

## Plan

---

- Algorithmes
- Types abstraits de données
- Pointeurs
- Listes
- Piles
- Files
- Tris**
- Complexité

01/09/2006

1

## Tris > Définition

---

- Données de base : une liste de  $n$  éléments. A chaque élément est associée une clé. Les clés appartiennent à un ensemble sur lequel on dispose d'un ordre total.
- Résultat : une liste dont les éléments sont une permutation des éléments de la liste d'origine.
- Les clés sont croissantes quand on parcourt la liste séquentielles (cas général des listes triées) .

01/09/2006

2

## Tris > La sorte Clé

---

- Sorte Clé  
Utilise Booléen
- Opérations :
- $\leq$  : Clé  $\otimes$  Clé  $\rightarrow$  Booléen ;
- Avec :
- $x, y, z$  : Clé

01/09/2006

3

## Tris > La sorte Clé

---

- Axiomes :  
 $x \leq y = \text{vrai}$   
 $(x \leq y) \wedge (y \leq x) \Rightarrow x = y$   
 $(x \leq y) \wedge (y \leq z) \Rightarrow x \leq z$

On appelle clé l'application, qui à chaque élément associe sa clé :

Clé: Élément  $\rightarrow$  Clé

01/09/2006

4

## Tris > Tri stable

---

- ❑ tri stable : conserve l'ordre d'origine des éléments dont les clés sont égales.
- ❑ Important en cas de tri multi-critères (multi-clés)

01/09/2006

5

## Tris > Tris internes et tris externes

---

- ❑ Tri interne : opère sur des données présentes en mémoire centrale.
- ❑ Tri externe : opère sur des données appartenant à des fichiers.
- ❑ Problème d'optimisation différents :  
En tri interne, on cherche à réduire le nombre de comparaison et de toutes les opérations internes. En tri externe, on s'intéresse aux comparaisons et surtout aux entrées sorties.

01/09/2006

6

## Tris > Tris itératifs

---

- ❑ On trie le tableau d'un seul bloc, en le parcourant élément par élément, case par case.
- ❑ Deux méthodes:
  - Prendre chaque case du tableau et choisir l'élément devant être stockés dans la case courante: c'est le tri par sélection,
  - Prendre chaque élément dans l'ordre où il se présente, et le mettre à la bonne place : c'est le tri par insertion.

01/09/2006

7

## Tris > Tris itératifs > Tris par sélection

---

- ❑ Le principe du tri par sélection:
  - On parcourt le tableau de la première à la dernière case. A chaque étape, on recherche dans la partie non triée du tableau l'élément à placer dans la case courante.
- ❑ Comment ?
  - 1ère itération: On place le plus petit élément dans la première case.
  - 2ème itération: Il ne nous reste plus qu'à trier le reste du tableau, de la case 2 à la case  $n$ . On place alors le plus petit élément dans la deuxième case.
  - 3ème itération: Les deux premières cases sont triées, on trie alors le reste du tableau, de la case 3 à la case  $n$ .
  - Ainsi de suite...
- ❑ Pour résumer :
  - Pour  $i$  allant de 1 à  $n$ , on place le minimum de  $t[i..n]$  en  $t[i]$

01/09/2006

8

Tris > Tris itératifs > Tris par sélection > Tri par minimum successifs

□ Principe :

- Pour chaque case du tableau  $i$ , on recherche la position du minimum dans le sous-tableau  $t[i..n]$ . Puis on échange le contenu entre la case du minimum et la case courante. La case courante contient donc son élément final après l'échange.

□ Algorithme :

- Pour  $i$  allant de 1 à  $n$  :  
on recherche séquentiellement le minimum dans  $t[i..n]$ ,  
on échange  $t[i]$  et le minimum.

01/09/2006

9

Tris > Tris itératifs > Tris par sélection > Tri par minimum successifs

Procédure Tri-min ( In\_Out t : Tableau [ 1..n ] de élément ) ;

```

Var
  i, j, k : 1..n ;
Début
  POUR i := 1 A n-1 FAIRE
    j := i ;
    POUR k := i + 1 A n FAIRE
      SI t[k] < t[j]
        ALORS
          j := k ;
    FINSI ;
    FINPOUR ;
    SI i != j
      ALORS
        Permuter ( t[j], t[i] ) ;
    FINSI ;
  FINPOUR ;
Fin ;
    
```

