Faculté informatique

Master 1 BioInformatique
Rapport Projet

Projet BioAlgo : Table des suffixes

Etudiants :
MOUALHI Sarah
181831031232
RABHI Oussama Ziad
181831066142

Enseignant :
Mme. BOUKHEDOUMA

Table des matières

Table des figures

Chapitre 1

Introduction :

Comme on a déjà vu dans les chapitres de ce module,il y existe des algorithmes de recherche de motifs qui sont utilisés pour trouver un motif ou une sous-chaîne à partir d'une autre chaîne plus grande (texte) dont l'objectif principal de la conception de ce type d'algorithmes est de réduire la complexité temporelle.
Souvent , dans le domaine de l'informatique et en algorithmique, la recherche efficace de motifs et de répétitions dans un texte est une problématique fondamentale et fréquente .Donc ,il est essentiel de disposer d'outils performants capables d'identifier rapidement et précisément et d'une façon efficace les occurrences d'un motif donné .L'un de ses outils est l'utilisation des tables de suffixes qui sont une structure de données cruciale utilisé pour accélérer la recherche de motifs. Les tables de suffixes sont des représentations compactes d'un texte qui permettent de stocker et d'interroger efficacement les positions des différents suffixes d'un texte. Ces structures d'indexation offrent des possibilités remarquables pour l'analyse des motifs et des répétitions, en permettant une recherche en temps quasi linéaire.

Dans ce rapport, nous allons nous mettre en œuvre une structure d'index basée sur les tables de suffixes afin d'accélérer la recherche de motifs et de répétitions dans un texte donné. Nous nous intéresserons tout particulièrement à l'analyse de la complexité spatiale des structures de données utilisées , ainsi qu'à l'évaluation de la complexité temporelle des traitements en comparant les performances en termes de temps d'exécution et d'espace mémoire requis pour différentes tailles de texte .

Chapitre 2

Description des structures de données utilisées :

2.1 Table des suffixes :

La table des suffixes (Suffix Array) pour S est une structure de données construit en temps linéaire et contenant tous les suffixes de S en ordre lexicographique .
Chaque suffixe est associé à sa position dans le texte d'origine. La construction de la table des suffixes permet d'accélérer la recherche de motifs et de répétitions dans le texte.
L'avantage des tables des suffixes est qu'il offrent une plus grande efficacité en espace.
La complexité spatiale de la table des suffixes est généralement de l'ordre de $O(n^2)$, où n est la taille du texte.

2.2 Table HTR (Table préfixes communs) :

C'est une structure auxiliaire de la table des suffixes , utilisée dans le traitement des chaînes de caractères. Elle stocke les longueurs des préfixes communs les plus longs entre tous les suffixes consécutifs triés d'un mot. En d'autres termes, il s'agit de la longueur du plus long préfixe communs entre chaque paire de suffixes consécutifs d'un tableau de suffixes triés.

Cette table est utilisée pour identifier les facteurs répétés dans le texte. La complexité spatiale de la table HTR est de l'ordre de O(n), où n est la taille du texte.

2.3 Inverse de la table des suffixes (ITS) :

L'inverse d'un tableau consiste à créer un nouveau tableau dans lequel les valeurs du tableau d'origine deviennent les clés, et les clés les valeurs. En d'autres termes, il s'agit d'intervertir les positions des éléments dans le tableau.
L'inverse de la table des suffixes est tout simplement une structure qui inverse l'ordre des

indices dans la table des suffixes. Cela signifie que chaque suffixe est associé à son indice de fin dans le texte d'origine. L'inverse de la table des suffixes est utilisé dans certaines opérations de recherche et d'analyse de motifs.
La complexité spatiale de l'inverse de la table des suffixes est également de l'ordre de O(n).

2.4 Table des longueurs de candidats (LgCandidat) :

C'est une structure utilisée pour identifier les plus courts facteurs uniques présents dans le texte.

Un plus court facteur unique est défini comme un facteur, c'est-à-dire une sous-chaîne consécutive, qui apparaît seulement une fois dans le texte en sorte que tous les sous-facteurs plus petits de ce facteur unique doivent être des répétitions, cad qu'ils doivent apparaître plusieurs fois dans le texte.

La table des longueurs de candidats est une structure qui stocke les longueurs de tous les facteurs candidats du texte.D'ou on peut déterminer les longueurs des plus courts facteurs uniques présents dans le texte.

