Structures de données et Algorithmes

ESILV S5 ⋈ 2018–2019

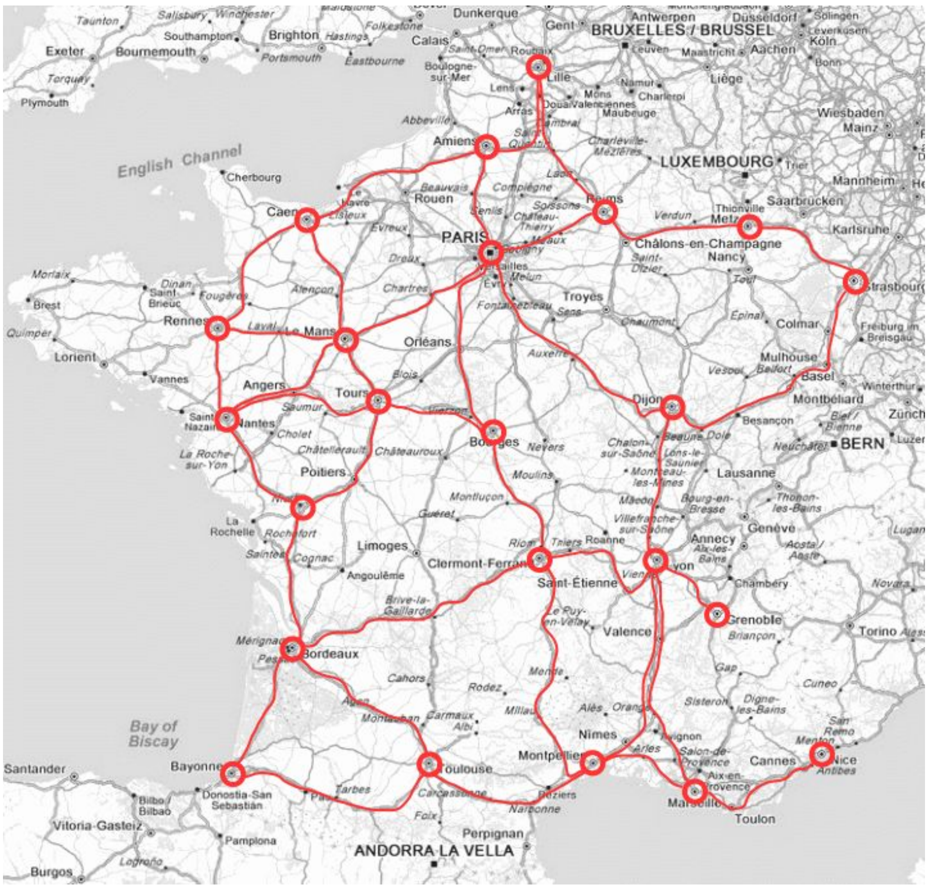
⌈ TD 7 & 8 – **Problème noté** ⌋

Attention bien lire cet encadré :

- Ce travail est à réaliser en binôme (du même groupe de TD).
- Il est à rendre au plus tard à 22h00 samedi 12 janvier 2019.
- Vous déposerez **sur le LMS un fichier** `.zip`, par binôme, contenant :
  - le **code source** (les fichiers `.h` et `.c`)
  - le **mini-rapport** en `PDF` de 5 pages maximum.
- Un rendu sans le rapport sera noté 00/20.
- Un programme avec des *Warnings* à la compilation sera noté avec un malus pouvant aller jusqu'à -5 points selon la nature des *Warnings*.
- Un programme ne compilant pas ou ne s'exécutant pas sera noté 00/20.
- Une triche entraîne une note de 00/20 à l'ensemble du module.

Énoncé du problème

On se propose de pouvoir définir un ou plusieurs trajets possibles pour relier en voiture une ville de départ à une ville d'arrivée parmi 23 villes françaises et 36 routes disponibles (*cf.* carte ci-dessous).



Données fournies

Le fichier `villes.csv`

— contient la liste des **23 villes** prises en compte.

