Automne 2019

Algorithmes et structures de données pour ingénieurs

GLO-2100

Travail à faire individuellement
À rendre avant la date indiquée
sur le portail du cours
(Voir modalités de remise à la fin de l'énoncé)

Travail Pratique #3

Algorithmes de plus courts chemins efficaces pour itinéraires sur le réseau de la RTC

UNIVERSITÉ LAVAL
Faculté des sciences et de génie
Département d'informatique
et de génie logiciel

1 Objectif général

Ce TP constitue le dernier TP d'une séquence de 3 TPs portant sur les données du réseau de transport de la capitale (RTC) de la ville de Québec. Au cours des 2 premiers TPs, vous deviez implémenter certaines méthodes de la classe DonneesGTFS vous permettant de gérer l'accès aux données GTFS de la RTC et certaines méthodes de la classe ReseauGTFS vous permettant de construire un graphe sur lequel vous pouvez exécuter un algorithme de plus court chemin vous donnant le meilleur itinéraire d'un voyage utilisant le réseau d'autobus de la RTC. Pour ce TP3, vous devez implémenter un algorithme de plus court chemin qui soit substantiellement plus efficace que celui fourni dans la classe Graphe (fournie avec cet énoncé).

Si vous n'avez pas réussi votre TP1, vous pouvez utiliser la librairie statique libTP1.a fournie avec cet énoncé qui implémente toutes les classes du TP1, y compris DonneesGTFS. Si vous n'avez pas réussi votre TP2, vous pouvez utiliser la librairie statique libTP2.a fournie avec cet énoncé qui implémente la classe ReaseauGTFS. En fait, nous recommandons à tous d'utiliser libTP1.a et libTP2.a au cas où vos précédents TPs comporteraient certaines erreurs n'ayant pas été détectées lors de la correction. Par contre, notez que libTP1.a et libTP2.a ne s'utilisent que dans la machine virtuelle du cours. Vous devez utiliser tous les fichiers .h fournis avec cet énoncé et utiliser le CMakeLists.txt fourni avec cet énoncé pour la compilation avec CMake.

Pour les étudiants en génie : ce TP sera utilisé pour évaluer la Qualité 3 « Investigation ». Si vous n'effectuez pas ce TP, cela pourrait compromettre l'obtention de cette Qualité à votre diplomation, et, conséquemment, mettre en péril votre démarche éventuelle en vue de l'obtention du titre d'ingénieur.

2 Travail à faire

En plus des archives libTP1.a et libTP2.a, nous vous fournissons la classe **Graphe** que vous devez utiliser. Cette classe contient une méthode **plusCourtChemin()** contenant une implémentation simpliste de l'algorithme de Dijkstra qui n'est pas très efficace au niveau du temps d'exécution. Vous devez remplacer le corps de cette méthode par une implémentation d'un algorithme de plus court chemin qui soit le plus efficace possible (au niveau du temps d'exécution).

Le méthode **main()**, fournie avec l'énoncé, vous permettra d'exécuter votre algorithme pour différentes paires (origine, destination) choisies aléatoirement. La variable **nbDeTests** vous permet de choisir le nombre des tests (ie, le nombre de paires (origine, destination) à essayer) désirés. La variable booléenne **afficherItinéraire** permet d'afficher l'itinéraire trouvé pour chaque test (donc, choisir false pour afficherItinéraire si nbDeTests est élevé). Les variables **today** et **now1** permettent de choisir la date et l'heure pour vos itinéraires. La variable **now2** est fixée à une heure si élevée que vous obtiendrez un graphe contenant tous les arrêts du reste de la journée. Ne modifiez pas now2. Le fichier out.txt, fourni avec cet énoncé, vous donne la sortie du programme avec le main() fourni et l'implémentation de Graphe fournie. Vous y trouverez également les temps d'exécution obtenus pour l'implémentation fournie de l'algorithme de Dijkstra sur les 10 instances testées. Ce résultat fut obtenu à partir d'un exécutable compilé à l'aide de l'option –O3 (tel que spécifiée dans le CMakeLists.txt fourni) et exécuté sur un MacBook Pro 2016.

