Université des Sciences et de la Technologie Houari Boumediene
Faculté d'Informatique

Bio ALGO

Mini projet TP : Les tables des suffixes

Fait par :
Nom et prénom : ABDELMALEK BENMEZIANE
Matricule : 171731046778
Spécialité : M1 BIOINFO
Section : A

Contents

1 Introduction et objectifs du travail **1**

1.1 Introduction 1

1.2 Objectifs du travail 2

2 Description des structures de données utilisées et leur complexité spatiale **3**

2.1 La table des suffixes (TS) 3

2.1.1 Définition 3

2.1.2 La complexité spatiale 3

2.2 La table des préfixes communs (HTR) 4

2.2.1 Définition 4

2.2.2 La complexité spatiale 4

2.3 L'inverse de la table des suffixes (ITS) 4

2.3.1 Définition 4

2.3.2 La complexité spatiale 4

2.4 La table LgCandidat 5

2.4.1 Définition 5

2.4.2 La complexité spatiale 5

3 Pseudo algorithmes (pour chaque traitement) **6**

3.1 Construire la table des suffixes TS 6

3.2 La recherche d'un motif M 6

3.3 Construire la table HTR 7

3.4 En utilisant les tables TS et HTP 8
3.4.1 le(s) plus long(s) facteur(s) répété(s) 8
3.4.2 les facteurs qui se répètent au moins 3 fois 8
3.5 Construire l'inverse de la table des suffixes ITS 9
3.6 Déterminer les plus courts facteurs uniques en construisant la table LgCandidat 9
3.7 Les répétitions super-maximales 10
3.8 Le plus long facteur commun entre deux textes T1 et T2 . . . 11

4 Les tableaux des tests 12
4.1 Construire la table des suffixes TS 12
4.2 La recherche d'un motif M 12
4.3 Construire la table HTR 12
4.4 En utilisant les tables TS et HTR 12
4.4.1 le(s) plus long(s) facteur(s) répété(s) 12
4.4.2 les facteurs qui se répètent au moins 3 fois 13
4.5 Construire l'inverse de la table des suffixes ITS 13
4.6 Déterminer les plus courts facteurs uniques en con- struisant la table LgCandidat 13
4.7 Les répétitions super-maximales 13
4.8 Le plus long facteur commun entre deux textes T1 et T2 . . . 13

5 Les courbes (ou diagrammes) de variation temporelle 14
5.1 Construire la table des suffixes TS 14
5.2 La recherche d'un motif M 15
5.3 Construire la table HTR 16
5.4 En utilisant les tables TS et HTR 17
5.4.1 le(s) plus long(s) facteur(s) répété(s) 17
5.4.2 les facteurs qui se répètent au moins 3 fois 18
5.5 Construire l'inverse de la table des suffixes ITS 19

5.6 Déterminer les plus courts facteurs uniques en con- struisant la table LgCandidat 20
5.7 Les répétitions super-maximales 21
5.8 Le plus long facteur commun entre deux textes T1 et T2 . . . 22

6 Analyse des résultats et conclusions 23
6.1 Analyses et conclusions 23
6.2 Complexités temporelles théoriques pour quelques traitements 23

7 Le code source 24

List of Figures

5.1 Courbe 1 14
5.2 Courbe 2 15
5.3 Courbe 3 16
5.4 Courbe 4.1 17
5.5 Courbe 4.2 18
5.6 Courbe 5 19
5.7 Courbe 6 20
5.8 Courbe 7 21
5.9 Courbe 8 22

Chapter 1

Introduction et objectifs du travail

1.1 Introduction

Travailler sur les grandes chaines de caractères et traiter celles-ci afin d'en extraire les motifs récurrents par correspondance exacte ou approchée est un travail aujourd'hui courant pour les chercheurs en Sciences de la Vie ou en fouille de données. Si l'on ne considère que la base de données GenBank, la taille et le nombre des séquences qui y sont stockées doubles tous les 16 mois.Cette augmentation nécessite par conséquent le développement de méthodes à même de répondre rapidement à denombreuses requêtes sur des séquences toujours plus grandes, éventuellement distantes.

Les tables de suffixes sont une structure de données à même d'offrir une réponse rapide à des requetes s'apparentant à la recherche de patrons au sein d'une chaine de caractères.Ainsi,la recherche d'une correspondance exacte entre une chaine de caractères fournie en entrée et les sous-chaine d'une chaine de données peut être résolue en un temps proportionnel à la longeur de la chaine d'entrée.

1.2 Objectifs du travail

- ✓ L'implémentation d'une structure d'index permettant d'accélérer la recherche de motifs, de répétitions,...etc.
- ✓ Analyse de la complexité spatiale des structures de données utilisées.
- ✓ Analyse de la complexité temporelle des traitements.

Chapter 2

Description des structures de données utilisées et leur complexité spatiale

2.1 La table des suffixes (TS)

2.1.1 Définition

Un tableau des suffixes (parfois nommé table des suffixes, en anglais : suffix array) est une structure de données utilisée en informatique, et plus particulièrement en combinatoire des mots et en bio-informatique. Pour un mot donné, le tableau contient une liste d'entiers qui correspondent aux positions de début des suffixes du mot, lorsqu'ils sont triés selon l'ordre lexicographique.
L'objectif du tableau est de fournir les mêmes facilités de recherche qu'un arbre des suffixes tout en réduisant la taille mémoire utilisée. La structure a été introduite en 1990 par Manber et Myers et redécouverte en 1992.

2.1.2 La complexité spatiale

Sa complexité spatiale est : $O(n)$

2.2 La table des préfixes communs (HTR)

2.2.1 Définition

Le tableau des plus long préfixes commun entre deux suffixes consécutives est une structure de données auxiliaire du tableau de suffixes. Il stocke les longueurs des préfixes communs les plus longs entre toutes les paires de suffixes consécutifs dans un tableau de suffixes trié.