01/09/2006

10

Tris > Tris itératifs > Tris par sélection > Tri par minimum successifs > Exemple

3	2	5	1	5	4	3	6
---	---	---	---	---	---	---	---

- Ici,  $i = 1$ . On doit donc placer en première position le plus petit élément de  $t[1..n]$ , c'est-à-dire de tout le tableau. C'est la valeur 1. On inverse donc les places de ces deux valeurs :

1	2	5	3	5	4	3	6
---	---	---	---	---	---	---	---

01/09/2006

11

Tris > Tris itératifs > Tris par sélection > Tri par minimum successifs > Exemple

- $t[1..1]$  est maintenant trié, on va trier la suite du tableau,  $t[2..n]$  :

1	2	5	3	5	4	3	6
---	---	---	---	---	---	---	---

- 2, la plus petite valeur de  $t[2..n]$ , est déjà à sa place : pas d'inversion à faire ici.
- $t[1..2]$  est trié, on va trier  $t[3..n]$  :

1	2	5	3	5	4	3	6
---	---	---	---	---	---	---	---

01/09/2006

12

Tris > Tris itératifs > Tris par sélection > Tri par minimum successifs > Exemple

- Le plus petit élément est la valeur 3, que l'on va échanger avec la valeur 5 :

1	2	3	5	5	4	3	6
---	---	---	---	---	---	---	---

- $t[1..3]$  est trié, on trie maintenant  $t[4..n]$ , et ainsi de suite :

1	2	3	5	5	4	3	6
---	---	---	---	---	---	---	---

1	2	3	3	5	4	5	6
---	---	---	---	---	---	---	---

1	2	3	3	5	4	5	6
---	---	---	---	---	---	---	---

01/09/2006

13

Tris > Tris itératifs > Tris par sélection > Tri par minimum successifs > Exemple

1	2	3	3	4	5	5	6
---	---	---	---	---	---	---	---

1	2	3	3	4	5	5	6
---	---	---	---	---	---	---	---

1	2	3	3	4	5	5	6
---	---	---	---	---	---	---	---

1	2	3	3	4	5	5	6
---	---	---	---	---	---	---	---

1	2	3	3	4	5	5	6
---	---	---	---	---	---	---	---

01/09/2006

14

Tris > Tris itératifs > Tris par sélection > Tri à bulles

- Principe :

- On parcourt le tableau en inversant les éléments successifs, s'ils ne sont pas ordonnés. On répète l'opération jusqu'à l'obtention d'un tableau trié. Ainsi, les petits éléments se déplacent peu à peu au début du tableau, et les gros éléments sont poussés vers la fin du tableau.

- Algorithme

- Pour  $i$  allant de 1 à  $n$  :
  - on parcourt séquentiellement  $t[n..i+1]$ ,
  - on échange  $t[j]$  et  $t[j-1]$  s'ils ne sont pas ordonnés.

01/09/2006

15

Tris > Tris itératifs > Tris par sélection > Tri à bulles

Procédure Tri-bulles ( In\_Out t : Tableau [ 1..n ] de élément ) ;

Var

i, j : 1..n ;

Début

POUR i := 1 A n-1 FAIRE

POUR j := n A i + 1 PAS -1 FAIRE

SI t [ j ] < t [ j - 1 ]

ALORS

Permuter ( t [ j ] , t [ j - 1 ] );

FINSI;

FINPOUR;

FINTANTQUE;

Fin

01/09/2006

16

Tris > Tris itératifs > Tris par sélection > Tri à bulles > Exemple

□ Légende :

éléments comparés mais ordonnés,  
donc non inversés

éléments comparés et inversés,  
car non ordonnés précédemment

3	2	5	1	5	4	3	6
---	---	---	---	---	---	---	---

01/09/2006

17

Tris > Tris itératifs > Tris par sélection > Tri à bulles > Exemple

□ Ici,  $i = 1$ . On se déplace de  $n$  à  $2$ , en inversant successivement les valeurs mal ordonnées :

3	2	5	1	5	4	3	6
3	2	5	1	5	3	4	6
3	2	5	1	3	5	4	6
3	2	5	1	3	5	4	6
3	2	1	5	3	5	4	6
3	1	2	5	3	5	4	6
1	3	2	5	3	5	4	6

01/09/2006

18

Tris > Tris itératifs > Tris par sélection > Tri à bulles > Exemple

□ Désormais,  $t[1..1]$  est trié. Occupons-nous maintenant de  $t[2..n]$  :

