Mo-Mo-Motifs

Consignes

Les consignes suivantes **doivent** être respectées sous peine d'être pénalisé par des **points négatifs**.

Le code doit être clair.

- Évitez les lignes à rallonge. S'il faut introduire une variable intermédiaire, faîte le.
- Nommez vos variables avec du sens. Évitez toutefois des noms trop longs. Auquel cas décrivez le rôle de la variable en commentaire.
- Ne surchargez pas de commentaires. Les commentaires utiles sont :
  - Description du rôle d'une variable
  - Description du rôle d'une boucle complexe.

Soignez la présentation de votre code.

- Marquez clairement les indentations.
- Éviter les ratures trop nombreuses dans un code. Une ou deux, maximum.
- Écrivez votre instruction sur plusieurs lignes si elle est longue en ajoutant simplement une indentation après le retour à la ligne.

Respectez la syntaxe Python. - *0 à l'exercice si syntaxe inventée.*

- Vérifiez que ce que vous écrivez respecte la syntaxe.
- Les petites erreurs n'implique pas 0 (oublie de :, = au lieu de ==,…)
- Si vous avez une idée, mais que vous n'arrivez pas à l'écrire, n'inventez pas. Écrivez votre idée en commentaire.
- Seuls les instructions de la fiche de syntaxe sont autorisées.

Pour nommer une fonction et ses entrées, utilisez les noms donnés dans l'énoncé - *0 à l'exercice sinon*

Utilisation de `print` ou de `input` interdite - *0 à l'exercice sinon*

I Algorithmes de cours

Question 1. *Comptage*

Entrées : Liste l et un objet x
Sortie : Le nombre de fois que x apparaît dans l

Question 2. *Création d'une matrice* `n x m`

Entrées : n, m deux entiers positifs non nuls
Sortie : Une matrice de dimension $n \times m$ remplie de `None`

II Mot de 7 lettres

Question 3. *Hachage caractère*

Implémenter une fonction `hord(c)` qui prend en entrée un caractère `c` parmi les lettres minuscules de l'alphabet et renvoie sa position dans l'alphabet. `'a'` est à la position `1` et `'z'` à la position `26`.

On pourra supposer qu'il existe une variable `minascii` dans l'environnement global définie par :

`minascii = "abcdefghijklmnopqrstuvwxyz"`.

Pour une lettre `c`, on appellera *haché* de `c` la valeur de `hord(c)`.

Question 4. *Hachage par somme*

Implémenter une fonction `hsomme(s)` qui prend en entrée une chaîne de caractères alphabétiques minuscules et renvoie la somme des hachés des caractères de la chaîne.

Pour une chaîne `s` de caractères alphabétiques minuscules, on appellera *haché* de `s` la valeur de `hsomme(c)`.

Question 5. *Hachage égaux*

Donner deux chaînes de caractères différentes qui ont le même haché.

Question 6. *Hachage séquentiel*

On considère les deux chaînes suivante : `s1 = "place"` et `s2 = "lacer"`.

Expliquer comment passer du haché de `s1` au haché de `s2` à l'aide uniquement d'une addition et d'une soustraction.

III Mot de 7 lettres

On rappelle le principe de l'algorithme de la fenêtre glissante.

Pour chercher une chaîne `s` de longueur `n` dans un texte `txt`, on délimite une fenêtre de recherche dans `txt` de longueur `n`, initialement débutant à l'indice 0 et qui se déplace d'indice en indice tant que la fenêtre ne fait pas apparaître exactement `s`.

Par exemple pour chercher `lit` dans `paleolithique`, on crée une fenêtre qui compare successivement `lit` à `pal`, `ale`, `leo`, `eol`, `oli`, `lit`.

Question 7. *Fenêtre différente*

Implémenter une fonction `difference(s,txt,i)` qui prend en entrée :

- une chaîne de caractère `s` de longueur `n`,
- un seconde chaîne de caractère `txt`,
- un indice `i` valide de `txt` tel que la fenêtre de longueur `n` et d'indice `i` dans `txt` ne dépasse pas la longueur de `txt` ;

et qui renvoie l'indice, dans `s`, du *dernier* caractère de `s` qui ne correspond pas à la chaîne de la fenêtre. La fonction renvoie `-1` si `s` est égale à la chaîne de la fenêtre.

