SORBONNE UNIVERSITÉ

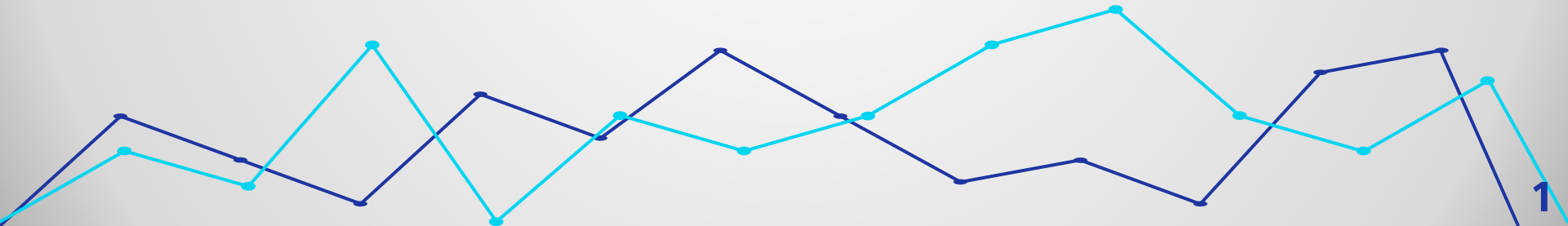
2022/2023
M1 STL

Projet UE Ouverture

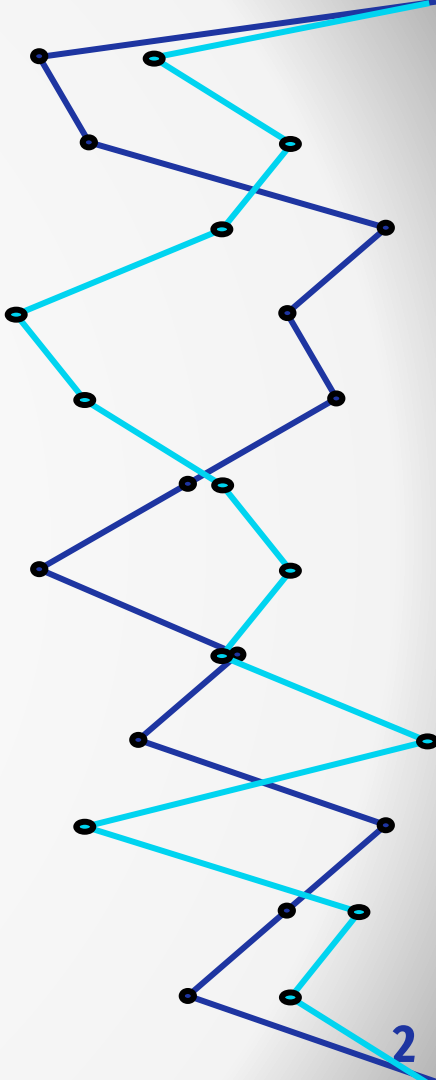
-

Détection de plagiat flagrant

Par Daniel SIMA et Amaury CURIEL

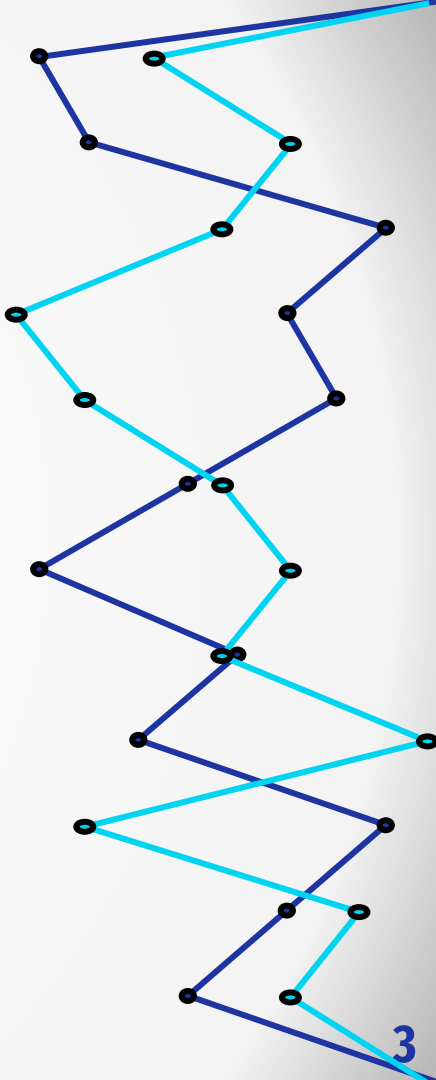


Sommaire



I. Présentation du projet

- Objectifs ?
- Outils utilisés ?
- Méthode de travail ?
- Changements de notation ?
  - Racine = “§”
  - Symbole de fin de chaîne “#” = “_”



II. Programmation dynamique - Exemple pour ANANAS BANANE

	A	N	A	N	A	S
B	0	0	0	0	0	0
A	1	0	1	0	1	0
N	0	2	0	2	0	0
A	1	0	3	0	3	0
N	0	2	0	4	0	0
E	0	0	0	0	0	0

II. Programmation dynamique - Comment récupérer la sous chaîne commune?

- En stockant l'indice maximal et en remontant la diagonale à ces indices.

	A	N	A	N	A	S
B	0	0	0	0	0	0
A	1	0	1	0	1	0
N	0	2	0	2	0	0
A	1	0	3	0	3	0
N	0	2	0	4	0	0
E	0	0	0	0	0	0

INDICE MAXIMAL

II. Programmation dynamique - Comment récupérer la sous chaîne commune?

- En stockant l'indice maximal et en remontant la diagonale à ces indices.

	A	N	A	N	A	S
B	0	0	0	0	0	0
A	**1**	0	1	0	1	0
N	0	**2**	0	2	0	0
A	1	0	**3**	0	3	0
N	0	2	0	**4**	0	0
E	0	0	0	0	0	0

INDICE MAXIMAL

II. Programmation dynamique - Comment récupérer la sous chaîne commune?

- En stockant l'indice maximal et en remontant la diagonale à ces indices.

	A	**N**	**A**	**N**	A	S
B	0	0	0	0	0	0
A	**1**	0	1	0	1	0
N	0	**2**	0	2	0	0
A	1	0	**3**	0	3	0
N	0	2	0	**4**	0	0
E	0	0	0	0	0	0

INDICE MAXIMAL

III. Arbre des suffixes non compressé - Construction

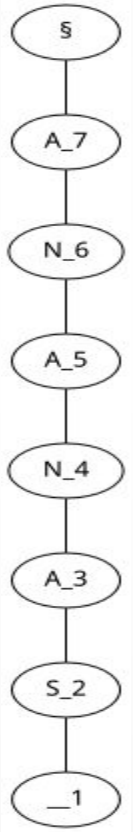
Pseudo-code:

ArbreSuffixe(c,a):

Pour chaque suffixe de c:
    vérifier si il existe un noeud dans a qui correspond à la
    première lettre du suffixe en cours:
        Si oui: appeler récursivement ArbreSuffixe sur c[:1] et le noeud qu'on a
            trouver
        Si non: ajouterFrère(c) à la racine de a.

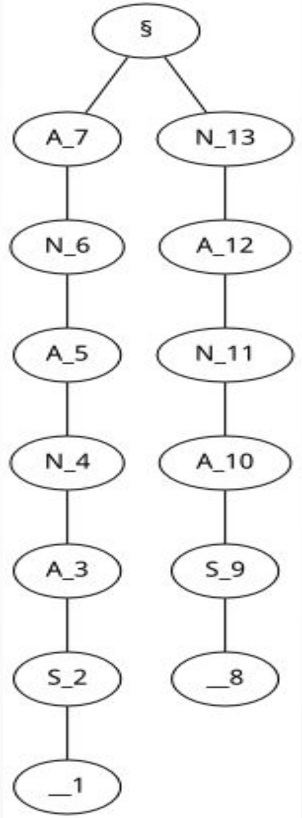
III. Arbre des suffixes non compressé - Exemple pour ANANAS_

ANANAS_



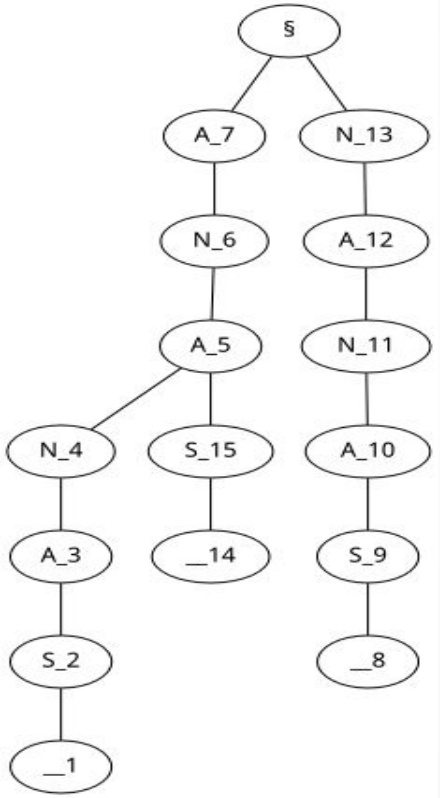
III. Arbre des suffixes non compressé - Exemple pour ANANAS_

ANANAS_



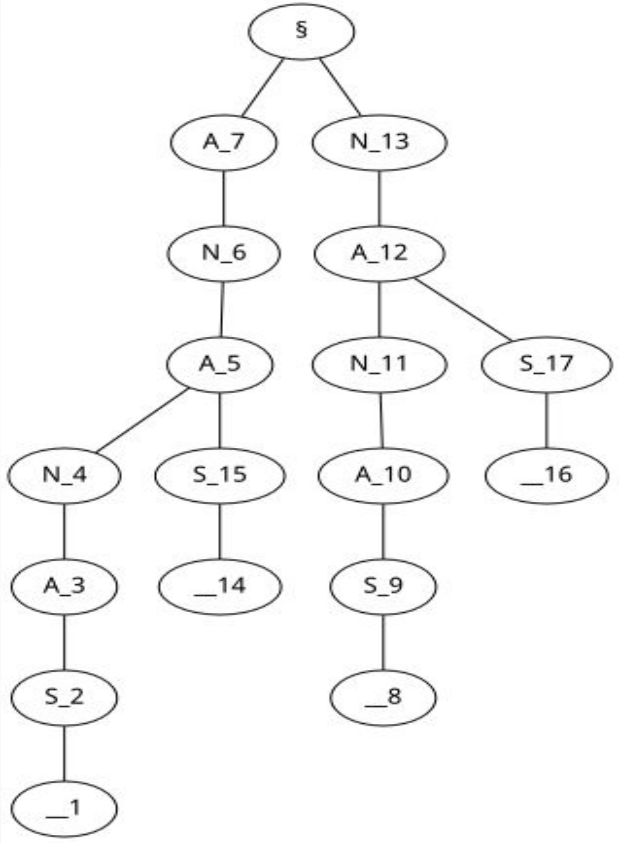
III. Arbre des suffixes non compressé - Exemple pour ANANAS_

ANANAS_



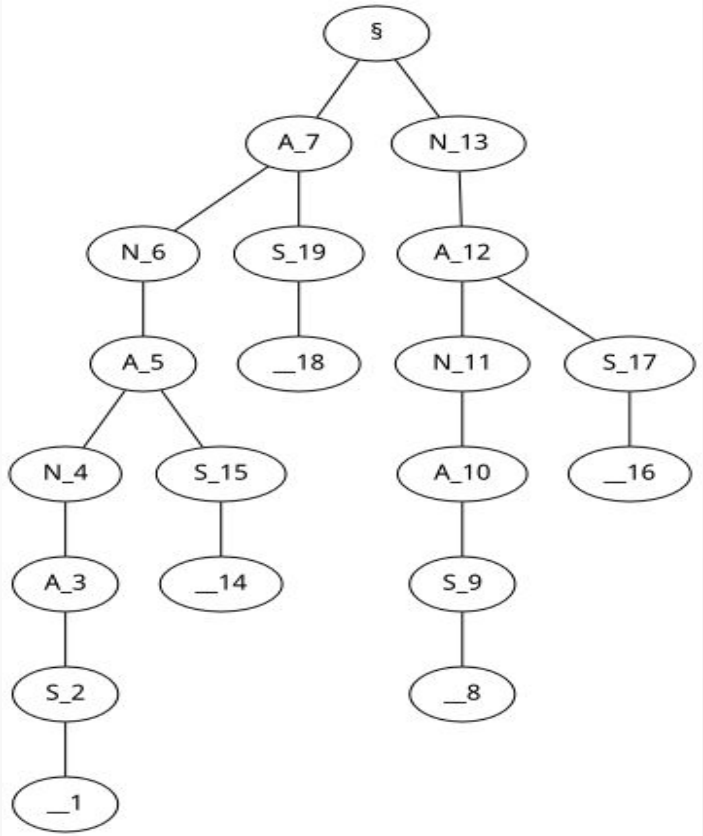
III. Arbre des suffixes non compressé - Exemple pour ANANAS_