La complexité spatiale de la table des longueurs de candidats dépend de la taille du texte et du nombre de facteurs uniques présents. Dans le pire des cas, si chaque caractère du texte est un facteur unique, la complexité spatiale peut être de l'ordre de O(n), où n est la taille du texte.Sinon, dans la plupart des cas, la complexité spatiale est inférieure à cela, car de nombreux facteurs seront des répétitions et n'auront donc pas besoin d'être stockés dans la table des longueurs de candidats.

2.5 Pseudo-algorithmes :

Dans cette partie on va citer les étapes de conception de chaque traitement en format de pseudo algorithmes :

2.5.1 Construction de la table des suffixes et l'afficher :

```
Function Creer_TS(texte):
    n = taille(texte)
    suffixes_list = ListeVide() \\ créer une liste vide suffixes
    suffixes_indexes = ListeVide() \\ liste des indices des suffixes

    Pour i allant de 0 à n-1:
        Ajouter(texte[i:n]) à la liste suffixes list
        Ajouter(i) a la liste des indexes

    list = Trier la liste suffixes par ordre lexicographique
    retourner list_suff , indexes \\ la liste suffixes

\\Affichage
tableSuffixes = construireTableSuffixes(texte)
Afficher(tableSuffixes)
```

2.5.2 Recherche d'un motif M dans le texte à l'aide de la table des suffixes TS :

```
Function Search_pattern(texte,motif):
    n = taille(texte)
    suffixes = Creer_TS(texte)
    indices = ListeVide() \\ liste des indices d apparition du motif

      Pour chaque suffixe dans suffixes:

          Si le motif est préfixe du suffixe:

            retourner l'indice du suffixe
    retourner -1 (si le motif n'est pas trouvé)

\\Affichage

indexes = rechercheMotif(texte, motif)
Afficher(indexes)
```

2.5.3 Construction de la table HTR de T :

```
Fonction HTR(text):
    suffixes = Creer_TS(text)
    htr_table = ListeVide()
    n = taille(suffixes)
    for i in range(1,n):
        suffix1 = suffixes[i]
        suffix2 = suffixes[i-1]
        prefix = PrefixeCommun(suffix1,suffix2)

        Ajouter Prefix à la table htr
    retourner htr_table
```

2.5.4 Le(s) plus long(s) facteur(s) répété(s) dans le texte :

```
Function Repeated_factors(texte):
    tableHTR = HTR(texte)
    tablesuffixes = Creer_TS(texte)
    factors_rep = ListeVide()
    longest_rep_factors = ListeVide()
    Pour chaque élément (suffixe1, suffixe2, longprefix_comm)
    dans tableHTR:
        Si taille(longprefix_comm) > 0:
            facteur = extraireFacteur(suffixe1, longprefix_comm)
            Si facteur appartient a factors_rep :
                Ajouter(facteur) à longest_rep_factors
            sinon:
                ajouter(facteur) à factors_rep
    retourner longest_rep_factors

\Affichage

plusLongsFacteurs = Repeated(texte)
afficher(plusLongsFacteurs)
```

2.5.5 les facteurs qui se répètent au moins 3 fois :

```
Function Repeated_Atleast_3(texte):
    tableHTR = HTR(texte)
    tablesuffixes = Creer_TS(texte)
    facteursRepetes = ListeVide()
    Pour chaque élément (suffixe1, suffixe2, longprefixcomm)
    dans tableHTR:
```

```
        Si longueurCommun > 0 :
            facteur = ExtraireFacteur(suffixe1 , longprefixcomm)
            occurrences = CompterOccurrences(facteur , tableSuffixes)
            Si occurrences >= 3 et facteur n'est pas déjà dans
            facteursRepetes :
                Ajouter facteur à facteursRepetes
    retourner facteursRepetes

\\Affichage

resultat = Repeated_Atleast_3(texte)
Afficher(resultat)
```

2.5.6 Construction de l'inverse ITS de la table des suffixes de T et l'afficher :

```
Function InverseTS(texte):
    indexes_Suffixes = Creer_TS(text)
    n = taille(tableSuffixes)
    inverse = ListeVide()
    Pour i  de 0 à n:
        index = indexes_suffixes[i]
        inverse[index] = i

    retourner inverse

\\Affichage

inverseTableSuffixes = InverseTS(texte)
Afficher(inverseTableSuffixes)
```

