Les algorithmes de tri

BUT du TP : concevoir des algorithmes de tri pour trier l'information

Votre mission : Concevoir plusieurs algorithmes de tri parmi :

- Le tri à bulle
- Le tri par sélection
- et éventuellement d'autres méthodes….

Elaborer un diaporama pour venir présenter le fonctionnement de votre algorithme.

I. Le tri à bulle :

Le principe de l'algorithme du tri à bulles est très simple à assimiler. Il est, et de loin, l'un des algorithmes de tri les plus simples qui soient.

Pour vous expliquer le principe, je vais d'abord vous donner une courte explication écrite puis nous allons concrètement trier une liste de nombres.

Le principe du tri à bulles est de comparer deux valeurs adjacentes et d'inverser leur position si elles sont mal placées. Alors, qu'entend-t-on par "mal placé" ? C'est très simple et surtout, c'est logique : si un premier nombre x est plus grand qu'un deuxième nombre y et que l'on souhaite trier l'ensemble par ordre croissant, alors x et y sont mal placés et il faut les inverser. Si, au contraire, x est plus petit que y, alors on ne fait rien et l'on compare y à z, l'élément suivant. C'est donc itératif. Et on parcourt ainsi la liste jusqu'à ce qu'on ait réalisé n-1 passages (n représentant le nombre de valeurs à trier) ou jusqu'à ce qu'il n'y ait plus rien à inverser lors du dernier passage.

Avec de la logique, on s'aperçoit qu'au premier passage, on place le plus grand élément de la liste au bout du tableau, au bon emplacement. Pour le passage suivant, nous ne sommes donc plus obligés de faire une comparaison avec le dernière élément ; et c'est bien plus avantageux ainsi. Donc à chaque passage, le nombre de valeurs à comparer diminue de 1.

Pour illustrer ce principe, prenons la suite de nombres suivante :

```
6 0 3 5 1 4 2
```

Nous voulons trier ces valeurs par ordre croissant. Commençons par le commencement. Nous allons faire un premier passage.

```
6 0 3 5 1 4 2   // On compare 6 et 0 : on inverse
0 6 3 5 1 4 2   // On compare 6 et 3 : on inverse
0 3 6 5 1 4 2   // On compare 6 et 5 : on inverse
0 3 5 6 1 4 2   // On compare 6 et 1 : on inverse
0 3 5 1 6 4 2   // On compare 6 et 4 : on inverse
0 3 5 1 4 6 2   // On compare 6 et 2 : on inverse
```

0 3 5 1 4 2 6 // Nous avons terminé notre premier passage

La question à se poser est la suivante : Devons-nous continuer ? Oui car lors du dernier passage, au moins un échange a été effectué.
Nous allons donc refaire un passage mais en omettant la dernière case.

0 3 5 1 4 2 6 // On compare 0 et 3 : on laisse
0 3 5 1 4 2 6 // On compare 3 et 5 : on laisse
0 3 5 1 4 2 6 // On compare 5 et 1 : on inverse
0 3 1 5 4 2 6 // On compare 5 et 4 : on inverse
0 3 1 4 5 2 6 // On compare 5 et 2 : on inverse
0 3 1 4 2 5 6 // Nous avons terminé notre passage

Comme promis, on ne compare pas 5 et 6 car c'est **inutile** et ce qui est inutile est à éviter, d'autant plus que cela ralenti l'algorithme.
Je suppose que vous avez bien compris le principe alors ce que je vous propose maintenant, c'est juste de vous montrer les dernières étapes avant d'aboutir à la liste triée.

0 3 1 4 2 5 6 // On compare 0 et 3 : On laisse
0 3 1 4 2 5 6 // On compare 3 et 1 : On inverse
0 1 3 4 2 5 6 // On compare 3 et 4 : On laisse
0 1 3 4 2 5 6 // On compare 4 et 2 : On inverse
0 1 3 2 4 5 6 // Nous avons terminé notre passage

0 1 3 2 4 5 6 // On compare 0 et 1 : On laisse
0 1 3 2 4 5 6 // On compare 1 et 3 : On laisse
0 1 3 2 4 5 6 // On compare 3 et 2 : On inverse
0 1 2 3 4 5 6 // Nous avons terminé notre passage

0 1 2 3 4 5 6 // On compare 0 et 1 : On laisse
0 1 2 3 4 5 6 // On compare 1 et 2 : On laisse
0 1 2 3 4 5 6 // Nous avons terminé notre passage

À ce moment-là, l'algorithme s'arrête car il n'y a plus eu d'échange lors du dernier passage et nous retrouvons notre liste belle et bien triée !

II. Le tri par sélection :

L'idée est simple : rechercher le plus grand élément d'un premier tableau, le placer dans un deuxième tableau en première position, recommencer avec le second plus grand du premier tableau, le placer en deuxième position et ainsi de suite jusqu'à avoir parcouru la totalité du premier tableau.

Pour la suite, on applique l'algorithme en recherchant l'élément le plus grand du tableau, et non le plus petit.

Regardons ensemble ce que donne l'algorithme appliqué à un exemple :

1. Soit le tableau d'entiers suivant :

Tableau 1 :	6	2	8	1	5	3	7	9	4	0	*Tableau 2 : vide*

2. L'élément le plus grand se trouve en 7ème position (si on commence à compter à partir de zéro) :

   Tableau 1 : 6 2 8 1 5 3 7 *9* 4 0 | *Tableau 2 : vide*

3. On échange l'élément le plus grand (en 7ème position) avec le dernier :

   Tableau 1 : 6 2 8 1 5 3 7 *0* 4 | *Tableau 2 : 9*

4. De même, on repère l'élément le plus grand en ignorant le dernier et on l'échange avec l'avant dernier :

   Tableau 1 : 6 2 4 1 5 3 7 0 | *Tableau 2 :* **8** **9**

5. Et ainsi de suite, en ignorant à chaque fois les éléments déjà triés (en gras).

   Tableau 1 : 6 2 4 1 5 3 0 | *Tableau 2 :* **7** **8** **9**

6.

   Tableau 1 : 0 2 4 1 5 3 | *Tableau 2 :* **6** **7** **8** **9**

7.

   Tableau 1 : 0 2 4 1 3 | *Tableau 2 :* **5** **6** **7** **8** **9**

8.

   Tableau 1 : 0 2 3 1 | *Tableau 2 :* **4** **5** **6** **7** **8** **9**

9.

   Tableau 1 : 0 2 1 | *Tableau 2 :* **3** **4** **5** **6** **7** **8** **9**

10.

    Tableau 1 : 0 1 | *Tableau 2 :* **2** **3** **4** **5** **6** **7** **8** **9**

11.

Tableau 1 : 0	*Tableau 2 :*	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**