Université de Lille 1 SCIENCES ET TECHNOLOGIES

CRIStAL
Centre de Recherche en Informatique, Signal et Automatique de Lille

RECOMBINAISON VDJ

Réduire les comparaisons avec Aho-Corasick et un tableau associatif

---

Cyprien Borée

Janvier 2018 – Mars 2018

Responsable du label recherche : Sylvain Salvati
Responsable de l'équipe Bonsai : Hélène Touzet
Superviseurs : Mathieu Giraud et Mikaël Salson

Sommaire

ADN et immunité

Structure de l’ADN

Selon la classification du microbiologiste Carl Woese, les organismes vivants se regroupent en trois domaines. Les eubactéries[1], les archées[2] et les eucaryotes[3] (l’être humain appartenant à ce dernier).[1]Pour la plupart d’entre eux, le rôle de l’ADN est de fournir l’information génétique (définissant le comportement, le développement et la reproduction des espèces).[2]

L’ADN est une molécule présente chez tous les êtres vivants (à l’exception de certains virus)[3], c’est un polymère formé par la répétition de monomères appelés nucléotides. Ces nucléotides sont composés de bases nucléiques telles la guanine, la thymine, la cytosine ou encore l’adénine.[4][5]Dans les schémas et les outils dédiés, ces bases nucléiques sont désignés par leur initiale et l’unité de mesure associé est *nt*. La taille du génome humain est d’approximativement $3x10^9$nt.

```
ACAAGATGCCATTGTCCCCCGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCC
CCTGGAGGGTGGCCCCACCGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAATAAGGAAAAGCAGC
CTCCTGACTTTCCTCGCTTGGTGGTTTGAGTGGACCTCCCAGGCCAGTGCCGGGCCCCTCATAGGAGAGG
AAGCTCGGGAGGTGGCCAGGCGGCAGGAAGGCGCACCCCCCCAGCAATCCGCGCGCCGGGACAGAATGCC
```

Texte 1: Un exemple de séquence d'ADN

L’ADN est constitué de deux brins, où les bases nucléiques sont liées de manière complémentaires. Ainsi l’adénine ne peut se lier qu’à la thymine et la cytosine à la guanine. De ce fait, l’ADN est capable de se répliquer (mécanisme essentiel pour la division cellulaire) en séparant ses deux brins. Une fois séparés, ils peuvent chacun de leur côté reformer les bases nucléiques nécessaires (voir illustration 1).[6]

Chez les êtres humains la molécule d’ADN est présente dans chaque cellule du corps, qu’elle soit dans les os, les cheveux ou les neurones.[7]Néanmoins il existe deux cas pour lesquels la structure de l’ADN diffère. L’un d’eux est chez les lymphocytes B[4], aussi appelés globules blancs, chargés de fabriquer les immunoglobulines (anticorps[5]).[8][9]

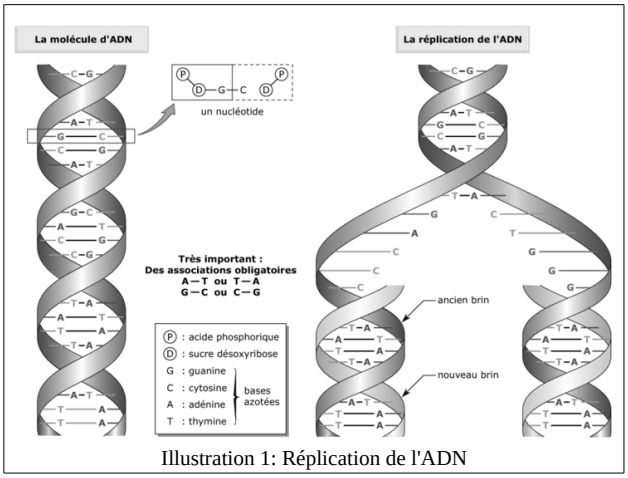

Illustration 1: Réplication de l'ADN

1 Le domaine des bactéries, micro-organismes procaryotes (qui ne contiennent pas de noyau).
2 Micro-organismes procaryotes, qui à la différence des bactéries, peuvent survivre à des conditions extrêmes.
3 Organismes constitués de cellules eucaryotes (avec un noyau), tels les végétaux, les champignons, les animaux, ...
4 L’autre cas étant pour les gamètes (les cellules reproductrices comme les ovules et spermatozoïdes).
5 Est anticorps, une protéine (globuline), produite par le système immunologique de l’organisme capable de réagir à la présence d’un antigène.

