

UE ALGO5 — TD2 — Séance 3 : Tri par segmentation

Objectifs

À la fin de cette séance, vous devriez être capable de :

- structurer un algorithme en utilisant des sous fonctions ayant une signature et des pré- et post-conditions ;
- spécifier et prouver des invariants de boucle ;
- écrire des algorithmes récursifs comme le tri par segmentation.

Exercice 1. Tri par segmentation

On rappelle le schéma algorithmique du tri par segmentation :

```
1 TriSegmentation(T):
   si la taille de T est > 1 alors
3     Soit  $p$  une valeur «pivot»
     Partitionner T en  $T_1, T_2$  t.q.:
5     –  $T_1$  contient les valeurs de T inférieures ou égales à  $p$ 
     –  $T_2$  contient les valeurs de T supérieures à  $p$ 
7     TriSegmentation( $T_1$ )
     TriSegmentation( $T_2$ )
```

Réalisation

Corrigé —

Avant la première question on peut réfléchir à la signature de TriSegmentation : arriver assez vite à la conclusion que «passer un tableau» en paramètre d'une procédure récursive n'est pas forcément raisonnable (recopies de tableaux). On travaille sur des tranches disjointes : on peut donc implémenter un «tableau» par un tableau (global) + deux indices (indice de début, indice de fin). Il n'y a ainsi pas de copie de tableaux inutiles.

T : un tableau sur $[1..N]$ d'entiers

```
2 TriSegmentation(données p, q : deux entiers)
4 { Trie le sous-tableau  $T[p..q]$ .
   Pré-condition :  $1 \leq p \leq q \leq N$ 
6   Post-condition :  $T[p..q]$  trié }

8 Appel initial : TriSegmentation(1,N)
```

Q 1. Le partitionnement sera réalisé par une procédure intermédiaire nommée Partition. Donner la spécification de cette procédure (signature, pré- et post-conditions).

Corrigé —

L'idée est ici de réfléchir à la spécification sans réaliser tout de suite la procédure.

- Comme pour TriSegmentation, il s'agit de partitionner un sous-tableau : deux données pour l'indice de début et de fin.
- Valeur du pivot ? Est-elle déterminée à l'intérieur de Partition ou connue à l'appel de procédure ? Pas beaucoup d'importance : on suppose ici que c'est à l'intérieur (pas besoin de cette valeur après Partition)
- Comment sont déterminés T_1 et T_2 pour les appels subséquents à TriSegmentation ? \Rightarrow nouvel indice intermédiaire (dernier indice de T_1 , premier de T_2 ... peu importe mais à définir).

Par exemple :

Partition(données p, q : deux entiers; résultat r : un entier)

```
2 { Pré-condition :  $1 \leq p \leq q \leq N$ . Soit  $T_0$  la valeur initiale de T.  
   Post-condition :  $p \leq r \leq q$   
4   Il existe x une valeur de  $T_0[p..q]$  telle que :  
     -  $T[p..r]$  contient toutes les valeurs de  $T_0$  inférieures ou égales à x  
6     -  $T[r+1..q]$  contient toutes les valeurs de  $T_0$  supérieures à x  
     -  $T[r] = x$   
8 }
```

Q 2. Donner une réalisation de TriSegmentation en utilisant Partition.

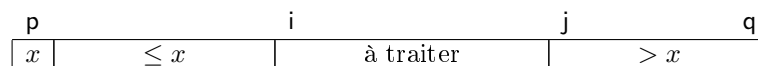
Corrigé —

```
TriSegmentation(p,q):  
2   r : un entier  
   debut  
4   si p < q alors  
     Partition(p,q,r)  
6     TriSegmentation(p,r-1)  
     TriSegmentation(r+1,q)  
8   finsi  
fin
```

Q 3. Donner une réalisation de Partition : commencer par exprimer sous forme de schéma l'invariant de la boucle principale.

