Algorithmique avancée
TP no 1 – les tableaux

DEUST 2ème année

9 octobre 2023

1 Afficher tous les éléments

Écrivez la méthode *afficher* qui, étant donnés un tableau d'entiers *tab* et un booléen *debut*, affiche tous les éléments du tableau du premier au dernier élément (si debut est *true*), ou du dernier au premier élément (si debut est *false*).

```
public static void afficher(int[] tab, boolean debut);
```

Exemple 1 :

```
int[] tab = {4, 38, 66, -21};
afficher(tab, true);
/* Resultat attendu :
tab[0] = 4
tab[1] = 38
tab[2] = 66
tab[3] = -21 */
```

Exemple 2 :

```
int[] tab = {39, 77, -33, 44};
afficher(tab, false);
/* Resultat attendu :
tab[3] = 44
tab[2] = -33
tab[1] = 77
tab[0] = 39 */
```

2 Ajouter un élément au début ou à la fin

Écrivez la fonction *ajouter* qui, étant donnés un tableau d'entiers *tab*, un entier naturel *v*, et un booléen *fin*, retourne un tableau de taille *tab.length+1* contenant les éléments de *tab* et tel que :

→ Si fin est vrai, alors le dernier élément du tableau est *v*,

→ Sinon, le premier élément du tableau est *v*.

```
public static int[] ajouter(int[] tab, int v, boolean fin);
```

Exemple 1 :

```
int[] t = {4, 38, 66, -21};
int[] tab = ajouter(t, 44, true);
afficher(tab, true);
/* Resultat attendu :
tab[0] = 4
tab[1] = 38
tab[2] = 66
tab[3] = -21
tab[4] = 44 */
```

Exemple 2 :

```
int[] t = {39, 77, -33, 44};
int[] tab = ajouter(t, -22, false);
afficher(tab, true);
/* Resultat attendu :
tab[0] = -22
tab[1] = 39
tab[2] = 77
tab[3] = -33
tab[4] = 44 */
```

3 Ajouter un élément à la position i

Écrivez la fonction *ajouter* qui, étant donnés un tableau d'entiers *tab*, un entier naturel *v*, et un entier naturel *pos*, retourne un tableau de taille *tab.length+1* contenant les éléments de *tab* et l'élément *v* à la position *pos*.

```
public static int[] ajouter(int[] tab, int v, int pos);
```

Exemple :

```
int[] t = {4, 38, 66, -21};
int[] tab = ajouter(t, 22, 2);
afficher(tab, true);
/* Resultat attendu :
tab[0] = 4
tab[1] = 38
tab[2] = 22
tab[3] = 66
tab[4] = -21 */
```

4 Supprimer un élément à la position i

Écrivez la fonction *supprimer* qui, étant donnés un tableau d'entiers *tab* et un entier naturel *pos*, retourne un tableau de taille *tab.length-1* contenant les éléments de *tab* sauf l'élément à la position *pos* du tableau *tab*.

```
public static int[] supprimer(int[] tab, int pos);
```

Exemple 1 :

```
int[] t = {4, 38, 66, -21};
int[] tab = supprimer(t, 1);
afficher(tab, true);
```

```
/* Resultat attendu :
tab[0] = 4
tab[1] = 66
tab[2] = -21 */
```

Exemple 2 :

```
int[] t = {4, 38, 66, -21};
int[] tab = supprimer(t, 4);
afficher(tab, true);
/* Resultat attendu :
tab[0] = 4
tab[1] = 38
tab[2] = 66
tab[3] = -21 */
```

5 Permuter deux éléments

Écrivez la méthode *permuter* qui, étant donnés un tableau d'entiers *tab*, un entier naturel i et un entier naturel j, retourne un tableau contenant les éléments de *tab* et où les éléments aux positions i et j ont été échangés.

```
public static int[] permuter(int[] tab, int i, int j);
```

Exemple :

```
int[] t = {4, 38, 66, -21};
int[] tab = permuter(t, 0, 3);
afficher(tab, true);
/* Resultat attendu :
tab[0] = -21
tab[1] = 38
tab[1] = 66
tab[2] = 4 */
```

6 Moyenne

Écrivez la fonction *moyenne* qui, étant donné un tableau d'entiers *tab*, retourne la moyenne des éléments du tableau.

```
public static double moyenne(int[] tab);
```

Exemple :

```
int[] t = {10, 25, 33, 44, 16, 99};
System.out.print("Moyenne = "+moyenne(t)+"\n");
/* Resultat attendu :
Moyenne = 37,833333 */
```

7 Maximum

Écrivez la fonction *maximum* qui, étant donné un tableau d'entiers *tab*, retourne un tableau d'entiers de taille 2 contenant l'entier le plus grand de *tab* ainsi que la position de sa permière occurrence dans *tab*.

```
public static int[] maximum(int[] tab);
```

Exemple :

```
int[] t = {1002, 33, 348, 92, 77, 5775, 20, 15};
int[] tmax = maximum(t);
System.out.println("Maximum : t["+tmax[1]+"] = "+tmax[0]);
/* Resultat attendu :
Maximum : t[5] = 5775 */
```

