MÉTHODE DE TRI RAPIDE (QUICKSORT)

***

La méthode de tri **quicksort** trie *sur place* un tableau d'objets quelconques comparables. C'est la méthode la plus rapide dans le cas *moyen*, lorsque le tableau à trier est "normalement" désordonné. Ses performances se dégradent si le tableau est déjà partiellement rangé.

La méthode consiste à *répartir* les valeurs du tableau par rapport à l'une quelconque de ces valeurs appelée le *pivot*. Les valeurs inférieures sont placées avant le pivot, les valeurs supérieures après. Chacune des deux parties est ensuite triée *récursivement*.

Exemple : Tableau à trier : **6**, 2, 6, 1, 8, 4, 9, 3, 5, 8, 7. Soit 11 valeurs.

Si l'on choisit **6** comme *pivot*, la répartition du reste du tableau pourrait donner (par exemple) : *5, 2, 6, 1, 3, 4,* **6**, 9, 8, 8, 7
où toutes les valeurs inférieures (ou égales) au pivot sont en italiques.

Il reste à trier récursivement ces valeurs en italique, puis celles supérieures à 6 :
1, 2, 3, 4, 5, 6, **6**, 7, 8, 8, 9.

Programmation : ceci doit se faire sans utiliser de tableau annexe.

Pour gérer la récursivité, on définit une procédure : *quick_rec (Tab, inf, sup)* chargée de trier la "tranche" de *Tab* comprise entre les indices *inf* et *sup*. La procédure principale l'appelle une seule fois avec les bornes effectives du tableau.

Dans *quick_rec*, prendre pour pivot la valeur à l'indice *inf*. La répartition se fait en déplaçant le moins possible de valeurs dans le tableau. Dans l'exemple, 5 déplacements suffisent !

On propose de gérer deux variables *bas* et *haut*, avec pour invariant : *toute valeur d'indice inférieur à* **bas** *est inférieure ou égale au pivot, et toute valeur d'indice supérieur à* **haut** *est supérieure ou égale au pivot*. Lorsque *bas = haut*, le pivot peut être placé à cet indice.

Au début, *bas = inf* et *haut = sup*. $Tab[inf]$ est placé dans *pivot*, et la case *bas* est donc libre. Faire d'abord décroître *haut* tant que $Tab[haut] \geq pivot$. Quand $Tab[haut] < pivot$, placer $Tab[haut]$ dans la case libre *bas*.

Sur l'exemple, où *bas=1, haut=11*, les valeurs en 11 et 10 sont supérieures au pivot, mais pas celle en 9. Celle-ci est amenée en *bas=1* :
5, 2, 6, 1, 8, 4, 9, 3, **?**, 8, 7, *et bas=2, haut=9.*
La case libre est maintenant *haut=9*. On repart alors en *montée : bas=3, puis 4,* et enfin *5,* qui est supérieure au pivot. On l'amène donc en *haut=9* :
5, 2, 6, 1, **?**, 4, 9, 3, 8, 8, 7, *et bas=5, haut=8, descente*
On reprend la décroissance de *haut* ; ici $Tab[haut] < pivot$ dès le départ :
5, 2, 6, 1, 3, 4, 9, **?**, 8, 8, 7, *et bas=6, haut=8, montée.*
5, 2, 6, 1, 3, 4, 9, **?**, 8, 8, 7, *et bas=7, haut=8, montée.*
5, 2, 6, 1, 3, 4, **?**, 9, 8, 8, 7, *et bas=7, haut=7.*
L'indice 7 peut donc être utilisé pour placer le pivot :
5, 2, 6, 1, 3, 4, **6**, 9, 8, 8, 7.

Il reste à appeler récursivement *quick_rec (Tab,1,6)*, puis *quick_rec (Tab,8,11)* pour que le tableau soit entièrement trié.

La répartition peut être effectuée par une seule itération, dans laquelle le sens montée-descente est indiqué par un booléen, que l'on inverse chaque fois qu'un élément est déplacé.

La récursivité s'arrête au plus tard si la "tranche" ne contient qu'un élément. Mais on gagne du temps en triant directement les tranches de longueur 2 (et même 3 ?).