

Longest repeated substring

Etant donnée une chaîne de caractères s , nous nous intéressons à la recherche du plus long facteur présent au moins deux fois dans s (*longest repeated substring problem*, ou problème LRS). Par exemple, si $s = \text{aacaagtttacaagc}$, le plus long facteur présent deux fois dans s est acaag , présent deux fois à partir des indices $i = 1$ et $j = 9$ de la chaîne. Il s'agit d'un problème ayant des applications en cryptographie et surtout en bio-informatique.

Pour tester les différentes fonctions que vous écrirez, récupérez à l'adresse <http://info-llg.fr/commun-mp/textes/> trois fichiers nommés respectivement `essai1000.txt`, `essai10000.txt` et `essai100000.txt`. Chacun de ces fichiers est formé d'une ligne de respectivement 1 000, 10 000 et 100 000 caractères pris dans $\{g, t, a, c\}$. Le plus long facteur présent deux fois dans chacune de ces trois chaînes est respectivement de longueur 10, 13 et 16.

```
f = open('chemin/vers/mon/fichier.txt', 'r')
c = f.readline()
f.close()

# c est alors une chaîne de caractères égale à la première ligne du fichier ouvert
```

FIGURE 1 – Script à utiliser pour importer une chaîne de caractères à partir d'un fichier texte.

Recherche par force brute

La démarche naïve consiste à tester tous les couples d'entiers $i < j$ à la recherche du plus long préfixe commun à $s[i:]$ et $s[j:]$.

Question 1.

- Définir une fonction `prefixe(s, i, j)` qui prend en arguments une chaîne de caractères s et deux entiers distincts i et j et qui renvoie la longueur du plus long préfixe commun à $s[i:]$ et $s[j:]$.
- En déduire une fonction `lrs1(s)` qui prend en argument une chaîne de caractères s et renvoie un des plus long facteurs présents au moins deux fois dans s .
- Exprimer son coût en fonction de la longueur n de la chaîne de caractère s et de la longueur k de ce facteur.
- Tester votre fonction sur les fichiers `essai1000.txt` et `essai10000.txt`, en calculant à chaque fois la durée d'exécution, puis estimez le temps qu'il faudrait pour exécuter `lrs1` sur le fichier `essai100000.txt`.

```
from time import time

d = -time()
# ici on place le script dont on souhaite mesurer le temps d'exécution
d += time()

# d contient alors la durée d'exécution en secondes.
```

FIGURE 2 – Mesure du temps en PYTHON.

Tableau des suffixes

Une façon d'accélérer la recherche consiste à considérer le *tableau des suffixes* de s , défini en regroupant les suffixes de s rangés par ordre lexicographique. La figure 3 représente tout d'abord les différents suffixes du mot $s = \text{aacaagtttacaagc}$ rangés par leur rang puis par ordre lexicographique.

Bien entendu, pour des raisons d'occupation en mémoire on ne stocke pas les suffixes eux-mêmes mais uniquement leur rang. Autrement dit, le tableau des suffixes de $s = \text{aacaagtttacaagc}$ sera représenté par le tableau des entiers $t = [0, 11, 3, 9, 1, 12, 4, 14, 10, 2, 13, 5, 8, 7, 6]$.

0	aacaagtttacaagc
1	acaagtttacaagc
2	caagtttacaagc
3	aagtttacaagc
4	agtttacaagc
5	gtttacaagc
6	tttacaagc
7	ttacaagc
8	tacaagc
9	acaagc
10	caagc
11	aagc
12	agc
13	gc
14	c

0	aacaagtttacaagc
11	aagc
3	aagtttacaagc
9	acaagc
1	acaagtttacaagc
12	agc
4	agtttacaagc
14	c
10	caagc
2	caagtttacaagc
13	gc
5	gtttacaagc
8	tacaagc
7	ttacaagc
6	tttacaagc

FIGURE 3 – Le tableau des suffixes de $s = \text{aacaagtttacaagc}$.

Question 2.

a) Rédiger une fonction $\text{lrs2}(s, t)$ qui prend en argument une chaîne de caractères s et son tableau des suffixes t et qui renvoie un des plus long facteurs présents au moins deux fois dans s .

b) Exprimer son coût en fonction de la longueur n de s et de la longueur k de ce facteur.

Calcul du tableau des suffixes

On se propose maintenant de calculer le tableau des suffixes en appliquant la méthode de tri *3-way radix quicksort* qui combine les avantages du tri radix et du tri rapide pour trier efficacement des chaînes de caractères.

