Travail pratique #3 : Les Voyageurs de commerce

---

1. Objectifs

- Utiliser des structures de données d'une bibliothèque normalisée (les conteneurs de la librairie standard de C++).
- Implémenter un type abstrait de données `Carte` basé sur la représentation de graphe.
- Implémenter un algorithme de recherche de chemin dans un graphe.
- Implémenter un algorithme d'optimisation pour trouver une solution optimale au problème classique du voyageur de commerce.

2. Problématique

Vous devez écrire un programme C++ nommé `tp3`. Votre programme `tp3` doit calculer le trajet optimal pour des voyageurs de commerce (exemple: agents d'assurance) devant visiter un certain nombre de lieux à chaque jour.

3. Structure des programmes

Pour bien amorcer ce travail, il est fortement recommandé de commencer avec le squelette de départ fourni dans tp3.zip. Vous pouvez modifier ces fichiers autant que vous le désirez. Toutefois, pour la correction automatique, vous devez préserver la syntaxe d'appel du programme et ses formats d'entrée et de sortie.

3.1 Syntaxe d'appel

Le programme `tp3` doit pouvoir être lancé en ligne de commande avec la syntaxe suivante.

```
./tp3 carte.txt [mission.txt]
```

où :

- le fichier `carte.txt` spécifie une carte;
- le fichier `missions.txt` contient les missions (liste de listes de lieux à visiter);

Si `missions.txt` n'est pas fourni, le programme `tp3` doit lire les missions depuis l'entrée standard (*stdin*), c'est-à-dire depuis le flux d'entrée C++ `std::cin`.

Les résultats produits par votre programme doivent être écrits dans la sortie standard (*stdout*) à l'aide du flux de sortie C++ `std::cout`.

3.2 Fichier carte

Un fichier `carte.txt` est constitué de :

1. Une liste de lieux (noeuds). Un lieu est spécifié par :
    - un nom (une chaîne de caractères);
    - une coordonnée de la forme (latitude,longitude).
2. Trois tirets (---) de séparation.
3. Une liste de routes. Une route est spécifiée par :
    - un nom de route (une chaîne de caractères);
    - un deux-points (:);
    - une liste de noms de lieux;
    - un point-virgule (;).

À titre d'exemple, voici le fichier (uqam-carte.txt) :

```
n1 (45.508377,-73.568755)
n2 (45.508662,-73.569259)
n3 (45.509128,-73.570289)
n4 (45.509331,-73.570793)
n5 (45.509944,-73.57056)
n6 (45.510452,-73.570348)
n7 (45.511113,-73.570010)
n8 (45.510579,-73.568894)
n9 (45.510106,-73.567939)
n10 (45.509910,-73.567435)
n11 (45.509346,-73.567929)
n12 (45.510004,-73.569399)
n13 (45.509561,-73.568428)
n14 (45.509497,-73.568256)
n15 (45.509474,-73.569881)
n16 (45.509523,-73.569694)
n17 (45.509636,-73.569715)
n18 (45.509741,-73.569656)
n19 (45.508940,-73.569045)
n20 (45.509170,-73.569173)
---
Jeanne-Mance : n1 n2 n3 n4 ;
Sherbrooke : n4 n5 n6 n7 ;
Sherbrooke : n7 n6 n5 n4 ;
Saint-Urbain : n7 n8 n9 n10 ;
President-Kennedy : n10 n11 n1 ;
Grands-Batisseurs : n6 n12 n13 n14 n11 ;
Grands-Batisseurs : n11 n14 n13 n12 n6 ;
UQAM1 : n3 n15 n16 n17 n18 n12 n8 ;
UQAM1 : n8 n12 n18 n17 n16 n15 n3 ;
UQAM2 : n2 n19 n14 ;
UQAM2 : n14 n19 n2 ;
UQAM3 : n9 n13 ;
```

```
UQAM3 : n13 n9 ;
UQAM4 : n19 n20 ;
UQAM4 : n20 n19 ;
```

Et voici la carte correspondante :

Remarque: dans le TP3, les sens uniques doivent être considérés. Lorsqu'une route est bidirectionnelle, elle est définie dans les deux directions.

3.3 Missions

Un fichier `missions.txt` contient les missions à calculer. Chaque mission est spécifiée sur une ligne dans le format suivant. La ligne débute par le lieu d'affaire (l'origine et la destination finale), suivi du symbole deux-points («:»), et enfin, la liste des lieux à visiter terminée par un point-virgule («;»).