2.5.7 La table LgCandidat des plus courts facteurs uniques du texte :

```
Function LgCandidat(texte):
   tablesuffixes = Creer_TS(text)
   LgCandidat = ListeVide()

    Pour chaque facteur dans tableSuffixes :
        estFacteurUnique = EstFacteurUnique(facteur , tableSuffixes)
        Si estFacteurUnique :
            ajouter facteur à LgCandidat
    retourner LgCandidat
```

```
plusCourtsFacteurs = LgCandidat(texte)

afficher(plusCourtsFacteurs)
```

2.5.8 Détermination des répétitions super-maximales du texte :

```
Function RSuperMaximales(texte):
    tableSuffixes = Creer_TS(texte)
    tableHTR = HTR(texte)
    repetitionsSM = ListeVide()
    Pour chaque élément (suffixe1, suffixe2,longprefixcomm)
    dans tableHTR:
        Si longprefixcomm > 0:
          Si aucun autre élément dans tableHTR
          a un préfixe commun plus long:
                Ajouter(longpreficomm) à repetitionsSM

    retourner repetitionsSM

repetitionsSuperMaximales = RSuperMaximales(texte)
Afficher(repetitionsSuperMaximales)
```

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

```
Function PlusLongFacteurC(T1, T2):

    tableSuffixesT1 = construireTableSuffixes(T1)
    tableSuffixesT2 = construireTableSuffixes(T2)
    plusLongFacteurCommun = ""

    Pour chaque suffixeT1 dans tableSuffixesT1:
        Pour chaque suffixeT2 dans tableSuffixesT2:
            Si suffixeT1 et suffixeT2 ont un préfixe commun plus long:
                facteurCommun = extraireFacteur(suffixeT1, longprefixcom)
                Si taille(facteurCommun) > taille(plusLongFacteurCommun):
                    plusLongFacteurCommun = facteurCommun

    retourner plusLongFacteurCommun

T1 = Texte()
T2 = Texte()
plusLongFacteurCommun = PlusLongFacteur(T1, T2)
Afficher(plusLongFacteurCommun)
```

Chapitre 3

Les tableaux des tests et courbes :