Le fichier `connexions.csv`

— contient la liste des 36 connexions disponibles dans un sens et des 36 connexions disponibles dans l'autre sens entre deux villes, soit **72 connexions** au total ;

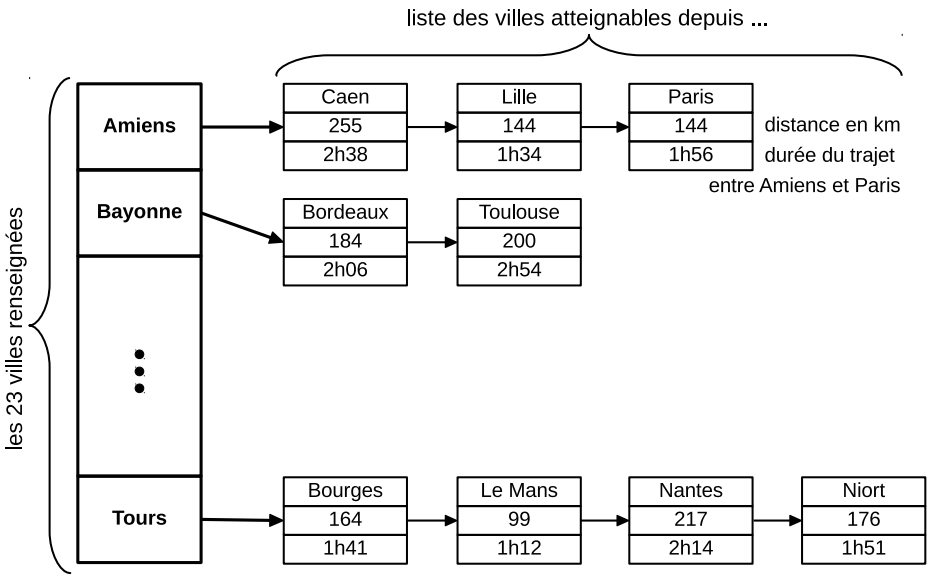
— pour chaque connexion reliant deux villes, on dispose de la **distance** les séparant en kilomètres ainsi que de la **durée** pour effectuer en voiture cette connexion inter villes.

Une connexion est donc un trajet direct entre deux villes (*i.e.* sans ville intermédiaire).

Ces deux fichiers sont au **format CSV**, c'est à dire un format texte où la première ligne renseigne le nom des colonnes séparées d'une virgule, puis sur les lignes suivantes une virgule sépare les valeurs de chaque colonne.

Structuration des données

Afin de pouvoir manipuler en mémoire les données fournies, une structuration générale est proposée (*cf.* figure ci-dessous).



Cette structuration dispose d'une part d'un tableau de 23 éléments, où chaque élément correspond à un nom de ville et une liste chaînée des connexions possibles depuis cette ville. D'autre part, pour chaque connexion (*i.e.* chaque maillon de la liste) sont renseignées la ville connectée, la distance et la durée du trajet sur cette connexion.

Par exemple, depuis la ville de Bayonne, il est possible d'atteindre directement la ville de Bordeaux en 2 heures et 6 minutes en parcourant 184 kilomètres ; il est aussi possible (toujours depuis Bayonne) d'atteindre directement Toulouse en 2h54 en parcourant 200 km.

Arbre des trajets

Pour pouvoir définir un ou plusieurs trajets entre une ville de départ et une ville d'arrivée, l'objectif principal de ce problème est d'être capable de construire un arbre $n-$aire particulier, que l'on appellera par la suite l'« arbre des trajets ».

Un **arbre** $n-$**aire** signifie qu'un nœud peut avoir un nombre quelconque de fils (zéro inclus, dans ce cas c'est une feuille). D'un point de vue de l'implémentation, un nœud – en plus de son contenu – ne disposera pas uniquement d'un pointeur vers le fils gauche et d'un pointeur vers le fils droit (ces pointeurs pouvant être `NULL`), mais d'un ensemble de pointeurs vers les fils. Cet ensemble peut être concrètement une liste chaînée ou un tableau, où chaque élément de l'ensemble pointe vers un fils (si pas de fils, alors l'ensemble est vide).

