Correction DS1 PCSI Python

Question de cours

```
def dicho(L,x):
    g=0
    d=len(L)-1
    while g<=d:
        m=(g+d)//2
        if L[m]<x:
            g=m+1
        elif L[m]>x:
            d=m-1
        else:
            return True
    return False
```

Problème

Partie I

1)`seq[4]` renvoie l'élément d'indice 4 de la liste `seq`, c'est-à-dire `'T'`, puisqu'on commence l'indexation à partir de 0.

`seq[-1]` renvoie le dernier élément de la liste, c'est-à-dire `'G'`.

`seq[2:6]` renvoie la sous-liste de `seq` formés des éléments dont les indices sont compris entre 2 et 5, c'est-à-dire : `['C','G','T','A']`.

2)Programme :

```
import random as rd
def generation(n):
    S=[]
    for k in range(n):
        r=rd.randint(1,4)
        if r==1:
            S.append('A')
        elif r==2:
            S.append('C')
        elif r==3:
            S.append('G')
        else:
            S.append('T')
    return S
```

3)a)Avec la boucle `while`, on parcourt les éléments de la liste en les répérant grâce à leur indice. Dans le premier test, la variable a, initialement nulle

augmente de 1 quand l'élément en cours est un `'A'`.
En sortie de boucle, a compte donc le nombres d'occurences de `'A'` dans la séquence.
De même, en sortie de boucle, b, c et d comptent respectivement le nombre d'occurences de `'C'`, `'G'` et `'T'`.
Puis, on retourne les pourcentages demandés avec la commande :

```
return [a*100/n,b*100/n,c*100/n,d*100/n]
```

b)A chaque itération de boucle, un seul test est effectué, dans lequel il y a 2 additions.
Il y a n itérations donc 2n additions en comptant tous les tours de boucle.

Partie II

4)a)Programme :

```
def recherche(M,S):
    m=len(M)
    n=len(S)
    for i in range(n-m+1):
        c,j=0,0
        while j<m:
            if S[i+j]==M[j]:
                c=c+1
                j=j+1
            else:
                j=m
        if c==m:
            return i
    return -1
```

> Remarque
> La borne `n-m+1` garantit que `S[i+j]` reste dans la liste : le plus grand indice i testé est $n-m$, et alors $i+j \leq n-1$.

b)Imaginons qu'on supprime le test `else`.
Si le motif M n'apparaît pas dans S, alors la condition `S[i+j]==M[j]` est réalisée moins de m fois et j ne pourra jamais atteindre la valeur m. La boucle tournera alors à l'infini.

5)Ici, $n = 3 \times 10^{10}$ et $m = 50$.
Le nombre d'opérations élémentaires vaut approximativement :
$3mn = 3 \times 50 \times 3 \times 10^{10} = 45 \times 10^{11}$.
Le temps nécessaire est : $\dfrac{4.5 \times 10^{11}}{10^{10}} = 45$ secondes.

6)Les préfixes sont : `['A']`, `['A','C']` et `['A','C','G']`.

Les suffixes sont : `['A']`, `['G','A']` et `['C','G','A']`.

7)`['A','C','G','T','A','C']` ne possède qu'une seule bordure : `['A','C']`.
`['A','C','A','C','A']` possède deux bordures : `['A']` et `['A','C','A']`.

8)Tableau :

boucle	i	j	F
$\times\times\times$	1	0	$[0]$
1	2	0	$[0,0]$
2	3	1	$[0,0,1]$
3	3	0	$[0,0,1]$
4	4	1	$[0,0,1,1]$
5	5	2	$[0,0,1,1,2]$
6	6	3	$[0,0,1,1,2,3]$

9) On lit `F[5]=3`. La plus longue bordure de S est bien de longueur 3 : c'est `['A','C','A']`, qui est à la fois le préfixe `S[0:3]` et le suffixe `S[3:6]`. Il n'y en a pas de plus longue : `['A','C','A','A']` $\neq$ `['A','A','C','A']` et `['A','C','A','A','C']` $\neq$ `['C','A','A','C','A']`.

10) Étudions l'évolution de $2i-j$ à chaque itération.

- si `S[j]==S[i]`, i et j augmentent de 1 donc $2i-j$ augmente de 1.
- si `S[j]!=S[i]` et $j=0$, seul i augmente de 1 donc $2i-j$ augmente de 2.
- si `S[j]!=S[i]` et $j>0$, i ne change pas et j est remplacé par `F[j-1]`. C'est la longueur d'une bordure de `S[0:j]` donc `F[j-1]` $< j$. Ainsi j diminue strictement et $2i-j$ augmente d'au moins 1.

Dans tous les cas, $2i-j$ est un entier qui augmente d'au moins 1 à chaque itération. Or $i \leqslant n$ et $j \geqslant 0$, donc $2i-j \leqslant 2n$. Une suite d'entiers strictement croissante et majorée est finie : la boucle `while` se termine.

11) Au départ, `F` contient un élément et $i=1$. À chaque fois que i augmente de 1, on ajoute exactement un élément à `F` (dans le premier et le troisième cas) ; quand i ne change pas (deuxième cas), `F` ne change pas non plus. Ainsi `len(F)`$=i$ à chaque début d'itération.
La boucle s'arrête quand $i=n$: `F` contient alors n éléments, comme S.