TD ITC^2 N° 8: PROGRAMMATION DYNAMIQUE:
PLUS LONGUE SOUS-SÉQUENCE COMMUNE PAR DIVERSES APPROCHES

On considère une suite finie de symboles pris dans un ensemble fini que l'on appellera **séquence**, par exemple des lettres prises dans l'alphabet. On appellera $X = (x_1 x_2 \dots x_n)$ cette séquence comportant n symboles. Toute séquence $X' = (x'_1 x'_2 \dots x'_p)$ est dite une **sous-séquence de** X si elle est composée de symboles de la séquence X pris dans l'ordre d'apparition dans X, mais non nécessairement contigus dans X.

On se propose de construire un algorithme performant capable de déterminer la plus longue sous-séquence commune (PLSSC) à deux séquences (non nécessairement de même longueur) $X = (x_1 x_2 \dots x_n)$ de longueur n et $Y = (y_1 y_2 \dots y_m)$ de longueur m.

On appellera $L(i, j)$ **la longueur de la plus longue sous-séquence commune** aux sous-séquences $X_i = (x_1 x_2 \dots x_i)$ et $Y_j = (y_1 y_2 \dots y_j)$ avec $0 \leq i \leq n$ et $0 \leq j \leq m$.

Important: dans toute la suite, les séquences (et sous-séquences) considérées seront des chaines de caractères (`str`) (composées de lettres majuscules dans les exemples concrets proposés).

1 Approche par force brute

1. Proposer une fonction `verif(Yk:str,Y:str)` permettant de vérifier si Yk est une sous séquence de Y. **Cette fonction devra obligatoirement posséder une complexité en** $O(m)$.
2. Déterminer le nombre de sous-séquences de X. En déduire en fonction de m et n la complexité qu'aurait une fonction générant toutes les sous-séquences de X, utilisant pour chacune la fonction `verif(Yk:str,Y:str)`, et renvoyant finalement la plus longue sous-séquence commune. Conclure.

2 Approche récursive

On admettra le théorème suivant (assez évident!):

Posons que $Z =$ "$z_1 z_2 \dots z_k$" est une plus longue sous séquence commune à X et Y. On a alors:

- *Si $x_n = y_m$ alors $z_k = x_n (= y_m)$ et Z_{k-1} est une plus longue sous séquence commune à X_{n-1} et Y_{m-1}*
- *Si $x_n \neq y_m$ alors si $z_k \neq x_n$ on a Z qui est une plus longue sous séquence commune à X_{n-1} et Y*
- *Si $x_n \neq y_m$ alors si $z_k \neq y_m$ on a Z qui est une plus longue sous séquence commune à X et Y_{m-1}*

On pourra utiliser pour la suite la fonction Python `max(a:int,b:int)` renvoyant le plus grand des deux entiers a et b.

3. Que vaut $L(i, j)$ pour $i = 0$ ou $j = 0$, c'est à dire pour une séquence X ou Y vide? Déterminer ensuite pour $(i, j) > (0, 0)$ les deux possibilités de récurrence définissant $L(i, j)$ en fonction de $L(i-1, j-1)$, $L(i-1, j)$, $L(i, j-1)$ suivant que $x_i = y_j$ ou bien $x_i \neq y_j$.
4. En déduire une fonction récursive `Long_PLSSC_rec(X:str,Y:str)` $\rightarrow$ `int` recevant les deux chaines de caractères X et Y, et renvoyant la longueur de la plus longue sous-séquence commune entre X et Y.
5. Déterminer dans le pire des cas, c'est à dire lorsque X et Y n'ont aucun élément en commun, la complexité de la fonction `PLSSC_rec`.

3 Approche par programmation dynamique

6. Expliquer en quoi le problème est éligible à la programmation dynamique.

   On propose de présenter le problème sous la forme d'un tableau L de dimensions $(n+1, m+1)$ dans lequel $L[i, j]$ représente la longueur de la plus longue sous-séquence commune aux deux sous-séquences $X[0:i]$ et $Y[0:j]$.

7. Compléter le tableau suivant pour $X =$ "$ABCBA$" et $Y =$ "$BACBDA$":

	long. séquence Y →	0	1	2	3	4	5	6
long. séquence X ↓	séquence X↓ / séquence Y→		B	A	C	B	D	A
0			**0**	**0**	**0**	**0**	**0**	**0**
1	A	**0**						
2	B	**0**						
3	C	**0**						
4	B	**0**						
5	A	**0**						

8. Proposer une fonction `Long_PLSSC_dyn(X:str,Y:str)` $\rightarrow$ `int` renvoyant la longueur de la plus longue sous-séquence commune entre les deux chaines de caractères X et Y en exploitant une démarche dynamique de type `BOTTOM-UP`.
9. Déterminer la complexité de cette dernière fonction.

   On souhaite désormais modifier le code précédent afin qu'il renvoie **la plus longue sous-séquence commune** en plus d'en donner sa longueur.

10. A partir du tableau renseigné en question **7** (et qui peut également être renvoyé par `Long_PLSSC_dyn(X,Y)` en changeant sa dernière ligne en `return L`) , proposer une stratégie de reconstruction de la plus longue sous-séquence commune.
11. Proposer une fonction `PLSCC_dyn(X:str,Y:str)` $\rightarrow$ `str` renvoyant la plus longue sous-séquence commune aux deux chaines de caractères X et Y.