	a	t	a

$$h(\texttt{cat}) = ((rank(\texttt{c}) \times 2 + rank(\texttt{a})) \times 2 + rank(\texttt{t}) = 194$$

$$shift[194] = 9$$

$$shift[194] \leftarrow 5$$

Hachage de q-grams

Exemple

i	0	1	2	3	4	5	6	7	8	9	10	11
$x[i]$	c	a	t	a	c	a	t	a	a	a	t	a

$$h(\texttt{ata}) = ((rank(\texttt{a}) \times 2 + rank(\texttt{t})) \times 2 + rank(\texttt{a}) = 205$$

$$shift[205] = 8$$

$$shift[205] \leftarrow 4$$

Hachage de q-grams

Exemple

i	0	1	2	3	4	5	6	7	8	9	10	11
$x[i]$	c	a	t	a	c	a	t	a	a	a	t	a

$$h(\texttt{taa}) = ((rank(\texttt{t}) \times 2 + rank(\texttt{a})) \times 2 + rank(\texttt{a}) = 243$$

$$shift[243] = 10$$

$$shift[243] \leftarrow 3$$

Hachage de q-grams

Exemple

i	0	1	2	3	4	5	6	7	8	9	10	11
$x[i]$	c	a	t	a	c	a	t	a	a	a	t	a

$$h(\texttt{aaa}) = ((rank(\texttt{a}) \times 2 + rank(\texttt{a})) \times 2 + rank(\texttt{a}) = 167$$

$$shift[167] = 10$$

$$shift[167] \leftarrow 2$$

Hachage de q-grams

Exemple

i	0	1	2	3	4	5	6	7	8	9	10	11
$x[i]$	c	a	t	a	c	a	t	a	a	a	t	a

$$h(\texttt{aat}) = ((rank(\texttt{a}) \times 2 + rank(\texttt{a})) \times 2 + rank(\texttt{t}) = 186$$

$$shift[186] = 10$$

$$shift[186] \leftarrow 1$$

Hachage de q-grams

Exemple

i	0	1	2	3	4	5	6	7	8	9	10	11
$x[i]$	c	a	t	a	c	a	t	a	a	a	t	a

$$h(\texttt{ata}) = ((rank(\texttt{a}) \times 2 + rank(\texttt{t})) \times 2 + rank(\texttt{a}) = 205$$

$$shift[205] = 4 \Longrightarrow sh1 \leftarrow 4$$

$$shift[205] \leftarrow 0$$

Hashing q-grams

Algorithme New$q(x, m, y, n)$ pour $q = 3$

▷ Prétraitement
pour $i \leftarrow 0$ **à** 255 **faire** $shift[i] \leftarrow m - 2$
pour $i \leftarrow 2$ **à** $m - 2$ **faire**
$h \leftarrow ((x[i-2] \times 2 + x[i-1]) \times 2) + x[i]$
$shift[h \mod 256] \leftarrow m - 1 - i$
$h \leftarrow ((x[m-3] \times 2 + x[m-2]) \times 2) + x[m-1]$
$sh1 \leftarrow shift[h \mod 256]$
$shift[h \mod 256] \leftarrow 0$

Hachage de q-grams

Algorithme New$q(x, m, y, n)$ pour $q = 3$

▷ Recherche
$y[n..n+m-1] \leftarrow x$ ▷ Sentinelle
$j \leftarrow m-1$
tantque VRAI **faire**
  $sh \leftarrow 1$
  tantque $sh \neq 0$ **faire**
    $h \leftarrow ((y[j-2] \times 2 + y[j-1]) \times 2) + y[j]$
    $sh \leftarrow shift[h \mod 256]$
    $j \leftarrow j + sh$
  si $j < n$ **alors**
    si $x = y[j-m+1..j]$ **alors** REPORTER$(j-m+1)$
    $j \leftarrow j + sh1$
  sinon RETOURNER