Corrigé —

Exemple d'invariant :



Algorithme associé :

```
1 Partition(p,q,r):  
   i,j : deux entiers  
3   x : un entier  
   debut  
5    $x \leftarrow T[p]$   
    $i \leftarrow p + 1$   
7    $j \leftarrow q + 1$   
   tant que i < j  
9     si  $T[i] \leq x$  alors  
        $i \leftarrow i + 1$   
11    sinon  
         $j \leftarrow j - 1$   
13    échanger( $T[i], T[j]$ )  
    fin si  
15  fin tant que  
     $r \leftarrow i - 1$   
17  échanger( $T[p], T[r]$ )  
fin
```

Analyse

Q4. Quel est le coût dans le pire cas, dans le meilleur cas de Partition? de TriSegmentation? Donner des exemples (de taille 10) de pire et meilleurs cas.

Corrigé —

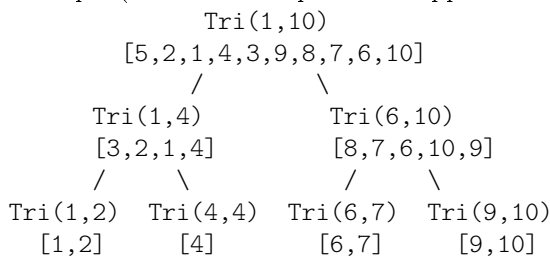
Partition : coût de $O(n)$ dans tous les cas (n =taille du sous-tableau; n passages dans la boucle puisque i incrémenté ou j décrémenté à chaque itération).

TriSegmentation :

—Meilleur cas : à chaque appel récursif, partition en deux sous-tableau de taille identique.

Coût $C(n) = 2.C(n/2) + O(n) = O(n \log n)$

Exemple (à construire à partir des appels récursifs) :



—Pire cas : à chaque appel récursif, partition en deux sous-tableau de taille $(n - 1)$ et 0.

Coût $C(n) = C(n - 1) + O(n) = O(n^2)$

Exemple : tous les éléments du tableau ont la même valeur.

Q5. Donner le coût de Partition(T) dans le cas où tous les éléments de T ont la même valeur. Proposer une correction de Partition susceptible d'améliorer ce coût.

Corrigé —

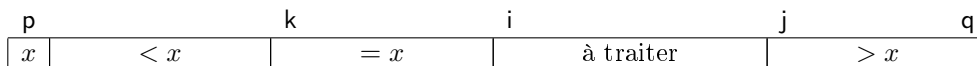
On peut remarquer que la question se pose de manière plus générale pour les tableaux comportant un nombre important de fois la même valeur (ce qui arrive dans de nombreuses applications).

Identifier le problème : la partition effectuée met toutes les valeurs "égales" au pivot d'un seul côté. Il faudrait au contraire minimiser la taille de la plus grande partition.

Plusieurs pistes possibles :

—les valeurs égales au pivot peuvent très bien être placées dans l'une ou l'autre partition : alterner entre les deux (à l'aide d'un booléen par exemple) ?

—plus rigoureux : faire un vrai «drapeau hollandais», en regroupant toutes les valeurs égales au milieu du tableau.



Partition(p,q,r):

2 i,j,k : trois entiers

x : un entier

4 debut

x ← T[p]

6 i ← p + 1

k ← p + 1

8 j ← q + 1

tant que i < j

10 **si** T[i] = x **alors**

i ← i + 1

12 **sinon si** T[i] < x **alors**

échanger(T[i],T[k])

14 i ← i + 1

```

        k ← k + 1
16     sinon
        j ← j - 1
18     échanger(T[i],T[j])
        fin si
20     fin tant que
    échanger(T[p],T[k-1])
22     r ← max(k-1,min((p+q)/2,i-1))
fin

```

On peut ensuite remarquer que le sous-tableau contenant les valeurs égales au pivot est trié : on peut changer la spécification de Partition pour donner comme résultat les trois partitions (on ajoute un résultat s, T[r..s] est le sous-tableau contenant les valeurs égales au pivot). L'algorithme de tri devient :

```

1  TriSegmentation(p,q):
    r : un entier
3  debut
    si p < q alors
5      Partition(p,q,r,s)
      TriSegmentation(p,r-1)
7      TriSegmentation(s+1,q)
    fin si
9  fin

```