Par exemple, si A := [aab, ab, abaab, b, baab] est un tableau de suffixes, le plus long préfixe commun entre A[1] = aab et A[2] = ab est a qui a une longueur de 1, donc HTR [2] = 1 dans le tableau HTR. De même, le plus long préfixe commun de A[2] = ab et A[3] = abaab est ab, donc HTR[3] = 2.

2.2.2 La complexité spatiale

Sa complexité spatiale est : $O(n)$

2.3 L'inverse de la table des suffixes (ITS)

2.3.1 Définition

Linverse de la table des suffixes est une structure de données de telle sorte que l'indice du table de suffixes devient l'indice de début du suffixe et l'indice de début du suffixe devient indice du table, c'est à dire : ITS[j] = i équivaut TS[i] = j.
Par exemple, on a dans la table des suffixes TS[0] = 15, quand on construit la table ITS on aure ITS[15] = 0.

2.3.2 La complexité spatiale

Sa complexité spatiale est : $O(n)$

2.4 La table LgCandidat

2.4.1 Définition

La table LgCandidat est une table construite à partir de la table HTR et la table ITS, son intérêt est de déduire les plus courts facteurs uniques et les répétitions super-maximales.

2.4.2 La complexité spatiale

Sa complexité spatiale est : $O(n)$

Chapter 3

Pseudo algorithmes (pour chaque traitement)

3.1 Construire la table des suffixes TS

1. Générer tous les suffixes de la chaîne de caractères.
2. Trier ces suffixes par ordre lexicographique
3. Mettre les indices de début de chaque suffixe dans chaque case dans une table en suivant l'ordre lexicographique.

3.2 La recherche d'un motif M

1. L'algorithme effectue une recherche dichotomique sur la table des suffixes.
2. Nous commençons par initialiser deux indices (l=0) et (r = n-1)
3. Si le suffixe trouvé correspond au motif M, l'algorithme retourne l'indice de départ de ce suffixe dans la chaîne S.
4. Sinon, l'algorithme retourne -1 pour indiquer que le motif M n'a pas été trouvé dans la chaîne S.

3.3 Construire la table HTR

1. Nous commençons par initialiser une table de taille ‘n‘ (où ‘n‘ est la longueur de la chaîne de caractères ‘S‘) à 0.
2. L’initialisons également une variable ‘k‘ à 0 qui représente la longueur du préfixe commun actuellement considéré.
3. On parcourons ensuite la table des suffixes ‘tableSuffixes‘, en partant de la première position jusqu’à l’avant-dernière position.
4. Si le suffixe à la position ‘i‘ dans ‘tableSuffixes‘ est le dernier suffixe (c’est-à-dire s’il s’agit du suffixe correspondant à la chaîne de caractères entière), alors nous réinitialisons ‘k‘ à 0 et continuons avec l’itération suivante.
5. Sinon, nous récupérons les positions dans ‘S‘ des deux suffixes adjacents ‘i‘ et ‘i+1‘, en utilisant les valeurs de ‘tableSuffixes[i]‘ et ‘tableSuffixes[i+1]‘ respectivement. Nous commençons ensuite à comparer les caractères de ‘S‘ à partir de ces positions, en les comparant deux à deux.
6. Tant que les caractères comparés sont égaux et que nous ne sommes pas à la fin de l’une des chaînes, nous continuons à comparer les caractères suivants. À chaque comparaison réussie, nous augmentons la valeur de ‘k‘ de 1.
7. Lorsque la comparaison échoue (c’est-à-dire lorsque deux caractères différents sont comparés ou que nous atteignons la fin d’une des chaînes), nous stockons la valeur actuelle de ‘k‘ dans ‘LCP[i]‘, qui représente la longueur du plus long préfixe commun entre les suffixes ‘i‘ et ‘i+1‘.
8. Si ‘k‘ est supérieur à 0, cela signifie que nous avons trouvé un préfixe commun entre les suffixes ‘i‘ et ‘i+1‘ de longueur ‘k‘. Dans ce cas, nous décrémentons ‘k‘ de 1, pour nous assurer que le préfixe commun suivant que nous allons comparer aura bien une longueur inférieure à celui-ci.

9. Une fois que nous avons parcouru toute la table des suffixes, la table 'LCP' contient les longueurs des plus longs préfixes communs entre chaque paire de suffixes consécutifs. Cette table peut être utilisée dans l'algorithme de recherche de motifs de la même manière que la table des suffixes.

3.4 En utilisant les tables TS et HTP

3.4.1 le(s) plus long(s) facteur(s) répété(s)

1. L'initialisons d'une liste pour stocker les plus longues sous-chaînes communes trouvées.
2. Nous parcourons la table HTR, et pour chaque paire de suffixes adjacents ayant un préfixe commun de longueur max, nous récupérons la sous-chaîne correspondant à ce préfixe commun en utilisant la position des suffixes dans TS.
3. Nous vérifions que cette sous-chaîne n'a pas déjà été trouvée et ajoutée à la liste. Si ce n'est pas le cas, nous l'ajoutons.
4. Une fois que nous avons parcouru toute la table HTR, nous retournons la liste des plus longues sous-chaînes communes trouvées.

3.4.2 les facteurs qui se répètent au moins 3 fois

1. Nous commençons par récupérer la longueur n de la chaîne s.
2. L'initialisons une chaîne vide 'result' pour stocker la sous-chaîne répétée la plus longue trouvée.
3. On parcourons la table HTR, et pour chaque paire de suffixes adjacents ayant un préfixe commun de longueur supérieure ou égale à 1, nous cherchons les suffixes suivants ayant un préfixe commun d'au moins la même longueur. Cela se fait en comparant les valeurs HTR[j] avec la

longueur initiale 'length'.

