# TP1 : Comparaison d’algorithmes de tri

Soit un tableau T de taille N où chaque cellule ci avec 1 ≤ i ≤ N contient un nombre. Trier T par ordre croissant consiste à réordonner les valeurs des cellules de manière à ce que :

√ 1≤i ≤ N et 1 ≤j≤ N : ci ≤ cj

Pour passer d’un tableau non trié à un tableau trié par ordre croissant, il existe plusieurs méthodes de tri. L’objectif de ce TP est de programmer deux de ces méthodes pour pouvoir ensuite comparer leur efficacité.

Exemple

Pour illustrer les deux méthodes de tri décrites ci-dessous, nous prendrons le tableau T comprenant 5 éléments et dont les éléments sont :

**8**

**1**

**5**

**2**

**7**

**3**

**1**

**4**

**2**

**5**

Exercice 1 : Classe Tableau

Créer une classe Tableau qui contiendra un tableau de N entiers.

Faire trois constructeurs :

* Un constructeur sans paramètre dans lequel on demandera à l’utilisateur de saisir l’ensemble des valeurs du tableau sous la forme : N V1 V2 … VN où N est le nombre de valeurs souhaité et V1 à VN les N valeurs.
* Un constructeur avec un entier N en paramètre, où N correspond au nombre de valeurs souhaité. Les N valeurs seront données aléatoirement par la machine et seront comprise entre 1 et 100.
* Un constructeur par recopie qui prend un objet de type Tableau en paramètre.

Ecrire une méthode qui permet d’afficher le tableau.

Exercice 2 : Insertion d’un élément

Pour réaliser les deux algorithmes de tri décrits ci-dessous, vous aurez besoin d’une fonction d’insertion d’un élément. Considérons que cet élément se trouve à la position i et doit être insérer à la position j (avec j < i). Pour insérer T[i] à la position j, il faut :

1. Sauvegarder la valeur de T[i] dans une variable x ;
2. Décaler d’une position tous les éléments compris entre la position j et la position i-1 ;
3. Mettre la valeur de x à la position j.

Ecrire une méthode dans la classe Tableau qui permette de faire l’insertion d’un élément

Exercice 3 : Tri par sélection

Le tri par sélection consiste à chercher le plus petit élément du tableau et à le placer en première position. Une fois le plus petit élément positionné en première position, on recommence la même opération en commençant à l’indice 2 : on cherche le plus petit élément compris dans le tableau entre la position 2 et la fin du tableau, puis on place cet élément à la position 2. Puis on recommence ainsi en partant de l’élément 3, 4 et ainsi de suite jusqu’au bout du tableau.

|  |  |
| --- | --- |
| Soit le tableau de départ suivant | **8**  **1**  **5**  **2**  **7**  **3**  **1**  **4**  **2**  **5** |
| Le plus petit élément du tableau est 1. On l’insère à la première position du tableau | **1**  **1**  **8**  **2**  **5**  **3**  **7**  **4**  **2**  **5** |
| On recommence en partant de l’indice 2. Entre l’élément 2 et la fin du tableau, le plus petit élément est 2. On l’insère donc en seconde position. | **1**  **1**  **2**  **2**  **8**  **3**  **5**  **4**  **7**  **5** |
| On recommence en partant de l’indice 3. Entre l’élément 3 et la fin du tableau, le plus petit élément est 5. On l’insère donc en troisième position. | **1**  **1**  **2**  **2**  **5**  **3**  **8**  **4**  **7**  **5** |
| On recommence en partant de l’indice 4. Entre l’élément 4 et la fin du tableau, le plus petit élément est 7. On l’insère donc en quatrième position. | **1**  **1**  **2**  **2**  **5**  **3**  **7**  **4**  **8**  **5** |

Ecrire une méthode dans la classe Tableau qui permette de réaliser le tri par sélection.

Exercice 4 : Tri par insertion

Le tri par insertion consiste à classer les deux premiers éléments du tableau. Une fois que les deux premiers sont ordonnés, on prend l’élément qui suit et on le classe à son tour dans ce qui a déjà été classé. Pour chaque élément i du tableau, on sait que les éléments de 1 à i-1 sont déjà classés. On va chercher la position j parmi les i-1 premiers éléments de manière à ce que T[i] < T[j]. On insère alors T[i] à la position j.

|  |  |
| --- | --- |
| Soit le tableau de départ suivant | **8**  **1**  **5**  **2**  **7**  **3**  **1**  **4**  **2**  **5** |
| On commence par classer les 2 premiers éléments | **5**  **1**  **8**  **2**  **7**  **3**  **1**  **4**  **2**  **5** |
| Puis on prend l’élément à la position i=3 (T[3] = 7), on cherche la position j dans le sous tableau déjà trié ([5 ; 8]) : j=2, x=T[3]=7, on décale tous les éléments d’une case entre j et i-1 soit ici entre 2 et 2. Donc seul 8 est décalé d’une case. Enfin on insère x à la position j. | **5**  **1**  **7**  **2**  **8**  **3**  **1**  **4**  **2**  **5** |
| On prend ensuite l’élément à la position i=4 (T[4] = 1), on cherche la position j dans le sous tableau déjà trié ([5 ; 7 ; 8]) : j=1, x=T[4]=1, on décale tous les éléments d’une case entre j et i-1 soit ici entre 1 et 3. Donc 5, 7 et 8 sont décalés d’une case. Enfin on insère x à la position j. | **1**  **1**  **5**  **2**  **7**  **3**  **8**  **4**  **2**  **5** |
| Enfin, on prend l’élément à la position i=5 (T[5] = 2), on cherche la position j dans le sous tableau déjà trié ([1 ; 5 ; 7 ; 8]) : j=2, x=T[5] = 2, on décale tous les éléments d’une case entre j et i-1 soit ici entre 2 et 4. Donc 5, 7 et 8 sont décalés d’une case. Enfin on insère x à la position j. | **1**  **1**  **2**  **2**  **5**  **3**  **7**  **4**  **8**  **5** |

Ecrire une méthode dans la classe Tableau qui permette de réaliser le tri par insertion.

Exercice 5 : Comparaison des différents tris

Les méthodes de tris seront évaluées au moyen de deux variables : le nombre d’opération et le temps nécessaire pour réorganiser le tableau. Le nombre d’opération est le nombre de tests effectués couplé au nombre d’affectations réalisées.

Il vous est demandé d’évaluer ces deux méthodes de tri pour des tableaux de taille de 50 à 1000 éléments en augmentant à chaque fois la taille du tableau de 50 éléments. Pour chaque taille, les tris seront évalués 20 fois chacun afin de voir si le temps d’exécution et le nombre d’opération sont variable ou non. A chaque fois, vous devrez créer un tableau de la taille demandée et remplir chaque cellule avec une valeur générée aléatoirement. Ce tableau devra alors être trié successivement avec les deux méthodes de tri.

Pour réaliser ces comparaisons, créez une classe Mesure qui contient deux attributs :

* un de type entier qui compte le nombre d’opération ;
* un de type long pour stocker le temps d’exécution.