Module 1 –Informatique – Algorithmique et Programmation Objet

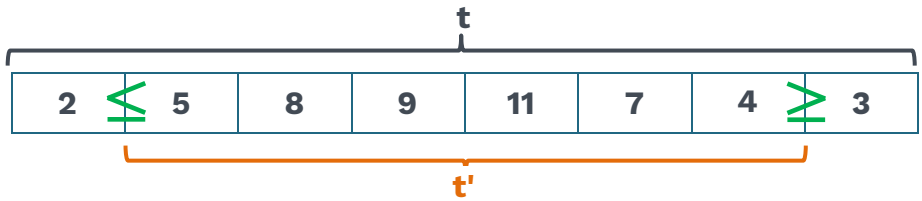
Travaux Pratiques (12), Licence 1ère Année

La récursivité (suite)

Exercice 1 Tableau en chapeau

Un tableau “en chapeau” est un tableau dont la valeur des éléments est croissante jusqu’à la moitié du tableau puis décroissante jusqu’à la fin du tableau.

Principe : (voir schéma ci-dessous) un tableau **t** est "en chapeau" si son premier élément est inférieur au suivant **ET** son dernier élément est inférieur au précédent **ET** le sous-tableau **t’** (égal à **t** sans le premier et dernier éléments) est "en chapeau". On réitère le principe jusqu’à un sous-tableau de 1 ou 2 éléments.



Écrire une méthode récursive **boolean enChapeau(int[] t, int deb, int fin)** qui indique si un tableau passé en paramètre est “en chapeau” ou pas.

Note : au lieu de construire à chaque fois un nouveau tableau, il est préférable de passer à la méthode les indices de début (**deb**) et de fin (**fin**) du sous-tableau considéré.
La méthode sera appelée initialement avec **deb = 0** et **fin = t.length()-1**, ce qui correspond à la totalité du tableau **t**.

Exemple

```
5    public static void main(String[] args) {
6        int[] t1 = {1, 2, 3, 2, 1};
7        int [] t2 = {1, 2, 3, 4, 2, 1};
8        int [] t3 = {1, 2, 4, 3, 2, 1};
9        int [] t4 = {1, 4, 2, 3, 2, 1};
10       System.out.println("t1 est en chapeau ? "+enChapeau(t1, 0, t1.length-1));
11       System.out.println("t2 est en chapeau ? "+enChapeau(t2, 0, t2.length-1));
12       System.out.println("t3 est en chapeau ? "+enChapeau(t3, 0, t3.length-1));
13       System.out.println("t4 est en chapeau ? "+enChapeau(t4, 0, t4.length-1));
```

Problems @ Javadoc Declaration Console

```
<terminated> TP_11_Tableaux_2 [Java Application] C:\Program Files\Java\jdk1.7.0_60\bin\javaw.exe (19 nov. 2018 00:05:09)
t1 est en chapeau ? true
t2 est en chapeau ? true
t3 est en chapeau ? true
t4 est en chapeau ? false
```

Exercice 2 Recherche séquentielle

Écrire une méthode récursive **boolean rechercheRec(int [] t, int val, int i)**, acceptant en paramètres un tableau d’entiers **t** et un entier **val**, qui indique si la valeur **val** est présente ou pas dans le tableau **t**.

Note : le paramètre **i** correspond à l’indice courant du tableau **t** reçu. Ceci évite de construire un nouveau tableau avant chaque appel.

Exemple

```
11          System.out.println("\t" + 6 + " est présent dans t[] ? " + rechercheRec(t, 6, 0));
12          System.out.println("\t" + 5 + " est présent dans t[] ? " + rechercheRec(t, 5, 0));
```

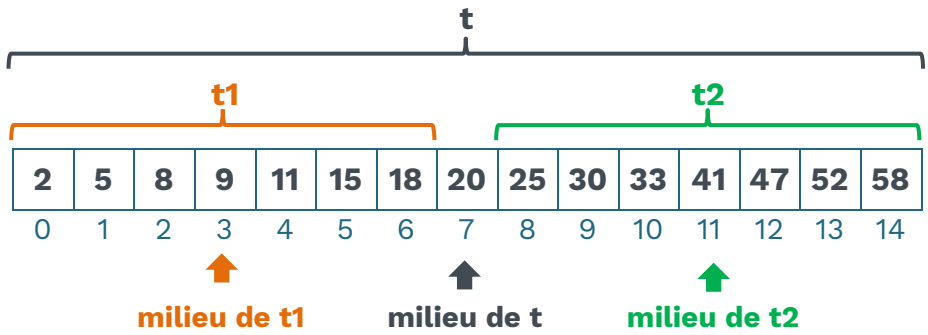
Problems | Javadoc | Declaration | Console

```
<terminated> TP_11_Tableaux_2 [Java Application] C:\Program Files\Java\jdk1.7.0_60\bin\javaw.exe (14 nov. 2019 14:52:40)
Contenu du tableau :    { 2 , 4 , 6 , 8 , 10 , 12 , 14 , 16 , 18 , 20 }
        6 est présent dans t[] ? true
        5 est présent dans t[] ? false
```

Exercice 3 Recherche dichotomique

Lorsqu'un tableau **t** contient des éléments ordonnés, il est plus rapide de retrouver un élément donné par la méthode dite de dichotomie que par une recherche séquentielle.

Principe : (voir schéma ci-dessous) on compare l'élément recherché avec l'élément se trouvant au milieu du tableau **t**. S'il ne correspond pas, et s'il est inférieur (respectivement supérieur), on recommence avec l'intervalle constitué de la moitié inférieure (tableau **t1**) (respectivement supérieure (tableau **t2**)) du tableau **t**. On réitère le processus jusqu'à correspondance ou bien jusqu'à ce que l'intervalle considéré soit constitué d'un seul élément.



Écrire une méthode récursive **boolean rechercheDicho(int [] t, int deb, int fin, int val)**, de façon que la recherche d'une valeur donnée se fasse selon le principe de dichotomie.

Note : au lieu de construire à chaque fois un nouveau tableau, il est préférable de passer à la méthode les indices de début (**deb**) et de fin (**fin**) du sous-tableau considéré.
La méthode sera appelée initialement avec **deb = 0** et **fin = t.length()-1**, ce qui correspond à la totalité du tableau **t**.

Exemple

```
6   int[] t = { 2, 4, 6, 8, 10, 12, 14, 16, 18, 20 };
7   int n1 = 6;
8   int n2 = 5;
9   afficheTab(t);
10  System.out.println("\t" + n1 + " est présent dans t[] ? " + rechDicho(t, 0, t.length - 1, n1));
11  System.out.println("\t" + n2 + " est présent dans t[] ? " + rechDicho(t, 0, t.length - 1, n2));
```

Problems | Javadoc | Declaration | Console

```
<terminated> TP_11_Tableaux_2 [Java Application] C:\Program Files\Java\jdk1.7.0_60\bin\javaw.exe (19 nov. 2018 00:22:31)
Contenu du tableau :    { 2 , 4 , 6 , 8 , 10 , 12 , 14 , 16 , 18 , 20 }
        6 est présent dans t[] ? true
        5 est présent dans t[] ? false
```