ANANAS_



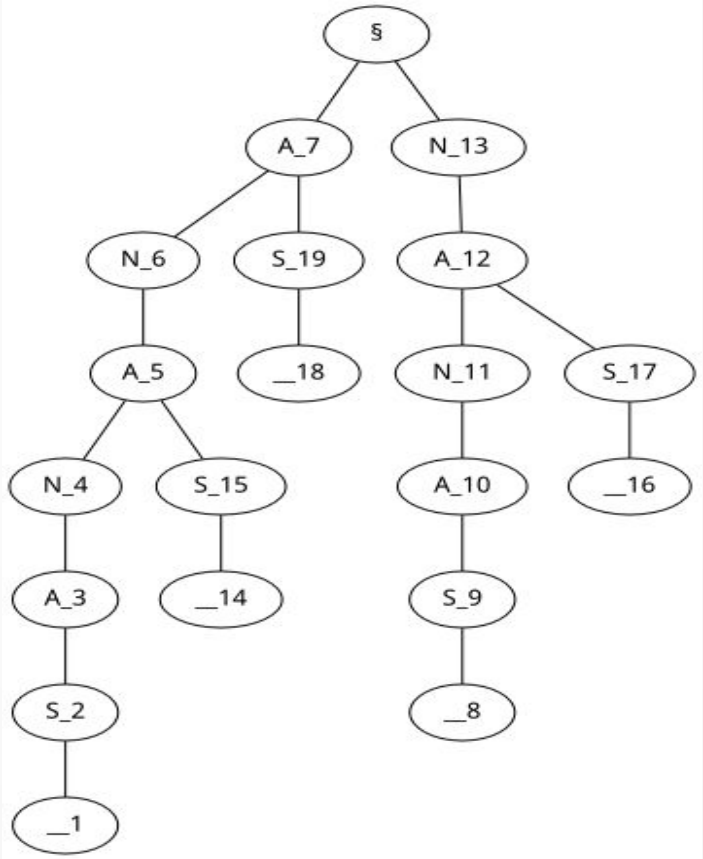
III. Arbre des suffixes non compressé - Exemple pour ANANAS_

ANANAS_



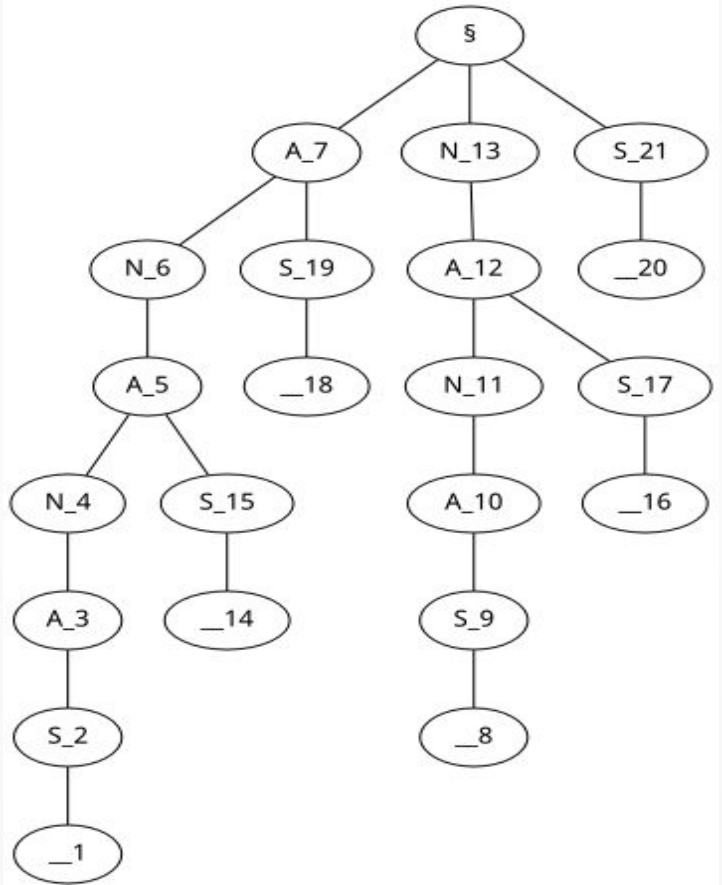
III. Arbre des suffixes non compressé - Exemple pour ANANAS_

ANANAS_



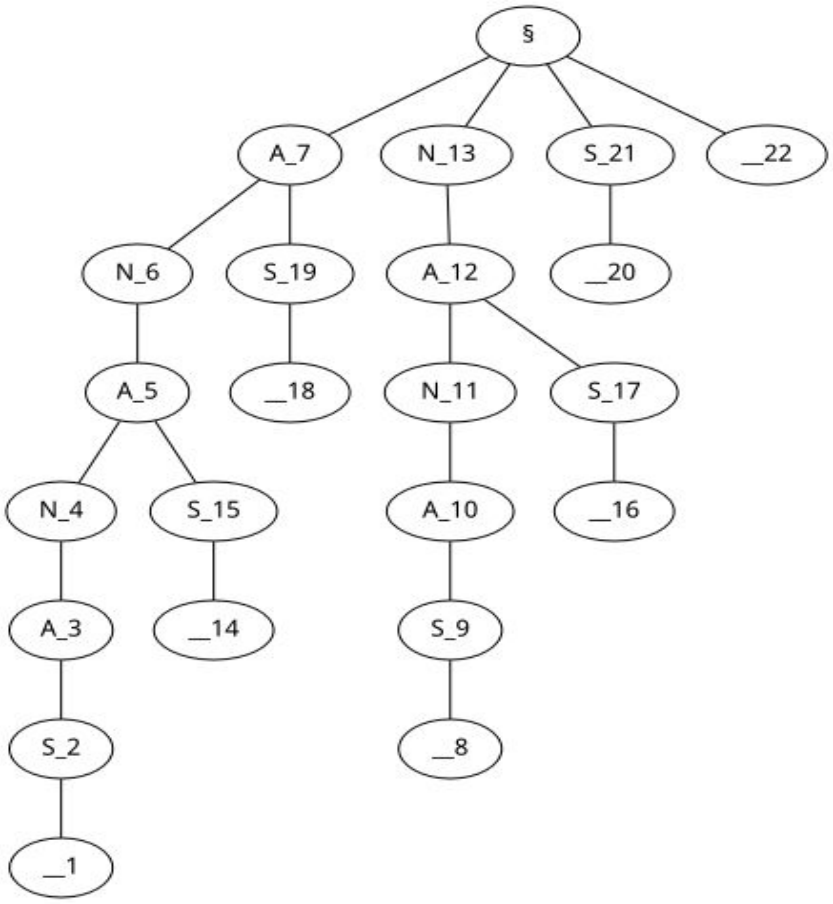
III. Arbre des suffixes non compressé - Exemple pour ANANAS_

ANANAS_



III. Arbre des suffixes non compressé - Exemple pour ANANAS_

ANANAS_



III. Arbre des suffixes non compressé - Récupération de la sous chaîne commune.

- Définition d'une nouvelle structure de donnée RichArbre

  Un noeud comporte maintenant:

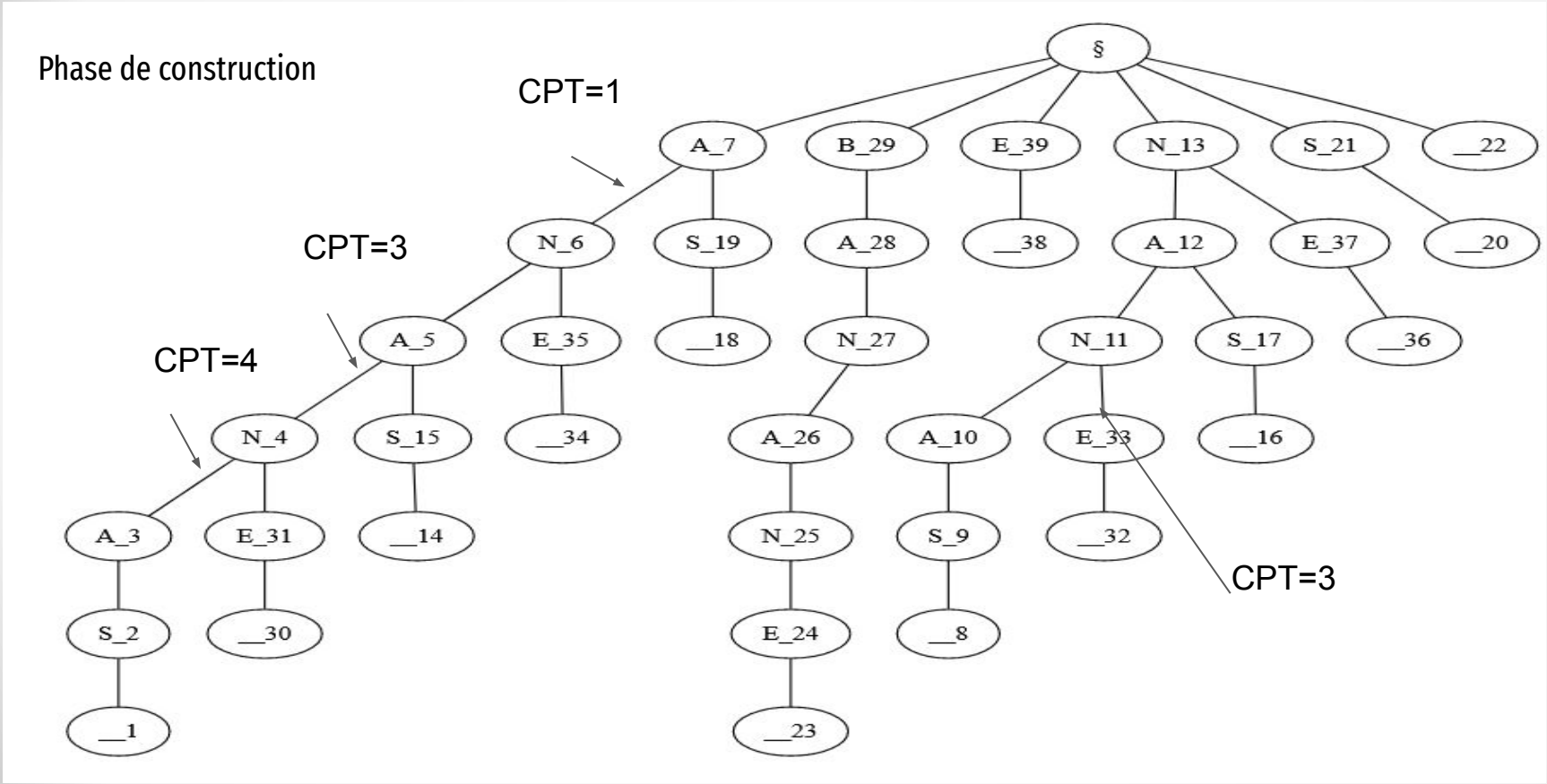
  - Un label
  - Une liste de fils
  - Un compteur
  - Un pointeur vers son père

  Définition plus tard dans le projet un type IDarbre qui ajoute un identifiant afin de séparer le label de l'id

