[LC600]

# *TP C : Tri rapide (suppression de la récursion)*

L' algorithme de tri rapide (**quick sort**) est basé sur une procédure récursive, du type "diviser pour résoudre", dont la structure est la suivante (pour un tableau d'entiers) :

Structure récursive du Tri rapide

```
TriRapide (int t[], int g, int d)
   {
    int i;
    if (d>g)
      {
       i=Partitionnement(t,g,d);
       TriRapide(t,g,i-1);
       TriRapide(t,i+1,d);
      }
   }
```

```
/* t est le tableau à trier, g et d sont
   les index des éléments le plus à gauche
   et le plus à droite respectivement. On
   pourrait aussi n'utiliser que deux
   arguments qui seraient des pointeurs vers
   ces éléments de gauche et de droite.   */
```

La seule partie délicate est le partitionnement (qui dans la procédure ci-dessus apparaît comme une fonction, mais ce n'est pas indispensable et ce serait plutôt à éviter).

Le principe du partitionnement est de :

- Choisir un élément pivot dans le tableau et le mettre de côté (Il ne doit pas être parcouru dans l'étape suivante).
- Parcourir le tableau depuis la gauche jusqu'à rencontrer un élément supérieur (ou égal) au pivot et parcourir le tableau depuis la droite jusqu'à rencontrer un élément inférieur (ou égal) au pivot. Les deux éléments ainsi trouvés sont alors échangés. En continuant ce processus on s'assure que tous les éléments situés à gauche de l'élément courant gauche sont inférieurs (ou égaux) au pivot et que tous les éléments situés à droite de l'élément courant droit sont supérieurs (ou égaux) au pivot. On arrête ce processus lorsque les pointeurs vers les deux éléments courants se croisent. Il ne reste plus alors qu'à remettre le pivot à la bonne place.

Pour ne pas surcharger les tests des boucles de parcours, il est bon de prévoir des éléments "sentinelles" (internes ou externes au tableau), c'est à dire des éléments dont on est sur qu'il vérifieront le test de la boucle correspondante. (Voir plus loin une solution pour obtenir de telles sentinelles).

La figure ci-dessous illustre le partitionnement. L'élément pivot est en fait dans cet exemple l'élément le plus à droite (**R**) du tableau, ce qui est commode pour le "mettre de côté", mais pas forcément le meilleur choix de valeur. Son placement final consiste alors en un échange avec l'élément le plus à gauche du sous-tableau de droite (**T**).

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| U | N | E | P | H | R | A | S | E | A | T | R | I | E | **R** |
| **U** | | | | | | | | | | | | | E | R |
| E | N | E | P | H | **R** | | | | | | | **I** | U | R |
| E | N | E | P | H | I | A | **S** | | | | **R** | R | U | R |
| E | N | E | P | H | I | A | R | E | A | **T** | S | R | U | R |
| E | N | E | P | H | I | A | R | E | A | **R** | S | R | U | **T** |

g d

Une autre possibilité, pour le choix du pivot, consiste à prendre trois éléments (par exemple : le plus à gauche, celui du milieu et le plus à droite) et à les classer (de façon élémentaire) afin que le plus petit soit à gauche, le plus grand au milieu et le médian à droite (et ce sera le pivot). Cette méthode conduit à une plus grande probabilité d'obtenir un pivot de valeur moyenne, c'est à dire conduisant à un partitionnement

équilibré. En outre elle place automatiquement des "sentinelles" internes aux extrémités du tableau à trier, puisque l'élément le plus à droite, égal au pivot, arrêtera le parcours vers la droite et que l'élément le plus à gauche, inférieur ou égal au pivot, arrêtera le parcours vers la gauche.

## Questions posées :

**1)** Ecrire la procédure de tri rapide récursive complète (avec son partitionnement et en utilisant le choix de pivot par 3 éléments). Si le nombre d'éléments à trier est inférieur à une limite (que vous choisirez) on se contentera de faire un tri élémentaire.

**2)** Utiliser le procédé de "suppression de récursion" vu sur les exemples du cours pour obtenir une procédure de tri rapide non récursif. On expliquera dans le dossier les étapes intermédiaires qui conduisent au résultat.

**3)** Ecrire un programme principal ayant deux argument sur la ligne de commande :

- le premier argument est une lettre indiquant le tri à faire :
  **r** = tri rapide récursif du 1°)
  **n** = tri rapide non récursif du 2°)
  **c** = tri rapide de la bibliothèque standard du C (fonction **qsort**)
  **a** = aucun tri (afin de mesurer le temps mis pour faire les copies)
- le deuxieme argument est le nombre de fois *n* que le tri (plus précisément les points (2) et (3) ci-dessous) est exécuté (~~défaut=1~~). Cet argument sert à faire des chronométrages plus précis (en divisant le temps trouvé par ce nombre).

Après l'analyse de ses arguments, ce programme principal devra faire :

(1) une saisie d'un tableau A d'entiers (à partir de l'entrée standard ou à partir d'un fichier)

*n* fois

(2) une copie du tableau A vers un autre tableau B le plus rapidement possible (utiliser par exemple la fonction memcpy)

(3) un appel du tri sélectionné par le premier argument pour le tableau B. Si aucun tri, ce point est sauté.

(4) un affichage du tableau B trié sur la sortie standard

**4)** En utilisant la commande **time** de UNIX (voir le **man**uel), mesurer, avec une valeur suffisamment grande pour *n*, le temps d'exécution CPU du programme précédent pour trier un même tableau[1] avec chacune des procédures de tri. On mesurera aussi dans le cas aucun tri, afin de pouvoir déterminer précisément par différence, puis par division par *n*, les durées d'exécutions respectives des trois procédures de tri sur ce tableau.

Indiquez dans votre dossier les différents temps d'exécution trouvés, ainsi que la taille du tableau trié. Eventuellement donnez aussi le tableau s'il n'est pas trop grand. ~~Il est possible que je mette~~ Il y a un fichier **tablent** contenant un tableau à trier dans le répertoire /users/tpinfo/~~blanc~~.

Conclure.

**5)** Prolongements possibles :

a) Si vous connaissez d'autres algorithmes de tris, vous pouvez écrire les procédures correspondantes et déterminer leurs temps d'exécution respectifs par le même procédé et les comparer à ceux que vous aurez trouvés pour les procédures de tri rapide.

b) Imaginez, et réalisez en utilisant la bibliothèque **curses,** une visualisation des tris précédents.

[1] Il est conseillé pour cela d'utiliser les redirections d'entrée/sortie standard. Il est conseillé en outre de choisir un tableau de taille suffisante pour que le temps de tri ne soit pas trop faible

argc argv[].