DS n° 02 – Concours blanc

I Exercice de cours – Classement d'une liste

On considère une liste de valeurs flottantes non classées. La liste des valeurs est déjà définie dans le programme et est stockée dans une variable appelée `tab`.
Écrire en langage Python un algorithme qui permet d'obtenir à la fin une nouvelle liste des valeurs classées par ordre croissant, stockée dans une variable appelée `tab2`.
Le programme suivant qui permet de classer une liste.

```
tab2=[]
for i in range(1,len(tab)-1):
    min=i
    for j in range(i+1,len(tab)):
        if tab[j] < tab[min]:
            min=j
    tmp=tab[i]
    tab[i]=tab[min]
    tab[min]=tmp
    tab2.append(tab[min])
```

II Exercice de TP – Recherche d'un mot dans une chaîne de caractères

1.
```
def estIci(motif,texte,i):
    k = 0
    p = len(motif)
    # on parcourt le texte à partir de l'index i
    # tant que on ne dépasse pas la longueur de motif
    # et que texte et motif sont identiques
    while k<p and motif[k] == texte[k+i]:
        k = k+1
    # si on a parcouru p termes, alors, motif est dans texte à l'index i
    if k==p:
        resultat=True
    else:
        resultat=False
    return resultat
```

2. Le pire des cas s'obtient avec un texte type : `texte='aaaaa...aaaaa'` et un motif du type : `'aa..aab'`. On note n la longueur du texte et p celle du motif.
Compté grossièrement : dans `estIci(motif,texte,i)`, on effectue p tests. On répète $n-p$ fois cette opération dans `recherche(motif,texte)`. Donc on s'attend à une complexité en : $\boxed{O((n-p)p).}$
On obtient $O(np)$ si $n >> p$

   Plus précisément, avec le pire des cas :
   dans `estIci(motif,texte,i)`, on a :
   — 2 affectations avant la boucle
   — on répète p fois :
      — 2 tests
      — 1 affectation
      — 1 addition
   — 1 test
   — 1 affectation

   $\boxed{\text{Au total, on a comme nombres d'opérations : } 2+4p+2=4p+4=O(p).}$

   `recherche(motif,texte)`, on a :
   — 2 affectations

— dans la boucle else : 2 affectations
— on répète $n - p$ fois :
  — 2 tests
  — 1 affectation
  — 1 calcul de `estIci`
  — 1 afffectation
  — 1 addition

Au total, on a comme nombres d'opérations :

$$\boxed{2 + 2 + (n - p) \times (5 + 4p + 4) = 4 + (n - p)(4p + 9) = O((n - p)p).}$$

III Exercice – Croissance d'une suite

```
n=len(u)

i=0
while i<=n-2 and u[i]<u[i+1]:
    i=i+1

if i==len(u)-1:
    print 'u est strictement croissante'
else:
    print 'a partir du rang',i,'la suite n est plus strictement croissante'
```