


# Outillage fonction récursive

## RAPPELS :

- une fonction renvoie un résultat et un seul
- l'outillage d'une fonction suppose que l'on renvoie :
  - ✓ le résultat de la fonction
  - ✓ le nombre d'actions "intéressantes" (ex : nombre de comparaisons) effectuées par la fonction **2 résultats...**
- **Solution proposée dans ce cours :**
  - ✓ modifier le type du résultat de la fonction : nouveau type = objet de 2 attributs (un attribut pour le **résultat initial de la fonction**, un attribut pour le **compteur**)
  - ✓ compléter le code de la fonction avec les instructions de comptage des actions "intéressantes"

# Recherche dichotomique récursive

## le Modèle

```
public static int rechIndDichoRec(ArrayList<Integer> vInt, int unInt) {  
    // { vInt non vide, trié }  
    // => { résultat = indice de la 1ère occurrence de unInt dans vInt }  
    if (vInt.get(vInt.size() - 1) < unInt) {  
        return -1;  
    } else {  
        return rechIndDichoRecWorker(vInt, unInt, 0, vInt.size() - 1);  
    }  
}
```

## le Worker

```
public static int rechIndDichoRecWorker(ArrayList<Integer> vInt,  
                                         int unInt, int inf, int sup) {  
    // { vInt[inf..sup] non vide, trié }  
    // => { résultat = indice de la 1ère occurrence de unInt dans vInt[inf..sup] }  
    //      RECHERCHE DICHOTOMIQUE RÉCURSIVE }  
    if (inf == sup) {  
        if (vInt.get(sup) == unInt) {  
            return sup;  
        } else {  
            return -1;  
        }  
    } else {  
        int m = (inf + sup) / 2;  
        if (vInt.get(m) < unInt) {  
            return rechIndDichoRecWorker(vInt, unInt, m + 1, sup);  
        } else {  
            return rechIndDichoRecWorker(vInt, unInt, inf, m);  
        }  
    }  
}
```

# Pour outiller une fonction (RAPPEL)

*la classe PaireResultatCompteur*

```
/** Classe générique pour outiller une fonction
 * Elle propose uniquement un constructeur et 2 getters
 * @param <R> : R est le type du résultat de la fonction outillée
 */
public class PaireResultatCompteur <R> {
    private R res; // résultat de la fonction outillée
    private int compteur; // compteur du code observé

    public PaireResultatCompteur(R res, int compteur) {
        this.res = res;
        this.compteur = compteur;
    }
    public R getRes() {
        return res;
    }
    public int getCompteur() {
        return compteur;
    }
}
```

# Recherche dichotomique récursive outillée (1)

## le Modèle

```
public static PaireResultatCompteur<Integer> indRDichoRec0(ArrayList<Integer> vInt,
                                                         int unInt) {
    // {vInt non vide, trié}
    // => { à la fin du traitement, ind est l'indice de la 1ère occurrence
    //       de unInt dans vInt, ou est égal à -1 si non trouvé
    //
    //       résultat = variable de type PaireResCompteur avec :
    //               - res égal à ind
    //               - compteur égal au nombre de comparaisons effectuées
    //               entre inInt et un élément de vInt[inf..sup]
    if (vInt.get(vInt.size() - 1) < unInt) {
        return new PaireResultatCompteur<>(-1, 1);
    } else {
        return indRDichoRecWorker0(vInt, unInt, 0, vInt.size() - 1);
    }
}
```

