Projet programmation 2021

Le but du projet est de d'analyser la base de données des articles de recherche en informatique provenant du site `dblp.org`. Votre programme devra ouvrir la base de données, la parser, la stocker dans une structure adaptée, la stocker dans un autre fichier au format binaire adapté et l'analyser à l'aide d'algorithmes de graphe. L'analyse permetra notamment de calculer le plus court chemin entre deux auteurs. Dans ce contexte, un chemin est une suite d'auteurs `(a_1, a_2, ..., a_n)` telle que, pour tout `1 <= i < n`, il existe un article où `a_i` et `a_{i+1}` sont co-auteurs.

Votre programme doit accepter des arguments de ligne de commandes nomminatifs (avec getopt ou autre), et accepter au moins un argument `-i BASE_DE_DONNEES`, indiquant le chemin vers le fichier xml de la base de données (ou vers le fichier binaire que vous aurez généré à partir de la base de données dans votre propre format). Des options supplémentaires seront ajoutées en fonction de ce que vous implémentez.

Le projet doit s'effectuer seul ou en binôme. Vous devez écrire un rapport (maximum 8 pages) expliquant rapidement la structure de votre programme et vos choix d'implémentation. Le code source et le rapport doivent être déposés sur un projet git sur le git de l'unistra (l'enseignant doit être ajouté dans les membres "maintainer" du projet pour y avoir accès). Le rapport et le code source doivent être déposés sur le git du projet avant le 23 Janvier 2022, 23h59.

Attention, si vous utilisez du code source trouvé sur internet, vous devez le citer clairement dans le code source et vous devez comprendre ce que vous avez copié (je me réserve le droit de vous poser des questions dessus). La difficulté du projet dépend de la quantité de choses que vous aurez implémentée vous-même et est prise en compte dans la notation.

La grille de notation tient compte de la clareté du code, de la structure, des choix d'implémentation, des tests et de la quantité de choses réalisées.

Notamment vous devez:

- organiser vos fichiers correctement (dans plusieurs dossiers).
- écrire un makefile avec des règles appropriées.
- rendre un projet qui compile sans erreurs ni warning (en activant tous les warning), et sans erreur valgrind.
- écrire des tests séparés qui permettent de vérifier que vos fonctions sont correctes.
- prendre en charge le signal SIGINT (qui se produit lors d'un "Ctrl+C").

Détail sur le projet

Votre programme doit parser le fichier dblp.xml disponible sur le site dblp.org. Le fichier étant très volumineux, il est fortement conseillé de se "fabriquer" un fichier plus petit possédant les mêmes caractéristiques afin de tester le fonctionnement de votre programme. **Attention, pensez à mettre un fichier .gitignore dans votre projet qui contient le chemin vers le fichier .xml, afin qu'il ne soit pas ajouté par git. Si le fichier .xml apparait lorsque vous faites un `git status`, c'est qu'il n'est pas correctement ignoré**.

L'étape de parsage du fichier doit permettre de stocker les informations pertinentes du fichier dans des structures (on peut ainsi ignorer beaucoup d'informations). Le parsage peut être spécifique au fichier mais il serait préférable que cette étape puisse tolérer des petites variations dans le format, ou en tout cas, afficher des erreurs lisibles en cas de problème de format (s'il manque une balise fermante par exemple).

Une fois le parsage terminé, votre programme doit être capable de sauvegarder la structure dans un fichier binaire (indiqué avec une option `-o OUTPUT_FILE`) ayant un format bien défini que vous devrez documenter. Cette étape permettra notamment de recharger la structure en mémoire beaucoup plus rapidement que de le faire depuis le fichier xml.

Une fois la structure en mémoire, vous devez implémenter des algorithmes de graphes sur le graphe des auteurs. Chaque algorithme est exécuté en donnant une option de ligne de commande. Par exemple

- l'option `-c` pourra correspondre au calcul du nombre de composantes connexes avec leur diamètre.
- l'option `-p AUTEUR1 -p AUTEUR2` permettra de calculer un plus court chemin entre les deux auteurs indiqués.
- l'option `-l MOT` permettra de lister les auteurs qui contiennent le mot donné.
- l-option `-a AUTEUR` permet d'obtenir les informations d'un auteur (liste des articles).
- l-option `-a AUTEUR -n N` permet d'obtenir les co-auteurs à distance `N` (ou moins) d'un auteur.

D'autres options peuvent être ajoutées afin, par exemple, de ne prendre en compte que les articles dans une période donnée (délimitée par des années passées en options) ou ne considérer les liens que lorqu'il y a plus de `X` articles en commun, ou bien de calculer d'autres métriques de graphe (la centralité le coefficient de clustering, ...).

Vous pourrez aussi implémenter un affichage graphique afin de visualiser le chemin, ou la liste des voisins d'un auteurs, etc...

Tests

Le projet doit avoir un Makefile avec une règle clean et dont la première règle permet de compiler l'exécutable principal.

le dossier tests doit contenir un Makefile permettant de compiler un programme `tests.c` qui teste les différentes fonctions que vous avez implémentées. Il pourra aussi y avoir un script `test.sh` qui exécute le programme sur des fichiers de tests afin de vérifier que la sortie est correcte.