AP1 TD5 – Tri d'un tableau : correction

Exercice 1 – Tri par sélection

La technique du tri par sélection est la suivante : on met en première position l'élément le plus petit. Puis on met en deuxième position l'élément le plus petit parmi ceux restant. Et ainsi de suite jusqu'au dernier.

Illustrons cette technique sur le tableau suivant :

45	122	12	3	21	78	64	53	89	28	84	46

On commence par rechercher, parmi les douze valeurs, le plus petit élément. On l'identifie en quatrième position (c'est le nombre 3), et on l'échange alors avec le premier élément (le nombre 45). Le tableau devient :

3	122	12	**45**	21	78	64	53	89	28	84	46

On recommence à chercher le plus petit élément, mais cette fois seulement à partir du deuxième (puisque le premier est maintenant correct, on n'y touche plus). On le trouve en troisième position (c'est le nombre 12). On échange donc le deuxième avec le troisième :

3	**12**	**122**	45	21	78	64	53	89	28	84	46

On recommence à chercher le plus petit élément à partir du troisième (les deux premiers sont maintenant bien placés), et on le place correctement, en l'échangeant, ce qui donnera :

3	12	**21**	45	**122**	78	64	53	89	28	84	46

Le processus peut donc être résumé de la manière suivante :

- Boucle principale : on prend comme point de départ le premier élément, puis le second, etc, jusqu'à l'avant dernier.
- Boucle secondaire : à partir de ce point de départ mouvant, on recherche jusqu'à la fin du tableau quel est le plus petit élément et on l'échange avec le point de départ.

Travail à faire. Ecrire l'algorithme du tri à sélection en supposant qu'il est appliqué sur un tableau d'entiers déjà rempli de taille **N** donnée.

```
ALGORITHME tri_selection

VARIABLES :
    T: tableau [1..N] d'entiers
    i, j, min, rgmin : entier

DEBUT
    POUR i ALLANT DE 1 A N-1 FAIRE
    |   min ← T[i]              { technique du « min courant » }
    |   rgmin ← i               { rgmin contient l'indice du min courant }
    |   POUR j ALLANT DE i+1 A N FAIRE
    |   |   SI T[j] < min ALORS
    |   |   |   min ← T[j]
    |   |   |   rgmin ← j
    |   |   FINSI
    |   |   { on inverse le min trouvé et T[i] }
    |   |   T[rgmin] ← T[i]
    |   |   T[i] ← min
    |   FINPOUR
    FINPOUR
FIN
```

Exercice 2 – Tri à bulles

L'idée de départ du tri à bulles consiste à se dire qu'un tableau trié en ordre croissant, c'est un tableau dans lequel **tout élément est plus petit que celui qui le suit**. Cette constatation percutante semble digne de Jacques II de Chabannes, seigneur de La Palice, un ancien voisin à moi. Mais elle est plus profonde – et plus utile - qu'elle n'y parait.

En effet, le principe consiste à prendre chaque élément du tableau et à le comparer avec l'élément qui le suit. Si l'ordre n'est pas bon, on permute ces deux éléments. Et on recommence jusqu'à ce que l'on n'ait plus aucune permutation à effectuer. Les éléments les plus grands « remontent » ainsi peu à peu vers les dernières places, comme les bulles dans une flûte de champagne, ce qui explique la poétique dénomination de « tri à bulle ».

Il est à noter que cette technique de tri nécessite l'utilisation d'un flag. En effet, on ne sait pas combien de « remontées de bulles » seront à réaliser. On sait uniquement que le tableau est bien trié quand il n'y a plus de permutation.

Travail à faire. Ecrire l'algorithme du tri à sélection en supposant qu'il est appliqué sur un tableau d'entiers déjà rempli de taille **N** donnée.

Après étude de la méthode, il ressort que les plus grands éléments remontent le plus vite puisqu'on échange deux cases si celle de gauche est supérieure à celle de droite. On peut par exemple remarquer que l'élément maximum du tableau est placé à sa bonne position dès la première itération car il sera toujours plus grand que n'importe quelle cellule de droite. Ainsi, la fin du tableau est triée « plus rapidement » que le début. C'est donc une piste d'optimisation de l'algorithme : à chaque itération principale, une case de plus est triée à la fin du tableau.

Une question importante à se poser est : combien d'itérations faut-il effectuer pour que le tableau soit complètement trié ? Considérons :

- Le cas le plus favorable : le tableau est déjà trié. Dans ce cas, **une itération** suffit (pour constater que effectivement le tableau est trié).
- Le cas le plus défavorable : le plus petit élément est situé dans la dernière case. Comme il ne peut remonter que d'une case à chaque itération, **N itérations** sont nécessaires.

En conclusion, le nombre d'itérations n'est pas connu à l'avance, mais on est sur d'en faire au moins une : la boucle **REPETER JUSQU'A** s'impose. On construira donc un algorithme qui cessera d'itérer dès que possible, c'est à dire quand le tableau est trié, c'est à dire quand aucune inversion n'est réalisée lors d'une itération.

```
ALGORITHME tri_bulles

VARIABLES :
    T: tableau [1..N] d'entiers
    i, sauve : entiers
    trié : booléen

DEBUT
    REPETER
    |   trié ← vrai                 { on suppose que le tableau est trié }
    |   POUR i ALLANT DE 1 A N-1 FAIRE
    |   |   SI T[i] > T[i+1] ALORS
    |   |   |   sauve ← T[i]
    |   |   |   T[i] ← T[i+1]
    |   |   |   T[i+1] ← sauve
    |   |   |   trié ← faux     { une inversion est réalisée, donc tableau non trié }
    |   |   FINSI
    |   FINPOUR
    |   N ← N-1                   { la fin du tableau est triée : on n'y touche plus }
    JUSQU'A trié = vrai
FIN
```

Exercice 3 – Pour la semaine prochaine...

« Les derniers seront les premiers » - Céline Dion

Ecrire un algorithme qui inverse l'ordre des éléments d'un tableau de taille **N**.

Par exemple :

4	8	7	12

devient

12	7	8	4

On prendra soin d'optimiser l'algorithme de manière à effectuer le moins d'itérations possibles.