1	3	2	5	3	5	4	6
1	3	2	5	3	5	4	6
1	3	2	5	3	4	5	6
1	3	2	5	3	4	5	6
1	3	2	3	5	4	5	6
1	3	2	3	5	4	5	6
1	2	3	3	5	4	5	6

01/09/2006

19

Tris > Tris itératifs > Tris par sélection > Tri à bulles > Exemple

□ La valeur 2 est à sa place, et maintenant  $t[1..2]$  est trié. Continuons avec  $t[3..n]$  :

1	2	3	3	5	4	5	6
1	2	3	3	5	4	5	6
1	2	3	3	5	4	5	6
1	2	3	3	4	5	5	6
1	2	3	3	4	5	5	6
1	2	3	3	4	5	5	6

01/09/2006

20

## Tris > Tris itératifs > Tris par sélection > Tri à bulles > Exemple

- Désormais,  $t[1..3]$  est trié, nous nous occupons de  $t[4..n]$ . Le tableau est en fait déjà trié, mais l'algorithme ne le "sait" pas encore. Il va s'en rendre compte ici :

1	2	3	3	4	5	5	6
1	2	3	3	4	5	5	6
1	2	3	3	4	5	5	6
1	2	3	3	4	5	5	6
1	2	3	3	4	5	5	6

- Dans ce parcours, on n'a effectué aucune inversion: le tableau est trié, la procédure se termine.

01/09/2006

21

## Tris > Tris itératifs > Tri par insertion

- Principe :

- On part du principe que  $t[1..1]$  est trié. Puis, pour chaque élément  $i$  du tableau, on recherche la position de l'élément  $t[i]$  dans le tableau  $t[1..i-1]$ . Puis on insère  $t[i]$  dans  $t[1..i-1]$  en fonction de cette position. Comme  $t[1..i-1]$  est trié,  $t[1..i]$  l'est aussi, après insertion.

- On aura donc :

- Pour chaque élément  $i$ :
  - Rechercher la position d'insertion de  $t[i]$
  - Insérer  $t[i]$  dans  $t[1..i-1]$ .

01/09/2006

22

## Tris > Tris itératifs > Tri par insertion

```

Procédure Tri-insert ( In_Out t : t[ 1..n ] de Elément ) ;
Var
  i, j : 1..n; k : 2..n; temp : Elément;
Début
  POUR i := 2 à n FAIRE
    k := 1;
    TANTQUE (t[i] >= t[k]) ET (k < i) FAIRE
      k := k + 1;
    FINTANTQUE;
    SI k ≠ i
    ALORS
      temp := t[i];
      POUR j := i À k+1 PAS -1 FAIRE
        t[j] := t[j-1];
      FINPOUR
      t[k] := temp;
    FINSI
  FINPOUR;
Fin
    
```

01/09/2006

23

## Tris > Tris itératifs > Tri par insertion > Exemple

- Légende :

Partie du tableau déjà triée							
élément courant, avant son insertion							
éléments déplacés après une insertion							

3	2	5	1	5	4	3	6
---	---	---	---	---	---	---	---

- Comme on le voit,  $t[1..1]$  est considéré comme étant trié. On commence donc l'algorithme, avec  $i = 2$  :

3	2	5	1	5	4	3	6
---	---	---	---	---	---	---	---

01/09/2006

24

### Tris > Tris itératifs > Tri par insertion > Exemple

- On insère donc l'élément courant, la valeur 2, à sa place dans la première partie du tableau, ce qui nous oblige à décaler la valeur 3 :

2	3	5	1	5	4	3	6
---	---	---	---	---	---	---	---

- Itération suivante :  $i = 3$ . L'élément courant est 5.

2	3	5	1	5	4	3	6
---	---	---	---	---	---	---	---

01/09/2006

25

### Tris > Tris itératifs > Tri par insertion > Exemple

- Ici, l'élément était bien placé : il n'y a donc pas de décalage à effectuer pour l'insérer.