À titre d'exemple, voici le fichier uqam-missions-0.txt.

```
n4 : n8 ;
```

Et le fichier uqam-missions-1.txt.

```
n4 : n8 n2 ;
n1 : n2 n8 ;
n1 : n8 n2 ;
n1 : n2 n19 ;
n1 : n2 n3 n19 ;
n1 : n19 n2 n3 ;
```

3.4 Format de sortie pour tp3

Le programme `tp3` doit afficher pour **chaque** mission :

1. sur la première ligne, la séquence de nœuds du chemin le plus court reliant l'origine et la destination;
2. sur la deuxième ligne, la liste des noms de route à emprunter;
3. sur la troisième ligne, la longueur totale du trajet en mètres, arrondie à l'unité près, suivi de la chaîne " m".

Exemple:

```
$ ./tp3 uqam-carte.txt uqam-mission-0.txt
n4 n5 n6 n12 n8 n12 n18 n17 n16 n15 n3 n4
Sherbrooke Grands-Batisseurs UQAM1 Jeanne-Mance
540 m
$ ./tp3 uqam-carte.txt uqam-mission-1.txt
n4 n5 n6 n12 n8 n9 n13 n14 n19 n2 n3 n4
Sherbrooke Grands-Batisseurs UQAM1 Saint-Urbain UQAM3 Grands-Batisseurs UQAM2
Jeanne-Mance
735 m
n1 n2 n3 n15 n16 n17 n18 n12 n8 n9 n13 n14 n11 n1
Jeanne-Mance UQAM1 Saint-Urbain UQAM3 Grands-Batisseurs President-Kennedy
681 m
n1 n2 n3 n15 n16 n17 n18 n12 n8 n9 n13 n14 n11 n1
Jeanne-Mance UQAM1 Saint-Urbain UQAM3 Grands-Batisseurs President-Kennedy
681 m
n1 n2 n19 n14 n11 n1
Jeanne-Mance UQAM2 Grands-Batisseurs President-Kennedy
329 m
```

```
n1 n2 n19 n2 n3 n15 n16 n17 n18 n12 n13 n14 n11 n1
Jeanne-Mance UQAM2 Jeanne-Mance UQAM1 Grands-Batisseurs President-Kennedy
604 m
n1 n2 n19 n2 n3 n15 n16 n17 n18 n12 n13 n14 n11 n1
Jeanne-Mance UQAM2 Jeanne-Mance UQAM1 Grands-Batisseurs President-Kennedy
604 m
```

À noter que la longueur totale du trajet doit être arrondie, avec la fonction `round`, uniquement pour l'affichage. Les longueurs des parcours intermédiaires, dans votre algorithme, ne doivent jamais être arrondies, au risque d'obtenir de mauvais résultats.

4. Hypothèses et Constantes

1. Les routes sont des arcs sur la surface de la Terre. La distance entre deux lieux (noeuds) sur la carte se calcule avec la fonction `Coordonnee::distance()` fournie. Celle-ci retourne la longueur en mètres en supposant que la Terre est une sphère parfaite d'un rayon de 6371 km.
2. Les voyageurs se déplacent à une vitesse constante de 40 km/h. L'accélération et la décélération du véhicule et l'attente aux arrêts et feux rouges ne sont pas considérées.
3. Les restrictions de virage, incluant les virages à gauche et en U, ne sont pas considérées.

5. Contraintes

5.1 Librairie standard C++ obligatoire

Contrairement aux TP1 et TP2, **vous devez maintenant utiliser, autant que possible, les conteneurs de la librairie standard de C++ (*Standard Template Library*)**. Cette contrainte vise à mettre en pratique l'utilisation d'une bibliothèque normalisée. Évidemment, vous pouvez créer vos propres structures lorsque justifié.