Rôle des lymphocytes B

Pour qu'un anticorps soit actif, il doit être en contact avec un antigène[6] reconnu. Les lymphocytes B sont des cellules participant à la réponse immunitaire adaptative, c'est-à-dire qu'un lymphocyte B donné ne peut réagir que contre un antigène précis (voir illustration 3). Cette spécificité est donnée par le récepteur des lymphocytes B (voir illustration 2). Pour que chaque lymphocyte B puisse reconnaître un antigène particulier, il s'opère une recombinaison de l'ADN sur son génome[7]. Cette recombinaison s'opère sur une région particulière de l'ADN connu sous le nom de VDJ. Cette combinaison (ou recombinaison) VDJ permet d'identifier un lymphocyte B d'un autre. Elle a été introduite par le microbiologiste Susumu Tonegawa[8].

Distinguer les lymphocytes B entre eux permet de les quantifier, suivre leur population. C'est ce qui est effectué dans de nombreux cas en médecine (connaître l'avancée des cellules cancéreuses, l'évolution d'un traitement, ...).

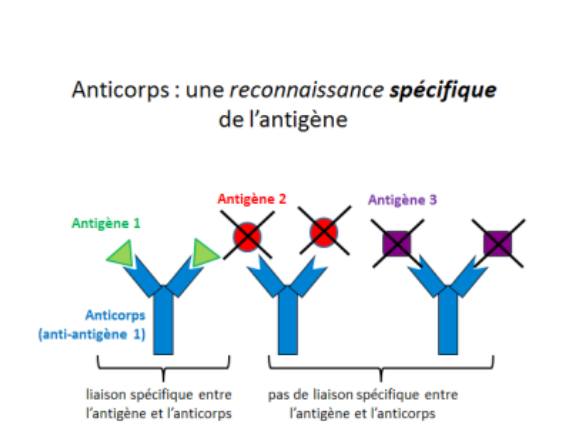

Illustration 2: Un anticorps spécifique par antigène

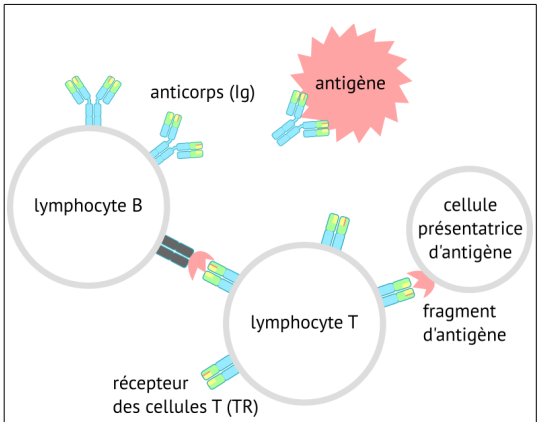

Illustration 3: récepteur des lymphocytes B en contact avec les antigènes

Recombinaison VDJ

Locus et taille des gènes

Les récepteurs des lymphocytes B sont composés de chaînes de protéines lourdes et légères. Chacune d'entre elles contient une partie constante et une partie variable encodés sur trois gènes se trouvant à trois loci[9] différents.

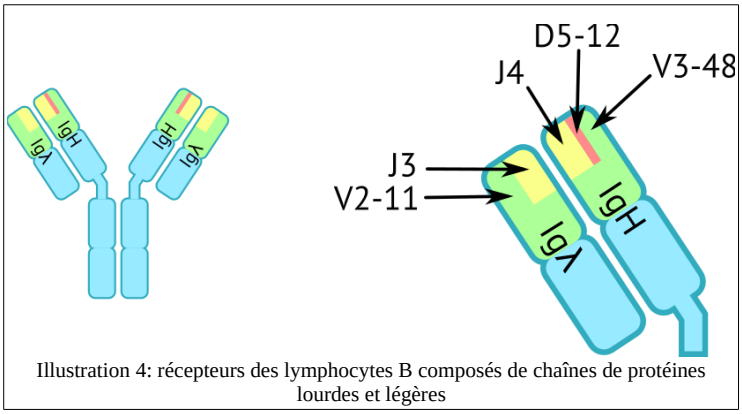

Illustration 4: récepteurs des lymphocytes B composés de chaînes de protéines lourdes et légères