2	3	5	1	5	4	3	6
---	---	---	---	---	---	---	---

- Désormais,  $t[1..3]$  est trié. On passe à l'itération  $i = 4$ , avec pour valeur à insérer 1 :

2	3	5	1	5	4	3	6
---	---	---	---	---	---	---	---

01/09/2006

26

### Tris > Tris itératifs > Tri par insertion > Exemple

- L'élément va en première position : il faut décaler toutes les premières valeurs du tableau :

1	2	3	5	5	4	3	6
---	---	---	---	---	---	---	---

- $t[1..4]$  est trié, maintenant  $i = 5$  :

1	2	3	5	5	4	3	6
---	---	---	---	---	---	---	---

01/09/2006

27

### Tris > Tris itératifs > Tri par insertion > Exemple

- Pas de décalage à effectuer pour l'insertion :

1	2	3	5	5	4	3	6
---	---	---	---	---	---	---	---

- $i = 6$  :

1	2	3	5	5	4	3	6
---	---	---	---	---	---	---	---

01/09/2006

28

## Tris > Tris itératifs > Tri par insertion > Exemple

- On insère l'élément à sa place, ce qui nous amène à décaler une partie seulement des éléments :

1	2	3	4	5	5	3	6
---	---	---	---	---	---	---	---

- t [1..6] est trié :

1	2	3	4	5	5	3	6
---	---	---	---	---	---	---	---

01/09/2006

29

## Tris > Tris itératifs > Tri par insertion > Exemple

- i = 7 :

1	2	3	4	5	5	3	6
---	---	---	---	---	---	---	---

1	2	3	3	4	5	5	6
---	---	---	---	---	---	---	---

- t [1..7] est trié :

1	2	3	3	4	5	5	6
---	---	---	---	---	---	---	---

- On passe au dernier élément, i = 8 :

1	2	3	3	4	5	5	6
---	---	---	---	---	---	---	---

01/09/2006

30

## Tris > Tris itératifs > Tri par insertion > Exemple

- Pas de décalage à effectuer : le tableau est désormais trié :

1	2	3	3	4	5	5	6
---	---	---	---	---	---	---	---

01/09/2006

31

## Tris > Tris récursifs

- Dans un tri récursif, on applique le principe *diviser pour mieux régner* ! En effet, on va diviser le tableau en deux, trier chacun des deux tableaux séparément, puis combiner les deux tableaux triés.
- On aura donc :
  - Diviser le tableau en 2
  - Appeler récursivement la procédure pour chacun des tableaux
  - Combiner les deux tableaux

01/09/2006

32

## Tris > Tris récursifs > Tri par fusion

### □ Diviser

- Dans le tri par fusion, on découpe le tableau à trier en deux tableaux de même longueur (à un élément près si le nombre d'éléments est impair).
- On aura donc deux sous-tableaux :  $t$  [*debut*..*milieu*] et  $t$  [*milieu+1*..*fin*].

### □ Appel récursif

- On appelle la procédure récursivement pour  $t$  [*debut*..*milieu*] et pour  $t$  [*milieu+1*..*fin*].

### □ Combiner

- Les deux sous-tableaux sont triés, mais rien n'indique que tous les éléments les plus petits soient placés dans le premier sous-tableau, et inversement. Il va donc falloir *fusionner* les deux sous-tableaux.

01/09/2006

33

## Tris > Tris récursifs > Tri par fusion

Procédure Tri-fusion( In\_Out t : Tableau [ 1..n ] de Elément ) ;  
Var

```
m : 1..n ;  
Début  
  SI n > 2  
  ALORS  
    m := n DIV 2 ;  
    Tri-fusion ( t [ 1..m ] ) ;  
    Tri-fusion ( t [ m + 1..n ] ) ;  
    Fusionner ( t [ 1..m ] , t [ m + 1..n ] ) ;  
  SINON  
    Si t[1] > t[n]  
    ALORS  
      Permuter(t[1],t[n]);  
    FINSI ;  
  FINSI ;  
Fin
```

01/09/2006

34

## Tris > Tris récursifs > Tri par fusion

□ Procédure Fusionner ( In\_Out t : Tableau [ 1..m ] de Elément, In\_Out t2 : Tableau [ 1..n ] de Elément ) ;

```
Var  
  i : 1..m+1 ; j : 1..n+1 ;  
  Res : Tableau [ 1..m+n ] de Elément;  
Début  
  i := 1 ; j := 1 ;  
  POUR k := 1 A m+n FAIRE  
    SI ((i < m+1) ET (j < n+1) ET (t[i] < t2[j])) OU (j = n+1)  
    ALORS  
      Res[k] := t[i]  
      i := i + 1 ;  
    SINON  
      Res[k] := t2[j]  
      j := j + 1 ;  
    FINSI ;  
  FINPOUR ;  
  t[1..m] := Res[1..m] ;  
  t2[1..n] := Res[m+1..m+n] ;  
Fin
```

01/09/2006

35

## Tris > Tris récursifs > Tri par fusion > Exemple de fusion

□ Voici deux tableaux triés, t1 et t2, à fusionner :

I			
1	5	7	10

□ et

J			
3	4	6	9

01/09/2006

36

## Tris > Tris récursifs > Tri par fusion > Exemple de fusion

- Fusionnons ces deux tableaux : on prend le plus petit élément entre  $t1[i]$  et  $t2[j]$  : c'est  $t1[i]$ , à savoir 1. On place 1 au début du tableau, et on incrémente  $i$ .

