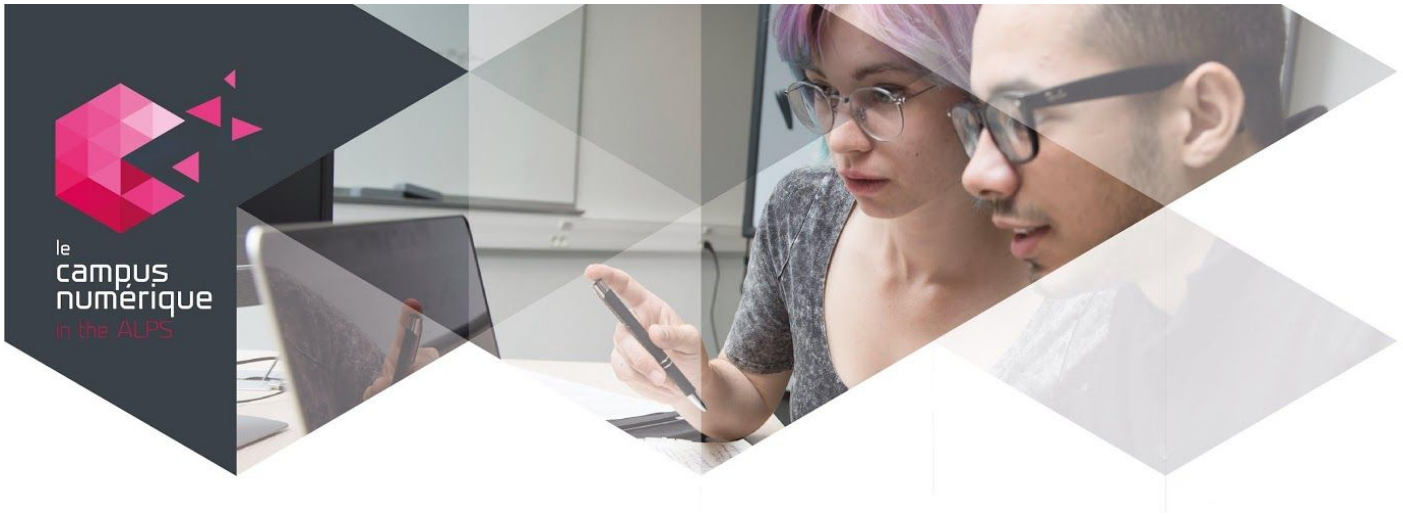


Algorithmique
Activité algorithmes de tri

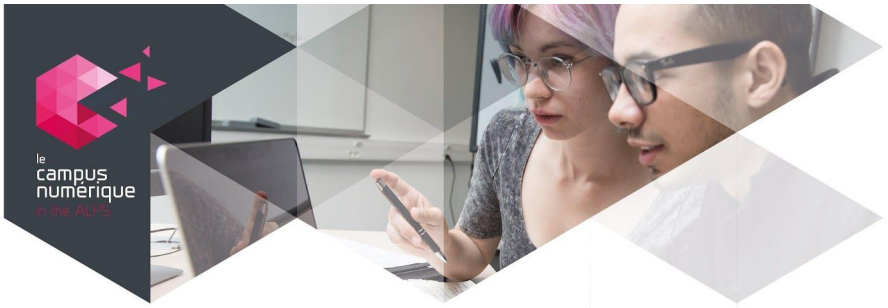
Activité préliminaire

Ordinateur éteint, les îlots vont devoir reproduire pendant une demi journée le fonctionnement des algorithmes qui leur seront distribués (une feuille de matériel par îlot). L'idée étant de matérialiser à l'aide de post-its les étapes successives des différents algorithmes de tri. Il est conseillé pour commencer de réaliser cet exercice sur le tri par insertion qui est les plus intuitif. Les îlots peuvent ensuite tester la liste d'algorithmes de leur choix. Les algorithmes permettant la validation du complément au diplôme sont :

- Le tri par insertion
- Le tri par sélection
- Le tri à bulle
- Le tri rapide

Il est vivement conseillé aux élèves ayant des difficultés de compréhension algorithmique de réaliser cet exercice individuellement. La liste de valeurs à trier est :

