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. Pour découvrir la récursivité, lisez attentivement la ressource donnée.

RESSOURCES

- Cours d'introduction à la récursivité
  https://moodle.insa-rouen.fr/pluginfile.php/164408/mod_resource/content/1/07-Recursivite.pdf

---

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.*

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 » : https://fr.wikipedia.org/wiki/Diviser_pour_r%C3%A9gner_(informatique)
- Cours d'introduction au tri récursif
  https://moodle.insa-rouen.fr/pluginfile.php/165294/mod_resource/content/1/TrisRecursifs.pdf
- Visualisation des différents algorithmes de tri https://www.youtube.com/watch?v=kPRA0W1kECg

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

1h30 — 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 ?

RESSOURCES

- Connaître l'identité d'une instance https://docs.python.org/3/library/functions.html#id

COMPÉTENCES ASSOCIÉES

- Analyse de la complexité d'un algorithme

ITÉRATION 3 (facultative)

Plus dur, meilleur, plus rapide, plus fort

⚠ Cette itération est facultative. Elle n'est pas nécessaire pour valider les compétences du module. C'est du bonus pour ceux qui iraient vite. Assurez-vous d'avoir bien terminé les itérations 1 et 2 et d'avoir validé toutes les compétences avant de vous lancer sur cette troisième itération.

3.1 — Recherches supplémentaires

1h — Individuel

Quel type d'algorithme de tri est utilisé par la méthode `sort()` de Python ? Renseignez-vous sur cet algorithme, ses forces et ses faiblesses.

Une des propriétés des tris, non étudiée jusqu'à présent, est la stabilité. Qu'est-ce qu'un tri stable ?

3.2 — Soyez dynamique lorsque vous programmez !

2h — Individuel

La suite de Fibonacci commence par les termes `u[0] = 1` et `u[1] = 1`. Pour obtenir les termes suivants, la formule est `u[n] = u[n-1] + u[n-2]`. Proposez une fonction récursive, qui prend un entier n en entrée et qui retourne la n-ième valeur de la suite de Fibonacci.

Testez votre fonction sur les valeurs `[0, 1, 2, 5, 15, 45]`. Vous devez obtenir les résultats suivants :

```
print(fibonacci(0))   # 1
print(fibonacci(1))   # 1
print(fibonacci(2))   # 2
print(fibonacci(5))   # 8
print(fibonacci(15))  # 987
print(fibonacci(45))  # 1836311903
```

Qu'est ce que vous remarquez ? Tout en gardant votre algorithme récursif, réfléchissez à une solution pour optimiser votre fonction. Si vous n'avez pas d'idée, demandez-moi un indice.

3.3 — C'est en résolvant des problèmes qu'on devient un bon algorithme. Ou un truc dans le genre... 6h — Individuel

Il existe des calendriers de l'avent sous de très nombreuses formes. Un développeur propose une version un peu spéciale, qui demande de résoudre chaque jour un petit problème en deux parties, à l'aide d'un algorithme de votre conception.

Essayez de résoudre le jour 14 du calendrier 2021 : https://adventofcode.com/2021/day/14

Si vous avez terrassé les deux parties du jour 14, premièrement, bravo ! Deuxièmement, cherchez une solution au jour 23 : https://adventofcode.com/2021/day/23

Si vous avez réussi à coder une solution aux jours 14 et 23, vous avez bien mérité une petite pause. Mais si vraiment vous vous ennuyez, il reste tous les autres jours...
