N. S. I. Term. - DS07 (vendredi 11 février 2022) – 2 heures

Exercice 1. Sujet 1 – Exercice 1

Programmer la fonction `recherche`, prenant en paramètre un tableau non vide `tab` (type `list`) d'entiers et un entier `n`, et qui renvoie l'indice de la **dernière** occurrence de l'élément cherché. Si l'élément n'est pas présent, la fonction renvoie la longueur du tableau.

Exemples :

```
>>> recherche([5, 3],1)
2
>>> recherche([2,4],2)
0
>>> recherche([2,3,5,2,4],2)
3
```

Exercice 2. Sujet 2 – Exercice 1

Programmer la fonction `moyenne` prenant en paramètre un tableau d'entiers `tab` (type `list`) qui renvoie la moyenne de ses éléments si le tableau est non vide et affiche `'erreur'` si le tableau est vide.

Exemples :

```
>>> moyenne([5,3,8])
5.333333333333333
>>> moyenne([1,2,3,4,5,6,7,8,9,10])
5.5
>>> moyenne([])
'erreur'
```

Exercice 3. Sujet 10 – Exercice 1

Écrire une fonction `maxi` qui prend en paramètre une liste `tab` de nombres entiers et renvoie un couple donnant le plus grand élément de cette liste, ainsi que l'indice de la première apparition de ce maximum dans la liste.

Exemple :

```
>>> maxi([1,5,6,9,1,2,3,7,9,8])
(9,3)
```

Exercice 4. Sujet 13 – Exercice 1

1) Préliminaire :

On considère la liste $L = [1, 52,\ 6, -9,\ 12]$, donner les différents états de la liste, à la fin de chaque étape de l'application de l'algorithme de tri-sélection.

2) Exercice à traiter

Écrire une fonction `tri_selection` qui prend en paramètre une liste `tab` de nombres entiers et qui renvoie le tableau trié par ordre croissant.

On utilisera l'algorithme suivant :

- on recherche le plus petit élément du tableau, et on l'échange avec l'élément d'indice 0 ;
- on recherche le second plus petit élément du tableau, et on l'échange avec l'élément d'indice 1 ;
- on continue de cette façon jusqu'à ce que le tableau soit entièrement trié.

Exemple :

```
>>> tri_selection([1,52,6,-9,12])
[-9, 1, 6, 12, 52]
```

Exercice 5. Sujet 5 – Exercice 2

1) Préliminaire :

On considère la liste $L = [2, 5, -1, 13,\ 0,\ 28]$, donner les différents états de la liste, à la fin de chaque étape de l'application de l'algorithme de tri-insertion.

2) Exercice à traiter

La fonction `tri_insertion` suivante prend en argument une liste `L` et trie cette liste en utilisant la méthode du tri par insertion. Compléter cette fonction pour qu'elle réponde à la spécification demandée.

```
def tri_insertion(L):
    n = len(L)

    # cas du tableau vide
    if ...:
        return L

    for j in range(1,n):
        e = L[j]
        i = j

    # A l'étape j, le sous-tableau L[0,j-1] est trié
    # et on insère L[j] dans ce sous-tableau en déterminant

    # le plus petit i tel que 0 <= i <= j et L[i-1] > L[j].
        while  i > 0 and L[i-1] > ...:
            i = ...

        # si i != j, on décale le sous tableau L[i,j-1] d'un cran
        # vers la droite et on place L[j] en position i
        if i != j:
            for k in range(j,i,...):
                L[k] = L[...]
            L[i] = ...
    return L
```

Exemples :

```
>>> tri_insertion([2,5,-1,7,0,28])
[-1, 0, 2, 5, 7, 28]
>>> tri_insertion([10,9,8,7,6,5,4,3,2,1,0])
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Exercice 6. Sujet 23 – Exercice 2

1) Préliminaire :

On considère les listes triées, $L1 = [1, 6, 10]$ et $L2 = [0, 7, 8, 9]$. Donner les différents états de la liste, $L12$, à la fin de chaque étape de l'application de l'algorithme fusion de deux listes triées aux listes $L1$ et $L2$ (il s'agit de donner l'état de la liste $L12$ à la fin de chaque itération de chaque boucle `while` de l'algorithme à compléter donné ci-après).

2) Exercice à traiter

La fonction `fusion` prend deux listes `L1,L2` d'entiers triées par ordre croissant et les fusionne en une liste triée `L12` qu'elle renvoie.

Le code Python de la fonction est

```
def fusion(L1,L2):
    n1 = len(L1)
    n2 = len(L2)
    L12 = [0]*(n1+n2)
    i1 = 0
    i2 = 0
    i = 0
    while i1 < n1 and ... :
        if L1[i1] < L2[i2]:
            L12[i] = ...
            i1 = ...
        else:
            L12[i] = L2[i2]

            i2 = ...
        i += 1
    while i1 < n1:
      L12[i] = ...
      i1 = i1 + 1
      i = ...
    while i2 < n2:
      L12[i] = ...
      i2 = i2 + 1
      i = ...
    return L12
```

Compléter le code.

Exemple :

```
>>> fusion([1,6,10],[0,7,8,9])
[0, 1, 6, 7, 8, 9, 10]
```

3) Donner les étapes du tri-fusion pour la liste $L = [9, 8, 12, 3, 5, 14, 6, 8]$, sous la forme de deux arbres, le premier représentant les étapes de scission de liste, le deuxième les étapes de fusion de listes triées.

4) Donner une implémentation récursive de l'algorithme de tri-fusion de deux listes, utilisant la fonction fusion précédente, sous la forme d'une fonction `fusion` prenant en argument une liste d'entiers.
   On prendra soin de distinguer le traitement des cas de base (ne nécessitant pas d'appel récursif) et le traitement du cas général (nécessitant un ou plusieurs appels récursifs).