TP 03

Recherche séquentielle dans un tableau unidimensionnel. Algorithmes opérant sur une structure séquentielle par boucles imbriquées.

Savoirs et compétences :

- *Th. 2 : Algorithmes opérant sur une structure séquentielle par boucles imbriquées.*

Proposition de corrigé

Activité 1 –

Question 1 *Écrire une fonction* `Nb_Ventes(ventes:dict)` *qui prend en entrée un dictionnaire et renvoie le nombre total de ventes dans la société.*

```
def Nb_Ventes(ventes):
    """Renvoie le nombre total de ventes dans la societe"""
    S=0
    for elt in ventes.items():
        S+=elt[1]
    return S
```

Question 2 *Écrire une fonction* `Nom_vendeur(ventes:dict)` *qui prend en entrée un dictionnaire et renvoie le nom du vendeur ayant réalisé le plus de ventes. Si plusieurs vendeurs sont ex-aequo, la fonction devra retourner le nom de l'un d'entre eux seulement.*

```
def Nom_vendeur(ventes):
    """Renvoie le nombre total de ventes dans la societe"""
    max=0
    nom=''
    for elt in ventes.items():
        if elt[1]>max:
            max=elt[1]
            nom=elt[0]
    return nom
```

Activité 2 – Recherche dans un tableau

Question 3 *On appelle distance minimale, la distance entre deux éléments les plus proches (éventuellement égaux) et d'indices distincts. Écrire une fonction* `distance_min(L)` *qui renvoie la distance minimale de* `L`.

```
def distance_min(L): # On cherche min |Ti-tj| pour i<j
    n=len(L)
    min=abs(L[1]-L[0])
    for i in range(n):
        for j in range(i+1,n):
            if abs(L[j]-L[i])<min:
                min=abs(L[j]-L[i])
    return(min)
```

Question 4 *Écrire une fonction* `indices_distance_min2(L)` *qui renvoie un couple d'indices réalisant la distance minimale*

```
def indices_distance_min2(L): # On cherche min |Ti-tj| pour i<j
    n=len(L)
    min=abs(L[1]-L[0])
    p,q=0,1
    for i in range(n):
        for j in range(i+1,n):
            if abs(L[j]-L[i])<min:
                min=abs(L[j]-L[i])
                p,q=i,j
    return p,q
```

Question 5 *Écrire une fonction* `indices_distance_min3(L)` *qui, étant donnée une liste* `L`*, réalise ces opérations et renvoie un couple solution*

```
 def indices_distance_min3(L):
    D={}
    n=len(L)
    for i in range(n-1):
        m=abs(L[i+1]-L[i])
        j=i+1
        for k in range(i+1,n):
            if abs(L[k]-L[i])<m:
                m=abs(L[k]-L[i])
                j=k
        D[str(i)]=[j,m]
    p,q=0,D["0"][0]
    min=D["0"][1]
    for elt in D.items():
    # elt de la forme ["3",[4,min {|Lk-L3|, k>4}]
        if elt[1][1]<min:
            p,q=int(elt[0]),elt[1][0]
            min=elt[1][1]
    return(p,q)
```

Activité 3 – Recherche d'un mot dans un texte

Question 6 *Écrire une fonction* `est_ici(texte,motif,i)` *qui, étant données deux chaines de caractères* `texte`*,* `motif` *et un indice* `i`*, renvoie* `True` *ou* `False` *selon que* `motif` *est ou n'est pas dans* `texte` *au rang* `i`*. On utilisera une boucle* `while`*.*

```
def est_ici(texte, motif, i):
    p=len(motif)
    j=0
    while j<=p-1 and motif[j]==texte[i+j]:
            j=j+1
    return(j==p)
```

Question 7 *Écrire une fonction* `est_sous_mot(texte,motif)` *qui renvoie* `True` *ou* `False` *selon que* `motif` *est dans* `texte` *ou pas. On utilisera une boucle* `while`*.*

```
def est_sous_mot(texte, motif):
    n,p=len(texte), len(motif)
    i=0
    while i<=n-p and not est_ici(texte, motif, i):
        i=i+1
    return(i<=n-p)
```

Question 8 *Écrire une fonction* `position_sous_mot(texte,motif)` *qui renvoie la liste de toutes les occurences de l'indice de position de la première lettre du mot* `motif` *dans* `texte`*. On utilisera une boucle* `for`*.*

```
def position_sous_mot(texte, motif):
    n,p=len(texte), len(motif)
    L=[]
    for i in range(n-p+1):
        if est_ici(texte, motif, i):
            L.append(i)
    return(L)
```

Activité 4 – Tri à bulles