8 Minimum

Écrivez la fonction *minimum* qui, étant donné un tableau d'entiers *tab*, retourne un tableau d'entiers de taille 2 contenant l'entier le plus petit de *tab* ainsi que la position de sa permière occurrence dans *tab*.

```
public static int[] minimum(int[] tab);
```

Exemple :

```
int[] t = {1002, 33, 348, 92, 77, 5775, 20, 15};
int[] tmin = minimum(t);
System.out.print("Minimum : t["+tmin[1]+"] = "+tmin[0]+"\n");
/* Resultat attendu :
Minimum : t[7] = 15 */
```

9 Compter

Écrivez la fonction *compteSup* qui, étant donnés un tableau d'entiers *tab* et un entier naturel v, retourne le nombre d'entiers supérieurs à v dans *tab*.

```
public static int compteSup(int[] tab, int v);
```

Exemple :

```
int[] t = {1002, 33, 348, 92, 77, 5775, 20, 15};
int v = 77;
System.out.println("Nombre d'entiers >= a "+v+" : "+compteSup(t, v));
/* Resultat attendu :
Nombre d'entiers >= a 77 : 5 */
```

10 Rechercher

Écrivez la fonction *rechercher* qui, étant donnés un tableau d'entiers *tab* et un entier naturel v, retourne la position (l'indice) de la première occurrence de v dans *tab*. Si *tab* ne contient pas v, alors la fonction retournera -1.

```
public static int rechercher(int[] tab, int v);
```

Exemple :

```
int[] t = {1002, 33, 348, 92, 77, 5775, 20, 15};
int v = 20;
int pos = rechercher(t, v);
if (pos >= 0)
   System.out.print("La valeur "+v+" a ete trouvee en "+pos+"eme
      position du tableau\n");
else
   System.out.print("La valeur "+v+" n'est pas dans le tableau\n");
/* Resultat attendu :
La valeur 20 a ete trouvee en 6eme position du tableau */
```

11 Rechercher dans un tableau trié

Nous reprenons les contraintes de l'exercice précédent sauf que, cette fois-ci, le tableau *tab* donné en entrée est supposé trié par ordre croissant des valeurs. Améliorez le code de la fonction précédente pour prendre en compte ce cas particulier.

```
public static int rechercherTrie(int[] tab, int v);
```

Exemple :

```
int[] t = {-55, -22, -9, 0, 15, 20, 66, 104};
int v = 20;
int pos = rechercherTrie(t, v);
if (pos >= 0) {
   System.out.println("La valeur "+v+" a ete trouvee en "+pos+"eme
      position du tableau");
} else {
   System.out.println("La valeur "+v+" n'est pas dans le tableau");
}
/* Resultat attendu :
La valeur 20 a ete trouvee en 5eme position du tableau */
```

12 Rechercher les tous

Écrivez la fonction *rechercherTous* qui, étant donnés un tableau d'entiers *tab* et un entier naturel v, retourne un tableau d'entiers contenant la position (l'indice) de chacune des occurences de la valeur v dans *tab*. Si *tab* ne contient pas la valeur v, alors la fonction retournera un tableau de taille 1 contenant -1 en première position.

```
public static int[] rechercherTous(int[] tab, int v);
```

Exemple :

```
int[] t = {10, 33, 10, 10, 55, 7777, 11, 9, 10, 5, 10};
int v = 10;
int[] pos = rechercherTous(t, v);
afficher(pos, true);
/* Resultat attendu :
tab[0] = 0
tab[1] = 2
tab[2] = 3
tab[3] = 8
tab[4] = 10 */
```

13 Tableau trié ?

Écrivez la fonction *estTrie* qui, étant donné un tableau d'entiers *tab*, retourne *true* si les valeurs contenues dans *tab* sont triées par ordre croissant, et *false* sinon.

```
public static boolean estTrie(int[] tab);
```

Exemple :

```
int[] t = {1, 2, 3, 4, 5, 6, 7, 5};
if (estTrie(t))
   System.out.print("Le tableau est trie :-) !\n");
else
   System.out.print("Le tableau n'est pas trie :-( !\n");
/* Resultat attendu :
Le tableau n'est pas trie :-( ! */
```

14 Concaténer deux tableaux d'entiers

Écrivez la fonction *fusion* qui, étant donnés deux tableaux d'entiers *tab*1 et *tab*2, retourne un unique tableau contenant les éléments de *tab*1 suivis de ceux de *tab*2.

```
public static int[] fusion(int[] tab1, int[] tab2);
```

Exemple :

```
int[] t1 = {44, 38, 29, 16, 39};
int[] t2 = {22, 17, 18, -5, 16};
int[] t = fusion(t1, t2);
afficher(t, true);
/* Resultat attendu :
tab[0] = 44
tab[1] = 38
tab[2] = 29
tab[3] = 16
tab[4] = 39
tab[5] = 22
tab[6] = 17
tab[7] = 18
tab[8] = -5
tab[9] = 16 */
```