	I		
1	5	7	10

- et

J			
3	4	6	9

- résultat :

1							
---	--	--	--	--	--	--	--

01/09/2006

37

## Tris > Tris récursifs > Tri par fusion > Exemple de fusion

- Cette fois, le plus petit élément est  $t2[j]$  : on place 3 dans le tableau, on place 3 dans le tableau, et on incrémente  $j$ .

	I		
1	5	7	10

- et

J			
3	4	6	9

- résultat :

1	3						
---	---	--	--	--	--	--	--

- Et ainsi de suite...

01/09/2006

38

## Tris > Tris récursifs > Tri par fusion > Exemple de fusion

- Pour l'avant dernière affectation,  $t2[j]$  est le plus petit élément. on place 3 dans le tableau, et on incrémente  $j$ .

			I
1	5	7	10

- et

3	4	6	9

J = 5

- résultat :

1	3	4	5	6	7	9	
---	---	---	---	---	---	---	--

01/09/2006

39

## Tris > Tris récursifs > Tri par fusion > Exemple

- Découpons le tableau en 2 sous-tableaux de même taille :

3	2	5	1	5	4	3	6
---	---	---	---	---	---	---	---

- Nous allons appeler récursivement la procédure sur les tableaux vert et rose. Il faudra alors fusionner ces tableaux.

3	2	5	1				
---	---	---	---	--	--	--	--

- En divisant ce tableau en 2, on obtient :

3	2	5	1				
---	---	---	---	--	--	--	--

01/09/2006

40

## Tris > Tris récursifs > Tri par fusion > Exemple

- Il nous faut maintenant trier chacun de ces sous-tableaux. Ce sont des tableaux de 2 éléments, donc des cas terminaux : si les éléments ne sont pas ordonnés, les permuter. Ce qui nous donnera :

2	3	1	5				
---	---	---	---	--	--	--	--

- Reste à fusionner les deux tableaux :

1	2	3	5				
---	---	---	---	--	--	--	--

- Fin de la procédure pour le sous-tableau de gauche. Il faut maintenant trier celui de droite :

				5	4	3	6
--	--	--	--	---	---	---	---

01/09/2006

41

## Tris > Tris récursifs > Tri par fusion > Exemple

- Après division, on a :

				5	4	3	6
--	--	--	--	---	---	---	---

- On trie le sous-tableau de gauche, puis celui de droite. Ce sont des cas terminaux, nous ne détaillons donc pas ici.

				4	5	3	6
--	--	--	--	---	---	---	---

- Reste à fusionner ces deux tableaux, puis fin de l'appel récursif :

				3	4	5	6
--	--	--	--	---	---	---	---

01/09/2006

42

## Tris > Tris récursifs > Tri par fusion > Exemple

- Nos deux sous-tableaux initiaux sont donc maintenant triés:

1	2	3	5	3	4	5	6
---	---	---	---	---	---	---	---

- Reste à les fusionner, ce qui donnera:

1	2	3	3	4	5	5	6
---	---	---	---	---	---	---	---

01/09/2006

43

## Tris > Tris récursifs > Tri rapide

- Diviser**
  - Il faut d'abord choisir un élément dans le tableau, nommé pivot et noté ci-après  $p$ .
  - Une fois  $p$  choisi, il faut réorganiser le tableau de telle sorte que tous les éléments inférieurs au pivot soient placés au début du tableau, et tous les éléments supérieurs ou égaux soient placés à la fin du tableau. Ainsi, on peut découper le tableau de telle sorte que : pour tout élément  $x$  de  $t1$  et pour tout  $y$  de  $t2$ , on ait  $x \leq y$ .
  - Il est à noter que les deux sous-tableaux ainsi obtenus ne sont pas forcément de la même taille, en fonction du pivot qui aura été choisi.
- Appel récursif**
  - On appelle la procédure récursivement pour  $t1$  et pour  $t2$ .
- Combiner**
  - Puisque tous les éléments du premier sous-tableau sont inférieurs à n'importe quel élément du second, une fois chacun des sous-tableaux trié, il suffit de les accoler pour obtenir le tableau initial trié. La phase de combinaison est donc implicite.

