EFREI 2021/2022.
SM503
L3

Projet

Recherche de chemin de valeur minimale par l'algorithme de Floyd-Warshall

IMPORTANT : Lisez bien l'intégralité de ce document avant de commencer à travailler. Cela vous évitera de mauvaises surprises.

L'algorithme de Floyd-Warshall permet de calculer les chemins de valeurs minimales de tout sommet à tout autre sommet dans un graphe orienté et valué.
Cet algorithme est basé sur celui de Roy-Warshall dédié au calcul de la fermeture transitive d'un graphe. Le calcul de cette fermeture transitive est adapté afin de conserver, parmi tous les chemins reliant 2 sommets, un de ceux ayant la valeur la plus faible.

L'algorithme de Floyd-Warshall n'est abordé ni en cours, ni en travaux dirigés. Ce projet intègre donc un petit travail de recherche de votre part. Une explication de l'algorithme, avec un exemple, est présente dans les diapos du cours ; vous êtes libres à chercher d'autres explications ailleurs.

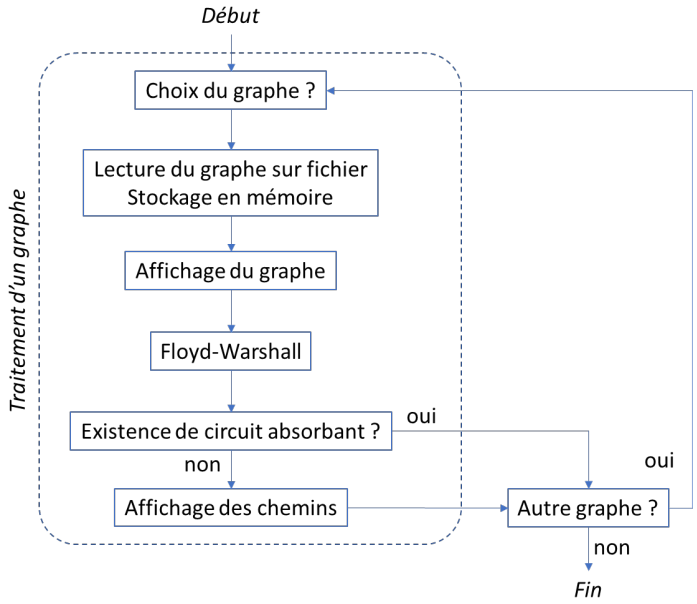
CONDITIONS GENERALES

Le projet est à réaliser par équipes de 5 personnes en règle générale. Votre enseignant de TD vous indiquera combien votre groupe TD doit constituer d'équipes.
Vous devrez utiliser l'un des langages de programmation suivants : C, C++, Java ou Python.
Une soutenance de projet est prévue entre le 4 et le 7 janvier. Vous devrez avoir envoyé à votre enseignant le résultat de votre travail le **3 janvier**. Des graphes de tests, à partir desquels votre programme sera testé, vous seront communiqués **avant** le **3 décembre**. Lors de l'élaboration du programme, vous n'avez pas besoin de ces graphes de test : il doit marcher sur n'importe quel graphe.

Le temps de présentation est limité à 15 minutes. Après 15 minutes, votre enseignant interrompra la présentation quelque soit la situation, et si tout le monde n'a pas eu le temps de présenter, cela aura une grande influence négative sur la note. Cette limitation de 15 minutes ne sera possible que si vous répétez la présentation avant la soutenance plusieurs fois, et si vous évitez tout ce qui n'est pas strictement nécessaire (des phrases générales, l'utilité du projet, des explications de la structure du fichier texte (c'est déjà là dans l'énoncé !), etc.
Une présentation utilisant les lignes de code est proscrite. Ce n'est pas le code qui nous intéresse, c'est l'algorithme, donné sous forme de pseudocode ou de diagramme. La seule exception pourrait être un programme Python s'il est aussi bien lisible et aussi laconique qu'un pseudocode.