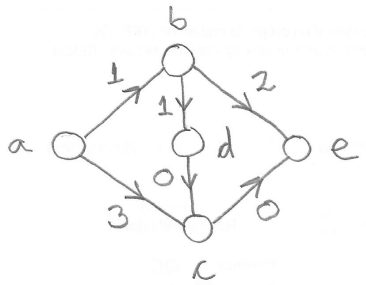
On vous demande de fournir un algorithme de plus court chemin substantiellement plus efficace que celui fourni. Pour évaluer la performance de votre algorithme, vous devez fournir le **facteur d'accélération** défini par le temps d'exécution moyen de l'algorithme de Dijkstra fourni divisé par le temps d'exécution moyen de votre algorithme. Notez que plus le nombre de tests effectués, pour calculer les moyennes de temps d'exécution, est élevé et plus précis sera votre estimé du facteur d'accélération. Pour ce TP, disons que nbDeTests = 200 nous donne une estimation suffisamment précise du facteur d'accélération de votre algorithme. Cependant, pour que vos résultats soient valides, il est indispensable que votre algorithme et celui fourni soient testés dans des conditions les plus similaires possibles (par exemple, avec aucune autre application exécutant sur votre ordinateur et l'utilisation des mêmes options de compilation).

Votre code doit compiler correctement avec le CMakeLists.txt fourni et l'exécutable résultant doit s'exécuter correctement (sans planter). Les correcteurs compileront votre code avec le CMakeLists.txt fourni et testeront ensuite l'exécutable obtenu sur la machine virtuelle du cours.

Votre démarche d'investigation, pour arriver à trouver l'algorithme de plus court chemin le plus efficace possible pour le réseau de la RTC, devra être l'objet d'un **rapport écrit à rendre**. Dans votre réflexion, vous devez vous en tenir à des algorithmes non parallèles car un seul "thread" d'exécution sera à la disposition de l'utilisateur de votre application. Nous vous conseillons de commencer votre démarche d'investigation en examinant, la structure du réseau de transport de la RTC. En effet, c'est sur ce réseau que vous ferez vos tests, donc autant analyser ses caractéristiques (densité, existence ou non de cycles, valeurs possibles des poids, nombre de nœuds et d'arcs, etc) avant de commencer votre recherche. Veuillez consigner vos observations dans votre rapport. Suite à vos observations, examinez les algorithmes que nous avons vus dans le cours et faites vos choix en fonction de leur propriétés et de leur complexité algorithmique (temps d'exécution). Vous pouvez également explorer d'autres pistes et d'autres algorithmes que nous n'avons pas vus au cours. À cet effet notez que certains algorithmes possèdent des exigences afin qu'ils soient assurés de toujours trouver le chemin le plus court. **Vous devrez faire en sorte que l'algorithme que vous proposez soit assuré de toujours trouver le chemin le plus court**. Pour cette raison, vous ne devriez pas proposer l'algorithme A* car ce dernier nécessite l'utilisation d'un heuristique valide qui est une borne inférieure sur la distance du plus court chemin de tout nœud vers le nœud destination. Or certains arcs (de type voyage) du graphe ont un poids de 0; ce qui implique l'absence d'un heuristique valide et utile pour A* pour le réseau de la RTC.

Examinez de près toutes les pistes et justifiez le choix qui vous semble être le plus prometteur. Consignez les raisons de vos choix dans le rapport. À cet effet, l'algorithme de plus court chemin donnant le plus faible ordre de croissance du temps d'exécution en pire cas est souvent le plus prometteur. Par contre, lorsque l'on utilise un algorithme de plus court chemin à origine unique (comme Dijkstra, Bellman Ford, etc) pour trouver le plus court chemin d'une paire (origine, destination), il faut également prendre en compte certaines caractéristiques de l'algorithme: comme le fait que certains doivent parcourir tout le graphe avant de terminer alors que d'autres peuvent être terminés dès que le plus court chemin à la destination choisie est trouvée. Ainsi, il est normal que vous ayez à tester plusieurs algorithmes avant de trouver le meilleur. De plus, si vous utilisez un Tas Min pour accélérer l'algorithme de Dijkstra, notez que lorsque que la distance d'un sommet du graphe change suite au relâchement d'un arc, il faut alors (en principe) pouvoir aller directement (en temps O(1)) au nœud du tas associé à ce sommet (qui ne se trouve pas nécessairement à la racine du tas), modifier sa distance et reconstruire le tas suite à cette modification de distance. Or, le conteneur priority_queue de la STL et la bibliothèque <algorithm> ne fournit aucune méthode permettant aux utilisateurs d'aller directement à un nœud à l'**intérieur** du tas et de reconstruire le tas suite à la modification de sa valeur de clé. En