3	9	7	1	6	2	8	4	5



Ces valeurs doivent être inscrites sur des post-its. Il est conseillé de représenter au tableau les indices des post-its (les algorithmes numérotent les tableaux de 1 à N) :

1	2	3	4	5	6	7	8	9(N)
3	9	7	1	6	2	8	4	5
i								k
j								

Il peut également être utile d'utiliser des post-its pour les incréments (i, j et k). Pour chaque algorithme il est important de consigner le nombre de permutations et le nombre de comparaisons afin d'avoir une estimation de la "qualité" de l'algorithme. La demi journée suivante est dédiée à l'implémentation des algorithmes en JS.

Trier des villes

Il vous est demandé aujourd'hui d'analyser par groupes de 4 élèves 8 algorithmes de tri. L'objectif étant de trier des villes en fonction de leur distance par rapport à Grenoble. Cette classification est par exemple très utile lors de la réalisation d'une application de cartographie.

Pour ce faire, un jeu de données contenant les différentes villes et villages de france ainsi que leurs coordonnées GPS est disponible sur le site data.gouv.fr.

Un code minimal permettant de charger le fichier et d'afficher la distance entre les différentes villes est fourni : GitHub
Ce code minimal contient 3 jeux de données pour tester votre algorithme :

1. Small.csv : 28 premiers échantillons des villes de frances
2. Isere.csv : les 537 communes de l'Isère
3. Fr.csv : les 36850 communes française

⚠ Pensez à lancer le serveur comme l'indique la documentation GitHub

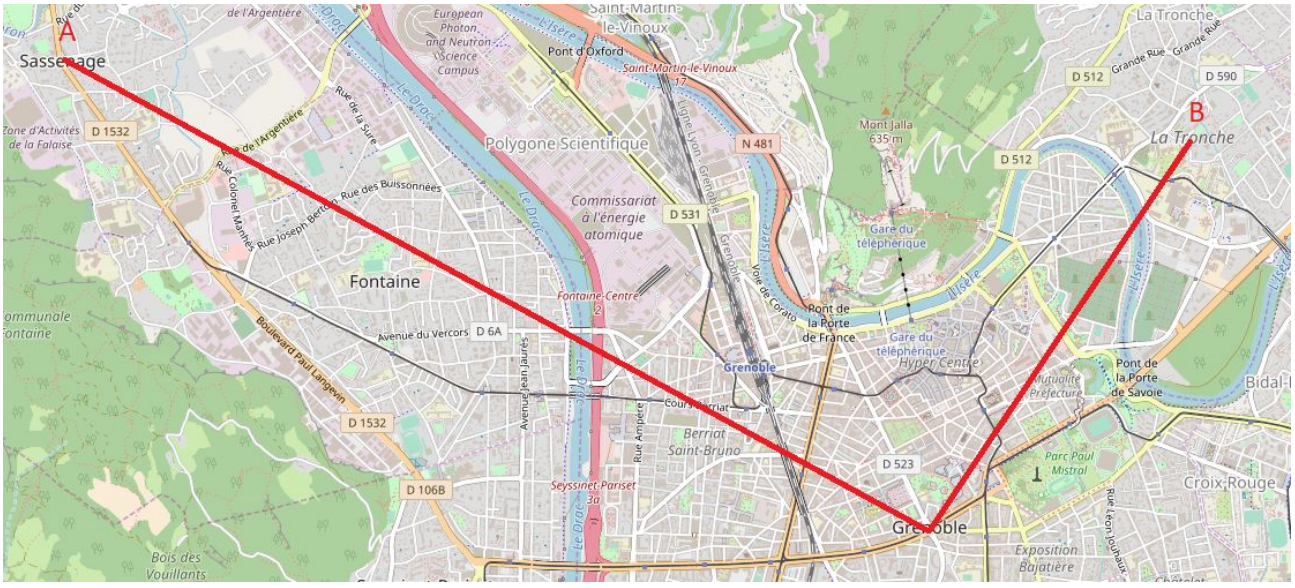
Un peu de maths

Avant d'aborder la question du tri d'un tableau, il est nécessaire d'aborder la question de la comparaison entre 2 éléments du tableaux.
L'élément A est il plus grand que l'élément B ?

