TP 2 : Tris récursifs

Objectif du TP

L'objectif de cette séance est d'appliquer la récursivité sur les tris et de les manipuler. Vous apprendrez également à créer une classe de tests unitaires.

Recommandations

Immédiatement après chaque séance de TD/TP, chaque étudiant ou groupe d'étudiants enverra un mail à l'enseignant responsable du groupe un compte rendu de l'activité réalisée pendant la séance au format PDF dans lequel les réponses aux questions auront été saisies. Il est fortement conseillé de faire des copies d'écran de vos programmes pour aller plus vite.

Le sujet du mail devra obligatoirement respecter le format suivant : **[S3T ou S3D ou S3A][M313][Gx][Seance_y] Nom1 / Nom2** ou *x* représente le numéro de groupe et *y* le numéro de la séance (par exemple : mailto:denis.pallez@unice.fr?subject=[S3A][M313][G1][Seance2]Turing). Le fichier PDF devra également être renommé de la façon suivante : **<NomEtudiant1_NomEtudiant2>_TP<N° du TP>_<date de la séance de TP en anglais>.pdf** (par ex : Pallez_TP1_20141306.pdf).

Vous ne pouvez pas faire plus de 2 séances avec le même binôme et vous ne pouvez pas faire plus de la moitié des séances seul. Tout manquement à ces règles pourra entrainer des retraits de points sur la note de contrôle continu.

Exercice 1 Classe de tests pour les tris

a) Créez une nouvelle classe de tests `TriStandardTest` en cochant la case de création des différentes méthodes (cf. précédent TP) ;
b) Ajoutez-y un tableau d'entiers `nombres` (déclaration mais pas instanciation), les constantes `Taille` pour la taille du tableau à trier et `Valeur_Max` pour la valeur maximale que peut prendre un élément du tableau et un générateur de nombres aléatoire (classe `Random`) ;
c) Modifiez la méthode `setUp()` pour qu'elle instancie le tableau et le générateur de nombres ;
d) Créez une méthode `boolean triCroissantOK(int[])` qui précise si le tableau passé en paramètre est trié de manière croissante ou pas. Vous pourrez afficher sur la sortie standard le tableau avec l'instruction `System.out.println(Arrays.toString(tableau))` afin de vérifier que votre vérification est correcte ;
e) Créez la méthode de test `testVide` qui vérifie qu'un tableau vide (aucun élément n'a été initialisé) peut être trié. Pour trier, on utilisera la méthode `Arrays.sort()` ;
f) Créer la méthode de test `testUnSeulElement` qui vérifie qu'un tableau avec un seul élément peut être trié ;
g) Créer une méthode d'initialisation aléatoire `void initRandom()` ;
h) Créez la méthode de test `testRandomCase` qui initialise le tableau, le tri et vérifie qu'il est trié. Si le tableau n'est pas correctement trié après le tri alors le test devra faire une erreur (méthode prédéfinie *fail*). *Remarque : il sera difficile de vérifier le cas où le tri ne fonctionne pas car vous utilisez un tri qui fonctionne mais cela risque de ne pas être le cas plus tard quand vous développerez votre propre tri !*
i) Créez la méthode de test `testWorstCase` qui fait le même test que précédemment sauf que `nombres` sera initialisé déjà trié de manière décroissante (par ex : `nombres[i]=Valeur-Max-i`). Pensez à créer une autre méthode d'initialisation `initWorst()` ;

j) Créez la méthode de test `testBestCase` qui fait le même test que précédemment sauf que `nombres` sera initialisé déjà trié de manière croissante (par ex : `nombres[i]=i`) ;
k) Créez la méthode de test `testSameValuesCase` qui fait le même test que précédemment sauf que `nombres` sera initialisé avec plusieurs valeurs identiques à différent endroits dans le tableau (par ex : `[5,5,6,6,4,4,5,5,4,4,6,6,5,5]`) ;
l) Lancer le test sur cette classe. Augmenter le nombre d'élément dans le tableau. Que constatez-vous ?

Exercice 2 Comparaison des tris

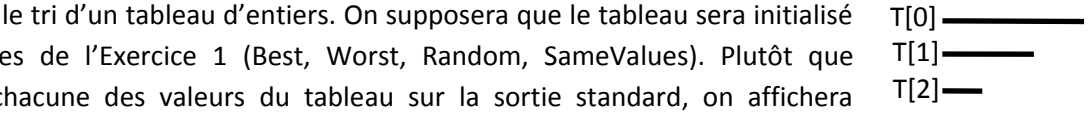
a) Créez une classe `Tris` qui contiendra 2 méthodes statiques `triRapide(int[])` et `triFusion(int[])` sans remplir les 2 méthodes.
b) Avant de développer les méthodes de tris, on souhaite mettre en place la classe de test. Ecrivez une classe de test `TrisTest` pour tester le tri par fusion dans chacun des cas (`WorstCase`, `BestCase`, `RandomCase`, `SameValuesCase`). La POO nous autorise plusieurs solutions :
   i. Faire une copie de `TriStandardTest`. Ce n'est pas forcément la solution la plus élégante ;
   ii. Faire un héritage de `TriStandardTest`. Mais dans ce cas, sachez que lorsque vous lancerez les tests sur `TrisTest` vous lancerez également les tests sur `TriStandardTest` ;
   iii. Déplacez les méthodes d'initialisation, le générateur de nombres et le tableau dans une nouvelle classe de base `TrisBase` qui ne fera aucun tests mais qui sera héritée par `TriStandardTest` et par `TrisTest` ;
c) Développer le tri par fusion vu en cours.
d) Ajouter des variables à l'algorithme de tris qui permettent de mesurer le nombre de copies de tableaux effectuées, le nombre d'échange effectués, le nombre d'affectation …
e) Lancez plusieurs tris sur des tableaux de tailles différentes (5, 10, 100, 500, 1000, 5000, 10000) et avec à chaque fois les 4 méthodes d'initialisation. Retenez pour chacun de ces tris, les valeurs des variables décrites en c) et tracez une courbe pour chaque type d'initialisation en fonction de la taille du tableau.

Exercice 3 Quicksort

a) Dans la classe `Tris`, écrivez une méthode `int[] partitionPivot(int[], int min, int max, int pivot)` qui se base sur `partitionV2OK` mais qui considère le prédicat `"< Pivot"`.
b) Implémentez le tri Rapide.
c) Créez les tests pour ce tri.
d) Introduisez également les variables introduites précédemment.
e) Changer la méthode de sélection du pivot et comparer avec la méthode précédente.

Exercice 4 Affichage graphique

On souhaite visualiser le tri d'un tableau d'entiers. On supposera que le tableau sera initialisé par une des méthodes de l'Exercice 1 (Best, Worst, Random, SameValues). Plutôt que d'afficher bêtement chacune des valeurs du tableau sur la sortie standard, on affichera graphiquement un segment dont la longueur est proportionnelle à la valeur d'un élément du tableau. Par exemple, si T=[3,2,1], alors le programme devra afficher la zone graphique ci-contre et la faire évoluer tout au long du tri du tableau afin de visualiser graphiquement le tri.



Pour aller plus loin

Exercice 5 Le tri par fusion complétement récursif

Le cours a proposé un tri par fusion avec une fusion des 2 tableaux en itératif. Programmer une méthode de fusion complètement récursive.

Sources pour ce TP

Tutoriel Eclipse – Junit : Mon premier test automatiques

Quicksort in Java – Tutorial, Lars Vogel