- Locus n°1 : IGH → chromosome 14 : contient la séquence pour la chaîne lourde
- Locus n°2 : IGK(k) → chromosome 2 : contient la séquence pour une partie de la chaîne légère
- Locus n°3 : IGL(λ) → chromosome 22 : contient la séquence pour le reste de la chaîne légère

6 Est antigène, toute substance que le système immunologique reconnaît comme étrangère et qui provoque une réponse de la part d'anticorps.
7 Le génome est l'ensemble de l'ADN dans un organisme vivant (ou ARN pour les virus ARN).
8 Cela lui a valu un prix Nobel de médecine en 1987.
9 En génétique un locus (pluriel : loci) est une position sur un chromosome.

Qu'une chaîne de protéine soit lourde ou légère elle est constituée de deux parties majeurs : une partie variable et une partie constante. Dans le cas d'une chaîne lourde on retrouve trois types de segments appelés V (gènes variables), D (gènes de diversité) et J (gènes de jonction). On ajoute ensuite à cela la partie constante (C). Chez les chaînes légères les gènes de diversité (D) sont inexistants. Enfin, pour une chaîne lourde, on retrouve environ 50 gènes V, 20 gènes D, 6 gènes J et 9 gènes C. Ce qui donne 50 x 20 x 6 = 6000 possibilités auxquelles on ajoute les gènes constants (6009). Le nombre de possibilités est moindre chez les chaînes légères où les gènes de diversité sont inexistants.

Ordre de recombinaison

Dans le cas d'une chaîne lourde, la première recombinaison s'effectue D et un locus J. S'il y a de l'ADN entre ces deux, il est supprimé. Cette recombinaison D-J est suivie par la jointure au gène V. De même l'ADN entre le D et le V est supprimé. Pour les chaînes légères, le D étant absent, seul une jonction entre le V et le J a lieu (illustration 5).

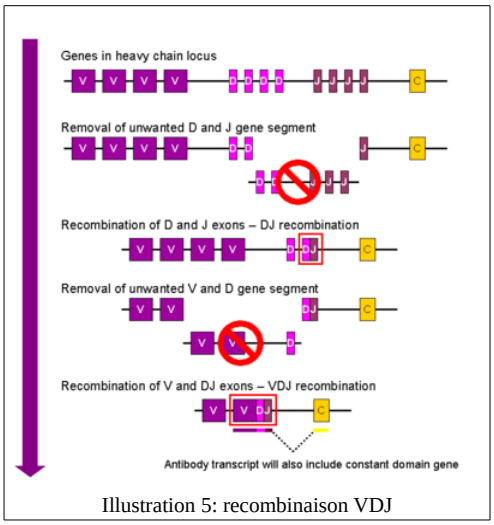

Illustration 5: recombinaison VDJ

Équipe Bonsai et VIDJIL

C'est sur ce principe de recombinaison VDJ que le logiciel VIDJIL[10] est développé depuis 2011 par l'équipe de recherche Bonsai au sein du laboratoire Cristal. Son but premier est d'identifier les recombinaisons VDJ propres à chaque lymphocyte B. L'interface se fait grâce à la ligne de commande et à une application web (depuis 2014).

L'application web permet la visualisation, l'inspection et l'analyse d'une population de lymphocytes au cours du temps. Elle permet également de visualiser les données calculées par l'algorithme principal de VIDJIL ou par d'autres outils d'analyse VDJ.

Ce projet est sous licence libre (GPLv3) et est supporté par la fondation INRIA. Il est indirectement rémunéré par les instituts de santé partenaires (L'institut d'Hématologie-Transfusion du CHRU de Lille par exemple). Le choix de la licence permet à de nombreux organismes médicaux de participer à l'utilisation du logiciel et de faire part de leurs retours.