4. Nous comptons le nombre de suffixes qui ont un préfixe commun de longueur au moins égale à length et stockons ce nombre dans la variable count.
5. Si count est supérieur ou égal à 3, cela signifie que nous avons trouvé une sous-chaîne répétée d'au moins trois fois. Nous récupérons cette sous-chaîne répétée en utilisant la position des suffixes dans TS.
6. Nous vérifions si la longueur de cette sous-chaîne répétée est supérieure à celle de result. Si c'est le cas, nous mettons à jour 'result' avec cette sous-chaîne répétée.
7. Une fois que nous avons parcouru toute la table HTR, nous retournons la sous-chaîne répétée la plus longue trouvée.

3.5 Construire l'inverse de la table des suffixes ITS

1. On crée un tableau ITS de même taille que TS, initialisé avec des zéros.
2. On remplit ITS en parcourant TS et en stockant l'indice de chaque élément de TS dans ITS à la position correspondante.
3. Finalement, la fonction retourne le tableau ITS.

3.6 Déterminer les plus courts facteurs uniques en construisant la table LgCandidat

1. On commence par construire le tableau TS contenant les indices des suffixes de T.
2. Ensuite, on construit la table HTR.
3. Après, on construit le tableau ITS.
4. On construit la table lgCandidat1 en parcourant les indices i de 0 à n-1,

en calculant pour chaque indice i la longueur du plus long motif répété commençant par le suffixe i et stockant cette valeur dans lgCandidat1[i].

5. On filtre les motifs candidats en parcourant lgCandidat1 et en ajoutant les motifs correspondant aux indices i tels que i + lgCandidat1[i] <= n et lgCandidat1[i] <= lgCandidat1[i+1] à la liste resultat.
6. Finalement, on retourne la liste resultat contenant les motifs répétés de T.

3.7 Les répétitions super-maximales

1. Construction de la table des suffixes TS et la table HTR correspondante pour la chaîne de caractères.
2. On Parcourons les entrées de la table HTR, et pour chaque entrée (i, lcp[i]), suivez les étapes suivantes :

✓ Si lcp[i] est supérieur à la longueur de la plus longue répétition super-maximale trouvée jusqu'à présent, alors cela peut être une nouvelle répétition super-maximale.

✓ Trouvez toutes les entrées j dans la table des suffixes pour lesquelles lcp[j] est supérieur ou égal à lcp[i].

✓ Pour chaque entrée j, vérifiez si le suffixe correspondant s[SA[j]:] contient la sous-chaîne correspondant à la répétition super-maximale potentielle s[SA[i]:SA[i]+lcp[i]]. Si c'est le cas, alors cette sous-chaîne est une répétition super-maximale.

✓ Si vous trouvez une nouvelle répétition super-maximale, ajoutez-la à une liste des répétitions super-maximales.

3. Retournez la liste de toutes les répétitions super-maximales trouvées.

3.8 Le plus long facteur commun entre deux textes T1 et T2

1. Concaténer les deux textes txt1 et txt2 en ajoutant un caractère spécial pour distinguer leurs limites $ pour txt1 et # pour txt2). Le résultat est stocké dans txt.
2. Déterminer la longueur n du texte concaténé txt.
3. Construire le tableau des suffixes TS pour le texte concaténé txt.
4. Construire la table des préfixes communs HTR pour le texte concaténé txt à partir du tableau des suffixes TS.
5. Initialiser la longueur maximale max_len à 0 et la position de début start_pos à -1.
6. Itérer sur la table des préfixes communs htr pour trouver le plus grand préfixe commun à txt1 et txt2 et sa position de début correspondante :

✓ Si le suffixe associé à l'élément courant de la table des préfixes HTR commence par txt1 et que le suffixe suivant ne commence pas par txt1, alors le préfixe commun est trouvé.

✓ Si la longueur de ce préfixe commun est supérieure à max_len, alors mettre à jour max_len et start_pos avec la nouvelle longueur et la nouvelle position de début respectivement.

✓ Si la longueur maximale max_len est supérieure à 0, alors retourner le plus long préfixe commun trouvé dans le texte concaténé txt. Sinon, retourner une chaîne vide.

Chapter 4

Les tableaux des tests

4.1 Construire la table des suffixes TS

Taille de text	100	200	600	1000	2000	5000
Temps exécution	0.02	0.14	0.24	1.37	3.61	25.73

4.2 La recherche d'un motif M

Taille de text	100	200	600	1000	2000	5000
Temps exécution	0.001	0.002	0.002	0.005	0.01	0.05

4.3 Construire la table HTR

Taille de text	100	200	600	1000	2000	5000
Temps exécution	0.06	0.17	0.86	2.30	7.05	37.82

4.4 En utilisant les tables TS et HTR

4.4.1 le(s) plus long(s) facteur(s) répété(s)

Taille de text	100	200	600	1000	2000	5000
Temps exécution	0.001	0.002	0.006	0.006	0.01	0.06

4.4.2 les facteurs qui se répètent au moins 3 fois

Taille de text	100	200	600	1000	2000	5000
Temps exécution	0.002	0.004	0.002	0.015	0.024	0.71

4.5 Construire l'inverse de la table des suffixes ITS

Taille de text	100	200	600	1000	2000	5000
Temps exécution	0.001	0.002	0.004	0.008	0.024	0.073

4.6 Déterminer les plus courts facteurs uniques en construisant la table LgCandidat

Taille de text	100	200	600	1000	2000	5000
Temps exécution	0.001	0.005	0.012	0.013	0.031	0.090

4.7 Les répétitions super-maximales

Taille de text	100	200	600	1000	2000	5000
Temps exécution	0.0	0.001	0.003	0.005	0.013	0.083

4.8 Le plus long facteur commun entre deux textes T1 et T2

Taille de text	100	200	600	1000	2000	5000
Temps exécution	0.003	0.006	0.006	0.011	0.040	0.129

