TP 03

Structure séquentielle par boucles imbriquées

Savoirs et compétences :

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

Consignes

1. Commencez la séance en créant un dossier au nom du TP dans le répertoire dédié à l'informatique de votre compte.
2. Après la séance, vous devez rédiger un compte-rendu de TP et l'envoyer au format électronique à votre enseignant.
3. Ayez toujours un crayon et un papier sous la main. Quand vous réfléchissez à une question, utilisez les !
4. Essayer d'être le plus autonome possible.
5. Ce TP est à faire en binôme, vous ne rendrez donc qu'un compte-rendu pour deux. Votre fichier portera un nom du type : `tp03_durif_kleim.py`, où les noms de vos enseignants sont à remplacer par ceux des membres du binôme. Le nom de ce fichier ne devra comporter ni espace, ni accent, ni apostrophe, ni majuscule. Dans ce fichier, vous respecterez les consignes suivantes.
   - Écrivez d'abord en commentaires (ligne débutant par #), le titre du TP, les noms et prénoms des étudiants du groupe.
   - Commencez chaque question par son numéro écrit en commentaires.
   - Les questions demandant une réponse écrite seront rédigées en commentaires.
   - Les questions demandant une réponse sous forme de fonction ou de script respecteront pointilleusement les noms de variables et de fonctions demandés.

Le but de ce TP est l'entrainement à l'enchaînement des boucles itératives et quelques notions de complexité.

Activité 1 – Recherche dans un tableau

1. On se donne un tableau `T` unidimensionnel. Écrire une fonction `distance_min1(T)` qui renvoie les deux éléments qui sont les plus proches ie dont la valeur absolue de la différence est minimale. On indiquera les valeurs obtenues ainsi que les indices correspondants.
2. Pour un tableau à n cases, montrer que le nombre de comparaisons $C(n)$ faites dans cette fonction est tel que la suite $\left(\dfrac{C(n)}{n^2}\right)$ est bornée : on dit que la **complexité est quadratique**.
3. On représentera un tableau bidimensionnel par une liste dont les éléments sont des listes représentants les lignes du tableau. `T` sera donc une matrice pas nécessairement carrée par exemple de la forme

   `T=[[1,2,3],[6,4,3],[3,8,9],[3,-2,0]]` :

   chaque élément de `T` désignera une ligne du tableau. Le nombre de ligne est le nombre d'éléments de `T`, le nombre de colonnes est le nombre d'éléments de `T[0]`.
   Écrire pour un tableau bidimensionnel `T` une fonction `distance_min2(T)` qui renvoie les deux éléments qui sont les plus proches ie dont la valeur absolue de la différence est minimale. On indiquera les valeurs obtenues ainsi que les indices correspondants.

Activité 2 – Recherche d'un mot dans un texte

Question 1 *Écrire une fonction* `est_ici(texte,motif)` *qui, étant données deux chaines de caractères* `texte` *et* `motif`, *renvoie* `True` *ou* `False` *selon que mot est ou n'est pas dans texte. On n'utilisera pas de slicing mais on fera deux méthodes, l'une sans booléen explicite, l'autre avec.*

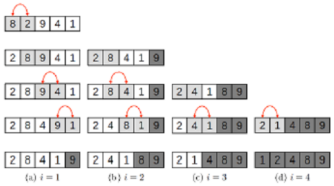
Question 2 *Écrire une fonction* `est_sous_mot(texte,motif)` *qui renvoie* `True` *ou* `False` *selon que motif est dans texte ou pas.*

Question 3 *Écrire une fonction* `position_sous_mot(texte,motif)` *qui renvoie la liste de toutes les occurences de l'indice de position du mot* `motif` *dans* `texte`.

Activité 3 – Tri à bulles

Le tri à bulles est un algorithme de tri classique. Son principe est simple, et il est très facile à implémenter. On considère un tableau de nombres T, de taille N. L'algorithme parcourt le tableau, et dès que deux éléments consécutifs ne sont pas ordonnés, les échange. Après un premier passage, on voit que le plus grand élément se situe bien en dernière position. On peut donc recommencer un tel passage, en s'arrêtant à l'avant-dernier élément, et ainsi de suite.

Au i-ème passage on fait remonter le i-ème plus grand élément du tableau à sa position définitive, un peu à la manière de bulles qu'on ferait remonter à la surface d'un liquide, d'où le nom d'algorithme de tri à bulles.



Question 4 *Appliquer l'algorithme de tri à Bulles «à la main» au tableau ci-dessous :*

2	1	6	9	8	4

Question 5 *Écrire une fonction* `est_trie(T)` *qui renvoie* `True` *ou* `False` *selon que le tableau* `T` *est trié ou pas.*

Question 6 *Écrire une fonction* `TriBulles(T)` *triant le tableau* `T` *par l'algorithme de tri à bulles*[1]

Question 7 *Vérifier que la fonction ci-dessus est correcte en utilisant la fonction* `EstTrie` *sur des tableaux de nombres aléatoires. On rappelle que la bibliothèque random permet de créer des nombres aléatoires.* `random.randint(a,b)` *renvoie un entier aléatoire compris entre* a *et* b.

Question 8 *Montrer que la complexité est quadratique (notion définie dans I)*

Question 9 *Écrire une fonction améliorée* `TriBulles2(T)` *qui sort de la fonction dès que le tableau a été parcouru sans faire d'échange (auquel cas, il est trié et donc inutile de continuer).*

Question 10 *Une variante du tri à bulles est le tri cocktail : il consiste à changer de direction à chaque passage. Lors du premier parcours, on se déplace du début du tableau vers la fin. Puis lors du second parcours on part de la fin du tableau pour arriver au début. À la troisième itération on part du début et ainsi de suite ... C'est une légère amélioration car il permet non seulement aux plus grands éléments de migrer vers la fin de la série mais également aux plus petits éléments de migrer vers le début.*

Question 11 *Écrire une fonction* `TriCocktail(T)` *qui étant donné un tableau* `T` *le renvoie trié selon la méthode du tri cocktail.*

Question 12 *Justifier en quoi cette variante peut être intéressante selon le type de tableau à trier mais vérifier néanmoins que la complexité reste quadratique.*

1. cette fonction doit agir sur place concernant le tableau `T` ie doit le modifier par effet de bord.