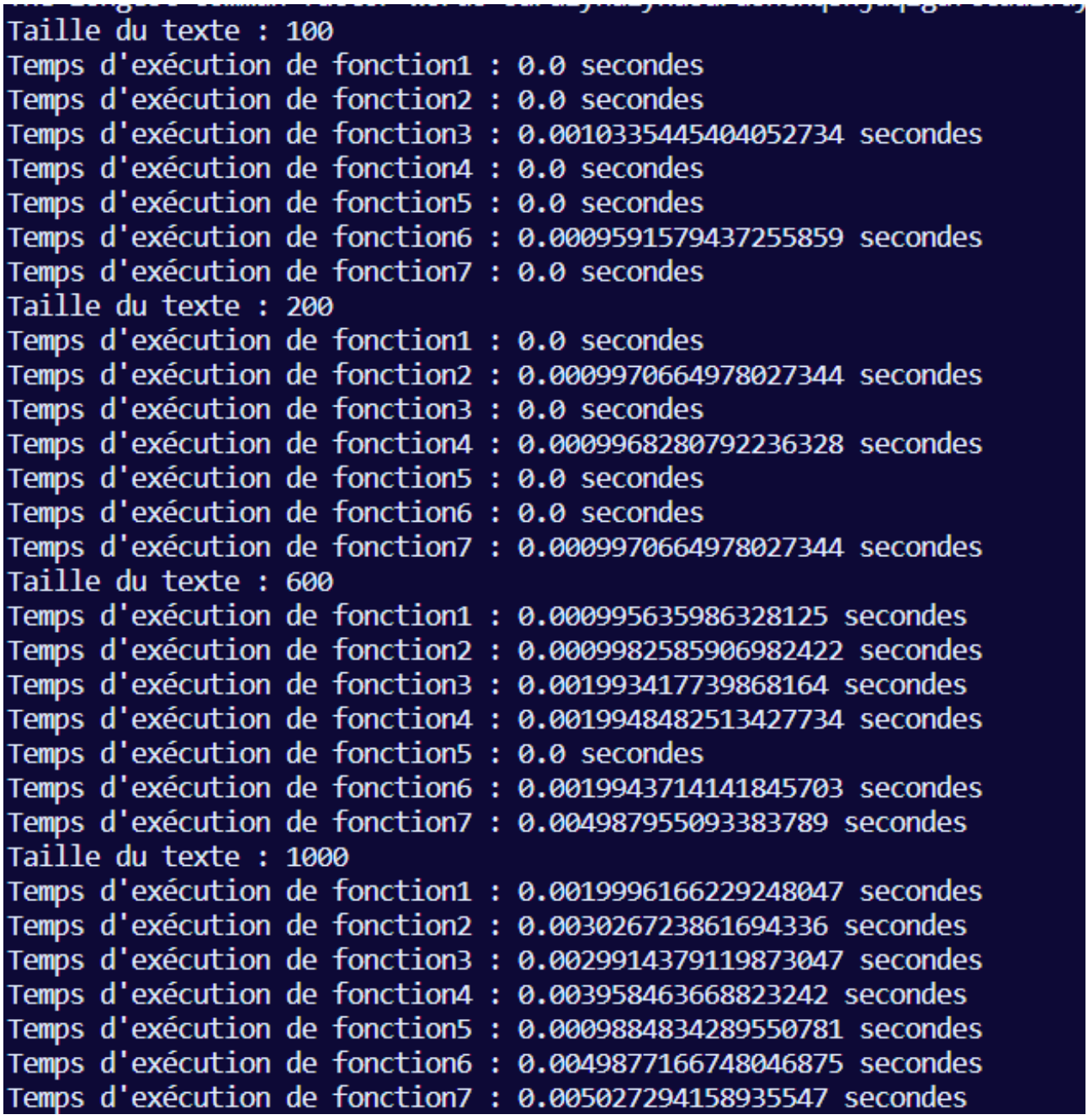


Fig. 3.1 : Temps d'éxécution

```
Temps d'exécution de fonction7 : 0.005027294158935547 secondes
Taille du texte : 2000
Temps d'exécution de fonction1 : 0.00502467155456543 secondes
Temps d'exécution de fonction2 : 0.006981611251831055 secondes
Temps d'exécution de fonction3 : 0.01197361946105957 secondes
Temps d'exécution de fonction4 : 0.010940074920654297 secondes
Temps d'exécution de fonction5 : 0.005011320114135742 secondes
Temps d'exécution de fonction6 : 0.01496577262878418 secondes
Temps d'exécution de fonction7 : 0.017948627471923828 secondes
Taille du texte : 5000
Temps d'exécution de fonction1 : 0.0329127311706543 secondes
Temps d'exécution de fonction2 : 0.043921470642089844 secondes
Temps d'exécution de fonction3 : 0.05086207389831543 secondes
Temps d'exécution de fonction4 : 0.06478714942932129 secondes
Temps d'exécution de fonction5 : 0.022938251495361328 secondes
Temps d'exécution de fonction6 : 0.07779455184936523 secondes
Temps d'exécution de fonction7 : 0.10475897789001465 secondes
```