Plan

Résultats expérimentaux

Conditions

- Processeur Intel Pentium à 1300MHz
- Linux Red Hat version 2.4.20-8
- gcc avec l'option d'optimisation `-O3`
- fonction `clock`
- 100 mots choisis pseudo-aléatoirement dans les textes

Textes

- Alphabet binaire, pseudo-aléatoire (distribution uniforme), 4 Mo
- `E.coli` du Large Canterbury Corpus, 4,6 Mo
- Alphabet de cardinalité 8, pseudo-aléatoire (distribution uniforme), 4 Mo
- `world192.txt` du Large Canterbury Corpus, 4,3 Mo

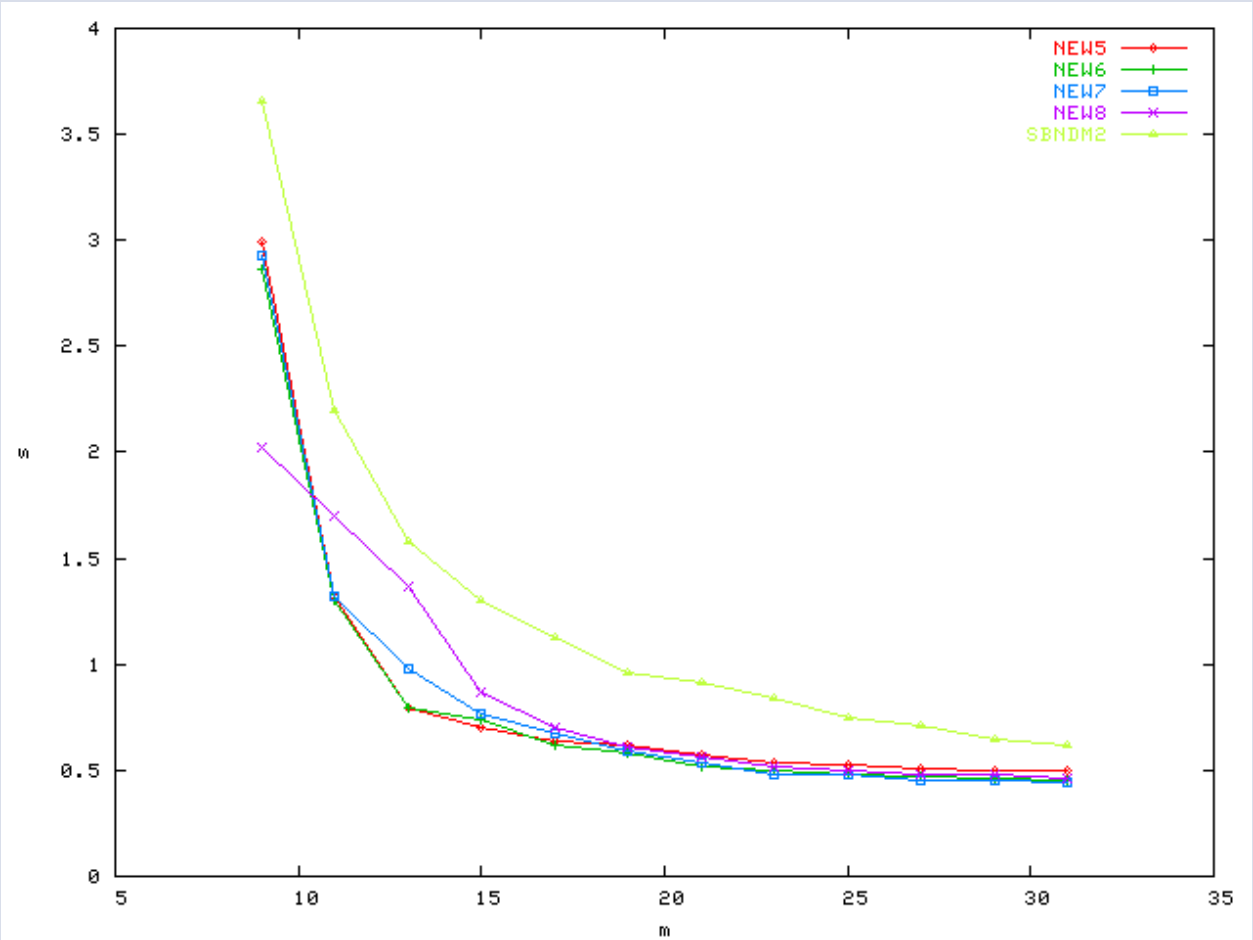
Résultats expérimentaux

Algorithmes

- `BM2fast` Boyer–Moore meilleur décalage et boucle rapide
- `NEW`q pour $q \in [3, 8]$
- `TBM` Tuned Boyer–Moore (Hume & Sunday, 1991)
- `SSABS` (Sheik, Aggarwal, Poddar, Balakrishnan & Sekar, 2004)
- `SBNDM2` (Holub & Durian, 2005)

