

Examen de Algorithmique et Struct. de Données

Durée : 0h30

Aucun document autorisé

Mobiles interdits

Nom :

Prénom :

Une petite amélioration de la recherche linéaire consiste à faire avancer vers le début de la liste un élément que l'on recherche et que l'on a trouvé. Ainsi, si on recherche plusieurs fois un même élément, il sera alors amené progressivement en début de liste et la recherche sera plus rapide. On parle de recherche *auto-adaptative*.

- 1. Dans classe `ListeTri` donnée en annexe, écrivez en Java la méthode `rechercheLinéaireAuto` qui recherche dans une liste un élément de clé `c`. Si la clé est trouvée, l'élément est avancé de la moitié de la distance entre le début de la liste et sa position courante. La méthode renvoie l'élément s'il est trouvé ou l'exception `CléNonTrouvée` si la recherche échoue.

```
public V rechercheLinéaireAuto(C c)
throws CléNonTrouvéeException
{
    for (int i=0; i<l.size(); i++) {
        Élément<C,V> e = l.get(i);
        if (c.compareTo(e.clé())==0) {
            //trouvé ! => l'échanger avec l'élément du milieu si i!=0
            if (i!=0) {
                int m = i/2;
                Élément<C,V> em = l.get(m);
                l.set(m, e);
                l.set(i,em);
            }
            return e.valeur();
        }
    }
    //non trouvé
    throw new CléNonTrouvéeException();
}
```

- 2. Soit la liste de valeurs entières 11, 1, 14, 4, 2, 0, 0, 7, 10, 5. Écrivez l'évolution de cette liste après chaque étape du tri par insertion linéaire.

```
11 1 14 4 2 0 0 7 10 5
1 11 14 4 2 0 0 7 10 5
1 11 14 4 2 0 0 7 10 5
```

```
1 4 11 14 2 0 0 7 10 5
1 2 4 11 14 0 0 7 10 5
0 1 2 4 11 14 0 7 10 5
0 0 1 2 4 11 14 7 10 5
0 0 1 2 4 7 11 14 10 5
0 0 1 2 4 7 10 11 14 5
0 0 1 2 4 5 7 10 11 14
```

-
- 3. Pour trier n valeurs, quel est le nombre de comparaisons effectuées par ce tri par insertion linéaire dans la pire des cas, dans le cas moyen, dans le meilleur des cas ?

.....

$n(n-1)/2$, $n(n-1)/4$, $n-1$.

-
- 4. Écrivez en Java la méthode `boolean estTriée()` qui teste si la liste courante est triée ou pas.

```
public boolean estTriée() {
    C cp = l.get(0).clé();
    for (int i=1; i<l.size(); i++) {
        C cc = l.get(i).clé();
        if (cc.compareTo(cp)<0) return false;
        cp = cc;
    }
    return true;
}
```