TRAVAIL A REALISER

Illustration :



Boucle principale :

Votre programme doit pouvoir exécuter les traitements demandés sur une suite de graphes, au choix de l'utilisateur. Lors de la soutenance, **il ne doit pas être nécessaire de relancer votre programme pour le tester avec plusieurs graphes.**

Traitement d'un graphe :

Dans un premier temps, l'utilisateur indique le graphe à analyser. **Les graphes seront identifiés par des numéros.**

Le graphe est ensuite chargé en mémoire à partir de sa description contenue dans un fichier au format « texte ». Voir annexe.
Le graphe « lu » est stocké dans des structures de données. **A partir de ce stade, votre programme ne doit en aucun cas accéder une seconde fois au fichier : seule la représentation en mémoire est utilisée.**

La représentation en mémoire du graphe sous la forme de chaînes de caractères reprenant les lignes du fichier texte ne sera pas efficace pour un programme opérationnel devant traiter des graphes volumineux. Cela est vrai quelle que soit sa mise en œuvre précise, utilisant des tableaux, des vecteurs, des ensembles, …
Utilisez les matrices d'adjacence ou les listes d'adjacence !

Le graphe est ensuite affiché à l'écran sous forme matricielle : une matrice d'adjacence binaire plus une matrice de valeurs, ou les deux combinées.
Faites très attention à la lisibilité de cet affichage, notamment l'alignement des colonnes et l'identification des sommets (titres des lignes et colonnes).

L'algorithme de Floyd-Warshall est exécuté. Les valeurs intermédiaires des données de l'algorithme (matrices habituellement nommées 'L' et 'P') doivent être affichées.

A la lecture du résultat de l'algorithme, votre programme indique si le graphe contient au moins un circuit absorbant.

Si aucun circuit absorbant n'est présent, le dernier traitement consiste à afficher les chemins de valeurs minimales. Vous pouvez les afficher tous à la fois, ou prévoir une boucle d'interface avec l'utilisateur :

Chemin ?
Si oui alors

Sommet de départ ?
Sommet d'arrivée ?
Affichage du chemin
Recommencer
Si non alors arrêter

GRAPHES A PRENDRE EN COMPTE

Votre programme doit pouvoir traiter correctement n'importe quel graphe qui répond aux critères ci-dessous :

- Graphe orienté
- Graphe valué – Valeurs numériques entières quelconques (valeurs négatives admises, valeur '0' admise)
- Sommets représentés par des nombres entiers, de '0' à 'nombre de sommets moins 1'
- Au plus un arc d'un sommet x vers un sommet y.

Votre programme ne doit imposer aucune limite pour le nombre de sommets, ni pour les valeurs des arcs, ni pour le nombre d'arcs.

DEROULEMENT DU PROJET

Constitution des équipes

Nombre d'équipes par groupe TD/TP : autour de 8
Votre enseignant vous l'indiquera par voie d'annonce sur Moodle dans les jours qui suivent.

Constitution des équipes : à remettre à votre enseignant au plus tard **le 28 octobre**.
Aucun changement ne sera possible après cette date. A défaut d'une constitution des équipes fournie par **les délégués** de chaque groupe TD, les enseignants pourront décider eux-mêmes de la constitution des équipes, sans possibilité de modification.

Problèmes éventuels : S'il y a problème au niveau d'une équipe (un élève qui ne participe pas au travail des autres, un élève malade, etc…), vous êtes tenus d'en informer votre enseignant immédiatement.
Aucune modification d'équipe sans l'aval direct de l'enseignant ne sera acceptée.

Langage de programmation

Au choix : C, C++, Python, Java
Le langage choisi doit cependant être suffisamment maîtrisé par tous les membres d'une même équipe afin que tous puissent participer. Durant la soutenance, votre enseignant pourra poser n'importe quelle question à n'importe quel membre de l'équipe, qu'il participe à l'élaboration de la partie du programme concerné ou non.

Votre programme doit pouvoir être compilé / exécuté par votre enseignant, sur son PC. Il vous indiquera ce dont il dispose. Vous devrez vous y adapter. Vous ne devez pas utiliser des bibliothèques spéciales que votre enseignant devrait télécharger.

Soutenance

Les soutenances ont une durée limitée. Votre enseignant à un planning très serré. Il ne pourra pas continuer les soutenances après le créneau horaire prévu.
Il vous est donc vivement conseillé d'être vraiment prêt à l'heure du début de votre soutenance, ce qui implique (au cas où une soutenance physique a lieu) :

- attendre « à la porte de la salle » et « entrer » dès que c'est à votre tour ;
- avoir préparé votre ordinateur, y avoir inclus tous vos programmes et fichiers contenant les graphes ;
- avoir vérifié le chargement de la batterie de l'ordinateur ;
- l'avoir démarré et mis en mode veille ;