A > B ?

Dans le contexte d'un tri de distance, la question à se poser est : Ma ville A est elle plus proche de Grenoble que ma ville B ? Ou encore, la distance entre A et Grenoble est elle supérieure à la distance entre B et Grenoble ?

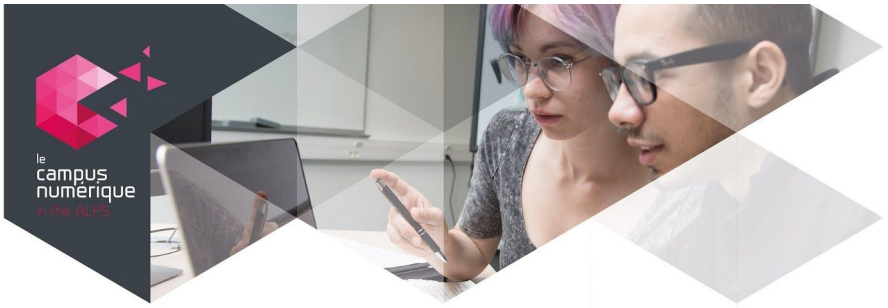
dist(A,Grenoble) > dist(B,Grenoble) ?



Pour répondre à cette question, vous pouvez utiliser la ressource suivante : http://www.movable-type.co.uk/scripts/latlong.html

Ecrire la fonction **distanceFromGrenoble**(city) qui calcule la distance à Grenoble. Le paramètre "city" est un objet. Utilisez console.log pour voir ce qu'il contient.

⚠ La distance ne doit pas forcément être calculée avec une très grande précision



Le programme contient un tableau de villes qui sont chargées dans la variable **csvData**.

Cette variable est initialisée lors que l'utilisateur appuie sur 

.

Ecrire la fonction **isLess**(i, j) qui prends en paramètre les indices sur 2 villes du tableaux et retourne vrai si csvData[i] est plus prêt de Grenoble que csvData[j].

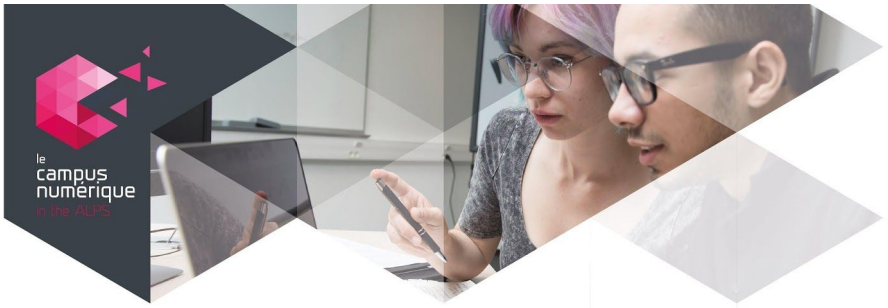
Permutation

Écrivez la fonction **swap(**i, j**)** qui prend comme paramètre les indices de 2 villes dans le tableau **csvData** et permute ces 2 villes.

Algorithmes de tri

Voici les différents algorithmes de tri à implémenter :
https://www.toptal.com/developers/sorting-algorithms

- Tri par insertion
  - Animation et algorithme
  - Danse roumaine
- Tri par sélection
  - Animation et algorithme
  - Danse gitane
- Tri à bulles
  - Animation et algorithme
  - Danse hongroise
- Tri de Shell
  - Animation et algorithme
  - Danse hongroise
- Tri par fusion
  - Animation et algorithme
  - Danse transylvanienne
- Tri par tas
  - Animation et algorithme
  - Danse hongroise
- Tri rapide
  - Animation et algorithme



- [Danse hongroise]
- [Animation et algorithme] (3 partitions)

Pour aller plus loin

Pour en savoir plus :

- [Un mémoire d'étudiant des arts et métiers]

Vous pouvez maintenant tester la distance routière entre 2 villes en utilisant une API :

- [Open Source Routing Machine]
- [Google Distance Matrix]