5.2 Environnement de développement

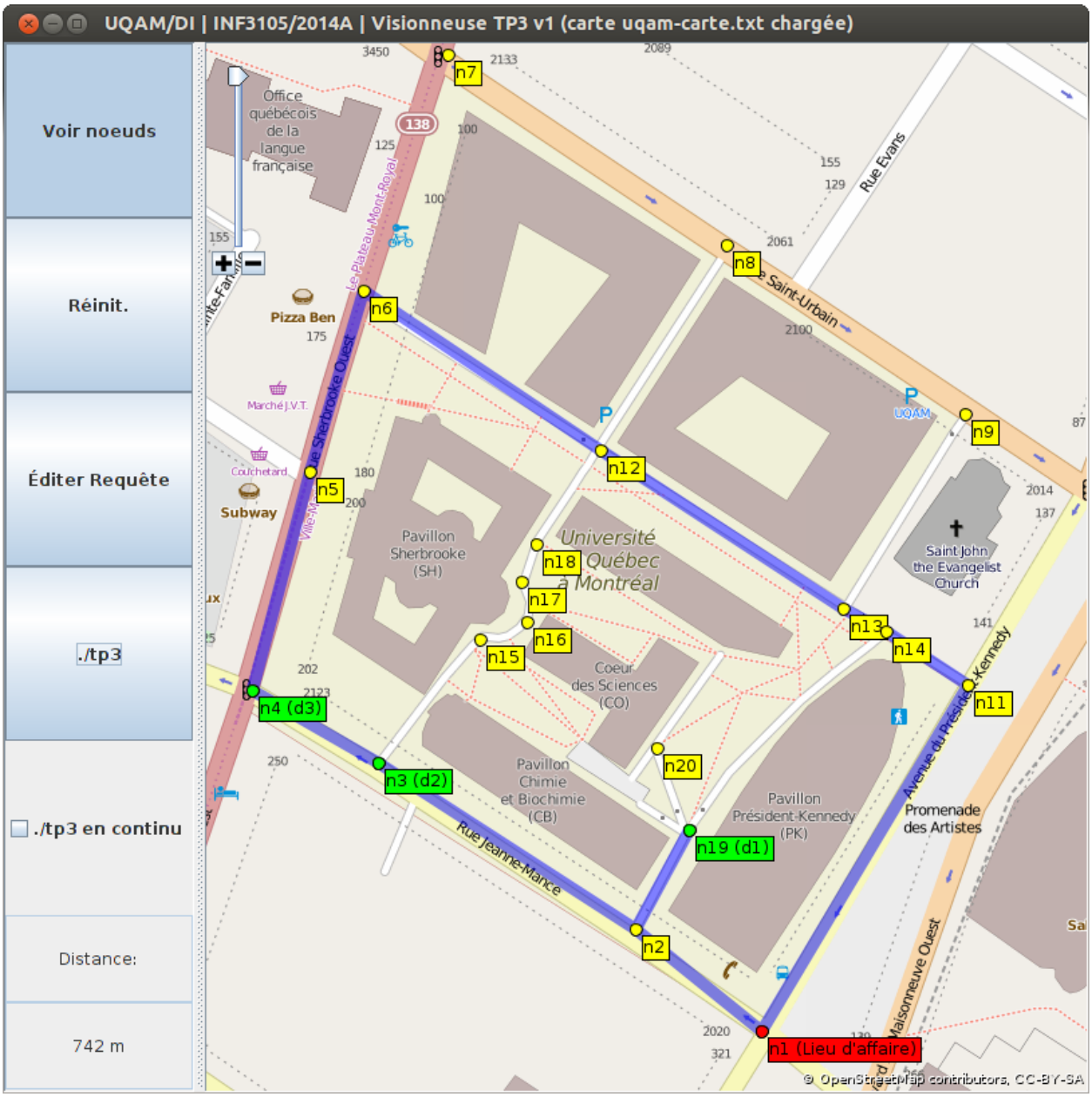
Relisez les Politiques et les directives sur les outils informatiques dans le cours INF3105.

5.3 Taille des équipes

Vous pouvez faire ce travail en équipe de 1 ou 2. **Toutefois, tous les membres de l'équipe doivent contribuer à l'ensemble du travail et non à seulement quelques parties. Le travail d'équipe vise à favoriser les discussions et l'entraide. Le travail d'équipe ne vise pas à réduire la tâche. Ainsi, se diviser la tâche en deux n'est pas une méthode de travail d'équipe appropriée dans ce cours. Tous les membres de l'équipe doivent être en mesure de comprendre et d'expliquer l'ensemble du travail. La participation inadéquate d'une étudiante ou d'un étudiant peut être considérée comme du plagiat.** Le professeur et le correcteur pourront sélectionner quelques équipes au hasard afin de vérifier que tous les membres sont capables d'expliquer l'ensemble du travail.

6. Interface graphique

Pour vous aider à développer et à valider vos programmes, une visionneuse est fournie avec dans tp3.zip (VisionneuseTP3.jar).



