TRAVAUX PRATIQUES 2

RECHERCHES DANS UNE CHAÎNE

Voici les instructions éventuellement utiles dans ce TP :

`len(S)`	donne la longueur de la chaîne de caractères `S`
`S = ''`	`S` est la chaîne vide
`S[i]`	donne le caractère d'indice `i` dans la chaîne
`S[i:j]`	donne la sous-chaîne des caractères d'indice `i` inclus à `j` exclus
`S+T`	concatène (juxtapose) les chaînes `S` et `T`
`len(L)`	donne la longueur de `L`
`L=[]`	`L` est la liste vide
`L[i]`	donne la valeur de l'élément de la liste d'indice `i`
`L+M`	concatène (juxtapose) les listes `L` et `M`
`L.append(a)`	ajoute `a` à la fin de la liste
`L.insert(i,a)`	ajoute l'élément `a` à la i^e position
`del L[i]`	supprime et retourne le i^e élément
`for k in range(i,j)`	boucle **pour** `k` allant **de** `i` (inclus) **à** `j` (exclus)
`while`	boucle **tant que**

1. **Rechercher un caractère dans une chaîne.**

   Ecrire une fonction `cherche(c,S)` qui prend en arguments un caractère `c` et une chaîne de caractères `S` ; si le caractère est présent dans la chaîne, cette fonction revoie `True` et la liste des indices de ce caractère ; sinon la fonction revoie False.

   Par exemple, si on considère la chaîne `S='je vois et je comprends'`, alors :

   `chercher('e',S)` → `(True,[1,8,12,19])` ; `chercher('a',S)` → `(False,[])`

2. **Rechercher un mot dans un texte.**

   Soit une chaîne de caractères `S='TGACTGGTCACT'`, on appelle sous-chaîne de caractères de `S` une suite de caractères incluse dans `S`. Par exemple, `'TGG'` est une sous-chaîne de `S` mais `'TAG'` n'est pas une sous-chaîne de `S`.

   L'objectif est de rechercher une sous-chaîne de caractères `M` de longueur m appelée motif dans une chaîne de caractères `S` de longueur n.

   Il s'agit d'une problématique classique en informatique, qui répond aux besoins de nombreuses applications. On trouve plus de 100 algorithmes différents pour cette même tâche, les plus célèbres datant des années 1970 ; mais plus de la moitié ont moins de 10 ans.

   Principe de l'algorithme naïf : On parcourt la chaîne. À chaque étape, on regarde si on a trouvé le bon motif. Si ce n'est pas le cas, on recommence avec l'élément suivant de la chaîne de caractères.

   Cet algorithme a une complexité en $O(nm)$ avec n, la taille de la chaîne de caractère et m, la taille du motif.

   Écrire une fonction `rechercher(M,S)` qui à une sous-chaîne de caractères `M` et une chaîne de caractères `S` renvoie `False` si `M` n'est pas dans `S`, et `True` et la position de la première lettre de la chaîne de caractères `M` si `M` est présente dans `S`. Par exemple :

   `rechercher('TGG',S)` → `(True,4)` ; `rechercher('TAG',S)` → `False`