Comme il est question d'analyser des séquences d'ADN, les données peuvent se révéler lourdes et donc il est nécessaire d'opter pour les algorithmes et les technologies les plus efficaces. Ainsi le logiciel est principalement écrit en C++11, couvert par des tests et soumit à de l'intégration continue.[10]

10 Le nom a été constitué de manière à contenir le V, le D et le J de la recombinaison VDJ.

L'algorithme principal du logiciel compare une séquence d'entrée à tous les gènes connus un à un. Les séquences d'ADN pouvant se révéler longues et nombreuses, cela n'entre pas dans la volonté du projet, où l'on préfère des calculs rapides et efficaces. C'est pourquoi la problématique à résoudre fût de réduire ce nombre de comparaisons et par conséquent le temps d'exécution. L'approche employée au sein de ce rapport aborde des notions de programmation dynamique et d'automate d'Aho-Corasick qui se veulent être solution du problème posé.

Programmation dynamique et alignements

Programmation dynamique

Pour trouver qu'une séquence d'ADN ressemble à tel gène connu, on applique sur elle un algorithme de programmation dynamique[11]. C'est-à-dire qu'on tente de résoudre des sous-problèmes avant le problème majeur.[11]Dans le cas qui nous intéresse, on souhaite connaître la proximité entre deux séquences d'ADN[12]. Pour ce faire, on décompose la comparaison en sous-problèmes, caractère par caractère.

Dans l'évaluation de deux séquences, on retrouve trois opérations : L'insertion de caractère (si la chaîne est trop courte), la suppression de caractère (si la chaîne est trop longue) et la substitution (on remplace un caractère par un autre). On peut ainsi définir un coût propre à chacune de ces trois opérations. Pour prioriser les séquences similaires, le coût d'une insertion et d'une délétion sont plus élevés qu'une substitution. Aussi, on peut définir le coût minimal comme étant la substitution d'un caractère par lui-même.

Dans l'exemple suivant, on compare la séquence ACGA et ATGCTA. Une insertion ou suppression coûte 2, une substitution quelconque coûte 1 et une substitution d'un caractère par lui-même coûte 0. La chaîne la plus similaire étant celle ayant le coût le moins élevé.

Substituer A par A	**A**CGA	0
Substituer C par T	A**T**GA	1
Substituer G par G	AT**G**A	0
Insérer C	ATG**C**A	2
Insérer T	ATGC**T**A	2
Substituer A par A	ATGCT**A**	0

Le coût final est donc 5. Si nous avions choisi la chaîne ATGC le résultat aurait été 4, ce qui aurait été plus avantageux.

Alignement global

En bio-informatique, on parle d'alignement de séquence pour référer à la comparaison de séquences entre elles afin de distinguer les plus similaires. Les méthodes présentées dans ce rapport appartiennent à la programmation dynamique, bien qu'il existe des méthodes heuristiques (programme BLAST).[12]

Il existe plusieurs alignements de séquence, le plus général étant l'alignement global. Il consiste à calculer la distance entre l'ensemble d'une séquence x et l'ensemble du séquence y. L'algorithme Needleman-Wunsch permet de réaliser des alignements globaux de manière optimale.

Pour calculer le score total d'une séquence alignée à une autre, on définit en premier lieu une matrice de similarité. Cette dernière doit contenir le coût de substitution entre les caractères de l'alphabet (ici A, C, G et T). Si substituer un caractère par un autre a le même coût pour tous les caractères alors la matrice est remplie de la même valeur, autrement on définit un nombre différent pour chaque (voir illustration 6).

11 La programmation dynamique repose sur le principe d'optimalité de Bellman
12 On parle de distance de Levenshtein pour mesurer la différence entre deux chaînes de caractères

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

Illustration 6: exemple matrice de similarité

Dans cette matrice de similarité, substituer un caractère par lui-même à un coût nul. Tandis que substituer n'importe quel caractère par un autre à un coût de 1, sauf dans le cas où on substitue un caractère par un « T » (qui vaut 2).