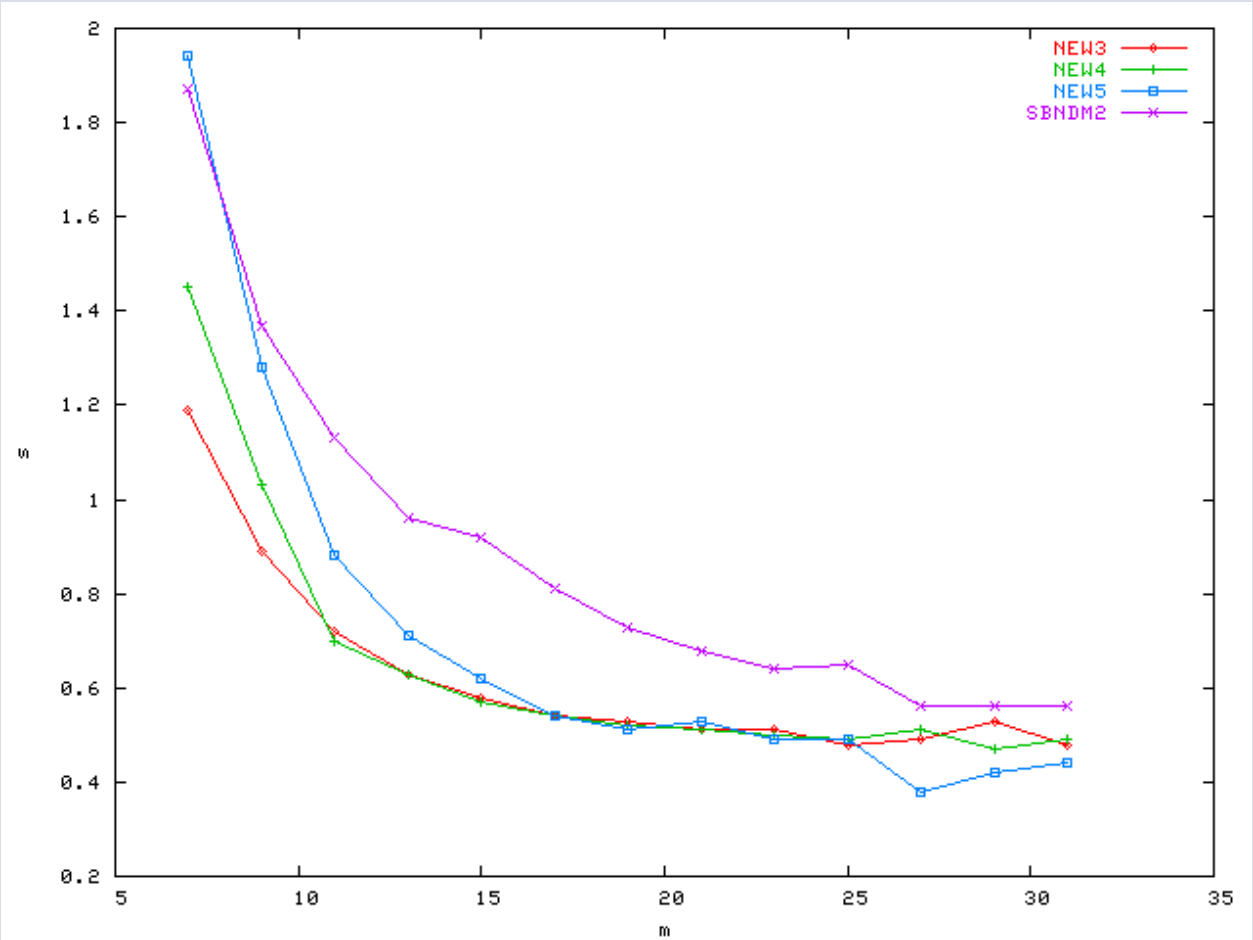
Résultats expérimentaux

Alphabet binaire



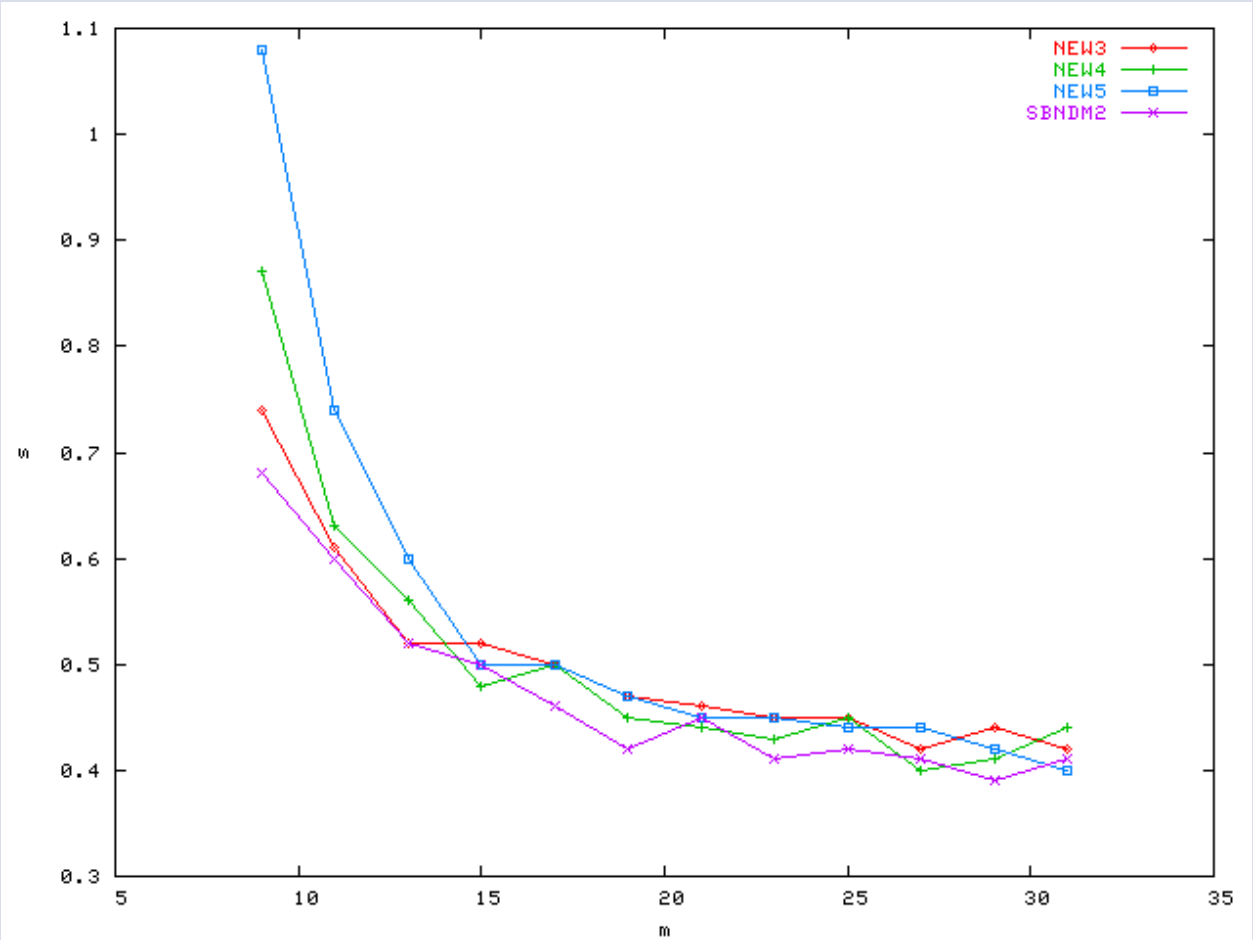
Résultats expérimentaux

E.coli du Large Canterbury corpus



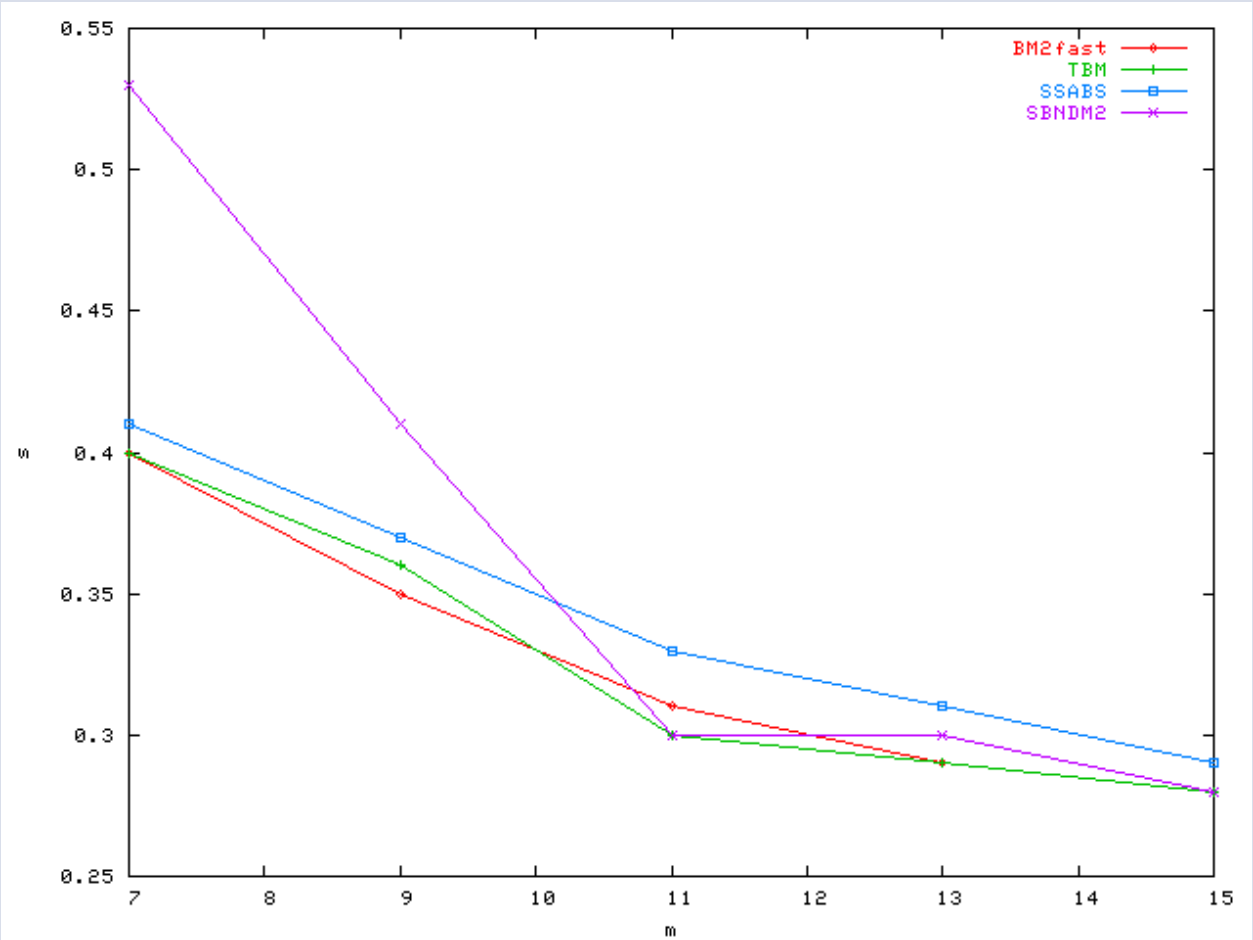
