Jouer avec les statistiques sur les communes de France

Un Apprentissage Par Problème (APP) destiné aux étudiants du module Algorithmie avancée 1

01/03/2021

Jouer avec les statistiques sur les communes de France

Le sujet s'appuie sur les données contenues dans le fichier https://www.data.gouv.fr/fr/datasets/r/554590ab-ae62-40ac-8353-ee75162c05ee, qu'il vous est demandé de télécharger (dans le même dossier que ce notebook).

Le but de l'APP est de fournir des algorithmes permettant de trier les communes de France selon certains critères.

Les données sont au format .csv (comma separated values). Vous devez importer ces données dans un script python nommé `numero_groupe_APP1.py`

Vous devez implémenter des algorithmes permettant de trier les communes de France par ordre croissant de distance à une ville donnée. Le but est de proposer au final l'algorithme le plus rapide possible.

L'implémentation que vous proposerez ne doit utiliser ni la fonction `sorted` ni la fonction `sort`.

Votre programme donnera en plus les statistiques suivantes : min (ville la plus proche de la ville passée en paramètre), premier quartile, médiane, max. L'implémentation d'une fonction donnant le quantile d'ordre α pourra être utile (rappel : la médiane est le quantile d'ordre 12, par exemple).

Lorsque la commune de référence est le premier arrondissement de Paris, une sortie possible (mais pas obligatoire) est :

```
                min :       0.82 km pour PARIS 02 (75002)
   premier quartile :     187.15 km pour PONT SUR SAMBRE (59138)
            médiane :     318.88 km pour BELLEFONTAINE (88370)
troisième quartile :     459.81 km pour QUEYSSAC (24140)
                max :    9422.32 km pour ST PHILIPPE (97442)
```

Vous serez confrontés à des choix. N'hésitez pas à explorer les données avec un tableur.

Les ressources pour traiter la situation-problème

Documents

Toolbook – les algorithmes de tri.ipynb

Mimo APP2 :

- Module 9 : Notions et dénombrement
- Module 10 : Analyse de deux tris itératifs
- Module 11 : Notations pour l'analyse asymptotique

Définition

Analyse de la complexité des algorithmes

https://fr.wikipedia.org/wiki/Analyse_de_la_complexit%C3%A9_des_algorithmes

> L'analyse de **la complexité d'un algorithme** consiste en l'étude formelle de la quantité de ressources (par exemple de temps ou d'espace) nécessaire à l'exécution de cet algorithme. Celle-ci ne doit pas être confondue avec la théorie de la complexité, qui elle étudie la difficulté intrinsèque des problèmes, et ne se focalise pas sur un algorithme en particulier.

Algorithme de tri

https://fr.wikipedia.org/wiki/Algorithme_de_tri

> Un algorithme de tri est, en informatique ou en mathématiques, un algorithme qui permet d'organiser une collection d'objets selon une relation d'ordre déterminée. Les objets à trier sont des éléments d'un ensemble muni d'un ordre total.

Formule de Haversine :

https://fr.wikipedia.org/wiki/Formule_de_haversine

Matplotlib : bibliothèque graphique en Python

https://openclassrooms.com/fr/courses/4452741-decouvrez-les-librairies-python-pourla-data-science/4740942-maitrisez-les-possibilites-offertes-par-matplotlib

Objectifs d'apprentissage de l'APP (AAV) : à l'issue de la séance « RETOUR » de cet APP, chaque étudiant doit être capable de…

- Expliquer ou reformuler les définitions des concepts de stabilité et de la gestion en place de la mémoire.
- Caractériser un tri en fonction de sa stabilité et de gestion en place de la mémoire.
- Réaliser en Python un algorithme de tri à partir d'un algorithme en pseudo-code.
- Evaluer un programme de tri en le modifiant afin de compter le nombre de comparaisons effectuées.
- Estimer la durée de l'application d'un algorithme de tri en fonction du nombre d'éléments à trier
- Exprimer la complexité temporelle d'un algorithme écrit en Python
- Choisir un algorithme parmi d'autres en connaissant uniquement leur complexité avec la notation O
- Choisir, parmi ceux qu'ils ont étudié, l'algorithme le plus pertinent en fonction du nombre de données et du type de données à traiter.
- Justifier le choix d'un tri.
- Comparer expérimentalement deux algorithmes de tri.
- Identifier le pire cas et le meilleur cas pour un algorithme de tri.
- Evaluer empiriquement le nombre d'opérations d'un algorithme en fonction du nombre des données d'entrées.
- Déterminer théoriquement la complexité avec la notation Landau d'un algorithme en fonction du nombre des données d'entrées.
- Définir ce qu'est la complexité spatiale ou temporelle d'un algorithme.
- Prédire le temps de calcul d'un algorithme de tri en fonction du nombre d'éléments.