intitule

Groupe s180498-s170220: Martin RANDAXHE, Cyril RUSSE

Table des matières

1 Le code source

Le code source sera fourni en C “standard” et utilisera les bibliothèques usuelles. Il sera correct, efficace et intelligible. Votre code doit pouvoir être compilé, sans erreur (ni ‘warning’), sous gcc. Si des options particulières sont nécessaires à la compilation, par exemple –std=c99, il est indispensable de le mentionner en préambule (ou de fournir un Makefile). Un code ne compilant pas entraine une note de zéro au projet. Une alternative est de fournir le code source en Python 3 “standard”. On peut utiliser des modules usuels mais il faut bien évidemment coder les parties principales inhérentes au projet choisi (et ne pas utiliser une libraire toute faite). On peut par exemple utiliser NetworkX pour les manipulations basiques de graphes. Quelques consignes qu'il est indispensable de respecter : — Le choix des noms de variables et de sous-programmes doit faciliter la lecture et la compréhension du code. — L'emploi de commentaires judicieux est indispensable : entrée/sortie des différents sous-programmes, points clés à commenter, boucles, etc. — Enfin, l'indentation et l'aération doivent aussi faciliter la lecture de votre code en identifiant les principaux blocs. Un code peu clair, même si le programme “tourne”, sera pénalisé. Une interface rudimentaire graphes.c/graphes.h est disponible en ligne sur http ://www.discmath.ulg.ac.be/. Celle-ci est détaillée à la fin des notes de cours (chapitre V). Libre à vous de l'utiliser ou non, voire de l'améliorer.

2 Le rapport

Le rapport ne doit pas être un copier-coller du code source (ce dernier étant fourni par ailleurs). Le rapport, au format pdf et idéalement rédigé sous L A TEX, est court : maximum 5 pages. Il doit décrire la stratégie utilisée, les choix opérés, les grandes étapes des différentes procédures ou fonctions. Il pourra aussi présenter les difficultés/challenges rencontrés en cours d'éla- boration ou reprendre certains résultats expérimentaux (benchmarking sur des exemples types ou générés aléatoirement).

3 La présentation orale

La présentation est limitée à 10 minutes maximum. Sans que cela soit né- cessaire, les étudiants ont le droit d'utiliser un ordinateur (pour faire tourner leur programme, pour présenter leur code, pour présenter leur travail avec un support type “power point”). Un projecteur vidéo est à disposition. Cette présentation se veut être une synthèse/explication du travail fourni. Elle est suivie par une séance de questions. Le but de ces questions est de déterminer la contribution et l'implication de chacun. Ainsi, des ques- tions différentes seront posées individuellement et alternativement aux deux membres du groupe. L'ensemble présentation/questions ne devrait pas dé- passer 20 minutes. Un ordre de passage des différents groupes sera déterminé.

4 Dates importantes

— lundi 12 octobre 2020 : choix individuel des modalités d'examen, ré- partition en groupes et choix des sujets. — vendredi 11 décembre 2020 : dépôt du code et du rapport sous forme d'une archive envoyée par mail au titulaire du cours. Cette archive contiendra deux répertoires, un pour le code à compiler, l'autre pour le rapport. — lundi 14 décembre 2020 — ordre de passage à déterminer : présentation orale. — janvier 2020 : examen écrit (commun pour tous).

5 Les projets

Sauf problème, les étudiants proposent une répartition par groupes de deux (en cas d'un nombre impair d'étudiants, un unique groupe de 3 étu- diants sera autorisé) et l'attribution des sujets aux différents groupes (un même sujet ne peut pas être donné à plus de 2 groupes — le sujet choisi par l'éventuel groupe de 3 étudiants ne peut être attribué à un second groupe). La répartition devra être validée par le titulaire du cours. Si un accord entre les étudiants n'est pas trouvé, le titulaire procédera à un tirage au sort (des groupes et des sujets). Le plagiat est, bien entendu, interdit : il est interdit d'échanger des solu- tions complètes, partielles ou de les récupérer sur Internet. Citer vos sources ! Néanmoins, vous êtes encouragés à discuter entre groupes. En particulier, il vous est loisible d'utiliser des fonctions développées par d'autres groupes (et qui ne font pas partie du travail qui vous est assigné). Mentionner les sources utilisées/consultées. Les projets listés ci-dessous sont "génériques", il est loisible à chaque groupe d'aller plus loin, d'adapter et de développer plus en avant les fonc- tionnalités de son code (par exemple, meilleure gestion des entrées/sorties, optimisation des structures de données, fournir des exemples "types" dans un fichier, etc.). Vérifiez que votre solution tourne même sur les cas pathologiques (par exemple, quel est le comportement attendu, si le graphe fourni n'est pas connexe, ne satisfait pas aux hypothèses, si le fichier est mal structuré, etc.). Essayez de construire un ensemble "témoin" de graphes "tests" sur lesquels faire tourner votre code. Tous les projets n'ont pas la même difficulté, il en sera tenu compte pour la cotation.

ENONCE :

Un graphe (simple, non orienté) est k-dégénéré s'il est possible de supprimer un à un ses sommets de telle sorte que chaque sommet supprimé soit de degré au plus k dans le sous-graphe induit par les sommets restant. Ce projet comporte deux parties : une procédure de test pour déterminer si un graphe est k-dégénéré (k est un paramètre) et en cas de réponse positive, fournir une suite convenable de sommets à supprimer. Ensuite, générer des graphes k-dégénérés maximaux à n sommets (k, n sont des paramètres). La maximalité signifie que le graphe obtenu est k-dégénéré et que si l'on ajoute une quelconque arête, il n'a plus cette propriété.

CONSEILS :

— Pensez à l'utilisateur qui teste votre programme : préparer un makefile, donner des conseils sur l'utilisation (fournir quelques fichiers de test), quelles entrées fournir, quelles sorties attendues ? Décrivez un exemple typique d'utilisation. — Préparez une petite bibliographie, citer les sources utilisées (même les pages Wikipédia !). Si vous avez exploiter une source, un autre cours, mentionnez-le explicitement !

— Avez-vous tester votre programme sur de gros graphes ? De quelles tailles ? Eventuellement produire un petit tableau de "benchmarking" indiquant, sur une machine donnée, le temps de calcul en fonction des tailles de graphes testés. — Relisez (et relisez encore) votre rapport ! Faites attention à l'ortho- graphe (accords, conjugaison), au style. — Si vous développez des heuristiques, avez-vous des exemples de graphes (ou de familles de graphes) qui se comportent mal par rapport à cette heuristique ?