FIG. 3.2 : Temps d'exécution

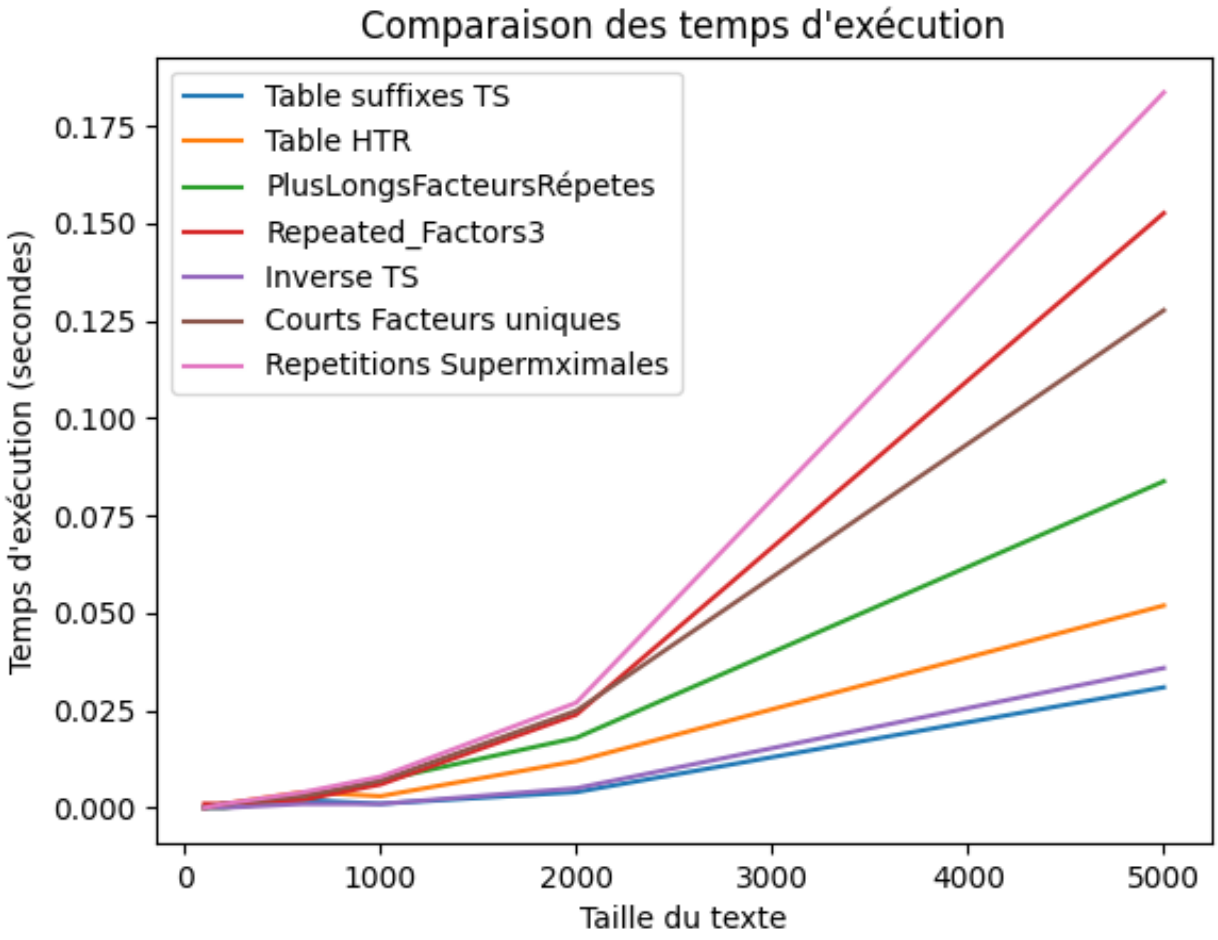
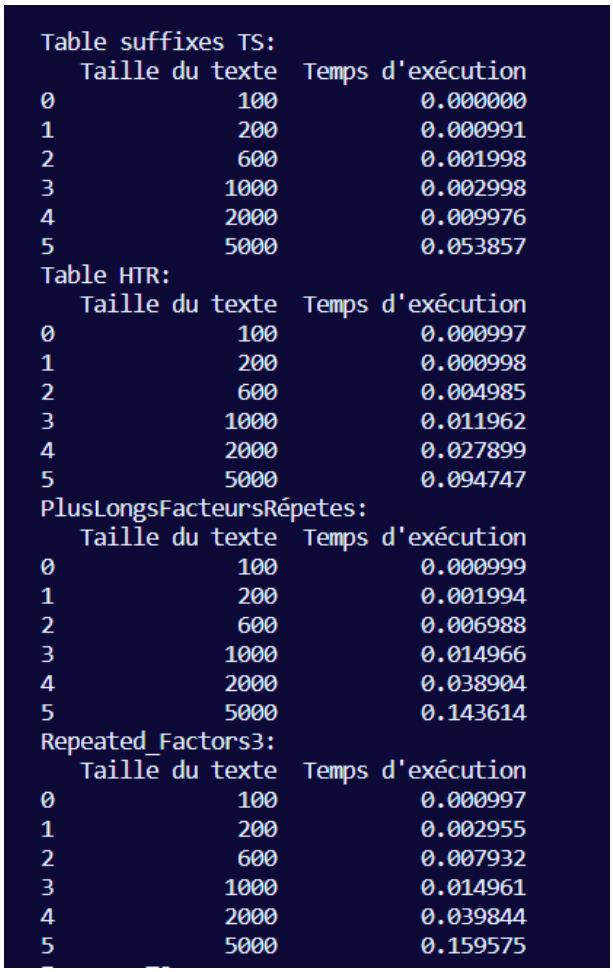
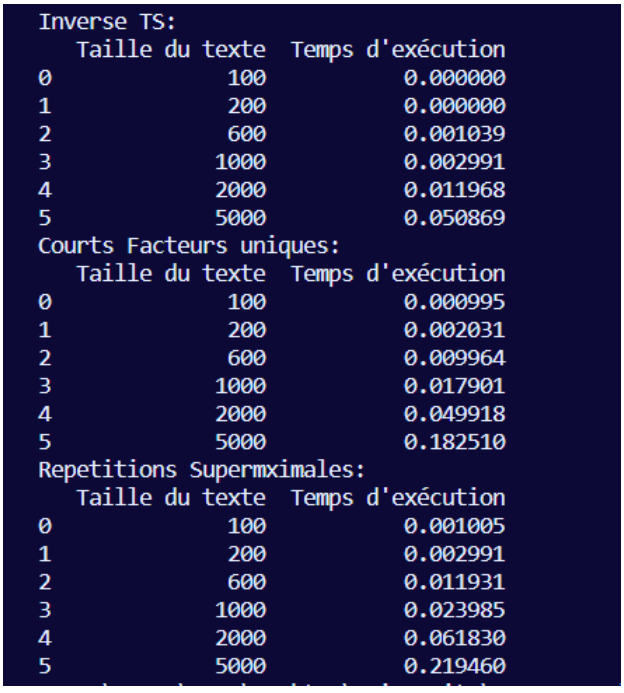