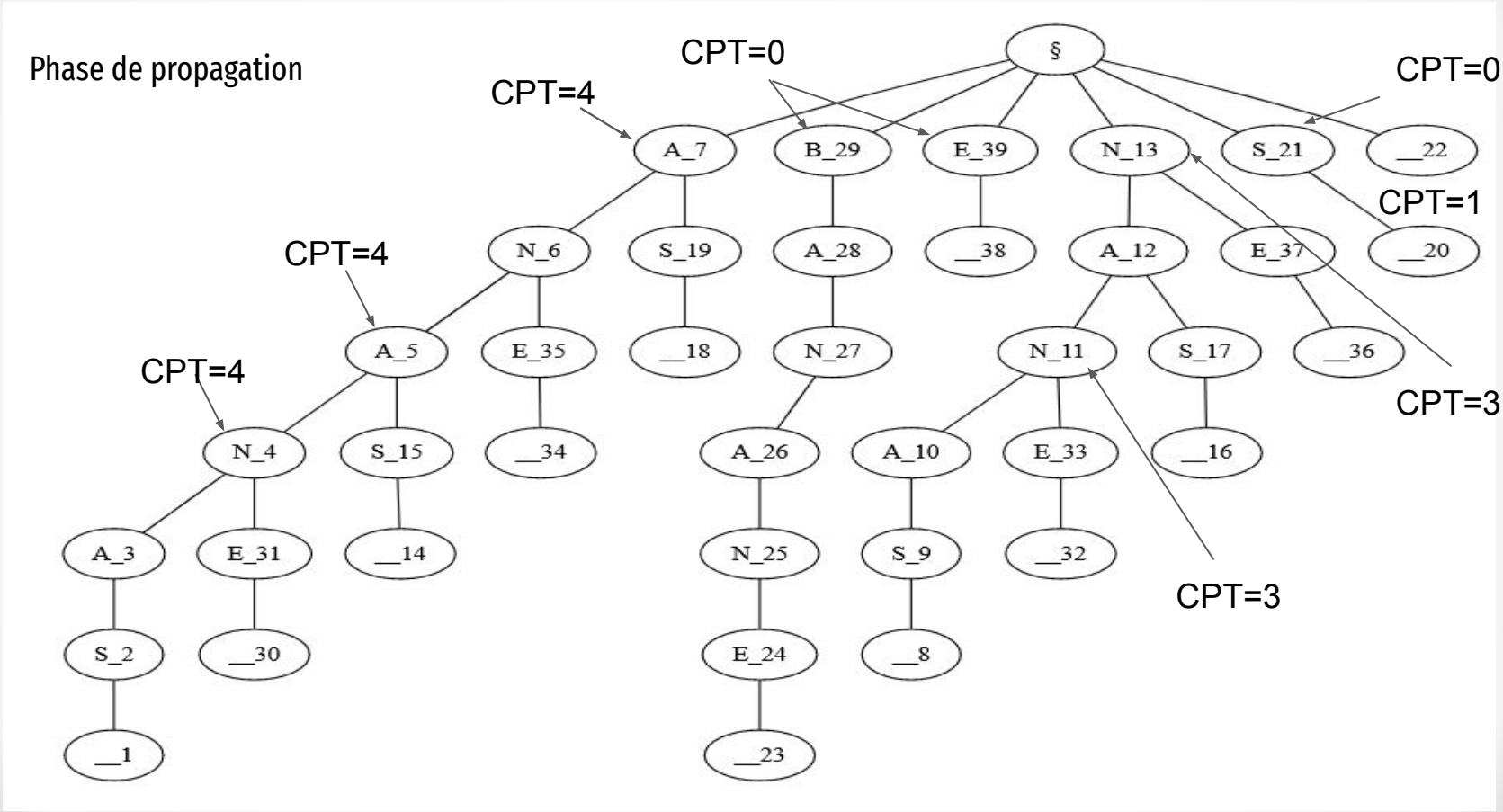
III. Arbre des suffixes non compressé - Récupération de la sous chaîne commune.

Lors de de la création du graphe, chaque compteur de chaque nœud est modifié et propage sa valeur vers la racine. Pour récupérer la sous chaîne commune, il suffit de suivre le compteur le plus grand.

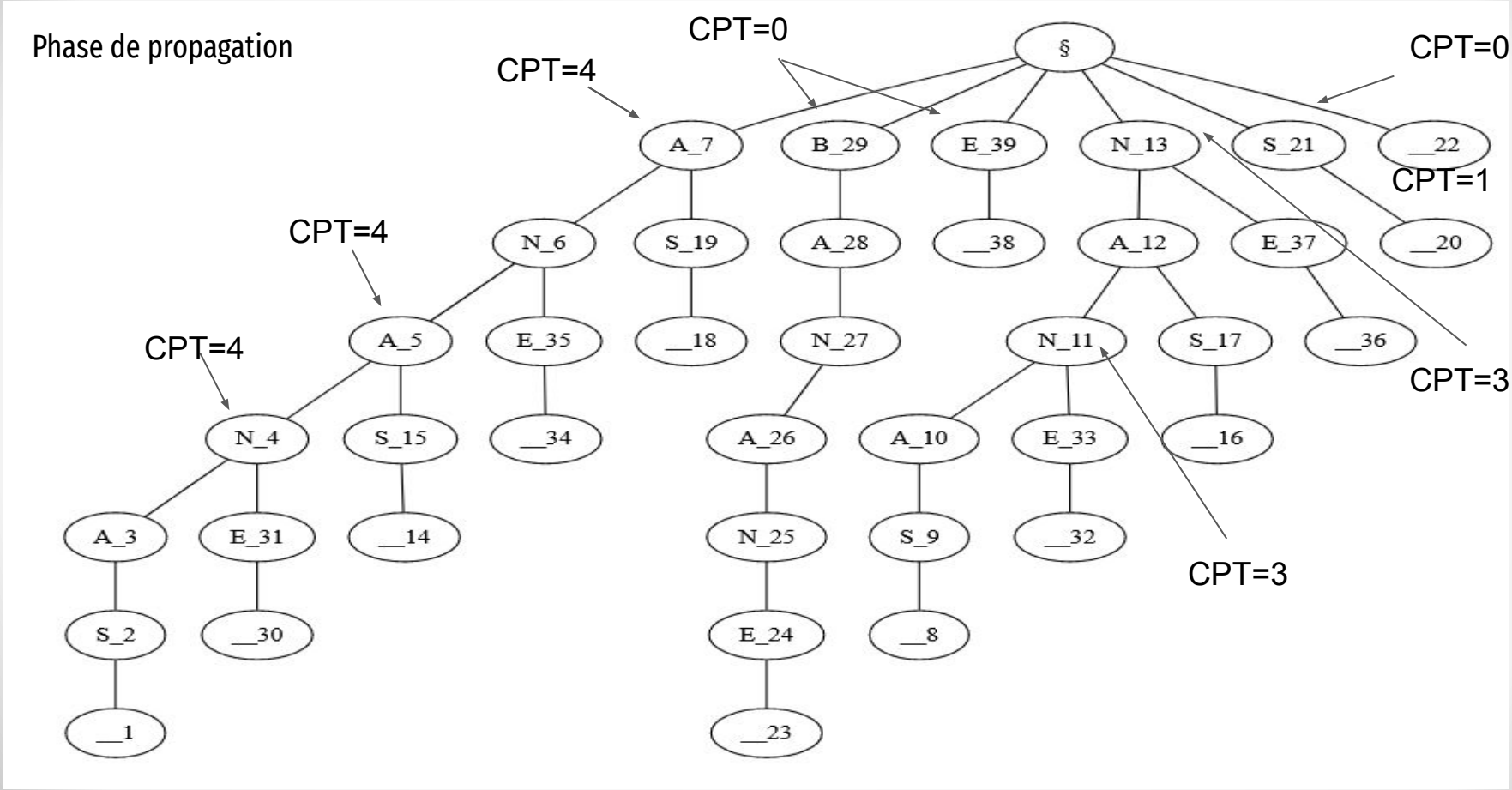
III. Arbre des suffixes non compressé - Exemple



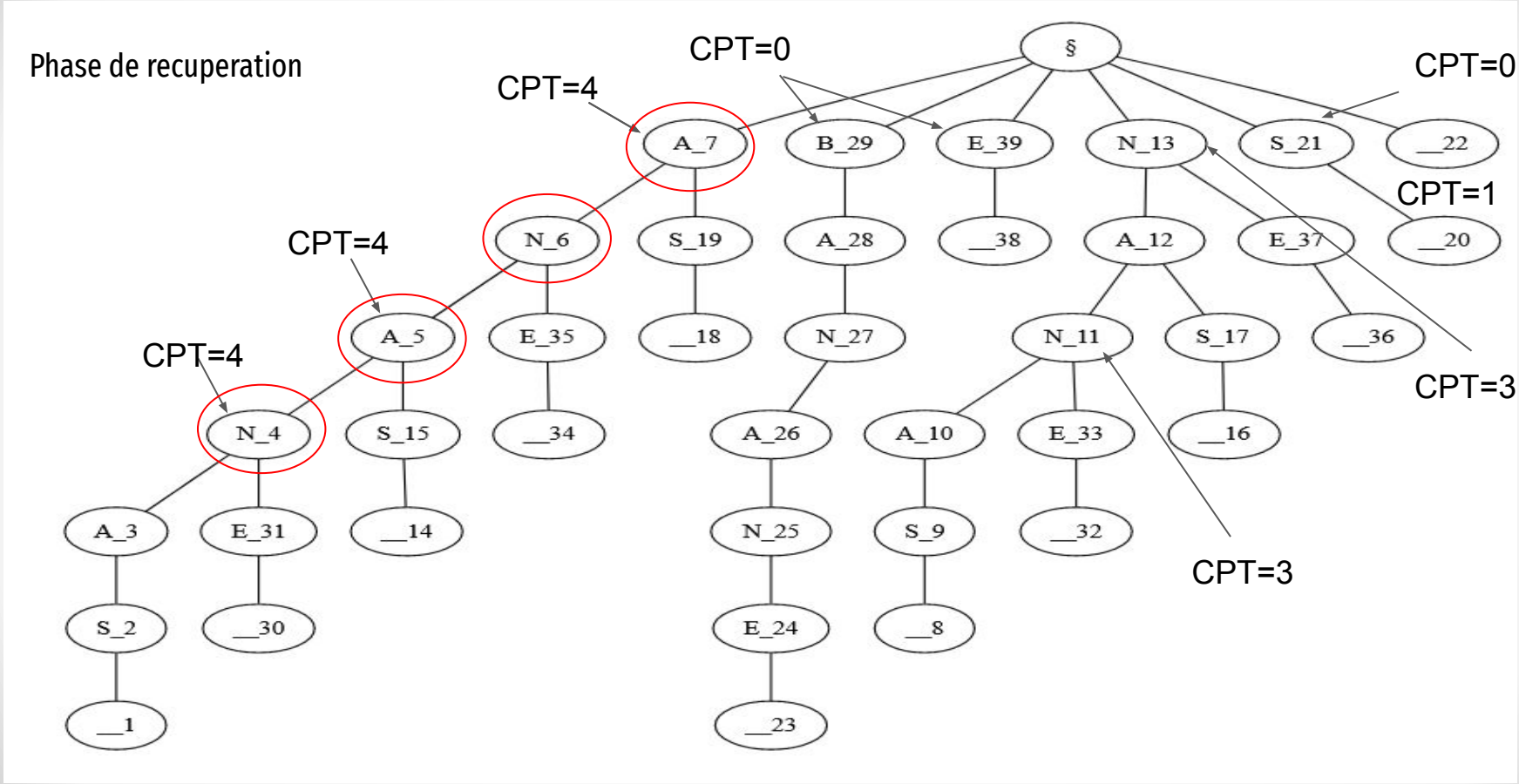
III. Arbre des suffixes non compressé - Exemple



III. Arbre des suffixes non compressé - Exemple



III. Arbre des suffixes non compressé - Exemple



IV. Compression des arbres des suffixes

Un arbre que l'on ne compresse pas contient énormément de nœuds et à une taille conséquente
exemple: l'arbre des suffixes de l'alphabet: 351 nœuds pour 26 caractères initiaux!

IV. Compression des arbres des suffixes

Un arbre que l'on ne compresse pas contient énormément de nœuds et à une taille conséquente
exemple: l'arbre des suffixes de l'alphabet: 351 nœuds pour 26 caractères initiaux!

Solution: Fusionner les fils uniques avec leur père.

IV. Compression des arbres des suffixes

Un arbre que l'on ne compresse pas contient énormément de nœuds et à une taille conséquente
exemple: l'arbre des suffixes de l'alphabet: 351 nœuds pour 26 caractères initiaux!

Solution: Fusionner les fils uniques avec leur père.

