Université de Montpellier 2019/2020
ULIN507 Durée: 20h

TER L3
TD – Séance n

Exercice 1 – Problème du VERTEX-COVER

VERTEX-COVER
Input: Un graphe $G = (V, E)$, un entier k.
Question: Trouver $V' \subset V$ telle que chaque arête est couverte par un sommet de V'.

Nous considérons le problème classique VERTEX-COVER. Ce problème a été largement étudié du point de vue de la complexité, de l'approximation et des méthodes exactes. Dans cet TER, le but est d'étudier VERTEX-COVER sous plusieurs approches proposées ci-après.

1. Relaxation d'un programme linéaire en nombres entiers
2. Algoritme primal-dual
3. Décomposition sous forme d'un arbre borné
4. Kernelization
5. Recherche d'un couplage maximum
6. Couverture par des cliques maximales
7. Parcours en profondeur.
8. ...

Compétences souhaitées : un goût pour l'algorithmique, programmation

1 Objectifs du projet et travail à faire

1.1 Programme à écrire

1.1.1 Présentation des algorithmes, des structures de données, du programme et des fonctions

Le *cahier de programmation* que vous devrez rendre devra obligatoirement comporter les éléments suivants.

Algorithme en pseudo code

- Une description et une justification précise (par des arguments d'efficacité, de lisibilité, etc) des structures de données que vous utilisez.
- Une étude bibliographique des problèmes
- Une évaluation aussi précise que possible de la complexité en temps et en mémoire de votre algorithme.

Programme en C

- Un mode d'emploi de votre application.
- Le listing complet et commenté de votre programme C.
- Une traduction en C des structures de données que vous utilisez.

— La liste complète de toutes vos fonctions avec, pour chacune d'elle, les éléments suivants.
  — L'entête complète de la fonction telle qu'elle apparaît dans le listing.
  — La description des paramètres d'entrée avec leurs noms et significations.
  — La description du *résultat* de cette fonction ainsi que la description des paramètres d'entrée qui peuvent être modifiés pendant son exécution.
  — Une description sommaire de ce que fait cette fonction.
— Un graphe de dépendance des fonctions entre elles. Chaque fonction devra être représentée dans ce graphe. Vous placerez un arc d'une fonction f_1 vers une fonction f_2 si f_1 fait appel à f_2 dans son code.
— Des jeux de tests de votre programme.

1.2 Document à rendre : contenu forme et date

Vous devrez rendre un compte rendu de votre travail sous la forme d'un rapport. Celui-ci devra obligatoirement contenir les éléments suivants.

— Une introduction présentant le problème.
— Une présentation des algorithmes
— Les divers documents demandés à la section 1.1.1.
— Une conclusion synthétisant le travail fait ainsi que les difficultés rencontrées et les extensions possibles.

2 Conseils pour le rapport

1. Un code est lisible, ce qui prend plusieurs aspects : il est commenté, il est structuré, il est autant que possible simple (peu d'astuces, des choix d'identificateurs explicites, pas trop de niveaux d'imbrications, . . .). En particulier toute astuce, tout choix algorithmique sophistiqué doit être justifié. Un code doit être auto-suffisant, et doit pouvoir être maintenu, remanié par une autre personne que la personne l'ayant rédigé (en particulier, par l'étudiant en binôme). Qui plus est, il doit être lisible par une personne non experte du langage de programmation.
2. Un jeu de tests (commenté) met en lumière les cas nominaux où cela marche, les cas particuliers, les cas aux limites, et éventuellement les cas de figure pour lesquels une solution ad hoc a été adoptée, pour ne pas dire, un traitement d'exception expéditif. Un jeu de tests est autant que possible complet : cela signifie que toutes les fonctionnalités importantes sont testées, qu'une stratégie de test a été appliquée pour choisir les tests, . . .
3. Un rapport synthétise et justifie les choix de conception effectués lors du projet. Il peut s'agir des choix de structures de données, des choix algorithmiques, des choix d'organisation logicielle (si le projet est découpé en plusieurs grandes fonctionnalités), des choix de répartition de travail dans le binôme, et surtout des choix relatifs à l'élargissement ou à la restriction du sujet initial (ou tout au moins relatifs aux contours du sujet).
4. Autant pour avoir une note correcte, il convient de traiter le sujet comme il a été demandé dans le sujet, autant pour bénéficier d'un petit plus, il peut être bénéfique de compléter le projet soit par une étude de performances comparées, soit par l'ajout de remarques pertinentes quant à l'intérêt du sujet, aux extensions possibles du projet, à la remise en cause d'une solution mise en œuvre, ... Ce point souligne en fait qu'une certaine originalité est souvent appréciée, dès lors que l'essentiel du contrat est rempli.

5. Le rapport doit être synthétique (et donc ne pas dépasser en pratique 20 pages) et dans la mesure du possible écrit en bon français. De plus, il doit être lisible sans faire référence sans cesse au code : on peut donc y insérer un passage du code que l'on considère comme clé pour étayer le commentaire.
6. Un projet peut recéler des difficultés cachées. Le commencer deux jours avant la date limite est donc un jeu dangereux, même si on est sûr de ses compétences techniques.
7. Il est bien sûr important que le projet réponde au sujet et pas à côté et que bien sûr, cela marche ! ! Ne pas hésitez à demander des compléments d'informations aux encadrants sur les objectifs du projet. Le moment le plus favorable est à la fin d'un cours et sinon sur rendez-vous (venez en groupe pour éviter de demander plusieurs fois la même chose).