Université Paris 7 - Denis Diderot
L1 Sciences

IF2 : Structures de données et Objets JAVA
Année 2008-2009, 2^{eme} semestre

Correction CC en TD

Durée de l'examen : 1h - le barème est donné à titre indicatif.
Aucun document n'est autorisé - Les exercices sont indépendants.

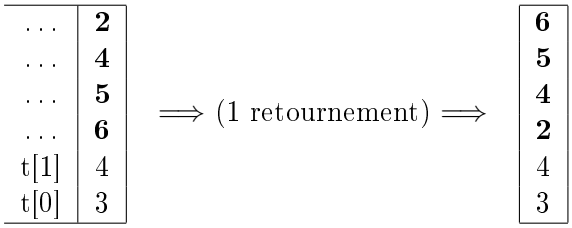
Exercice 1 ***Tri par insertion*** - 4 points

60	1	20	43	21

1. Déroulez le fonctionnement de l'algorithme du tri par insertion sur le tableau précédent. Bien représenter toutes les étapes.
2. Ecrire une méthode `static void triInser(int[] tab)` qui effectue le tri par insertion du tableau passé en argument.
3. Quel est l'invariant du programme ?
4. Prouver sa terminaison.
5. Quel est le pire cas en terme de nombre d'echanges ? Donner la complexité de l'algorithme dans le cas moyen, dans le meilleur cas.

`Tout cet exercice repose sur la connaissance du cours. Cf celui-ci.`

Exercice 2 Le crépier débutant - 4 points
Un crepier debutant n'a pas réussi à faire des crêpes de même taille, et il se retrouve devant un tas de crêpes de tailles différentes. Il veut les ranger en ordre croissant, les plus petites en haut. Pour ce faire il ne dispose que d'une operation : retourner le haut du tas de crêpes en insérant son palet en bois entre deux crêpes (ou entre l'assiète et la première crêpe).



On représente le tas de crèpes par un tableau d'entier. La crèpe en contact avec l'assiète est dans la case 0 du tableau.Les entiers contenus dans le tableau correspondent aux tailles des différentes crêpes.

1. Ecrire une méthode `public static void miroir(int[] t,int i)` qui renverse les éléments compris entre l'indice i (inclus) et la fin du tableau. l'appel `miroir(t,0)` doit donc réaliser l'inversion de tout le tableau.
2. Ecrire une méthode `public static int rechercheMax(int[],int d)` qui retourne l'indice du plus grand élement présent dans le tableau t dans l'intervalle [d,t.length-1]
3. Quelle méthode le crépier doit-il employer pour pouvoir effectuer le tri ? (expliquez le principe)

4. Ecrire une méthode récursive `public static void triCrepier(int [] tab,int d)` qui réalise ce tri.

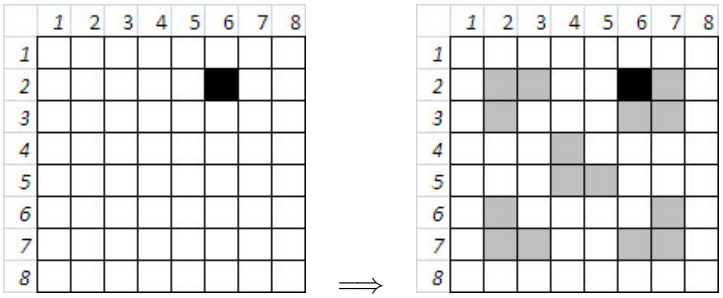
```
Principe du triCrépier :
1° recherche de la ième plus grande crèpe. On récupère son \texttt{indice}.
   Cas d'arret quand i=taille de la pile de crèpe.
2° insertion du palet en dessous de la crèpe grace à l'\texttt{indice} et
   on retourne tout le tas qui est au dessus.
   --> la ième plus grande crèpe est au sommet de la pile
3° on insère le palet à la position i ou doit se trouver la crèpe une fois
   rangée et on retourne le tas.
4° on incrémente i et on recommence.

//version récursive
public static void miroir(int [] t, int i){
         if (i<t.length/2){
                int x=t[i];
                t[i]=t[t.length-i-1];
                 t[t.length-i-1]=x;
                 miroir(t,i+1);
          }
        }

        //version non récursive
        public static void miroir2(int[] t, int i){
                int m=(t.length-i)/2;
                for(int k=0; k<m;k++){
                        int x=t[k+i];
                        t[k+i]=t[t.length-k-1];
                        t[t.length-k-1]=x;
                }
        }

        private static int rechercheMax(int [] t,int d){
                int indMax=d;
                or (int i=d+1;i<t.length;i++){
                        if (t[i]>t[indMax]){
                                indMax=i;
                        }
                }
                return indMax;
        }

        public static void triCrepier(int [] t,int d){
                if (d<t.length-1){
                        int ind=rechercheMax(t, d);
                        if (ind!=d){
                                miroir2(t,ind);
```

```
                    miroir2(t,d);
                }
                triCrepier(t,d+1);
            }
        }
```