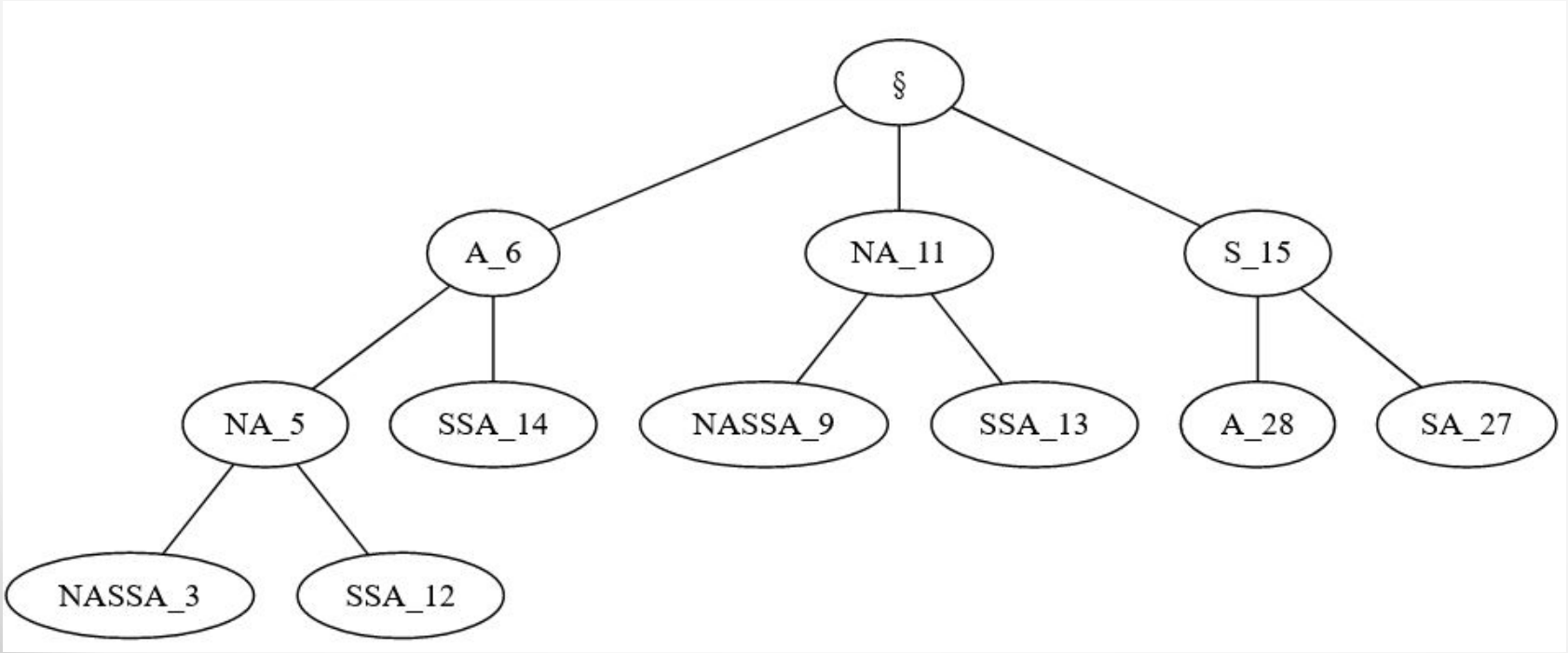
Conséquence: |Arbre des suffixes de l'alphabet | : 351 —> 26 noeuds

IV. Compression des arbres des suffixes

La compression d'un arbre suffixe nécessite d'ajouter un caractère spécial !

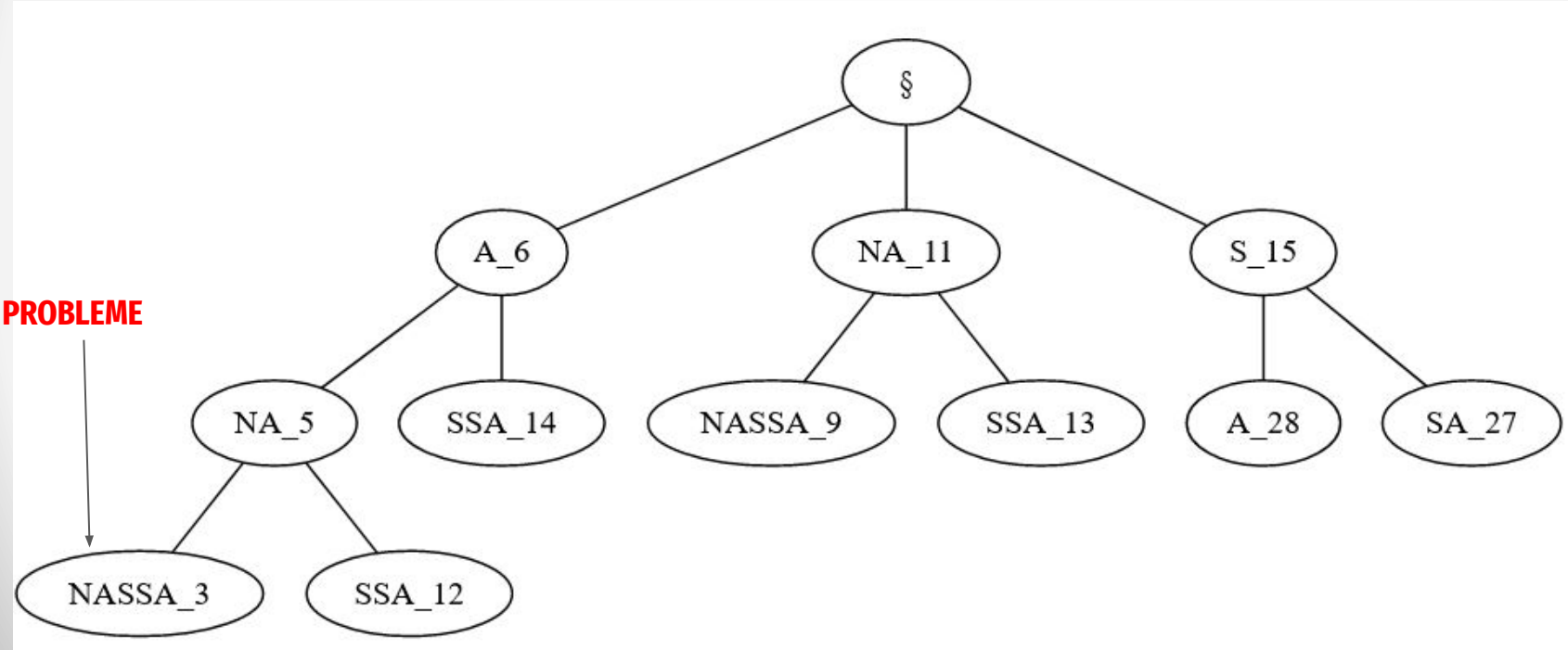
IV. Compression des arbres des suffixes

Exemple pour ANANAS ANANASSA :



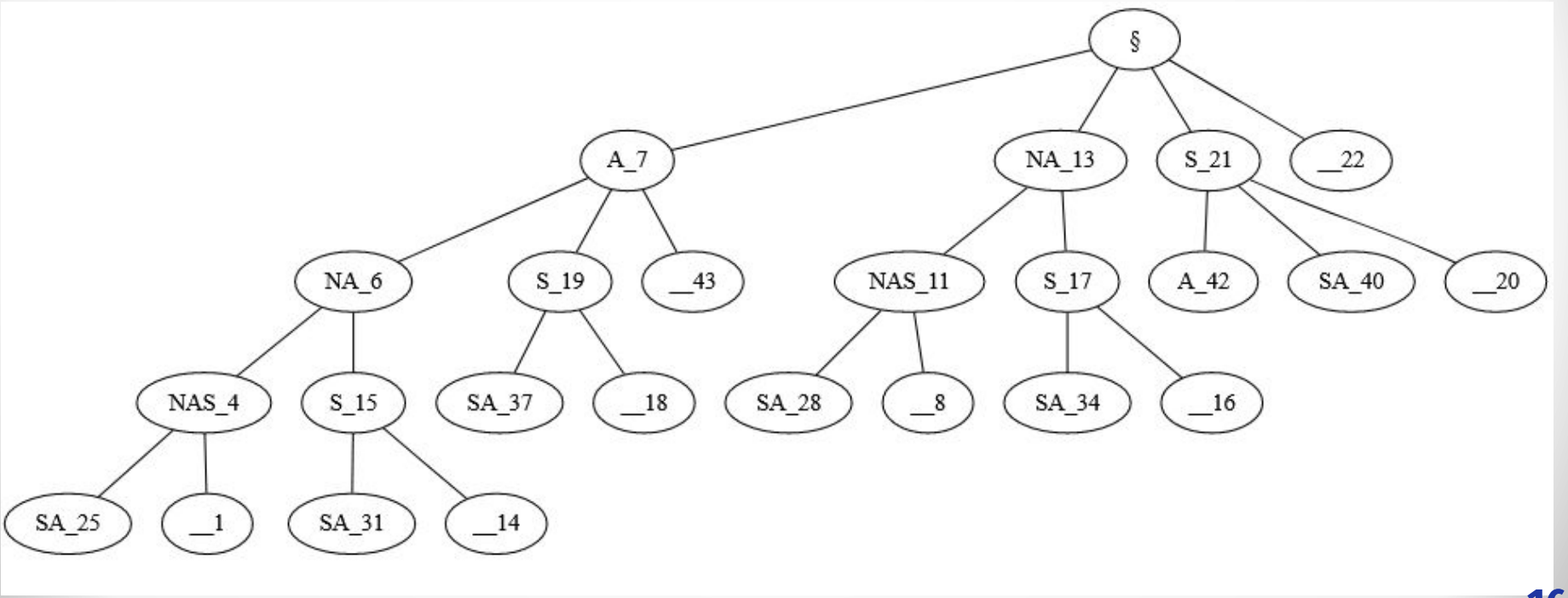
IV. Compression des arbres des suffixes

Exemple pour ANANAS ANANASSA :



IV. Compression des arbres des suffixes

Exemple pour ANANAS ANANASSA :

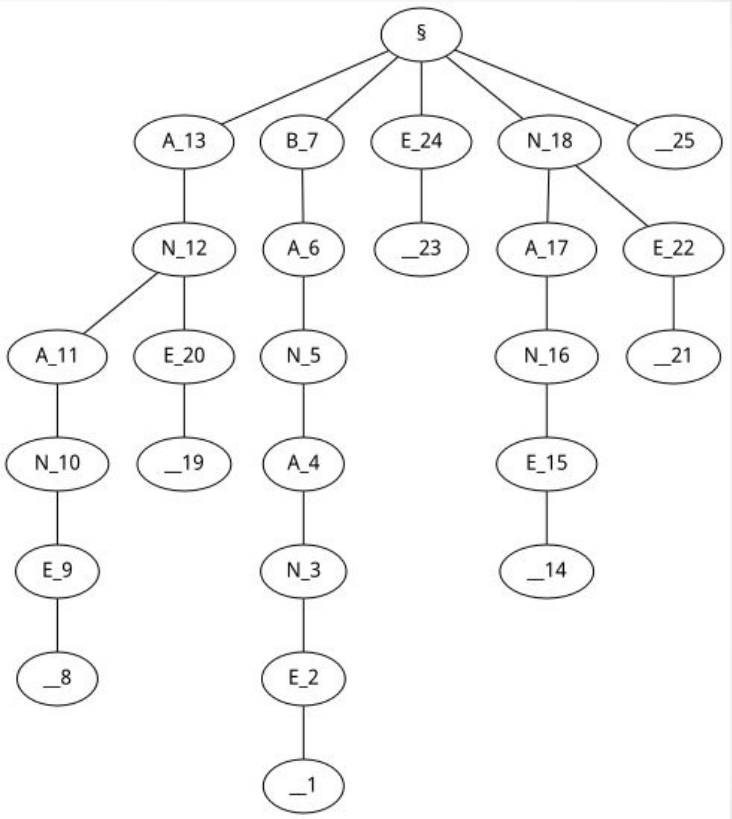


IV. Compression des arbres des suffixes - Complexités

	Complexité temporelle au pire(comparaison)	Complexité spatiale au pire
Programmation dynamique	O(nm)	O(nm)(en chiffre)
Arbre des suffixes (sans compression)	$O(n^2)$	$O(n^2)$(en caractère)
Arbre des suffixes avec compression	$O(n^2)$	$O(n^2)$(en caractère)