Utilisation:

- Commande pour lancer la visionneuse :

  ```
  java -jar VisionneuseTP3.jar
  ```

- Par défaut, la visionneuse tente de charger la carte depuis le premier fichier existant dans la séquence : `uqam-carte.txt`, `montreal-carte.txt`, `montreal.osm`, `carte.txt`.
- Pour charger une carte spécifique, il suffit d'ajouter les noms de fichier. La visionneuse supporte des fichiers carte en format `.txt` ou `.osm` (XML). Exemples :
- `java -jar VisionneuseTP3.jar uqam-carte.txt`
- `java -jar VisionneuseTP3.jar montreal-carte.txt`
- `java -jar VisionneuseTP3.jar montreal-carte.osm`
- Pour ajouter le lieux d'affaires et les destinations, il suffit de double-cliquer sur la carte.
- Le bouton **Réinit.** permet de réinitialiser, c'est-à-dire d'effacer les points choisis.
- Le bouton **Éditer Requête** permet de saisir une mission. Exemple : «`n4 : n8 ;`». Cela vous sera utile pour tester des missions spécifiques.
- Le bouton **./tp3** permet de tester votre programme `tp3` ou `tp3.exe`.
    - Le programme `tp3` doit être dans le répertoire courant.
    - La commande `./tp3` est exécutée avec comme paramètres la carte chargée.
    - La visionneuse écrit une ligne dans l'entrée standard de processus de `tp3`.
    - La visionneuse lit trois lignes depuis la sortie standard de processus de `tp3`.
    - Le trajet calculé est affiché dans la visionneuse.
    - Limitations : la visionneuse ne vérifie pas la validité et l'optimalité des solutions générées.
- La case à cocher **./tp3 en continue** permet de garder le programme en mémoire et exécuter une succession de missions. Cela permet d'éviter de recharger la carte à chaque requête.

7. Tests

Fichiers test : tp3-tests.tar.bz2 (15 Mo)

7.1 Évaluation sur votre propre machine (Linux, Mac OS et Cygwin)

Les fichiers test, le script `evaluer.sh` et le valideur sont déjà sur **malt** dans le répertoire `/home/inf3105/tp3/`. Vous n'avez pas à télécharger les fichiers test.

```
$ cd tp3
$ make
$ tar xvfj tp3-tests.zip
$ ./tests/evaluer.sh
...
Évaluation du programme tp3 ...
...
Test                    Miss    CPU     Mém.(k) Valid   Optimal DistOK  RouteOK
        /Total  Réussi?
uqam-missions-0.txt     1       0.00    1480k   1       1       1       1
/1      Optimal
uqam-missions-1.txt     6       0.00    1484k   6       6       6       6
/6      Optimal
uqam-missions-D0.txt    1       0.00    1480k   1       1       1       1
/1      Optimal
...
```

```
uqam-missions-F12.txt   10      0.66    1496k   10      10      10      10
/10     Optimal
uqam-missions-F13.txt   10      0.87    1500k   10      10      10      10
/10     Optimal
...
montreal-missions-D0.txt        1       0.27    31712k  1       1       1
1       /1      Optimal
montreal-missions-D1.txt        2       0.31    35260k  2       2       2
2       /2      Optimal
montreal-missions-D2.txt        10      0.47    56920k  10      10      10
10      /10     Optimal
montreal-missions-D3.txt        100     2.07    285528k 100     100     100
100     /100    Optimal
montreal-missions-D4.txt        500     9.89    1319252k        500     500
500     500     /500    Optimal
montreal-missions-D5.txt        1000    20.40   1319252k        1000    1000
1000    1000    /1000   Optimal
montreal-missions-D6.txt        5000    98.01   1319252k        5000    5000
5000    5000    /5000   Optimal
montreal-missions-F02.txt       10      0.66    68044k  10      10      10
10      /10     Optimal
montreal-missions-F03.txt       10      0.85    77588k  10      10      10
10      /10     Optimal
montreal-missions-F04.txt       10      1.28    107624k 10      10      10
10      /10     Optimal
montreal-missions-F05.txt       10      1.43    121712k 10      10      10
10      /10     Optimal
montreal-missions-F07.txt       10      1.70    134540k 10      10      10
10      /10     Optimal
montreal-missions-F08.txt       10      1.97    146680k 10      10      10
10      /10     Optimal
montreal-missions-F09.txt       10      2.29    148624k 10      10      10
10      /10     Optimal
montreal-missions-F10.txt       10      3.97    173632k 10      10      10
10      /10     Optimal
montreal-missions-F11.txt       10      14.56   184624k 10      10      10
10      /10     Optimal
montreal-missions-F12.txt       10      48.00   198132k 10      10      10
10      /10     Optimal
montreal-missions-F13.txt       10      288.34  212128k 10      10      10
10      /10     Optimal
```

7.3 Temps de référence

Exemples de temps sur deux solutions.

8. Remise

Vous devez remettre le TP3 **au plus tard le 29 avril à 23h59**.