conséquence, une implémentation d'un Tas Min utilisant priority_queue ou les méthodes fournies par <algorithm> et qui n'apporterait aucune modification au tas suite au relâchement d'un arc (et à la modification de la distance du sommet pointé par l'arc), ne trouverait pas le plus court chemin allant du sommet **a** au sommet **e** pour le graphe suivant



et ce, même si cette implémentation retourne un plus court chemin correct pour un ensemble de (disons) 1000 paires (origine, destination) choisies au hasard dans le graphe de la RTC. Cela illustre bien le fait qu'une solution correcte pour un nombre fini d'instances ne signifie pas pour autant que l'algorithme est **valide** : i.e., qu'il retourne **toujours** une solution correcte, **peu importe l'instance**. Or, ici l'algorithme que vous proposez doit être valide.

Au niveau du code, on vous demande de modifier la méthode Graphe::plusCourtChemin() afin qu'elle soit substantiellement plus efficace que celle fournie. Vous devez donc respecter à la lettre le prototype de cette méthode afin qu'on puisse la tester. On vous demande également de ne pas modifier les autres méthodes de Graphe et les autres fichiers fournis avec cet énoncé. Par contre, vous pouvez ajouter d'autres méthodes et d'autres attributs membres pour supporter votre algorithme.

Votre rapport d'investigation doit être un document PDF d'au plus 3 pages qui contiendra les éléments suivants.

1. La formulation, en vos propres mots, de ce que vous devez faire pour ce TP.
2. Un résumé sur les algorithmes de plus court chemin que vous avez envisagés et les raisons qui ont conduit au choix de votre algorithme dans le cadre du réseau de la RTC.
3. Une description complète des étapes algorithmiques qui ont conduit à votre implémentation finale. Par exemple, les structures de données que vous avez dû modifier dans un algorithme, les opérations que vous avez dû optimiser. En gros tout votre processus de planification depuis le moment où vous avez choisi une certaine direction de résolution du problème.
4. Une présentation des résultats obtenus suite à votre démarche d'investigation et votre analyse de ces résultats.
5. Vos conclusions et recommandations.

Important : notez que nous vous fournissons le fichier CMakeLists.txt pour votre projet. Dans ce cas, les exécutables et les librairies statiques produites se trouvent dans le même répertoire que les fichiers sources. Le répertoire build (ou le répertoire cmake-build-debug lorsque l'on utilise CLion) se trouve également dans ce répertoire. De plus le dossier RTC contenant les fichiers de données doit se trouver dans ce répertoire. C'est ce CMakeLists.txt qu'utiliseront les correcteurs pour corriger votre TP sur la machine virtuelle du cours. **Vous devez donc vous assurer que votre programme compile (et s'exécute) sans erreur sur la machine virtuelle du cours avec le CMakeLists.txt fourni.** Prenez note

que CLion modifie parfois le CMakeLists.txt lors de la création d'un projet ou il produit un nouveau CMakeLists.txt dans un autre répertoire. Assurez-vous donc, que c'est bien le CMakeList.txt fourni avec l'énoncé qui est utilisé pour votre projet.

3 Fichiers à remettre

Vous devez remettre un fichier rapport.pdf contenant votre rapport ainsi que les fichiers graphe.h et graphe.cpp que vous devez insérer dans une archive ayant pour nom NomDeFamille_Prénom.zip. SVP, ne remettre aucun autre fichier. Les autres fichiers fournis avec l'énoncé ne doivent donc pas être modifiés. Assurez-vous que la méthode plusCourtChemin() contienne le code de votre algorithme que vous jugez le plus performant. Vous devez remettre votre travail dans la boite de dépôt du portail du cours. Aucune remise par courriel ne sera acceptée. Tout travail remis en retard perdra 3 points par heure de retard. Donc, un retard de 1 seconde à 3599 secondes occasionne la perte de 3 points, et après un retard de plus de 33 heures, votre note tombera automatiquement à zéro. Vous pouvez remettre autant de versions que vous le désirez. Seul le dernier dépôt sera corrigé.

ATTENTION : vous avez la responsabilité de vérifier l'intégrité des fichiers que vous avez remis dans la boite de dépôt. Donc, nous vous demandons de vérifier ce que vous avez remis (en téléchargeant sur votre poste ce que vous avez téléversé dans la boite de dépôt) afin d'en vérifier l'intégrité. Le mécanisme de téléversement du Portail ne corrompt pas les fichiers. Cependant, il peut arriver que votre archive ait été corrompue si, lorsque vous l'avez créée, certains des fichiers étaient ouverts par d'autres applications (Eclipse ou CLion par exemple...). Vérifiez donc l'intégrité de votre archive après l'avoir construite!
Tout fichier remis qui est inutilisable sera considéré comme un fichier non remis.