01/09/2006

44

## Tris > Tris récursifs > Tri rapide

```

PROCEDURE Trirapide (
  T : IN OUT Vect;
  DEBUT,
  FIN : Integer) IS
  Pivot : Integer;
  N : Integer := FIN - DEBUT + 1;
  BEGIN
  IF (N = 2) THEN
    IF (T(DEBUT) > T(FIN)) THEN
      Permuter(T,DEBUT,FIN);
    END IF;
  ELSE
    IF (N > 2) THEN
      Partition (T, DEBUT, FIN, Pivot);
      Trirapide(T,DEBUT,Pivot-1);
      Trirapide(T,Pivot+1, FIN);
    END IF;
  END IF;
END Trirapide;

```

01/09/2006

45

## Tris > Tris récursifs > Tri rapide

```

PROCEDURE Partition (T : IN OUT Vect; DEBUT, FIN : Integer; Pivot : IN OUT Integer) IS
  L : Integer;
  M : Integer;
  BEGIN
  -- calcul et positionnement du pivot au début du tableau
  Pivot := (DEBUT+FIN) / 2 + (DEBUT+FIN) MOD 2;
  Permuter(T, Pivot, DEBUT);
  L:=DEBUT+1;
  M:=FIN;
  -- Déplacement des cases inférieures au pivot en début de tableau, des cases supérieures à la fin
  WHILE (L < M) LOOP
    WHILE (T(M) > T(DEBUT)) AND THEN M > DEBUT LOOP
      M := M - 1;
    END LOOP;
    WHILE (T(L) <= T(DEBUT)) AND THEN L < FIN LOOP
      L := L + 1;
    END LOOP;
    IF (L < M) THEN
      Permuter(T,L,M);
      M := M - 1;
      L := L + 1;
    END IF;
  END LOOP;

```

01/09/2006

46

## Tris > Tris récursifs > Tri rapide

```

-- Recherche de la position final du pivot
L:=DEBUT+1;
WHILE L <= FIN AND THEN (T(L) <= T(DEBUT)) LOOP
  L := L + 1;
END LOOP;

Pivot := L-1;
Deplacer(T, DEBUT, Pivot);
END Partition;

```

01/09/2006

47

## Tris > Tris récursifs > Tri rapide > Exemple

3	2	5	1	5	4	3	6
---	---	---	---	---	---	---	---

□ Pivot = 5

5	2	5	1	3	4	3	6
---	---	---	---	---	---	---	---

□ Déplacement des cases

5	2	5	1	3	4	3	6
---	---	---	---	---	---	---	---

□ Remplacement du pivot

2	5	1	3	4	3	5	6
---	---	---	---	---	---	---	---

01/09/2006

48

### Tris > Tris récursifs > Tri rapide > Exemple

2	5	1	3	4	3	5	6
---	---	---	---	---	---	---	---

□ Pivot = 4

3	5	1	2	4	3	5	6
---	---	---	---	---	---	---	---

□ Déplacement des cases

3	3	1	2	4	5	5	6
---	---	---	---	---	---	---	---

□ Remplacement du pivot

3	1	2	3	4	5	5	6
---	---	---	---	---	---	---	---

01/09/2006

49

### Tris > Tris récursifs > Tri rapide > Exemple

3	1	2	3	4	5	5	6
---	---	---	---	---	---	---	---

□ Pivot = 2

1	3	2	3	4	5	5	6
---	---	---	---	---	---	---	---

□ Déplacement des cases

1	3	2	3	4	5	5	6
---	---	---	---	---	---	---	---

□ Remplacement du pivot

1	3	2	3	4	5	5	6
---	---	---	---	---	---	---	---

01/09/2006

50

### Tris > Tris récursifs > Tri rapide > Exemple

1	3	2	3	4	5	5	6
---	---	---	---	---	---	---	---

□ Echange(n=2)

1	2	3	3	4	5	5	6
---	---	---	---	---	---	---	---

□ Tableau trié

1	2	3	3	4	5	5	6
---	---	---	---	---	---	---	---

01/09/2006

51