Université Paris 7 - Denis Diderot
L1 Sciences

IF2 : Structures de données et Objets JAVA
Année 2008-2009, 2ème semestre

TD n°6

Diviser pour régner, l'exemple du Quick Sort

Exercice 1 Segmentation
On considère un tableau d'entiers. Après avoir choisi un élément quelconque de ce tableau que l'on nommera *pivot*, on veut regrouper au début tous les éléments inférieurs au pivot et à la fin tous les éléments strictement supérieurs au pivot avec une complexité proportionnelle à la taille du tableau.

1. Essayer d'effectuer ce rangement "à la main" sur le tableau suivant avec le pivot 5 :

5	10	1	3	7	4	2	6	9	8

   Chercher une méthode avec une complexité proportionnelle à la taille du tableau.
2. En déduire un invariant pour le problème.
3. Ecrire la methode `partitionner(tab:tableau d'entiers; d,f,indPivot : entier)` où `d` est le premier indice du tableau, `f` le dernier indice et `indPivot` l'indice du pivot.
4. Cet algorithme vérifie-t-il l'invariant choisi précédemment ?
5. Prouver sa finitude.

Exercice 2 QuickSort
On considère un tableau d'entiers. On veut trier de ce tableau en utilisant la méthode de segmentation+ de l'exercice précédent grâce à une tactique "diviser pour régner" :
- On choisit un pivot.
- On segmente le tableau par rapport à ce pivot.
- On recommence sur les deux sous-tableaux séparés par le pivot.

1. Appliquer cette méthode de tri au tableau suivant en choisissant toujours comme pivot le premier élément du segment à trier.

5	10	1	3	7	4	2	6	9	8

2. En quoi s'agit-il d'une méthode de type "diviser pour régner" ?
3. Est-ce que le pivot segmente toujours le tableau de manière satisfaisante ? Quel est le meilleur des cas et le pire des cas ? Y-a-t-il une bonne manière de choisir le pivot pour être toujours proche du meilleur des cas ?
4. faites proposition d'algorithme : Comme on a besoin de connaître l'indice où l'on peut déplacer le pivot après la segmentation, il faut modifier la méthode écrite dans l'exercice précédent afin de retourner celui-ci en fin de fonction. Modifiez la méthode `partitionner` écrite dans l'exercice 1 et faites une proposition pour la fonction `procedure tri_rapide(tab:tableau d'entiers; d,f : entier)`.
5. Pourquoi l'instruction `echanger tab[d] et tab[p]` conserve-t-elle la segmentation ?
6. Pourquoi a-t-on ajouté cette instruction ? Prouver la finitude de l'algorithme. En quoi l'instruction `echanger tab[d] et tab[p]` est-elle essentielle pour la finitude ?

Correction

```
public class Quicksort{
  public static int segmenter(int [] tab, int d, int f, int indPivot){
    int i=d,j=f, pivot=tab[indPivot];
    while(i<=j) {
      while (i<=f && tab[i]<=pivot)  i++;
      while (j>=d && tab[j]>pivot)  j--;
      if (i<j) {
        echanger(tab,i,j);
        i++;
        j--;
      }
    }
    /*a la fin on a j=i+1
    * NB : on peut remplacer while(i<=j) par while(i<j) si on n'actualise pas i et j
    * apres un echange
    */
    return j;
  }


  public static void echanger(int[]tab, int j, int k){
    int temp = tab[j];
    tab[j]=tab[k];
    tab[k]=temp;
  }

  public static void quicksort(int[] tab, int d, int f){
    int p;
    if(d<tab.length){
      if(d<f){
        System.out.println("Partition de: "+d+" a fin: "+f+";
                                par le pivot "+d+" : "+tab[d]);
        p=segmenter(tab,d,f,d);
        System.out.println(toStringP(tab));
        System.out.println("fin partie 1 : "+p);
        echanger(tab,d,p);
        System.out.println("Echange des cases "+d+" et "+p);
        System.out.println(toStringP(tab)+"\n");
        quicksort(tab,d,p-1);
        quicksort(tab,p+1,f);
      }
    }
  }

  public static void quicksort(int[] tab){
    quicksort(tab,0,tab.length-1);
  }
```

```
  public static String toStringP(int[]tab){
    int n = tab.length;
    String s ="[ ";
    for(int i=0; i<n; i++){
      s+=tab[i]+" ";
    }
    return s+"]";
  }

  public static void main(String[] args){
    int[] tab = {5,10,1,3,7,4,2,6,9,8};
    //int[] tab1 = {115,10,11,113,7,4,12,62,92,82};
    //int[] tab2 = {7,11,10,4};
    System.out.println("Tableau a trier\n"+toStringP(tab));
    quicksort(tab);
    System.out.println("Tableau trie\n"+toStringP(tab));
  }
}
```