Saé 1.02 : Comparaison d'approches algorithmiques

L'objectif de cette Saé est de comparer plusieurs approches pour la résolution de problèmes et d'effectuer des mesures de performance simples.

Le rendu est :
- le code Java des algorithmes implantés ;
- des tests JUnit montrant que vos programmes fonctionnent ;
- un document décrivant les algorithmes implantés, leur fonctionnement et leurs caractéristiques, et discutant leurs performances.

L'évaluation sera complétée par un entretien pendant lequel vous devrez présenter le travail que vous aurez réalisé et les résultats que vous aurez obtenus.

L'évaluation prendra en compte la qualité du code et des tests, la qualité du document rendu, la qualité des explications orales et réponses aux questions et l'ampleur du travail réalisé.

Les questions marquées (fondamental) permettent d'obtenir la moyenne.

Chaque fois que vous expérimentez un temps de calcul, faites plusieurs essais. Vous indiquerez dans votre compte rendu le nombre d'essais réalisés, les temps observés ainsi que la moyenne.

Question préliminaire (fondamental)
Récupérez sur Moodle le programme SAé2.java.
Lisez-le, exécutez-le.
Combien de temps la méthode Java prédéfinie *Arrays.sort(int[])* prend-elle pour trier un tableau de longueur 200 000 ?
Combien de temps prend-elle pour trier un tableau de longueur 10 000 000 ?

Question 1 (fondamental)
Programmez vos propres méthodes de tri en implantant deux algorithmes (au choix) parmi les algorithmes suivants : tri par sélection, tri par insertion, tri à bulles, tri cocktail.
Combien de temps vos méthodes de tri prennent-elles pour trier un tableau de longueur 200 000 ?
Sachant que ces tris ont tous une complexité en moyenne de n^2 (lorsque la taille du tableau à trier est multipliée par n le temps de calcul est multiplié par n^2), estimez combien de temps prendraient vos méthodes de tri pour un tableau de longueur 10 000 000.
Discutez les résultats obtenus.

Question 2 (avancé)
Le tri à peigne est une variante du tri à bulles qui améliore fortement ses performances. Programmez un tri à peigne, évaluez ses performances, discutez les résultats obtenus.

Question 3 (fondamental)
Le tri comptage est très efficace, mais il ne fonctionne que si l'on connaît la plus petite et la plus grande valeur du tableau (et que l'intervalle entre la plus petite et la plus grande valeur est de taille raisonnable). Programmez un tri comptage, évaluez ses performances, discutez les résultats obtenus. Vous pourrez vous intéresser aussi à la performance en utilisation d'espace mémoire.

Question 4 (avancé)
On souhaite obtenir les n plus grandes valeurs d'un tableau. Une méthode naïve consiste à trier le tableau en utilisant *Arrays.sort* pour ensuite y récupérer les n derniers éléments. Cela fonctionne bien sur des tableaux de taille petite ou moyenne. Cependant, sur des très grands tableaux, trier l'intégralité du tableau prend inutilement du temps.

Programmez une méthode qui retourne les n plus grandes valeurs d'un tableau de int, évaluez ses performances, discutez les résultats obtenus.
L'objectif est d'obtenir les 100 plus grandes valeurs d'un tableau de longueur 100 000 000 plus rapidement qu'en utilisant *Arrays.sort.*

Vos programmes seront mis en compétition, les plus rapides gagneront des points bonus : 3 points pour le premier, deux points pour le second, un point pour les cinq suivants.

Question 5 (avancé)
Programmez dans un autre langage (par exemple Python ou C) un des algorithmes que vous avez implanté. Comparez et discutez ses performances avec celles de la version en Java.