Par exemple `difference("abus","meticuleusement",6)` renvoie `1` car le `b` de `abus` est le dernier caractère différent dans la fenêtre `leus`.

Question 8. *Fenêtre glissante*

Ecrire une fonction `rglissante(s,txt)` qui implémente l'algorithme de le fenêtre glissante et renvoie l'indice de la première fenêtre contenant `s` dans `txt` si elle existe, `-1` sinon. Par exemple `rglissante("lit","paleolithiq` renvoie `5`.

On utilisera, obligatoirement, la fonction `difference` implémentée précédemment.

Question 9. *Fenêtre hachée*

L'algorithme de Rabin-Karp utilise le haché de la chaîne `s` et celui de la fenêtre courante de recherche pour améliorer les performances de la comparaison. Deux chaînes ayant des hachés différents sont forcément différentes, par contre deux chaînes ayant le même haché ne sont pas forcément égales, comme on l'a vu précédemment. L'algorithme procède donc à une vérification à l'aide de la fonction `difference` si le haché de `s` est égal au haché de la chaîne de la fenêtre courante.

```
def rabin_karp(s,txt) :
    n, m = len(s), len(txt)
    hs, hf = hsomme(s), hsomme(txt[0:n])
    for i in range(# Trou 1 ) :
        if hs == hf :
            if # Trou 2 :
                return i
        if i+n < m :
            hf = # Trou 3
    return -1
```

Compléter les trois trous de l'algorithme de Rabin-Karp pour qu'il renvoie l'indice de la première fenêtre contenant `s` dans `txt` si elle existe, `-1` sinon.

IV Mot en 5 lettres

Question 10. *Saut dans un motif*

Soit `s = "lit"` et `txt = "paleolithique"` . Prenons la fenêtre d'indice `2` et de longueur 3 : `leo` . Expliquer en quoi il est possible, en regardant seulement la fenêtre `2`, de passer directement à la recherche dans la fenêtre `5`, sans vérifier la fenêtre `3` et `4`.

Soit `s = "polit"` et `txt = "paleolithique"` donner, pour les fenêtres suivantes, la première fenêtre suivante utile à vérifier :

- fenêtre `i = 1` : `aleol`
- fenêtre `i = 5` : `lithi`
- fenêtre `i = 6` : `ithiq`

Question 11. *Table de saut du motif - Exemple*

Soit `s` une chaîne de caractère de taille `n`. On note `s' = s[:n-1]`, la chaîne `s` privée de son dernier caractère.

La table de saut d'une chaîne de caractère `s` de longueur `n` est une liste de 26 cases tel que pour chaque caractère alphabétique minuscule `c`, la case `hord(c)-1` de la liste soit égale à :

- `n-p-1` avec `p` l'indice de la dernière occurrence de `c` dans `s'` si `c` est dans `s'`.
- `n` si `c` n'est pas dans `s'`.

Donner la table de saut de `"denadine"`.

Question 12. *Table de saut du motif*

Implémenter une fonction `ts_motif(s)` qui prend en entrée une chaîne de caractère et renvoie la table de saut de `s`.

Question 13. *Recherche de motif*

L'algorithme d'Horspool propose d'exploiter la table de saut dans la recherche pour *sauter* des fenêtres de recherche et donc gagner en performance.

Dans l'algorithme de la fenêtre glissante, si `s` ne correspond pas à la fenêtre de recherche courante, d'indice `i`, l'algorithme décale la fenêtre d'une case : on passe à la fenêtre d'indice `i+1`.

Dans le même cas, l'algorithme d'Horspool propose de regarder le dernier caractère de la fenêtre, `txt[i+n-1]` et de décaler la fenêtre suivant la table de saut, *i.e.* de la décaler de `ts[hord(txt[i+n-1])-1]` avec `ts` la table de saut de `s`.

Ecrire une fonction `horspool(s,txt)` qui implémente l'algorithme d'Horspool et renvoie l'indice de la première fenêtre contenant `s` dans `txt` si elle existe, `-1` sinon.