TP 8 : méthodes de tri

Les méthodes de tri jouent un rôle essentiel en informatique pour organiser les données et optimiser leur traitement. Les algorithmes de tri permettent de réarranger les éléments d'une liste selon un ordre donné (croissant ou décroissant). Trois méthodes sont à connaître (parties III, IV et V).

I Modification d'un objet structuré par effet de bord

Sous Python, il est possible qu'une fonction agisse directement sur sa variable d'entrée si c'est un objet mutable [1].

■ **Exercice 1.** Tester les deux fonctions suivantes et vérifier si la variable d'entrée a été modifiée ou non par la fonction.

Sans effets de bord

```
def touchepas(L):
    M=L[:]
    M[0]=-15
    return M
```

```
>>L=[1,2,3]
>>L2 = touchepas(L)
```

Avec effets de bord

```
def touche(L):
    L[0]=-15
```

```
>>L=[1,2,3]
>> touche(L)
```

II Méthodes de tri basiques

■ **Exercice 2. Tri naïf**

1. Programmer une fonction `pos_min(L)` prenant en entrée une liste `L` de flottants et renvoyant en sortie l'indice dans `L` de son élément minimal.
2. En déduire une fonction `tri_basique(L)` prenant en entrée une liste `L` et renvoyant en sortie la liste `M` obtenue en rangeant les éléments de `L` dans l'ordre croissant.

■ **Exercice 3. Tri d'une liste avec répétitions.** *Pour ce tri, on suppose, quitte à d'abord calculer le maximum des éléments de la liste à trier, que les éléments de la liste à trier sont à valeurs dans* $\{1\dots N\}$, N *étant une constante connue.*

1. Écrire une fonction `multi(L,a)` prenant en entrée une liste `L`, un flottant `a` et renvoyant en sortie le nombre d'occurrences de `a` dans `L`. Par exemple, `multi([1,2,1,3,4,1,2],1)` devrait renvoyer `3`, et d'occurrences de `a` dans `L`. Par exemple, `multi([1,2,1,3,4,1,2],6)` devrait renvoyer `0`.
2. Déduire de cette dernière fonction une autre fonction `tri_bis(L)` de tri de liste de flottants.

III Tri à bulles (bubblesort)

Principe. Il repose sur le fait qu'une suite finie $(u_j)_{0\leq j\leq n-1}$ est triée si et seulement si :

$$\forall j \in \{0,\dots,n-2\} \qquad (P_j) : \quad u_j \leq u_{j+1}$$

Algorithme.

1. Rappel : parmi les objets structurés mutables on trouve les listes. Sont non mutables en revanche : tuples, chaîne de caractères.

M. Patel
2024-2025
Classes de B_2, Rennes

1. On effectue un parcours des éléments de la liste `L`, et chaque fois que l'on trouve lors de ce parcours deux termes consécutifs `L[j]`, `L[j+1]` ne vérifiant pas la propriété (P_j), on les permute. Cette dernière permutation sera réalisée avec la fonction auxiliaire `permute(L,j)`.
2. Au bout du premier parcours, le plus grand élément de la liste (au moins) est bien trié.

Question : expliquer pourquoi.

3. Ensuite, on en conclut deux choses :

   a) Si la liste est de longueur n, au bout d'un passage, on se retrouve à trier une liste de longueur $n-1$ à l'étape suivante.

   b) Au parcours suivant, il est inutile d'examiner la dernière paire puisque cette dernière n'aura pas à être permutée.

   c) En au plus n parcours, où n est la longueur de la liste, on est certain que la liste est triée.

■ Exercice 4.

1. Écrire une fonction `permute(L,i,j)` prenant en entrée une liste flottants `L`, deux entier `i,j` et qui renvoie en sortie la liste `M` obtenue en échangeant les positions des éléments `L[i]` et `L[j]`.
2. En déduire une fonction `bubbleosrt(L)` prenant en entrée une liste de flottants `L` et modifiant `L` par effet de bord et triée dans l'ordre croissant.

IV Tri insertion

C'est la méthode la plus intuitive de tri, puisque c'est celle que l'on utilise naturellement par exemple pour trier ses cartes dans une partie de belote ou de tarot. Pour autant, elle n'est pas la plus simple à coder.

Algorithme.

1. On suppose qu'à chaque étape de l'algorithme, les éléments situés à gauche d'un élément privilégié de la liste, appelé **pivot** de la liste (et qui sera défini ci-après) sont correctement triés. C'est ce pivot qui sera à insérer correctement pour réaliser le tri.
2. À la première étape, le pivot p est le premier élément de la liste.
3. À chaque étape, on compare le pivot p à l'élément z qui le **précède** dans la liste. Ensuite :
   — Si $p < z$, c'est que p est mal placé d'après le point `1.` : donc on le rétrograde dans la liste (à gauche donc) jusqu'à ce qu'il soit en bonne position dans la liste.
   — Sinon c'est que p est bien placé.
4. On répète le point **3.** (ce qui constitue une étape), mais avec pour pivot le terme w qui était à droite de p à l'étape précédente.
5. Comme la position du pivot augmente strictement à chaque étape, il se retrouve en dernière position au bout d'un nombre fini d'étapes. D'après **1.**, la liste est triée.

Questions.

1. Écrire une fonction `retrograde(L,i)` qui prend en entrée une liste `L` de flottants, un entier `i`, et qui renvoie en sortie la liste `L` modifiée par effet de bord obtenue en rétrogadant `L[i]` de sorte que le premier élément précédent `L[i]` dans lui soit strictement inférieur.
2. En déduire une fonction `tri_insertion(L)` prenant en entrée une liste de flottants `L` modifiant la liste `L` par effet de bord et triée dans l'ordre croissant.

V Tri rapide (quicksort)

C'est un tri récursif, par définition. Son intérêt est qu'il est plus rapide (en termes de nombre d'opérations) que les autres méthodes de tri qui ont été programmées. Voici une implémentation :

```
def qs(L):
    n  = len(L)
    if n<=1:
        return L
    else:
        pivot = L[0]
        Lgauche =[]
        Ldroite =[]
        for i in range(1,n):
            if L[i] < pivot:
                Lgauche.append(L[i])
            else:
                Ldroite.append(L[i])
    return qs(Lgauche)+[pivot]+qs(Ldroite)
```

■ **Exercice 5.** Expliquer son principe.

VI Applications des méthodes de tri

■ **Exercice 6. Médiane d'une statistique univariée**

Définition. Soit $x = (x_1 < \cdots < x_n)$ une série statistique de longueur n (triée). La médiane Q_2 de la série est :

$$Q_2 = \begin{cases} x_{1+\frac{n-1}{2}} & \text{si n est impair} \\ \frac{1}{2}\left(x_{\frac{n}{2}} + x_{1+\frac{n}{2}}\right) & \text{si n est pair} \end{cases}$$

Programmer une fonction `mediane(L)` prenant en entrée une liste de flottants `L` représentant une série statistique et donnant en sortie la valeur de sa médiane (attention à la numérotation des éléments d'une liste en python)

■ **Exercice 7. Tri d'une liste en fonction d'une autre**

1. Écrire une fonction `tri_couples(L)` prenant en entrée une liste `L` de 2-tuples (a, b) tels que a est un flottant, et renvoyant en sortie la liste `M` obtenue en rangeant les éléments de `L` par ordre croissant des a.
2. En déduire une fonction `tri2(L1,L2)` prenant en entrée deux listes de même longueur `L1`, `L2`, la liste `L1` étant une liste de flottants, et renvoyant en sortie la liste `L2` triée par ordre croissant des éléments de `L1`.