4 Critères de correction

Le TP sera évalué en fonction du code **et** du rapport fourni. Cependant, le barème de correction se base d'abord sur les éléments suivants contenu dans le rapport :

1. **Formulation claire de la tâche à accomplir (10 points)** : Vous devez formuler, en vos propres mots, de ce que vous devez faire pour cette partie du TP.
2. **Examen des options possibles et justification du/des choix (15 points)** : Vous devez mentionner les algorithmes de plus court chemin que vous avez envisagés au début de votre investigation et les raisons qui ont conduit à éliminer certains de ces algorithmes. Notez également les raisons qui penchent en faveur de votre choix d'algorithme dans le cadre du réseau de la RTC.
3. **Description des étapes algorithmiques conduisant à l'implémentation finale (30 points) :** Fournissez une description complète des étapes algorithmiques qui ont conduit à votre implémentation finale. Par exemple, les structures de données que vous avez dû modifier dans un algorithme, les opérations que vous avez optimisées. En gros, présentez tout votre processus de planification depuis le moment où vous avez choisi une certaine direction de résolution du problème.
4. **Présentation des résultats et analyse (35 points) :** Présentez les résultats obtenus suite à votre démarche d'investigation et votre analyse de ces résultats.
5. **Conclusion et recommandations (10 points) :** Présentez vos conclusions et vos recommandations finales quand à l'algorithme à utiliser. Ils doivent découler et être supportés par votre analyse.

La note obtenue sera ensuite modulée en fonction des critères suivants :

1. **Qualité du français** : vous pouvez perdre jusqu'à 20% des points selon la qualité du français.
2. **Respect de ce qui a été demandé** : vous pouvez perdre jusqu'à 100% des points si vous avez pas complété le travail. Par exemple, un étudiant perdra 100% des points s'il fourni un rapport, mais ne fourni aucun algorithme de plus court chemin codé. L'étudiant perdra également 100% des points si la classe Graphe fournie plante durant l'exécution ou ne compile pas sur la machine virtuelle du cours. Notez que l'algorithme proposé doit être valide. Sinon, vous pouvez perdre jusqu'à 40% des points.
3. **Résultats conformes au code fourni** : L'étudiant perdra jusqu'à 100% des points si les résultats du rapport ne sont pas appuyés par les tests du correcteur sur le code fourni (ex : le rapport prétend un facteur d'accélération de 18 pour l'algorithme fourni mais les tests du correcteur donnent un facteur d'accélération de 7).
4. **Exactitude du code demandé** : vous pouvez perdre jusqu'à 50% des points si certains tests unitaires échouent. Testez donc votre classe Graphe !
5. **Efficacité du code** : Vous perdrez 40% des points si l'algorithme de plus court chemin, se trouvant dans la méthode plusCourtChemin() possède un facteur d'accélération plus petit que 10 par rapport à l'algorithme de Dijkstra fourni. Au delà de ce facteur d'accélération (10), vous devriez perdre 0 point. À titre d'exemple, le professeur a implémenté un algorithme donnant un facteur d'accélération de 19,4. Notez que votre code doit s'exécuter sur un seul thread.

5 Questions ?

Si vous avez des questions pertinentes à propos de ce TP, nous vous demandons de les poser sur le forum des TPs qui se trouve sur le portail du cours. Comme cela, la réponse sera visible à toute la classe, et non seulement à un seul individu. **Cependant, étant donné la nature de ce TP, on vous demande de ne pas dévoiler sur le Forum votre stratégie et vos idées au sujet de ce qui constituerait le meilleur algorithme de plus courts chemins pour le réseau de la RTC. On vous demande également de ne pas susciter l'opinion des autres à ce sujet**. Toute question envoyée directement au professeur par courriel sera laissé sans réponse afin d'éviter de donner de l'information pertinente à un seul individu.

6 Plagiat

Tel que décrit dans le plan de cours, le plagiat est interdit. Une politique stricte de tolérance zéro est appliquée en tout temps et sous toutes circonstances. Tous les cas seront référés à la direction de la Faculté. **Prenez note qu'il est interdit de publier (sur le Web par exemple) ce TP et sa solution et ce, même après la fin de cette session.**