La **construction** de l'arbre des trajets se fera de la manière suivante :

- le nœud racine correspond à la ville de départ.
- les fils d'un nœud correspondent à toutes les villes connectées à la ville de ce nœud, sauf celles qui font partie des ascendants du nœud.
- la construction d'une branche de l'arbre s'arrête soit parce que la ville d'arrivée est atteinte, soit parce que la hauteur maximale a été atteinte.
  ✎ cette hauteur max est simplement un paramètre du programme.

L'**utilisation** de l'arbre des trajets est alors évidente : en partant de la racine, on parcourt l'arbre en profondeur en ne tenant compte que des feuilles dont la ville est l'arrivée souhaitée. On obtient alors les différents trajets calculés.

Remarque : bien prendre le temps de définir ce qu'est un nœud. Quels doivent-être les éléments présents ? C'est à dire ceux correspondant au contenu et ceux nécessaire à la structure de l'arbre $n-$aire.

Travail à réaliser

1. Programme

Créer un programme `C`

- qui au lancement, à partir des fichiers `villes.csv` et `connexions.csv`, charge en mémoire les informations routières au sein d'une ou plusieurs structures adaptées.
- qui demande à l'utilisateur la ville de départ et la ville d'arrivée souhaitées, puis à partir de ces deux villes renseignées **crée l'arbre des trajets**.
- qui ensuite affiche un **menu** proposant les fonctionnalités suivantes :
  1. afficher un trajet trouvé
  2. afficher tous les trajets trouvés
  3. afficher le trajet le plus court (ou les plus courts si égalité)
  4. afficher le trajet le plus rapide (ou les plus rapides si égalité)
  5. afficher l'arbre des trajets
- qui pour finir affiche le résultat souhaité.
  ✎ puis possibilité de (re)proposer à l'utilisateur de faire un nouveau choix avec éventuellement la modification préalable des villes de départ et/ou d'arrivée.

Attention : l'affichage d'un trajet correspond à afficher *a minima* les noms des villes parcourues entre la ville de départ et la ville d'arrivée (en incluant éventuellement ces deux villes). Peuvent en plus être affichées les distances et durées intermédiaires (de chaque connexion), ainsi que la distance totale et la durée totale du trajet.

Remarque : la hauteur maximale de l'arbre des trajets peut être renseignée en dur dans le code (variable initialisée par un valeur fixée dans le code) ou bien demandée à l'utilisateur au lancement du programme. Par défaut, on prendra 8 comme hauteur maximale de l'arbre.

Contraintes

Une attention particulière devra être portée sur la qualité du code :
- découpage en (sous-)fonctions ⇒ donc pas tout dans le `main` ou dans une grosse fonction !
- noms des variables et fonctions explicites
- organisation adaptée du code (fichiers `.h` / `.c`)
- commentaires à bon escient

Seules les bibliothèques suivantes peuvent être utilisées : `<stdlib.h>`, `<stdbool.h>`, `<string.h>`, `<time.h>` et `<stdio.h>`.

2. Mini-rapport

Un mini-rapport de 5 pages maximum est à rendre avec le code.

Doivent y figurer :
- les nom et prénom des élèves du binôme et leur groupe de TD ;
- une explication en français de comment sont structurées les données en mémoire (réponse rapide si la structuration utilisée est celle suggérée sur le sujet) ;
- une explication en français du/des types choisis pour définir un arbre des trajets ;
- une explication en français de comment est codée chaque fonctionnalité pour
  - construire automatiquement l'arbre des trajets, et
  - manipuler l'arbre des trajets ;
- une explication en français de comment fonctionne le menu ;
- et enfin un exemple d'affichage en Console d'un arbre des trajets, des différents trajets trouvés, dont le plus court et le plus rapide, pour une ville de départ et d'arrivée choisies (en précisant le hauteur maximale utilisée lors de la construction de l'arbre).