Le tri rapide

1 Affectations avec XCAS

1.1 Copie de l'objet ou de l'adresse

Pour la séquence suivante :

Xcas
```
a:=1:;b:=a:;a:=3:;a;b
```

on obtient l'affichage : 3,1
b est un "nouvel" objet, une modification de *a* ne modifie pas *b*.

Pour la séquence suivante :

Xcas
```
a:=[1,2,3]:;b:=copy(a):;a[0]:=15:;a;b
```

on obtient : [15,2,3],[1,2,3]
b est inchangé, l'instruction copy ayant créé une copie de la liste *a* à une nouvelle adresse.

Mais la copie de listes demandant du temps et de l'espace mémoire, il est souvent intéressant d'éviter la création d'une copie de la liste.
La séquence suivante :

Xcas
```
a:=[1,2,3]:;b:=a:;a[0]:=15:;a;b
```

donne le même effet mais pour des raisons différentes : lorsqu'on tape b:=a, b pointe sur la même adresse que l'objet a. Mais écrire a[0]:=15 crée une duplication de la liste *a* en modifiant l'élément d'indice 0.

Si l'on entre maintenant la séquence :

Xcas
```
a:=[1,2,3]:;b:=a:;a[0]=<15:;a;b
```

on obtient : [15,2,3],[15,2,3]
L'affectation avec =< a permis d'éviter la création d'une copie du tableau *a* : l'élément d'indice 0 a été modifié sur l'objet initial. Comme *b* pointe sur la même adresse, il se trouve du coup également modifié.

C'est ce dernier type d'affectation qui nous intéresse dans l'algorithme de tri puisqu'il évite de dupliquer la liste (ce qui est une perte de temps et de place mémoire).

Remarque. Pour les différences de signification entre ces affectations, vous pouvez consulter le menu « aide » de xcas. Dans la rubrique « rechercher un mot », entrez le mot « affectation » et choisissez par exemple, dans les rubriques qui seront alors proposées, la rubrique « affectation infixée ».

1.2 Illustration avec procédure

Entrons la procédure suivante en XCAS qui est censée échanger les deux premiers éléments de la liste passée en paramètre :

Xcas
```
echanger(T):={
local tmp;
tmp=<T[0];T[0]:=T[1];T[1]=<tmp;
};;
```

et testons la avec la ligne de commandes :

Xcas
```
B:=[2,3,4,5];
```

suivie de la ligne de commandes :

Xcas
```
echanger(B);;B
```

Et on obtient l'affichage : [2,3,4,5]
B n'a pas été modifiée, c'est une copie de B qui a été modifiée.
Par contre avec :

Xcas
```
echange(T):={
local tmp;
tmp=<T[0];T[0]=<T[1];T[1]=<tmp;
};;
```

on constate que B est bien modifiée en sortie.

2 Fonction de partition "sur place"

2.1 Principe de la procédure partition

L'objectif est d'exécuter l'étape principale sans créer de liste mais en effectuant des échanges entre des éléments de l'unique liste créée.
Illustration avec la liste A :=[5,2,12,3,7,4,14,6,9,8,1,13] .

Détails du déroulement
clef := $A[0]$, deb = 0, fin = 11.

1. Étape 1 :

indices	0	gauche=1	2	3	4	5	6	7	8	9	10	droite=11
valeurs	5	2	12	3	7	4	14	6	9	8	1	13

2. Étape 2 :
   Tant que le curseur gauche indique une cellule de contenu <=clef, on incrémente gauche. Et tant que le curseur droite indique une cellule de contenu > clef, on décrémente droite :

indices	0	1	gauche=2	3	4	5	6	7	8	9	droite=10	11
valeurs	5	2	12	3	7	4	14	6	9	8	1	13

On échange alors les contenus des cellules ciblées par gauche et gauche :

indices	0	1	gauche=2	3	4	5	6	7	8	9	droite=10	11
valeurs	5	2	1	3	7	4	14	6	9	8	12	13

et on incrémente gauche et décrémente droite :

indices	0	1	2	gauche=3	4	5	6	7	8	droite=9	10	11
valeurs	5	2	1	3	7	4	14	6	9	8	12	13

3. Étape 3

   On recommence à incrémenter gauche tant qu'il indique une cellule qui doit rester à gauche et à décrémenter droite tant qu'il indique une cellule qui doit rester à droite :

indices	0	1	2	3	gauche=4	droite=5	6	7	8	9	10	11
valeurs	5	2	1	3	7	4	14	6	9	8	12	13

   on échange :

indices	0	1	2	3	gauche=4	droite=5	6	7	8	9	10	11
valeurs	5	2	1	3	4	7	14	6	9	8	12	13

   et on incrémente gauche et décrémente droite :

indices	0	1	2	3	droite=4	gauche=5	6	7	8	9	10	11
valeurs	5	2	1	3	4	7	14	6	9	8	12	13

4. Étape 4

   A ce stade (gauche>droite), il ne reste plus qu'à placer la clef à son emplacement définitif en échangeant le premier élément du tableau avec l'élément de la case ciblée par le curseur droite :

indices	0	1	2	3	4	5	6	7	8	9	10	11
valeurs	4	2	1	3	5	7	14	6	9	8	12	13

5. Étape 5

   La fonction retourne 4, c'est à dire la position de l'élément clef qui délimite les tableaux gauche et droite (qui seront triés par appels récursifs).

2.2 Procédure partition

Une traduction Xcas de la procédure décrite ci-dessus :

Xcas

```
partition(T,deb,fin):={
local g,d,tmp,clef;
/* g : curseur pour tableau gauche, d : curseur pour tableau droit */
clef:=T[deb];g:=deb+1;d:=fin;
repeter
        tantque g<=fin faire
                si T[g]<=clef alors g:=g+1 sinon break;fsi;
        ftantque;
        tantque d>=deb faire
                si T[d]>clef alors d:=d-1 sinon break;fsi;
        ftantque;
        si g<d alors
                /* échange des contenus de T[g] et T[d] : */
                tmp=<T[g];T[g]=<T[d];T[d]=<tmp;
                g:=g+1;d:=d-1;
        fsi;
jusqu_a g>d;
/*echange de T[d] et T[clef] afin que la valeur clef soit définitivement bien
    placée : */
T[deb]=<T[d];T[d]=<clef;
retourne d }:;
```

2.3 Essai de la procédure

On entre la ligne de commande A:=[5,2,12,3,7,4,14,6,9,8,1,13]
puis la ligne de commande : partition(A,0,11);A
On obtient : 4,[4,2,1,3,5,7,14,6,9,8,12,13]

3 Procédure tri rapide

3.1 La procédure tri rapide

La procédure en xcas :

Xcas

```
trirap(T,deb,fin):={
local indiceclef;
si deb<fin alors
indiceclef:=partition(T,deb,fin);
trirap(T,deb,indiceclef-1);trirap(T,indiceclef+1,fin);
fsi;}:;
```

3.2 Essai de la procédure

on entre la ligne de commandes :

Xcas

```
B:=[5,10,12,3,7,4,2,6,9,8,1,13]:;  trirap(B,0,dim(B)-1):;B
```

et on obtient :
[1,2,3,4,5,6,7,8,9,10,12,13]

Si on tient à garder B intacte :

Xcas

```
B:=[5,10,12,3,7,4,2,6,9,8,1,13]:;C:=copy(B):;  trirap(C,0,dim(C)-1):;B;C
```

Dans ce cas B est inchangée, C est triée.