Votre projet, en format zip, doit être remis sur le site Moodle dans la section Travaux pratiques. Veuillez mettre vos codes permanents dans le nom de votre fichier.

Vous pouvez soumettre votre TP autant de fois que vous voulez. Seule la dernière soumission sera considérée.

Vous devez remettre tous vos fichiers sources, incluant un fichier Makefile et votre rapport.

Rapport

- **Auto-évaluation** indiquant si votre programme fonctionne correctement, partiellement ou aucunement.

- **Un tableau montrant les temps d'exécution sur les fichiers tests fournis.**
  Si un test prend plus de 60 secondes, vous pouvez l'arrêter ([Ctrl]+[C]) et simplement écrire **> 60** dans votre rapport.

- Vous **pourrez** générer un rapport automatiquement en lançant la commande `/home/inf3105/tp3/evaluer.sh` sur `malt` ou `rayon[12]` à partir du répertoire où vous avez compilé votre exécutable `tp3`.

9. Évaluation

Ce travail pratique vaut 15% de la note finale.

9.1 Grille de correction

Critère	Description	Pondération
A.	**Respect des directives pour la remise.** • Fichiers sources seulement (Makefile, .h, .cpp). Aucun fichier source manquant. Aucun fichier intermédiaire (.o, .obj, .gch, etc.) ou exécutable (tp3, tp3.exe). Aucun fichier test. • Remise par Moodle. • Compilable avec `make` sans modifications. • Exécutable sans modification.	/ 1

B.	**Appréciation générale.**
	
	• Structure du programme.
	  ◦ Découpage du programme (tout n'est pas dans la fonction `main`).
	  ◦ Justesse des types et des structures de données.
	  ◦ Classes et fonctions.
	  ◦ Usage du mot clé `const`.
	  ◦ Usage des références et des pointeurs.
	• Qualité du code.
	  ◦ Nomination des identificateurs (noms significatifs), lisibilité du code, etc.
	  ◦ Présence et pertinence des commentaires; etc.
	• Encapsulation.
	  ◦ Respect des principes de l'abstraction;
	  ◦ Cachez le maximum de la représentation des objets en rendant un maximum d'attributs privés;
	  ◦ Évitez autant que possible les bris d'abstraction, comme des *getters* et *setters* qui retournent ou affectent directement des attributs d'un type abstrait de donnée. Par exemple, les fonctions `getX()` et `getY()` ne devraient pas exister dans une classe `Point`. Mais, une fonction `getNom()` dans une classe `Station` peut être justifiée. La représentation d'une classe `Date` devrait être privée. Une structure ou classe `Intervalle` peut avoir deux attributs publics `debut` et `fin`.
	  ◦ Utilisation appropriée des modificateurs d'accès `public`, `protected` et `private`, et du mot clé `friend`, etc.
		/ 2

	• Gestion de la mémoire. ◦ Toute la mémoire allouée dynamiquement doit être correctement libérée au moment approprié et avant la fin de la fonction `main`.	
C.	**Génération de chemins valides.** Les chemins générés n'ont pas besoin d'être optimaux.	/ 3
D.	**Génération de trajets optimaux pour des missions contenant une seule destination.**	/ 4
E.	**Efficacité (temps d'exécution) pour générer des trajets optimaux pour des missions contenant une seule destination.**	/ 3
F.	**Génération de trajets optimaux pour des missions contenant plusieurs destinations.** L'efficacité n'est pas directement considérée.	/ 2
	Total :	/ 15
G.	**Boni : Efficacité pour les missions contenant plusieurs destinations.**	+ 1
	Note maximale :	16 / 15

Pour les cas problématiques, jusqu'à 2 points peuvent être retranchés pour la qualité de la langue et de la présentation.

9.2 Machine d'évaluation

- CPU : Intel(R) Core(TM) i5-3550 CPU @ 3.30GHz
- OS : Ubuntu Linux 14.04 LTS / x86_64
- g++ --version : g++-4.8.real (Ubuntu 4.8.2-19ubuntu1) 4.8.2
- Mémoire : maximum 2 Go fixé par `ulimit -t 60 -v 2097152 -f 131072` dans `evaluer.sh`

10. Crédits et licences

- Les données des cartes fournies sont extraites d'OpenStreetMap et sont régies par la licence *Open Database License (ODbL) v1.0*.
- Une partie du code source de la visionneuse provient du projet JMapViewer.
- Les tuiles (images) des cartes proviennent d'OpenStreetMap et sont distribuées sous la licence *Creative Commons* paternité – partage à l'identique 2.0 (CC-BY-SA).