IV. Compression des arbres des suffixes - Arbre avec indices

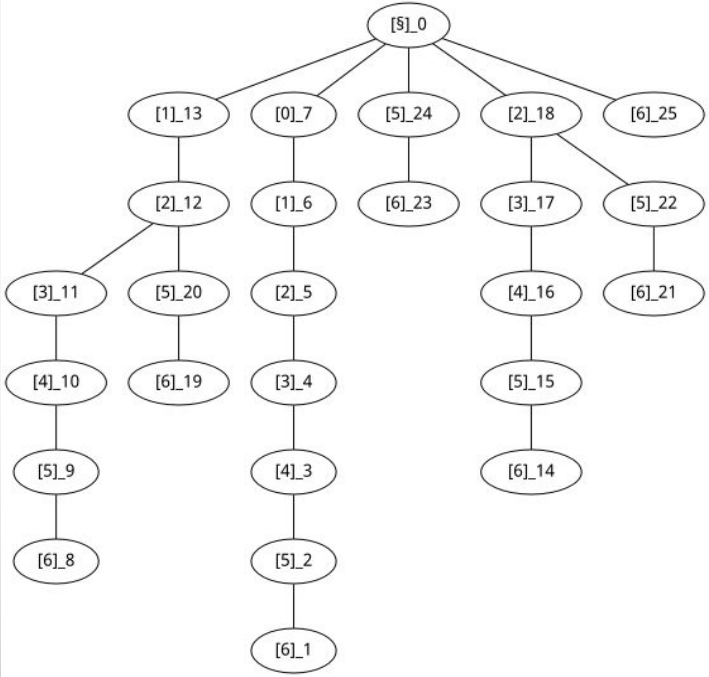
- Transformation des étiquettes dans l'arbre des suffixes
- Passage des lettres aux indices à travers une table de hachage de taille la longueur de la chaîne/données



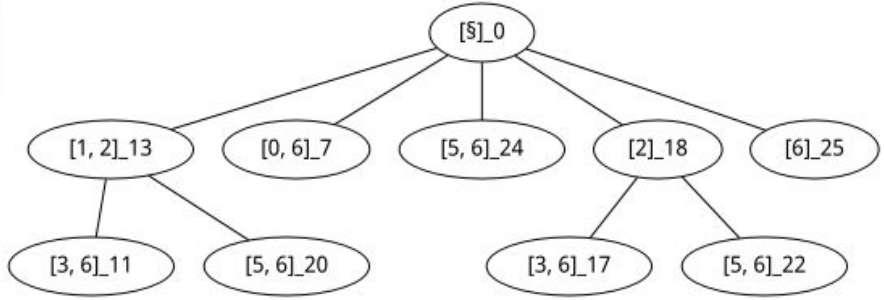
BANANE_

IV. Compression des arbres des suffixes - Arbre avec indices compressé

- Compression des arbres suffixes avec indices à partir de l'arbre non compressé des indices selon la même règle que pour les arbre suffixes avec lettres.
- On ne garde que l'indice de début et de fin pour les suites de caractères.
  - Gain en complexité espace.



⇔



Complexité en O(n).

V. Compression directe des arbres des suffixes - Récupération de la plus longue sous-chaîne commune

- Récupération basée sur un compteur comme pour les arbres non compressés.

Pseudo-code de l'algorithme de récupération de la plus longue sous-chaîne commune pour les arbres suffixes avec caractères et indices (pour les indices la vérification d'égalité des occurrences se fait avec la table de hachage).

Algorithme (C1, C2):

Création de l'arbre de suffixes compressés de C1 avec comme racine §.

Pour chaque sous-chaîne SSC de C2 **faire**:

Si le premier caractère de SSC n'existe pas parmi les nœuds de la racine § **alors** ajouter SSC à § avec comme compteur -1 # Pour identifier les SSC qui ne sont pas dans C1

Sinon itérer tant que les lettres des nœuds et de la SSC sont égales et augmenter un compteur passé par référence:

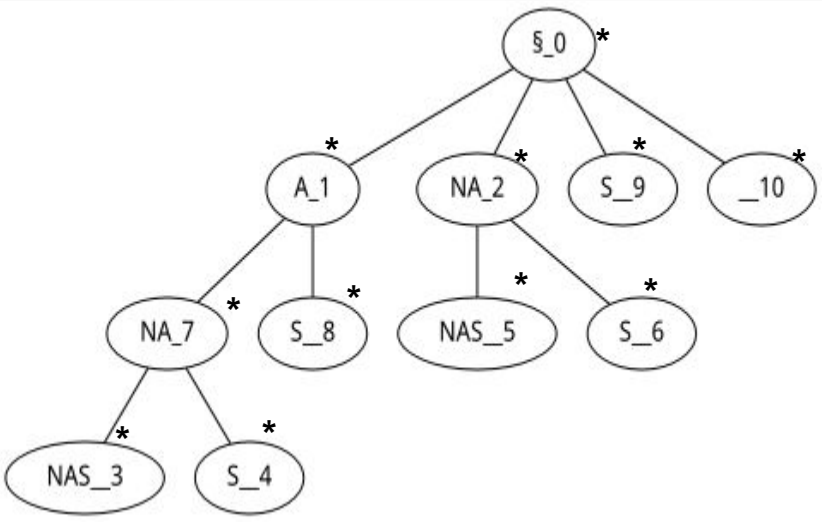
Si différence créer deux fils, l'un qui prendra la SSC restante et l'autre qui aura comme label l'étiquette restante du nœud ou il y avait la différence et aura comme fils les fils de son père + propager le compteur si compteur != -1.

Sinon s'arrêter si la SSC se termine + propager le compteur si compteur != -1.

V. Compression directe des arbres des suffixes - Récupération de la plus longue sous-chaîne commune

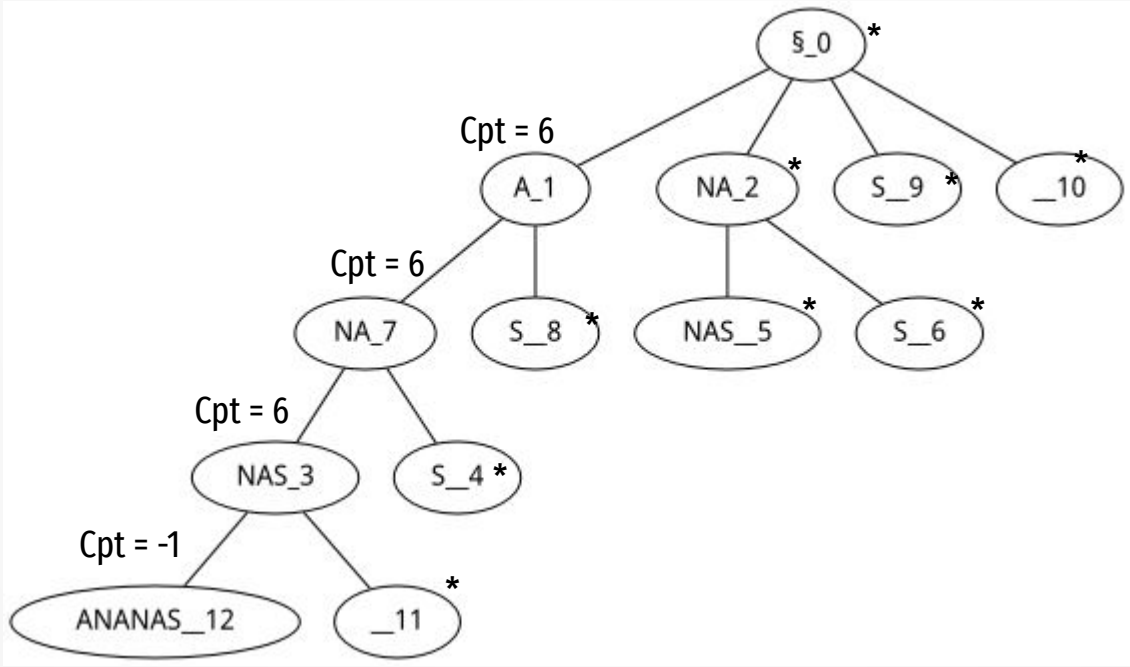
- Récupération de la plus longue sous-chaîne commune de “**ANANAS_**” “**ANANASANANAS_**”:

Cpt = 0 (*)



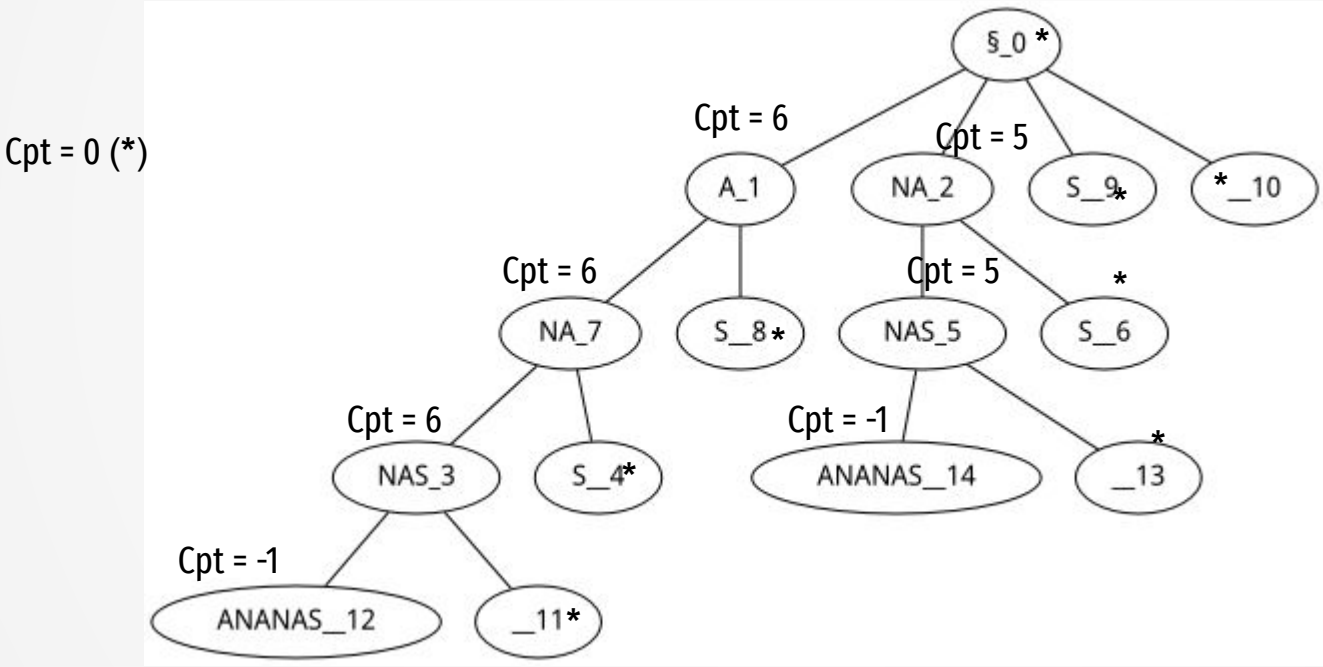
V. Compression directe des arbres des suffixes - Récupération de la plus longue sous-chaîne commune

- Récupération de la plus longue sous-chaîne commune de “**ANANAS_**” “**ANANASANANAS_**”:



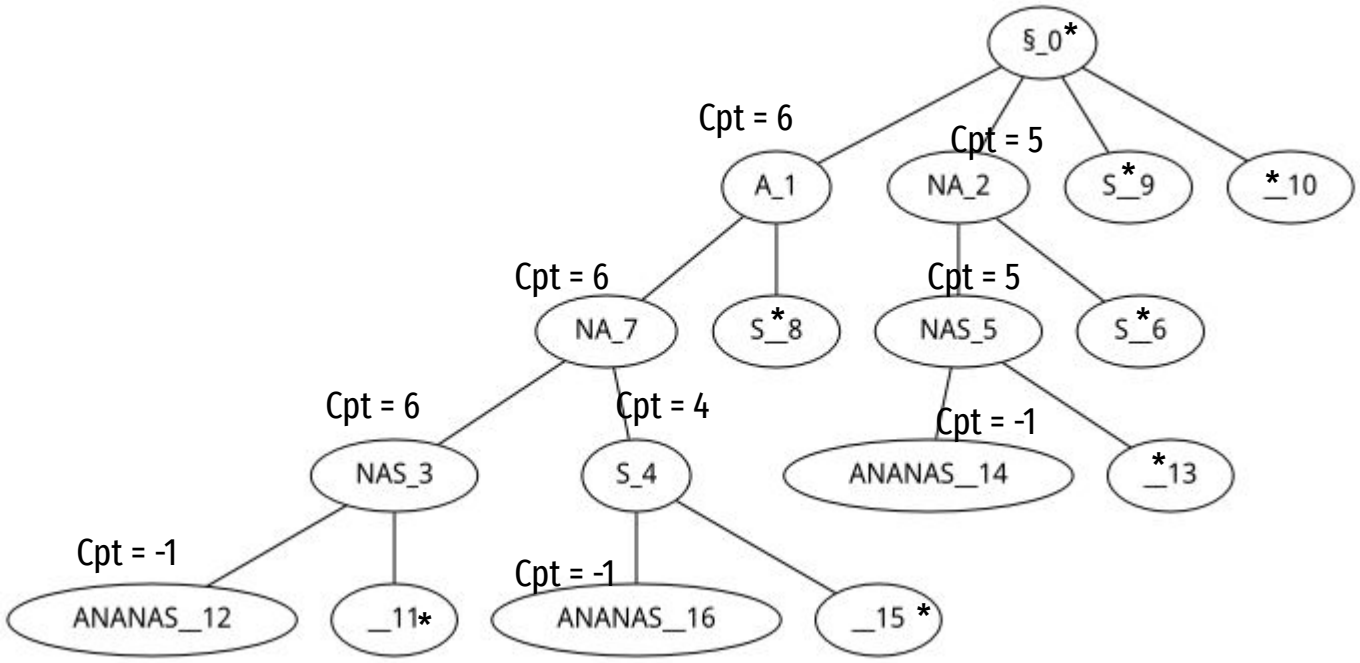
V. Compression directe des arbres des suffixes - Récupération de la plus longue sous-chaîne commune

