Le projet GeoJSON

Romain Tavenard

1 Organisation

- Groupes de 2 ou 3 étudiants **du même groupe de TD**
- Déclaration des groupes :
  - au plus tard le jeudi 19/11/2020, par dépôt sur CURSUS d'un fichier contenant le nom des membres du groupe
  - si pas inscrit dans un groupe à cette date : note de 0 au projet
- Rendu final : au plus tard le vendredi 18/12/2020, 23h59

2 Énoncé

Il existe de plus en plus d'applications de suivi des coureurs à pied pendant leurs courses (exemple : livetrail). Dans ce projet, vous êtes dans la peau d'un programmeur auquel il a été demandé d'implémenter la brique de localisation des coureurs sur la trace GPS de la course.

2.1 Coeur du projet

Plus précisément, étant données une trace GPS (parcours de la course, sous la forme d'une suite de positions GPS) et une position instantanée (typiquement la position d'un coureur à un moment donné de la course, sous la forme d'une position GPS), il s'agira de localiser cette position sur la trace GPS et de fournir un résumé de la distance déjà parcourue et de la distance restant à parcourir. On estimera pour effectuer ces calculs que le point recherché est assimilable à son plus proche voisin dans la trace GPS fournie.

Attention, les limites en vigueur sur le nombre de points dans les trajectoires lors de l'appel à GraphHopper font que le calcul de la distance totale d'une (sous-)trajectoire ne pourra pas se faire pour des trajectoires comportant plus de 5 points. À vous de trouver une parade permettant dans ces conditions de répondre tout de même au problème posé (notez notamment que la limite du nombre de requêtes par jour est relativement élevée, elle).

3 Données à votre disposition

La trace GPS est fournie dans un fichier au format GeoJSON, comme dans l'exemple du fichier `marseille-cassis.json` fourni sur CURSUS.

La position instantanée devra être fournie comme entrée à votre programme.

Si vous avez besoin d'utiliser des API web pour lesquelles l'authentification est requise, les données d'identification devront être lues dans un fichier séparé (idéalement un seul fichier contenant les informations d'identification pour l'ensemble des services utilisés) et ce fichier ne devra pas être inclus dans le rendu.

3.1 Extensions

Il vous est proposé d'implémenter une ou plusieurs des extensions suivantes :

1. **Barre de progression**. Vous afficherez, en plus de la distance parcourue et de la distance restant à parcourir, une barre de progression générée à l'aide de caractères ASCII (exemple : `[===          ]  31 %`) qui permette de se faire une idée rapide de l'état d'avancement de la course pour le coureur en question
2. **Suggestion de points de rencontre pour les groupes de suiveurs**. Étant données les positions d'un nombre fixé de suiveurs, proposer un point de rencontre pour que ces suiveurs viennent encourager leur coureur sur la portion du parcours non encore empruntée. Le point choisi sera celui (ou un de ceux) qui minimise la distance maximale aux positions actuelles des suiveurs.
3. **Publication de l'état courant sur Twitter**. Étant données des informations d'identification pour un compte Twitter, publier sur le fil Twitter de ce compte un résumé de l'état courant, incluant les infos issues de l'extension 1 si elle est implémentée. Vous ajouterez un lien vers le tweet généré dans votre rendu.
4. **Affichage graphique**. En vous inspirant des pages d'aide du module `folium` et de ce tutoriel, vous générerez une sortie au format HTML représentant, sur une carte, en bleu le chemin déjà parcouru, et en rouge le chemin restant à parcourir.

4 Évaluation

Pour l'évaluation, il sera porté une attention particulière à ce que votre code soit lisible (commentaires : ni trop ni trop peu, nom de variables / fonctions pertinents, etc.) et bien découpé en blocs comme vous avez pu le voir lors du TD "Analyse et decomposition de problèmes".

De plus, les éléments notés "Extension" plus haut ne sont pas des "bonus" : il n'est pas envisageable d'obtenir la note maximale sans les avoir implémentés.