Chapter 5

Les courbes (ou diagrammes) de variation temporelle

5.1 Construire la table des suffixes TS

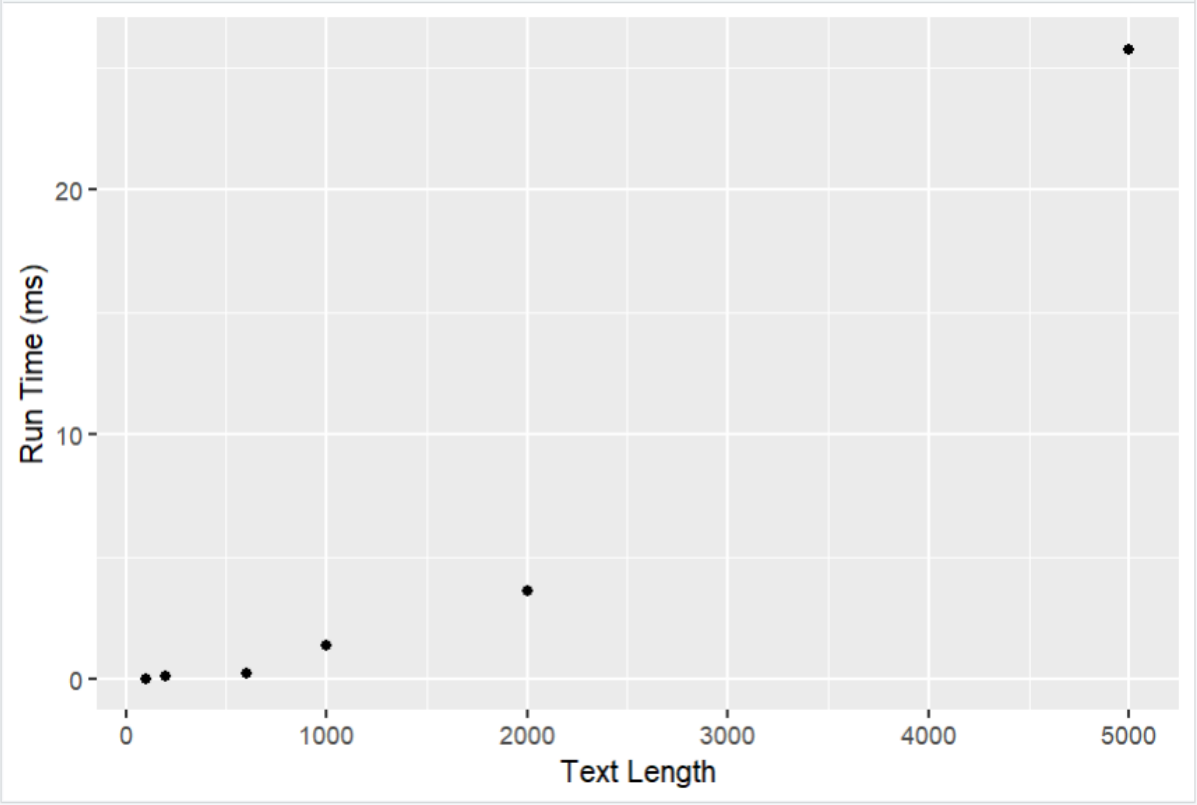

Figure 5.1: Courbe 1

5.2 La recherche d'un motif M

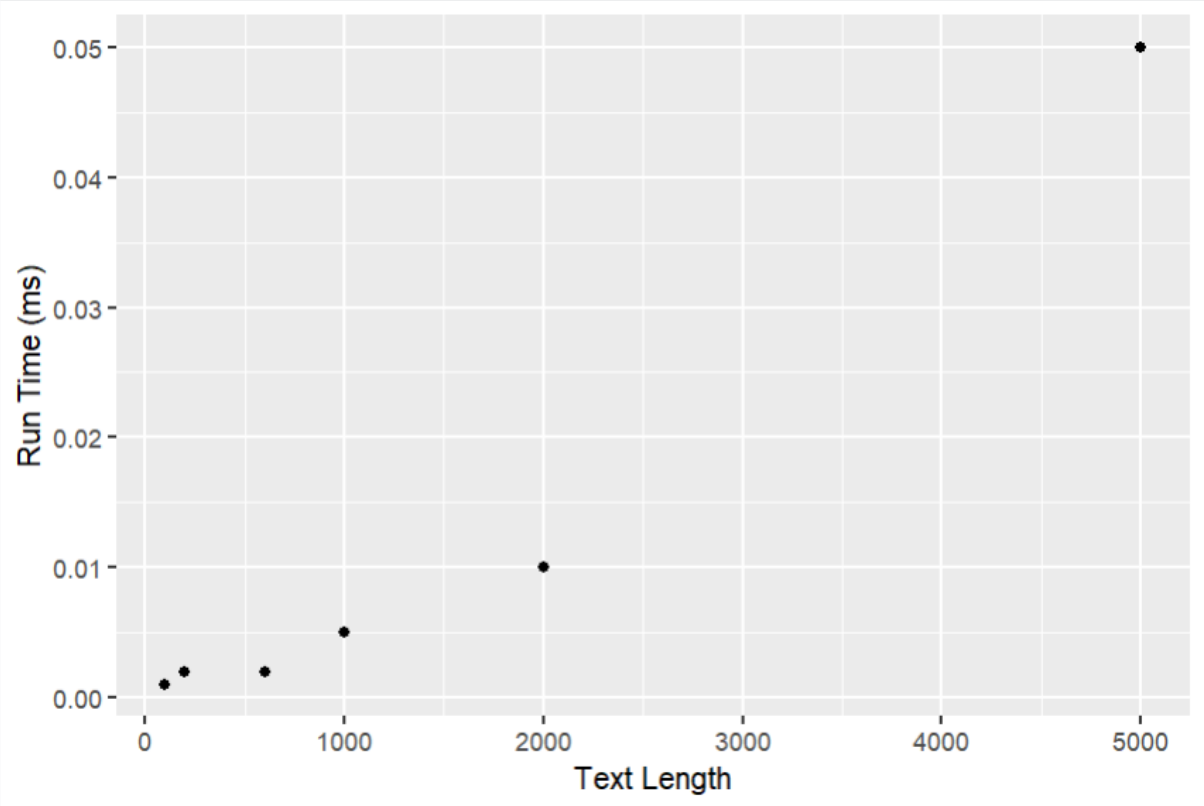