Résultats expérimentaux

Alphabet de cardinalité 8



Résultats expérimentaux

Langage naturel



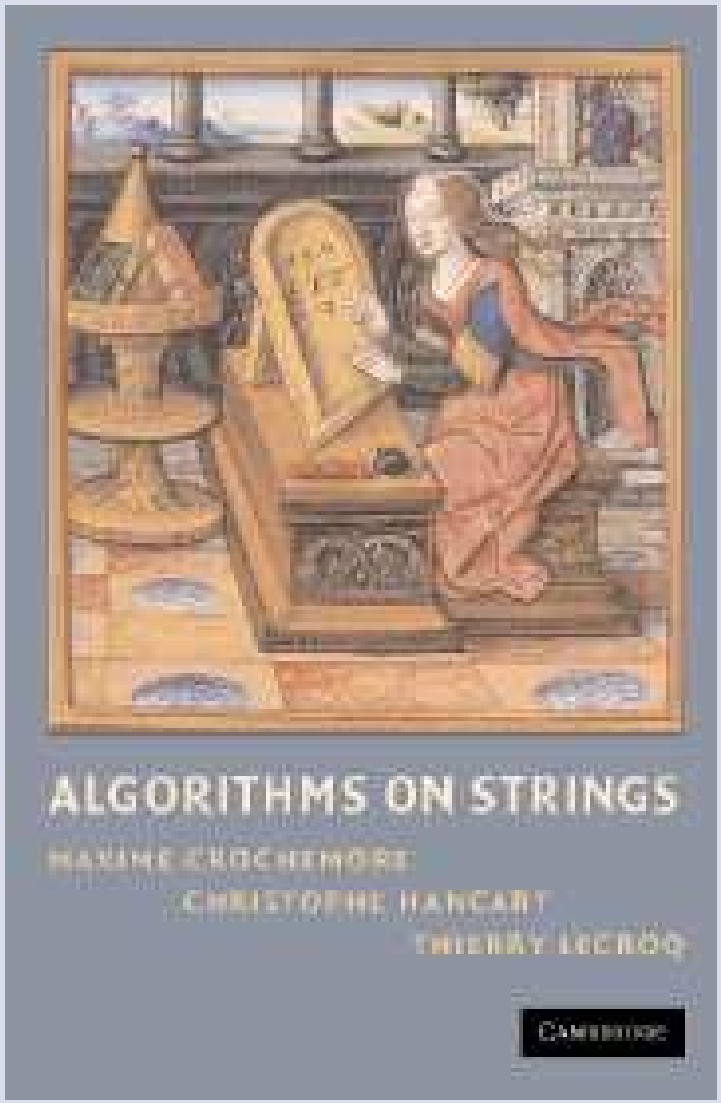
Conclusions

- Alphabet bianire : `NEW5-8` for $m \in [9; 256]$
- Alphabet de cardinalité 4 : `NEW3-5` for $m \in [7; 128]$
- Alphabet de cardinalité 8 : `NEW3-5` for $m \in [13; 64]$
- Langage naturel : `BM2fast` for $m \in [7; 15]$

Perspectives

Utiliser une meilleure fonction de hachage

Références



À paraître bientôt...
Meilleur décalage et bien d'autres choses encore plus intéressantes

Références


M. Crochemore et T. Lecroq
A fast implementation of the Boyer–Moore string matching algorithm
Soumis


T. Lecroq
Fast string matching algorithms
Information Processing Letters
À paraître