# Recherche dichotomique récursive outillée (2)

## le Worker

```
public static PaireResultatCompteur<Integer> indRDichoRecWorker0(ArrayList<Integer> vInt,
                                                                    int unInt, int inf,
                                                                    int sup) {

    // { vInt[inf..sup] non vide, trié }
    // => { à la fin du traitement, ind est l'indice de la 1ère occurrence
    //       de unInt dans vInt[inf..sup], ou est égal à -1 si non trouvé
    //       résultat = variable de type PaireResCompteur avec :
    //           - res égal à ind
    //           - compteur égal au nombre de comparaisons effectuées
    //           entre inInt et un élément de vInt[inf..sup]
    //
    // RECHERCHE DICHOTOMIQUE RÉCURSIVE }

    if (inf == sup) {
        if (vInt.get(sup) == unInt) {
            return new PaireResultatCompteur<>(sup,1);
        } else {
            return new PaireResultatCompteur<>(-1,1);
        }
    } else {
        int m = (inf + sup) / 2;
        PaireResultatCompteur<Integer> paireRC;
        if (vInt.get(m) < unInt) {
            paireRC = indRDichoRecWorker0(vInt, unInt, m+1, sup);
        } else {
            paireRC = indRDichoRecWorker0(vInt, unInt, inf, m);
        }
        return new PaireResultatCompteur<>(paireRC.getRes(),paireRC.getCompteur() + 1);
    }
}
```

# Recherche dichotomique récursive outillée (3)

**$v = [5, 6, 7, 8, 9, 12, 12, 14, 17]$  et on cherche l'indice de l'entier 6 dans  $v$**

Appel de `rechIndDichoRecWorkerComp` sur  $V[0,8]$   
 $m = (0+8)/2 = 4$   
COMPARAISON DE  $vInt[4]$  (9) à `unInt` (6)  
 $vInt[4] \geq \text{à } unInt$   
 $\Rightarrow$  Nouvelle valeur de sup : 4 / inf = 0

Appel de `rechIndDichoRecWorkerComp` sur  $V[0,4]$   
 $m = (0+4)/2 = 2$   
COMPARAISON DE  $vInt[2]$  (7) à `unInt` (6)  
 $vInt[2] \geq \text{à } unInt$   
 $\Rightarrow$  Nouvelle valeur de sup : 2 / inf = 0

Appel de `rechIndDichoRecWorkerComp` sur  $V[0,2]$   
 $m = (0+2)/2 = 1$   
COMPARAISON DE  $vInt[1]$  (6) à `unInt` (6)  
 $vInt[1] \geq \text{à } unInt$   
 $\Rightarrow$  Nouvelle valeur de sup : 1 / inf = 0

Appel de `rechIndDichoRecWorkerComp` sur  $V[0,1]$   
 $m = (0+1)/2 = 0$   
COMPARAISON DE  $vInt[0]$  (5) à `unInt` (6)  
 $vInt[0] < \text{à } unInt$   
 $\Rightarrow$  Nouvelle valeur de inf : 1 / sup = 1

Appel de `rechIndDichoRecWorkerComp` sur  $V[1,1]$   
inf (1) est égal à sup (1)  
COMPARAISON DE  $vInt[1] = 6$  à `unInt` (6)  
Nombre de comparaisons effectuées : 1

$vInt[1] = 6 \Rightarrow$  1ère occ. de 6 trouvée à l'indice 1  
pRC(res : 1, compteur : 1)

## RÉSULTAT DE `rechIndDichoRecComp`

pRC = (res : 1, compteur : 5)

qui s'interprète ainsi :

- l'entier 6 a été trouvé à l'indice 1
- 5 comparaisons ont été faites pour le trouver

*Retour Appel de `rechIndDichoRecWorkerComp` sur  $V[0,4]$*

Nombre de comparaisons effectuées : 5

pRC = (res : 1, compteur : 5)

*Retour Appel de `rechIndDichoRecWorkerComp` sur  $V[0,2]$*

Nombre de comparaisons effectuées : 4

pRC = (res : 1, compteur : 4)

*Retour Appel de `rechIndDichoRecWorkerComp` sur  $V[0,1]$*

Nombre de comparaisons effectuées : 3

pRC = (res : 1, compteur : 3)

*Retour Appel de `rechIndDichoRecWorkerComp` sur  $V[1,1]$*

Nombre de comparaisons effectuées : 2

pRC = (res : 1, compteur : 2)