Exercice 3 ***Pavage d'un tableau de n lignes par n colonnes*** - 3 points
On dispose d'un ensemble de motifs de la forme suivante :

A l'aide de ces motifs, on veux recouvrir totalement un tableau à l'exception d'une case spéciale de coordonnées (x,y) données. Les cases ne doivent être recouvertes qu'une seule fois.

Exemple : n=8, (x,y)=(2,6)



1. Donner un principe de la forme "diviser pour résoudre" permettant de résoudre automatiquement le problème pour l'exemple ci-dessus.
2. Quelle est la propriété dépendant de n qui doit être vérifiée pour que le problème admette une solution dans le cas général ?

```
 - Diviser le tableau de manière récursive en 4 sous tableaux égaux en placant
le motif de façon à ce que chaque sous-tableau est une seule case couverte (on
commence donc par placer le motif situé au centre du tableau).
 - Pour un tableau n*n, on doit couvrir $n^{2}-1$ cases,
il faut donc que $n^{2}-1$ soit divisible par 3.
```

Correction CC en TP

Durée de l'examen : 2h - le barème est donné à titre indicatif.
Aucun document n'est autorisé - Les exercices sont indépendants.

Exercice 4 Problème des 3 couleurs - 5 points
On considère des tableaux contenant uniquement les valeurs 1, 2 et 3. On veut regrouper les 1 au début et les 3 à la fin avec une complexité proportionnelle à la taille du tableau.

tab = | 2 | 3 | 1 | 2 | 3 | 1 | 2 | 1 |.

1. Cherchez une méthode ayant une complexité **proportionnelle à la taille du tableau**, expliquez là.
2. Ecrire une méthode `public static void probleme3Couleurs(int [] tab)` conforme aux spécifications.

```
On demande une complexité proportionnelle à la taille du tableau, soit O(n).
On peux donc parcourir le tableau k fois. La solution la plus simple revient
à compter le nombre d'occurrences de chaque élément puis à les écrires dans
l'ordre demandé. Cette méthode engendre 2 parcours du tableau.
Une solution plus efficace (mais non demandée) permet de ne parcourir
le tableau qu'une seule fois. Celle-ci fonctionne sur le même principe que
la résolution du problème des 2 couleurs vu en td.

//version optimale
public static void probleme3Couleurs(int [] t){
int i,j,k;

i=0;
j=0;
k=t.length-1;

while (j<=k){
  while (j<=k && t[j]==2) j++;
  while (j<=k && t[k]==3) k--;

  if (j<=k){
       if (t[j]==1){
      if (i<j) echanger(t,i,j);
          i++;
      j++;
    }
       else{
  echanger(t,j,k);
```

```
k--;
  }
}
  }
```

Exercice 5 Liste statique - 8 points
On souhaite créer une liste d'étudiants identifiés par leur *nom* (une chaîne de caractères) et leur numéro d'étudiant (un entier), *numEtu*, en utilisant pour cela un tableau. Une liste va correspondre à une classe (au sens des élèves d'une classe dans un lycée par exemple).

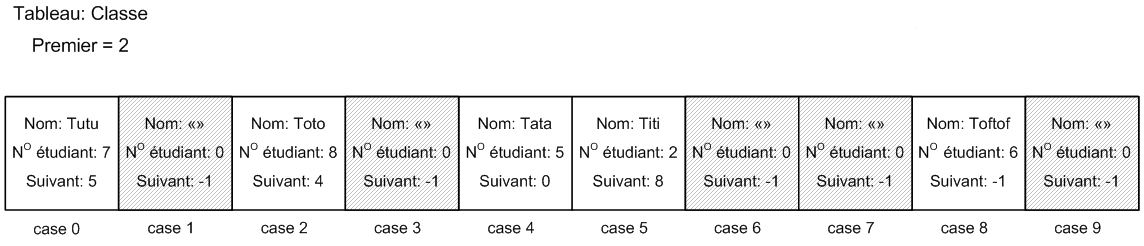


Fig. Exemple de la liste statique à l'aide d'un tableau qui contient des « trous »

