

# Algorithmes de tri (1)

## Plan

1. Tri par insertion
2. Tri par sélection
3. Une borne inférieure

Objectif: Familiarisation avec l'écriture de programmes.

## 1. Tri par insertion

Soit à trier une suite  $a_0, \dots, a_{n-1}$  en ordre croissant.

*Principe* : Insérer le  $i$ -ème élément dans la suite des  $i - 1$  éléments déjà triés.

*Algorithme* :

pour  $i$  de 1 à  $n - 1$  faire  
    insérer  $(a, i)$ ;

L'insertion proprement dite se fait par :

insérer  $(a, i)$  :  
     $j := i; v := a_i$ ;  
    tantque  $j > 0$  et  $a_{j-1} > v$  faire  
        {  $a_j := a_{j-1}; j := j - 1;$  }  
     $a_j := v$ ;

*Exemple* : Un paquet de cartes

```

static void insere(int[] a, int i) {
    int j = i, v = a[i];
    while (j > 0 && a[j-1] > v) {
        a[j] = a[j-1];
        --j;
    }
    a[j] = v;
}

```

```

static void triInsertion(int[] a) {
    for (int i = 1; i < a.length; ++i)
        insere( a, i);
}

```

---

```

class TriParInsertion {

    final static int N = 10;

    static void initialisation(int[] a) {
        for (int i = 0; i < a.length; ++i)
            a[i] = (int) (Math.random() * 128);
    }

    static void impression(int[] a) {
        for (int i = 0; i < a.length; ++i)
            System.out.print (" " + a[i] + " ");
        System.out.println ("");
    }

    public static void main(String args[]) {
        int[] a = new int[N];
        initialisation(a);
        impression(a);
        triInsertion(a);
        impression(a);
    }
}

```

---

Coût de l'insertion de  $a_i$  : le nombre de  $j < i$  tels que  $a_j > a_i$ .

## 2. Tri par sélection

Soit à trier une suite  $a_0, \dots, a_{n-1}$  en ordre croissant.

On cherche l'indice  $m$  du plus petit élément  $a_m$  de  $a_0, \dots, a_{n-1}$ , on échange  $a_0$  et  $a_m$  et on recommence sur  $a_1, \dots, a_{n-1}$  :

*Algorithme :*

```
pour  $i$  de 0 à  $n - 2$  faire
  calculer
     $m = \text{argMin}(a, i)$ ;
  échanger( $a_i, a_m$ )
```

Par définition

$$\text{argMin}(a, i) = \min\{j \in \{i, \dots, n - 1\} \mid a_j \leq a_i, \dots, a_{n-1}\}$$

Le calcul de  $m$  se fait par :

```
argMin( $a, i$ ) :
   $m := i$ ;
  pour  $j$  de  $i + 1$  à  $n - 1$  faire
    si  $a_j < a_m$  alors  $m := j$ ;
  retourner  $m$ ;
```

*Analyse du tri par sélection :*

Le nombre de comparaisons est  $n(n - 1)/2$ .

```

static int argMin(int[] a, int i) {
    int m = i;
    for (int j = i + 1; j < a.length; j++)
        if ( a[j] < a[m] ) m = j;
    return m;
}

static void echange(int [] a, int i, int j) {
    int t = a[i]; a[i] = a[j]; a[j] = t;
}

static void triSelection(int[] a) {
    for (int i = 0; i < a.length - 2; ++i) {
        int m = argMin(a, i);
        echange(a,i,m);
    }
}

```

---

### 3. Une borne inférieure

**Théorème** *Tout algorithme de tri, fondé uniquement sur des comparaisons d'éléments deux-à-deux, fait au moins  $O(n \log n)$  comparaisons.*

Voir polycopié, pages 90–91.