Figure 5.2: Courbe 2

5.3 Construire la table HTR

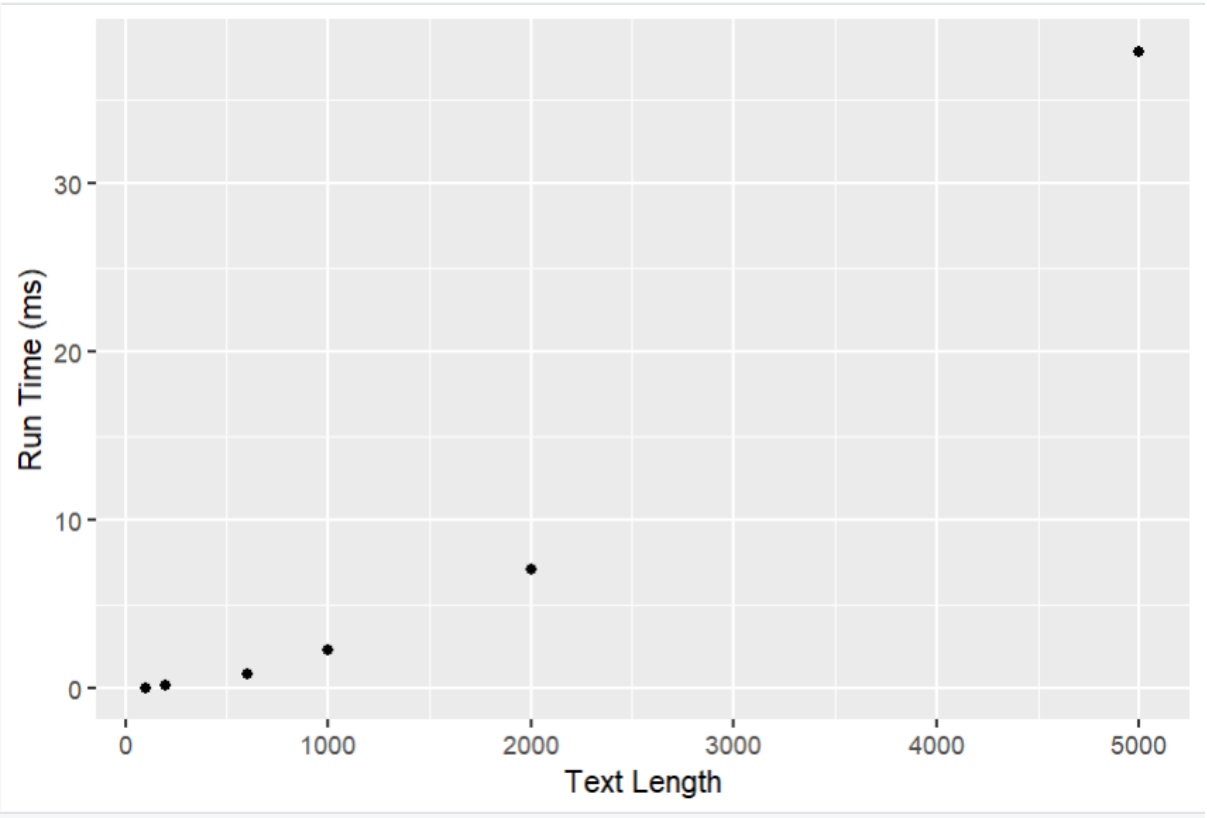

Figure 5.3: Courbe 3

5.4 En utilisant les tables TS et HTR

5.4.1 le(s) plus long(s) facteur(s) répété(s)