Le principe de ce tri est de segmenter le tableau des suffixes à partir d'une de leurs lettres. Par exemple, si on segmente les suffixes du mot $s = \text{aacaagtttacaagc}$ à partir de leurs premières lettres en choisissant pour pivot la lettre c on obtient une segmentation en trois parties du tableau :

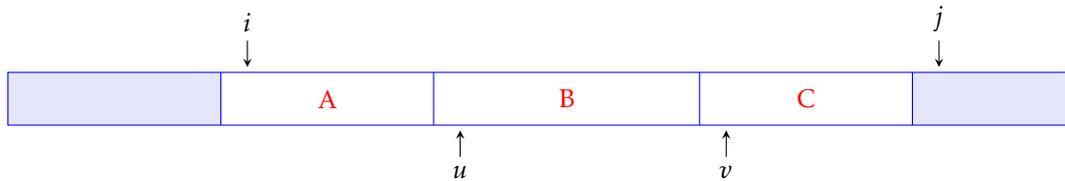
0	aacaagtttacaagc
1	acaagtttacaagc
3	aagtttacaagc
4	agtttacaagc
9	acaagc
11	aagc
12	agc

2	caagtttacaagc
10	caagc
14	c

5	gtttacaagc
6	tttacaagc
7	ttacaagc
8	tacaagc
13	gc

Le premier et le troisième de ces segments sont de nouveau segmentés à partir des premières lettres des mots qu'ils contiennent, tandis que le deuxième segment est segmenté à partir des *secondes* lettres des mots, puisqu'ils commencent tous par la lettre c . Le processus se répète ensuite récursivement.

Question 3. Rédiger une fonction $\text{segmente}(s, t, i, j, k)$ qui prend en argument une chaîne de caractères s , le tableau des suffixes à segmenter t et trois entiers i, j, k , et qui réalise la segmentation du tableau $t[i : j]$ à partir de la $(k+1)^{\text{e}}$ lettre du suffixe de rang $t[i]$. Concrètement, après appel à $\text{segmente}(s, t, i, j, k)$ le tableau t respectera l'invariant suivant :



- $\forall x \in A$, la $(k + 1)^{\text{e}}$ lettre de $s[x:]$ est strictement inférieure au pivot choisi ;
- $\forall x \in B$, la $(k + 1)^{\text{e}}$ lettre de $s[x:]$ est égale au pivot choisi ;
- $\forall x \in C$, la $(k + 1)^{\text{e}}$ lettre de $s[x:]$ est strictement supérieure au pivot choisi.

En outre, cette fonction renverra le couple d'indices (u, v) .

On notera que la $(k + 1)^{\text{e}}$ lettre d'un suffixe $s[x:]$ n'existe pas dès lors que $x + k \geq n$. On conviendra que dans ce cas celle-ci est égale au caractère ' ' (un espace), caractère strictement inférieur à toute lettre de l'alphabet dans l'ordre lexicographique.

Question 4.

- a) À l'aide de la fonction `segmente`, rédiger une fonction récursive `quick3way(s, t)` qui prend pour arguments la chaîne de caractères s et le tableau des suffixes t et qui trie en place ce dernier suivant la méthode *3-way radix quicksort*.
- b) En déduire une fonction `suffixes(s)` qui prend en argument une chaîne de caractères s et qui renvoie le tableau trié de ses suffixes.
- c) Tester cette méthode sur les trois fichiers `essai1000.txt`, `essai10000.txt` et `essai100000.txt`, en mesurant à chaque fois la durée d'exécution.

```

a a c a a g t t t a c a a g c a t g a t g c t g t a c t a g g a g a g t t a t a c t g g t c g
t c a a a c c t g a a c c t a a t c c t t g t g t g t a c a c a c a c t a c t a c t g t c g t
c g t c a t a t a t c g a g a t c a t c g a a c c g g a a g g c c g g a c a a g g c g g g g g
g t a t a g a t a g a t a g a c c c c t a g a t a c a c a t a c a t a g a t c t a g c t a g c
t a g c t c a t c g a t a c a c a c t c t c a c a c t c a a g a g t t a t a c t g g t c a a c
a c a c t a c t a c g a c a g a c g a c c a a c c a g a c a g a a a a a a a a c t c t a t a t

```

FIGURE 4 – Un exemple avec une chaîne de longueur 282.