FIG. 3.3 : Courbe de comparaison



```
Table suffixes TS:
   Taille du texte  Temps d'exécution
0              100           0.000000
1              200           0.000991
2              600           0.001998
3             1000           0.002998
4             2000           0.009976
5             5000           0.053857
Table HTR:
   Taille du texte  Temps d'exécution
0              100           0.000997
1              200           0.000998
2              600           0.004985
3             1000           0.011962
4             2000           0.027899
5             5000           0.094747
PlusLongsFacteursRépetes:
   Taille du texte  Temps d'exécution
0              100           0.000999
1              200           0.001994
2              600           0.006988
3             1000           0.014966
4             2000           0.038904
5             5000           0.143614
Repeated_Factors3:
   Taille du texte  Temps d'exécution
0              100           0.000997
1              200           0.002955
2              600           0.007932
3             1000           0.014961
4             2000           0.039844
5             5000           0.159575
```

FIG. 3.4 : Les resultats de temps en secondes de chaque fonction



```
Inverse TS:
   Taille du texte  Temps d'exécution
0              100           0.000000
1              200           0.000000
2              600           0.001039
3             1000           0.002991
4             2000           0.011968
5             5000           0.050869
Courts Facteurs uniques:
   Taille du texte  Temps d'exécution
0              100           0.000995
1              200           0.002031
2              600           0.009964
3             1000           0.017901
4             2000           0.049918
5             5000           0.182510
Repetitions Supermximales:
   Taille du texte  Temps d'exécution
0              100           0.001005
1              200           0.002991
2              600           0.011931
3             1000           0.023985
4             2000           0.061830
5             5000           0.219460
```