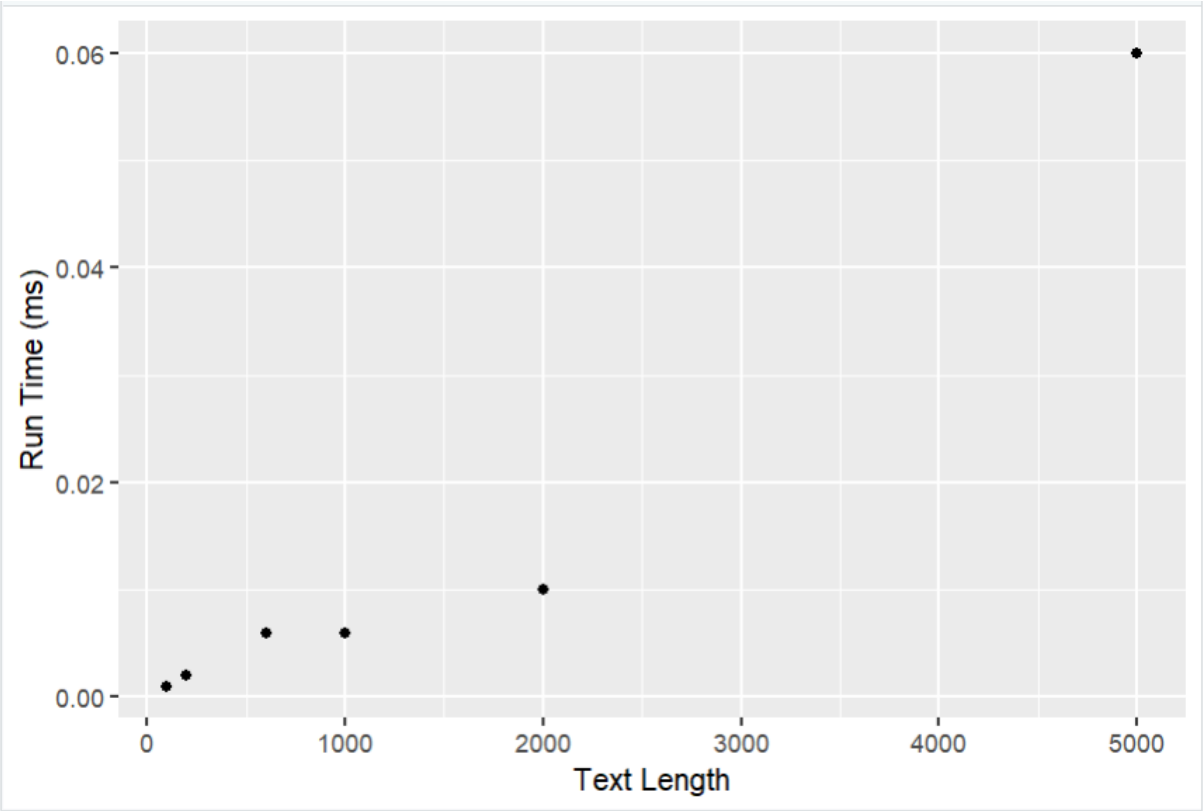

Figure 5.4: Courbe 4.1

5.4.2 les facteurs qui se répètent au moins 3 fois

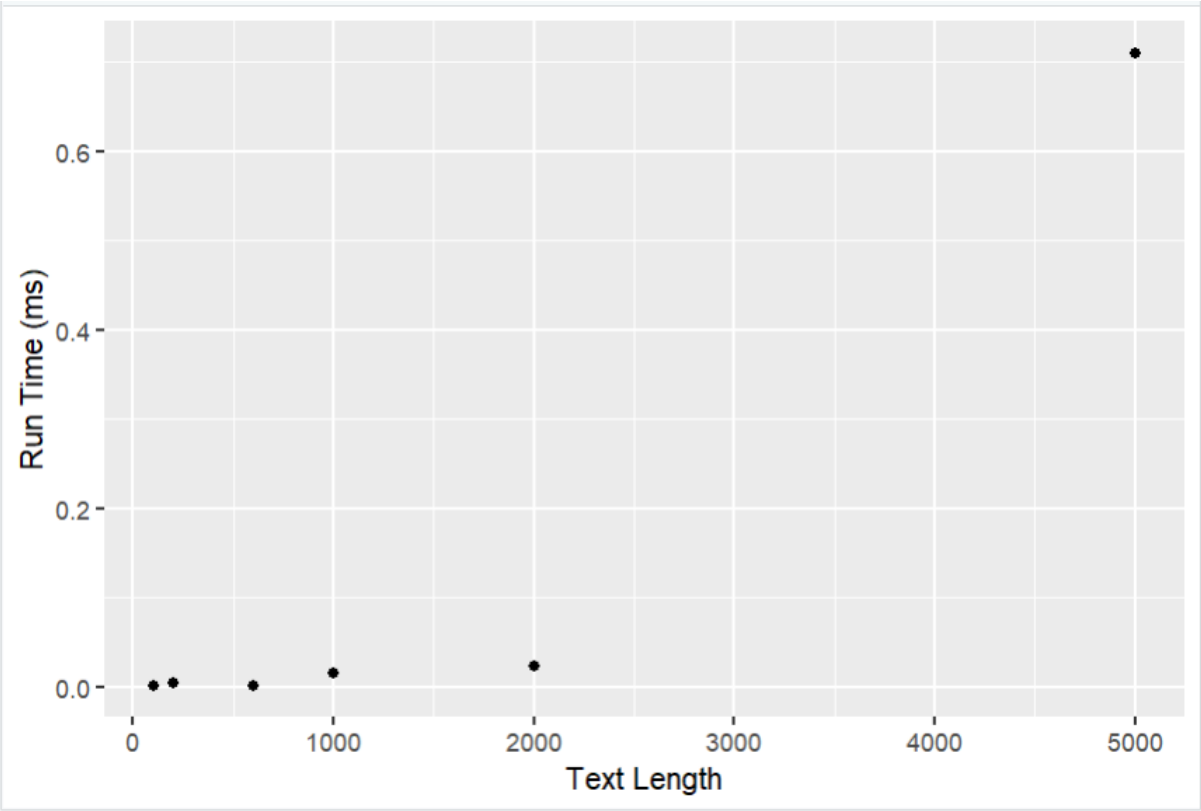

Figure 5.5: Courbe 4.2

5.5 Construire l'inverse de la table des suffixes ITS

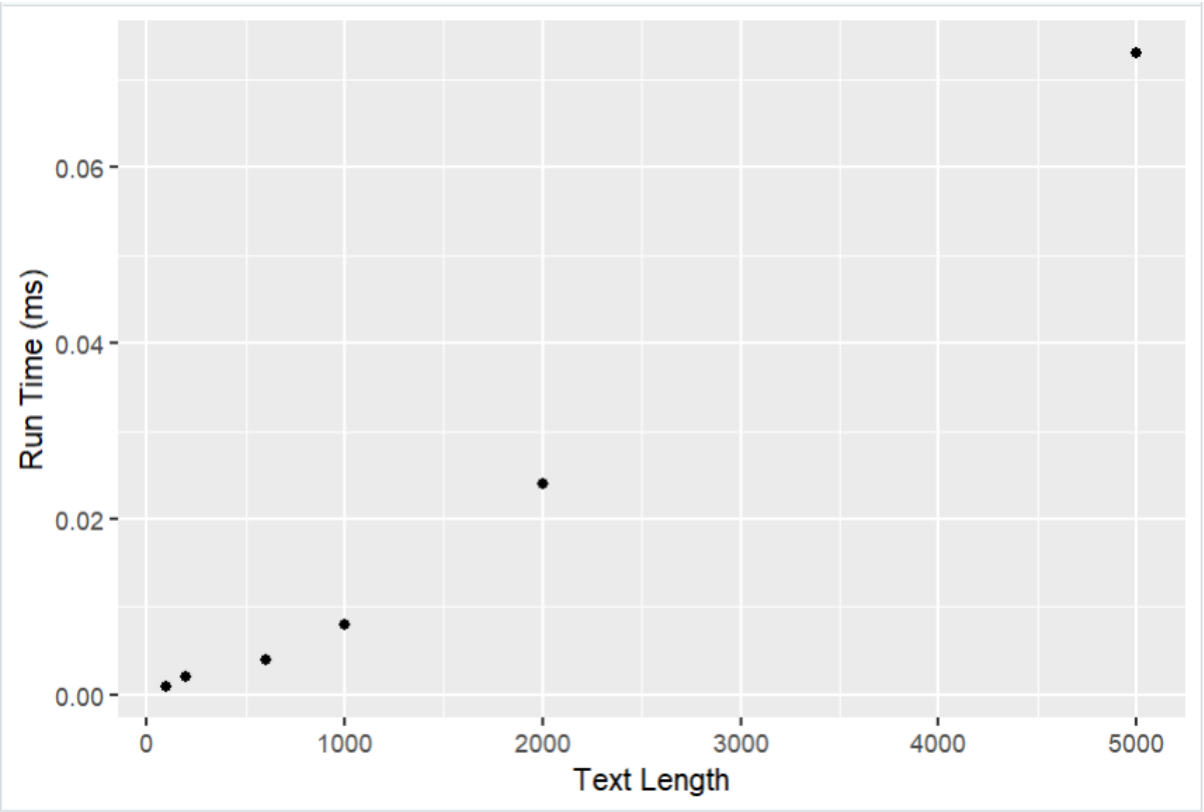

Figure 5.6: Courbe 5

5.6 Déterminer les plus courts facteurs uniques en construisant la table LgCandidat

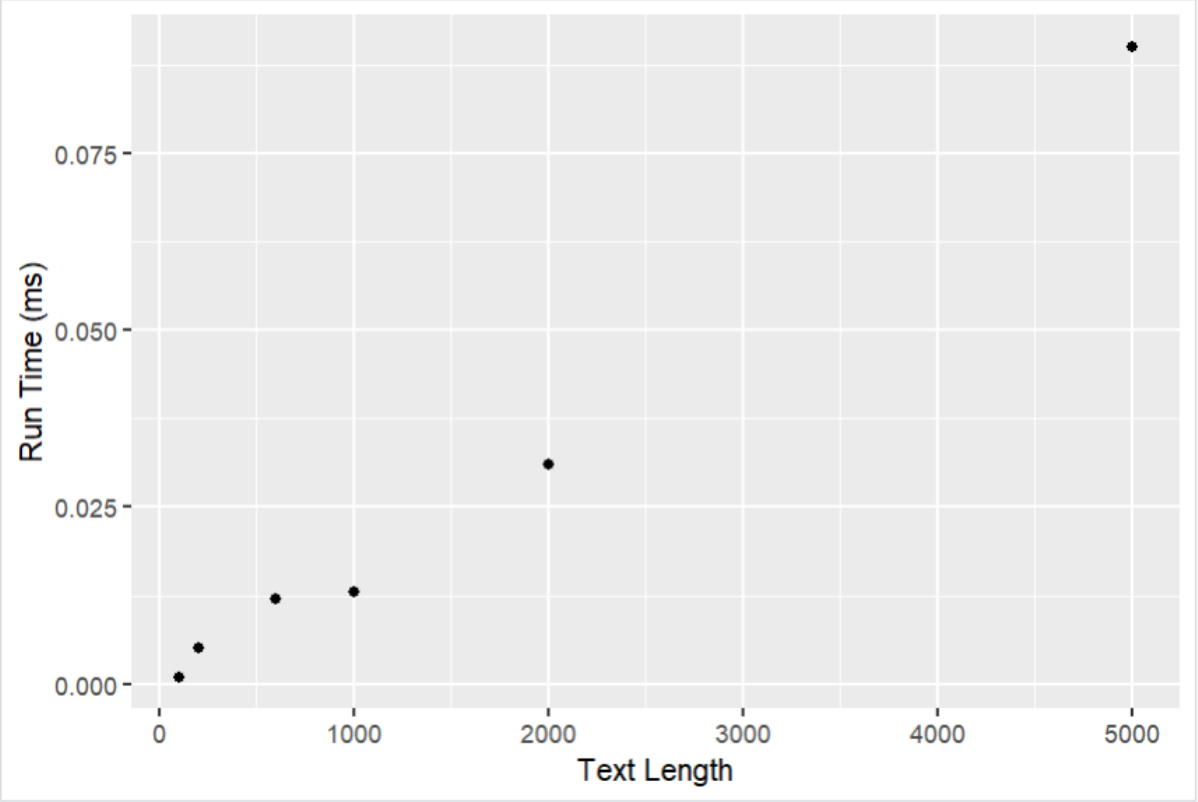

Figure 5.7: Courbe 6

5.7 Les répétitions super-maximales

