

Master 1, Bio Informatique :
TD 5 du 3 mars : Automates et recherche de facteurs et sous-mots.

Exercice 1

En Python un mot peut être considéré comme une liste de lettres par exemple si on écrit :

```
u = "abbabba"  
print u[0], " ", u[3]
```

On voit s'afficher deux fois **a** car la première lettre (`u[0]`) et la quatrième lettre (`u[3]`) de `u` sont des **a**.

Un exemple d'algorithme sur les mots est le test de l'égalité, écrit ci-dessous :

```
def sontEgaux(u,v):  
    n = len(u)  
    if len(v) != n:  
        return False  
    else :  
        for i in range (n):  
            if u[i] != v[i] :  
                return False  
    return True
```

Question 1. Ecrire la fonction, implantant l'algorithme vu en cours qui teste si le mot `u` est sous-mot du mot `v`; elle donnera pour valeur `True` si tel est le cas et `False` sinon.

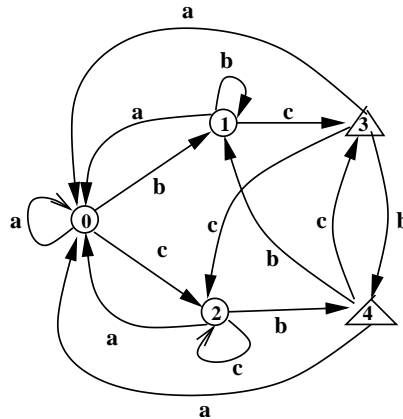
```
def estSousMot (u,v):
```

Question 2. Même question pour la fonction qui teste si le mot `u` est facteur du mot `v` :

Exercice 2

Un automate sera représenté en Python de façon telle que les états sont des entiers et la table de transition par un dictionnaire dans lequel les clés sont les lettres de l'alphabet et les valeurs sont des listes d'entiers représentant les actions de la lettre sur les états. Dans la liste correspondant la lettre \hat{x} on trouve en position `i` l'état extrémité de l'arc d'origine `i` et d'étiquette `x`.

Les états terminaux seront donnés par une liste. Par exemple considérons l'automate dessiné ci dessous



L'action de la lettre **a** est telle qu'elle fait passer tous les états 0, 1, 2, 3, 4 à l'état 0, ainsi la valeur associée à cette lettre est la liste [0,0,0,0,0]. La lettre **b** fait passer des états 0, 1, 2, 3, 4 aux états 1, 1, 4, 4, 1 respectivement la liste associée à **b** est donc [1,1,4,4,1]. Plus précisément le dictionnaire et la liste qui représentent l'automate sont :

```

autom = {'a': [0,0,0,0,0],
         'b': [1,1,4,4,1],
         'c': [2,3,2,2,3]}
terminaux = [3,4]

```

Question 1. Décrire le dictionnaire et liste représentant l'automate vu en cours et reconnaissant les mots qui contiennent le facteur **abaabab**.

Question 2. Ecrire la fonction

```
def table(autom, x, s):
```

qui étant donné le dictionnaire **autom** représentant un automate, une lettre **x** et un état représenté par son numéro **s** donne pour résultat l'état extrémité de l'arc d'origine **s** et d'étiquette **x**

Question 3. Ecrire la fonction :

```
def reconnu(autom, terminaux, f):
```

qui étant donné le dictionnaire **autom** représentant un automate, et un mot **f** calcule l'état extrémité du chemin d'origine l'état 0 et dont la suite des étiquettes des arcs forme le mot **f**. Puis donne pour résultat **True** ou **False** suivant que cet état soit ou non terminal.

Question 4.

Construire sur papier un automate sur l'alphabet $\Sigma = \{A, C, G, T\}$ qui reconnaît si un mot contient le facteur **CGAGAGAG** entrer cet automate en Python et vérifier sur quelques exemples le résultat.

Exercice 3

On se propose ici de donner un algorithme qui détermine la longueur du plus long sous-mot de deux mots. On note dans la suite $MSM(u, v)$ cette longueur.

Question 1. Supposant que l'on connaisse $MSM(u, v)$ quelle est la valeur de $MSM(ua, va)$ où a est une lettre ?

Question 2. Soit a et b deux lettres différentes. Donner une formule qui détermine la valeur de $MSM(ua, vb)$ en fonction de $MSM(ua, v)$ et $MSM(u, vb)$

Pour deux mots u, v de longueurs respectives n et m on construit un tableau `msm` tel que pour $0 \leq i < n$ et $0 \leq j < m$ `msm[i][j]` contient la longueur du plus long sous-mot commun à $u_0u_1 \dots u_i$ et $v_0v_1 \dots v_j$

Question 3.

Ecrire en Python les instructions qui calculent la première ligne `msm[0]` de ce tableau.

Question 4. Ecrire en Python les instructions qui calculent la i -ème ligne `msm[i]` de ce tableau en connaissant la $(i - 1)$ -ème ligne. En déduire une fonction qui calcule tout le tableau.

Question 5. Comment déterminer le plus long sous-mot commun en ayant calculé le tableau `msm` ?