FIG. 3.5 : Les resultats de temps en secondes de chaque fonction

3.1 Exemples d'éxécution :

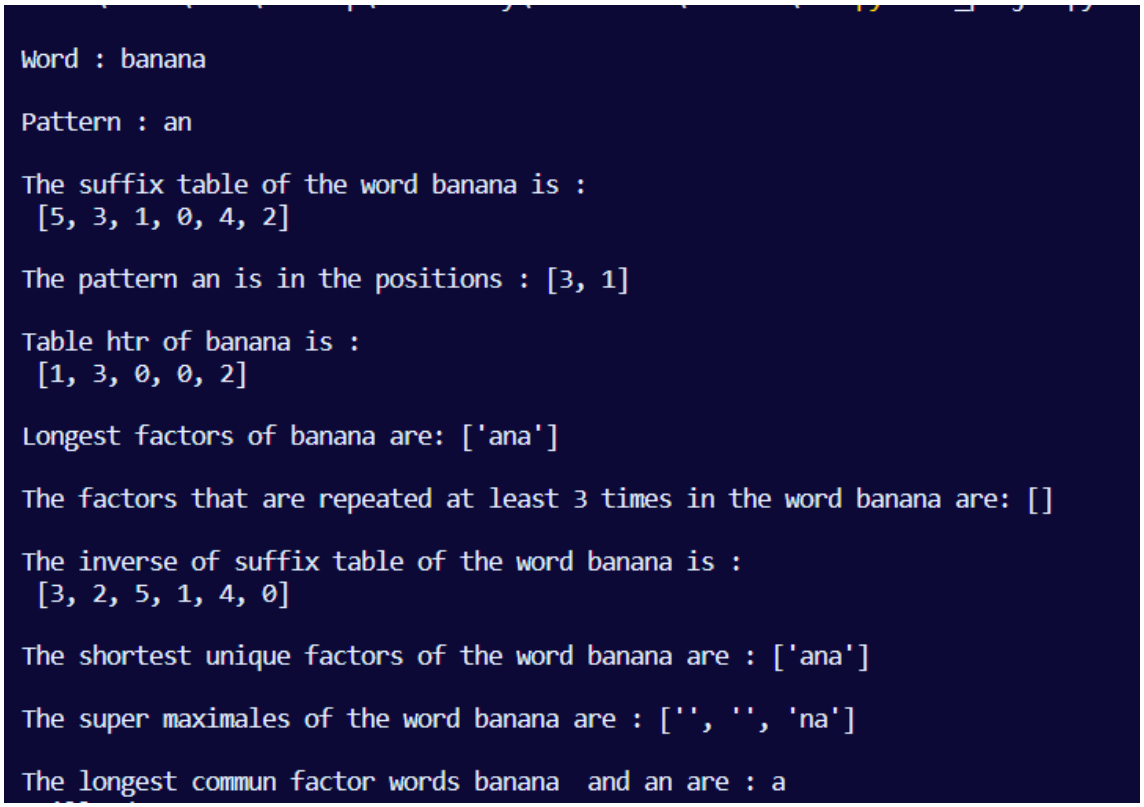


FIG. 3.6 : Exemple 1mot = banana

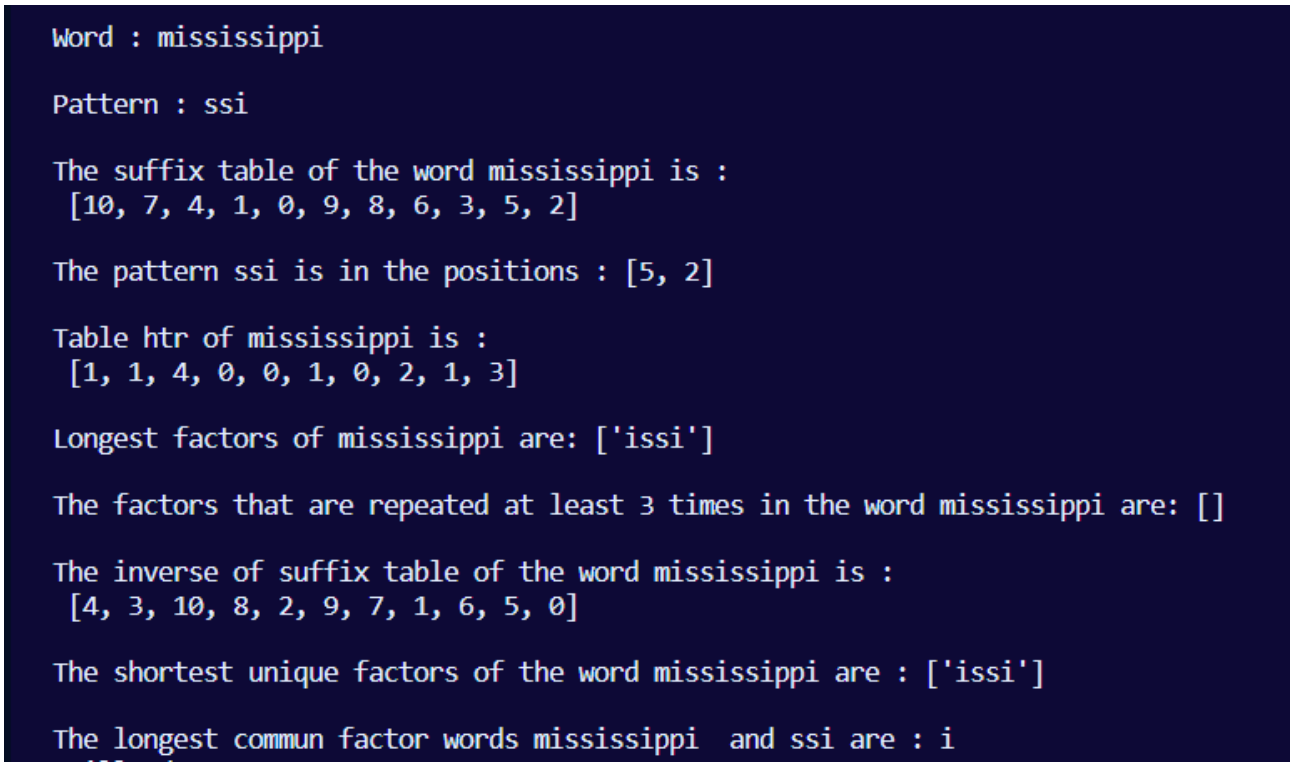


FIG. 3.7 : Exemple mot = mississippi

```
Word : sazralynassaralynao

Pattern : lyna

The suffix table of the word sazralynassaralynao is :
 [13, 4, 17, 11, 8, 1, 14, 5, 16, 7, 18, 12, 3, 10, 0, 9, 15, 6, 2]

The pattern lyna is in the positions : [14, 5]

Table htr of sazralynassaralynao is :
 [5, 1, 1, 1, 1, 0, 4, 0, 2, 0, 0, 6, 0, 2, 1, 0, 3, 0]

Longest factors of sazralynassaralynao are: ['ralyna']

The factors that are repeated at least 3 times in the word sazralynassaralynao are: []

The inverse of suffix table of the word sazralynassaralynao is :
 [14, 5, 18, 12, 1, 7, 17, 9, 4, 15, 13, 3, 11, 0, 6, 16, 8, 2, 10]

The shortest unique factors of the word sazralynassaralynao are : []

The longest commun factor words sazralynassaralynao  and lyna are : alyna
```

FIG. 3.8 : Exemple 3

```
Word : ACABBCASGHSBACSNXCBAC

Pattern : AC

The suffix table of the word ACABBCASGHSBACSNXCBAC is :
 [2, 19, 0, 12, 6, 18, 11, 3, 4, 20, 1, 5, 17, 13, 8, 9, 15, 10, 7, 14, 16]

The pattern AC is in the positions : [19, 0, 12]

Table htr of ACABBCASGHSBACSNXCBAC is :
 [1, 2, 2, 1, 0, 3, 1, 1, 0, 1, 2, 1, 1, 0, 0, 0, 0, 1, 1, 0]

Longest factors of ACABBCASGHSBACSNXCBAC are: ['BAC']

The factors that are repeated at least 3 times in the word ACABBCASGHSBACSNXCBAC are: []

The inverse of suffix table of the word ACABBCASGHSBACSNXCBAC is :
 [2, 10, 0, 7, 8, 11, 4, 18, 14, 15, 17, 6, 3, 13, 19, 16, 20, 12, 5, 1, 9]

The shortest unique factors of the word ACABBCASGHSBACSNXCBAC are : []

