Structure de données et algorithmes
Projet 1: Algorithmes de tri

Pierre Geurts – Jean-Michel Begon - Romain Mormont

23 février 2018

L'objectif du projet est d'implémenter, de comparer et d'analyser les algorithmes de tri suivants:

(a) L'algorithme InsertionSort (fourni).

(b) L'algorithme QuickSort (à implémenter).

(c) L'algorithme MergeSort (à implémenter).

(d) L'algorithme HeapSort (à implémenter).

(e) L'algorithme OtherSort (voir Section 2, à implémenter).

1 Algorithmes vus au cours: analyse expérimentale

Dans cette partie du projet, vous allez étudier les performances de vos implémentations de InsertionSort, QuickSort, MergeSort et HeapSort. Dans votre rapport, nous vous demandons de répondre aux questions suivantes:

1.a. Soit N le nombre de données à trier dans un tableau. Calculez empiriquement le temps d'exécution moyen de ces quatre algorithmes pour différentes valeurs de N (10, 100, 1.000, 10.000, 100.000 et 1.000.000) lorsque les tableaux sont générés de manière complètement aléatoires. Reportez ces résultats dans une table au format donné ci-dessous.

N	InsertionSort	QuickSort	MergeSort	HeapSort
10				
100				
1.000				
10.000				
100.000				
1.000.000				

1.b. Commentez ces résultats en comparant les algorithmes :

- les uns par rapport aux autres
- par rapport à leur complexité théorique

Remarques:

- Les fonctions pour générer les tableaux vous sont fournies dans le fichier `Array.c`.
- Les temps reportés doivent être des temps moyens établis sur base de 10 expériences.

2 Un nouvel algorithme de tri: OtherSort

Fort de vos nouvelles connaissances en algorithmique, vous avez décroché un poste chez *Sort & Cie.*, entreprise spécialisée dans la création d'algorithmes de tri de tableaux. Vous venez d'imaginer l'algorithme suivant:

(i) Si le tableau a une taille inférieure ou égale à 1, ne rien faire.

(ii) Si l'élément le plus à gauche du tableau est plus large que l'élément le plus à droite, les échanger

(iii) S'il y a au moins trois éléments dans le tableau, (i) on trie les premiers deux tiers[1] du tableau récursivement, (ii) on trie les derniers deux tiers du tableau, (iii) on retrie les premiers deux tiers à nouveau.

Avant de suggérer l'algorithme à votre patron, vous décidez d'en faire une analyse fouillée pour vérifier l'intérêt de l'algorithme:

2.a. Donnez le pseudo-code de OtherSort.

2.b. Prouvez que cet algorithme est correct.

2.c. Est-ce que cet algorithme de tri est stable ? Est-il en place ? Justifiez vos réponses.

2.e. Etudiez expérimentalement la complexité en temps de cet algorithme. Pour ce faire, suivez le même protocole qu'à la section 1 et ajoutez une colonne pour OtherSort. Comment se compare OtherSort avec les autres algorithmes?

2.d. Etudiez et justifiez la complexité en temps et en espace de OtherSort. Comparez avec les temps mesuré à la question précédente.

2.f. Concluez quant à l'intérêt pratique de ce nouvel algorithme de tri.

Les réponses à ces questions doivent être présentes dans votre rapport.

Deadline et soumission

Le projet est à réaliser **en binôme** pour le **vendredi 23 mars 2018** à **23h59** au plus tard. Le projet est à remettre via la plateforme de soumission de Montefiore: `http://submit.montefiore.ulg.ac.be/`.

Il doit être rendu sous la forme d'une archive `tar.gz` contenant:

(a) Votre rapport (5 pages maximum) au format PDF. Soyez bref mais précis et respectez bien la numération des (sous-)questions.

(b) Un fichier `QuickSort.c`.

(c) Un fichier `MergeSort.c`.

(d) Un fichier `HeapSort.c`.

(e) Un fichier `OtherSort.c`.

Respectez bien les extensions de fichiers ainsi que les noms des fichier `*.c` (en ce compris la casse). Les fichiers suivants vous sont fournis:

[1] Si la taille du tableau n'est pas divisible par 3, les tailles des sous-tableaux sont arrondies vers le haut.

- `Array.h` et `Array.c`: une petite bibliothèque pour générer différents types des tableaux.
- `Sort.h`: le header contenant le prototype de la fonction de tri.
- `InsertionSort.c`: implémentation de l'algorithme de InsertionSort donnée à titre d'exemple pour vos propres implémentations.
- `main.c`: un petit fichier de test.

Vos fichiers seront évalués sur les machines ms8xx avec la commande[2]:

```
gcc main.c Array.c InsertionSort.c --std=c99 --pedantic -Wall -Wextra -Wmissing-prototypes
-lm -o test
```

Ceci implique que:

- Le projet doit être réalisé dans le standard C99.
- La présence de *warnings* impactera négativement la cote finale.
- Un projet qui ne compile pas avec cette commande sur ces machines recevra une cote nulle (pour la partie code du projet).

Un projet non rendu à temps recevra une cote globale nulle. En cas de plagiat avéré, l'étudiant se verra affecter une cote nulle à l'ensemble des projets.

Les critères de correction sont précisés sur la page web des projets.

Bon travail !

[2]En substituant adéquatement `InsertionSort.c` par l'implémentation de l'algorithme que nous voulons tester.