- avoir un ordinateur de secours sur lequel vous avez aussi vérifié le bon fonctionnement de votre programme, au cas où…

Tous les ans, il y a plein d'étudiants qui pensent ne jamais avoir de problème. Tous les ans, il y en a qui en ont. Résultat : ils sont stressés et ont moins de temps pour leur soutenance. Dommage…

Chaque étudiant sera individuellement interrogé sur un des aspects de votre programme (question concrète portant sur un choix de réalisation, ou question plus générale liée au fonctionnement de l'algorithme de Floyd-Warshall).
Vous devez donc passer du temps pour discuter « théoriquement » du fonctionnement de Floyd-Warshall, ainsi que pour partager la maîtrise du code (chacun expliquant aux autres les parties qu'il/elle a développées).

Graphes de test

Des graphes de test vous seront fournis avant la soutenance sous forme graphique, dans un délai suffisant pour vous permettre de les coder par des fichiers .txt au format que vous aurez choisi (**avant le 3 décembre**). Ces fichiers devront impérativement être préparés à l'avance, et ils feront partie du rendu. Ils doivent être disponibles sur votre ordinateur au moment de la soutenance.

N'attendez pas ces graphes pour tester votre programme ! Il serait alors trop tard.
N'hésitez pas pendant votre développement à utiliser tous les graphes vus en cours et travaux dirigés, et bien d'autres encore. Plus vous ferez de tests, plus vous pourrez être certains que votre programme fonctionne correctement.

Rendu du travail

Toutes les équipes devront remettre leur travail à leur enseignant à la même date de **3 janvier**.
Le contenu du rendu :

- Code source : Tout fichier code **que vous avez tapé vous-même** à l'exclusion de tout autre fichier (**donc aucun fichier produit par le logiciel durant la compilation ou exécution)**, bien commenté.
  Tous les ans, il y a des équipes qui ne respectent pas cette consigne et qui ont des points de pénalité à cause de cette négligence. Essayez de l'éviter.
- Tous les fichiers .txt des graphes de test (oui, malgré le fait que ce sont vos enseignants qui vous ont fourni les graphes de test), dans le même répertoire que le code.
- Un ppt ou pdf de la présentation (facultatif : vous pouvez le joindre au rendu de votre travail si vous l'avez déjà, mais vous serez autorisés à le modifier **jusqu'à votre soutenance**).
- Les traces d'exécution sous forme d'un fichier .txt. Vous devrez exécuter votre programme sur l'intégralité des graphes de test mentionnés ci-dessus et fournir les traces d'exécution correspondantes.
  Un graphe non testé, ou pour lequel les traces d'exécution ne seront pas fournies, sera considéré comme un graphe sur lequel votre programme ne fonctionne pas correctement.

Tout fichier que vous utilisez (et transmettrez à votre enseignant) doit être **préfixé par votre numéro d'équipe** : par exemple, si vous avez un fichier « main » et si vous utiliser le langage C++, et que vous êtes dans l'équipe B2, ce fichier doit avoir le nom **B2-main.cpp** (ou B2_main.cpp). Attention ! Cela concerne *tous* les fichiers ! Tous les ans, on rencontre des rendus dans lesquels une partie des fichiers ne respectent pas cette condition. Cela résulte en un malus…

Annexe – Exemple de fichier permettant de représenter un graphe

Important : Cette annexe ne contient qu'un exemple de structure de fichier. Vous avez le droit de décider de votre propre structure aux conditions suivantes :
- la structure du fichier doit être simple, facilement compréhensible ;
- le graphe représenté dans le fichier doit pouvoir être modifié directement dans le fichier, de façon extrêmement simple (par exemple pour ajouter ou supprimer un arc, ou pour changer la valeur d'un arc).

Votre fichier peut, *par exemple,* avoir la structure suivante :

Ligne 1	Nombre de sommets
Ligne 2	Nombre d'arcs
Lignes 3 à 3 + « nombre d'arcs »	Extrémité initiale, suivie de l'extrémité terminale, suivie de la valeur de l'arc

Avec ce modèle de fichier, s'il contient le texte suivant :

```
4
5
3 1 25
1 0 12
2 0 -5
0 1 0
2 1 7
```

cela correspond à un graphe contenant 4 sommets numérotés de 0 à 3 (numérotation contiguë), et 5 arcs (3,1), (1,0), (2,0), (0,1), (2,1) de valeurs respectives 25, 12, -5, 0 et 7.