CPE Lyon - 3ICS - Année 2020/21

Développement informatique - 1

TP 2 - Algorithmes de tri

1/ Travail à faire

Dans ce TP vous allez devoir :

- Implémenter 4 algorithmes de tri (tri a bulle, tri par sélection, tri par insertion, tri pas tas) permettant un tri croissant ou décroissant d'un tableau de nombre a virgules (float)

- Écrire le code permettant de faire des mesures du temps d'exécution de chacun de ces quatre algorithmes en fonction de la taille des tableaux à trier. Les résultats des benchmarks permettront d'avoir une estimation de la complexité temporelle.

- Stocker les résultats des tests dans un fichier au format csv permettant d'utiliser un grapheur (excel, gnuplot, ...)

- Générer une documentation technique à partir des annotations et documentations présentes dans le code.

- Écrire le fichier README.md décrivant l'objet du projet, l'usage des commandes, les résultats attendus et les évolutions à venir.

Pour que les résultats soient probants : vous devez respecter 4 contraintes :

- Les tableaux avant tri doivent être les mêmes pour chaque algorithme testé (cf notion de nombre pseudo aléatoire et de graine)
- Chaque algorithme devra être testé 3 fois avec des tableaux différents. Vous ne garderez que la valeur moyenne des trois tests.
- Vous devez trier des tableaux contenant des valeurs aléatoires comprises entre 0 et 10^6
- Vous devez réaliser les tests avec des tableaux contenants respectivement 100, 10^3, 10^4, 10^5,10^6,10^7 valeurs

2/ Consignes de Travail

- Le projet C doit se nommer Benchme.
- Le travail est a réalisé en binôme (un étudiant issu de la filière développement avec un étudiant issu de la filière réseau). Le travail devra être reparti équitablement.

-Le code doit être versionné sur le serveur Gitlab.

- Vous devrez vous connecter sur le serveur, créez un compte avec vos nom et prénom réels.
- Vous donnerez un accès total au dépôt créer à votre binôme.
- Vous êtes invité à utiliser le workflow « gitflow »

- Le code doit respecter les bonnes pratiques de programmation.
- Le code doit être structuré. Vous créerez plusieurs fichiers (.c et .h). La répartition du code dans les fichiers doit être pertinente.

-Le code doit être évolutif (ajout à terme d'un nouveau algorithme de tri ?, changement de la taille des tableaux de valeurs aléatoires ?)

-Le code doit être correctement indenté, commenté (selon les besoins) et systématiquement documenté/annoté. Vous pouvez utiliser les annotations au format javadoc ou au format doxygen.

-La documentation technique doit pouvoir être extraite à la demande grâce a une commande que vous aurez pris le soin d'écrire dans le Makefile (→ rappel : en C la commande « `make all` » permet par exemple de lancer la compilation de tous les fichiers .c , « `make clean` » permet de nettoyer les répertoires contenant le code sources.) . Cette commande sera « `make documentation` »

-Le programme que vous allez écrire prendra en argument le nom du fichier contenant les résultats du benchmark i.e. « `benchme data.txt` » ou encore « `benchme result.csv` »