The super maximales of the word ACABBCASGHSBACSNXCBAC are : ['A', '', 'B', 'B', '', 'C', 'C', 'C', '', '', '', '', 'S', 'S', '']

The longest commun factor words ACABBCASGHSBACSNXCBAC  and AC are : ABBC
Taille du texte : 100
```

FIG. 3.9 : Exemple 4 :

Chapitre 4

Conclusion :

En conclusion , Le projet avait pour objectif d'explorer et d'évaluer plusieurs fonctions clés pour l'analyse de chaînes de caractères . Ces fonctions incluaient la construction de la table de suffixes, la recherche de motifs, le calcul de la table HTR, ainsi que la recherche de facteurs répétés.Et à la fin ,une étude comparative de ces dernières pour des textes de tailles différentes .

D'après l'analyse comparatif depuis la courbe de comparaison,on peut constater des variations remarquables dans les performances des fonctions.A chaque fois la taille du texte est plus grande ,les fonctions prennent plus de temps d'exécution ce qui signifie que les temps d'exécution varie en fonction de la taille du texte. Certaines fonctions ont démontré une performances relativement constante (une stabilité)avec des temps d'exécution similaires indépendamment de la taille du texte , comme la construction de la table de suffixes et la recherche de motifs. Tandis que, d'autres fonctions ont montré une tendance à augmenter les temps d'exécution avec des textes plus longs : comme la recherche de facteurs répétés et la recherche de répétitions supermaximales, des temps d'exécution plus élevés pour les textes plus longs.

Enfin ,Ces résultats soulignent l'importance de la taille du texte lors de l'utilisation de ces fonctions et d'en prendre en comptes les implifications en termes de temps d'exécution et d'efficacité algorithmique pour chaque fonction, et en particulier textes volumineux (grandes tailles).