1. La variable « premier » présente sur le schéma précédent permet d'indiquer la position du premier élement de la liste par sa position dans le tableau. Donnez (sous la forme d'un commentaire) le nom des différents étudiants présents dans cette classe en respectant l'ordre imposée par la liste.
2. Créez une classe « etudiant » représentant un élément de la liste dont les trois attributs : *nom* , *numEtu*, et *suivant* sont privés.
3. Ecrivez le constructeur par défaut qui initialisera la variable *nom* à une chaîne vide, *numEtu* à 0, et *suivant* à -1.
4. Ecrivez un constructeur qui prend en argument une chaîne de caractères destinée à l'initialisation du nom d'un étudiant et un entier destiné à l'initialisation de son numéro. La variable *suivant* est initialiée à -1.
5. Créez une classe « classe » possédant 3 attributs privés :
   - *listeEtudiants* : un tableau de type « etudiant »
   - *premier* : un entier qui contiendra l'indice dans le tableau du premier élément de la liste (c.à.d. la tête).
   - *nbEtudiants* : un entier représentant le nombre d'éléments non nuls dans *listeEtudiants*
6. Définissez également une constante TAILLE, de type entier, initialisée à 10 et représentant la taille du tableau.
7. Ecrivez le constructeur par défaut pour cette classe qui initialisera *premier* à -1,*nbEtudiants* à 0 et le tableau *listeEtudiants* en faisant appel au constructeur par défaut de la classe « etudiant ».
8. Ecrivez la méthode `insereEntete` qui insère un nouvel étudiant dans la liste à partir de son nom et de son numéro étudiant. On l'insère dans la première case disponible du tableau. On suppose qu'au départ le tableau ne contient que des éléments vides, c.à.d. avec un numéro d'étudiant égal à 0. Cette méthode retourne l'indice du tableau où le nouvel étudiant a été inséré, ou « -1 » s'il n'y a plus de place. Ce nouveau élément devient également la nouvelle tête de la liste.

9. Ecrivez une méthode `rechercheEtu` qui recherche un étudiant à partir de son numéro d'étudiant en parcourant la liste (et non le tableau). Cette méthode retourne l'indice où se trouve l'étudiant dans le tableau, ou « -1 » si on ne le trouve pas.
10. Ecrivez une méthode `suppression` qui supprime un étudiant de la liste à partir de son numéro d'étudiant. Cette méthode renvoie l'indice de l'élément supprimé dans le tableau, ou « -1 » s'il n'existe pas.
11. Ecrivez la méthode `public boolean insereApres(int no, String nomEtu, int noEtu)` qui insère un nouvel étudiant dans la liste à partir de son nom et de son numéro en l'ajoutant derrière l'étudiant de numéro *no*. Elle renvoie vrai si le nouvel étudiant a été inséré, faux sinon.
12. Ecrivez un programme principal qui :
    (a) Initialise la liste,
    (b) Demande les noms et les numéros de 4 nouveaux étudiants, et les ajoute à la liste s'ils n'y sont pas déjà (controle uniquement sur le numéro d'étudiant).Fait de multiples ajouts/suppressions successifs de façon à ce que le tableau comporte des trous.

`Cf fichiers Java joints`

Exercice 6 ***Algorithmes de Tri et recherche d'information*** - 6 points

60	1	20	43	21	9	17	30	52	81

.

1. Ecrire une méthode `static int [] triFusion(int[] tab)` qui effectue le tri fusion du tableau passé en argument.
2. Quel est le pire cas en terme de nombre d'echanges ? En déduire la complexité de l'algorithme dans le pire cas, dans le meilleur.

3. Rappelez brièvement le principe de la recherche par dichotomie d'une valeur dans un tableau.
4. Quelle(s) est(sont) la(les) précondition(s) de cet algorithme de recherche ?
5. Ecrire une méthode récursive `static int recherche(int [] t, int d, int f, int n)` qui recherche par dichotomie l'entier n dans le tableau t entre les indices d inclus et f inclus. La méthode retourne l'indice où se trouve la valeur n dans t ou -1 si elle en est absente.
6. Quelle est sa complexité dans le meilleur cas, dans le pire cas ?
7. Connaissez-vous une méthode de recherche plus efficace ?

```
Tout cet exercice, tri Fusion et recherche dichotomique,
repose sur la connaissance du cours. Cf celui-ci.
```