- Récupération de la plus longue sous-chaîne commune de “**ANANAS_**” “**ANANASANANAS_**”:



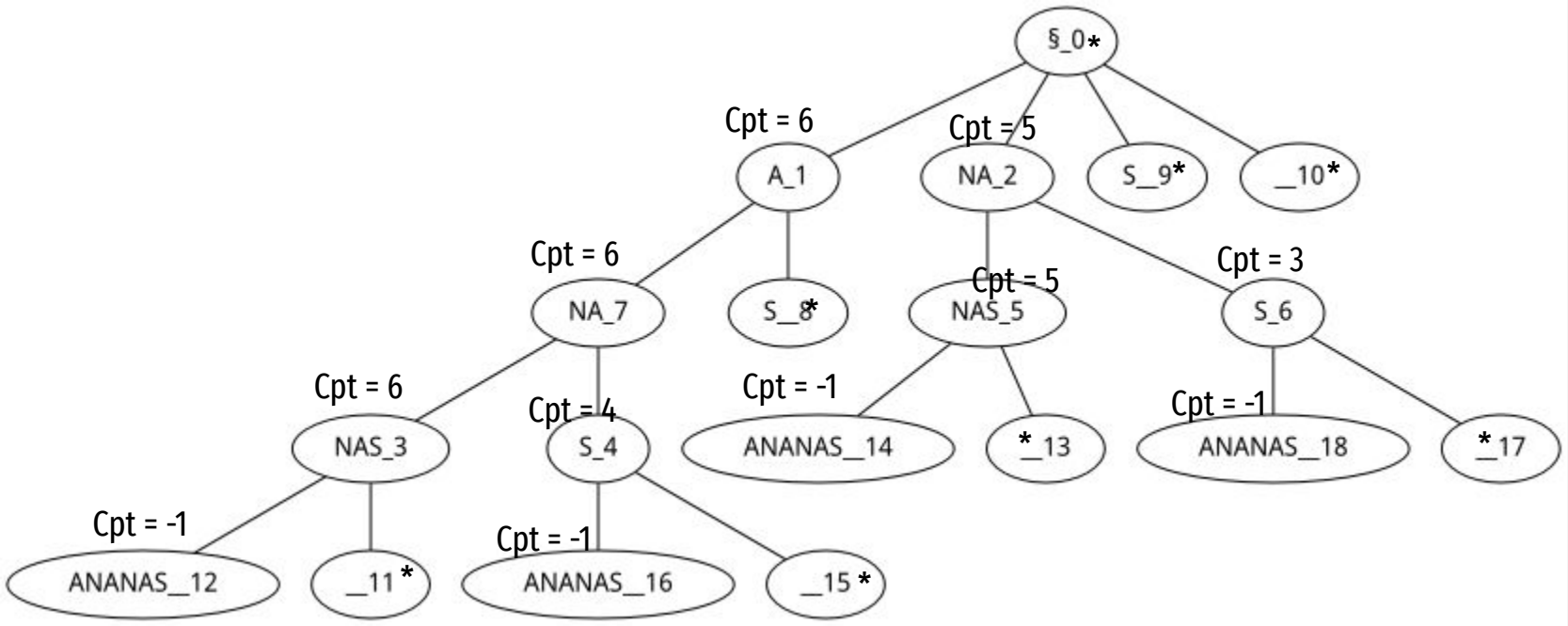
V. Compression directe des arbres des suffixes - Récupération de la plus longue sous-chaîne commune

- Récupération de la plus longue sous-chaîne commune de “**ANANAS_**” “**ANANASANANAS_**”:



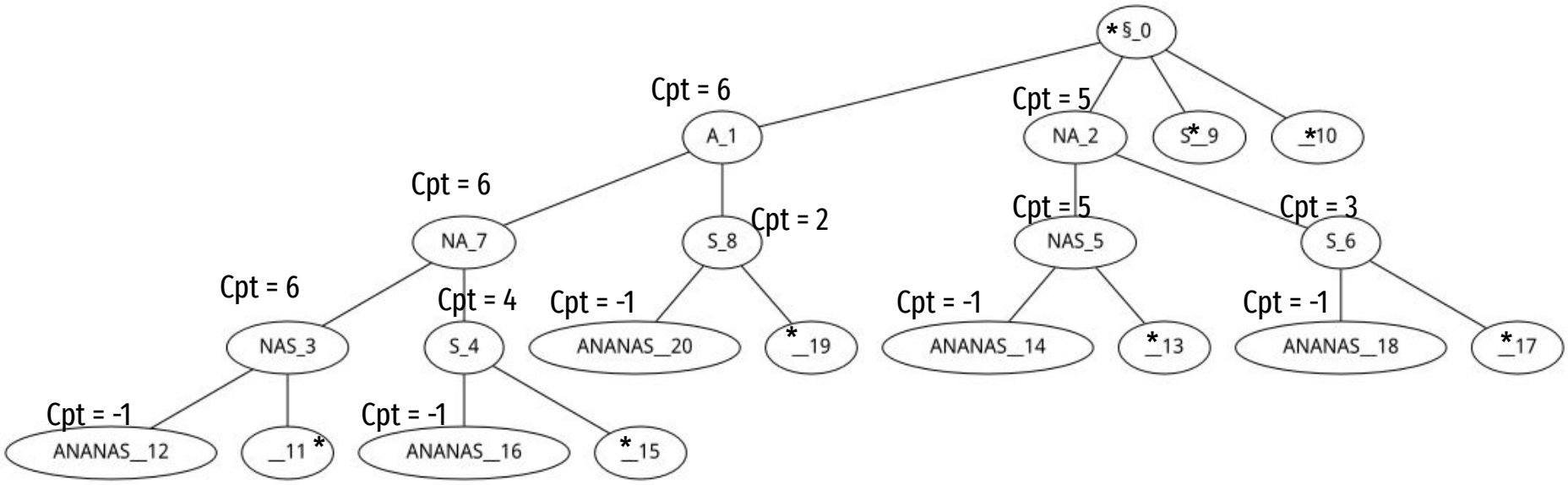
V. Compression directe des arbres des suffixes - Récupération de la plus longue sous-chaîne commune

- Récupération de la plus longue sous-chaîne commune de “**ANANAS_**” “**ANANASANANAS_**”:



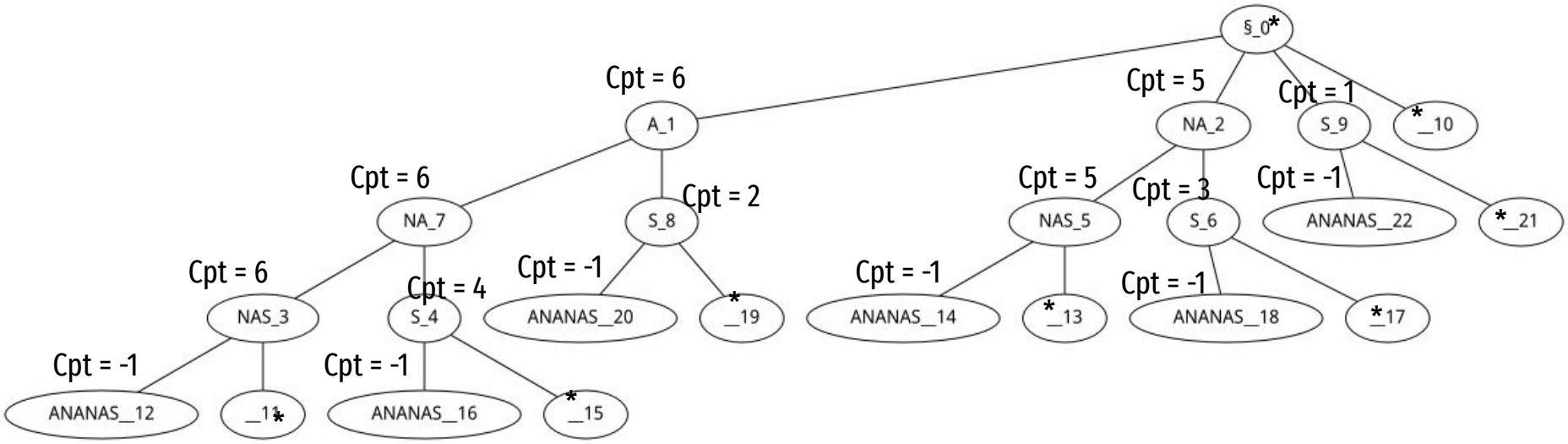
V. Compression directe des arbres des suffixes - Récupération de la plus longue sous-chaîne commune

- Récupération de la plus longue sous-chaîne commune de “**ANANAS_**” “**ANANASANANAS_**”:



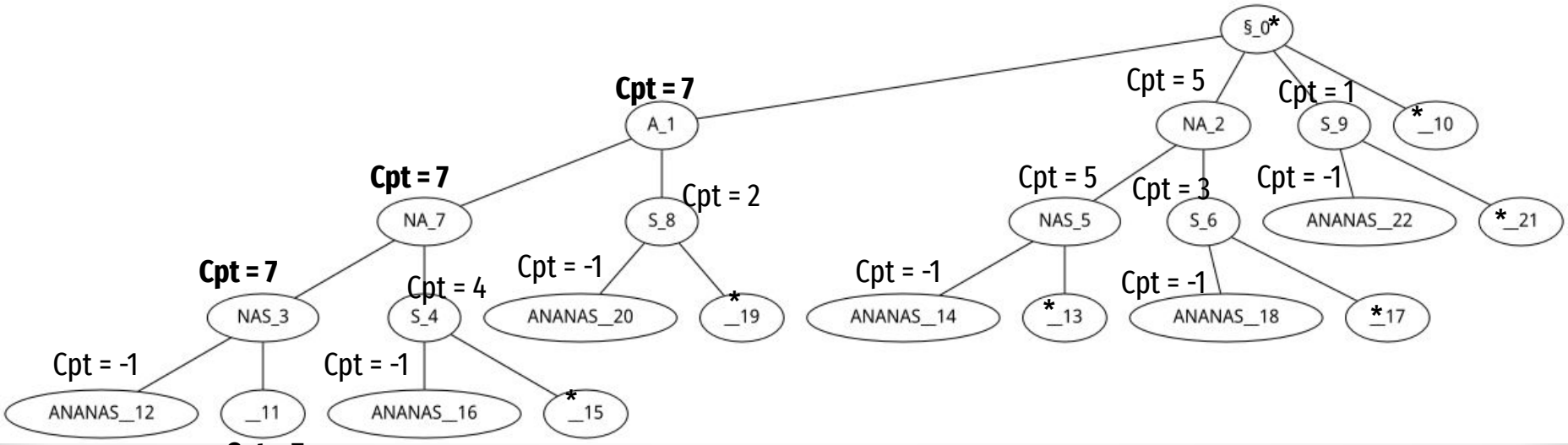
V. Compression directe des arbres des suffixes - Récupération de la plus longue sous-chaîne commune

- Récupération de la plus longue sous-chaîne commune de “**ANANAS_**” “**ANANASANANAS_**”:



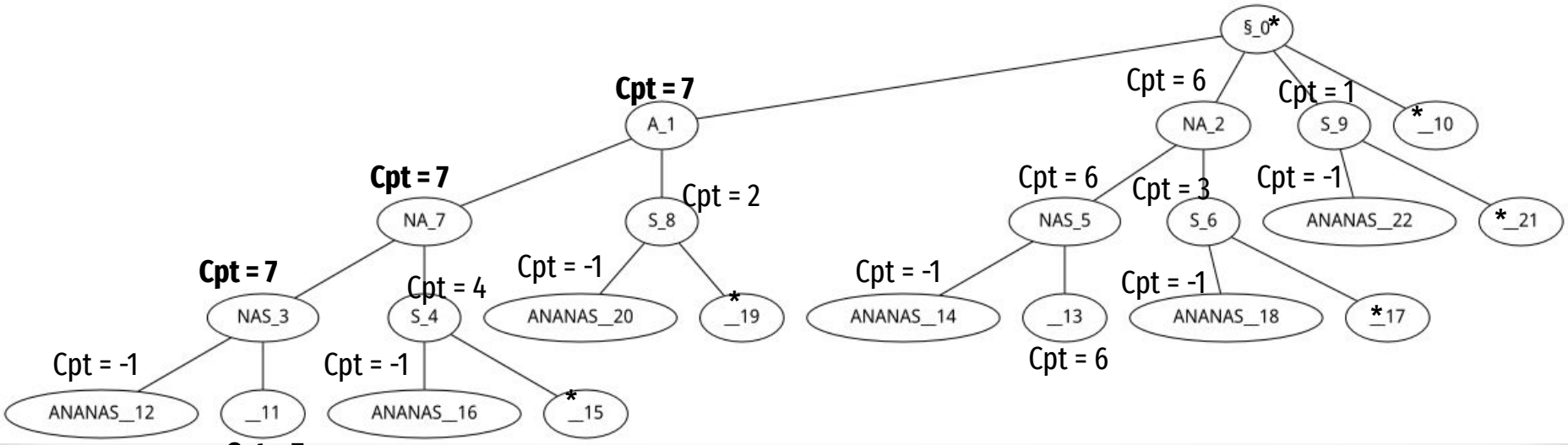
V. Compression directe des arbres des suffixes - Récupération de la plus longue sous-chaîne commune

- Récupération de la plus longue sous-chaîne commune de “**ANANAS_**” “**ANANASANANAS_**”:



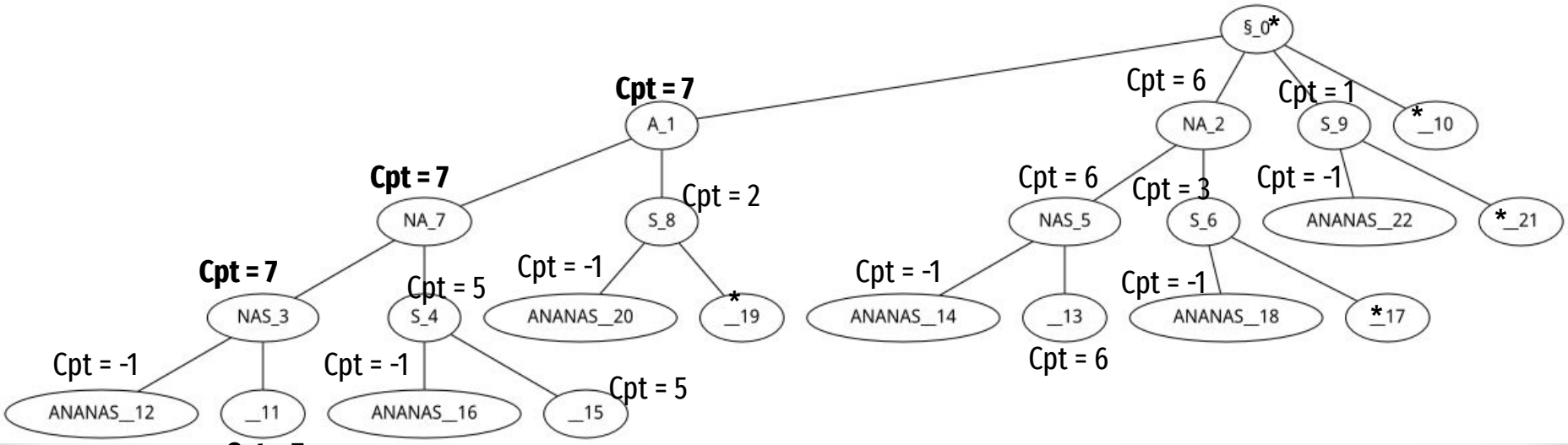
V. Compression directe des arbres des suffixes - Récupération de la plus longue sous-chaîne commune

- Récupération de la plus longue sous-chaîne commune de “**ANANAS_**” “**ANANASANANAS_**”:



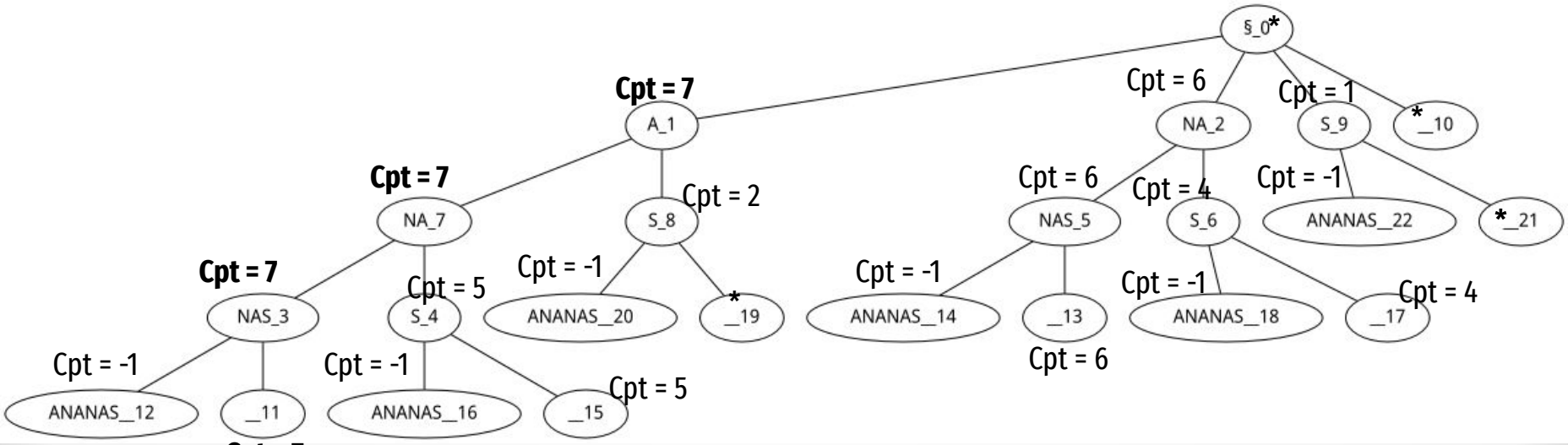
V. Compression directe des arbres des suffixes - Récupération de la plus longue sous-chaîne commune

- Récupération de la plus longue sous-chaîne commune de “**ANANAS_**” “**ANANASANANAS_**”:



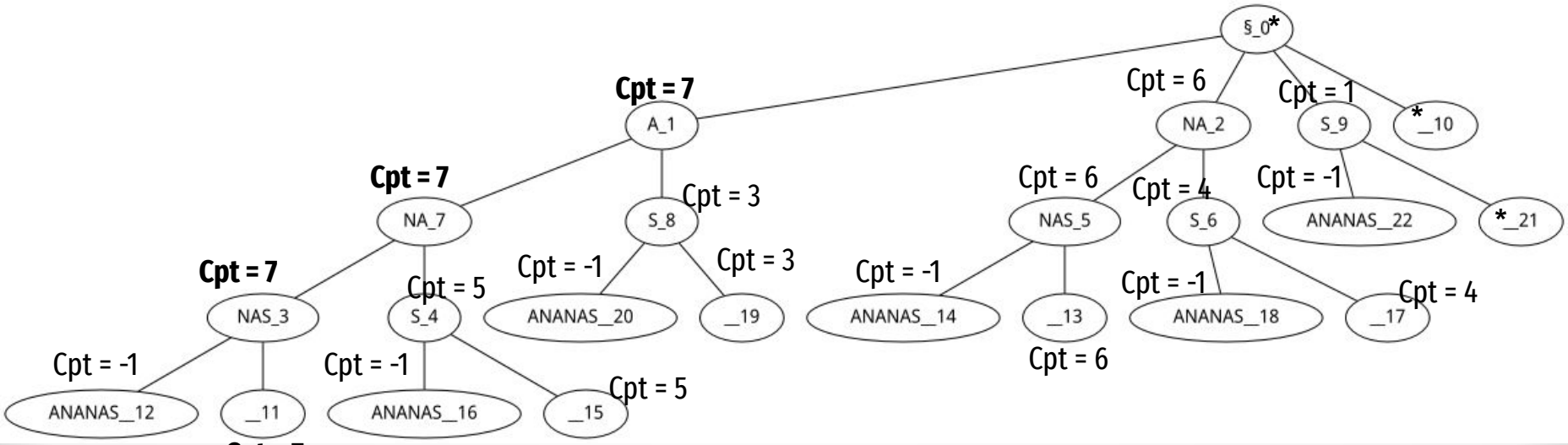
V. Compression directe des arbres des suffixes - Récupération de la plus longue sous-chaîne commune

- Récupération de la plus longue sous-chaîne commune de “**ANANAS_**” “**ANANASANANAS_**”:



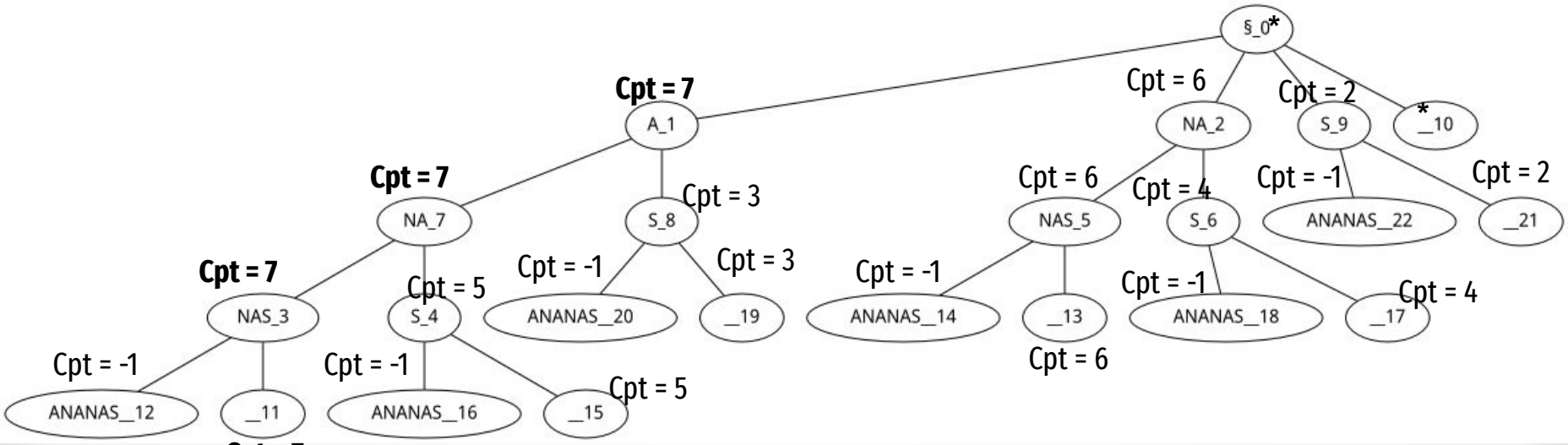
V. Compression directe des arbres des suffixes - Récupération de la plus longue sous-chaîne commune

- Récupération de la plus longue sous-chaîne commune de “**ANANAS_**” “**ANANASANANAS_**”:



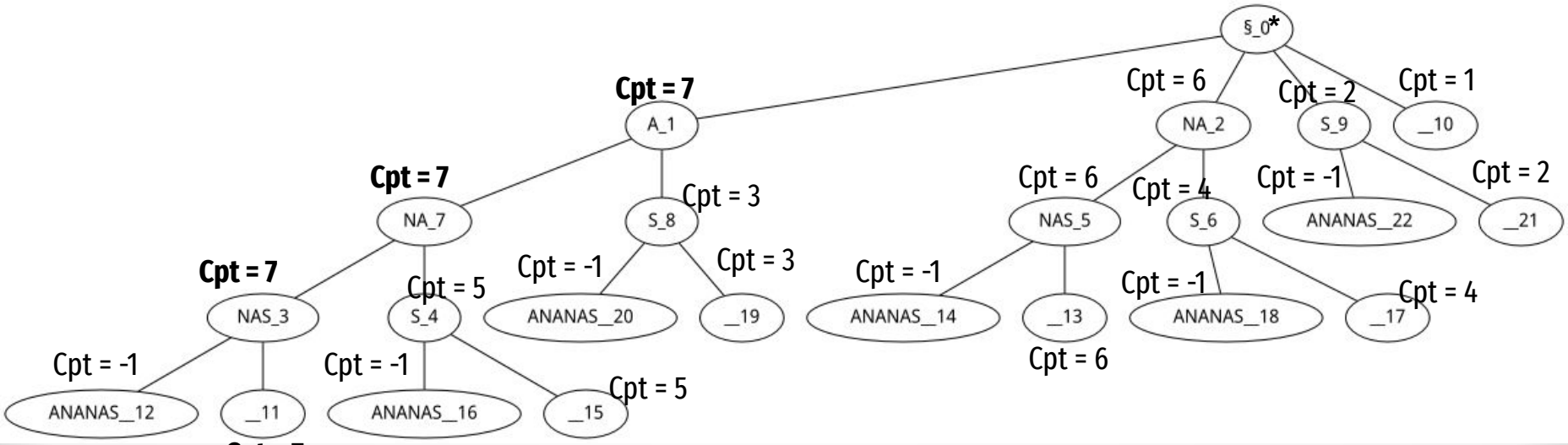
V. Compression directe des arbres des suffixes - Récupération de la plus longue sous-chaîne commune

- Récupération de la plus longue sous-chaîne commune de “**ANANAS_**” “**ANANASANANAS_**”:



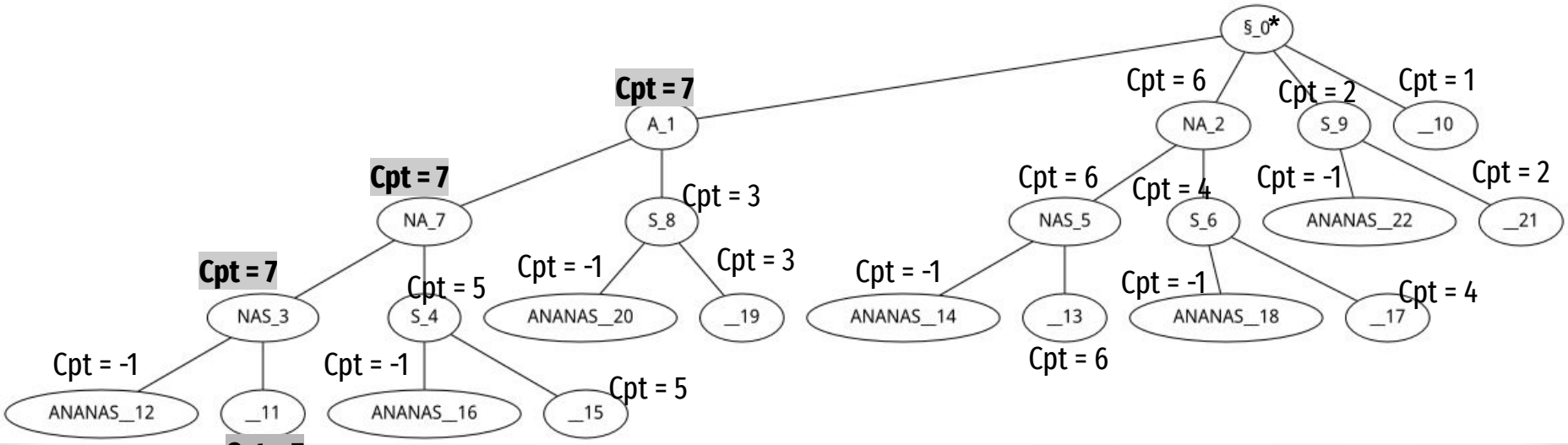
V. Compression directe des arbres des suffixes - Récupération de la plus longue sous-chaîne commune

- Récupération de la plus longue sous-chaîne commune de “**ANANAS_**” “**ANANASANANAS_**”:



V. Compression directe des arbres des suffixes - Récupération de la plus longue sous-chaîne commune

- Récupération de la plus longue sous-chaîne commune de “**ANANAS_**” “**ANANASANANAS_**”:



Complexité en $O(n^2)$.

VI. Expérimentation

```
//-------------------------------------------------------------------------------------------//
donnee0.txt vs donnee0.txt

§d\'abord confin\233e dans les monast\232res et limit\233e essentiellement au domaine religieux, la traduction s\'\233tend au domaine profane d\232s le xe  si\232cle. bient\244t apparaissent en langue romane des fabliaux, com\233dies ou romans imit\233s d\'oeuvres de l\'antiquit\233, et les premiers grands po\232tes - chr\233tien de troyes, marie de france, rutebeuf, jean de meung - sont avant tout traducteurs (\224 une \233poque o\249 traduction, imitation et cr\233ation ne sont gu\232re diff\233renci\233es). ce n\'est toutefois qu\'au xive si\232cle, marqu\233 par la cr\233ation des premi\232res universit\233s, que la traduction quitte les monast\232res et conna\238t un bref essor gr\226ce \224 la protection de la cour._
===> Methode Arbre suffixes compresse avec indices executee en: 0.389047 secondes

d\'abord confin\233e dans les monast\232res et limit\233e essentiellement au domaine religieux, la traduction s\'\233tend au domaine profane d\232s le xe  si\232cle. bient\244t apparaissent en langue romane des fabliaux, com\233dies ou romans imit\233s d\'oeuvres de l\'antiquit\233, et les premiers grands po\232tes - chr\233tien de troyes, marie de france, rutebeuf, jean de meung - sont avant tout traducteurs (\224 une \233poque o\249 traduction, imitation et cr\233ation ne sont gu\232re diff\233renci\233es). ce n\'est toutefois qu\'au xive si\232cle, marqu\233 par la cr\233ation des premi\232res universit\233s, que la traduction quitte les monast\232res et conna\238t un bref essor gr\226ce \224 la protection de la cour._
===> Methode Programation Dynamique executee en: 0.027099 secondes

§d'abord confin€e dans les monast€res et limit€e essentiellement au domaine religieux, la traduction s'€tend au domaine profane d€s le xe  si€cle. bient€t apparaissent en langue romane des fabliaux, com€dies ou romans imit€s d'oeuvres de l'antiquit€, et les premiers grands po€tes - chr€tien de troyes, marie de france, rutebeuf, jean de meung - sont avant tout traducteurs (€ une €poque o€ traduction, imitation et cr€ation ne sont gu€re diff€renci€es). ce n'est toutefois qu'au xive si€cle, marqu€ par la cr€ation des premi€res universit€s, que la traduction quitte les monast€res et conna€t un bref essor gr€ce € la protection de la cour._
===> Methode Arbre suffixes compresses avec lettres executee en: 0.028202 secondes

//-------------------------------------------------------------------------------------------//
donnee0.txt vs donnee1.txt

§d\'abord confin\233e dans les monast\232res et limit\233e essentiellement au domaine religieux, la traduction s\'\233tend au domaine profane d\232s le xe  si\232cle. bient\244t apparaissent en langue romane des fabliaux, com\233dies ou romans imit\233s d\'oeuvres de l\'antiquit\233, et les premiers grands po\232tes - chr\233tien de troyes, marie de france, rutebeuf, jean de meung - sont avant tout traducteurs (\224 une \233poque o\249 traduction, imitation et cr\233ation ne sont gu\232re diff\233renci\233es). ce n\'est toutefois qu\'au xive si\232cle, marqu\233 par la cr\233ation des premi\232res universit\233s, que la traduction quitte les monast\232res et conna\238t un bref essor gr\226ce \224 la protection de la cour.
===> Methode Arbre suffixes compresse avec indices executee en: 0.380449 secondes

d\'abord confin\233e dans les monast\232res et limit\233e essentiellement au domaine religieux, la traduction s\'\233tend au domaine profane d\232s le xe  si\232cle. bient\244t apparaissent en langue romane des fabliaux, com\233dies ou romans imit\233s d\'oeuvres de l\'antiquit\233, et les premiers grands po\232tes - chr\233tien de troyes, marie de france, rutebeuf, jean de meung - sont avant tout traducteurs (\224 une \233poque o\249 traduction, imitation et cr\233ation ne sont gu\232re diff\233renci\233es). ce n\'est toutefois qu\'au xive si\232cle, marqu\233 par la cr\233ation des premi\232res universit\233s, que la traduction quitte les monast\232res et conna\238t un bref essor gr\226ce \224 la protection de la cour.
===> Methode Programation Dynamique executee en: 0.03401 secondes

§d'abord confin€e dans les monast€res et limit€e essentiellement au domaine religieux, la traduction s'€tend au domaine profane d€s le xe  si€cle. bient€t apparaissent en langue romane des fabliaux, com€dies ou romans imit€s d'oeuvres de l'antiquit€, et les premiers grands po€tes - chr€tien de troyes, marie de france, rutebeuf, jean de meung - sont avant tout traducteurs (€ une €poque o€ traduction, imitation et cr€ation ne sont gu€re diff€renci€es). ce n'est toutefois qu'au xive si€cle, marqu€ par la cr€ation des premi€res universit€s, que la traduction quitte les monast€res et conna€t un bref essor gr€ce € la protection de la cour.
===> Methode Arbre suffixes compresses avec lettres executee en: 0.034893 secondes
```

642 caractères vs 642 caractères

3

1

2

642 caractères vs 838 caractères

3

1

2

VI. Expérimentation

```
//-------------------------------------------------------------------------------------------//
donnee0.txt vs donnee2.txt

§ les premiers grands po\232tes - chr\233tien de troyes, marie de france, rutebeuf, jean de meung - sont
===> Methode Arbre suffixes compresse avec indices executee en: 0.109658 secondes

 les premiers grands po\232tes - chr\233tien de troyes, marie de france, rutebeuf, jean de meung - sont
===> Methode Programation Dynamique executee en: 0.033458 secondes

§ les premiers grands po?tes - chr?tien de troyes, marie de france, rutebeuf, jean de meung - sont
===> Methode Arbre suffixes compresses avec lettres executee en: 0.027521 secondes


//-------------------------------------------------------------------------------------------//
donnee0.txt vs donnee3.txt

§ des premi\232res universit\233s
===> Methode Arbre suffixes compresse avec indices executee en: 0.067488 secondes

 des premi\232res universit\233s
===> Methode Programation Dynamique executee en: 0.018764 secondes

§ des premi?res universit?s
===> Methode Arbre suffixes compresses avec lettres executee en: 0.018265 secondes


//-------------------------------------------------------------------------------------------//
donnee0.txt vs donnee4.txt

§ des premi\232res universit\233s
===> Methode Arbre suffixes compresse avec indices executee en: 0.067325 secondes

 des premi\232res universit\233s
===> Methode Programation Dynamique executee en: 0.019437 secondes

§ des premi?res universit?s
===> Methode Arbre suffixes compresses avec lettres executee en: 0.018809 secondes
```

642 caractères vs 858 caractères

3

2

1

642 caractères vs 497 caractères

3

2

1

642 caractères vs 518 caractères

3

2

1

VI. Expérimentation

```
//-------------------------------------------------------------------------------------------------//
donnee0.txt vs donnee5.txt

§ s\'\233tend
===> Methode Arbre suffixes compresse avec indices executee en: 0.050408 secondes

 s\'\233tend
===> Methode Programation Dynamique executee en: 0.008387 secondes

§ s'tend
===> Methode Arbre suffixes compresses avec lettres executee en: 0.013365 secondes



//-------------------------------------------------------------------------------------------------//
donnee0.txt vs donnee6.txt

§ dans l
===> Methode Arbre suffixes compresse avec indices executee en: 0.044108 secondes

 dans l
===> Methode Programation Dynamique executee en: 0.009522 secondes

§ dans l
===> Methode Arbre suffixes compresses avec lettres executee en: 0.012181 secondes
```

642 caractères vs 210 caractères

3

1

2

642 caractères vs 213 caractères

3

1

2

VII. Conclusion

- Sur des moyennes/grandes données, classement des algorithmes les plus rapides:
  1. Programmation Dynamique ≃ Algorithme suffixes compressés lettres
  2. Algorithme suffixes compressés indices

- Sur des petites données, classement des algorithmes les plus rapides:
  1. Programmation Dynamique
  2. Algorithme suffixes compressés lettres
  3. Algorithme suffixes compressés indices

- Sur des grandes données on préférera l'algorithme suffixe compressés indices pour un gain en espace.

- Possible utilisation dans les images en récupérant la plus longue sous-chaîne commune en bits?

Merci pour votre attention !