Donner page 1 seule d'abord

Objectif : Découvrir un nouveau paradigme de programmation.

1 Problématique

Le *tri* est une fonctionnalité très utilisée en programmation. Il est donc primordial qu'il soit le plus efficace possible.

Peut-on concevoir un outil de comparaison des algorithmes de tri ?

2 Algorithmes de tri

Activité 1 :

1. Choisir un des trois algorithmes de tri ci-après et implémenter une fonction qui accepte une liste et renvoie cette dernière triée.
2. Construire (en compréhension) une liste **tab** de 1000 entiers compris entre 0 et 100.
3. Par groupe de trois ayant choisis un algorithme différent, partager puis présenter les codes.

— utiliser jupyter ?
— docstring (+doctest ?)
— amélioration tri à bulles possible : Si lors du parcours de la liste une permutation au moins a été effectuée, recommencer un nouveau parcours.
— invariant ? en exercice

2.1 Tri par sélection

En partant du début du tableau et pour chaque élément de rang n :

— Rechercher la plus petite valeur et la permuter avec l'élément de rang n.

2.2 Tri par insertion

En partant du début du tableau et pour chaque élément de rang n :

— Mémoriser l'élément de rang n.
— En partant de l'élément n-1, décaler vers la droite les éléments qui sont plus grands que celui mémorisé.
— Placer dans le trou l'élément mémorisé.

2.3 Tri à bulles

En partant du début du tableau et pour chaque élément de rang n :

— Parcourir le tableau en comparant chaque élément avec son successeur.
— Si ce dernier est le plus petit des deux, les permuter.

Christophe Viroulaud

3 Notion de complexité

Il est légitime de se demander si certains algorithmes sont plus rapides que d'autres. La première question à se poser est de déterminer quels sont les paramètres qui influencent la durée d'exécution. Le tri par sélection ci-après est composé de deux boucles. Chaque boucle dépend de la taille (notée n) du tableau à trier. La durée d'exécution dépend donc du facteur $n \times n$.

Le tri par sélection a une **complexité en temps** en $O(n^2)$.

```
1 def tri_selection(tab):
2     taille = len(tab)
3     for i in range(taille):
4         rang_mini = i
5         for j in range (i+1,taille):
6             if tab[j] < tab[rang_mini]:
7                 rang_mini = j
8         tab[i],tab[rang_mini] = tab[rang_mini],tab[i]
9     return tab
```

— Évoquer complexité en espace

— tri bulles/insertion : meilleur des cas = (n-1) comparaisons, linéaire. Pire des cas : tableau trié à l'envers, quadratique $\frac{n.(n-1)}{2}$ comparaisons

— évoquer stabilité (insertion et bulles) ← on reverra en exo

Activité 2 : Déterminer la complexité en temps des algorithmes de tri à bulles et tri par insertion.

4 Comparaison des temps d'exécution

4.1 Contexte

Nous désirons comparer la durée d'exécution des trois fonctions de tri précédentes. Nous pouvons déjà remarquer que ces fonctions implémentent le même schéma, à savoir elles :

— acceptent une liste en entrée,

— renvoient la liste triée en sortie.

La bibliothèque *time* propose la fonction :

time.time() → float
Renvoie le temps en secondes depuis *epoch* sous forme de nombre à virgule flottante.

Pour Unix, epoch est le 1er janvier 1970 à 00:00:00 (UTC : Universel Temps Coordonné).

Christophe Viroulaud

4.2 Paradigme fonctionnel : une fonction est une donnée comme une autre

Contexte historique

$\lambda - calcul$ de Church / machine de Turing = principe équivalent qui peret de montrer calculabilité (cherche à identifier la classe des fonctions qui peuvent être calculées à l'aide d'un algorithme)
Quand von Neumann décrit son architecture d'un ordinateur, un des plus grands bouleversements tient au fait que données et programmes sont stockés dans la mémoire : un programme peut être considéré comme une donnée.
Ocaml, Scheme. langages sans effet de bord = plus facile à maintenir.

Un des premiers principes du *paradigme fonctionnel* est de considérer une fonction comme une donnée. Elle peut donc être passée en variable à une autre fonction.

Activité 3 :

1. Implémenter une fonction **duree_tri(fonction, tab : list) → float** qui mesure la durée que met *fonction* pour trier la liste *tab* et renvoie cette durée.
2. Tester la fonction **duree_tri** avec une fonction de tri et la liste *tab*.

4.3 Paradigme fonctionnel : données immuables

4.3.1 Effet de bord

Une fonction est dite à *effet de bord* si elle modifie un état en dehors de son environnement local.

Activité 4 : Un enseignant a obtenu des résultats décevants au dernier devoir donné. Il veut tester différentes majorations.

```
notes = [7,12,8,5,19,10,7,6,1,15,13,8]

def majoration(bonus: int):
    for i in range(len(notes)):
        if notes[i] <= 8:
            notes[i] += bonus
```

Que devient la liste *notes* après les deux appels suivants ?

```
majoration(2)
majoration(3)
```

4.3.2 Tuple

Afin de s'affranchir des effets de bord, la programmation fonctionnel n'utilise que des données non mutables. En Python les *tuples* possèdent ces caractéristiques.

Activité 5 :

1. Construire (en compréhension) un tuple **tab_immuable** de 1000 entiers compris entre 0 et 100.
2. Implémenter une fonction **duree_tri_fonctionnel(fonction, tab : tuple) → float** qui mesure la durée que met *fonction* pour trier *tab* et renvoie cette durée.

Christophe Viroulaud

— pour tuple en compréhension, voir slides : il faut ajouter *tuple* sinon on crée un générateur
— list(t) convertit tuple t en liste

4.4 Évolution du temps d'exécution

À faire en devoir ?

Le module *pyplot* de la bibliothèque *matplotlib* permet de réaliser des représentations graphiques facilement. Il est commun de l'importer avec un alias :

```
1 import matplotlib.pyplot as plt
```

Le site ci-après présente un rapide tutoriel permettant de tracer la courbe représentative d'une fonction :

https://zestedesavoir.com/tutoriels/469/introduction-aux-graphiques-en-python-avec-matplotlib-pyplot/

https://vu.fr/Pqyi

Activité 6 :

1. Réaliser une représentation graphique de la durée d'exécution d'un des tris étudiés, pour des tailles de listes variant de 10 à 5000 items. Il est conseillé d'utiliser au moins 10 listes pour avoir un résultat significatif.
2. La courbe obtenue confirme-t-elle les résultats du chapitre 3 ?

Christophe Viroulaud