ITÉRATION 2

Les tris récursifs

2.1 — Un peu de théorie

1h — Individuel

La récursivité est un concept simple à comprendre, mais souvent délicat à mettre en pratique.

RESSOURCES

- Récursivité — Wikipédia
- https://ldm_terminale.forge.apps.education.fr/terminale/05_Recursivite/2_Presentation_de_la_recursivite/
- Towers of Hanoi: A Complete Recursive Visualization

2.2 — Vous reprendrez bien un peu de théorie ?

2h — Individuel

Vous allez maintenant utiliser le paradigme de récursivité connu sous le nom « Divide-and-conquer » pour implémenter des algorithmes de tri plus avancés : le « quick sort » et le « merge sort ».

Récupérez le jupyter notebook `AlgosTriRecursifs.ipynb` et regardez attentivement son contenu. Pour chaque algorithme : une vidéo (pédagogique ?) et un pseudocode vous ont été fournis.

⚠ *Sur internet vous trouverez des implémentations différentes à celle proposée dans le notebook. C'est normal…*

Dans un premier temps, il est conseillé de se familiariser avec les principes à partir du notebook. N'hésitez pas à consulter les ressources si nécessaire.

RESSOURCES

- Article Wikipédia pour comprendre le paradigme algorithmique « divide and conquer » : Diviser pour régner (informatique) — Wikipédia

- Visualisation des différents algorithmes de tri - 15 Sorting Algorithms in 6 Minutes

---

2.3 — Le tri fusion

1h30 — Individuel

Implémenter le tri fusion.

⚠ *Vérifiez la justesse de vos conditions d'arrêt. Si un bug existe à la condition d'arrêt, votre algorithme ne répondra pas aux caractéristiques d'un bon algo : fin d'exécution. Dans ce cas, arrêtez et relancez jupyter.*

Quelques questions intéressantes à se poser :

- Quelle est la complexité O de l'algorithme ?
- Votre algorithme est-il une procédure ou une fonction ?

2.4 — Le tri rapide

1h30 — Individuel

Implémenter le tri rapide.

Quelques questions intéressantes à se poser :

- Le choix de pivot est-il optimal ? Quel est l'enjeu lié à ce choix ?
- Quels autres choix de pivot seraient possibles ? (🕹️ bonus : expérimentez)

2.5 — Création d'un module Python

30min — Individuel

Vous allez créer votre premier module python ! Pour commencer, récupérez le fichier `sorting.py`. Placez le fichier dans le dossier où se trouvent vos *notebooks*.

💡 *`sorting.py` est un fichier python qui se reconnaît par son extension `.py`. Ce fichier propose une conception préliminaire du module, c.à.d. les signatures des procédures/fonctions sont déclarées, mais pas leur implémentation.*

A l'aide d'un éditeur de texte (e.g. gedit, vim ou autre), remplissez chaque signature avec vos implémentations.

Créer un nouveau notebook et importer votre module à l'aide de l'instruction :

```
from sorting import *
```

Toutes les signatures qui se trouvent dans votre fichier sont désormais utilisables depuis n'importe quel *notebook*.

Quelles autres instructions d'import sont possibles ? Quels sont les avantages et inconvénients de chaque méthode d'import ?

COMPÉTENCES ASSOCIÉES

- Créer une bibliothèque en python

Livrables

➔ Le module `sorting.py` (à mettre en commentaire de la compétence associée sur Campus Skills)

2.6 — Comparaison des complexités empiriques

2h30 — Individuel

Testez différentes tailles de tableaux N comme données d'entrée de vos algorithmes de tri. Affichez les résultats sous la forme d'une figure, afin de pouvoir facilement comparer les algorithmes.

⚠ *Pour une comparaison juste, réfléchissez à une manière de tester l'efficacité de chaque algorithme avec les mêmes valeurs d'entrée.*

Quel algorithme semble être le plus efficace ?
Chaque algorithme peut être plus ou moins efficace selon l'état de désordre du tableau d'entrée. Quels sont les différents types de (dés)ordre possible ?

COMPÉTENCES ASSOCIÉES

- Analyse de la complexité d'un algorithme