Nous devons également définir une pénalité de trou, là où il n'y aurait pas de caractères dans la séquence à comparer, ici nous prenons une pénalité de 3.

```
AGACTAGTTAC
CGA---GACGT
```

Illustration 7: exemple alignement séquence global

Avec les coûts cités précédemment et les séquences ci-dessus nous obtenons le résultat suivant :

$$S(A, C) + S(G, G) + 3 \times 3 + S(G, G) + S(T, A) + S(T, C) + S(A, G) + S(C, T)$$

$$1 + 0 + 3 \times 3 + 0 + 2 + 2 + 1 + 2 = 17$$

Alignement local

À la différence de l'alignement global, l'alignement local se contente de calculer la distance entre un sous-mot d'une séquence x et un sous-mot d'une distance y. On utilise généralement l'algorithme de Smith-Waterman[13] pour le réaliser.[13][14] Le principe est de choisir le score le plus avantageux entre deux caractères en se basant sur les scores déjà calculés précédemment. Pour cet exemple on choisira le score le plus haut (même si les coûts sont négatifs le minimum reste 0).

On définit une matrice de taille (longueur de x, longueur de y) où x et y seront posés sur les axes. La première ligne ainsi que la première colonne sont remplis de 0, les coûts des opérations sont définis sur l'illustration n°9. Sur l'illustration n°8 la matrice a été calculée en utilisant la matrice de similarité BLOSUM62 et un coût d'indel Δ de -6. Le meilleur score de la matrice (+20) est indiqué en rose. Le chemin pour arriver à ce score est surligné en jaune. L'alignement résultant est indiqué à droite.

$$M(i,j) = \max \begin{cases} 0 \\ M(i-1,j-1) + D(A_i, B_j) \\ M(i-1,j) + \Delta \\ M(i,j-1) + \Delta \end{cases}$$

Illustration 9: coûts exemple alignement local

		F	A	T	C	A	T	Y
	0	0	0	0	0	0	0	0
T	0	0	0	5	0	0	5	0
C	0	0	0	0	14	8	2	3
A	0	0	4	0	8	18	12	6
G	0	0	0	2	2	12	16	10
S	0	0	0	5	1	6	17	14
F	0	6	0	0	3	0	11	20
A	0	0	10	4	0	7	5	14

```
FATCA-TY
 ||| ::
 TCAGSFA
```

Illustration 8: matrice exemple alignement local

Il existe aussi un alignement semi-global dont le principe est basé sur celui de l'alignement local. La différence est qu'un alignement semi-global opère sur l'ensemble d'une séquence x et un sous-mot d'une séquence y. L'algorithme reste le même.

13 Il existe également la méthode de recherche heuristique BLAST

Aho-Corasick

Intérêts de l'automate

Comme décrit précédemment, la problématique est d'optimiser l'algorithme principal de VIDJIL. Plus précisément il s'agit de réduire le nombre de comparaisons entre les gènes[14]. Un des moyens de réduire ce nombre est de ne sélectionner que les gènes les plus intéressants.[15]

L'automate d'Aho-Corasick permet de compter le nombre de fois qu'un gène est reconnu. Autrement dit, il permet de créer une table de correspondance [G, N] où G est le gène connu et N le nombre de fois qu'il apparaît dans la séquence à analyser. La dernière étape consiste à trier cette table sur le critère N en ne sélectionnant que les plus grands. Ainsi il devient possible d'aligner la séquence seulement sur les gènes apparaissant le plus de fois.

A l'origine, l'algorithme d'Aho-Corasick était utilisé dans l'utilitaire grep, disponible sous UNIX. Sa complexité algorithmique (complexité linéaire), en fait un algorithme de choix pour la recherche de motif.

Construction

L'automate d'Aho-Corasick pour un ensemble de mots X est un arbre orienté K satisfaisant les 4 contraintes suivantes :

- Chaque arrête est étiquetée par un et un seul caractère
- Deux arrêtes sortantes d'un même sommet ont des étiquettes différentes
- Chaque mot x de X est associé à un sommet v de K : les caractères étiquetant le chemin de la racine de K à v forment le mot x.
- Chaque feuille de K est associé à un mot de X.