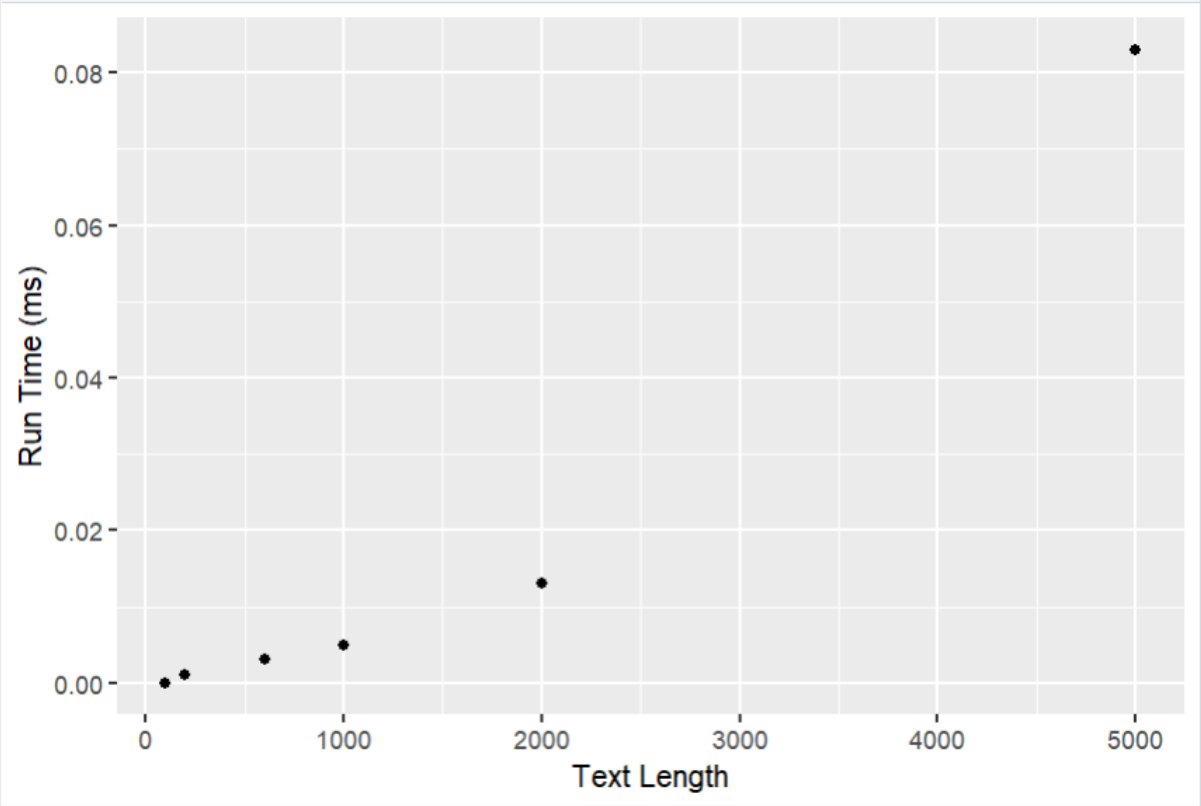

Figure 5.8: Courbe 7

5.8 Le plus long facteur commun entre deux textes T1 et T2

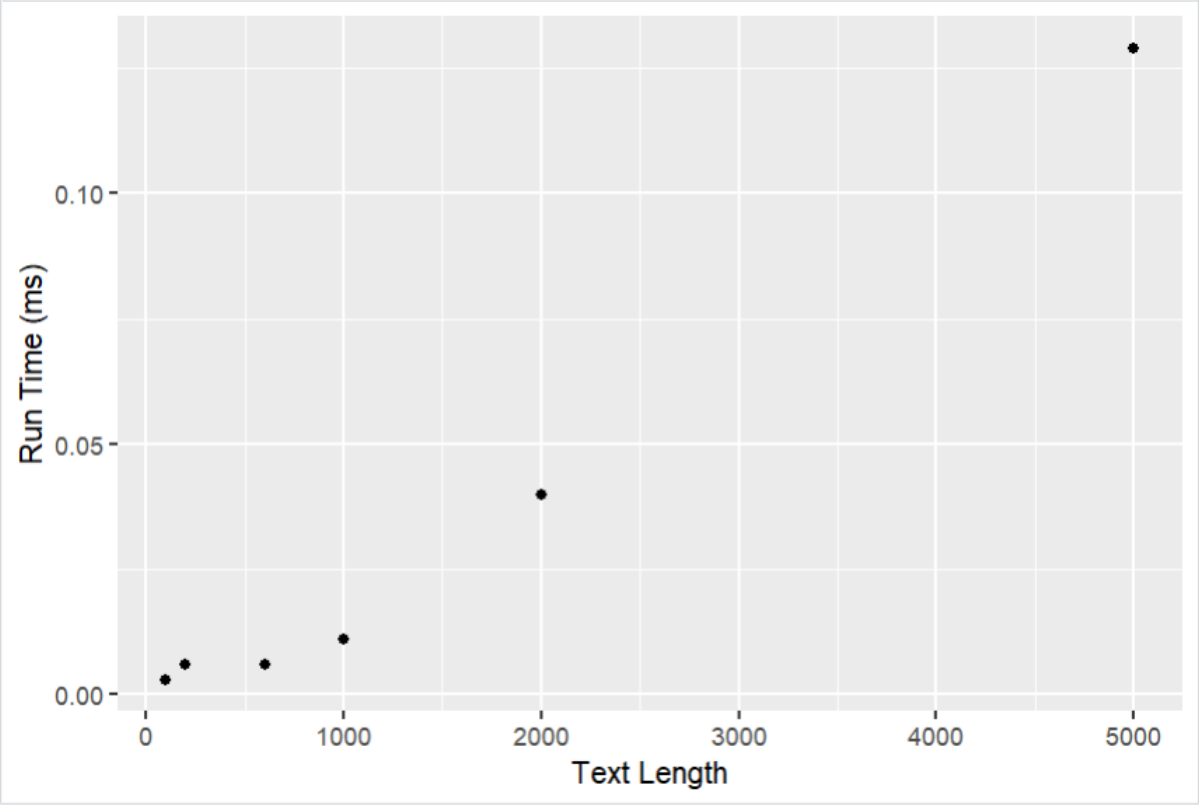

Figure 5.9: Courbe 8

Chapter 6

Analyse des résultats et conclusions

6.1 Analyses et conclusions

- La complexité temporelle augmente avec l'augmentation de la taille du texte dans touts les traitements.
- La complexité temporelle représentée par les graphes évolue d'une manière logarithmique.

6.2 Complexités temporelles théoriques pour quelques traitements

Traitement	complexité temporelle
Construire la table des suffixes TS	$O(n^2 * logn)$
La recherche d'un motif M	$O(m * logn)$
Construire la table HTR	$O(n)$
Le plus long facteur répété	$O(n * logn)$

Les complexités temporelles expérimentales des algorithmes (qui sont représentés par les courbes précédentes) sont presque en accord avec les complexités temporelles théoriques.

Chapter 7

Le code source

Le code source est dans un fichier word à part intitulé : "suffix_array".