TP5: Tris

1 Préliminaires

Procédez comme pour les TPs précédents pour **importer** le projet `/share/l1ie/SI1/TPtri/TPtri.zip`

2 Proposer un algorithme (correct) de tri

Définir une fonction qui trie par ordre croissant un tableau d'entiers. La signature de cette fonction sera

```
void tri(int[] t)
```

Pour vous assurer de la correction de votre fonction :

1. sélectionner des cas de tests pertinents ;
   - **il faut** commencer par des tests sur des petits tableaux (taille 0, puis 1, puis 2, puis 3, etc.). Les bugs seront plus faciles à localiser et à corriger sur des petits exemples ;
   - **il est inutile** de tester plusieurs fois des cas similaires ;
   - des tableaux non triés, des tableaux triés ;
   - des tableaux avec des répétitions ;
   - etc. creusez vous la tête !
   - pour vérifier si une fonction a été suffisamment testée avec JUnit, vous pouvez utiliser Pitest. Regardez le tutoriel vidéo. Pour lancer la couverture de tests, faire un clic droit sur le fichier `MainTest.java` sélectionner "Run as>PIT Mutation Testing".
2. écrire les tests JUnit correspondants (voir TP1). Pour tester l'égalité de deux tableaux `t1` et `t2` il faut utiliser `assertArrayEquals(t1,t2);`
3. tester de façon automatique avec JUnit ;
4. rechercher les erreurs éventuelles dans votre programme à l'aide du débugger (voir TP1).

On rappelle qu'il ne faut **jamais** effacer un test même si celui-ci passe !

3 Estimation de la complexité et optimisation de l'algorithme de tri

Améliorez autant que possible les performances de votre algorithme :

1. En utilisant `ACX.tracerTriInt` et des fonctions de référence bien choisies, estimez la complexité de votre algorithme de tri ; Vous pouvez, pour cela, utiliser les fonctions suivantes :
   - `ACX.tracerTriInt(String[] fonctions)` trace les courbes de temps de calcul pour les fonctions de tri dont les noms figurent dans le tableau `fonctions`.

- `ACX.tracerTriInt(String[] fonctions, String[] fonctionsReference)` trace les courbes de temps de calcul pour les fonctions de tri dont les noms figurent dans le tableau `fonctions` et trace les courbes pour des fonctions de référence dont les noms figurent dans le tableau `fonctionsReference`. La signature des fonctions de référence doit être `int f(int x)`.

Remarque : si votre fonction de tri est rejetée par `ACX.tracerTriInt` mais passe vos tests, c'est que vous n'en avez pas écrit suffisamment ou qu'ils ne sont pas pertinents !

2. Tentez d'optimiser votre algorithme. Parfois, cela demande de changer d'algorithme ;
3. Vérifiez que vous repassez les tests JUnit !
4. et ainsi de suite... il existe des algorithmes dont les temps de calcul passent sous la courbe `refLinearithmique` ;

4 Bonus : automatisation des tests par le test aléatoire

Pour étendre les tests à des tableaux de taille plus importante, définissez une fonction qui permet de savoir si un tableau d'entiers est trié. La signature de cette fonction sera :

```
boolean estTrie(int[] t)
```

1. à l'aide de tests unitaires JUnit bien choisis, vérifiez que votre fonction `estTrie` est correcte.
2. testez automatiquement votre fonction `tri`, sur des tableaux pertinents, en générant aléatoirement des tableaux d'entiers (voir bonus TP1), en les triant, et en vérifiant que le tableau obtenu est accepté par `estTrie`.
3. enfin, utilisez la fonction `estTrie` dans vos tests unitaires des fonctions de tri programmées précédemment.
4. **Bonus :** la fonction `estTrie` n'est pas strictement suffisante pour vérifier que vos fonctions de tri sont correctes. Vous devez également vérifier que le tableau obtenu après le tri contient bien les mêmes éléments que le tableau de départ : le tableau trié doit être une permutation du tableau de départ. Comme vos fonctions de tri réalisent des tris en place, elles modifient le tableau initial. Pour réaliser un test automatique d'une fonction de tri, il faut donc :
   (a) Générer aléatoirement un tableau d'entiers `t` ;
   (b) Produire une copie de `t` nommée `tcopie` ;
   (c) Appeler la fonction de tri sur `t` ;
   (d) Vérifier que `estTrie(t)` est vrai ;
   (e) Vérifier que `t` est une permutation de `tcopie`.

   Nous vous suggérons donc de programmer une fonction de copie de tableau d'entiers, ainsi qu'une fonction vérifiant qu'un tableau est une permutation d'un autre. En suivant la méthodologie proposée plus haut, utilisez ces fonctions pour tester la correction de vos fonctions de tri.