Si on prend X l'ensemble de mots {GTAGCT, TAG, AGCT, CTAGT} on peut construire l'automate de la manière suivante :

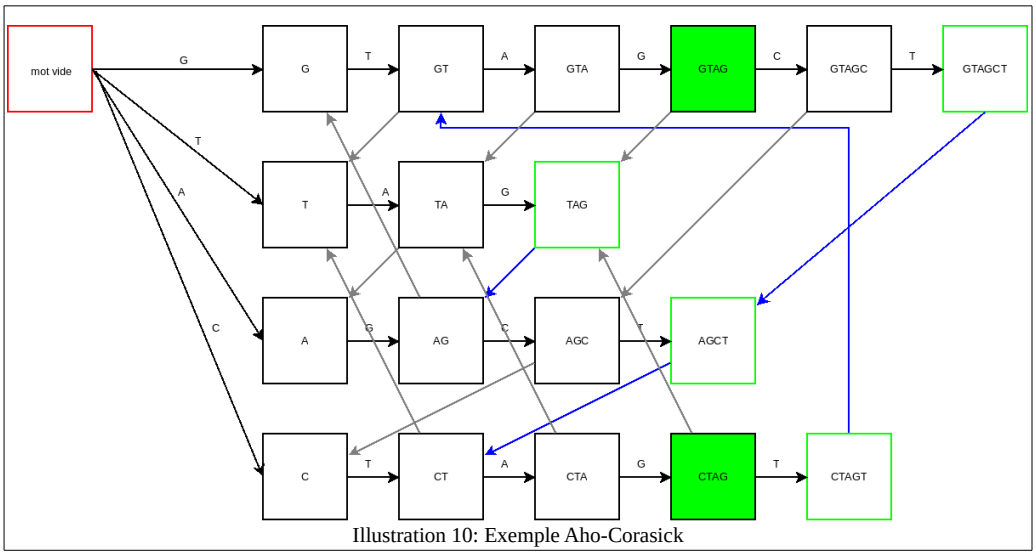

Illustration 10: Exemple Aho-Corasick

14 Nous pourrions également opter pour une méthode de comparaison plus optimale, paramètre difficile à réaliser étant donné que les alignements et la programmation dynamique sont ce qu'il y a de plus performant.

Si nous prenons la séquence GTAGCTGTACGTTAG sur l’exemple illustré précédemment, nous obtenons la table de correspondance suivante :

GTAGCT	2
TAG	3
AGCT	0
CTAGT	0

Les gènes connus les plus probants sont GTAGCT et TAG, donc on peut effectuer un alignement sur ces deux gènes seulement et non sur tous.

L’emploi d’Aho-Corasick permet de réduire le nombre de comparaisons et de passer d’une complexité quadratique à une complexité linéaire. Il était primordial de réduire ce nombre là où de grandes séquences de gènes étaient inutilement alignés.

Malgré l’implémentation déjà existante d’Aho-Corasick au sein de VIDJIL, il restait à en réaliser son utilisation. Le travail réalisé autour de ce rapport a permis de créer une table de correspondance (une Map<G, N>) avec les résultats de l’automate et de filtrer ceux qui apparaissent le plus. Cette dernière fonctionnalité n’ayant pu se concrétiser à temps, il est impossible d’obtenir un comparatif sur l’échelle du temps d’exécution.

Bibliographie

[1]: C. R. Woese, W. E. Balch, L. J. Magrum, G. E. Fox et R. S. Wolfe, "An ancient divergence among the bacteria", 1977
[2]: A. D. Hershey et Martha Chase , "Independent functions of viral protein and nucleic acid in growth of bacteriophage", 1952
[3]: Daniel Kolafofsky, "A short biased history of RNA viruses", 2015
[4]: Alberts B, Johnson A, Lewis J, Raff M, Roberts K, Walter P, Molecular Biology of the Cell, 2014
[5]: Purcell A., Basic Biology, 2017
[6]: Roger Prat et Gilles Furelaud, Expérience de Meselson et Stahl, 2002
[7]: Dahm, R, "Discovering DNA: Friedrich Miescher and the early years of nucleic acid research", 2008
[8]: Susumu Tonegawa, "Somatic generation of antibody diversity", 1983
[9]: Cooper, Max D., "The early history of B cells", 2015
[10]: Vidjil team, www.vidjil.org, 2011
[11]: Richard Bellman, Dynamic Programing, 1957
[12]: S. B. Needleman et C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins", 1970
[13]: Temple F. Smith et Michael S. Waterman, « Identification of Common Molecular Subsequences », 1981
[14]: Frédéric Dardel et François Képès, "Bioinformatique. Génomique et post-génomique", 2002
[15]: Alfred V. Aho et Margaret J. Corasick, "Efficient string matching: An aid to bibliographic search", 1975

Illustration Index