Cours de programmation séquentielle

Mesure de complexité

1 Buts

- Implémentations d'algorithmes de tris.
- Mesure de complexité et de performances.
- Utilisation de `make`.

2 Énoncé

L'objectif de ce travail pratique est d'implémenter les tris vus en cours et de mesurer leurs performances respectives. En particulier, vous devez écrire les codes en C:

- du tri par base,
- du tri par fusion,
- du tri par sélection,
- du tri rapide,
- du tri à bulles,
- du tri par insertion,
- du tri cocktail.

A l'exception du tri cocktail, les (pseudo-)codes de ces tris se trouve dans les slides du cours 7 et du cours 8. Pour le tri cocktail, une bonne description se trouve sur Wikipedia.

La structure de votre code devrait être la suivante:

```
.
├── cocktail_sort.h
├── cocktail_sort.c
├── merge_sort.h
├── merge_sort.c
├── quick_sort.h
├── quick_sort.c
├── bubble_sort.h
├── bubble_sort.c
├── radix_sort.h
├── radix_sort.c
├── insertion_sort.h
├── insertion_sort.c
├── selection_sort.h
├── selection_sort.c
```

```
├── utils.h
├── utils.c
├── sort.c
└── Makefile
```

Il est **impératif** de créer un `Makefile`, car la quantité de fichiers à gérer est beaucoup trop grande pour que la compilation puisse être gérée manuellement.

Le point d'entrée de votre programme doit être contenu dans le fichier `sort.c`. Il doit générer un tableau de taille `N` rempli de valeurs aléatoires et les trier à l'aide d'un des tris ci-dessus. Finalement, il doit vérifier que le tableau est correctement trié.

L'utilisation de votre programme devrait être de la forme

```
./sort <num> <N>
```

où `num` est le numéro correspondant au tri, et `N` est la taille du tableau à trier.

En sortie, votre programme doit afficher le temps qu'il a fallu pour effectuer le **tri** du tableau (on exclut l'allocation et initialisation du tableau). Pour mesurer le temps d'exécution vous pouvez vous inspirer du cours 7.

Le contenu de la plupart des autres fichiers contiennent les tris à proprement par

Vous remarquez probablement l'existence des fichiers `utils.h` et `utils.c`. Ces fichiers doivent contenir les fonctions **utilitaires** de votre application telles que l'affichage d'un tableau, l'initialisation du tableau, l'échange de valeurs, etc.

Une fois que vous avez implémenté ces fonctionnalités et les tris, vous devrez faire des graphes de performances (le temps d'exécution) de chaque tri en fonction de la taille des tableaux à trier (la taille du tableau sur l'axe horizontal, et le temps d'exécution sur l'axe vertical). Vérifiez que la complexité algorithmique est bien celle prédite dans le cours (en gros soit ça sera N^2 soit $N \cdot \log_2(N)$).

2.1 Remarques

2.1.1 Options de compilation

Afin de mesurer les performances, utilisez l'option `-O3` de `gcc` ou `clang` pour que le compilateur optimise votre code au maximum.

2.1.2 Mesures de performances

Afin de mesurer les performances de façon aussi précise que possible il est recommandé de faire deux choses importantes:

1. Faire un *tour de chauffe* de votre algorithme de tri. Avant de faire la mesure, triez plusieurs fois de suite le tableau sans mesurer le temps.
2. Une fois le tour de chauffe terminé triez plusieurs fois le tableau en mesurant le temps d'exécution et moyennez le temps pris par le tri.