Le tri rapide : conception, algorithme, correction et complexité

1 Principe du tri rapide

- Je partitionne le tableau en fonction d'un *pivot* (ici, le premier élément du tableau).
- Je place le pivot à la bonne place.
- Je trie récursivement sur chacun des tableaux à sa gauche et à sa droite.

2 Analyse - Algorithme

On écrit une action **récursive** qui prend en paramètre le tableau à trier ainsi que deux indices qui disent entre quels indices trier :

- Si les indices satisfont $premier = dernier - 1$, cela veut dire que l'on est en train de trier un tableau de taille 2, et on sait faire !
- Sinon, on fait les trois étapes suivantes :
  - on appelle la fonction `partition` qui est supposé placer l'élément d'indice `premier` à la bonne place, et qui retourne l'indice de placement (`ipivot`) ;
  - on trie récursivement à gauche de cet indice ;
  - on trie récursivement à droite.

Ce qui est réalisé par le pseudo-code ci-dessous :

```
Action tri-rapide-bis(T,premier,dernier)
|  D: premier,dernier : Entiers
|  D: T : Tableau[N] d'entiers
|  L: ipivot : Entier                                  {indice du pivot}
|  Si premier=dernier−1 alors
|  |  Si T[premier]>T[dernier] alors
|  |  |  Permuter(T,premier,dernier)
|  |  Fsi
|  Sinon
|  |  Si premier<dernier alors
|  |  |  ipivot←partition(T,premier,dernier)
|  |  |  tri-rapide-bis(T,premier,ipivot-1)
|  |  |  tri-rapide-bis(T,ipivot+1,dernier)
|  |  Fsi
|  Fsi
FAction
```

Du coup, l'algorithme qu'on nous demande est défini par :

- `tri-rapide(T)` : ne rien faire si la taille du tableau est 0 ou 1.
- `tri-rapide(T)=tri-rapide-bis(T,0,N-1)` sinon

L'écriture du pseudo-code est laissé au lecteur.

3 Partionnement

Il reste à écrire *partition*, qui est en fait la partie la plus difficile car il faut un peu se battre avec les indices [1] :

```
Fonction partitionne(T,premier,dernier)
    D: premier,dernier : Entiers
    D: T : Tableau[N] d'entiers
    L: cpt,pivot,i : Entiers
    i← premier+1 ;j← dernier ; pivot=t[premier];
    Tq (i<j) faire
        Si (t[i]<=pivot) alors
            i++ ;
        Sinon
            Si t[j]>=pivot alors
                j−
            Sinon
                Permuter(t,i,j) ;i++ ;j− ;
            Fsi
        Fsi
    Ftq
    Si (i==j) alors
        Si (t[i]<=pivot) alors
            j++ ;
        Sinon
            i− ;
        Fsi
    Sinon
        i− ;j++ ;
    Fsi
    Echanger(t,premier,i);
    Retourner i
FFonction
```

4 Correction

Pour prouver la correction, il faut :
- Prouver que `partition` est correct
- Ensuite, montrer que "l'appel à tri-rapide sur un tableau de taille i trie le tableau"

Pour trouver le coût, on prend tout d'abord un tableau de taille $N = 2^p$, on obtient une relation de récurrence qui se résoud en $O(N \ln_2 N)$ en moyenne.

Par contre, dans le pire des cas, on obtient $O(N^2)$ (si le tableau est trié !)

1. donc, faire des dessins !