Laboratoire : Les algorithmes de tri

1 Objectifs

Le but de ce laboratoire est de découvrir et implémenter divers algorithmes de tri.

2 Travail à faire

Vous devez implémenter les tris suivants :

1. Tri rapide (*quicksort*)
2. Tri par tas (*heapsort*)
3. Tri fusion (*merge sort*)
4. Tri par insertion (*insertion sort*)
5. Tri par sélection (*selection sort*)
6. Tri Shell (*Shell sort*)
7. Tri bulle amélioré (*cocktail sort*)

De plus, nous vous posons les restrictions suivantes :

- Pour le tri rapide, le pivot est le centre du tableau.
- Pour le tri par tas, le monceau doit être créé à même la liste passée en paramètre.
- Vous devez utiliser un minimum d'espace mémoire. Pour les algorithmes récursifs, vous devez utiliser des bornes de gauche et droite pour délimiter le sous-tableau dans la récursion.
- Les « gaps » pour le tri Shell doivent être : $[1, 3, 7, 15, 31, 63, \ldots, 2^{\lfloor \log_2(N) \rfloor} - 1]$. La récursion utilisée est $2^k - 1, k > 0$. Ces « gaps » assurent une complexité en $O(N^{1.5})$.

Basez-vous sur les tests *Google Test* fournis pour développer de nouveaux tests évaluant la performance de chacun des tris implémentés.

3 Questions supplémentaires

Répondez aux questions suivantes :

- Quel(s) tri(s) sont stable(s) ?
- Quelle est la complexité en pire cas de chacun des tris ? Trouvez un exemple d'une liste à trier qui correspond à ce pire cas.

4 Documentation supplémentaire

Wikipédia offre une excellente documentation et des démonstrations des tris à implémenter. Consultez les différentes pages au besoin :

1. *Quicksort*
2. *Heapsort*
3. *Merge Sort*
4. *Insertion sort*
5. *Selection Sort*
6. *Shell sort*
7. *Cocktail sort*

5 Documentation

Voir la section Documentation/Normes sur le site Web du cours. Vous y trouverez la description des commentaires attendus dans un programme ainsi que des normes de programmation en vigueur dans notre cours.

6 Important

1. Nous vous fournissons des tests unitaires *Google Test* que vous pouvez utiliser pour vérifier votre implémentation.

2. Vous êtes tenu de faire la gestion des exceptions dans les méthodes que vous avez à implémenter. Référez-vous à la documentation *Doxygen* fournie avec l'énoncé pour connaître les types d'exception que vous avez à gérer pour chacune d'elles. Vous devez utiliser le cadre de la théorie du contrat que nous avons préparé dans les fichiers `ContratException.cpp` et `ContratException.h` disponibles sur le site du cours.
3. Vous devez documenter chaque méthode, que vous avez à implémenter, avec les commentaires de *Doxygen*. Référez-vous à section de Doxygen sur le site Web du cours ainsi qu'aux exemples de cette semaine pour découvrir les commentaires que vous avez à écrire. Assurez-vous de respecter les normes de programmation en vigueur disponibles également sur le site du cours.
4. Vous devez générer la documentation en format HTML. À cette fin, nous vous fournissons un fichier de configuration `sdd.doxyfile` que vous pouvez utiliser tel quel.
5. Nous vous encourageons à utiliser au maximum la partie privée de la classe afin d'y ajouter des fonctions utilitaires privées. Elles permettent d'augmenter la lisibilité du code et de réduire la duplication de code identique.

Bon travail !